6-1991

Connectionist expert system with adaptive learning capability

B. T. LOW

Hochung LUI

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Hoonheng TEH

Citation

LOW, B. T.; LUI, Hochung; TAN, Ah-hwee; and TEH, Hoonheng. Connectionist expert system with adaptive learning capability. (1991). *IEEE Transactions on Knowledge and Data Engineering*. 3, (2), 200-207.
Available at: https://ink.library.smu.edu.sg/sis_research/5200

least, enable us to ask better questions based on insight gained from knowledgeable people.

Acknowledgment

We wish to acknowledge the assistance provided by R. Doring, F. Pierson, and B. Smith, Key Corp. Inc., Albany, NY, and Prof. J. Durgee, Management Policy and Organization, School of Management, Rensselaer Polytechnic Institute.

References

[1] B. E. Collins and H. Guetzkow, *A Social Psychology of Group Processes for Decision Making.* New York: Wiley, 1964.

[2] E. F. Fern, "The use of focus groups for idea generation: The effects of group size, acquaintanceship, and moderator on response quantity and quality," *J. Marketing Res.,* vol. 19, pp. 1–13, 1982.

[3] ——, "Focus groups: A review of some contradictory evidence, implications and suggestions for future research," *Advances Consumer Res.,* vol. 10, pp. 121–126, 1983.

[4] B. A. Fisher, *Small Group Decision Making.* New York: McGraw-Hill, 1974.

[5] A. E. Goldman, "The group depth interview," in *Focus Group Interviews: A. Reader,* J. Higginbotham, and R. Cox, Eds. Chicago, IL: American Marketing Association, 1979.

[6] E. Harrison, *The Managerial Decision-Making Process.* Boston, MA: Houghton-Mifflin, 1987.

[7] J. M. Hess, "Group interviewing," in *1968 ACR Conference Proceedings,* R. L. King Ed. IL: American Marketing Association, 1968.

[8] R. Hirokawa and R. Pace, "A descriptive investigation of the possible communication-based reasons for effective and ineffective group decision making," *Commun. Monographs,* vol. 50, pp. 363–380, 1983.

[9] R. R. Hoffman, "A survey of methods for eliciting the knowledge of experts," *SIGART Newsletter,* vol. 108, pp. 19–27, 1989.

[10] K. L. McGraw and K. Harbison-Briggs, *Knowledge Acquisition Principles and Guidelines.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[11] K. L. McGraw and M. R. Seele, "Knowledge elicitation with multiple experts: Considerations and techniques," *Artif. Intell. Rev.,* vol. 2, pp. 31–44, 1988.

[12] D. L. Morgan, *Focus Groups and Qualitative Research,* Sage Univ. Paper Series on Qualitative Research Methods, Sage, CA, vol. 16, 1988.

[13] N. Shadbolt and A. M. Burton, "The empirical study of knowledge elicitation techniques," *SIGART Newsletter,* vol. 108, pp. 5–18, 1989.

[14] H. A. Simon, *The New Science of Management Decisions.* New York: Harper and Row, 1960.

[15] W. A. Wolf, "Knowledge acquisition from multiple experts," *SIGART Newsletter,* vol. 108, pp. 138–140, 1989.

Connectionist Expert System with Adaptive Learning Capability

B. T. Low, H. C. Lui, A. H. Tan, and H. H. Teh

***Abstract*—This paper describes a neural network expert system called Adaptive Connectionist Expert System (ACES) which will learn adapatively from past experience. ACES is based on the neural logic network which is capable of doing both pattern processing and logical inferencing. We discuss two strategies here: pattern matching ACES and rule inferencing ACES. The pattern matching ACES makes use of past examples to construct its neural logic network and fine-tunes itself adaptively during its use by further examples supplied. The rule inferencing ACES conceptualizes new rules based on the frequencies of use on the rule-based neural logic network. This new rule could be considered as a new pattern matching example and be incorporated into pattern matching ACES.**

***Index Terms*—Adaptive learning, expert systems, logic programming, neural logic network, neural network.**

Manuscript received October 1, 1989; revised October 1, 1990.

The authors are with the Institute of Systems Science, National University of Singapore, Kent Ridge 0511, Singapore.

IEEE Log Number 9144307.

I. Introduction

One of the major areas for AI applications is in equipment troubleshooting. To develop a rule-based diagnostic expert system, the designer needs to interview the domain expert and formalize the expert's knowledge as inference rules and data structure. There are several difficulties in this knowledge acquisition phase. First, it is a very time-consuming process, involving much of the expert's valuable time. Second, the expert may not be able to express his knowledge in a form which can be easily encoded as a rule. Moreover, it is difficult to check whether the acquired knowledge base is complete and consistent. As a result, knowledge acquisition is typically an iterative process and the knowledge base will be modified until the performance of the system is satisfactory.

