Rule Refinement using Expert Networks

Cathie LeBlanc
Department of Computer Science
Florida State University
Tallahassee, FL 32306

R.C. Lacher
Department of Computer Science
Florida State University
Tallahassee, FL 32306

Kristin Adair
Department of Computer Science
Florida State University
Tallahassee, FL 32306

John W. Elling
Measurement Technology Group
Los Alamos National Laboratory
Los Alamos, NM 87545

Susan I. Hruska
Department of Computer Science
Florida State University
Tallahassee, FL 32306

Abstract

Expert networks are networks of neural objects derived from expert systems. The hybrid nature of such networks allows the expert knowledge to be refined and augmented using sample data. The benefit of combining expert systems with neural network-like learning from data has been illustrated in a number of diagnosis and decision-making problem domains. The ability of these systems to learn illuminates knowledge previously unknown to human experts. This paper contains a synopsis of work in expert networks over the last several years, some of the findings we have found useful and interesting, and an indication of current directions for this research.

1 BACKGROUND

Knowledge acquisition and engineering are integral parts of any attempt to encode human expert knowledge in a reasoning system. In the creation of rule-based expert systems, for example, the knowledge engineer must extract not only rules about the problem domain but also, in most cases, some measure of certainty for each of those rules. The human expert must quantify his or her certainty about the rules, a difficult process. In addition, in many cases, even after the knowledge engineering phase, the rule-based system does not perform at the level of the human expertise.

Many researchers have developed hybrid tools which combine the pattern recognition capabilities and parallel processing of neural systems while retaining the domain knowledge encoded in expert systems (Medsker, 1995). *Expert networks* boost the process of extracting rules, refining rules, and adjusting system performance by allowing pieces of the expertise to be learned from example data.

1.1 WHAT IS AN EXPERT NETWORK?

Traditional neural networks can be generalized into a broader class of networks called *computational networks*. A computational network is essentially a directed graph in which each component has data processing functionality. Computational networks are not required to have any sort of layer structure. An *expert network* is a computational network which can be derived from expert knowledge (Lacher and Nguyen, 1994). How can an expert network be derived from expert knowledge?

1.2 EQUIVALENCES

The derivation of expert networks from expert knowledge is based in part on the type and form of knowledge to be encoded, and the reasoning engine to be used with the knowledge. In our work, the earliest expert networks were built from Emycin-based (Buchanan and Shortliffe, 1984) rule-based expert systems. To create an expert network from a rule-based expert system, the rules are parsed into antecedents, consequents and certainty factors. Assertions (the antecedents and consequents) become nodes in the expert network. Nodes representing the antecedent and consequent from an actual rule in the expert system are connected via a weighted connection. The certainty factor associated with the rule serves as the initial weight placed on the connection between the nodes (Kuncicky, *et al.*, 1992). Figure 1 illustrates some of the types of simple and complex transformations from rules into a network topology.

Whenever rules in the rule base contain **AND**, **OR**, or **NOT**, the node that represents the operation is broken down into its constituent parts. A node is created for each of these parts and a separate node is created for the operation. The connections between the constituent parts and the operation node are given a weight of 1.0. These connection weights will be unchanged during training and are thus called *hard* connections; all other connections are trainable, and are called *soft* connections. The last rule in Figure 1 illustrates this type of transformation (Kuncicky, *et al.*, 1992).

Note that transformation from expert system rules results in a somewhat nontraditional network, one which has no identifiable layer structure and is sparsely connected. In addition, the combining, firing and output functions for each node in the expert network are derived from the inference engine dynamics of the expert system shell in which the expert system was defined (Hruska and Kuncicky, 1992). For example, if the expert system shell is M.1 (Teknowledge, 1986), a node which corresponds to an **AND** would have a firing function which outputs the minimum of its inputs because this is how the inference engine of M.1 implements **AND**.

