Mapping rule-based systems into neural architecture

Li-Min Fu and Li-Chen Fu*

A novel approach has been developed that maps a rule-based expert system into the neural architecture in both the structural and behavioural aspects. Under this approach, the knowledge base and the inference engine are mapped into an entity called 'conceptualization', where a node represents a concept and a link represents a relation between two concepts. A concept node is designated by a small number of language symbols. In the neural system transformed from a knowledge-based system, the inference behaviour is characterized by propagating and combining activations recursively through the network, and the learning behaviour is based upon a mechanism called 'back-propagation', which allows proper modification of connection strengths in order to adapt the system to the environment. This approach is based on the analogies observed between a belief network and a neural network, and its validity has been demonstrated by experiments. Finally, the advantages and disadvantages of this approach are discussed with respect to inference and learning.

Keywords: expert systems, rule-based systems, neural architecture

The neural network approach, also referred to as connectionism, has been an increasingly important approach to artificial intelligence[1-4]. Under this approach, information processing occurs through interactions among a large number of simulated neurons, each of which is quite limited in its processing capabilities. The knowledge of a neural network (a *connectionist*) lies in its connections and associated weights.

It has long been argued that a close resemblance between the computer's internal representations and neural nets is neither necessary nor feasible. Neural networks implemented earlier are often severely limited in the kinds of computations they can perform[5]. However, with recent successes of the neural network approach to such problems as learning to speak[6], medical reasoning[7], and recognizing hand-written characters, there is a growing interest in taking this approach to artificial intelligence. These successess can be explained in part by the invention of a number of useful learning algorithms such as 'back-propagation', and in part by hardware advances in the construction of massively parallel computers that enable much faster simulation of neural networks[8]. Furthermore, researchers have begun to explore a new computer architecture called the *neural computer* (see, for example, Reference 9), which resembles biological brains in structure and behaviour. Such computers hold the promise of solving some hard problems faster than current computers by many orders of magnitude.

Since the early eighties, when several knowledge-based systems such as DENDRAL, PROSPECTOR and CADUCEUS proved to be successful, the knowledge-based approach has become the most important approach to artificial intelligence. As shown in Figure 1, knowledge base, inference engine and user interface are three main components in a knowledge-

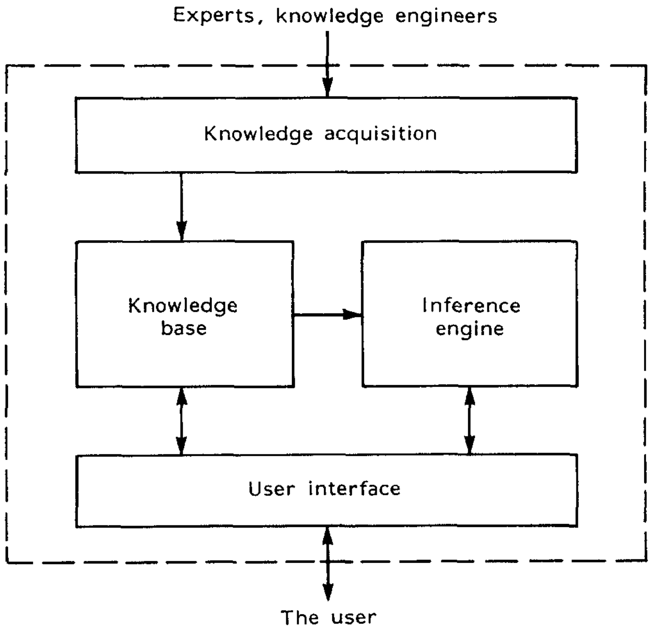

Figure 1. The basic components of a knowledge-based system

Department of Electrical Engineering and Computer Science, College of Engineering and Applied Science, The University of Wisconsin-Milwaukee, PO Box 784, Milwaukee, WI 53201, USA
* Department of Computer Science, The National Taiwan University, 1 Roosevelt Rd IV, Taipei, Taiwan

Paper received 17 February 1989. Revised paper received 30 August 1989. Accepted 13 September 1989

based system. Knowledge representation, knowledge processing (inference) and knowledge acquisition (learning) constitute three primary issues in building such systems.

While the neural network approach has produced encouraging results, particularly in low-level perceptual and signal processing tasks, it has had limited success in high-level cognitive areas where the knowledge-based approach has shown promise. On the other hand, the knowledge-based approach may be inadequate or inappropriate for performing perceptual reasoning. To combine these two approaches is an important direction for developing an artificial intelligence system in the future.

This paper describes a novel approach that maps rule-based systems into the neural architecture. Under this approach, the knowledge base and the inference engine are mapped into an entity called *conceptualization**, where a node represents a concept and a link represents a relation between two concepts. A concept node is designated by a small number of language symbols. In the neural system transformed from a knowledge-based system, the inference behaviour is characterized by propagating and combining activations recursively through the network, and the learning behaviour is based upon a mechanism called *back-propagation*, which allows proper modification of connection strengths in the adaptation to the environment.

