Survey and critique of techniques for extracting rules from trained artificial neural networks

Robert Andrews, Joachim Diederich and Alan B Tickle

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. This survey gives an overview of techniques 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.

Keywords: fuzzy neural networks, rule extraction, rule refinement, knowledge insertion, inferencing

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 process for 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 [1].

However, the success of the ANN paradigm is achieved at a cost. This is an inherent inability to explain in a comprehensible form the process through which a given decision or output generated by an ANN has been reached. For artificial neural networks to gain an 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 should become 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 *safety critical* applications such as airlines and power stations. In these cases, it is imperative that a system user should 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

Neurocomputing Research Centre, Queensland University of Technology, Box 2434 GPO, Brisbane 4001, Queensland, Australia
Paper received 15 November 1994. Revised paper received 27 January 1995. Accepted 22 February 1995

provides a vehicle for traversing the boundary between the connectionist and symbolic approaches.

This paper is 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.

For the purposes of this discussion, the focus will be restricted to rule extraction from general feed-forward multilayered 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 [2] and Cascade, developed by Fahlman and Lebiere [3], 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 in the development of 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 [4,5,6]. In this area of research the focus is on having an ANN simulate a deterministic finite state automation (FSA) which recognises regular grammars. The second is the work of Gallant [7] on extracting explanations of individual problem instances from trained artificial neural networks, as distinct from complete sets of rules.

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 of including rule extraction techniques as an adjunct to conventional artificial neural network techniques include the following.

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, e.g. a trace of rule firings, a proof structure etc., which can be used to answer *how* questions. Further, Gallant [7] 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 meaningfulness and coherence) is absolutely crucial for the user acceptance of such systems [8]. 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 realisations 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 [9,10,11]. 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 (e.g. calculations), they may include repetitions (e.g. if an inference has been made more than once), and the granularity of the explanation is often inappropriate [9]. Perhaps one clear lesson learned from 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 rule traces from a poorly organised rule base with perhaps hundreds of premises per rule cannot 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 [11]. 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 [11], there is little doubt that current systems are still too inflexible, unresponsive, incoherent, insensitive and rigid [12].

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 of 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 accessible, and to be able to be interpreted unambiguously. The 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 suboptimal.

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, and medical surgery. 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 a trained artificial neural network. i.e. they do not prove that a network behaves according to some specification. However, rule extraction algorithms provide a mechanism for either partially or completely *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 dataset 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 being used. 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 datasets. As Craven and Shavlik [13] observe, '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 nonlinear relationships, these relationships are encoded incomprehensibly as weight vectors within the trained ANN, and hence cannot easily support 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 *knowledge acquisition* problem for symbolic AI systems [14,15]. Further, as Sestito and Dillon [16, p 156] observe, the most difficult, time-consuming, and expensive task in building an expert system is that of 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 [7]. An extension of these ideas is to use trained artificial neural networks as a vehicle for synthetising the knowledge that is crucial for the success of knowledge-based systems. Alternatively, domain knowledge which is acquired using a knowledge engineering process may be used to constrain the size of the space searched during the learning phase and hence contribute to improve 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.* [17], which show that trained artificial neural networks can outperform symbolic machine learning methods in certain problem domains. The second are techniques for extracting symbolic rules from trained artificial neural networks which can 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 [18]. 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 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: (a) train the network on the available dataset(s), and (b) 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.

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 the rudimentary classification scheme used by Craven and Shavlik [13], 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 takes into consideration (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, namely the rules themselves. To date, one line in the development of rule extraction techniques has been directed towards presenting the output as a set of rules expressed using conventional (i.e. 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, e.g. *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 are in the form of (a) propositional/Boolean logic (i.e. *if...then...else*), or (b) nonconventional logic (i.e. the resulting representation can include probabilistic or fuzzy logic rules which may also incorporate an *if...then...else* structure). While it is not deemed necessary at this stage, at some future point, it may be expedient to introduce a third category to cater for extracted rules represented in first-order logic form, i.e. rules with quantifiers and variables.

