QUT Meritorious Project 1995

An Evaluation And Comparison Of Techniques For Extracting And Refining Rules From Artificial Neural Networks

Robert Andrews
Russell Cable
Joachim Diederich
Shlomo Geva
Mostefa Golea
Ross Hayward
Chris Ho-Stuart
Alan B. Tickle

robert@fit.qut.edu.au
cable@fit.qut.edu.au
joachim@fit.qut.edu.au
geva@fit.qut.edu.au
golea@fit.qut.edu.au
hayward@fit.qut.edu.au
hostuart@fit.qut.edu.au
tickle@fit.qut.edu.au

Neurocomputing Research Centre
Queensland University of Technology
Box 2434 GPO Brisbane 4001
Queensland, Australia

Version: 9:18am 18 March 1996

An abridged version of this report appears in the Knowledge-Based Systems Journal Vol 8, No 6 (December 1995)
ABSTRACT
It is becoming increasingly apparent that without some form of explanation capability, the full potential of trained Artificial Neural Networks (ANNs) may not be realised. The primary purpose of this report is to survey techniques which have been developed to redress this situation. Specifically the survey focuses on mechanisms, procedures, and algorithms designed to insert knowledge into ANNs (knowledge initialisation), extract rules from trained ANNs (rule extraction), and utilise ANNs to refine existing rule bases (rule refinement). The survey also introduces a new taxonomy for classifying the various techniques, discusses their modus operandi, and delineates criteria for evaluating their efficacy. An additional facet of the report is a comparative evaluation of the performance of a set of techniques developed at the NeuroComputing Research Centre at the QUT to extract knowledge from trained ANNs as a set of symbolic rules.

Keywords. rule extraction from neural networks, rule refinement using neural networks, knowledge insertion into neural networks, fuzzy neural networks, inferencing, rule generation

INTRODUCTION
The rapid and successful proliferation of applications incorporating Artificial Neural Network (ANN) technology in fields as diverse as commerce, science, industry and medicine, offers a clear testament to the capability of the ANN paradigm. Of the three salient characteristics of ANNs which underpin this success, the first is the comparatively direct and straightforward manner in which Artificial Neural Networks acquire information/‘knowledge’ about a given problem domain through a training phase. This process is quite distinct from the more complicated knowledge engineering/acquisition processes of symbolic AI systems. The second characteristic is the compact (albeit completely numerical) form in which the acquired information/‘knowledge’ is stored within the trained ANN and the comparative ease and speed with which this ‘knowledge’ can be accessed and used. The third characteristic is the robustness of an ANN solution in the presence of ‘noise’ in the input data. In addition to these characteristics, one of the most important advantages of trained Artificial Neural Networks is the high degree of accuracy reported when an ANN solution is used to generalise over a set of previously unseen examples from the problem domain [38].

However the success of the ANN paradigm is at a cost - an inherent inability to explain in a comprehensible form, the process by which a given decision or output generated by an ANN has been reached. For Artificial Neural Networks to gain a even wider degree of user acceptance and to enhance their overall utility as learning and generalisation tools, it is highly desirable if not essential that an ‘explanation’ capability becomes an integral part of the functionality of a trained ANN. Such a requirement is
mandatory if, for example, the ANN is to be used in what are termed as ‘safety critical’ applications such as airlines and power stations. In these cases it is imperative that a system user be able to validate the output of the Artificial Neural Network under all possible input conditions. Further the system user should be provided with the capability to determine the set of conditions under which an output unit within an ANN is active and when it is not, thereby providing some degree of transparency of the ANN solution.

Apart from the direct contribution to enhancing the overall utility of Artificial Neural Networks, the addition of an ‘explanation’ capability is also seen as having the potential to contribute to the understanding of how symbolic and connectionist approaches to Artificial Intelligence (AI) can be profitably integrated. It also provides a vehicle for traversing the boundary between the connectionist and symbolic approaches.

This report provides a survey and critique of the more significant techniques developed to date to extract rules from trained Artificial Neural Networks and hence provide the requisite explanation capability. An additional facet of the report is a comparative evaluation of the performance of a set of techniques developed at the NeuroComputing Research Centre at the QUT.

For the purposes of this discussion, the focus will be restricted to rule extraction from general feed-forward multi-layered Artificial Neural Network architectures utilising a ‘supervised’ learning regime such as back-propagation. Hence techniques for rule extraction from ‘specialised’ Artificial Neural Network learning architectures such as KBANN developed by Towell and Shavlik [40] and Cascade developed by Fahlman and Lebiere [8], are included in the review whereas techniques for rule extraction from architectures based on unsupervised or reinforcement learning are excluded.

At this stage it is also worth commenting on two related areas of development in rule extraction techniques which do not form part of this initial survey, although both are important in their own right. The first is the extraction of symbolic grammatical rules from trained Artificial Neural Networks and, in particular, recurrent neural networks [16, 17, 42]. In this area of research the focus is on having an ANN simulate a deterministic Finite State Automata (FSA) which recognises regular grammars. The second is the work of Gallant [13] on extracting explanations of individual problem instances from trained Artificial Neural Networks as distinct from complete sets of rules.

THE IMPORTANCE OF RULE-EXTRACTION ALGORITHMS

Since rule extraction from trained Artificial Neural Networks comes at a cost in terms of resources and additional effort, an early imperative in any discussion is to delineate the reasons why rule extraction is an important, if not mandatory, extension of conventional ANN techniques. The merits in
including rule extraction techniques as an adjunct to conventional Artificial Neural Network techniques include:

**Provision of a `user explanation` capability**

Within the field of symbolic AI the term `explanation` refers to an explicit structure which can be used internally for reasoning and learning, and externally for the explanation of results to a user. Users of symbolic AI systems benefit from an explicit declarative representation of knowledge about the problem domain, typically in the form of object hierarchies, semantic networks, frames etc. The explanation capability of symbolic AI also includes the intermediate steps of the reasoning process eg a trace of rule firings, a proof structure etc., which can be used to answer `How` questions. Further, Gallant [13] observes that the attendant benefits of an explanation capability are that it also provides a check on the internal logic of the system as well as enabling a novice user to gain insights into the problem at hand.

Experience has shown that an explanation capability is considered to be one of the most important functions provided by symbolic AI systems. In particular, the salutary lesson from the introduction and operation of Knowledge Based systems is that the ability to generate even limited explanations (in terms of being meaningful and coherent) is absolutely crucial for the user-acceptance of such systems [6]. In contrast to symbolic AI systems, Artificial Neural Networks have no explicit declarative knowledge representation. Therefore they have considerable difficulty in generating the required explanation structures. It is becoming increasingly apparent that the absence of an `explanation` capability in ANN systems limits the realisation of the full potential of such systems and it is this precise deficiency that the rule extraction process seeks to redress.

While provision of an explanation capability is a significant innovation in the ongoing development of Artificial Neural Networks, of equal importance is the `quality` of the explanations delivered. It is here that the evolution of explanation capabilities in symbolic AI offers some valuable lessons into how this task of extracting rules from trained Artificial Neural Networks might be directed. For example practitioners in the field of symbolic AI have experimented with various forms of user explanation vehicles including in particular, rule traces. However for some time it has been clear that explanations based on rule traces are too rigid and inflexible [14, 26, 27]. Indeed one of the major criticisms of utilising rule traces is that they always reflect the current structure of the knowledge base. Further, rule traces may have references to internal procedures (eg. calculations); might include repetitions (eg. if an inference was made more than once); and the granularity of the explanation is often inappropriate [14]. Perhaps one clear lesson form using rule traces is that the transparency of an explanation is by no means guaranteed. For example experience has shown that an explanation based on a rule traces from a poorly organised rule-base with
perhaps hundreds of premises per rule could not be regarded as being 'transparent'.

A further example of the limitations of explanation capabilities in symbolic AI systems which should, if possible, be obviated in the extraction of rules from trained Artificial Neural Networks, comes from Moore and Swartout [27]. They note that the early use of 'canned' text or templates as part of user explanations has been shown to be too rigid, that systems always interpret questions in the same way, and that the response strategies are inadequate. Further, although efforts have been made to take advantage of natural-language dialogues with artifices such as mixed initiatives, user-models and explicitly planned explanation strategies [27], there is little doubt that current systems are still too inflexible, unresponsive, incoherent, insensitive and too rigid [36].

In summary while the integration of an explanation capability (via rule extraction) within a trained Artificial Neural Network is crucial for user acceptance, such capabilities must if possible obviate the problems already encountered in symbolic AI.

**Extension of ANN systems to `safety-critical` problem domains**

While the provision of a 'user explanation' capability is one of the key benefits in extracting rules from trained ANNs, it is certainly not the only one. For example within a trained Artificial Neural Network the capability should also exist for the user to determine whether or not the ANN has an optimal structure or size. A concomitant requirement is for ANN solutions to not only be transparent as discussed previously but also for the internal states of the system to be both accessible and able to be interpreted unambiguously. Satisfaction of such requirements would make a significant contribution to the task of identifying and if possible excluding those ANN-based solutions that have the potential to give erroneous results without any accompanying indication as to when and why a result is sub-optimal.

Such a capability is mandatory if neural-network based solutions are to be accepted into a broader range of applications areas and in particular, 'safety-critical' problem domains such as air traffic control, the operation of power plants, medical surgery etc. Rule-extraction offers the potential for providing such a capability.

**Software verification and debugging of ANN components in software systems**

A requirement of increasing significance in software-based systems is that of verification of the software itself. While the task of software verification is important it is also acknowledged as being difficult, particularly for large systems. Hence if Artificial Neural Networks are to be integrated within larger software systems which need to be verified, then clearly this requirement must be met by the ANN as well. At their current level of
development, rule-extraction algorithms do not allow for the verification of trained Artificial Neural Network ie they do not prove that a network behaves according to some specification. However rule extraction algorithms provide a mechanism for either partially or completely \textit{decompiling} a trained Artificial Neural Network. This is seen as a promising vehicle for at least indirectly achieving the required goal by enabling a comparison to be made between the extracted rules and the software specification.

**Improving the generalisation of ANN solutions**

Where a limited or unrepresentative data set from the problem domain has been used in the ANN training process, it is difficult to determine when generalisation can fail even with evaluation methods such as cross-validation. By being able to express the knowledge embedded within the trained Artificial Neural Network as a set of symbolic rules, the rule-extraction process may provide an experienced system user with the capability to anticipate or predict a set of circumstances under which generalisation failure can occur. Alternatively the system user may be able to use the extracted rules to identify regions in input space which are not represented sufficiently in the existing ANN training set data and to supplement the data set accordingly.

**Data exploration and the induction of scientific theories**

Over time neural networks have proven to be extremely powerful tools for data exploration with the capability to discover previously unknown dependencies and relationships in data sets. As Craven and Shavlik [5] observe, \textit{`a (learning) system may discover salient features in the input data whose importance was not previously recognised.'} However, even if a trained Artificial Neural Network has learned interesting and possibly non-linear relationships, these relationships are encoded incomprehensibly as weight vectors within the trained ANN and hence cannot easily serve the generation of scientific theories. Rule-extraction algorithms significantly enhance the capabilities of ANNs to explore data to the benefit of the user.

**Knowledge acquisition for symbolic AI systems**

One of the principal reasons for introducing machine learning algorithms over the last decade was to overcome the so-called \textit{`knowledge acquisition'} problem for symbolic AI systems [31, 32]. Further, as Sestito and Dillon [34, p156] observe, the most difficult, time-consuming, and expensive task in building an expert system is constructing and debugging its knowledge base.