Human beings, however, possess enormous capability to acquire knowledge automatically. It is generally observed that for a technician new to the job, he needs to rely heavily on the toubleshooting manuals, and to follow the diagnostic flowcharts faithfully until the faulty component is located. However, after a few years of experience, he develops a good knowledge about the troubleshooting process so that when he looks at the fault symptoms, he can guess with good confidence which component is at fault. Internally, he also develops a ranking about symptoms, and is able to focus on those which are more important, using the others as supporting evidence. The automatic development of expert skills has been extensively studied by cognitive psychologists. Research on novice-to-expert shift has been done on chess [1], physics [2] as well as computer programming [3]. Anderson suggested that the development of expertise occurs in stages [4]. He summarized it as follows:

> Skill learning occurs in three steps: (1) a cognitive stage, in which a description of the procedure is learned; (2) an associative stage, in which a method for performing the skill is worked out; and (3) an autonomous stage, in which the skill becomes more and more rapid and automatic.

and

> Underlying the development of expertise is the transformation of problem solving from a basis in serial processing and deduction to a basis in memory retrieval and pattern matching.

Quinlan proposed a technique for generating production rules from decision tree [24]. Our aim is to develop an expert system which does not only stop at rules, but can learn from rules as well as past experience and exhibit characteristics of skill learning as suggested by Anderson. The model which we have adopted is the connectionist model (also call neural network).

A connectionist expert system (CES) consists of many nodes (or neurons), each of which is a simple computation element. Nodes are heavily interconnected and their computation can be done in parallel. In recent years, there have been quite a few successful expert systems developed based on this approach. Application domains include career guidance [5], medical diagnosis [6], and solar flare forecasting [7]. In his series of publications [8], [9], Gallant has laid the foundation for building expert systems from connectionist models. In this approach, the knowledge base is distributed across the entire network and is represented by the connection weights between nodes. Powerful learning algorithms, such as the pocket algorithm [8] and the back-propagation algorithm [10], exist so that the knowledge base can be trained from a set of examples. Thus, the tedious effort of knowledge acquisition can be greatly reduced. After training, the system behaves as though it follows rules. Sejnowski calls it rule-following as opposed to rule-based [11]. In [7], the performance of the connectionist approach is compared to that of the rule-based method. It took the developer one man-year to build the rule-based system (with 700 rules), whereas the connectionist approach was developed in less than one week using a simpler simulator. Yet both systems performed as well as the human expert. In addition, the time for inferencing was also greatly reduced from five minutes to a few milliseconds.

While CES is capable of learning from examples, it does so by finding a (nonlinear) mapping between the input attributes and the output categories. In many practical situations, such input/output associations can be easily obtained. For example, technicians may have a log book which keeps records on fault symptoms and the corresponding diagnostic results. In credit card application screening, past records relating the applicant's financial conditions and the decision of approval/denial are usually available. These data can be used to train the CES using the automatic learning algorithms. However, although the knowledge base is built from examples, it cannot be updated easily when new information is available during operating. The main difficulty is that the learning algorithms are not well suited for incremental learning. For example, the popular back-propagation algorithm uses the gradient descent procedure to find the nonlinear mapping function. When a new associative pair is to be incorporated to the existing knowledge base, the entire knowledge base may have to be retrained, using new and previous examples. Training in such an algorithm can be very time consuming. Sejnowski reported that it took 12 hours of VAX 780 CPU time to train the NETtalk system [11]. As a result, few CES's provide adaptive, incremental learning facilities.

Another drawback of this approach is that the system is essentially doing pattern matching. While the learning algorithm can find the association between the input and the output, the system does not know how and why they are related. As a result, the system has poor explanation facility, and this may be intolerable in some applications.

For many diagnostic applications, the troubleshooting manuals are readily available. These manuals typically instruct the user to first perform certain measurement, then based on the result proceed to the next step. These instructions can be encoded into conditional IF THEN rules; and are a valuable source of knowledge for diagnostic systems.

We are developing expert systems which have incremental learning capabilities so that the knowledge base can be updated as new information is available. We refer to our system as Adaptive Connectionist Expert System (ACES). Two types of adaptive strategies are discussed, the first one is based on pattern matching while the other is based on rule inferencing. We adopted the neural logic model [13], which is capable of handling both types of computations, as the basic element in our system.

Section II of this paper gives a general overview of the neural logic model and its characteristics in logical inference and pattern classification. Section III discusses how to build the knowledge base using existing and new examples and the corresponding inferencing mechanism. Section IV discusses the realization of propositional rules in neural logic network, the adaptive search strategy, and how special input patterns can be detected to create new examples. With this capability, the system can transform its problem solving strategy gradually from deductive reasoning to pattern matching—a phenomenon similar to the novice–expert shift to technicians.

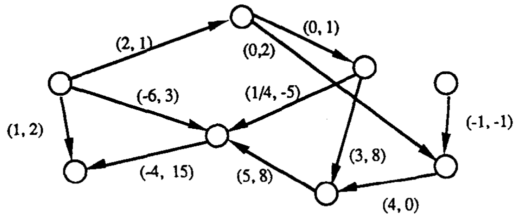


Fig. 1. A neural logic network.

II. Neural Logic Networks

Neural logic network [12], [13] incorporates both pattern processing capability of multilayer perceptrons [10] and logical inferencing capability of Boolean logic inference networks [19] within a single frame of neural network environment. This class of neural networks can model classical three-valued Boolean logic effectively and can be extended to perform probabilistic [16] and fuzzy logic [17].

