N92-16593

EXPERT NETWORKS IN CLIPS[1]

S.I. Hruska, A. Dalke, J.J. Ferguson, and R.C. Lacher

Department of Computer Science
Florida State University, Tallahassee

Abstract. Rule-based expert systems may be structurally and functionally mapped onto a special class of neural networks called *expert networks*. This mapping lends itself to adaptation of connectionist learning strategies for the expert networks. Following a process introduced by Kuncicky, Hruska, and Lacher, a parsing algorithm to translate CLIPS rules into a network of interconnected assertion and operation nodes has been developed. The translation of CLIPS rules to an expert network and back again is illustrated. Measures of uncertainty similar to those used in MYCIN-like systems are introduced into the CLIPS system and techniques for combining and firing nodes in the network based on rule-firing with these certainty factors in the expert system are presented. Several learning algorithms are under study which automate the process of attaching certainty factors to rules.

EXPERT NETWORKS

The idea of mapping an expert system onto a neural network in order to make the most of what each technology offers is a timely one [Fu and Fu 1990][Gallant 1988] [Hall and Romaniuk 1990] [Kuncicky 1990] [Kuncicky et al. 1991] [Lacher et al. 1991a] [Towell et al.1990]. Hruska, Kuncicky, and Lacher define the special class of networks created from an expert system as *expert networks*. In these systems, rules of the form *if A then B* where A and B are assertions are mapped in an intuitive manner to nodes labelled A and B with a forward connection tying the two together. In expert systems which incorporate uncertainty, the certainty factor, a value between -1 and $+1$, is mapped to the connection strength in the network.

The functioning of the network is also based on the expert system, with the combining and firing functions of the network nodes derived from the inference engine functions for the expert system. Inference engine functions which have counterparts in the network include those for combining evidence for rules that have consequents in common, methods of handling conjunction and negation in rules, thresholding functions, and those functions that govern firing of rules. The nodes are not simple perceptron-type nodes, but rather are more complex, based on how the expert system works internally. When activated with input, an expert network functions identically to the expert system inference engine operating on the rule base from which the network was derived.

One of the most compelling reasons for pursuing the mapping of expert system concepts onto neural networks is the possibility for introducing connectionist learning techniques into traditional AI systems. Automation of even part of the expensive and time-consuming knowledge acquisition process is a valuable contribution of this technology.

[1] Research partially supported by the Florida High Technology and Industry Council, the US Office of Naval Research, and Oak Ridge Associated Universities.

There are several levels of knowledge acquisition for an expert system [Hruska et al. 1991a][Hruska et al. 1991b]. These include refining or adjusting the certainty factors for rules, forming new rules from existing concepts, and forming new concepts. Ideally, most of the knowledge acquisition process could be done directly from data. This training data consists of examples of input information and the associated desired (correct) output. Bypassing the knowledge elicitation process in which a human expert must express their expertise in the form of if-then rules is an ultimate goal. Automation would be useful at any of the levels listed above.

The current research efforts reported in this paper focus on the area of automating the refinement of knowledge. Certainty factors represent perhaps the most subtle form of knowledge in a system, the confidence which the expert has in the consequent being related to the antecedent of the rule. In decision-making, the use of certainty factors enables the system to rank outcomes of the system based on the certainty factors attached to them. In CLIPS, we are in the process of implementing certainty factors via the *declare* option available in CLIPS rule formats.

THE TRANSLATION PROCESS

In order to apply the connectionist learning techniques under development, it is useful to translate the expert system rule base to expert network format. The process of translating an expert system rule base to the type of expert network described above was prototyped by Kuncicky and colleagues [Kuncicky 1990] [Kuncicky et al. 1991] for the expert system M.1 [M1 1986]. Following these ideas, an implementation of the translation process was written for a subset of CLIPS. Preliminary work on this project is summarized in [Johnson and Franke 1989].

The translation is performed in two steps. First, the rule base is simplified somewhat by replacing rules of designated complexity (such as those with disjunctions in their antecedents) with several simpler but collectively equivalent rules. For example, rules (with cf certainty factors) of the form

if A or B then C (cf)

are simplified to

if A then C (cf)

if B then C (cf).

Another type of rule base simplification is reduction of the rule

if A and (B or C) then D (cf)

to

if A and B then D (cf)

if A and C then D (cf)

which distributes conjunction over disjunction. Rules of the form

if not(A or B) then C (cf)

are simplified via DeMorgan's law to

if not A and not B then C (cf).

Simplification of conjunction in the consequent converts rules of the form

if A then B and C (cf)

to the equivalent set of rules

if A then B (cf)

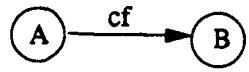
if A then C (cf).

Through this process, disjunctions in the antecedents and conjunctions in the consequents are eliminated. Disjunctions in the consequents are not allowed.