The notion of using trained Artificial Neural Networks to assist in the knowledge acquisition task has existed for some time [13]. An extension of these ideas is to use trained Artificial Neural Networks as a vehicle for synthesising the knowledge that is crucial for the success of knowledge-based systems. Alternatively domain knowledge which is
acquired by a knowledge engineering process may be used to constrain the size of the space searched during the learning phase and hence contribute to improved learning performance.

The necessary impetus for exploring these ideas further could now come from two recent developments. The first is a set of recent benchmark results such as those of Thrun et al [37] where trained Artificial Neural Networks have been shown to outperform symbolic machine learning methods. The second is from developments in techniques for extracting symbolic rules from trained Artificial Neural Networks which could be directly added to the knowledge base.

PROBLEM OVERVIEW

Having identified the importance of rule extraction to the continued development and success of the ANN paradigm, the next step is to provide a succinct expression of the problem to be addressed. A useful starting point is a basic recognition that within a trained artificial Neural Network, knowledge acquired during the training phase is encoded as: (a) the network architecture itself (e.g., the number of hidden units); (b) an activation function associated with each (hidden and output) unit of the ANN; and (c) a set of (real-valued) numerical parameters (called weights).

In essence, the task of extracting explanations (or rules) from a trained Artificial Neural Network is therefore one of interpreting in a comprehensible form the collective effect of (a), (b), and (c).

An ancillary problem to that of rule extraction from trained ANNs is that of using the ANN for the "refinement" of existing rules within symbolic knowledge bases. Whereas the rule extraction process normally commences with an empty symbolic rule base, the starting point for the rule-refinement process is some initial knowledge about the problem domain expressible in the form of symbolic rules. A crucial point however is that the initial set of rules may not necessarily be complete or even correct [15]. Irrespective of the quality of the initial rule base, the goal in rule refinement is to use a combination of ANN learning and rule extraction techniques to produce a "better" (i.e., a "refined") set of symbolic rules which can then be applied back in the original problem domain. In the rule refinement process, the initial rule base (i.e., what may be termed "prior knowledge") is inserted into an ANN by programming some of the weights. (In this context, "prior knowledge" refers to all of the production rules known prior to commencement of the ANN training phase.) The rule refinement process then proceeds in the same way as normal rule extraction viz (1) train the network on the available data set(s); and (2) extract (in this case the "refined") rules - with the proviso that the rule refinement process may involve a number of iterations of the training phase rather than a single pass.
A CLASSIFICATION SCHEME FOR RULE EXTRACTION ALGORITHMS

Over time a number of different strategies have been developed to achieve the desired goals of both extracting rules from trained Artificial Neural Networks and the use of ANNs to refine an existing rule set. However apart from a rudimentary classification scheme used by Craven and Shavlik [5], to date little has appeared in the form of a rigorous and systematic attempt to develop a coherent and consistent taxonomy for classifying these approaches. Given the proliferation of rule extraction/ rule refinement techniques, the need for such a schema is now well established. The method of classification proposed here is in terms of: (a) the expressive power of the extracted rules; (b) the 'translucency' of the view taken within the rule extraction technique of the underlying Artificial Neural Network units; (c) the extent to which the underlying ANN incorporates specialised training regimes; (d) the 'quality' of the extracted rules; and (e) the algorithmic 'complexity' of the rule extraction/rule refinement technique.

The primary dimension in our proposed classification scheme labelled as 'the expressive power of the extracted rules' focuses directly on the actual output presented to the end user from the rule extraction/rule refinement process viz the rules themselves. To date one line of effort in the development of rule extraction techniques has been directed towards presenting the output as a set of rules expressed using conventional (ie two-valued Boolean) symbolic logic in the form if...then...else.... A substantial parallel effort has also been directed towards expressing the knowledge embodied in the ANN using concepts drawn from 'fuzzy' logic. This allows rules to be expressed in a form which also use an if...then...else.... structure but in lieu of two-valued logic, such rules use the concept of membership functions to deal with what are termed 'partial' truths eg if x is low and y is high then z is medium; where low, high, and medium are fuzzy sets with corresponding membership functions.

Hence for the purposes of this exercise the two primary categories selected to represent this dimension are that the rules extracted be in the form of: (a) propositional/Boolean (ie if...then...else) or (b) non-conventional (ie the resulting representation can include probabilistic or 'fuzzy logic' rules which may also incorporate an if...then...else structure). While not deemed necessary at this stage, at some future point it may expedient to introduce a third category to cater for extracted rules represented in first-order logic form ie rules with quantifiers and variables.

The second proposed dimension to the classification scheme extends the classification schema used by Craven and Shavlik [5]. The 'translucency' dimension of classification is designed to reveal the relationship between the extracted rules and the internal architecture of the trained ANN. It comprises two basic categories of rule extraction techniques viz 'decompositional' and 'pedagogical' and a third - labelled as 'eclectic' - which combines elements of the two basic categories.
The distinguishing characteristic of the `decompositional' approach is that the focus is on extracting rules at the level of individual (hidden and output) units within the trained Artificial Neural Network. Hence the `view' of the underlying trained Artificial Neural Network is one of `transparency'. A basic requirement for rule extraction techniques in this category is that the computed output from each hidden and output unit in the trained Artificial Neural Network must be mapped into a binary (yes/no) outcome which corresponds to the notion of a rule consequent. Hence each hidden or output unit can be interpreted as a `step' function or a Boolean rule which reduces the rule extraction problem to one of determining the situations in which the `rule' is true ie a set of incoming links whose summed weights guarantee the unit's bias is exceeded regardless of the activation value present on other incoming links. The rules extracted at the individual unit level are then aggregated to form the composite rule base for the ANN as a whole.

The translucency dimension - `pedagogical' is given to those rule extraction techniques which treat the trained ANN as a `black box' ie the view of the underlying trained Artificial Neural Network is `opaque'. The core idea in the `pedagogical' approach is to `view rule extraction as a learning task where the target concept is the function computed by the network and the input features are simply the network's input features' [5]. Hence the `pedagogical' techniques aim to extract rules that map inputs directly into outputs. Where such techniques are used in conjunction with a symbolic learning algorithm, the basic motif is to use the trained Artificial Neural Network to generate examples for the learning algorithm.

As indicated above the proposed third category in this classification scheme are composites which incorporate elements of both the `decompositional' and `pedagogical' (or `black-box') rule extraction techniques. This is the `eclectic' group. Membership in this category is assigned to techniques which utilise knowledge about the internal architecture and/or weight vectors in the trained Artificial Neural Network to complement a symbolic learning algorithm.

The next classification dimension used is the extent of the requirement within a given rule extraction technique for a specialised ANN training regime. This is an important consideration because it provides some measure of the `portability' of the rule extraction technique across various ANN architectures. An additional facet of this classification dimension is whether the underlying ANN is modified or left intact by the rule extraction process.

The fourth dimension in our proposed classification scheme is the `quality' of the extracted rules. The specific intent here is to attempt some measure of how well the task of extracting the required `explanation' has been performed [15, 40]. Criteria for evaluating rule `rule-quality' include: (a)
accuracy; (b) fidelity; (c) consistency; and (d) comprehensibility.

In this context a rule set is considered to be accurate if it can correctly classify a set of previously unseen examples from the problem domain. Similarly a rule set is considered to display a high level of fidelity if it can mimic the behaviour of the Artificial Neural Network from which it was extracted by capturing all of the information embodied in the ANN. Under Towell and Shavlik's criteria, an extracted rule set is deemed to be consistent if, under differing training sessions, the Artificial Neural Network generates rule sets which produce the same classifications of unseen examples. Finally the comprehensibility of a rule set is determined by measuring the size of the rule set (in terms of the number of rules) and the number of antecedents per rule.

From the work of Giles et al. [15] additional rule-quality criteria which are pertinent to the ancillary problem of rule refinement are that the overall process must preserve genuine knowledge/rules, and correct wrong prior information/rules. Based on the work of Weiss et al. [43] one could similarly extend the criteria for evaluating rule quality to include some measure of model complexity (e.g. the number of hidden units for single hidden-layer back-propagation Artificial Neural Networks).

Clearly an assessment of the quality of the rules produced by a given rule-extraction/rule-refinement technique is potentially of significant value to a prospective user. Unfortunately, as will be revealed in the ensuing discussion on the evolution of rule extraction/rule refinement techniques, rigorous assessment of the quality of the rules produced by a given technique is only a relatively recent phenomenon. This limits the scope for comparison of the various techniques using the criterion of rule quality. In passing it should also be observed that in many 'real world' problem domains the simultaneous optimisation of all multiple evaluation criteria may not always be desirable. For example in 'safety critical' applications it is imperative that the Artificial Neural Network be validated under all possible input conditions. This may create a situation for example where rule comprehensibility is sacrificed for rule accuracy and fidelity. One final comment on the issue of rule quality in general and rule comprehensibility in particular is that the focus of discussion is exclusively on rule syntax and not on the more problematic area of rule semantics.

The final dimension in the proposed classification scheme is that of the algorithmic 'complexity' of a given rule extraction procedure. The inclusion of such a dimension reflects an almost universal requirement for the algorithms underpinning the rule extraction process to be as efficient as possible. In particular a crucial issue in developing a rule extraction process/algorithm is how to constrain the size of the solution space to be searched. As in the case for using rule quality as a classification dimension, the difficulty is that not all authors have reported on the issue, and the
discussion by those who have is occasionally superficial.

In summary then, of the five(5) proposed dimensions in the classification scheme, the first three \textit{viz} (a) the expressive power of the extracted rules; (b) the \textit{`translucency'} of the view taken within the rule extraction technique of the underlying Artificial Neural Network units; and (c) the extent to which the underlying ANN incorporates specialised training regimes; appear to be the most reliable given the scope of the information available. In particular for the purpose of the ensuing discussion is proposed to use the first two dimensions of the aforementioned taxonomy (\textit{viz `the expressive power of the extracted rules' and `the translucency of the underlying ANN architecture'}) as the primary classifiers to illustrate the diversity of approaches adopted for rule extraction/rule refinement. Comments are also included on the remaining classification dimensions. Wherever possible it is the intention to also identify how the various ideas have extended previous work. A detailed critique of what are considered to be some of the more prominent rule extraction techniques is given in Appendix 1.

\textbf{RULE EXTRACTION TECHNIQUES}

\textbf{Boolean rule extraction using decompositional approaches}

Artificial Neural Networks utilising standard \textit{back-propagation} as the training regime have been successfully applied to problem domains involving learning and generalisation. Hence there has existed from the outset a strong impetus to develop and apply rule extraction techniques to such ANNs. Some of the earliest work in this area adopted what has been termed in the previous section as the \textit{`decompositional'} approach \textit{ie} focussing on searching for and extracting conventional Boolean rules at the level of the individual (hidden and output) units within the trained ANN [4, 9]. Of particular interest are two approaches [9, 40] in which the basic motif is to search initially for sets of weights containing a single link/connection of sufficient (positive) value to guarantee that the bias on the unit being analysed is exceeded irrespective of the values on the other links/connections. If a link is found which satisfies the criterion, it is written as a rule. The search then proceeds to subsets of two elements \textit{et seq} and the rules extracted at the individual unit level are then aggregated to form the composite rule base for the ANN as a whole. A schematic of the basic algorithm as reported by Towell and Shavlik [40] is given in Table 1.