A three-valued neural logic network as shown in Fig. 1 is a finite directed graph. A set of nodes is selected to be input nodes and the other set be output nodes, and each node can take one of the three ordered pair activation values:

$$(1, 0) \text{ for TRUE,}$$
$$(0, 1) \text{ for FALSE and}$$
$$(0, 0) \text{ for DON'T-KNOW.}$$

Every edge or link is associated with an ordered pair (x, y) where x and y are real numbers of positive, negative, or zero value. The use of ordered pair weightages allows the models to be more flexible than a single-value model which emphasizes both positive and negative input equally.

The propagation rule of neural logic network is defined as follows.

Let P be a particular node and $\{Q_1, Q_2, \cdots, Q_k\}$ be the set of all nodes which are linked to P. Suppose the existing value of Q_i is (a_i, b_i) and the weight of the edge linking Q_i to P be (x_i, y_i).

Step 1: Compute net excitatory input $\alpha = \sum a_i x_i$, and net inhibitory input $\beta = \sum b_i y_i$.

Step 2: Value of node

$$\begin{aligned} P = {} & (1, 0) \text{ if } \alpha - \beta >= 1. \\ & (0, 1) \text{ if } \alpha - \beta <= -1 \\ & (0, 0) \text{ otherwise.} \end{aligned}$$

Let us look at an example in Fig. 2:

We have

$$\alpha = 4, \quad \beta = 2$$
$$\alpha - \beta = 2 > 1.$$

Hence, the final value for node P is (1,0).

A. Logical Inference Based on Three-Valued Boolean Logic

Classifical Boolean logic is developed based on two truth values "TRUE" and "FALSE". In three-valued Boolean logic, we include one more value "DON'T KNOW." As a generalization of inference networks [19], neural logic network is able to represent logical operations AND, OR, NOT, IF..THEN.. and implication.

For example, a two-input three-valued logical OR operation can be realized by a network in Fig. 3(a). When the above-mentioned propagation rule is applied to the network, it yields the truth table as shown in Fig. 3(b). Similarly, by assigning different weight values

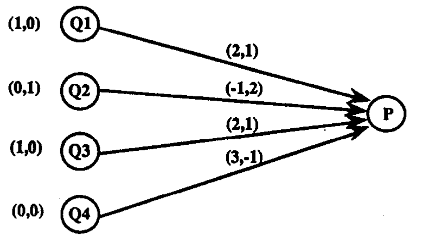
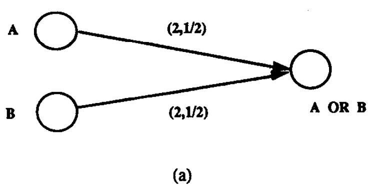


Fig. 2. Propagation of activation values.



A	B	A OR B
(1,0)	(1,0)	(1,0)
(1,0)	(0,0)	(1,0)
(1,0)	(0,1)	(1,0)
(0,0)	(1,0)	(1,0)
(0,0)	(0,0)	(0,0)
(0,0)	(0,1)	(0,0)
(0,1)	(1,0)	(1,0)
(0,1)	(0,0)	(0,0)
(0,1)	(0,1)	(0,1)

(b)

Fig. 3. Operation OR.

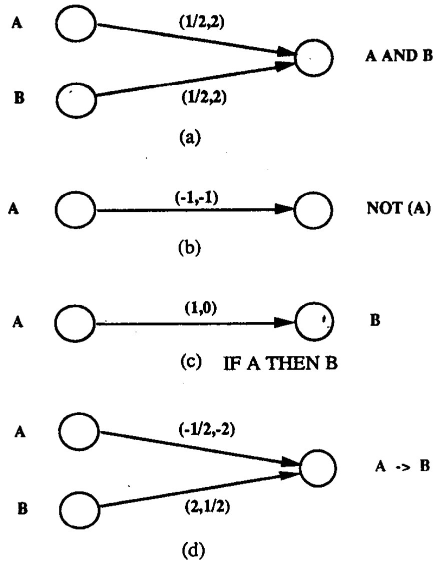


Fig. 4. Logical operation AND, NOT, IF..THEN, and IMPLY.

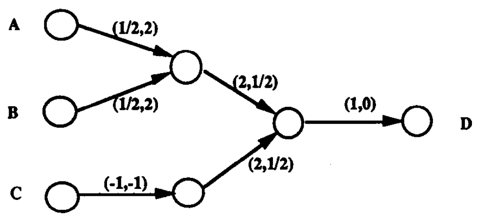


Fig. 5. Network representation of a sample rule.

to the connection links, the neural logic network can realize different three-valued logic functions as shown in Fig. 4.

Using the above definition, we can easily construct a neural logic network for any rule involving any combination of logical operators. For an example, the logical statement

$$\text{IF } (A \text{ AND } B) \text{ OR } (\text{NOT } C) \text{ THEN } D$$

can be represented by a network as shown in Fig. 5.

We can build a representation of the inferencing process using a neural logic network with each node in the network representing a proposition. The input nodes are assigned with known values whereas the others are all assigned with DON'T-KNOW $(0,0)$. A new value for each node is then calculated according to the propagation rule. The process is repeated until there is no change in the values, i.e., the network settles in a stable state. At this point, if the value of a output node is still $(0,0)$, then it means that existing information is insufficient to arrive at a definitive conclusion; those output nodes having value of $(1,0)$ or $(0,1)$ will indicate the truth value of the corresponding proposition (true or false).