MAPPING RULE-BASED SYSTEMS INTO NEURAL ARCHITECTURE

The neural network approach contrasts with the knowledge-based approach in several aspects. The knowledge of a neural network lies in its connections and associated weights, whereas the knowledge of a rule-based system lies in rules. A neural network processes information by propagating and combining activations through the network, but a knowledge-based system reasons through symbol generation and pattern matching. The knowledge-based approach emphasizes knowledge representation, reasoning strategies and the ability to explain, whereas the neural network approach does not. The key differences between these two approaches are summarized in Table 1.

A rule-based system (knowledge represented in rules) can be transformed into an inference network where each connection corresponds to a rule and each node corresponds to the premise or the conclusion of a rule, as seen in Figure 2. Reasoning in such systems is a process of propagating and combining multiple pieces of evidence through the inference network until final conclusions are reached. Uncertainty is often handled by adopting the certainty factor (CF) or the probabilistic schemes which associate each fact with a number called the *belief value*. An important part of reasoning tasks is to determine the belief values of the predefined final hypotheses given the belief values of observed evidence. The network of an inference system through which belief values of evidences or hypotheses are propagated and combined is called the *belief network*. Correspondence in structural and behavioural aspects exists between neural networks and belief networks, as shown in Table 2. For instance, the summation function in neural networks corresponds to the function for combining certainty factors in MYCIN-like systems or to the Bayesian formula for deriving *a posteriori* probabilities in PROSPECTOR-like systems. The thresholding function in neural networks corresponds to predicates such as SAME (in MYCIN-like systems), which cuts off any certainty value below 0.2.

Since belief networks correspond to neural networks in every structural and behavioural attribute shown in Table 2, any algorithm that is applicable to neural networks characterized by no more than these attributes may also be applicable to belief networks. 'Back-propagation' is just such an algorithm.

A rule-based system is mapped into a neural network by mapping the knowledge base and the inference engine into a kind of neural network called conceptualization, which stores knowledge and performs inference and learning. Furthermore, to construct a conceptualization, the following mappings need to be done.

- Final hypotheses are mapped into output neurons (neurons without connections pointing outwards).
- Data attributes are mapped into input neurons (neurons without connections pointing inwards).
- Concepts that summarize or categorize subsets of data or intermediate hypotheses that infer final hypotheses are mapped into middle (also known as hidden) neurons.
- The strength of a rule is mapped into the weight of the corresponding connection.

If there are no data errors, input neurons can represent both the observed and the actual data. In case of possible data errors, the observed data and the actual data are represented by two different levels of neurons, with a connection established between each observed and actual input neurons referring to the same data attribute. One example is shown in Figure 3 where, for instance, observed input neuron E_1' corresponds to actual input neuron E_1.

KNOWLEDGE REPRESENTATION

In this section the knowledge representation language in MYCIN[10] or similar systems is reviewed. The issue of how to map such language into a conceptualization is then examined, and knowledge representation for the neural network is described.

In MYCIN, facts are represented by context–attribution–value (or object–attribute–value) triples. Each triple is a term. For instance, the term 'throat which is the site of the culture' is represented by the triple ⟨CULTURE SITE THROAT⟩. Each triple is associated with a certainty factor, which is described later.

A sentence is represented by a predicate–context–attribute–value quadruple. For instance, the sentence

*'Conceptualization' is also a technical term in the artificial intelligence literature.

Table 1. Comparison between the neural network and the knowledge-based approaches

	Neural network approach	Knowledge-based approach
Knowledge	Connections	Rules
Computation	Numbers Summation and thresholding Simple, uniform	Numbers, symbols Pattern matching Complicated, various
Reasoning	Non-strategic	Strategic, meta-level
Tasks	Signal level	Knowledge level

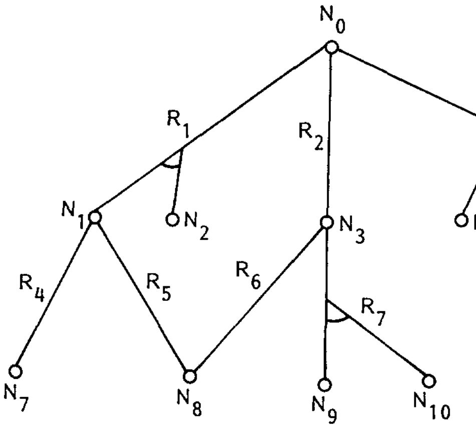

Figure 2. An inference network

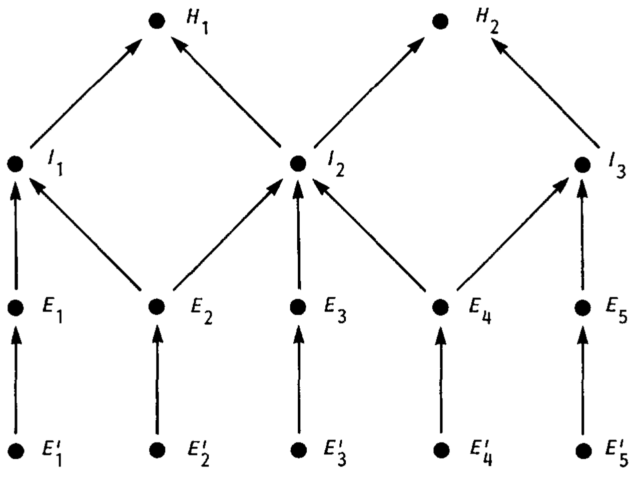

Figure 3. Organization of the knowledge base and input data as a neural network

'the site of the culture is throat' is represented by quadruple ⟨SAME CULTURE SITE THROAT⟩. The truth value of a sentence is determined by whether the triple satisfies the predicate in terms of its CF.