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
\textbf{Table 1: The \textit{SUBSET} algorithm} \\
\hline
\end{tabular}
\end{table}
for each hidden and output unit:

extract up to $S_p$ subsets of the positively-weighted incoming links for
which the summed weight is greater than the bias on the unit;

for each element $p$ of the $S_p$ subsets:

search for a set $S_N$ of a set of negative-attributes so that the
summed weights of $p$ plus the summed weights of $N - n$
(where $N$ is the set of all negative-attributes and $n$ is an
element of $S_N$) exceed the threshold on the unit;

with each element $n$ of the $S_N$ set, form a rule: 'if $p$ and
NOT $n$, then the concept designated by the unit'.

The earliest implementation of this style of algorithm was the $KT$ algorithm
developed by Fu [9, 12] (see Appendix 1). In this implementation the
problem of mapping the output from each (hidden and output) unit into a
Boolean function was achieved by the simple artifice viz if $0 \leq output \leq\)$
$threshold_1 \Rightarrow no; if threshold_2 \leq output \leq 1 \Rightarrow yes; where threshold_1 <$
$threshold_2$. A more recent example of this line of approach is the $Subset$
algorithm developed by Towell and Shavlik [40]. In their implementation, the
Artificial Neural Network is constructed in a way such that the computed
value of the activation function in each hidden and output unit is either
`near` a value of one (ie `maximally` active) or `near` a value of zero (ie
`inactive`). Hence `links carry a signal equal to their weight or no signal at
all` [40].

Fu [9, 12] reported initial success in applying the $KT$ algorithm to the
problem domain of wind shear detection by infrared sensors. Similarly
Towell and Shavlik [40] showed that their $Subset$ implementation is capable
of delivering a set of rules which are, at least `potentially' tractable and
`smaller than many handcrafted expert systems' [ibid]. However a major
concern with both the $KT$ and $Subset$ algorithms as reported by Towell and
Shavlik is that the solution time for finding all possible subsets is a function
of the size of the power set of the links to each unit [40] ie the algorithm is
exponential. One option used by Fu for restricting the size of the solution
search space is to place a ceiling on the number of antecedents per
extracted rule [9]. Unfortunately this potentially has adverse implications for
rule quality since some rules may be omitted. Notwithstanding their
limitations, the inherent simplicity of this class of algorithms still makes
them extremely useful devices for explaining the mechanics of rule
extraction. It also offers the capability to provide transparency of the trained
ANN solution at the level of individual hidden and output units.

The $RuleNet$ technique/The Connectionist Scientist Game of McMillan et.
al. [24] is one of the earliest examples where a specialised ANN training
regime incorporating the decompositional approach is used as the basis for
extracting Boolean rules (see Appendix 1). The basic modus operandi in RuleNet is to iterate through the following steps: (a) train an ANN; (b) extract the symbolic rules (using the connection strengths in the network); and (c) inject the extracted rules back into the network. The process terminates when the resulting base of extracted rules adequately characterises the problem domain. The specialised ANN training regime is based on the work of Jacobs et al. [21] and incorporates an input layer, an output layer, and an intermediate layer of what are termed ‘condition units' with one condition unit per rule. The major criticism of this approach is that the specialised ANN architecture/training regime is tailored for a specific problem domain and therefore the approach lacks generality [34, 40].

An important development in the utilisation of specialised ANN architectures, was the publication of the M-of-N algorithm which is one component of the total KBANN package utilised by Towell and Shavlik [40] (see Appendix 1). The M-of-N concept is a means of expressing rules in the form:

\[
\text{if (M of the following N antecedents are true) then ...}
\]

Towell and Shavlik cite as one of the main attractions of the M-of-N approach a natural affinity between M-of-N rules and the `inductive bias' of Artificial Neural Networks. The phases of M-of-N algorithm are shown in Table 2.

Table 2: The M-of-N technique

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>generate an Artificial Neural Network using the KBANN system and train using back-propagation. With each hidden and output unit, form groups of similarly-weighted links;</td>
</tr>
<tr>
<td>2.</td>
<td>set link weights of all group members to the average of the group;</td>
</tr>
<tr>
<td>3.</td>
<td>eliminate any groups which do not significantly affect whether the unit will be active or inactive;</td>
</tr>
<tr>
<td>4.</td>
<td>holding all link weights constant, optimise biases of all hidden and output units using the back propagation algorithm;</td>
</tr>
<tr>
<td>5.</td>
<td>form a single rule for each hidden an output unit; the rule consists of a threshold given by the bias and weighted antecedents specified by the remaining links;</td>
</tr>
<tr>
<td>6.</td>
<td>where possible, simplify rules to eliminate superfluous weights and thresholds.</td>
</tr>
</tbody>
</table>

The authors undertook a detailed comparison of their technique with both the Subset algorithm and symbolic induction techniques. They reported significant improvements particularly in terms of the key performance criteria of the ability of the extracted rule set too generalise to examples not seen during training and rule quality. They also noted the `occasional superiority of M-of-N's rules to the networks from which they were extracted' [ibid]. This latter characteristic was attributed to a `reduced overfitting of the
training examples’. From a subsequent critique of Towell and Shavlik’s work by Craven and Shavlik [5] (in the context of discussing the relative advantages of their rule extraction technique based on a ‘pedagogical’ approach) it is possible to bring into focus some of the core requirements of the KBANN/M-of-N approach viz: (a) the requirement for either a ‘rule set’ to initialise the ANN [40] or a special training algorithm that uses a ‘soft-weight sharing’ algorithm to cluster weights; (b) the requirement for a special network training regime; (c) the requirement for hidden units to be approximated as threshold units (this is achieved by setting the parameter $s$ in the activation function $1/[1 + e^{-sx}]$ to be greater than a value of 5.0); and (d) the requirement that the extracted rules use an intermediate term to represent each hidden unit. This gives rise to the concern that the approach may not enable a sufficiently accurate description of the network to be extracted [ibid]. It is also worth noting that one of the basic tenets of the M-of-N approach is that the meaning of a hidden unit in the Artificial Neural Network generated as part of the initialisation process, does not change during the training process. Given that M-of-N is essentially a rule refinement system this may be true in general and, in fact, Towell and Shavlik [40] report empirical confirmation from trained ANNs in their study. However, in the case where the meaning of a unit does change during training, the comprehensibility of the extracted rules may be significantly degraded.

The M-of-N approach has been tested successfully on a diverse range of problem domains including two from the field of molecular biology viz the promoter recognition problem and the splice-junction determination problem. Towell and Shavlik [40] also undertook a detailed comparison of the quality of the rules extracted using their technique with other ANN rule extraction techniques as well as symbolic learning techniques.

An important milestone in the evolution of techniques to initialise (prestructure) ANNs according to a set of propositional rules, is the approach of Tresp et al [41] (see Appendix 1). In particular they show how an initial rule base can be encoded as a set of multivariate Gaussian basis functions. The authors propose four different strategies for preserving the initial knowledge whilst still being able to refine the rule base during the ANN training process. The authors also propose two methods by which the comprehensibility of the extracted rules may be enhanced. The selected application problem domain is the prediction of housing prices in a Boston (USA) neighbourhood as a function of thirteen potentially relevant input features including for example the number of rooms in the dwelling. A salient characteristic of the technique is a probabilistic interpretation of the ANN architecture which allows the Gaussian basis functions to act as classifiers and which ultimately manifests itself in extracted rules of the form:

if the number of rooms is approximately 5.4 and the pupil/teacher value is approximately 20.2 then the value of the home is
Another significant development in this area is the RULEX technique of Andrews and Geva [1, 2] (See Appendix 1). The RULEX technique is designed to exploit the manner of construction and consequent behaviour of a particular type of multilayer perceptron, the Constrained Error Backpropagation (CEBP) MLP which is a representative of a class of local response ANN that performs function approximation and classification in a manner similar to Radial Basis Function (RBF) networks. The hidden units of the CEBP network are sigmoid-based locally responsive units (LRU's) that have the effect of partitioning the training data into a set of disjoint regions, each region being represented by a single hidden layer unit. Each LRU is composed of a set of ridges, one ridge for each dimension of the input. A ridge will produce appreciable output only if the value presented as input lies within the active range of the ridge. The LRU output is the thresholded sum of the activations of the ridges. In order for a vector to be classified by an LRU each component of the input vector must lie within the active range of its corresponding ridge. This gives rise to propositional rules of the form shown in Table 3 which can be readily extracted from the LRU's.

Table 3: Propositional rules extracted using the RULEX technique

<table>
<thead>
<tr>
<th>IF</th>
<th>Ridge$_1$ is Active and Ridge$_2$ is Active and ... Ridge$_N$ is Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>THEN</td>
<td>the pattern belongs to the `Target Class'</td>
</tr>
</tbody>
</table>

The RULEX technique also contains procedures for handling negated antecedents as well as for removing redundant/distracting antecedents and redundant rules. Unlike other decompositional methods such as KT and Subset which employ variations on `search and test' techniques, RULEX performs rule extraction by directly interpreting weight vectors as rules. Consequently RULEX obviates the computational problems of other decompositional techniques and the attendant recourse to heuristics to control the search of the solution space. This technique has been adapted to accommodate both discrete, continuous and mixed data inputs.

Boolean rule extraction using pedagogical approaches

One of the earliest published `pedagogical' approaches to rule extraction is that of Saito and Nakano [31]. In this implementation the underlying Artificial Neural Network is treated as a `black box' with rules from a medical diagnostic problem domain being extracted from changes in the levels of the input and output units. Saito and Nakano also deal with the
problem of constraining the size of the solution space to be searched by avoiding meaningless combinations of inputs (ie medical symptoms in this problem domain) and restricting the maximum number of coincident symptoms to be considered. Even with these heuristics in place, the number of rules extracted on a relatively simple problem domain was exceedingly large. This result highlights one of the major concerns with rule extraction techniques viz that the end product is explanation and not obfuscation.

The VI-Analysis (VIA) technique developed by Thrun [38] is also the epitome of a ‘pedagogical’ approach in that it extracts rules that map inputs directly into outputs (see Appendix 1.) The algorithm uses a generate-and-test procedure to extract symbolic rules from standard back-propagation Artificial Neural Networks which have not been specifically constructed to facilitate rule extraction. The basic steps in the procedure are shown in Table 4.

**Table 4: The VIA Algorithm**

1. assign arbitrary intervals to all (or a subset of all) units in the ANN. These intervals constitute constraints on the values for the inputs and the activations of the output;
2. refine the intervals by iteratively detecting and excluding activation values that are provably inconsistent with the weights and biases of the network;
3. the result of step (2) is a set of intervals which are either consistent or inconsistent with the weights and biases of the network. (In this context an interval is defined as being inconsistent if there is no activation pattern whatsoever which can satisfy the constraints imposed by the initial validity intervals.)