The second proposed dimension to the classification scheme extends the classification schema used by Craven and Shavlik [13]. 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, namely the *decompositional* and *pedagogical* techniques, and a third, called *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, i.e. a set of incoming links whose summed weights guarantee the unit's bias is exceeded regardless of the activation value of 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.

In this translucency dimension, the pedagogical category comprises those rule extraction techniques which treat the trained ANN as a *black box*, i.e. 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' [13]. Hence the pedagogical techniques aim to extract rules that map inputs directly into outputs. Such techniques are typically used in conjunction with a symbolic learning algorithm, and the basic idea 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 comprises 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 aspect of this classification dimension is that of 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 to define some measure of how well the task of extracting the required *explanation* has been performed [2,18]. Criteria for evaluating the *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 previously unseen examples. 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. 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.* [18], 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. On the basis of the work of Weiss *et al.* [19], 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 that of 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 proposed dimensions in the classification scheme, the first three, (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, 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, it is proposed to use the first two dimensions of the aforementioned taxonomy (namely 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 the appendix.

RULE EXTRACTION TECHNIQUES

Boolean rule extraction using decompositional approaches

Artificial neural networks utilising standard *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 the decompositional approach, i.e. it focused on searching for and extracting conventional Boolean rules at the level of the individual (hidden and output) units within the trained ANN [20,21]. Of particular interest are two approaches [2,21] in which the basic idea 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 *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 subset algorithm as reported by Towell and Shavlik [2] is given as follows:

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 [21,22] (see the appendix). 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 of *if* $0 \leqslant$ *output* $\leqslant$ *threshold_1* $\Rightarrow$ *no*, and *if* *threshold _2* $\leqslant$ *output* $\leqslant 1 \Rightarrow$ *yes*, where *threshold _1* $<$ *threshold _2*. A more recent example of this line of approach is the subset algorithm developed by Towell and Shavlik [2]. In their implementation, the artificial neural network is constructed in such a way that the computed value of the activation function in each hidden and output unit is either *near* a value of 1 (i.e. *maximally* active), or *near* a value of 0 (i.e. *inactive*). Hence, 'links carry a signal equal to their weight or no signal at all' [23].

Fu [21,22] reported initial success in applying the KT algorithm to the problem domain of wind shear detection by infrared sensors. Similarly, Towell and Shavlik [2] 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 [2], i.e. 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 [21], 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 and the *connectionist scientist game* of McMillan *et al.* [24] is one of the earliest examples in which a specialised ANN training regime incorporating the decompositional approach is used as the basis for extracting Boolean rules (see the appendix). 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.* [25], and it 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 [2,16].

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 [2] (see the appendix). The *M-of-N* concept is a means of expressing rules in the form:

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 the *M*-of-*N* algorithm are as follows:

(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 the biases of all hidden and output units using the back propagation algorithm.

(5) Form a single rule for each hidden and output unit. The rule consists of a threshold given by the biases and weighted antecedents specified by the remaining links.

(6) Where possible, simplify rules to eliminate superfluous weights and thresholds.

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 to 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 [13] (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. These are (a) the requirement for either a rule set to initialise the ANN [2] 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 the *M*-of-*N* approach is essentially a *rule refinement system*, this may be true in general, and, in fact, Towell and Shavlik [2] 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, namely the *promoter recognition* problem and the *splice-junction determination* problem. Towell and Shavlik [2] 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.* [26] (see the appendix). 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, MA, USA, neighbourhood as a function of 13 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 following 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 approximately $14 000.

Another significant development in this area is the *RULEX* technique of Andrews and Geva [23,27] (see the appendix). The RULEX technique is designed to exploit the manner of construction and consequent behaviour of a particular type of multilayer perceptron, the constrained error back-propagation (CEBP) MLP, which is a representative of a class of local response ANNs that performs function approximation and classification in a manner similar to the way in which radial basis function (RBF) networks do. The hidden units of the CEBP network are sigmoid-based locally responsive units (LRUs) 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 below which can be readily extracted from the LRUs:

IF $ridge_1$ is active and
$ridge_2$ is active and
...
$ridge_N$ is active
THEN the pattern belongs to the 'target class'.