Judgemental and inferential knowledge is represented in production rules; i.e. if–then rules. If a rule's IF-part is evaluated to be true, its THEN part will be concluded. Each part is constituted by a small number of sentences. For instance, a MYCIN rule

Table 2. Correspondence between neural networks and belief networks

Neural networks	Belief networks
Connections	Rules
Nodse	Premises, conclusions
Weights	Rule strengths
Thresholds	Predicates
Summation	Combination of belief values
Propagation of activations	Propagation of belief values

- RULE124
  IF:
  1. The site of the culture is throat.
  2. The identity of the organism is Streptococcus.
  THEN: There is strongly suggestive evidence (.8) that the subtype of the organism is not group-D.

can be encoded in the MYCIN language as

- (RULE124 (($AND(SAME CULTURE SITE THROAT)
  (SAME ORGANISM IDENTITY STREPTOCOCCUS))
  ((CONCLUDE ORGANISM SUBTYPE GROUP-D −.8))))

Certainty factors are integers ranging from −1.0 to 1.0. A minus number indicates disbelief whereas a positive number indicates belief. The degree of belief or disbelief parallels the absolute value of the number. The extreme values −1.0 to 1.0 represent 'No' and 'Yes', respectively. A triple is associated with a CF indicating the current belief in the triple. A rule is assigned a CF representing the degree of belief in the conclusion given that the premise is true. For instance, the CF of RULE124 in the example above is −0.8. The CF of a conclusion based upon a rule can be computed by multiplying the CF of the premise and the CF of the rule. Each sentence (condition) in the premise on evaluation will return a number ranging from 0 to 1.0 representing the CF of the sentence. The CFs of all conditions in the premise are combined to result in the CF of the premise. As in the fuzzy set theory, $AND returns the minimum of the CFs of its arguments. CFs of a fact due to different pieces of evidence are combined according to certain formulae.

A sentence in the rule language is mapped into a concept node (a node in the conceptualization). Mapping at this level of abstraction can capture the analogies between a belief and a neural network shown in Table 2. Mappings at lower levels, such as mapping a word in a sentence into a concept node, lack a good justification.

Suppose the premise of a rule involves conjunction. Each sentence in the premise is mapped into a concept node. These concept nodes then lead into another concept node representing the conjunction.

The CF of a sentence is mapped into the activation level of the concept node designated by the sentence. The CF of a rule is mapped onto the weight of the connection between the two concept nodes, one designated by the premise and the other by the conclusion of the rule.

A neural network is a directed graph where each arc is labelled with a weight. Therefore, it is defined by a two-tuple (V, A), where V is a set of vertices and A is a set of arcs. The knowledge of a neural network is stored in its connections and weights. The data structure to represent a neural network should take into account how to use its knowledge. Here the scheme used to represent a neural network will be described.

Assume that the network is arranged as multiple layers. Each layer contains a certain number of nodes (processing elements). A node receives input from some other nodes which feed into the node. If node A leads into node B, we say that node A is adjacent to node B and node B is adjacent from node A. There is one list from each node in the network. The members in list i represent the nodes that are adjacent to node i. To make the access to these lists fast, all the nodes are stored in an array where each node points to the list associated with it, as shown in Figure 4. This scheme is known as 'inverse adjacency lists' in graph theory. Connection weights are stored in properly defined data fields in the adjacency lists. Since the activation level at a given node is computed based on the activations at the nodes adjacent to the node, inverse adjacency lists offer computational advantages. By contrast, the scheme of 'adjacency lists' which contain nodes adjacent from a given node is useful for back-propagation.

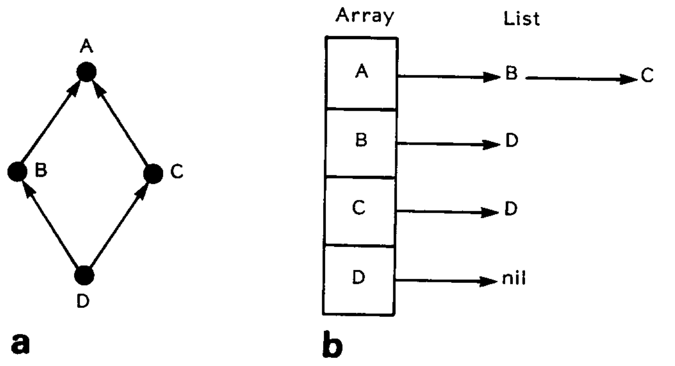

Figure 4. Representation of a neural network (a) as inverse adjacency lists (b)