Thrun [38] likens the approach to sensitivity analysis in that it characterises the output of the trained Artificial Neural Network by systematic variations in the input patterns and examining the changes in the network classification. The technique is fundamentally different from other techniques which analyse the activations of individual units within a trained ANN in that focus is on what are termed ‘validity intervals’. A validity interval of a unit specifies a maximum range for its activation value. The resultant technique provides a generic tool for checking the consistency of rules within a trained ANN. The VIA algorithm is designed as a ‘general purpose’ rule extraction procedure. Thrun uses a number of examples to illustrate the efficacy of his VIA technique including (1) the XOR problem; (2) the ‘Three Monks’ problem(s); and (3) a robot arm kinematics (ie continuously valued domain) problem. While the VIA technique does not appear to be limited to any specific class of problem domains Thrun [38] reports that VIA failed to generate a complete set of rules in a relatively complex problem domain involving the task of training a network to read aloud (NETtalk).
The `Rule-extraction-as-learning' approach of Craven and Shavlik [5] is another significant development in rule extraction techniques utilising the `pedagogical' approach (see Appendix 1.) A salient characteristic of this technique is that depending on the particular implementation used, the Rule-extraction-as-learning approach can be classified either as a `pedagogical' or `decompositional'. The key is in the procedure used to establish if a given rule agrees with the network. This procedure accepts a class label \( c \) and a rule \( r \), and returns \textit{true} if all instances covered by \( r \) are classified as members of class \( c \) by the network. If, for example, Thrun's VIA algorithm [38] (as discussed previously) is used for this procedure then the approach is `pedagogical' whereas if an implementation such as that of Fu [9] is used the classification of the technique is `decompositional'. As with the VIA technique discussed earlier, the Rule-extraction-as-learning technique does not require a special training regime for the network. The authors suggest two `stopping criteria' for controlling the rule extraction algorithm \textit{viz}: (1) estimating if the extracted rule set is a sufficiently accurate model of the ANN from which the rules have been extracted; or (2) terminating after a certain number of iterations have resulted in no new rules (ie. a `patience' criterion). The authors report both on the algorithmic complexity of the technique as well as the quality of the extracted rules (with particular emphasis on rule `fidelity' which is measured by comparing the classification performance of a rule set to the trained Artificial Neural Network from which the rules were extracted. The fidelity of a rule set is the fraction of examples on which the rule set agrees with the trained Artificial Neural Network.)

Another recent example of a pedagogical approach is \textit{RULENEG} developed by Pop \textit{et al} [30] which focuses solely on the task of extracting conjunctive rules. The genesis of the \textit{RULENEG} technique is the observation that every symbolic rule in propositional calculus can be expressed as a disjunction of conjunctions. Further a conjunctive rule holds only when all the antecedents in the rule are true and hence by changing the truth value of one of the antecedents, the consequent of the rule changes. Given a trained ANN and the patterns used in the ANN training phase, the rules learned by the ANN are extracted according to the algorithm shown in Table 5.
Table 5: The RULENEG algorithm

<table>
<thead>
<tr>
<th>initialise the Rule-Holder to empty</th>
</tr>
</thead>
<tbody>
<tr>
<td>for every pattern $s$ from the training set</td>
</tr>
<tr>
<td>find the class $C$ for $s$ by use of the ANN /* $C = \text{ANN}(s)$ */</td>
</tr>
<tr>
<td>if $s$ is not classified by the existing rules</td>
</tr>
<tr>
<td>initialise a new rule $r$ for class $C$</td>
</tr>
<tr>
<td>for every input $i$ into the network</td>
</tr>
<tr>
<td>make a copy $\hat{s}$ of $s$</td>
</tr>
<tr>
<td>negate the $i$-th entry in $\hat{s}$</td>
</tr>
<tr>
<td>find the class $\hat{C}$ for $\hat{s}$ by use of the ANN /* $\hat{C} = \text{ANN}(\hat{s})$ */</td>
</tr>
<tr>
<td>if $C$ is not equal to $\hat{C}$</td>
</tr>
<tr>
<td>add $i$-th input and its truth value to $r$</td>
</tr>
<tr>
<td>/* end for every input */</td>
</tr>
<tr>
<td>add $r$ to the Rule-Holder</td>
</tr>
<tr>
<td>/* end for every pattern */</td>
</tr>
</tbody>
</table>

RULENEG is designed to select only one conjunctive rule per input pattern, but will still be able to extract all the rules learned from the patterns. An initial prototype of the RULENEG technique has been demonstrated successfully on both structured sample problem domains and `real world' problem domains.

The BRAINNE system of Sestito and Dillon [31, 32, 33] is also designed to extract rules from an Artificial Neural Network trained using standard back-propagation. In this context it has been classified as pedagogical since the basic motif is to use a measure of the closeness between the network's inputs and outputs as the focal point for generating the rule set. The further classification of this approach as one requiring a specialised ANN training regime is based on their novel idea of taking an initial trained network with $m$ inputs and $n$ outputs and transforming it into a network with $m + n$ inputs (and $n$ outputs). This transformed network is then retrained. The next phase in the process is to perform a pair-wise comparison of the weights for the links between each of the original $m$ input units and the set of hidden units with the weights from each of the $n$ additional input units and the corresponding hidden units. The smaller the difference between the two values, the greater the contribution of the original input unit (ie an attribute from the problem domain) to the output. A major innovation in the BRAINNE technique is the capability to deal with continuous data as input without first having to employ a discretising phase. The BRAINNE technique both automatically segments the continuous data into discrete ranges and extracts corresponding if...then...else... rules directly. The authors report success in applying the BRAINNE technique to a `real-world' problem domain of interpreting Submarine Sonar data.

One additional approach which falls on the periphery of this category is the
DEDEC technique of Tickle, Orlowski and Diederich [39]. A key aim in developing this technique is to provide a general vehicle for disgorging the information contained in existing trained Artificial Neural Network solutions already implemented in various problem domains. To facilitate this process, one of the core algorithms in the DEDEC technique is designed to extract symbolic rules efficiently from a set of individual cases. In this algorithm the task of rule extraction is treated as a process akin to that of identifying the minimal set of information required to distinguish a particular object from other objects. Individual cases from which rules are extracted are created by presenting a set of inputs (ie attributes) to the underlying trained Artificial Neural Network and observing the resultant output ie the classic pedagogical approach. In order to search the solution space in as optimum a fashion as possible, another key element of the DEDEC technique is a procedure for ranking the cases to be examined in order of importance. This is achieved by using the magnitude of the weight vectors in the trained Artificial Neural Network to rank the ANN input units (ie the attributes in the problem domain) according to the relative share of their contribution to the ANN output(s). Hence the focus is on extracting rules from those cases which involve what are deemed to be the most important ANN input units. From the viewpoint of technique classification, it is the introduction of the procedure for ranking the ANN input units (ie the rule antecedents) based on an analysis of the ANN weight vectors which leads to the classification of the technique as `eclectic'. The DEDEC technique also employs heuristics to terminate the process either as soon as a measure of stability appears in the extracted rule set (ie a `patience' criterion) or the relative significance of an input unit selected for generating cases to be examined, falls below some threshold value. An initial prototype of the DEDEC technique has been demonstrated successfully on both structured sample problem domains and `real world' problem domains.

Extraction of Fuzzy rules
Parallel to the development of techniques for extracting Boolean rules from trained Artificial Neural Networks, has been the synthesis of corresponding techniques for extracting fuzzy rules - the so-called neurofuzzy systems. Analogous to the techniques discussed previously for conventional Boolean logic systems, typically, neurofuzzy systems comprise three distinct elements. The first is a set of mechanisms/procedures to insert existing expert knowledge in the form of fuzzy rules into an Artificial Neural Network structure (ie a knowledge initialisation phase). The essential difference here is that this step involves the generation of representations of the corresponding membership functions. The second element is the process of training the ANN which, in this case, focuses on tuning the membership functions according to the patterns in the training data. The third element in the process is the analysis and extraction of the refined knowledge embedded in the form of a set of modified membership functions. Horikawa [20] observes that the identification of the initial set of fuzzy inference rules to be modelled has proven to be a difficult task as has attempts at
simultaneously undertaking the tasks of rule identification and membership tuning.

One of the earliest works in this area was that of Masuoka et al. [22] who used a decompositional approach to refine an initial set of fuzzy rules extracted from experts in the problem domain. The technique incorporates a specialised three phase ANN architecture. In the input phase a three layer Artificial Neural Network comprising an input unit, one or two hidden units, and an output unit was used to represent the membership function of each rule antecedent (ie the input variables). The fuzzy operations on the input variables (eg AND, OR, etc.) are represented by a second distinct phase labelled as the Rule Net (RN) phase and the membership functions which constitute the rule consequents are represented in a third (output) phase using the same motif as for the input phase. In this technique the problem of eliciting a compact set of rules as the output is tackled by pruning at the RN phase those connections in the network which are less than a threshold value.

In a similar vein Berenji [3] demonstrated the use of a specialised Artificial Neural Network to refine an approximately correct knowledge base of fuzzy rules used as part of a controller. (The problem domain selected in this case was a cart-pole balancing application.) The salient characteristic of this technique is that the set of rules governing the operation of the controller are known and the ANN is used to modify the membership functions both for the rule preconditions and the rule conclusions.

Horikawa et al [20] developed three types of fuzzy neural networks which can automatically identify the underlying fuzzy rules and tune the corresponding membership functions by modifying the connection weights of the ANNs using the backpropagation algorithm. In this approach, the initial rule base is created either by using expert knowledge or by selectively iterating through possible combinations of the input variables and the number of membership functions. The fuzzy neural network model FuNe I developed by Halgamuge et al [18] generalises this work by using a (rule based) process to initially identify 'rule relevant nodes for conjunctive and disjunctive rules for each output'. Halgamuge et al report on the successful application of the FuNe I technique to a benchmark problem involving the classification of Iris species as well as three real-world problems involving the classification of solder joint images, underwater sonar image recognition, and handwritten digit recognition.

Both FNES (fuzzy neural expert system) of Hayashi [19] and the fuzzy-MLP model of Mitra [25] specifically address the problem of providing the end user with an explanation (justification) as to how a particular conclusion has been reached. In both techniques the set of rule antecedents is determined by analysing and ranking the weight vectors in the trained ANN to determine their relative influence (impact) on a given output (class).
However whereas FNES relies on the involvement of an expert at the input phase to convert the input data into the required format, in the fuzzy-MLP procedure, this process has been automated. Both the FNES and fuzzy-MLP models has been applied to the medical problem domain of diagnosing hepatobiliary disorders with the latter showing an improved set of results (in terms of rule accuracy) over the former. In part this improvement is attributable to the more complex ANN architecture used in fuzzy-MLP viz three hidden layers cf. one hidden layer in FNES. (The FNES architecture also includes direct connections between the input and output layer.)

Okada et al [28] incorporated elements of knowledge initialisation, rule refinement (via the tuning of membership functions), and rule extraction in a fuzzy inference system incorporating a seven-layer structured ANN. In this implementation, two layers of the model are used to provide representations of the membership functions for the input variables (presented in a separate input layer) and another layer is used to represent membership functions for the rule consequents. Separate layers are also used to construct the rule antecedents (incorporating mechanisms for supporting fuzzy logical operations) and rule consequents. The authors report a significant improvement in prediction accuracy of the model in comparison with a conventional three-layer neural network in the application problem domain of financial bond rating.

EXPERIMENTAL RESULTS

In order to demonstrate the overall efficacy of the notion of extracting rules from trained ANNs and to provide a comparative evaluation of the techniques developed at the QUT Neurocomputing Research Centre, a limited series of benchmark tests were conducted. The set of techniques used were:

1. RULEX (Andrews and Geva [1,2])
2. LAP (Hayward et al. [44])
3. RULENEG (Hayward et al. [30,45])
4. DEDEC (Tickle et. al [39])

As indicated previously, the raison d'être for these techniques is to extract rules from Artificial Neural Networks (ANNs) which have been trained on a set of data from a given problem domain. This is distinct from a class of techniques termed "symbolic induction techniques" which are designed to infer the classification rules directly from the data presented. TREX [46] which has been developed at the QUT Neurocomputing Research Centre is an example of this type of technique. For the purposes of comparison results for TREX have also been included in the benchmark evaluation along with the results from an implementation of a published technique by Craven and Shavlik [5] and two other symbolic induction techniques C4.5 [47] and T2 [48] both of which are based on a "decision tree" approach.
The initial problem domain selected was the set of Three Monks problems proposed by Thrun et al. [37] (See Appendix 2). Apart from the inherent complexity of these problems, their choice affords a basis of comparison with previous work particularly in the application of symbolic induction techniques.

The second problem domain selected was the Mushroom classification problem (Appendix 3).