1.3 LEARNING ALGORITHMS

Expert Network Back-Propagation (ENBP) is a learning algorithm similar to backpropagation seen in standard neural networks. Lacher and Nguyen have shown

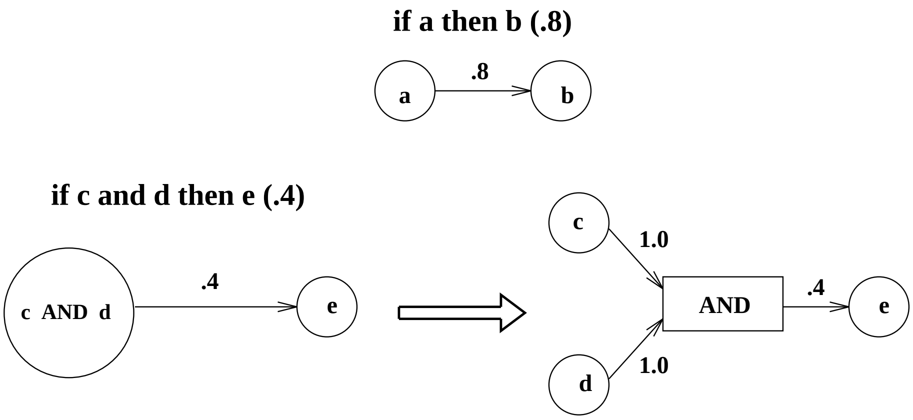

Figure 1: Example of how an expert network is created from a rule base

that standard back-propagation can be generalized to computational network back-propagation (CNBP) (Lacher and Nguyen, 1994). ENBP is then an instantiation of computational network back-propagation. The mechanics of training using ENBP will depend entirely upon the inference engine used to reason with the expert knowledge. Therefore, if the inference engine associated with the expert knowledge is a custom one, then the training algorithm for the expert network must also be customized.

ENBP, like traditional backpropagation algorithms, depends on the notion of back-propagation of error. A number of crucial differences between expert networks and traditional neural networks exist that will affect the manner in which error is back-propagated. First, expert networks are not required to process information via layer structure, but rather draw their structure from the chaining together of rules. Some reasoning pathways from input to output may be very short, while others are quite long. Second, each node in the expert network has unique, identifiable functionality. Each node represents a particular piece in the expert knowledge.

In order to generalize the back-propagation algorithm to computational networks, one must decouple the process of node error assignment from the weight correction step because a computational network does not necessarily have a layer structure. Adding the notion of influence factors allows such decoupling to occur. The influence factors are calculated during forward activation of the computational network and are associated with synaptic connections. The influence factor determines the influence of error in the output of node j on the error in the output of its downstream neighbor, node k (Lacher, 1993).

The influence factor, ϵ, is defined to be:

$$\epsilon_{kj} = \frac{\partial z_k}{\partial z_j} = \varphi_k^{'}(y_k) \times \frac{\partial \Gamma_k}{\partial x_{kj}}(x_{k1}, ..., x_{kn}) \times \sigma_{kj}^{'}(z_j), \tag{1}$$

where z_i is the output of node i, φ_k is the activation function of node k, y_k is the internal state of node k, Γ_k is the combining function of node k, $\mathbf{x}_k$ is the postsynaptic input vector to node k, and σ_{kj} is the synaptic function of the weighted connection from node j to node k. In other words, the influence factor determines how the output of node k changes with respect to changes in the output of node j, its immediate upstream neighbor. For a typical sigmoid unit, $\Gamma(\mathbf{x}) = \sum x_i$, making $\frac{\partial \Gamma}{\partial x_{kj}} = 1$. In expert network nodes, Γ can be much more complex, making the influence factor become a soft, or even crisp, multiplexing switch for reverse error assignment (Lacher, *et al.* 1992).

The ENBP algorithm works well whenever initialized sufficiently near an optimal knowledge state. (Often, starting with the correct sign on all certainty factors suffices.) In cases where no prior knowledge is available to create an initial knowledge state, global search methods have been used successfully. In particular, the Goal-Directed Monte Carlo [GDMC] learning method of Kuncicky (Kuncicky, 1991) works well in conjunction with ENBP in a two stage learning algorithm. In the first stage, GDMC is used to broadly choose where a good starting point for the weights might be. GDMC reduces error an average of one order of magnitude. At that point, ENBP is used to reduce error an additional two to three orders of magnitude.

As a two-stage training process, GDMC and ENBP have been tested extensively using an expert system which captures the expertise needed to choose a wine to go with a particular meal. This system, called the Wine Advisor, is fielded as an M.1-based system with 44 rules. Translating to an expert network results in a net with 97 nodes and 175 connections, 64 of which are soft. In one test, 25 soft connections were ablated to zero. GDMC is used to reduce error by a factor of ten, after which ENBP is applied, resulting in a reduction of error below the threshold of accuracy in the inference system. This training was carried out using only 22 training exemplars. The resulting knowledge state is perfect: the re-vitalized system reasons correctly on all of its more than 6900 valid inputs (Hruska, *et al.*, 1991).