INFERENCE

Inference in MYCIN or similar systems is to deduce the CFs of predefined hypotheses from given data. Such systems have been applied successfully to several types of problems such as diagnosis, analysis, interpretation and prediction. MYCIN uses a goal-oriented strategy to make inference. This means it invokes rules whose consequents deal with the given goal and recursively turns a goal into subgoals suggested by the rules' antecedents. By contrast, a system which adopts a data-driven strategy will select rules whose antecedents are matched by the database. Despite the difference in rule selection between these two strategies, inference in rule-based system is a process of propagating and combining CFs through the belief network. Since inference in the neural network involves a similar process, with CFs replaced by activation levels, the formulae for computing CFs can be applied to compute the activation level at each concept node in the conceptualization.

If a rule-based system involves circularity (cyclic reasoning), then inference in the neural networks mapped by such a system is characterized by not only propagation and combination of activations but also iterative search for a stable state. Starting with a noisy state, a network can reach a stable state, if it converges, in an extremely short period of time measured at the unit of the time constant of the neural circuit.

The inference capability of the neural network is derived from the collective behaviour of simple computational mechanisms at individual nodes. The output of a node is a function of the weighted sum of its inputs. In a biological neuron, if and only if its input exceeds a certain threshold, the neuron will fire. For an artificial neuron, continuous nonlinear transfer functions such as the sigmoid function and noncontinuous ones such as threshold logic have been defined. A neural network is often arranged as single-layered or multi-layered, and is organized as feedforward or with collateral or recurrent circuits. Different architectures are taken in accordance with the problem characteristics.

In a feedforward neural network, the inference behaviour is characterized by propagating and combining activations successively in the forward direction from input to output layers. Collateral inhibition and feedback mechanisms are implemented using collateral and recurrent circuits, respectively. They are employed for various purposes. For instance, the winner-take-all strategy can be implemented with collateral inhibition circuits. Feedback mechanisms are important in adaptation to the environment. As to the layered arrangement, multi-layered neural networks are more advantageous than single-layered networks in performing nonlinear classification. This advantage stems from the nonliner operation at the hidden nodes. For instance, exclusive-OR can be simulated by a bi-layered neural network but not by any single-layered one. The principle of maximum information preservation (*infomax principle*) has been proposed for information transmission from one layer to another in a neural network[11]. This principle can shed light on the design of a neural network for information processing.

The inference tasks performed by the neural network generally fall into four categories: pattern recognition, association, optimization and self-organization. A single-layered network can act as a linear discriminant, whereas a multi-layered network can be an arbitrary nonlinear discriminant. Association performed by the neural network is content-directed, allowing incomplete matching. Optimization problems can be solved by implementing cost functions as neural circuits and optimizing them. Self-organization is the way the neural network evolves unsupervisedly in response to environmental changes. Clustering algorithms can be implemented by neural networks with self-organization abilities.

MYCIN-like expert systems will be mapped into neural networks which are in general feedforward and multi-layered, and perform tasks close to pattern recognition. By capitalizing on all inference capabilities of the neural network, it is possible to develop expert systems more versatile than existing ones.

LEARNING

Learning in the conceptualization is the process of modifying connection weights in order to achieve correct inference behaviour. The following will show how to apply the back-propagation rule to learn and how to revise rules and/or data on the basis of the results through learning.

The learning problem

In a knowledge-based system, the issue of learning deals with acquiring new knowledge and maintaining integrity of the knowledge base. The knowledge base is constructed through a process called knowledge engineering (encoding of expert knowledge) or through machine learning.

When errors are observed in the conclusions made by a rule-based system, an issue is raised of how to identify and correct the rules or data responsible for these errors. The problem of indentifying the sources of errors is known as the *blame assignment* problem.

Previous approaches[12-15] only focus on how to revise the knowledge base. Among these, TEIRESIAS[12] is a typical work. It maintains the integrity of the knowledge base by interacting with experts. However, as the size of the knowledge base grows, it is no longer feasible for human experts to consider all possible interactions among knowledge in a coherent and consistent way. TMS[16] resolves inconsistency by altering a minimal set of beliefs, but it lacks the notion of uncertainty in the method itself. Symbolic machine learning techniques such as the RL program[17] can learn and debug knowledge but in general do not address the case when the knowledge involves intermediate concepts which are not used to describe the training samples.

TEIRESIAS may be confronted with the following problems. First, incorrect conclusions may be due to data errors. Second, experts know the strength of inference for each individual rule, but it may be difficult for them to determine the rule strengths in such a way that dependencies among rules are carefully considered in order to meet the system assumptions. For instance, in MYCIN, since certainty factors are combined under the assumption of independence, the certainty factors assigned to two dependent rules should be properly adjusted so as to meet this assumption. The approach presented here will address these problems.

Back-propagation of error

An error refers to the disagreement between the belief value generated by the system and that indicated by a knowledge source assumed to be correct (e.g., an expert) with respect to some fact. The back-propagation rule developed in the neural network approach[3] is a recursive heuristic which propagates backwards errors at a node to all nodes pointing to that node, and modifies the weights of connections leading into nodes with errors. First, we will restrict our attention to single-layered networks involving only input and output neurons.

