

in "Parallelization in Inference Systems," Proceedings of the International Workshop,
Dagstuhl, Germany, December 1990, Lecture Notes in Artificial Intelligence 590,
B. Fronhöfer and G. Wrightson (Eds.), Springer Verlag, pp. 303-317.

APPLICATION OF CONNECTIONIST MODELS TO FUZZY INFERENCE SYSTEMS

C. TOUZET & N. GIAMBIASI

LERI

(Laboratoire d'Etude et Recherche en Informatique)

Parc Scientifique Georges Besse

30000 Nîmes, FRANCE

Tel. : (+33) 66.38.70.29, Email : touzet@eerie.fr

Abstract

In this paper, we will try to shed light on the usefulness of neural networks by describing an application which combines connectionism and ruled-based systems. In the present fuzzy ruled production systems, propagation of uncertainty coefficients is carried out by means of computational formulae stemming from mathematical models of fuzzy reasoning. But the use of a formula provided by a general abstract model, and not intimately related to the application, can lead us to a fuzzy procedure not reflecting the fuzzy reasoning of the human expert. The connectionist approach proposed here solves this problem of fuzzy inference. An uncertainty propagation rule specific to the application domain is determined by learning from examples of fuzzy inferences.

Key words

Expert system - Production rule - Fuzzy inference - Connectionism - Backpropagation.

Introduction

Today, some people consider artificial neural networks as the panacea to all unresolved problems in artificial intelligence. Contrary to this, others describe a phenomenon of fashion without a future. The great majority do not take sides in the discussion, and are waiting for an answer to the principal question : What is the use of artificial neural networks? There are clearly other additional questions reflecting the curiosity of everyone: What are neural networks? How do they work? What are their applications? What are the practical problems of their use?

Since our goal is to answer these questions as well as possible, we will propose an illustrative utilisation of neural networks to resolve the problem of fuzzy inference. We have deliberately chosen to couple connectionism and artificial intelligence, fields often considered as competing, so as to prove that they are not exclusive of one another. The experiment is a typical example of a hybrid system [Amy & al. 90] making the best use of the advantages of both fields. The objective is to provide more specific expert systems of the application domain which will then be more performant.

The first part will serve as a reminder of artificial neural networks. In particular, the mechanisms involved in the selection of a potential application will be specified. The connectionist construction will be illustrated in detail concerning the problem of fuzzy inference as expressed by expert systems. We will describe the results obtained for two well-known fuzzy inference rules, those of Lee and MYCIN.

1 Artificial neural networks

Connectionism is a fairly new discipline which proposes numerous models and definitions. Those interested in an extensive overview of the field may refer to [Kohonen 88], [Karna & al. 89] and [Dayhoff 90].

Definition :

It is useful to recall the scientific context at the origin of connectionism. The central nervous system demonstrates the ability to process information which is far out of the reach of present-day computing techniques. The commonly admitted hypothesis is that brain-processing is linked to its structure. With the goal being to increase the abilities of computing systems, brain-like or brain-inspired computers have been under development for several decades now [McCulloch & Pitts 43] [Hebb 49]. Such models, inspired by the architecture and the processing of the central nervous system, are called artificial neural networks.

"Artificial neural networks are massively parallel interconnected networks of simple processing elements and their hierarchical organizations. These processing elements modelize a simple behaviour of the biological neuron."

Artificial neural networks have already achieved their goal in part by exhibiting properties such as learning by example, tolerance to faults in the network and to noise on the inputs and the possibility of parallel processing.

Subsequently, from among the one hundred of different connectionist models currently proposed [Hecht-Nielsen 89], we will focus to supervised learning models.

Description :

Each simple processing element of the neural network exchanges information with the other connected elements by sending its activation level. The activation value of a cell is computed by a transfer function from the weighted sum of its inputs. The behaviour of the network is determined by the transfer functions of its elements, the topology and the connection weights. A neural model can also be completely described by the specifications of the following three elements:

- the processing element,
- the topology of the network,
- the learning algorithm.

The processing element is called alternately : elementary processor, cell, neuron or unit. Each cell computes its state using an activation function based on the information received. In the case of the model used here, the cell state is computed from the weighted sum of the inputs using a continuous function of the sigmoidal type.

The topology of the network is described by the interconnection scheme of the processing elements. In this presentation, the network has a multilayer architecture. Cells activated by at least one input are part of the input layer. Cells whose values are accessible constitute what is called the output layer. Since all the other cells have no connection to the outside, they are called hidden cells. All connection weights are variable.

The learning algorithm allows us to adapt the network behaviour to the application. With this end in mind, the network weights are adjusted so as to gear the network response to that desired for the learning examples. An example of this is an input pattern associated with an output pattern. The input pattern is coded as an activation vector on the input cells. The coding is the affectation operation beginning with the input pattern to the activation vector. According to custom, the input pattern is improperly confused with the activation vector of the input cells and, likewise, the output pattern with the activation vector of the output cells. The most widely used learning algorithm for multilayer networks is the so-called gradient backpropagation technique.