Other applications of the expert network technology have been made in an expert system for advising engineers on the appropriate control chart to be used in an automated manufacturing environment (Hruska and Kuncicky, 1992). Studies of training sample selection, adding pathways for fault tolerance, refining ENBP and GDMC in various ways, and in developing visualization/interface tools for expert networks also form part of the body of work done on various applications of this technology.

2 CURRENT WORK

2.1 CONTAMINANT ANALYSIS EXPERT NETWORK

An extensive study of expert networks applied to the problem of instrument fault diagnosis has taken place over the last three years involving scientists at Los Alamos National Laboratory, Idaho National Engineering Laboratory, Varian Chromatography Systems, and Florida State University. The system developed is part of the push to automate the laboratory analysis of contaminants in soil, a critical task in the Department of Energy's environmental analysis mission. The expert network aids in automated diagnosis of machine faults in gas chromatography. This system uses a knowledge table to capture expertise rather than a traditional rule-based expert system. In the knowledge table, linguistic qualifiers relate symptomatic features of the graph recorded by the instrument to faults in the chromatograph or sample preparation. Diagnoses of the possible machine or sample faults are presented to the user in a ranked list, based on processing of evidence from the symptoms by the inference engine created for this problem. Automated tools for constructing the expert network from the table and refining both the structure and the inferencing of the system are now in use. Performance results of this expert network on a functioning chromatograph have shown the efficacy of this technology in this quite complex problem-solving domain (Adair, *et al.*, 1996).

As illustration of the architecture of this expert network, Figure 2 shows just the subnetwork used to diagnose the fault $ContaminatedSample$. For each symptom, the network contains one input node. For each symptom/fault combination, the network contains a filter node for each semantic qualifier that relates the symptom to the fault. The possible semantic qualifiers are $Always$, $Usually$, $Sometimes$, $Infrequently$, and $Never$. For each symptom set/fault combination, the network contains one combination node which will take input from the symptom set and combine the evidence for or against the particular fault. Finally, each fault is represented by one node in the output layer of the expert network. Because the expert reasoning uses a customized inference engine, a customized, back-propagation-like learning algorithm for the expert network is required (Levis, 1995).

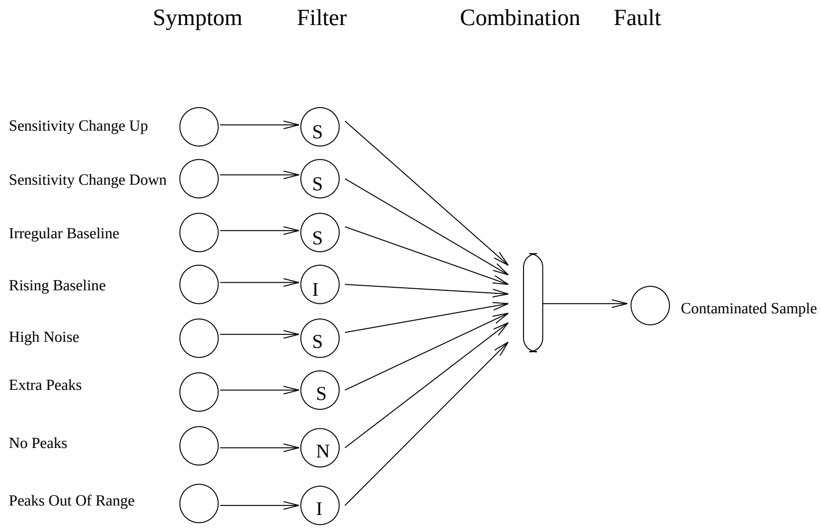

Figure 2: Portion of expert network for $ContaminatedSample$

2.1.1 Refinement Using Neural Network Clustering

Part of the challenge of automating this diagnosis problem is resolving the differences between historical recollections of human experts and the actual data gathered from instruments on which the fault was induced. A typical iteration of the refinement process includes building the expert network from the knowledge table, training it with available data, analyzing which faults are not diagnosed correctly, and going back to the expert for further clarification. In the clarification stage of this process, we have developed a number of techniques and tools which aid in rule refinement, discovery, and alternate pathway suggestion.

Focusing on one of these techniques, consider the problem of encoding the experts' qualification that several symptoms are $Sometimes$ associated with a particular fault. The human expert is able to say that each symptom is related to the fault, but may not be able to state how the symptoms are related to one another. However, relationships among symptoms can be gleaned from the data.