In each inference task, the system arrives at the belief values of final hypotheses given those of input data. The belief values of input data form an input pattern (or an input vector) and those of final hypotheses form an output pattern (or an output vector). System error refers to the case when incorrect output patterns are generated by the system. When system error arises, we use the instance consisting of the input pattern given for inference and the correct output pattern to train the network. The instance is repeatedly used to train the network until a satisfactory performance is reached. Since the network may be incorrectly trained by that instance, we also maintain a set of reference instances to monitor the learning process. This reference set is consistent with the knowledge base. If, during learning, some instances in the reference set become inconsistent, they will be added to the learning process.

On a given trial, the network generates an output vector given the input vector of the training instance. The discrepancy obtained by subtracting the network's vector from the desired output vector serves as the basis for adjusting the strengths of the connections involved. The back-propagation rule adapted from Reference 3 is formulated as follows:

$$\Delta W_{ji} = rD_j(\mathrm{d}O_j/\mathrm{d}W_{ji}) \tag{1}$$

where $D_j = T_j - O_j$, ΔW_{ji} is the weight (strength) adjustment of the connection from input node i to output node j, r is a trial-independent learning rate, D_j is the discrepancy between the desired belief value (T_j) and the network's belief value (O_j) at node j, and the term $\mathrm{d}O_j/\mathrm{d}W_{ji}$ is the derivative of O_j with respect to W_{ji}. According to this rule, the magnitude of weight adjustment is proportional to the product of the discrepancy and the derivative above.

The back-propagation rule is applicable to belief networks where the propagation and the combination of belief values are determined by differentiable mathematical functions. As shown in Equation (1), the mathematical requirement for applying the back-propagation rule is that the relation between the output activation (O_j) and the input weight (W_{ji}) is determined by a differentiable function. In belief networks, this relation is differentiable if the propagation and the combination functions are differentiable. Since combining belief values in most rule-based systems involves such logic operations as conjunction or disjunction, the back-propagation rule is applied after turning the conjunction operator into multiplication and the disjunction operator into summation.

A multi-layered network involves at least three levels: one level of input nodes, one level of output nodes and one or more levels of middle nodes. Learning in a multi-layered network is more difficult because the behaviour of middle nodes is not directly observable. Modifying the strengths of the connections pointing to a middle node entails the knowledge of the discrepancy between the network's value and the desired belief value at the middle node. The discrepancy at a middle node can be derived from the discrepancies at output nodes which receive activations from the middle node[3]. It can be shown that the discrepancy at middle node j is defined by

$$D_j = \sum_k (\mathrm{d}O_k \mathrm{d}O_j) D_k$$

where D_k is the discrepancy at node k. In the summation, each discrepancy D_k is weighted by the strength of the connection pointing from middle node j to node k. This is a recursive definition in which the discrepancy at a middle node is always derived from discrepancies at nodes at the next higher level.

In addition, the belief value of a middle node can be obtained by propagating the belief values at input nodes recursively and combining these values properly until the middle node is reached.

Distinguishing knowledge base from input data errors

A method has been devised that can distinguish knowledge base errors from input data errors. This method includes three tests. In the first test, we clamp all connections corresponding to the knowledge base so that only the strengths of the connections between the observed and the actual input data nodes remain adjustable during learning. In the second test, we clamp the connections between the observed and the actual inputs and allow only the strengths of the connections corresponding to the knowledge base to be modified. In the third test, we allow the strengths of all connections to be adjusted. In each test, success is reported if the error concerned can be resolved after learning; failure is reported otherwise. Consequently, there are eight possible outcomes combined from the results of these three tests, as shown in Table 3. These results are interpreted as follows. Outcome $O1$ suggests the revision of either the knowledge base or input data. In this case, an expert's opinion is needed to decide which should be revised. Outcome $O2$ is unlikely (in the experience of the authors) and is ignored. Outcome $O3$ suggests the revision of input data. Outcome $O4$ is unlikely and is ignored. Outcome $O5$ suggests the revision of the knowledge base. Outcome $O6$ is also unlikely and is ignored. Outcome $O7$ suggests the revision of both the knowledge base and input data. Outcome $O8$ is a deadlock which demands an expert to resolve.

Table 3. Distinguishing knowledge-base errors from data errors

Test 1	Test 2	Test 3	Test 4
S	S	S	$O1$
S	S	F	$O2$
S	F	S	$O3$
S	F	F	$O4$
F	S	S	$O5$
F	S	F	$O6$
F	F	S	$O7$
F	F	F	$O8$

S = success, F = failure

Revision operations

The results of the above tests will indicate whether the knowledge base or input data (or both) should be revised. The strengths of the connections in the network (representing the knowledge base and input data) have been revised after learning. The next question is how to revise the knowledge base and/or input data according to the revisions made in the network. The revision of the knowledge base will be dealt with first.

Basically, there are five operators for rule revision[12]:

- modification of strengths,
- deletion,
- generalization,
- specialization, and
- creation.