After the rule base simplification transformations are complete, the rule base is translated into an expert network. At this point, there are three basic forms of rules: regular rules, rules with conjunctions in the antecedent, and rules with negations in the antecedent. These three types of rules may be compounded to form rules of arbitrary complexity in the antecedent. a regular rule, of the form

if A then B (cf)

is converted to a portion of an expert network as

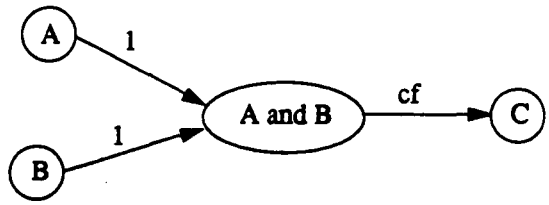


Rules with conjunctions and negations involve creation of special operator type nodes. Connections from assertions to these operator nodes are fixed with a weight of 1.0.

Typically, a rule involving a conjunction such as

if A and B then C (cf)

is converted to



while rules with negations of the form

if not A then B (cf)

become

A short example of the translations process in CLIPS is shown in Tables 1 and 2 and Figure 1.

As of this date, the translation program handles input rule bases which contain defrules, deffacts, and asserts. Plans are now underway to extend the flexibility of this prototype to accept variables, pattern matching, and retraction.

```
(defrule clear
(declare (certainty 0.96))
; If the barometer is high and there are no clouds then it is clear.
     (and (high)
          (not (cloudy)))
=>
     (printout t "Clear" crlf)
)
(defrule sky-water
(declare (certainty 0.7))
; If the barometer is low or it is cloudy then it is rainy.
     (or (not (high))
         (cloudy))
=>
     (assert (precipitation))
)
(defrule snowy
(declare (certainty 0.8))
; If there is precipitation but it is not hot then time for snow.
     (and (precipitation)
          (not (hot)))
=>
     (printout t "Snow!" crlf)
)
(defrule rainy
(declare (certainty 0.9))
; If there is precipitation and it is hot then it must be rainy.
     (and (precipitation)
          (hot))
=>
     (printout t "Get your umbrellas." crlf)
)
```

Table 1. Original Rule Base

```
(defrule clear-1
(declare (certainty 0.96))
; From rule number 1
     (high)
     (not (cloudy))
=>
     (printout t "Clear" crlf)
)
(defrule sky-water-2
(declare (certainty 0.7))
; From rule number 2
     (not (high))
=>
     (assert (precipitation))
)
(defrule sky-water-3
(declare (certainty 0.7))
; From rule number 2
     (cloudy)
=>
     (assert (precipitation))
)
(defrule snowy-4
(declare (certainty 0.8))
; From rule number 3
     (precipitation)
     (not (hot))
=>
     (printout t "Snow!" crlf)
)
(defrule rainy-5
(declare (certainty 0.9))
; From rule number 4
     (precipitation)
     (hot)
=>
     (printout t "Get your umbrellas." crlf)
)
```