For example, during the knowledge acquisition process, the expert suggests that when fault A occurs, sometimes symptom B occurs, sometimes symptom C occurs, sometimes symptom D and sometimes symptom E occurs. The data, however, reveals that symptoms B and C occur together in many cases for this fault and symptoms B, D and E also regularly appear together when fault A occurs.

A methodology using ART neural networks (Carpenter and Grossberg, 1991) to cluster symptom data for a particular fault into separate symptom sets has proven useful (Adair, 1996). Using multiple network reasoning paths built from these separate symptom sets, we can suggest the removal of some connections and the addition of other connections. The symptom sets and faults can then be fed into NetMedic (Timpany, 1995), which uses standard statistical analysis of data to suggest which semantic qualifiers should be associated with network connections.

Figure 3 shows the subnetwork created to diagnose *ContaminatedSample* based on suggestions from the clustering methodology described above. Note that Figure 2 and Figure 3 each have the same set of symptoms leading to the same fault but that Figure 3 shows two paths leading to the fault, each path containing a different symptom set along with different semantic qualifiers. The network incorporating Figure 2 was able to correctly identify 41.6% of the samples before training occurred. Adding the multiple paths of reasoning as shown in Figure 3 increased the correct identification rate to 71.2% before training (Adair, 1996).

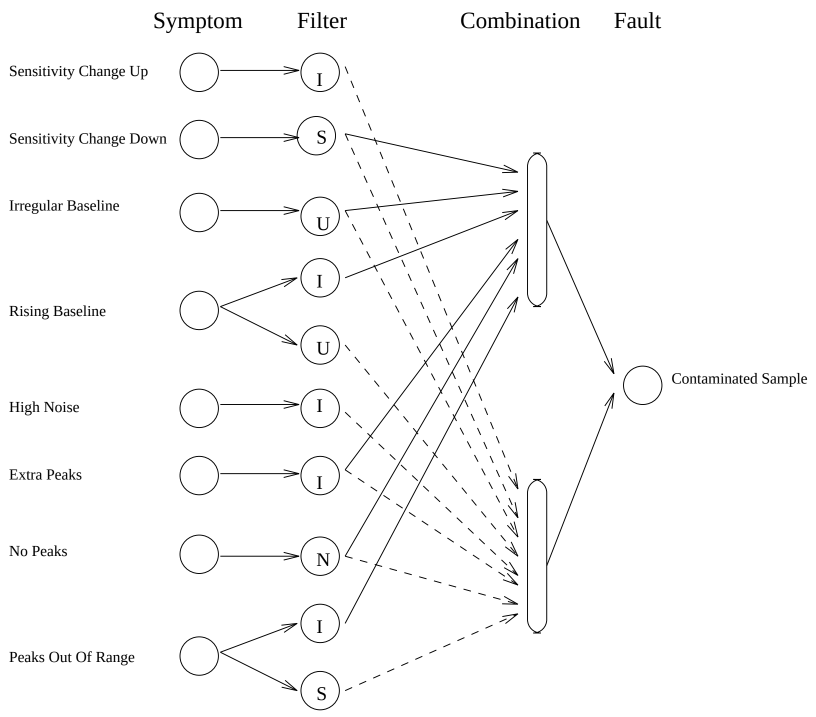

Figure 3: Network with multiple paths of reasoning

The discovery of alternate reasoning pathways is an example of how intelligent techniques help the human experts reason about the domain in new and different ways. Our methodology lends insight into how symptoms relate to each other which can then be incorporated into the human expertise, and which can also be used to dramatically improve the diagnostic capabilities of the automated system.

2.2 RECURRENT PROTEIN FOLDING EXPERT NETWORKS

Unlike the applications presented above, there are sets of classification tasks which do not lend themselves to use of a strictly feed-forward (acyclic) architecture. In particular, the solution to many problems requires that *state* information be maintained. State information is the context in which the problem is currently being solved. The context of a problem solution changes as the solution proceeds, and sometimes certain sets of rules need only be considered in certain contexts.

Such state information is difficult to manage using feed-forward architectures. Because of this fact, we are currently investigating the extension of expert networks to contain recurrent connections. An application area where such information is critical is protein secondary structure prediction. In this problem, one part of the fold of a protein is influenced by other, perhaps quite distant, regions of the protein. We have designed an expert network for protein secondary structure prediction based on well-known rules for folding, which is then refined by learning from data (LeBlanc, 1996). In this problem domain, theoretical and practical extensions to expert network technology again go hand in hand with insight into the problem being solved. Protein secondary structure prediction is another problem for which human expertise is imperfect, algorithmic solutions are incomplete, and the need for knowledge discovery is imperative.