However, not all the five operators are suitable in the neural network approach to editing rules. Each operator is examined below.

The *modification of strengths* operator is straightforward since the strength of a rule is just a copy of the weight of the corresponding connection and the weights of connections have been modified after learning with the back-propagation rule. If the weight change is trivial, we just keep the rule strength before learning.

The *deletion* operator is justified by Theorem 1.

Theorem 1. In a rule-based system, if the following conditions are met:

1 the belief value of the conclusion is determined by the product of the belief value of the premise and the rule strength,
2 the absolute value of any belief value and the rule strength is not greater than 1,
3 any belief value is rounded off to zero if its absolute value is below threshold k (k is a real number between 0 and 1),

then the deletion of rules with strengths below k will not affect the belief values of the conclusions arrived at by the system.

Proof. From conditions 1 and 2, if the strength of rule R is below k, the belief value of its conclusion is always below k. From condition 3, the belief value of the conclusion made by rule R will always be rounded off to zero. Since rule R is not effective in making any conclusion, it can be deleted. Thus, the deletion of such rules as rule R will not affect the system conclusions.$\square$

Accordingly, deletion of a rule is indicated when its absolute strength is below the predetermined threshold. In MYCIN-like systems, the threshold is 0.2.

The deletion operator is also justified by the following argument. Suppose we add some connections to a neural network that has already reached an equilibrium and assign weights to these added connections in such a way that incorrect output vectors are generated. Thus, these conditions are semantically inconsistent. Then, if we train the network with correct samples, the weights of the added connections will be modified in the direction of minmizing their effect. What happens is that the weights will go towards zero and even cross zero during training. In practice, we set a threshold so that when the shift towards zero for a connection weight is greater than this threshold, we delete the connection.

Generalization of a rule can be done by removing some conditions from its premise, whereas *specialization* can be done by adding more conditions to the premise. If the desired belief value of a conclusion is always higher than that generated by the network and the discrepancy resists decline during learning, it is suggested that rules supporting this conclusion be generalized. Or on the other hand, if the discrepancy is negative and resistant, specialization is suggested. However, generalization or specialization of a rule may involve qualitative changes of nodes. The back-propagation rule has not yet been powerful enough to make this kind of change except deletion of conditions for generalization.

Creation of new rules involves establishment of new connections. Whereas we delete a rule if its absolute strength is below a threshold, we may establish a new connection when its absolute strength is above the threshold. To create new rules we need to create some additional connections which have the potential to become rules. Without any bias, one may need an inference network where all data are fully connected to all intermediate hypotheses, which in turn are fully connected to all final hypotheses. This is not a feasible approach unless the system is small.

From the above analyses, we allow only the modification of strengths and deletion operators in the neural network approach to rule revision.

Revision of input data is much simpler. If the weight of the connection between an observed and an actual input node after learning is below a predetermined threshold, or the shift towards zero is above a certain value, the corresponding input data attribute is treated as false and deleted accordingly.

It has been known that noise associated with training instances will affect the quality of learning. In the neural network approach, since noise will be distributed over the network, its effect on individual connections is relatively minor. In practice, perfect training instances are neither feasible nor necessary. As long as most instances are correct, a satisfactory performance can be achieved.

The comparison between the TEIRESIAS approach and the neural network approach to error handling is shown in Table 4. The neural network approach may be more useful than TEIRESIAS in handling multiple errors or errors involving some unobservable concepts which human experts may have difficulties in dealing with. In addition, the back-propagation rule can be uniformly applied to the whole rule base, whereas human experts may focus on certain parts of the rule base consciously or subconsciously. Also, in Reference 15, it is suggested that the only proper way to cope with deleterious interactions among rules is to delete offending rules. In light of this view and the experience of the present authors, the deletion operator is very useful. While the neural network approach is still too simple to deal with errors involving qualitative changes of rules, reasoning strategies or meta-level knowledge, the techniques developed under this approach can supplement the current rule base technology.

EVALUATION

In this section, the validity of the developed mapping from a rule-based system into the neural architecture will be demonstrated on a practical domain, namely, the problem of diagnosing jaundice, and then the applicability of this approach to a large rule-based system with thousands of rules will be discussed.

Derived from JAUNDICE[17], the rule base used contains 50 rules, 5 final hypotheses, 3 intermediate hypotheses and 20 clinical attributes. This rule base is mapped into a bi-layered network with 5 output nodes, 3 middle nodes and 20 input nodes. Twenty training instances that can be diagnosed correctly by these 50 rules were collected from the JAUNDICE case library.

Ten experiments were carried out. In each experiment, a small number of incorrect connections (rules) that contradict medical knowledge were added to the network described above. These incorrect rules were provided by a medical expert. No incorrect rule was shared by two experiments. Then the neural network transformed from the rule base was used to diagnose the 20 training instances before and after learning. The objective of these experiments was to see whether those incorrect rules could be removed through learning and how good the inference capability is. In each experiment, the number of incorrect rules and the diagnostic accuracy were recorded before and after learning. The results are shown in Figure 5. The statistical paired t test was used to judge whether learning can remove incorrect rules and improve the system performance significantly.