B. Pattern Processing Capability

On the other hand, like any other neural network models, the neural logic network is also capable of performing pattern processing. Besides that the perceptron convergence procedure and the back-propagation algorithm [10] can be extended to train neural logic network, [12] and [13] showed that it is always possible to construct a neural logic network to match any input/output association. This theory forms the basis for a neural logic learning algorithm.

The Neural Logic Learning Algorithm consists of three subalgorithms, namely Construction Algorithm, Feature Extraction and Enforcement, and Tuning Algorithm.

Construction Algorithm: Given a set of training examples $\{E_1, E_2, \cdots, E_k\}$. For each example E_t, let X_t be its input attribute set (say, with size of n) and Z_t be its output attribute set (say, with size of m).

To match the pattern set $\{X_1, X_2, \cdots, X_k\}$ to $\{Z_1, Z_2, \cdots, Z_k\}$.

Step 1: Construct a directed graph with three columns as shown in figure a, column 1 has n nodes, column 2 has k nodes, and column 3 has m nodes. From every node in column 1, draw a directed edge to every node in column 2, and from every node in column 2, draw a directed edge to every node in column 3.

Step 2: For each node U_i in column 1 and each node U_j in column 2, attach the edge joining them by the ordered pair (α, β) obtained as follows:
Let (a, b) be the ith value of the vector X_j,
Case 1: If $(a, b) = (1, 0)$, we define $(\alpha, \beta) = (1/c, 0)$
Case 2: If $(a, b) = (0, 1)$, we define $(\alpha, \beta) = (0, -1/c)$ where c is the number of values in X_j with either $(1, 0)$ or $(0, 1)$.
Case 3: If $(a, b) = (0, 0)$, we define $(\alpha, \beta) = (-1/(d + 1), 1/(d + 1))$, where d is the number of values in X_j with $(0, 0)$.
Hence, $d + c = n$.

Step 3: For each node U_i in column 2 and each node U_j in column 3, attach the edge joining them by the ordered pair (α, β) obtained as follows.
Let (a, b) be the jth value of the vector Z_i,
Case 1: If $(a, b) = (1, 0)$, we define $(\alpha, \beta) = (1, 0)$
Case 2: If $(a, b) = (0, 1)$, we define $(\alpha, \beta) = (-1, 0)$

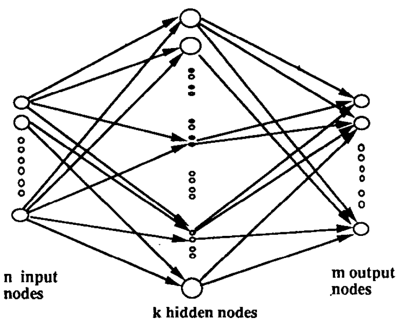


Fig. 6. A network constructed by neural logic learning algorithm.

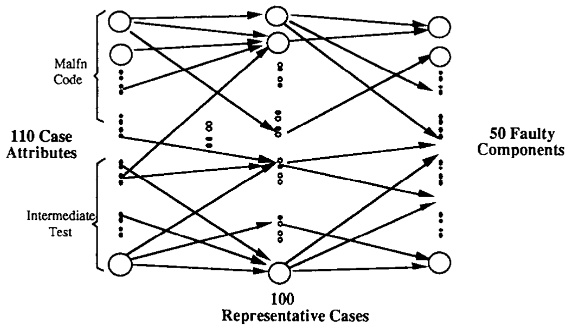


Fig. 7. Network KB for INS troubleshooting.

```
Case 3: If $(a, b) = (0, 0)$, we define $(\alpha, \beta) = (0, 0)$.
Step 4: Attach every node in this network with initial value of
        $(0, 0)$.
```

The nodes in column 1 are selected as input nodes while the nodes in column 3 are selected as output nodes. Direct verification will show that this neural logic network matches the input pattern set $\{X_1, X_2, \cdots, X_k\}$ to the output pattern set $\{Z_1, Z_2, \cdots, Z_k\}$.

2) Features Extraction and Enforcement:

—Extraction:

```
Step 1: Let $E$ be the given training example.
        Let $S$ be the set of examples not distinguishable from $E$.
Step 2: Initialize the identification set to be an empty set.
        Initialize $S$ to be the set of all training examples.
Step 3: While size $(S) > 1$ do
          for each input attribute $a$ in the example
            calculate count$[a]$ = number of training examples $T$
            in $S$ such that $T[a] = E[a]$
          find $m$ such that count$[m]$ is the minimum for all
          attributes
          put $m$ into the identification set
          $S = \{T/T$ in $S$ and $T[m] = E[m]\}$.
```

Upon identifying the identification set of each training example, the influence of those key features can be amplified by the following algorithm:

—Enforcement:

```
for each training example $E_t$,
  Let $I$ be the identification set of $E_t$ and $S$ be the size of $I$
  for each element $m$ in $I$
    Let $(\alpha, \beta)$ be the weight attached to the link joining the $m$th
    node in column 1 and the $t$th node in column 2.
      modify $(\alpha, \beta)$ such that
      $\alpha = \alpha^*(1 + 1/S)$
      $\beta = \beta^*(1 + 1/S)$.
```