For the purposes of this benchmark, the three (3) primary evaluation criteria used to compare the "quality" of the rules extracted using the various techniques were those identified earlier viz:

(a) rule accuracy;
(b) rule fidelity; and
(c) rule comprehensibility.

In this context, rule accuracy is a measure of the proportion of the cases from a given subset of the total problem domain (i.e. a "test" set) which have been correctly classified by the rules extracted from the ANN. Rule fidelity is a measure of the agreement between the classification of the "test" set by the ANN and by the rule set extracted from the ANN. Rule comprehensibility is reported as the number of rules produced by the rule extraction process and the average number of antecedents per rule. The fourth criterion of rule "consistency" indicates (as "yes" / "no") whether a given set of weight vectors which define a particular training instance of an ANN will always produce the same rule set under the rule-extraction technique in question.

A summary of the results is shown in Appendix 4. Even though some of the techniques developed at the NRC are still in their "prototype" phase, the results compare favourably with other published results.

OPEN RESEARCH QUESTIONS AND DIRECTIONS FOR FURTHER WORK

One of the most pressing problems in the field of rule-extraction is the formulation of a set of criteria for matching the set of techniques to the requirements of a given problem domain. For example at a practical level, what has not yet emerged is a means of determining which rule-extraction technique is optimal for application problem domains involving real valued data as distinct from discrete data. Further it is also uncertain as to whether the reported improvement in performance of ANN/rule-extraction techniques vis-a-vis other induction techniques for extracting rules from data, applies in all problem domains. Hence a pressing requirement is to extend the comparative benchmark tests reported here across an even broader range of problem domains.

A related issue is that in an increasing number of applications there are reports of situations in which the extracted rule-set has shown better
generalisation performance than the trained Artificial Neural Network from which the rule-set was extracted [40]. Similar observations have also been made in the area of extracting symbolic grammatical rules from recurrent Artificial Neural Networks [16, 17, 29]. However Giles and Omlin [16] also report that larger networks tend to show a poorer generalisation performance. While these results are significant, what is not clear at this stage is the extent to which this superior performance can be ascribed to the elimination of the remaining error over the output unit(s) after the Artificial Neural Network training has been completed (ie the ‘rest’ error). Hence an important research topic is also to identify the set of conditions under which an extracted rule set shows better generalisation than the original network.

A third area which warrants further investigation arises from the observation that a characteristic feature of the rule extraction techniques surveyed is for the rules to be extracted after the completion of the ANN training process. A question therefore arises as to whether there are points in the training process at which subsets of the final rule set could be extracted [40] and also if the tasks of rule insertion and rule extraction could occur during ANN training [15].

Issues relating to the complexity of the underlying rule extraction algorithms have also been the subject of discussion in the preceding review. The benefits of such discussion will be realised if as expected, they contribute to a situation in which some of the mistakes of early AI and ANN approaches are obviated. However an additional facet of the discussion on algorithm complexity and another area for further investigation is a determination of whether the problem of finding a minimal rule-set which imitates a network with high fidelity is a hard (possibly NP-complete) problem.

With the exception of the RULEX technique discussed above, the rule extraction techniques surveyed require some form of heuristics to constrain the size of the sample space to be searched. For example in the `decompositional' approaches, thresholds are used to filter those inputs which have no significant effect on the final decision produced by the ANN. Hence a potential research issue is an assessment of the impact of such heuristics on the quality and efficacy of the rules produced.

CONCLUSION
This report provides a survey of the salient features of a cross-section of published techniques for extracting rules from trained Artificial Neural Networks. The modus operandi of the various rule-extraction algorithms have been discussed, a new classification scheme for rule-extraction methods introduced and the most important evaluation criteria for rule-extraction techniques have been outlined. The report also includes a comparative evaluation of a subset of these techniques including in particular the set of techniques developed at the Neurocomputing Research
Centre at the QUT.

As evidenced by the diversity of `real-world' application problem domains in which rule extraction techniques have been applied, there appears to be a strong and continuing demand for the end-product of the rule extraction process viz a comprehensible explanation as to how and why the trained ANN arrived at a given result or conclusion. This demand appears to fall broadly within two groups: (1) ANN solutions which have already been implemented and where *ipso facto* the user is interested in identifying and possibly exploiting the potentially rich source of information which already exists within the trained ANN; and (2) a `green-field' situation where a user has a data set from a problem domain and is interested in what relationships exist both within the data given and what general conclusions can be drawn.

The first group requires the development of rule extraction techniques which can be applied to existing ANNs. At this stage it would appear that, notwithstanding the initial success of `decompositional' approaches such as that of the _KT_ algorithm of Fu [9, 12], the `pedagogical' approach is well-placed to serve this set. Similarly it could be argued that the second group might well become the province of those rule extraction techniques which use specialised Artificial Neural Network training regimes, given the reported success of, for example, _KBANN/M-of-N, BRAINNE, RULEX, _etc. However it also clear that no single rule extraction/rule refinement technique or method is currently in a dominant position to the exclusion of all others.

**ACKNOWLEDGMENTS**

The authors would also like to express their thanks to Mark Craven and two anonymous reviewers for their many helpful and incisive comments on the version of this report submitted for publication.
REFERENCES

1. Andrews, R and Geva, S `Rule extraction from a constrained error back propagation MLP' Proc. 5th Australian Conference on Neural Networks Brisbane Queensland (1994) pp 9-12
15. Giles, C L and Omlin C W `Rule refinement with recurrent neural networks' Proc. of the IEEE International Conference on Neural Networks (San Francisco CA) (March 1993) pp 801-806
16. Giles, C L and Omlin C W `Extraction, insertion, and refinement of


19 Hayashi, Y `A neural expert system with automated extraction of fuzzy if-then rules and its application to medical diagnosis' *Advances in Neural Information Processing Systems* (Denver CO) Vol 3 (1990) pp 578-584

20 Horikawa, S, Furuhashi, T and Uchikawa, Y `On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm' *IEEE Transactions on Neural Networks* Vol 3 No 5 (September 1992) pp 801-806


29 Omlin, C W, Giles, C L and Miller, C B `Heuristics for the extraction of rules from discrete time recurrent neural networks' *Proc. of the International Joint Conference on Neural Networks (IJCNN92)* (Baltimore MD) Vol 1 (1992) pp 33

30 Pop, E, Hayward, R, and Diederich, J `RULENEG: extracting rules
from a trained ANN by stepwise negation' QUT NRC (December 1994)

31 Saito, K and Nakano, R `Medical diagnostic expert system based on PDP model' Proc. of IEEE International Conference on Neural Networks (San Diego CA) Vol 1 (1988) pp 255-262


34 Sestito, S and Dillon, T `Automated knowledge acquisition' Prentice Hall (Australia) (1994)


36 Swartout, W R `Lecture on explanation' GMD (September 1989)


39 Tickle, A B, Orlowski, M, and Diederich, J `DEDEC: decision detection by rule extraction from neural networks' QUT NRC (September 1994)


Hayward R, Ho-Stuart C, Diederich J and Pop E  `RULENEG: extracting rules from a trained ANN by stepwise negation' *QUT NRC* (January 1996)

Geva S  `Direct production rule extraction' *QUT NRC* (November 1995)

Quinlan J R  `C4.5 programs for machine learning' *Morgan Kaufmann* 1993

Holte R C  `Very simple classification rules perform well on most commonly used datasets' *Machine Learning* 11 1993 pp63-91
Appendix 1

Survey Details

(Detailed description of what are considered to be some of the more important contributions in the field)
1. NAME/TITLE:  
\*RuleNet & the Connectionist Scientist Game\*

Bibliographic Reference(s):  
McMillan, Mozer, and Smolensky [24]

Classification:  
(a) Technique:  
deecompositional  
(b) Rule Type:  
propositional if...then...else  
(The authors use the term `condition-action' rules)

Description:  
RuleNet is an ANN that learns string→string mappings and from which explicit, symbolic, condition-action rules can be extracted. The Connectionist Science Game which describes the training algorithm employed by RuleNet, is based on the `scientific method' ie induce a hypothesis from observation, iteratively test and refine the hypothesis until the hypothesis explains the observations. An outline of the (iterative) process is shown in Table A1. The process continues until the extracted rules adequately characterise the problem domain.

<table>
<thead>
<tr>
<th></th>
<th>Iterative process for the <strong>Connectionist Science Game</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>train RuleNet on a set of input/output examples, (which corresponds to the scientist developing intuitions about a problem domain)</td>
</tr>
<tr>
<td>2</td>
<td>extract symbolic rules from the connection strengths in the network, (development of an explicit hypothesis)</td>
</tr>
<tr>
<td>3</td>
<td>inject the rules back into the network and continuing training, (testing the hypothesis)</td>
</tr>
</tbody>
</table>

RuleNet is a 3 layer network consisting of an input layer, a layer of condition units, and a layer of output action units. The weight vector \(c_i\) which connects input units to the condition units is used to detect the condition of rule to be learned, ie required characters and relations between characters. After training each condition unit represents a rule. The weight matrix, \(A_i\), connecting condition unit \(i\) to output units is set up to ensure that there is a unique mapping between \(input_\alpha\), (the character in position \(\alpha\) in the input vector), and \(output_\beta\), (the character in position \(\beta\) in the output vector). Rule extraction is achieved by decompiling the weight vector \(c_i\) to form the condition component of the rule, and the weight matrix \(A_i\) to form the action component of the rule. By having each condition unit represent a single rule and by establishing the unique mapping between input and
output characters via the condition unit and associated weight matrix, RuleNet exploits some degree of localisation of response which (a) overcomes the combinatorial search-and-test problem faced by other decompositional techniques, and (b) allows the rules to be directly extracted by inspection of the $c_i-A_i$ pairs.

**Problem Domain/Validation Example:**
The problem domain selected to illustrate the technique is restricted to those `rule based' domains that map input strings of $n$ symbols to output strings of $n$ symbols. The rules describing the problem domain are `mutually exclusive condition-action' rules where a condition is a `feature or combination of features present in each input in order for a given rule to apply', and the `action' describes the mapping to be performed on each symbol in the string if the condition applies. Test problems were constructed on an eight(8) character alphabet using strings of length four(4) characters. Four test problems were constructed using rule bases consisting of eight(8), three(3), three(3), and five(5) rules respectively.

**Algorithm Complexity:**
The network itself is constructed specifically so as to be able to detect the mapping between individual character positions in a string. The rule extraction process takes advantage of this and uses a process called `projection' to convert non-essential weights in $c_i$ and $A_i$ to 0, and essential weights to 1.

**Comments:**
The method is designed for a only specific problem domain and although the authors claim that this `can be viewed as an interesting abstraction of several interesting cognitive models in the connectionist literature', it still appears to lack generality.

2. **NAME/TITLE:**
*Subset*

**Bibliographic Reference(s):**
Towell and Shavlik [40] (see Comments below)
Fu [9, 12]

**Classification:**
(a) Technique: *decompositional*

(b) Rule Type: *propositional if...then...else*

**Description:**
In the `Subset' algorithm the focus is on extracting rules at the level of
individual (hidden and output) units within the trained Artificial Neural Network. The rules extracted at the individual unit level are then aggregated to form the composite rule base for the ANN as a whole. The algorithm is so named because the basic idea is to search for subsets of incoming weights to each unit which exceed the bias on a unit. (The key underlying assumption is that the corresponding `signal' strength associated with each connection to the unit is either zero or one ie `maximally active' or `inactive').