Two null hypotheses were formulated. The first states that there is no difference in incorrect rule numbers before and after learning. The second states that there is no difference in the system performance before and after learning. The t values for the first and the second hypotheses are 6.32 and 5.85, respectively. Both hypotheses are rejected at level of significance $\alpha < 0.01$. In other words, this approach is effective in these experiments. Another result is that no rule among the original 50 correct rules was deleted.

One important question is whether this approach can be scaled up to the order of rule bases large enough for industrial application (such as XCON, which contains thousands of rules). The 50 rule expert system used in the above experiments is certainly too small. However,

Table 4. Comparison between TEIRESIAS and the neural network approach

	TEIRESIAS	Neural network approach
Approach	Human experts	Back-propagation
Operators	Modifying strengths Deletion Addition Generalization Specialization	Modifying strengths Deletion
Errors	Rule errors	Rule and data errors

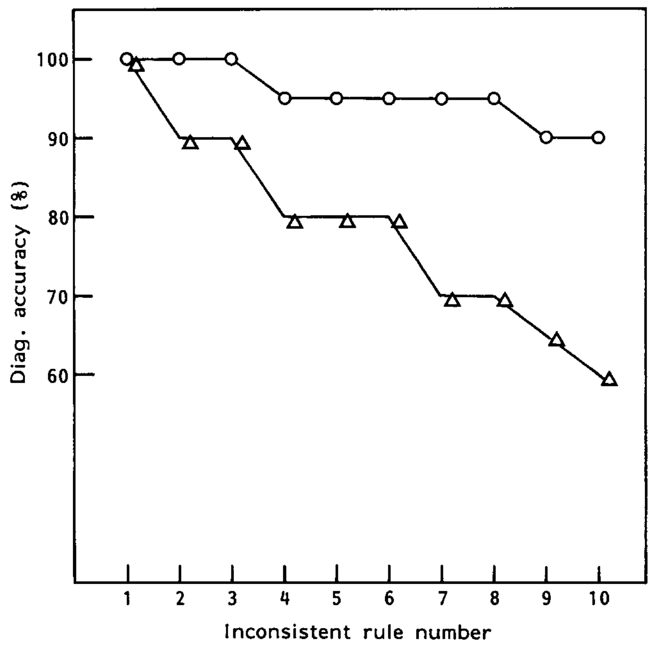

Figure 5. Performance measured by diagnostic accuracy. ○, *case I: after learning;* △, *case II: before learning*

the validity of the developed approach can be based on the following arguments. First, the analogies between neural networks and belief networks were shown earlier. Second, because knowledge-based networks are much sparser than ordinary neural networks and may be further decomposed into independent networks, they can escape the problems due to combinatorics.

DISCUSSION

In a simple production system, rules can interact only through the working memory, which becomes a bottleneck during processing of a large number of rules. One way to get around this problem is by using a distributed architecture. A rule base is mapped into a network so that a rule will fire when its premise node is activated. Rule interaction becomes distributive over the network rather than centralized through the working memory. In the literature, it has been shown that compiling a rule base into a network significantly increases the system performance. When a distributed architecture is taken, the performance then depends on how many rules can be processed in parallel. Consequently, a parallel distributed architecture such as the neural network is desirable for building rule-based systems.

Under the mapping as described, the neural network is able to compute the belief values of given hypotheses. Thus, it is especially suitable for handling uncertainty. However, the mathematical abstraction of a rule base as a belief network does not address adequately such aspects as symbol-based pattern matching, which are of major importance in a rule-based system. In addition, it should be noted that the mapping is straightforward for propositional logic but complicated for predicate logic. An architecture which combines the neural network and the rule-based network (e.g., compiled from the Rete algorithm) will be the ultimate choice.

Learning in the neural network is characterized by seeking local optima. In general, learning heuristics are kinds of gradient descent procedures. A fundamental problem is that local optima are not necessarily global optima. Learning performance critically depends on the choice of the initial state and the architecture. Without any knowledge or bias, the rules generated by such learning heuristics usually lack psychological meanings. Therefore, when mapping a rule-based system into the neural architecture, we preserve the system topology and the human-assigned rule strengths as the key bias.

The advantage of learning under such mapping lies in refining rather than creating a knowledge base. In particular, the back-propagation strategy is useful for debugging intermediate rules whose behaviour is not observed directly from samples. The approach presented here provides a means to delete inconsistent rules and refine rule strengths. However, this approach is quite limited in conducting more sophisticated rule revision operations involving qualitative changes in rules as well as in learning meta-level or control heuristics.

In the current technology, the knowledge-based system is suitable for high-level cognitive tasks, whereas the neural network proves useful for perceptual and signal processing tasks. The rule-based technique has become the most important one with which to build a knowledge-based system primarily because expertise can be easily captured and modified as rules. It has been found that rule-based systems are natural for some domains but awkward for others. Several problem characteristics have been identified that distinguish between domains for which rule-based systems are suitable and those for which rule-based systems are not[10]. In a suitable domain, the theory is diffuse, the control flow is simple and the use of knowledge is not predetermined. While rule-based systems have spanned a wide range of applicability, the inference model provided by a rule-based system is more appropriate for problems where solutions are selected from a predetermined set than for problems where solutions need to be constructed. In this sense, the presented approach is suitable for diagnostic or analytical problems, but may not be for planning problems.