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 the KT and subset algorithms 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 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 [14]. 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 (i.e. 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 for a relatively simple problem domain was exceedingly large. This result highlights one of the major concerns with rule extraction techniques, namely that the end product is explanation and not obfuscation.

The VI analysis (VIA) technique developed by Thrun [1] is also the epitome of a pedagogical approach in that it extracts rules that map inputs directly into outputs

(see the appendix). 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 of the VIA algorithm are as follows:

(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 which can satisfy the constraints imposed by the initial validity intervals.)

Thrun [1] 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 examines 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 the 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 (a) the XOR problem, (b) the 'three monks' problem(s), and (c) a robot arm kinematics (i.e. continuously valued domain) problem. While the VIA technique does not appear to be limited to any specific class of problem domains Thrun [1] reports that the VIA technique 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 [13] is another significant development in rule extraction techniques utilising the pedagogical approach (see the appendix). A salient characteristic of this technique is that, depending on the particular implementation used, the rule-extraction-as-learning approach can be classified as either a pedagogical approach or a decompositional approach. The key is in the procedure used to establish whether a given rule agrees with the network. This procedure accepts a class label c and a rule r, returns *true* if all instances covered by r are classified as members of class c by the network. If, for example, Thrun's VIA algorithm [1] (as discussed previously) is used for this procedure then the approach is pedagogical, whereas if an implementation such as that of Fu [21] 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: (a) estimating if the extracted rule set is a sufficiently accurate model of the ANN from which the rules have been extracted, or (b) terminating after a certain number of iterations have resulted in no new rules (i.e. a *patience* criterion). The authors report 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 with the trained artificial neural network from which the rules were extracted (the fidelity of a rule set is the fraction of examples for which the rule set agrees with the trained artificial neural network)).

Another recent example of a pedagogical approach is *RULENEG*, developed by Pop *et al.* [28], which focuses solely on the task of extracting conjunctive rules. The genesis of the 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 following RULENEG algorithm.

```
Initialise the rule holder to empty
 For every pattern s from the training set
  Find the class C for s by use of the ANN
                              /* C = ANN(s)*/
  If s is not classified by the existing rules
    Initialise a new rule r for class C
    For every input i into the network
      Make a copy ś of s
      Negate the ith entry in ś
      Find the class Ć for ś by use of the ANN
                              /* Ć = ANN(ś)*/
      If C is not equal to Ć
        Add ith input and its truth value to r
    /* End for every input */
    Add r to the rule holder
/* End for every pattern */
```

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

The *BRAINNE* system of Sestito and Dillon [14,15,29] 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 tacking 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 pairwise 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 (i.e. an attribute from the problem domain) to the output. A major innovation in the BRAINNE technique is the ability 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 [30]. 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 (i.e. attributes) to the underlying trained artificial neural network and observing the resultant output. This is the classical 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 magnitudes of the weight vectors in the trained artificial neural network to rank the ANN input units (i.e. the attributes in the problem domain) according to the relative shares of their contributions 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 (i.e. the rule antecedents) on the basis of 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 (i.e. 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 for both structured sample problem domains and real-world problem domains.

Extraction of fuzzy rules

In parallel with the development of techniques for extracting Boolean rules from trained artificial neural networks, corresponding techniques for extracting fuzzy rules have been synthetised, the so-called *neurofuzzy* systems. Analogously 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 (i.e. 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 [31] observes that the identification of the initial set of fuzzy inference rules to be modelled has proven to be a difficult task, as have attempts at simultaneously undertaking the tasks of rule identification and membership tuning.

Some of the earliest work in this area was that of Masuoka *et al.* [32], 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 (i.e. the input variables). The fuzzy operations on the input variables (e.g. *and, or* etc.) are represented by a second distinct phase labelled 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 at less than a threshold value.