3) Tuning: For each training example E_t,

```
assign values of its input attribute set $X_t$ to the input nodes
compute the values on the nodes in column 2.
For each node $U_i$ in column 1 and each node $U_j$ in column 2
Let $(\alpha, \beta)$ be the weight associated with the edge joining them,
$(a, b)$ be the $i$th value of $X_t$ and $(c, d)$ be the propagated value of
the $j$th node of column 2. Compute $r = c - d$.
We shall modify the values $(\alpha, \beta)$ according to the following cases:
Case 1: If $j = t$, put
          $\alpha = \alpha + \text{gain}^* a^*(1 - r)$
          $\beta = \beta + \text{gain}^* b^*(r - 1)$
Case 2: If $j <> t$, put
          $\alpha = \alpha + \text{gain}^* a^*(-r)$
          $\beta = \beta + \text{gain}^* b^*(r)$
          where gain is a small constant.
```

This process is repeated until there is no change in weight.

The Construction Algorithm involves an assignment of weightages to all the links in the network such that after the assignment, the network is able to match all the training examples perfectly. That is, given the input set of each training example, the network, after the propagation, will be able to produce the correct output set at the output layer. However, the resultant weightages will be too precise that the network will not tolerate any error in the input.

It is recognized in "The Bayes Connection" [21] that any pattern recognition problem must be defined by the fairly important correlations. We agree with this and believe that in reality, some input attributes are more important than others. We should identify them and enforce their influence for the pattern recognition of past examples. Formally, we define an ***identification set*** of an example E to be a set of input attributes that enable us to distinguish E from the rest of the examples. Given a training example, we can deduce its identification set using a simple algorithm. The influence of those key features can then be amplified by multiplying all the weightages on their outgoing links by a constant greater than 1.

After enforcing the identification sets, the neural logic network will not match the training examples correctly. A controlled tuning algorithm has thus to be applied on the first layer of the network to relearn back the heuristics. After the tuning, the network will be different from the originally constructed version but it will still match the given set of examples correctly and in addition, it will tolerate error in the user input. One advantage of the Neural Logic Algorithm is that it provides us with the necessary network topology, including the number of hidden nodes in the intermediate layer. Moreover, it ensures convergence because a solution has already been found (by the construction algorithm) and the tuning is easy and fast. With different emphasis on the enforced set of attributes, we may build

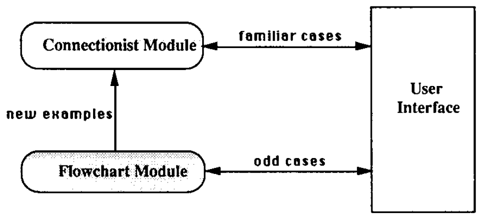


Fig. 8. Adaptive learning strategy of INSIDE.

several pattern matching ACES overlaying each other to provide a more comprehensive system.

III. Pattern Matching ACES

This section describes an adaptive connectionist expert system that can be constructed from past examples and tuned adapatively during its use by examples from another knowledge module. The system named Inertial Navigation System Interactive Diagnosis Expert (INSIDE) [14] was jointly developed by Institute of Systems Science and Singapore Airline for toubleshooting a piece of avionics equipment—Inertial Navigation System (INS).

There are detailed flowcharts which provide guidance in trouble shooting this piece of equipment. Nevertheless, once the technician has built up knowledge on the failure modes of the equipment, they will tend to rely more on their knowledge rather than the troubleshooting charts. A broad understanding of the equipment, coupled with a good knowledge of its failure modes gathered from past experience, proved to be more difficult.

In Singapore Airlines (SIA), a large database on the equipment failures has thus been accumulated over the years which represents the combined experience of many technicians. In building the connectionist knowledge base, heuristics regarding the diagnosis of INS were collected in the form of training examples using a combination of interviewing and referring to the documented cases in the database. Among the cases reviewed, a total of 100 troubleshooting cases with 110 fault attributes and 50 faulty components were selected for training the knowledge base. Using the Neural Logic Learning Algorithm, our knowledge base is a two-layered neural logic network with 110 input nodes, 100 hidden nodes, and 50 output nodes.

Note that, not all input attributes of a case would be known at the beginning of the consultation, an inference engine [14], [15] is needed to direct the flow of the consultation and to acquire more information from the users.

At the beginning of the consultation, the user will be asked to supply some input attributes values such as the malfunction codes or the complaints from pilots. Having received the initial set of information, the corresponding input nodes will propagate the value forward to the nodes in the other layers (forward chaining). If one of the intermediate node is fired, the value will be propagated to the output layer and the appropriate conclusion(s) will be reported to the user. Otherwise, the system calculates a confidence estimate of each intermediate node and identifies the intermediate node that has the highest confidence estimate. In pursuing that node, backward chaining occurs to identify the next input attributes to be pursued. To acquire that attribute value, the user might be asked to perform some intermediate test on the INS. After the user supplies the information, the system continues to infer forward again until a conclusion is reached.