Table 2. After Rule Base Transformations

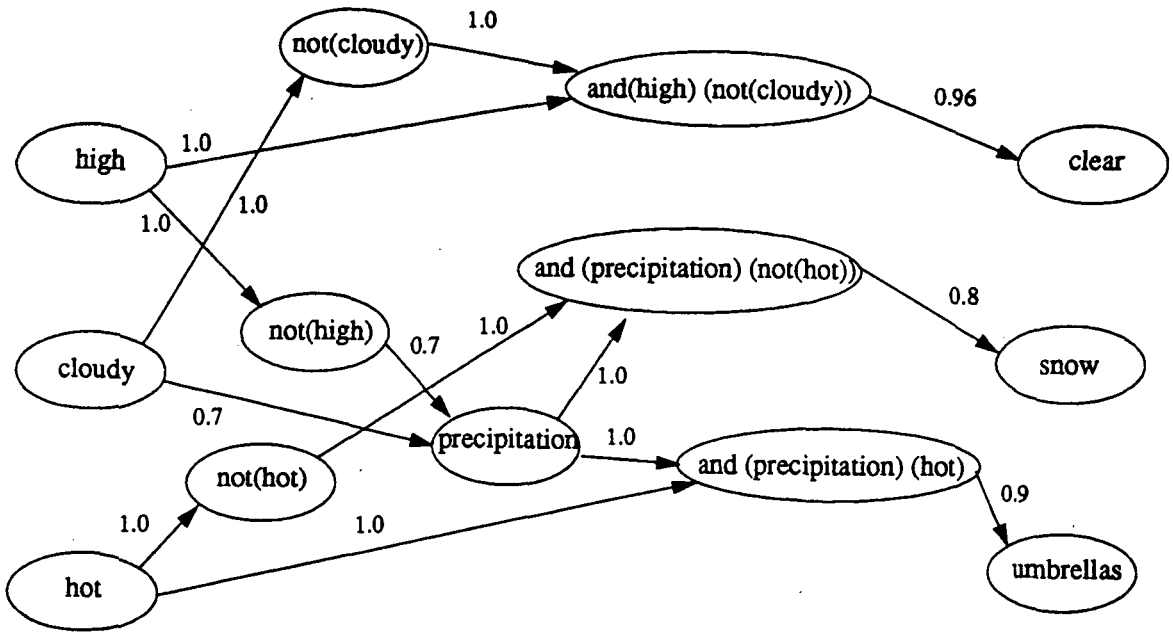


Figure 1. Expert Network From Rule Base

PROCESSING CERTAINTY FACTORS IN CLIPS

The study of M.1 and its manner of handling certainty factors has led us to contemplation of improvements which could be made in porting this process to CLIPS. Our introduction of certainty factors into CLIPS rule bases led to the necessity of altering the CLIPS source code to implement the processing of certainty factor information. This includes a set of algorithms for evidence accumulation, combining evidence, outputting a certainty factor value for the rule, and firing the rule based on that certainty factor output. One possibility, currently under consideration by Lacher and Traphan [Traphan 1991] is to build and improve on the inference engine for the expert system shell EMYCIN (upon which M.1 is also based). EMYCIN uses a threshold of 0.2 to determine whether a rule will fire or not. The proposal is to smooth this thresholding function, resulting in a system which yields analog values very close to those of the original EMYCIN system, but which is continuously differentiable. This feature will enable gradient descent learning techniques to be applied in a more consistent manner over the entire range of values. This work is currently in progress.

Learning techniques developed for expert networks include Goal-Directed Monte Carlo Search [Hruska et al. 1991b] and Expert System Backpropagation [Lacher et al. 1991b]. The first of these is designed using the principles of reinforcement learning with increasing levels of noise applied to the connections between nodes. The second is an adaptation of the standard backpropagation of error learning algorithm which uses gradient descent to find the weight settings (certainty factors) which minimize the difference between the system's output and the ideal (correct) output. The major innovation of Expert System Backpropagation is the use of the expert system's inference engine functions in the complex nodes of an expert network to perform the computations necessary for

gradient descent. Both of these learning algorithms have been tested on the M.1 system and are currently being upgraded to work with the altered version of CLIPS described above.

CONCLUSION

A system for translating a subset of CLIPS expert system rule bases into expert networks has been prototyped and is under ongoing development. CLIPS rules will have certainty factors attached to them which may be used to express uncertainty in the inferencing process. A proposal for smoothing the evidenciary combining and firing functions of traditional EMYCIN-like systems is described. Learning algorithms for expert networks are briefly described. As work progresses on this system, we draw nearer to realization of a tool for automating the knowledge acquisition process involved in traditional AI systems.

REFERENCES

Fu, L.M. and L.C. Fu (1990). Mapping rule-based systems into neural architecture, *Knowledge-Based Systems*, Vol 3, No 1.

Gallant, S.I. (1988). Connectionist expert systems, *Communications ACM*, **24**, 152-169.

Hall, L.O. and S.G. Romaniuk (1990). FUZZNET: Toward a fuzzy connectionist expert system development tool, *Proceedings IJCNN 90* (Washington, DC), vol. II, 483-486.

Hruska, S.I., D.C. Kuncicky, and R.C. Lacher (1991a). Learning in acyclic expert networks, *Proceedings of WNN-AIND 91*, February 1991.

Hruska, S.I., D.C. Kuncicky, and R.C. Lacher (1991b). Hybrid learning in expert networks, *Proceedings of IJCNN 91, Seattle*, July 1991.

Johnson, J. and J. Franke (1989). Design specification for CLIPS parser (CP ver. 0.5B), Undergraduate Honors Project for S.I. Hruska, Florida State University, Fall, 1989.

Kuncicky, D.C., S.I. Hruska, and R.C. Lacher (1991). Hybrid systems: The equivalence of rule-based expert system and artificial neural network inference, *International Journal for Expert Systems*, accepted with revisions, May 1991.

Kuncicky, D.C. (1990). Isomorphism of reasoning systems with applications to autonomous knowledge acquisition, PhD dissertation, (R.C. Lacher, major professor), Florida State University, December, 1990.

Lacher, R.C., S.I. Hruska, and D.C. Kuncicky (1991a). Expert networks: A Neural Network Connection to Symbolic Reasoning Systems, *Proceedings of FLAIRS-91*, April 1991.

Lacher, R.C., S.I. Hruska, and D.C. Kuncicky (1991b). Backpropagation learning in expert networks, *IEEE Transactions on Neural Networks*, accepted with revisions, May 1991.

M.1 Reference Manual (1986). (Software version 2.1), Teknowledge, Palo Alto, CA.

Towell, G.G., J.W. Shavlik, and M.O. Noordewier (1990). Refinement of approximate domain theories by knowledge-based neural networks, *Proceedings of AIII-90, Eighth National Conference on Artificial Intelligence*, July 1990.

Traphan, B. and R.C. Lacher (1991). Smoothing EMYCIN for backprop learning, work in progress, June 1991.