In a similar vein, Berenji [33] 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.* [31] developed three types of fuzzy neural network which can automatically identify the underlying fuzzy rules and tune the corresponding membership functions by modifying the connection weights of the ANNs using the back propagation 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.* [34] 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 [21] and the *fuzzy-MLP* model of Mitra [35] 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 have 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) in comparison with the former. In part, this improvement is attributable to the more complex ANN architecture used in the fuzzy-MLP procedure, which has three hidden layers, as opposed to one hidden layer in FNES. (The FNES architecture also includes direct connections between the input and output layers.)

Okada *et al.* [36] 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 the prediction accuracy of the model in comparison with a conventional three layer neural network in the application problem domain of financial bond rating.

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 the performance of ANN/rule-extraction techniques *vis-à-vis* other induction techniques for extracting rules from data applies in all problem domains. Hence, a pressing requirement is for a set of comparative benchmark results across a range of problem domains similar to that of the original *three monks* problem proposed by Thrun *et al.* [17].

A related issue is that, in an increasing number of applications, there are reports of situations in which the extracted rule set has had a better generalisation performance than the trained artificial neural network from which the rule set was extracted [2]. Similar observations have also been made in the area of extracting symbolic grammatical rules from recurrent artificial neural networks [4,5,37]. However, Giles and Omlin [4] 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 (i.e. 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 [2], and also as to whether the tasks of rule insertion and rule extraction could occur during ANN training [18].

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.

CONCLUSIONS

This survey provides an overview of the most important features of the set 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 has been introduced, and the most important evaluation criteria for rule-extraction techniques have been outlined.

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, namely a comprehensible explanation of how and why the trained ANN arrived at a given result or conclusion.These demands appear to fall broadly into two groups: (a) 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 (b) a *green-fields* situation where a user has a dataset from a problem domain and is interested in what relationships exist 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 appears that, notwithstanding the initial success of decompositional approaches such as that of the KT algorithm of Fu [21,22], 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, the KBANN/*M*-of-*N* approach, the BRAINNE technique, the RULEX algorithm etc. However, it is also clear that no single rule-extraction/rule-refinement technique or method is currently in a dominant position to the exclusion of all others.

ACKNOWLEDGEMENTS

The authors would like to express their thanks to Chris Ho-Stuart and Mostefa Golea for their valuable discussions about and insights into the key problem of determining the complexity of the rule-extraction problem.

The authors would also like to express their thanks to Mark Craven and two anonymous referees of this paper for their many helpful and incisive comments.

REFERENCES

[1] Thrun, S. B. Extracting provably correct rules from artificial neural networks, Technical Report IAI-TR-93-5, Institut for Informatik III Universitat Bonn, Germany (1994).

[2] Towell, G. and Shavlik, J. The extraction of refined rules from knowledge based neural networks, Machine Learning, Vol 131 (1993) pp 71–101.

[3] Fahlman, S. and Lebiere, C. The cascade-correlation learning architecture, in Lippman, R., Moody, J. and Touretzky, D. (Eds.) Advances in Neural Information Processing Systems – Vol 3: San Mateo (1991) pp 190–196.

[4] Giles, C. L. and Omlin, C. W. Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent networks, Connection Science, Vol 5, Nos 3 and 4 (1993) pp 307–328.

[5] Giles, C. L., Miller, C. B., Chen, D., Chen, H., Sun, G. Z. and Lee, Y. C. Learning and extracting finite state automata with second-order recurrent neural networks, Neural Computation, Vol 4 (1992) pp 393–405.

[6] Watrous, R. L. and Kuhn, G. M. Induction of finite-state languages using second-order recurrent networks, Neural Computation, Vol 4 (1992) pp 406–414.

[7] Gallant, S. Connectionist expert systems, Communications of the ACM, Vol 31, No 2 (February 1988) pp 152–169.

[8] Davis, R., Buchanan, B. G. and Shortliffe, E. Production rules as a representation for a knowledge-based consultation program, Artificial Intelligence, Vol 8, No 1 (1977) pp 15–45.

[9] Gilbert, N. Explanation and dialogue, Knowledge Engineering Review, Vol 4, No 3 (1989) pp 235–247.