Knowing that having insufficient training examples, the knowledge base might not be able to cover the entire problem domain, a troubleshooting flowchart module was developed to serve as a backup. During the user consultation, a connectionist module will be activated first. If the case is close enough to a case that was captured before, the module is able to derive the solution in just a few steps. Otherwise, the user will be directed to the flowchart module to continue the diagnosis with the help of the troubleshooting charts. While the connectionist module often provides a short cut to solve most familiar problem, the flowchart module resolves those uncommon cases. After the case is solved, it can be formulated as a new example to be acquired by the connectionist knowledge base. Note that by using the Neural Logic Learning Algorithm, the tuning of knowledge base is no longer a tedious and time consuming process, the knowledge base constructed does not behave just like a lookup table and it can relate new problems to old experience if they are close to each other. Under this KB updating strategy, the connectionist knowledge base is able to learn adaptively from new examples from the flowchart module as the system is being used. Besides providing an economical way for developing fault diagnostic systems in general, the learning process of the system highly resembles the way an expert acquires knowledge through experience.

IV. Rule Inferencing ACES

In [22], a network approach was proposed to process hierarchical knowledge. In this section, we will discuss another aspect of ACES that not only can handle hierarchical knowledge, but it also learns from these rules. This is based on the same framework employed by the pattern matching ACES: the neural logic network model described in Section II, and a particular adaptive control of resolution [20]. We express rules derived from the knowledge domain as propositional Horn-clauses and then converted these clauses into neural logic network as illustrated in the example below:

There are six rules

rule 1: a IF(b AND (NOT c)).
rule 2: a IF (c AND d).
rule 3: a IF (e AND f).
rule 4: d IF (g AND h).
rule 5: d IF (i AND j).
rule 6: d IF (k AND l).

By converting these rules into neural logic network representation as discussed in Section II of this paper, we will have a rule network as shown in Fig. 9.

When a deduction is made to the neural logic network, it starts from the proposition rule term, or the node as it is in the neural logic network, computes its truth value by searching and computing the truth values of all the related fragment of the network sequentially. The computation of node values is described in the propagation rule in Section II. Let us look at the previous example, and if we want to find out the truth value of the rule term "a," we can back-propagate from node "a" and follow through the entire network according to the sequence shown in Fig. 10 and obtain the answer.

However, with the expressiveness of neural logic network, we could improve this blind search and arrive at some adaptive search techniques for the deduction.

A. Adaptive Search

Since all related rule terms (or nodes) expressed by some rules in the neural logic network are connected together by links, priorities can be attached to these links to indicate the sequence of deductive search. We may preset the search sequence when compiling knowledge into rules but this could mean rigidity and often lead to inefficient knowledge consultations. A more desirable way is allowing the pattern of use to determine which path has higher priority.

Let us consider the same example again. Assuming after the expert system has been used for a period of time, analogous to Probabilistic Approached to dynamically update the certainty measure at each decision node [23], we attach the number of successful deductions during actual consultation to each link as shown in Fig. 11 below.

In Fig. 11, the total number of successful deductions for rule term "a" is 100 consultations and the distribution of these 100 successful passes are 10 passes for the first OR branch, 85 passes for the second OR branch, and five passes for the last OR branch. We may, therefore, rerank their search priorities according to their statistical data to

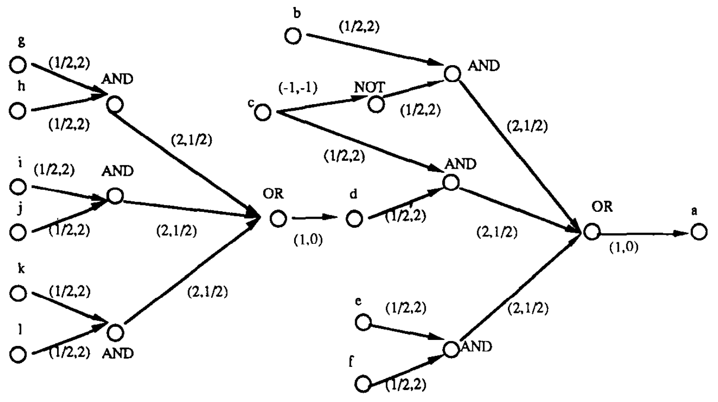


Fig. 9. A neural logic network.

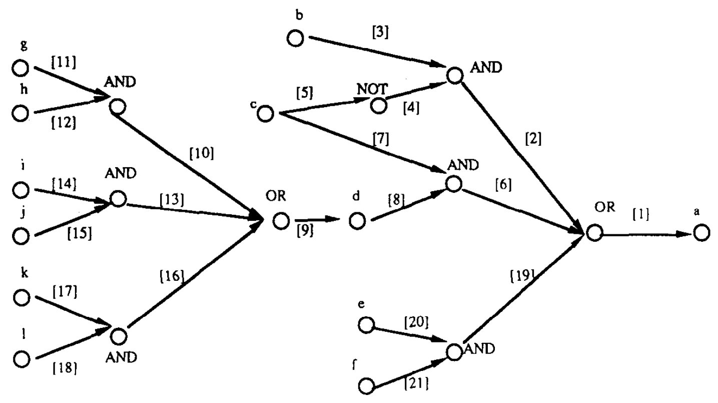


Fig. 10. A neural logic network with sequential search order.