The initial search is for sets containing a single link/connection of sufficient (positive) weight to guarantee that the bias on the unit is exceeded irrespective of the values on the other links/connections [40]. (Note that because the values of the activation functions are either near zero or 1, the search reduces to scanning the weight vectors only.) If a link is found which satisfies the criterion, it is written as a rule. The search then proceeds to subsets of two elements et seq. A schematic of the basic algorithm as reported by Towell and Shavlik [40] is given in Table A2

<table>
<thead>
<tr>
<th>Table A2: The SUBSET algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>for each hidden and output unit:</td>
</tr>
<tr>
<td>extract up to ( S_p ) subsets of the positively-weighted incoming links for which the summed weight is greater than the bias on the unit;</td>
</tr>
<tr>
<td>for each element ( p ) of the ( S_p ) subsets:</td>
</tr>
<tr>
<td>search for a set ( S_n ) of a set of negative-attributes so that the summed weights of ( p ) plus the summed weights of ( N - n ) (where ( N ) is the set of all negative-attributes and ( n ) is an element of ( S_n )) exceed the threshold on the unit;</td>
</tr>
<tr>
<td>with each element ( n ) of the ( S_n ) set, form a rule: 'if ( p ) and NOT ( n ), then the concept designated by the unit'.</td>
</tr>
</tbody>
</table>

**Problem Domain/Validation Example:**
The Subset rule extraction algorithm is effectively a 'general purpose' procedure which does not appear to be specific to a particular problem domain. Towell and Shavlik use the Subset algorithm as a `straw man' to illustrate the effectiveness of their M-of-N technique and hence an implementation of the Subset algorithm has been used in each of the sample problems they investigated. These included two from the field of molecular biology viz (a) prokaryotic promoter recognition, and (b) primate splice-junction determination as well as the perennial `Three Monks' problem(s).

**Algorithm Complexity:**
Towell and Shavlik comment extensively on the deficiencies of the Subset algorithm to illustrate the efficacy of their own M-of-N approach.
Rule Quality
Similarly Towell and Shavlik comment that although the individual rules extracted using the Subset algorithm are both comprehensible and tractable, (a) the algorithm has the potential to extract a large number of rules eg up to $B_pX(1 + B_p)$ for each hidden and output unit, (b) some of the rules may repetitive, and (c) the extracted rules may hide significant (eg. $M$ of $N$) structures.

Comments:
Towell and Shavlik base their implementation of the Subset on earlier work by Fu [9] and Saito and Nakano [31] but stress the implementation used for comparison with their M-of-N approach is their own version. They argue that because the solution time of the algorithm increases exponentially with the number of ANN input units, it is suitable only for simple networks or so-called 'small' problem domains.

3. NAME/TITLE:
M-of-N

Bibliographic Reference(s):
Towell and Shavlik [40]

Classification:
(a) Technique: decompositional
(b) Rule Type: (modified) boolean { see Comments below}

Description:
The phases of the M-of-N algorithm are shown in Table A3.
Table A3: The $M$-of-$N$ technique

1. generate an Artificial Neural Network using the KBANN system and train using back-propagation. With each hidden and output unit, form groups of similarly-weighted links;
2. set link weights of all group members to the average of the group;
3. eliminate any groups which do not significantly affect whether the unit will be active or inactive;
4. holding all link weights constant, optimise biases of all hidden and output units using the back propagation algorithm;
5. form a single rule for each hidden an output unit; the rule consists of a threshold given by the bias and weighted antecedents specified by the remaining links;
6. where possible, simplify rules to eliminate superfluous weights and thresholds.

Problem Domain/Validation Example:
The $M$-of-$N$ algorithm is designed as a ‘general purpose’ rule extraction procedure and its applicability does not appear to be limited to any specific class of problem domains. Towell and Shavlik use a number of examples to illustrate the efficacy of their $M$-of-$N$ technique including two from the field of molecular biology viz (a) prokaryotic promoter recognition, and (b) primate splice-junction determination as well as the perennial ‘Three Monks’ problem(s).

Algorithm Complexity:
The algorithm addresses the crucial question of reducing the complexity of rule searches by clustering the ANN weights into equivalence classes (and hence extracting $M$-of-$N$ type rules) [5]. Using three indicative parameters: (1) the number of units in the ANN ($u$), (2) the average number of links received by a unit ($l$), and (3) the number of training examples ($n$). Towell and Shavlik’s assessment of the complexity of the $M$-of-$N$ algorithm is shown in Table A4.
Table A4: Complexity of the M-of-N algorithm

<table>
<thead>
<tr>
<th>Step No.</th>
<th>Name</th>
<th>Estimated Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>clustering</td>
<td>$O(u \times i)$</td>
</tr>
<tr>
<td>2</td>
<td>averaging</td>
<td>$O(u \times l)$</td>
</tr>
<tr>
<td>3</td>
<td>eliminating</td>
<td>$O(n \times u \times l)$</td>
</tr>
<tr>
<td>4</td>
<td>optimising</td>
<td>precise analysis is inhibited by the use of back-propagation in this (bias-optimisation) phase</td>
</tr>
<tr>
<td>5</td>
<td>extracting</td>
<td>$O(u \times l)$</td>
</tr>
<tr>
<td>6</td>
<td>simplifying</td>
<td>$O(u \times l)$</td>
</tr>
</tbody>
</table>

Rule Quality:
Towell and Shavlik use two dimensions viz (a) `the rules must accurately categorise examples that were not seen during training', and (b) `the extracted rules must capture the information contained' in the knowledge based Artificial Neural Network (KNN), for assessing the quality of rules extracted both from their own algorithm and from the set of algorithms they use for the purposes of comparison. In their view the M-of-N idea inherently yields a more compact rule representation than conventional conjunctive rules produced by algorithms such as Subset. In addition the M-of-N algorithm outperformed a subset of published symbolic learning algorithms in terms of the accuracy and fidelity of the rule set extracted from a cross-section of problem domains.

Comments:
The salient feature of the M-of-N technique is the explicit searching for rules of form: If (M of the following N antecedents are true) then .... This follows from the premise that in a significant number of `real world' applications, individual rule antecedents do not have unique importance - hence the shift in focus to equivalence classes. In the implementation discussed by Towell and Shavlik, the ANN is generated via the KBANN: symbolic knowledge to ANN translator. (KBANN defines the topology and connection weights of the ANN created and the package is geared towards `rule refinement' viz symbolic rules $\Rightarrow$ ANN representation $\Rightarrow$ rule refinement $\Rightarrow$ symbolic rules.)

4. NAME/TITLE:
   Rule-extraction-as-learning

Bibliographic Reference(s):
Craven and Shavlik [5]

**Classification:**
(a) **Technique:** eclectic
(b) **Rule Type:** propositional if...then...else and M-of-N

**Description:**
The core idea is to ‘view rule extraction as a learning task where the target concept is the function computed by the network and the input features are simply the network’s input features’. A schematic outline of the overall algorithm is shown in Table A5.

**Table A5: The Rule-extraction-as-learning technique**

```plaintext
/* initialise rules for each class */
for each class c
  Rc := 0
repeat
  e := Examples()
  c := Classify(e)
  if e not covered by Rc then
    /* learn a new rule */
    r := conjunctive rule formed from e
    for each antecedent r_i of r
      r' := r but with r_i dropped
      if Subset(c,r') = true then r := r'
    Rc := Rc ∨ r
  until stopping criterion met
```

The role of the `Examples` function is to provide training examples for the rule-learning algorithm. The options used are (1) select members of the set used for training the Artificial Neural Network, (2) random sampling, or (3) random creation of examples of a specified class (see Table A6).
Craven and Shavlik use a function which they call Subset to determines if the modified rule still agrees with the network if all instances that are covered by the rule are members of the given class.

Problem Domain/Validation Example:
The algorithm is designed as a ‘general purpose’ rule extraction procedure and its applicability does not appear to be limited to any specific class of problem domains. Craven and Shavlik illustrate the efficacy of their technique on the prokaryotic promoter recognition problem from the field of molecular biology.

Algorithm Complexity:
One of the stated aims of the authors is to reduce the amount of computation to achieve the same degree of rule fidelity as the ‘decompositional (or search-based)’ algorithms. One of the crucial differences between this algorithm and search-based extraction methods is that it explores the space of rules from the bottom up as distinct from the conventional top down approach.

Using two indicative parameters: (1) the number of input features \( f \), and (2) the total number of values for all input features \( v \), Craven and Shavlik estimate the complexity of their algorithm as shown in Table A7. The authors also note that the complexity of the Examples function is \( O(f) \) when using a random sampling technique (ie Option 2 above). For Option 3 (ie Random creation of examples) the estimated complexity is \( O(v \log v) \) where \( v \) is the total number of values for all features. The complexity of the Subset function is greater than \( O(f) \) if for example, the VIA [38] algorithm is used in the implementation.
Table A7: Estimated complexity of the Rule-extraction-as-learning technique

<table>
<thead>
<tr>
<th>Operation No.</th>
<th>Description</th>
<th>Estimated Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>calls to the Examples oracle</td>
<td>$O(f)$</td>
</tr>
<tr>
<td>2</td>
<td>calls to the Subset oracle</td>
<td>$O(f)$</td>
</tr>
<tr>
<td>3</td>
<td>subsumption-check comparisons to the individual terms of $R_c$, the Disjunctive Normal Form (DNF) expression for class $c$</td>
<td>$O(f)$</td>
</tr>
</tbody>
</table>

**Rule Quality:**
The authors evaluate their algorithm in terms of what they call 'fidelity'. This is measured by comparing the classification performance of a rule set to the trained Artificial Neural Network from which the rules were extracted. The fidelity of a rule set is the fraction of examples on which the rule set agrees with the trained Artificial Neural Network.

**Comments:**
Much of the discussion on suggested improvements to the technique centre on directing the Examples function which produces training For examples for the rule-learning algorithm. For example one idea is to use the current set of extracted rules to influence sampling.

5. **NAME/TITLE:**
VIA (Validity Interval Analysis)

**Bibliographic Reference(s):**
Thrun [38]

**Classification:**
(a) **Technique:**
pedagogical
(b) **Rule Type:**
propositional if...then...else

**Description:**
An outline of the VIA algorithm is given in Table A8.
Table A8: The VIA Algorithm

1. assign arbitrary intervals to all (or a subset of all) units in the ANN. These intervals constitute constraints on the values for the inputs and the activations of the output;
2. refine the intervals by iteratively detecting and excluding activation values that are provably inconsistent with the weights and biases of the network;
3. the result of step (2) is a set of intervals which are either consistent or inconsistent with the weights and biases of the network. (In this context an interval is defined as being inconsistent if there is no activation pattern whatsoever which can satisfy the constraints imposed by the initial validity intervals.)

Problem Domain/Validation Example:
The VIA algorithm is designed as a ‘general purpose’ rule extraction procedure. Thrun uses a number of examples to illustrate the efficacy of his VIA technique including (1) the XOR problem, (2) the (perennial) ‘Three Monks’ problem(s), and a robot arm kinematics (ie continuously valued domain) problem. The VIA does not appear to be limited to any specific class of problem domains. However Thrun [38] reports that VIA failed to generate a complete set of rules in a relatively complex problem domain involving the task of training a network to read aloud (NETtalk).

Algorithm Complexity:
No detailed analysis of algorithm complexity is provided by the author. However one of the crucial steps in the procedure involves establishing the validity intervals in which a unit in the trained Artificial Neural Network becomes active. This step involves solving a linear programming problem and hence the algorithmic complexity is dependent on the particular linear programming algorithm selected.

The author notes that one of the salient characteristics of the VIA algorithm is the capability to constrain the size of the rule search space by allowing the validity of more general rules to be determined before specific rules are examined.