[10] Moore, J. D. A reactive approach to explanation in expert and advice-giving systems, PhD Thesis, University of California at Los Angeles, USA (1989).

[11] Moore, J. D. and Swartout, W. R. A reactive approach to explanation, *Proc. IJCAI-89* (1989) pp 1504–1510.

[12] Swartout, W. R. Lecture on explanation, GMD (September 1989).

[13] Craven, M. W. and Shavlik, J. W. Using sampling and queries to extract rules from trained neural networks, Machine Learning: Proceedings of the Eleventh International Conference, San Francisco, CA, USA (1994).

[14] Saito, K. and Nakano, R. Medical diagnostic expert system based on PDP model, Proc. IEEE International Conference on Neural Networks – Vol 1, San Diego, CA, USA (1988) pp 255–262.

[15] Sestito, S. and Dillon, T. The use of sub-symbolic methods for the automation of knowledge acquisition for expert systems, Proc. 11th International Conference on Expert Systems and their Applications, Avignon, France (July 1991) pp 317–328.

[16] Sestito, S. and Dillon, T. Automated Knowledge Acquisition: Prentice Hall, Australia (1994).

[17] Thrun, S. B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Dzeroski, S., Fahlman, S. E., Fisher, D., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R. S., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, K., Wenzel, W., Wnek, J. and Zhang, J. The MONK's problems: a performance comparison of different learning algorithms, CMU-CS-91-197, Carnegie Mellon University, USA (December 1991).

[18] Giles, C. L. and Omlin, C. W. Rule refinement with recurrent neural networks, Proc. IEEE International Conference on Neural Networks, San Francisco, CA, USA (March 1993) pp 801–806.

[19] Weiss, S. and Indurkhya, N. Optimised rule induction, IEEE Expert (December 1993) pp 61–69.

[20] Bochereau, L. and Bourgine, P. Extraction of semantic features and logical rules from a multilayer neural network, International Joint Conference on Neural Network – Vol 2, Washington DC, USA (1990) pp 579–582.

[21] Fu, L. M. Rule learning by searching on adapted nets, Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, CA, USA (1991) pp 590–595.

[22] Fu, L. M. Rule generation from neural networks, IEEE Transactions on Systems, Man, and Cybernetics, Vol 28, No 8 (1994) pp 1114–1124.

[23] Andrews, R. and Geva, S. Inserting and extracting knowledge from constrained error back propagation networks, Proc. 6th Australian Conference on Neural Networks, Sydney, NSW, Australia (1995).

[24] McMillan, C., Mozer, M. C. and Smolensky, P. The connectionist scientist game: rule extraction and refinement in a neural network, Proc. Thirteenth Annual Conference of the Cognitive Science Society, Hillsdale, NJ, USA (1991).

[25] Jacobs, R., Jordan, M., Nowlan, S. and Hinton, G. Adaptive mixtures of local experts, Neural Computation, Vol 3 (1991) pp 79–87.

[26] Tresp, V., Hollatz, J. and Ahmad, S. Network structuring and training using rule-based knowledge, Advances in Neural Information Processing Systems – Vol 5 (1993) pp 871–878.

[27] Andrews, R. and Geva, S. Rule extraction from a constrained error back propagation MLP, Proc. 5th Australian Conference on Neural Networks, Brisbane, Queensland, Australia (1994) pp 9–12.

[28] Pop, E., Hayward, R. and Diederich, J. RULENEG: extracting rules from a trained ANN by stepwise negation, QUT NRC (December 1994).

[29] Sestito, S. and Dillon, T. Automated knowledge acquisition of rules with continuously valued attributes, Proc. 12th International Conference on Expert Systems and their Applications, Avignon, France (May 1992) pp 645–656.

[30] Tickle, A. B., Orlowski, M. and Diederich, J. DEDEC: decision detection by rule extraction from neural networks, QUT NRC (September 1994).

[31] 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.