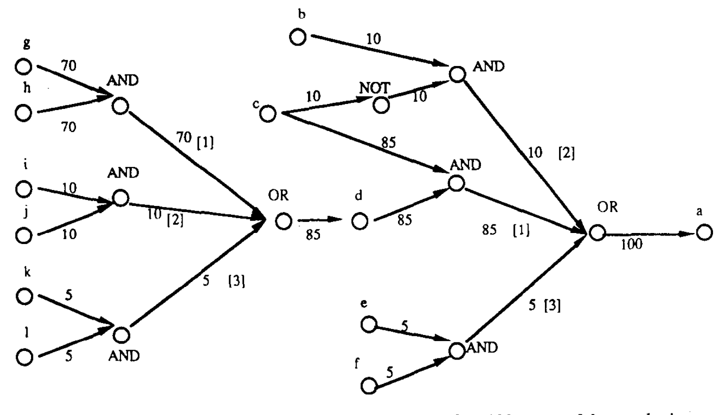


Fig. 11. A neural logic network with statistical data after 100 successful consultations.

improve on efficiency as indicated by the number of successful passes of each branch. This dynamic search order for each OR relation is also shown in Fig. 11 in square brackets.

We have so far dealt with alternative search branches, that is OR relation of a given node and we shall now extend it to the combinative branches—the AND relation. The rationale for AND relation is: if one of the branches fails, the AND relation is FALSE and we do not need to continue searching other branches anymore. In this case, the statistical data we need are the number of failures during past consultations for each AND branch and rank them accordingly. Fig. 12 below shows the same example with statistical data and ranking for the AND relations.

By making this reordering mechanism a dynamic feature in the rule inferencing ACES, we arrive at an expert system that will always adapt its search strategy to the pattern of use.

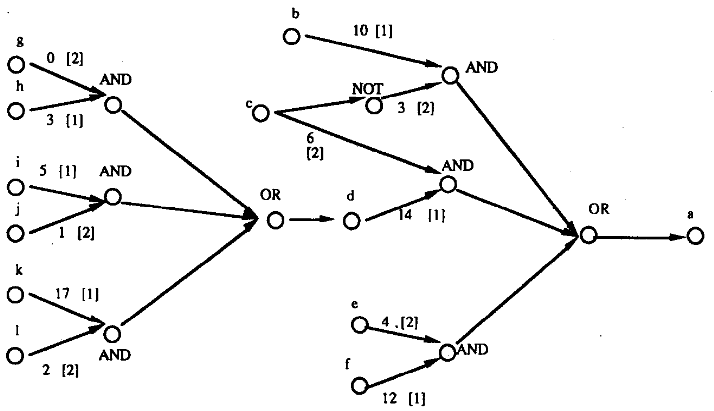


Fig. 12. A neural logic network with statistical data on unsuccessful consultations for AND relations.

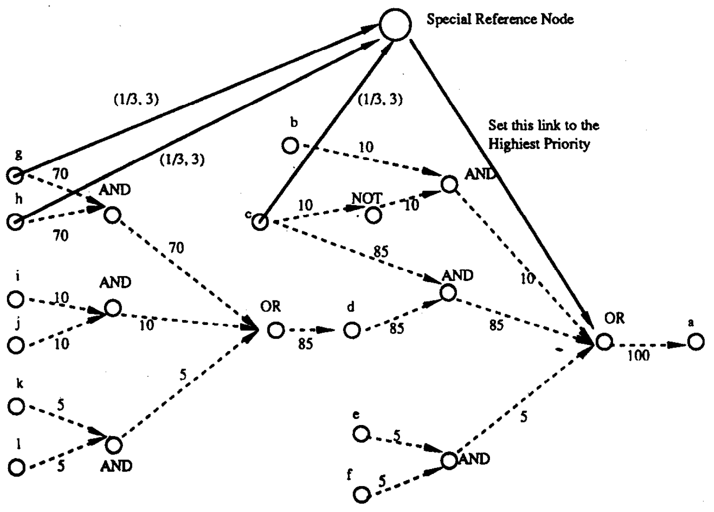


Fig. 13. Learning a special reference case from rules.

B. Learning from Rules

Human beings conceptualize knowledge from past experience. In the Rule Inferencing Connectionist Expert System, we could also learn and conceptualize special reference cases based on the experience gained during knowledge consultations. The Adaptive Search Mechanism discussed in the previous section provides a powerful source of record keeping. If we look at the example in Fig. 10 again, with the experience that the network has acquired so far, we can clearly identify that rule terms "c," "g," and "h" are the most frequently consulted input nodes set to deduce "a." We can view the relation between the resulting conclusion "a" and this set of rule terms as the knowledge learned from past consultations, which is equivalent to the short-cut repairing knowledge a technician gained from his past repairing experience.

To capitalize on this experience, we can always create an additional node in the neural logic network linking up the output of deduction, that is "a" in this case, and the set of rule terms, which can be consider as the inputs, as shown in Fig. 13. We call this new node "special reference case" and always set the highest priority to it. Whenever rule term "a" is consulted, the network will always refer to this special reference case before going into the deeply nested network of rules. If the statistical data change with time, we may automate this process by deleting the old reference case and create a new one according to the lastest situation.

The Rule Inferencing ACES using neural logic network with adaptive search and learning special reference capability will provide an environment for a rule-based expert system to continuously learn and improve its efficiency based on its past consultation experience.