The neural network architecture (connectionism) has been applied to solve problems which more or less can be categorized into pattern recognition, optimization, association and self-organization. Specific applications include recognition of hand-written characters, speech understanding, image or signal processing (e.g., noise filtering, data compression), machine diagnosis, analysis of sensor information, and financial and economic modelling. This architecture is suitable for systems where problems being considered can be modelled as a network consisting of a large number of processing elements and connections, and solutions to the problems depend on how those processing elements connect and what the connection weights are. Mapping MYCIN-like rule-based systems into the neural architecture is based on this consideration.

CONCLUSIONS

It has been long argued that the neural network approach to artificial intelligence is neither necessary nor feasible. However, recent successes of the neural network approach have forced a reconsideration of this route to production of intelligent behaviour. By contrast the knowledge-based approach emphasizes that knowledge is the key to generation of intelligent computer systems. An emerging trend is to combine these two approaches. It is thus justified to ask how to implement a rule-based system on the neural architecture. This question has been answered here in the context of MYCIN and similar systems.

A rule-based system has been mapped into a neural network by mapping the knowledge base and the inference engine into a kind of neural network called conceptualization. Under such mapping, a sentence in the rule base is mapped into a concept node in the conceptualization. The inference behaviour is characterized by propagating and combining activations recursively through the network and may involve an iterative search for a stable state. The learning behaviour is based upon a mechanism called back-propagation, which has been shown to be effective in both rule and data revision.

The advantages and disadvantages of this approach have been examined. The system performance can be greatly enhanced by using a distributed paralled architecture. However, the neural network does not address symbol-based pattern matching. The learning heuristics for neural networks are effective in deleting inconsistent rules and modifying rule strengths but do not allow more sophisticated rule revisions. An architecture that combines neural network and the rule-based network will be the ultimate alternative.

The authors argue for the generality of this approach based upon the analogies observed between a belief network and a neural network and the relative sparseness of a knowledge-based network. Finally this approach has been validated on a practical domain.

REFERENCES

1 **Feldman, J A and Ballard, D H** 'Connectionist models and their properties' *Cognitive Sci.* Vol 6 No 3 (1982) pp 205–254

2 **Feldman, J A, Fanty, M A, Goddard, N H and Lynne, K J** 'Computing with structured connectionist networks' *Commun. ACM* Vol 31 No 2 (1988) pp 170–187

3 **Rumelhart, D E, Hinton, G E and Williams, R J** 'Learning internal representation by error propogation, in *Parallel Distributed Processing Explorations in the Microstructures of Cognition* MIT Press, USA (1986)

4 **Smolensky, P** 'Connectionist AI, symbolic AI, and the brain' *Artif. Intell. Rev.* Vol 1 (1987) pp 95–109

5 **Minsky, M and Papert, S** *Perceptrons* MIT Press, USA (1969)

6 **Sejnowski, T J and Rosenberg, C R** *NETalk: a Parallel Network that Learns to Read Aloud* JHU/EECS-86/01, Johns Hopkins University, USA (1986)

7 **Gallant, S I** 'Connectionist expert systems' *Commun. ACM* Vol 31 No 2 (1988) pp152–169

8 **Jones, W P and Hoskins, J** 'Back-propogation: a generalized delta learning rule' *Byte* (October 1987) pp 155–162

9 **Hopfield, J J and Tank, D W** 'Computing with neural circuits: a model' *Science* Vol 233 No 4764 (1986) pp 625–633

10 **Buchanan, B G and Shortliffe, E H** *Rule-Based Expert Systems* Addison-Wesley, USA (1984)

11 **Linsker, R** 'Self-organization in a perceptual network' *Computer* (March 1988) pp 105–117

12 **Davis, R** *Application of Meta-level Knowledge to the Construction, Maintenance and Use of Large Knowledge Base* PhD thesis, Computer Science Dept., Stanford University, USA (1976)

13 **Politakis, P G** *Using Empirical Analysis to Refine Expert System Knowledge Base* PhD thesis, Rutger University, USA (1982)

14 **Suwa, M, Scott, A C and Shortliffe, E H** 'Completeness and consistency in a rule-based expert system' in **Buchanan, B G and Shortliffe, E H (Eds)** *Rule-Based Expert Systems* Addison-Wesley, USA (1984)

15 **Wilkins, D C and Buchanan, B G** 'On debugging rule sets when reasoning under uncertainty' in *Proc. AAAI-86* Philadelphia (1986) pp 448–454

16 **Doyle, J** 'A truth maintenance system' *Artif. Intell.* Vol 12 No 3 (1979) pp 231–272

17 **Fu, L-M** *Learning object-level and meta-level knowledge in expert systems* PhD thesis, Stanford University (1985)