[32] Masuoka, R., Watanabe, N., Kawamura, A., Owada, Y. and

Asakawa, K. Neurofuzzy systems – fuzzy inference using a structured neural network, Proc. International Conference on Fuzzy Logic and Neural Networks, Iizuka, Japan (1990) pp 173–177.
[33] Berenji, H. R. Refinement of approximate reasoning-based controllers by reinforcement learning, Proc. the Eighth International Machine Learning Workshop, Evanston, IL, USA, (1991) pp 475–479.
[34] Halgamuge, S. K. and Glesner, M. Neural networks in designing fuzzy systems for real world applications, Fuzzy Sets and Systems, Vol 65, No 1 (July 1994) pp 1–12.
[35] Mitra, S. Fuzzy MLP based expert system for medical diagnosis, Fuzzy Sets and Systems, Vol 65, Nos 2/3 (August 1994) pp 285–296.
[36] Okada, H., Masuoka, R. and Kawamura, A. Knowledge based neural network – using fuzzy logic to initialise a multilayered neural network and interpret postlearning results, Fujitsu Scientific and Technical Journal, Vol 29, No 3 (1993) pp 217–226.
[37] Omlin, C. W., Giles, C. L. and Miller, C. B. Heuristics for the extraction of rules from discrete time recurrent neural networks, Proc. International Joint Conference on Neural Networks (IJCNN 92) – Vol 1, Baltimore, MD, USA (1992) p 33.

BIBLIOGRAPHY

Diederich, J. Explanation and artificial neural networks, International Journal of Man–Machine Studies, Vol 37 (1992) pp 335–357.
Fu, L. M. Knowledge-based connectionism for revising domain theories, IEEE Transactions on Systems, Man, and Cybernetics, Vol 23, No 1 (1993) pp 173–182.
Fu, L. M. Neural networks in computer intelligence: McGraw–Hill, USA (1994)
Hayashi, Y. A neural expert system with automated extraction of fuzzy if–then rules and its application to medical diagnosis, Proc. Advances in Neural Information Processing Systems – Vol 3, Denver, CO, USA (1990) pp 578–584.
McClelland, J. L. and Rumelhart, D. E. Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises: MIT Press (1988).
Shultz, T. R. and Elman, J. L. Analysing cross connected networks, in Cowan, J. D., Tesauro, G. and Alspector, J. (Eds.) Advances in Neural Information Processing Systems – Vol 6, pp 1117–1124.

APPENDIX: SURVEY DETAILS

RuleNet and connectionist scientist game technique

Bibliographic reference
McMillan, Mozer and Smolensky [24].

Classification
Technique: decompositional. 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 scientist game, which describes the training algorithm employed by RuleNet, is based on the *scientific method*, i.e. induce a hypothesis from observation, and iteratively test and refine the hypothesis until the hypothesis explains the observations. An outline of the (iterative) process is as follows. The process continues until the extracted rules adequately characterise the problem domain.

(1) Train RuleNet on a set of input/output examples. (This corresponds to the scientist developing intuitions about a problem domain.)

(2) Extract symbolic rules from the connection strengths in the network (development of an explicit hypothesis).

(3) Inject the rules back into the network and continuing training (testing the hypothesis).

RuleNet is a three 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 the rule to be learned, i.e. the 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 α in the input vector), and $output_\beta$ (the character in position β in the output vector). Rule extraction is achieved by decompiling the weight vector c_i to form the condition component of the rule, and decompiling 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 character alphabet using strings of length four characters. Four test problems were constructed using rule bases consisting of eight, three, three, and five rules.

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 nonessential weights in c_i and A_i to 0, and essential weights to 1.

Comments
The method is designed only for a 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.

Subset algorithm

Bibliographic references
Towell and Shavlik [2] (see comments below) and Fu [21,22].

Classification
Technique: decompositional. 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 0 or 1, i.e. *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 [2]. (Note that because the values of the activation functions are either near 0 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.* The basic algorithm as reported by Towell and Shavlik [2] is given in the section on rule-extraction techniques.