Rule Quality:
The author does not address the topic of rule quality as a specific topic. However the author evaluates the efficacy of the rule extraction capability of the VIA algorithm in terms of rule quality criteria such as comprehensibility.

Comments:
The VI-Analysis technique is the epitome of a ‘pedagogical’ approach in that it extracts rules that map inputs directly into outputs. The author likens the approach to sensitivity analysis in that it characterises the output of the trained Artificial Neural Network by systematic variations in the input patterns and examining the changes in the network classification. The
technique is fundamentally different from other techniques which analyse
the activations of individual units within a trained ANN in that focus is on
what are termed ‘validity intervals’. A validity interval of a unit specifies a
maximum range for its activation value. The resultant technique provides a
generic tool for checking the consistency of rules within a trained Artificial
Neural Network.

6. NAME/TITLE:
RULEX

Bibliographic Reference:
Andrews and Geva [1, 2]

Classification:
(a) Technique: decompositional
(b) Rule Type: propositional if...then...else

Description:
The technique is designed to exploit the manner of construction and
consequent behaviour of a particular type of multilayer perceptron (MLP) viz
the Constrained Error Back-propagation (CEBP) MLP. This is a
representative of a class of local response ANN that performs function
approximation and classification in a manner similar to Radial Basis
Function (RBF), networks. The hidden units of the CEBP network are
sigmoid-based locally responsive units (LRU’s), that have the effect of
partitioning the training data into a set of disjoint regions, each region being
represented by a single hidden layer unit. Each LRU is composed of a set
of ridges, one ridge for each dimension of the input. A ridge will produce
appreciable output only if the value presented as input lies within the active
range of the ridge. The LRU output is the thresholded sum of the
activations of the ridges. In order for a vector to be classified by an LRU
each component of the input vector must lie within the active range of its
corresponding ridge. Hence the rule derived from an N dimensional local
bump will be of the form shown in Table A9.

Table A9: Propositional rules extracted using the RULEX technique

| IF Ridge₁ is Active and Ridge₂ is Active and ... Ridgeₙ is Active |
| THEN the pattern belongs to the 'Target Class' |
The active range for each ridge can be calculated from its centre, breadth, and steepness ($c_i$, $b_i$, $k_i$), weights in each dimension. This means that it is possible to directly decompile the LRU parameters into a conjunctive propositional rule of the form shown in Table A10.

Table A10: Conjunctive propositional rules using the RULEX technique

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF $c_i - b_i + 2k_i^{-1} \leq x_i \leq c_i + b_i - 2k_i^{-1}$</td>
<td>(1)</td>
</tr>
<tr>
<td>AND $c_2 - b_2 + 2k_2^{-1} \leq x_2 \leq c_2 + b_2 - 2k_2^{-1}$</td>
<td>(2)</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>AND $c_N - b_N + 2k_N^{-1} \leq x_N \leq c_N + b_N - 2k_N^{-1}$</td>
<td>(N)</td>
</tr>
<tr>
<td>THEN</td>
<td>the pattern belongs to the 'Target Class'</td>
</tr>
</tbody>
</table>

For discrete valued input it is possible to enumerate the active range of each ridge as an OR'ed list of values that will activate the ridge. In this case it is possible to state the rule associated with the LRU in the form shown in Table A11.

Table A11: Conjunctive propositional rules for discrete input values using the RULEX technique

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF $v_{i1a}$ OR $v_{i1b}$ ... OR $v_{1n}$</td>
<td>(1)</td>
</tr>
<tr>
<td>AND $v_{2a}$ OR $v_{2b}$ ... OR $v_{2n}$</td>
<td>(2)</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>AND $v_{Na}$ OR $v_{Nb}$ ... OR $v_{Nn}$</td>
<td>(N)</td>
</tr>
<tr>
<td>THEN</td>
<td>the pattern belongs to the 'Target Class'</td>
</tr>
</tbody>
</table>

(where \(v_{ia} \leq c_i - b_i + 2k_i^{-1}\) and \(v_{in} \leq c_i - b_i + 2k_i^{-1}\))

The technique provides mechanisms for removing redundant antecedent clauses, (ie, input dimensions that are not used in classification), from the extracted rule, and for removing redundant rules, (ie, replacing two rules with a single more general rule).

Problem Domain/Validation Example:
The author's have used a number of examples to demonstrate the applicability of the RULEX technique including (1) the 'Three Monks' Problem, (2) the DNA prokaryotic promoter recognition problem, and (3) the poisonous/edible mushroom classification problem.

Algorithm Complexity:
The authors argue that the algorithm is computationally extremely efficient
in that unlike other decompositional methods, this method is not a 'search
and test' method. Rather RULEX performs rule extraction by direct
interpretation of weight parameters as rules.

Rule Quality:
The RULEX algorithm employs an incremental training scheme on the
CEBP network results in a solution which has the minimum number of
LRU's and hence the extracted rule set will be composed of the minimum
number of rules. RULEX produces rules which are accurate and show very
high fidelity. The number of rules produced is generally equal to the number
of local functions used by the network in its solution. (If more than the
minimum number of local functions are used RULEX tries to absorb two or
more redundant rules into a single, more general rule.) The number of
antecedents per rule is in the range 1..N where N is the dimensionality of
the problem. RULEX guarantees to include only those dimensions actually
used in classifying patterns in the extracted rules thus making individual
rules easily comprehensible.

7. NAME/TITLE:
Network Structuring and Training Using Rule-based Knowledge

Bibliographic Reference(s):
Tresp, Hollatz and Ahmad [41]

Classification:
(a) Technique:  
decompositional
(b) Rule Type:  
propositional if...then...else

Description:
The technique is based on the premise that prior knowledge of the problem
domain is available in the form of a set of rules. An Artificial Neural Network
\( y = NN(x) \), which makes a prediction about the state of \( y \) given the state of
its input \( x \) can be instantiated as a set of basis functions, \( b_i(x) \), where each
basis function describes the premise of the rule that results in prediction \( y \).
The degree of certainty of the rule premise is given by the value of \( b_i(x) \)
which varies continuously between 0 and 1. The rule conclusion is given by
\( w_i(x) \) and the network architecture is given as:

\[
y = NN(x) = \frac{\sum_i w_i(x) b_i(x)}{\sum_j b_j(x)}
\]

The authors show how the basis functions can be formed by encoding
simple logical *if-then* expressions as multivariate Gaussians. A salient characteristic of the technique is a probabilistic interpretation of the ANN architecture which allows the Gaussian basis functions to act as classifiers. Training can proceed in any of four modes including (1) *Forget*, where training data is used to adapt \( NN^{\text{init}} \) by gradient descent (ie the sooner training stops, the more initial knowledge is preserved), (2) *Freeze*, where the initial configuration is frozen (ie if a discrepancy between prediction and data occurs, a new basis function is added), (3) *Correct* where a parameter is penalised if it deviates from its initial value; and (4) *Internal Teacher* where the penalty is formulated in terms of the mapping rather than in terms of the parameters.

After training is complete, a `pruning` strategy (rule refinement), is employed to arrive at a solution which has the minimum number of basis functions (rules), and the minimum number of conjuncts for each rule. The strategy is shown in Table A12.

**Table A12: Post-training in the Network Structuring and Training Using Rule-based Knowledge technique**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error &lt; Threshold</td>
<td>Retrain the network till no further improvement in error</td>
</tr>
<tr>
<td>Prune/Remove Basis Function</td>
<td>Either Prune/Remove Basis Function which has least importance to the network, (remove the least significant rule)</td>
</tr>
<tr>
<td>Prune Conjuncts</td>
<td>Or Prune Conjuncts by finding the Gaussian with the largest radius and setting this radius to infinity, (effectively removing the associated input dimension from the basis function)</td>
</tr>
</tbody>
</table>

Rule premises are extracted by directly decompiling the Gaussians are stated in terms of conjunctions of (Centre, Radius) pairs of the Gaussians which comprise the basis functions, \( b_i(x) \). Rule conclusions are determined by evaluating the conclusion \( w_r \).

**Problem Domain/Validation Example:**  
Approximation of a noisy sinusoid to demonstrate pruning, (no rules extracted), bicycle control problem, (no results presented), Boston housing problem. Here the task was to predict the price of a house in Boston given 13 potentially relevant features, (sample of total extracted rules provided).

**Algorithm Complexity:**  
No details are given by the authors. **Rule Quality:**  
Insufficient detail is provided to adequately assess the quality of rules.
extracted by this method.

8. NAME/TITLE:
   DEDEC

Bibliographic Reference(s):
Tickle, Orlowski, and Diederich [39]

Classification:
(a) Technique:
   pedagogical/ eclectic
(b) Rule Type:
   propositional if...then...else

Description:
The DEDEC technique is more appropriately labelled as a methodology for extracting explanations from a general class of "backpropagation" trained Artificial Neural Networks. In essence it seeks to provide the requisite explanation capability by eclectically combining the generalisation/ "noise robustness" capability of trained ANNs [38] with the inherent explanation power of symbolic induction algorithms. However a salient distinguishing characteristic of the DEDEC approach is to use both the trained ANN itself and information extracted from analysing the weight vectors in the trained ANN, to create a set of examples drawn from the entire spectrum of the problem domain. The requisite rule set is generated by using a symbolic induction algorithm to "learn" from these examples. Importantly such examples may (and almost invariably will) extend beyond those available in the set of data used to train the ANN.

Basically the DEDEC methodology comprises five distinct phases (Figure 1) viz.:
1. encoding the examples from the problem domain in a form suitable for training by an ANN
2. training the ANN;
3. analysing the resultant weight vectors in the trained ANN;
4. using the trained ANN "in situ" in conjunction with the information gleaned from Phase 3 to generate a controlled set of examples from across the entire problem domain; and
5. invoking a symbolic induction algorithm to "learn" these examples and disgorge a corresponding set of symbolic rules.

In essence, collectively, Phases 3, 4, and 5 constitute the knowledge discovery component of the DEDEC methodology.
THREE MONKS PROBLEMS

In the Three Monks problem domain, objects labelled as ‘monks’ are described by six different attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>head_shape</td>
<td>round</td>
</tr>
<tr>
<td>body_shape</td>
<td>round</td>
</tr>
<tr>
<td>is_smiling</td>
<td>yes</td>
</tr>
<tr>
<td>holding</td>
<td>sword</td>
</tr>
<tr>
<td>jacket_colour</td>
<td>red</td>
</tr>
<tr>
<td>has_tie</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>head_shape</td>
<td>square</td>
</tr>
<tr>
<td>body_shape</td>
<td>square</td>
</tr>
<tr>
<td>is_smiling</td>
<td>no</td>
</tr>
<tr>
<td>holding</td>
<td>balloon</td>
</tr>
<tr>
<td>jacket_colour</td>
<td>yellow</td>
</tr>
<tr>
<td>has_tie</td>
<td>no</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>head_shape</td>
<td>octagon</td>
</tr>
<tr>
<td>body_shape</td>
<td>octagon</td>
</tr>
<tr>
<td>is_smiling</td>
<td></td>
</tr>
<tr>
<td>holding</td>
<td>flag</td>
</tr>
<tr>
<td>jacket_colour</td>
<td>green</td>
</tr>
<tr>
<td>has_tie</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>head_shape</td>
<td></td>
</tr>
<tr>
<td>body_shape</td>
<td></td>
</tr>
<tr>
<td>is_smiling</td>
<td></td>
</tr>
<tr>
<td>holding</td>
<td></td>
</tr>
<tr>
<td>jacket_colour</td>
<td>blue</td>
</tr>
<tr>
<td>has_tie</td>
<td></td>
</tr>
</tbody>
</table>

(The problem domain contains a total of 432 cases)

The Three Monks problem domain comprises three distinct problem. In the first problem (Monk 1), the (compact) decision rule for membership of the target class (ie is ‘a monk’) is:

(head_shape = body_shape) or (jacket_colour = red).