V. Conclusion

This paper discusses two adaptive strategies to update the knowledge base of the expert system during operation. The first strategy describes how a pattern matching knowledge base constructed from past examples can be updated when new case is available, while the second method makes use of the frequency usage of rules to form a new rule. The new rule thus formed is a special reference case which can be considered as a "bypass" of the original set of rules, and it is applicable if all the antecedents of the rule are satisfied. As such, the new rule can also be considered as a new pattern matching example and hence can be incorporated to the example knowledge base using the first adaptive strategy. With this capability, the system can adapt to the operation environment and gradually transform its problem solving strategy from rule-reasoning to pattern-matching as it is being used.

REFERENCES

[1] W. G. Chase and H. A. Simon, "The mind's eye in chess" in *Visual Information Processing.* W. G. Chase, Ed. New York: Academic, 1973.

[2] J. Larkin, J. McDermott, D. P. Simon, and H. A. Simon, "Expert and novice performance in solving physics problems," *Science,* vol. 208, pp. 1335–1342, 1980.

[3] A. G. Bateson, R. A. Alexander, and M. D. Murphy, "Cognitive processing differences between novice and expert computer programmers," *Int. J. Man–Machine Studies.* vol. 26, pp. 649–660, 1987.

[4] J. R. Anderson, *Cognitive Psychology and Its Applications,* 2nd ed., San Fransico, CA" Freeman, 1985.

[5] A. H. Tan and L. K. Chee, "Connectionist expert system for intelligence advisory application," in *Proc. Expert Syst. Econom. Banking, Management,* and Singapore, Jan 11–13, 1989.

[6] K. Saito and R. Nakano, "Medical diagnostic expert system based on PDP model," in *Proc. IEEE ICNN,* Vol. II, San Diego, CA, July 24–27, 1988, pp. 525–532.

[7] G. Bradshaw, R. Fozzard, and L. Ceci, "A connectionist expert system that actually works," *Advances Neural Inform. Processing Syst., I,* pp. 248–255.

[8] S. I. Gallant, "Connectionist expert systems," *Commun. ACM,* Feb. 1988.

[9] ——, "Automatic generation of expert system from examples," in *Proc. 2nd Int. Conf. AI Appl.,* IEEE Press, New York, 1985, pp. 313–319.

[10] D. E. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,"* D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986.

[11] T. J. Sejnowski and C. R. Rosenbert, "Parallel networks that learn to pronounce English text," *Complex Syst.,* vol. 1, pp. 145–168, 1987.

[12] H. H. Teh and C. P. Yu Wellington, "A controlled learning environment of enhanced perceptron," in *IEEE Proc., Future Trend in Distributed Comput. Syst.,* 1988, Hong Kong.

[13] S. C. Chan, L. S. Hsu, S. Brody, and H. H. Teh, "Neural three-valued-logic networks," in *Proc., Inter-Faculty Seminar Neuronet Comput.,* June 1989, National Univ. of Singapore, pp. 54–75.

[14] A. H. Tan, Q. Pan, H. C. Lui, and H. H. Teh, "INSIDE: A neuronet based hardware fault diagnostic system," in *Proc. Int. Joint Conf. Neural Networks,* San Diego, CA, June 17–21, 1990.

[15] A. H. Tan and H. H. Teh, "Connectionist expert systems—An inductive cum deductive approach," *Inform Technol.*—J. Singapore Comput. Society, Special Issue on Knowledge Engineering, Feb. 1990.

[16] H. H. Teh, S. C. Chan, L. S. Hsu, and K. F. Loe, "Probabilistic neural-logic networks," in *Proc. Inter-Faculty Neuronet Seminar,* National Univ. of Singapore, June 1989.

[17] L. S. Hsu, H. H. Teh, S. C. Chan, and K. F. Loe, "Fuzzy decision making based on neural-logic networks," in *Proc. Inter-Faculty Neuronet Seminar,* National Univ. of Singapore, June 1989.

[18] T. Samad, "Towards connectionist rule-based systems," in *Proc. IEEE ICNN,* Vol. II, San Diego, CA, July 24–27, 1988, pp. 525–532.

[19] H. H. Teh, L. S. Hsu, and W. W. Tsang, "Modelling knowledge information systems using inference networks," *ARS Combinatoria,* vol. 23A, pp. 269–290, 1987.

[20] T. J. Reynolds, H. H. Teh, and B. T. Low, "Programming in neural logic," *Pacific Rim Int. Conf. AI'90,* Nov. 14–16, Nagoya, Japan.

[21] C. Anderson and E. Abrahams, "The Bayes connection," in *Proc. IEEE Int. Conf. Neural Networks,* San Diego, CA, 1987, pp. III105–112.

[22] L. Becker and J. Peng, "Networking processing of hierarchical knowledge for classification and diagnosis," in *Proc. IEEE Int. Conf. Neural Networks,* San Diego, CA, 1987, pp. II309–317.

[23] S. Chan, "Automated reasoning on neural networks: A probabilistic approach," in *Proc. IEEE Int. Conf. Neural Netorks,* San Diego, CA, 1987, pp. II373–378.

[24] J. R. Quinlan, "Generating production rules from decision trees," in *Proc. IJCAI 87,* Milan, pp. 304–307.