The simulation of a neural model on a sequential computer imposes the selection of a mode for updating the cell states (synchronous or asynchronous, for example). The adopted simulation process is a priori independent of the network model. Nevertheless, it usually interacts with the observed performances and thus is most often implicitly associated with the model.

2 Application domains

Application domains are: pattern recognition, signal processing, diagnosis, etc. In fact, an application is considered potential if it can be represented as a mapping between two spaces and if there is a sufficient number of representative examples of the mapping.

It should be pointed out that until the gradient backpropagation algorithm became widely used (in 1985), realizable functions were restricted to linear functions [Minsky & al. 88]. This learning algorithm for multilayer networks allows us to consider non-linear problems, the most representative of which is the exclusive or. The Darpa [DARPA 88] lists a certain number of demonstrative applications based on this paradigm.

The term "application" should however be used with care. As a matter of fact, one must distinguish between "candidate applications" which can, in principle, be solved by the connectionist technique, "developping applications" whose feasibility has been demonstrated on a toy problem and "proven applications" which are not numerous.

Determining a potential application

Neural networks can be viewed as systems with learning abilities in mapping whatever the field may be. For example, le Cun [Le Cun 87] proposed the use of connectionist techniques to solve problems in medical diagnosis of emergency cases. In this experiment, the network mapped the

Approximation of a mathematical function or mapping by neural networks

Theorem [Hecht-Nielsen 89] :

"Given any $\epsilon > 0$ and any L_2 function $f : [0, 1]^n \rightarrow \mathbb{R}^m$, there exists a three-layer backpropagation neural network that can approximate f to within ϵ mean squared error accuracy."

The space L_2 includes every function that could ever arise in a practical problem (continuous or discontinuous). It is important to realize that this theorem does not comment on the number of units in the hidden layer. On the other hand, if one hidden layer is a minimum, then, in practice, two or three layers are often used. Finally, although this theorem guarantees the ability of a multilayer network with the correct weights to accurately implement an arbitrary L_2 function, it does not give any idea regarding the learning law needed to compute these weights.

Nature of the developed applications

We have not yet described in precise terms the developed applications. For the great majority, these applications are purely connectionist. There are several reasons for this with the least justifiable being that the developer may not possess enough competency in the other domains to plan a coupling with more conventional techniques. Secondly, a purely connectionist development emphasizes the neural possibilities more advantageously. Last but not least, to couple classic and connectionist techniques is a difficult problem. However really interesting applications are to be found in this direction. Neural networks have not been around long enough to be able to propose a complete solution. On the contrary, artificial intelligence exhibits real-world applications resulting from years of research and development. It seems opportune to try to involve neural networks in these systems, so as to generate hybrid systems. Our study falls into this framework.

From among a certain number of unresolved problems in artificial intelligence, one has been selected for which we propose a connectionist solution.

3 One unresolved problem in artificial intelligence

In a rule-based system, the knowledge base is divided into two parts: the rule base and the fact base. The rule base contains knowledge expressed in the following form: If <condition> Then <action>. The inference engine is charged with executing the rules and facts. In simpler terms, its work consists in selecting and then applying rules whose left part agrees with the established facts. Generally in this kind of system, rules act as logic implications between propositions or predicates. Thus, reasoning is reduced to a succession of logic deductions. In certain applications, it is necessary to process unreliable, imprecise or incomplete knowledge. In such a case, we speak of fuzzy reasoning or, in more precise terms, non-exact reasoning.

Representation and processing of fuzzy knowledge

In the case of expert systems, the notion of fuzziness is generally modeled by introducing uncertainty coefficients taken in the real interval $[0,1]$. 0 means certainly false, 1 means certainly true and intermediary values mean more or less true/false. These uncertainty coefficients are associated together with the established facts and the rules. We have at our disposition a weighting of the facts and of the deduction itself.

Fuzzy inference

Manipulation of fuzzy rules and facts results in several practical problems [Giambiasi & al. 89]. The propagation of uncertainty coefficients, also called fuzzy inference, may be summarized in the following question:

How can the uncertainty coefficients of a rule and its premise be combined in order to determine the uncertainty coefficient of its conclusion?

In order to combine uncertainty coefficients, current expert systems refer to computational formulae derived from mathematical models of fuzzy reasoning. In these formulae, the basic idea consists in using the following conventional equations:

$$UV (P \& Q) = \min [UV (P) , UV (Q)]$$

$$UV (P \vee Q) = \max [UV (P) , UV (Q)]$$

$$UV (\neg P) = 1 - UV (P)$$

where P, Q indicate facts,

UV indicates the uncertainty measure (uncertainty value),

&, \vee , \neg indicate the logic connectors AND, OR, NOT.