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, prokaryotic promoter recognition and 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 for each hidden and output unit, (b) some of the rules may be repetitive, and (c) the extracted rules may hide significant (e.g. *M*-of-*N*) structures.

Comments
Towell and Shavlik base their implementation of the subset algorithm on earlier work by Fu [21] and Saito and Nakano [14], but they stress that 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.

M-of-*N* technique

Bibliographic reference
Towell and Shavlik [2].

Classification
Technique: decompositional. Rule type: (*modified*) Boolean (see comments below).

Description
The phases of the *M*-of-*N* algorithm are given in the section on rule-extraction techniques.

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 domain. 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, prokaryotic promoter recognition and 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 [13]. Three indicative parameters, (a) the number u of units in the ANN, (b) the average number l of links received by a unit, and (c) the number n of training examples, are used in Towell and Shavlik's assessment of the complexity of the *M*-of-*N* algorithm as shown in *Table 1*.

Rule quality
Towell and Shavlik use two dimensions, namely (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 from their own algorithm and from the set of algorithms that 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 the subset algorithm. 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 crosssection of problem domains.

Table 1 Complexity of *M*-of-*N* algorithm

Step number	Name	Estimated complexity
1	Clustering	O ($u \times l^2$)
2	Averaging	O ($u \times l$)
3	Eliminating	O ($n \times u \times l$)
4	Optimising	Precise analysis is inhibited by the use of back propagation in this (bias-optimisation) phase
5	Extracting	O ($u \times l$)
6	Simplifying	O ($u \times l$)

Comments
The salient feature of the *M*-of-*N* technique is the explicit searching for rules of the 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*, i.e. symbolic rules ⇒ ANN representation ⇒ rule refinement ⇒ symbolic rules.)

Rule-extraction-as-learning technique

Bibliographic reference
Craven and Shavlik [13].

Classification
Technique: eclectic. 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 as follows.

```
/* Initialise rules for each class */

For each class c
  R_c := 0
Repeat
  e := examples ()
  c := classify (e)
  If e not covered by R_c then
    /* Learn a new rule */
    r := conjunctive rule formed from e
    For each antecedent r_i of r
      ŕ := r but with r_i dropped
      If subset (c, ŕ) = true then r := ŕ
    R_c := R_c ∨ 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 (a) the selection of members of the set used for training the artificial neural network, (b) random sampling, or (c) the random creation of examples of a specified class, as follows.