3 CONCLUSIONS

Using expert networks to represent expert knowledge and to refine that knowledge by learning from data has proven to be a useful technique in a number of application areas. We have found that the use of expert network technology ameliorates the knowledge acquisition bottleneck by allowing knowledge to be learned from data. In addition, the knowledge learned from data often represents knowledge previously unknown to the human experts and thus, the knowledge augments the human expertise.

Using the expert network to provide information about where the encoded knowledge is incomplete or incorrect is of great importance in our work. Expert networks can provide an explanation facility similar to that of expert systems, allowing users to understand **why** the network comes to its conclusions. In addition, knowledge engineering tools based on statistical analysis, clustering, and neural network training can yield further insights into the problem domain.

Expert network technology has evolved over the years from its roots in rule-based expert systems. Expert networks can be used in a wide array of situations, including those where no expert system currently exists. In addition, we are currently removing the acyclicity requirement for expert network topologies. Expert networks with recurrent connections are being fielded on a class of problems which require maintenance of state information about the problem solution.

As we have shown in a number of application areas, expert networks provide a framework for the incorporation of multiple types of reasoning in a single system. We have created reasoning systems which combine clustering techniques, data-driven learning, rule-based structures, and data mining techniques. The combination of these various techniques results in robust systems which contain more complete knowledge about the problem domain than was available before the development of the system.

References

[1] Kristin L. Adair. *Data Driven Optimization of Expert Networks.* Master's thesis, Florida State University, 1996.

[2] Kristin L. Adair, Alan P. Levis, and Susan I. Hruska. An expert network development environment for automating machine fault diagnosis. In *Proceedings of SPIE AeroSense '96*, volume 2760, pages 506–515, Bellingham, WA, April 1996. International Society for Optical Engineering.

[3] B.G. Buchanan and E.H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA, 1984.

[4] Gail A. Carpenter and Stephen Grossberg. ART2: Self-organization of stable category recognition codes for analog input patterns. In Gail A. Carpenter and Stephen Grossberg, editors, *Pattern Recognition by Self-organizing Neural Networks*, chapter 12, pages 397–423. MIT Press, Cambridge, Massachusetts, 1991.

[5] S.I. Hruska, D. Kuncicky, and R.C. Lacher. Hybrid learning in expert networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume II, pages II117–II120, Piscataway, NJ, July 1991. IEEE.

[6] Susan I. Hruska and David C. Kuncicky. Automated knowledge refinement for control chart selection. *Heuristics*, 5:101–7, 1992.

[7] David C. Kuncicky. *Isomorphism of Reasoning Systems with Applications to Autonomous Knowledge Acquisition.* PhD thesis, Florida State University, 1991.

[8] D.C. Kuncicky, S.I. Hruska, and R.C. Lacher. Hybrid systems: the equivalence of expert system and neural network inference. *International Journal of Expert Systems*, 4:281–297, 1992.

[9] R.C. Lacher. Expert networks: Paradigmatic conflict, technological rapprochement. *Minds and Machines*, 3:53–71, 1993.

[10] R.C. Lacher, S.I. Hruska, and D.C. Kuncicky. Backpropagation learning in expert networks. *IEEE Transactions on Neural Networks*, 3:62–72, 1992.

[11] R.C. Lacher and K.D. Nguyen. Hierarchical architectures for reasoning. In R. Sun and L. Bookman, editors, *Computational Architectures for Integrating Neural and Symbolic Processes*, pages 117–150, Boston, 1994. Kluwer Academic Publishers.

[12] Cathie LeBlanc. Recurrent expert networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, volume 2, page 1366, Menlo Park, CA, August 1996. American Association for Artificial Intelligence, AAAI Press/MIT Press.

[13] Alan P. Levis. *Inferencing and Training in a Development Environment for Table-based Expert Networks.* Master's thesis, Florida State University, 1995.

[14] Larry R. Medsker. *Hybrid Intelligent Systems.* Kluwer Academic Publishers, Boston, 1995.

[15] Teknowledge. *M.1 Reference Manual.* Teknowledge, Palo Alto, CA, 1986.

[16] Robert G. Timpany. *Data Assisted Knowledge Acquisition for Alteration of Table-based Expert Networks.* Master's thesis, Florida State University, 1995.