Using the notation of Table A13, the expanded rules are:

head_shape = round and body_shape = round; or
head_shape = square and body_shape = square; or
head_shape = octagon and body_shape = octagon; or
jacket_colour = red.

The sub-set of the total problem domain used for the ANN training process contains 124 cases.

For the Monk 2 problem, the (compact form) of the decision rule for membership of the target class (ie is ‘a monk’) for this problem is:

exactly two of the six attributes have their first value.

This expands to a set of fifteen rules of the form:

if
\begin{verbatim}
head_shape = round and
body_shape = round and
not is_smiling = yes and
not holding = sword
and not jacket_colour = red and
not has_tie = yes then
'a monk'.
\end{verbatim}

If antecedent negation is not used then by enumerating all of the possible permutations it can be shown that there exists some 142 possible rules involving simple conjunctions of the antecedents of the form:
\begin{verbatim}
if
head_shape = round and
body_shape = round and
is_smiling = no and
holding = balloon
jacket_colour = yellow and
has_tie = no then
'a monk'.
\end{verbatim}

for deciding membership of the target class.

For the Monk 2 problem the sub-set of the total problem domain used for the ANN training process contains 169 cases.

For the Monk 3 problem the (compact) decision rule for membership of the target class (ie is `a monk') is:
\begin{verbatim}
(body_shape not octagon and jacket_colour not blue) or
(holding = sword and jacket_colour = green).
\end{verbatim}

{In passing it worth noting that these rules give rise to a simple and `sufficient' rule for exclusion from the target class (ie is not `a monk') viz
\begin{verbatim}
jacket_colour = blue.
\end{verbatim}

For the Monk 3 problem the sub-set of the total problem domain used for the ANN training process contains 122 cases. Of these six(6) have been deliberately corrupted.
## APPENDIX 3

### MUSHROOM CLASSIFICATION PROBLEM

Table A14: Attributes and their values for the Mushroom Classification problem

<table>
<thead>
<tr>
<th>Attribute</th>
<th>No.</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cap Shape</td>
<td>6</td>
<td>bell, conical, convex, flat, knobbed, sunken</td>
</tr>
<tr>
<td>Cap Surface</td>
<td>4</td>
<td>fibrous, grooves, scaly, smooth</td>
</tr>
<tr>
<td>Cap Colour</td>
<td>10</td>
<td>brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow</td>
</tr>
<tr>
<td>Bruises</td>
<td>2</td>
<td>bruised, no</td>
</tr>
<tr>
<td>Odour</td>
<td>9</td>
<td>almond, anise, creosote, fishy, foul, musty, none, pungent, spicy</td>
</tr>
<tr>
<td>Gill Attachment</td>
<td>4</td>
<td>attached, descending, free, notched</td>
</tr>
<tr>
<td>Gill Spacing</td>
<td>3</td>
<td>close, crowded, distant</td>
</tr>
<tr>
<td>Gill Size</td>
<td>2</td>
<td>broad, narrow</td>
</tr>
<tr>
<td>Gill Colour</td>
<td>12</td>
<td>black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow</td>
</tr>
<tr>
<td>Stalk Shape</td>
<td>2</td>
<td>enlarging, tapering</td>
</tr>
<tr>
<td>Stalk Root</td>
<td>7</td>
<td>bulbous, club, cup, equal, rhizomorphs, rooted, missing</td>
</tr>
<tr>
<td>Surface Above Ring</td>
<td>4</td>
<td>fibrous, scaly, silky, smooth</td>
</tr>
<tr>
<td>Surface Below Ring</td>
<td>4</td>
<td>fibrous, scaly, silky, smooth</td>
</tr>
<tr>
<td>Colour Above Ring</td>
<td>9</td>
<td>brown, buff, cinnamon, gray, orange, pink, red, white, yellow</td>
</tr>
<tr>
<td>Colour Below Ring</td>
<td>9</td>
<td>brown, buff, cinnamon, gray, orange, pink, red, white, yellow</td>
</tr>
<tr>
<td>Veil Type</td>
<td>2</td>
<td>partial, universal</td>
</tr>
<tr>
<td>Veil Colour</td>
<td>4</td>
<td>brown, orange, white, yellow</td>
</tr>
<tr>
<td>Ring Number</td>
<td>3</td>
<td>none, one, two</td>
</tr>
<tr>
<td>Ring Type</td>
<td>8</td>
<td>cobwebby, evanescent, flaring, large, none, pendant, sheathing, zone</td>
</tr>
<tr>
<td>Spore Print Colour</td>
<td>9</td>
<td>black, brown, buff, chocolate, green, orange, purple, white, yellow</td>
</tr>
<tr>
<td>Population</td>
<td>6</td>
<td>abundant, clustered, numerous, scattered, several, solitary</td>
</tr>
<tr>
<td>Habitat</td>
<td>7</td>
<td>grasses, leaves, meadows, paths, urban, waste, woods</td>
</tr>
</tbody>
</table>

The Mushroom Classification problem is based on a data set of 8124 instances which includes descriptions of hypothetical samples.
corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each sample is profiled using 22 attributes (Table A14) and each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. In the given data set, this latter class has been combined with the poisonous one yielding a class distribution of 4208 (51.8%) "edible" cases and 3916 (48.2%) "poisonous" cases.

Within the data set there are 2480 missing attribute values (denoted by "?") all for the attribute labelled as "stalk root".
APPENDIX 4

EXPERIMENTAL RESULTS

(1) RULEX (Andrews and Geva [1,2])

<table>
<thead>
<tr>
<th></th>
<th>Monk 1</th>
<th>Monk 2</th>
<th>Monk 3</th>
<th>Mushroom</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Of Rules</td>
<td>4</td>
<td>15</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>No. Of Antecedents</td>
<td>7</td>
<td>90</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Av. No. Of Antecedents Per Rule</td>
<td>1.75</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>98.53</td>
</tr>
<tr>
<td>Fidelity (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Consistency</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

(2) LAP (Hayward et al. [44])

<table>
<thead>
<tr>
<th></th>
<th>Monk 1</th>
<th>Monk 2</th>
<th>Monk 3</th>
<th>Mushroom</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Of Rules</td>
<td>13</td>
<td>42</td>
<td>1</td>
<td>n/a</td>
</tr>
<tr>
<td>No. Of Antecedents</td>
<td>50</td>
<td>381</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>Av. No. Of Antecedents Per Rule</td>
<td>3.8</td>
<td>9.1</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>95.0</td>
<td>-</td>
</tr>
<tr>
<td>Fidelity (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>Consistency</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
</tbody>
</table>
EXPERIMENTAL RESULTS (Cont'd)

(3) RULENEG (Hayward et al.[30,45])

<table>
<thead>
<tr>
<th></th>
<th>Monk 1</th>
<th>Monk 2</th>
<th>Monk 3</th>
<th>Mushroom</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Of Rules</td>
<td>13</td>
<td>64</td>
<td>8</td>
<td>300</td>
</tr>
<tr>
<td>No. Of Antecedents</td>
<td>43</td>
<td>352</td>
<td>16</td>
<td>8087</td>
</tr>
<tr>
<td>Av. No. Of Antecedents Per Rule</td>
<td>3.3</td>
<td>5.5</td>
<td>2</td>
<td>12.1</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>91.0</td>
<td>56.0</td>
<td>97.0</td>
<td>91.0</td>
</tr>
<tr>
<td>Fidelity (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Consistency</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

(4) DEDEC (Tickle et al. [39])

<table>
<thead>
<tr>
<th></th>
<th>Monk 1</th>
<th>Monk 2</th>
<th>Monk 3</th>
<th>Mushroom</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Of Rules</td>
<td>4</td>
<td>142</td>
<td>7</td>
<td>26</td>
</tr>
<tr>
<td>No. Of Antecedents</td>
<td>7</td>
<td>852</td>
<td>14</td>
<td>26</td>
</tr>
<tr>
<td>Av. No. Of Antecedents Per Rule</td>
<td>1.75</td>
<td>6.0</td>
<td>2.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>99.8</td>
</tr>
<tr>
<td>Fidelity (%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Consistency</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### EXPERIMENTAL RESULTS (Cont'd)

#### (5) TREX (Geva [46])

<table>
<thead>
<tr>
<th></th>
<th>Monk 1</th>
<th>Monk 2</th>
<th>Monk 3</th>
<th>Mushroom</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No. Of Rules</strong></td>
<td>4</td>
<td>n/a</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td><strong>No. Of Antecedents</strong></td>
<td>7</td>
<td>n/a</td>
<td>n/a</td>
<td>13</td>
</tr>
<tr>
<td><strong>Av. No. Of Antecedents Per Rule</strong></td>
<td>1.75</td>
<td>n/a</td>
<td>n/a</td>
<td>4.3</td>
</tr>
<tr>
<td><strong>Accuracy (%)</strong></td>
<td>100</td>
<td>85.0</td>
<td>95.4</td>
<td>100</td>
</tr>
<tr>
<td><strong>Fidelity (%)</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### (6) RULE-EXTRACTION-AS-LEARNING (Craven and Shavlik [5])

<table>
<thead>
<tr>
<th></th>
<th>Monk 1</th>
<th>Monk 2</th>
<th>Monk 3</th>
<th>Mushroom</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No. Of Rules</strong></td>
<td>16</td>
<td>56</td>
<td>8</td>
<td>155</td>
</tr>
<tr>
<td><strong>No. Of Antecedents</strong></td>
<td>128</td>
<td>644</td>
<td>35</td>
<td>6603</td>
</tr>
<tr>
<td><strong>Av. No. Of Antecedents Per Rule</strong></td>
<td>8</td>
<td>11.5</td>
<td>4.3</td>
<td>42.6</td>
</tr>
<tr>
<td><strong>Accuracy (%)</strong></td>
<td>77.0</td>
<td>62.0</td>
<td>97.0</td>
<td>98.0</td>
</tr>
<tr>
<td><strong>Fidelity (%)</strong></td>
<td>100</td>
<td>100</td>
<td>91.0</td>
<td>100</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
### EXPERIMENTAL RESULTS (Cont'd)

#### (7) C4.5 (Quinlan [47])

<table>
<thead>
<tr>
<th></th>
<th>Monk 1</th>
<th>Monk 2</th>
<th>Monk 3</th>
<th>Mushroom</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Of Rules</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>No. Of Antecedents</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Av. No. Of Antecedents Per Rule</td>
<td>1.75</td>
<td>4.0</td>
<td>2.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>100</td>
<td>64.8</td>
<td>100</td>
<td>99.8</td>
</tr>
<tr>
<td>Fidelity (%)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Consistency</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### (8) T2 (Holte [48])

<table>
<thead>
<tr>
<th></th>
<th>Monk 1</th>
<th>Monk 2</th>
<th>Monk 3</th>
<th>Mushroom</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Of Rules</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>No. Of Antecedents</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Av. No. Of Antecedents Per Rule</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>83.3</td>
<td>60.6</td>
<td>97.2</td>
<td>99.4</td>
</tr>
<tr>
<td>Fidelity (%)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Consistency</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Monk 1</td>
<td>Monk 2</td>
<td>Monk 3</td>
<td>Mushroom</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>----------</td>
</tr>
<tr>
<td>No. Of Rules</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. Of Antecedents</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Av. No. Of Antecedents Per Rule</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fidelity (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consistency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>