```
/* Create a random example */
For each feature e_i with possible values v_i1,...,v_in
e_i := randomly-select(v_i1,...,v_in)
Calculate the total input s to output unit (which has
  a threshold value θ)
If s ≥ θ then return e
Impose random order of all feature values
/* Consider the values in order */
For each value v_ij
  If changing feature e_i's value to v_ij increases s
    e_i := v_ij
  If s ≥ θ then return e
```

Craven and Shavlik use a function (this is also called the subset algorithm, but it differs from the one discussed previously in relation to [2,21,22]) to determine if the modified rule still agrees with the network, i.e. if all the 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 for 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, the number f of input features and the total number v of values for all input features, Craven and Shavlik estimate the complexity of their algorithm as shown in *Table 2*. The authors also note that the complexity of the examples function is O(f) when using a random sampling technique (i.e. the second option above). For the third option (i.e. the 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 algorithm [1] is used in the implementation.

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 with the trained artificial neural network from which the rules have been extracted. The fidelity of a rule set is the fraction of examples for which the rule set agrees with the trained artificial neural network.

Table 2 Estimated complexity of *rule-extraction-as-learning* technique

Operation number	Description	Estimated complexity
1	Calls to the *examples* oracle	$O(f)$
2	Calls to the *subset* oracle	$O(f)$
3	Subsumption-check comparisons with the individual terms of R_c, the disjunctive normal form expression for class c	$O(f)$

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.

VIA algorithm

Bibliographic reference
Thrun [1].

Classification
Technique: pedagogical. Rule type: propositional *if...then...else*.

Description
An outline of the VIA algorithm is given in the section on rule-extraction techniques.

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 the XOR problem, the (perennial) three monks problem(s), and a robot arm kinematics (i.e. continuously valued domain) problem. The VIA algorithm does not appear to be limited to any specific class of problem domains. However, Thrun [1] reports that the VIA algorithm 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 ability 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 VIA 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 it examines 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 it focuses 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.

RULEX technique

Bibliographic reference
Andrews and Geva [23,27].

Classification
Technique: decompositional. 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), the constrained error back-propagation MLP. This is a representative of a class of local response ANN that performs function approximation and classification in a manner similar to that in which radial basis function networks do. The hidden units of the CEBP network are sigmoid-based locally responsive units (LRUs) 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, there being 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 propositional rules derived from an N dimensional local bump using the RULEX technique are those given in the section on rule-extraction techniques.

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 following form:

IF $c_1 - b_1 + 2k_1^{-1} \leq x_1 \leq c_1 + b_1 - 2k_1^{-1}$
AND $c_2 - b_2 + 2k_2^{-1} \leq x_2 \leq c_2 + b_2 - 2k_2^{-1}$
$\vdots$
AND $c_N - b_N + 2k_N^{-1} \leq x_N \leq c_N + b_N - 2k_N^{-1}$
THEN the pattern belongs to the 'target class'

For discrete valued input, it is possible to enumerate the active range of each ridge as an ORed list of values that will activate the ridge. In this case, it is possible to state the rule associated with the LRU in the following form:

IF v_{1a} OR v_{1b} ... OR v_{1n}
AND v_{2a} OR v_{2b} ... OR v_{2n}
$\vdots$
AND v_{Na} OR v_{Nb} ... OR v_{Nn}
THEN the pattern belongs to the 'target class'

(where $v_{ia}, v_{ib}, \ldots, v_{in}$ are contiguous values in the ith input dimension, and $v_{ia} \geq c_i - b_i + 2k_i^{-1}$ and $v_{in} \leq c_i - b_i + 2k_i^{-1}$).

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

Problem domain/validation example
The authors used a number of examples to demonstrate the applicability of the RULEX technique including (a) the three monks problem, (b) the DNA prokaryotic promoter recognition problem, and (c) 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 the direct interpretation of weight parameters as rules.

Rule quality
The RULEX algorithm employs an incremental training scheme for the CEBP network results in a solution which has the minimum number of LRUs, and hence the extracted rule set is 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 $1 \ldots N$, where N is the dimensionality of the problem RULEX guarantees to include only those dimensions that are actually used in classifying patterns in the extracted rules, thus making individual rules easily comprehensible.

Network structuring and training using rule-based knowledge technique

Bibliographic reference
Tresp, Hollatz and Ahmad [26].

Classification
Technique: decompositional. 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 Gaussian functions. 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 the following:

- *forget*, where training data is used to adapt NN^{init} by gradient descent (i.e. the sooner training stops, the more initial knowledge is preserved),
- *freeze*, where the initial configuration is frozen (i.e. if a discrepancy between prediction and data occurs, a new basis function is added),
- *correct*, where a parameter is penalised if it deviates from its initial value,
- *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 as follows:

While error < threshold
  Either prune or remove the basis function which has the least importance to the network (remove the least significant rule)
  Or prune conjuncts by finding the Gaussian function with the largest radius, and setting this radius to infinity (effectively removing the associated input dimension from the basis function)
  Retrain the network till no further improvement in error
End while

Rule premises extracted by directly decompiling the Gaussian functions are stated in terms of conjunctions of (centre, radius) pairs of the Gaussian functions which comprise the basis functions $b_i(x)$. Rule conclusions are determined by evaluating the conclusion w_i.

Problem domain/validation example
Examples of applications were the approximation of a noisy sinusoid to demonstrate pruning (no rules extracted), a bicycle control problem (no results presented), and a Boston housing problem. Here, the task was to predict the price of a house in Boston given 13 potentially relevant features (a sample of the total extracted rules was provided).

Algorithm complexity
No details were given by the authors.

Rule quality
Insufficient detail was provided to adequately assess the quality of the rules extracted using this method.