The problem regarding the propagation of the uncertainty coefficients (fuzzy inference) goes back to determining a function g such as:

$$UV (Q) = g (UV (P), UV (P \rightarrow Q)) \quad \text{for each rule } P \rightarrow Q \text{ of the system,}$$

with:

$UV (P \rightarrow Q)$ = uncertainty coefficient of the rule $P \rightarrow Q$,

$UV (P)$ = uncertainty coefficient with which the condition P is established,

$UV (Q)$ = uncertainty coefficient of the conclusion Q, to be determined.

In present expert systems, the function g is computed using a mathematical model of the fuzzy inference (for example, Lee's fuzzy inference) or built by the knowledge engineer (for example, as

in the case of MYCIN). However, it is difficult to insure that the mechanism used reflects the expert's reasoning. In fact, we know of no mathematical models that perfectly express fuzzy reasoning by humans. Moreover, a general mathematical model is a priori independent of both the application and the domain of expertise. This is not the case in the real world. An analysis of the formulae used clearly shows that they are, in large part, arbitrary. This remark is still valid when dealing with formulae designed by the knowledge engineer (cf. for example, the formula used in MYCIN).

Fuzzy inference is a mapping function between the uncertainty coefficients (UCs) of the premises and that of the rule with the uncertainty coefficient of the conclusion. When the uncertainty coefficients are numerical values of the real interval $[0, 1]$, the fuzzy inference is a function $g : [0,1]^{n+1} \rightarrow [0,1]$ (n is the maximum number of premises).

4 Connectionist fuzzy inference

We propose using a neural network to approximate the fuzzy inference by learning from examples of fuzzy inference. If g is the function establishing the mapping between the uncertainty coefficients (UCs) of the premises and that of the rule with the UC of the conclusion, the learning examples given to the network will be vectors:

$$(v_1, v_2, \dots, v_n, v_{n+1}, v_0) \text{ with } v_0 = g(v_1, v_2, \dots, v_n, v_{n+1}).$$

The values (v_1, v_2, \dots, v_n) are those of the UCs of the premises. The value v_{n+1} is that of the UC of the rule and v_0 is the value of the UC of the conclusion. The learning law modifies the behaviour of the network f so it follows the behaviour described by the learning examples. We consider that the network has learned when the function f is equivalent to the function g .

Figure 2 shows schematically how to operate the learning on the network.

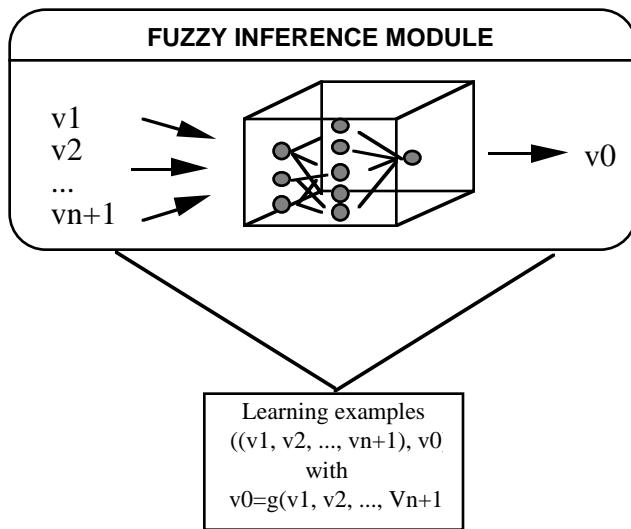


Figure 2. Learning of the fuzzy inference

5 Lee's fuzzy inference

Lee's fuzzy inference [Kaufmann 87] is an example of a mathematical model of fuzzy inference.

It is expressed by the following calculation:

$$\begin{array}{ll}
 [0, UV (P \rightarrow Q)] & \text{if } UV (P \rightarrow Q) = 1 - UV (P) \\
 UV (Q) = UV (P \rightarrow Q) & \text{if } UV (P \rightarrow Q) > 1 - UV (P) \\
 \emptyset & \text{if } UV (P \rightarrow Q) < 1 - UV (P)
 \end{array}$$

For practical reasons, we have reduced the size of the problem by limiting to one the number of premises for the rules ($n = 1$) and by allowing only ten discrete values for the uncertainty value ($p = 0.1$).

Table 1 shows the uncertainty coefficient of the conclusion UV (Q) computed from the uncertainty coefficients of the condition UV (P) and of the rule UV (P->Q) (to simplify the coding, we have replaced the interval [0, UV (P->Q)] by the single value UV (P->Q)).

UV(Q)		UV(P->Q)										
		0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
UV(P)	0											1
	.1										.9	1
	.2				0					.8	.9	1
	.3								.7	.8	.9	1
	.4							.6	.7	.8	.9	1
	.5						.5	.6	.7	.8	.9	1
	.6					.4	.5	.6	.7	.8	.9	1
	.7				.3	.4	.5	.6	.7	.8	.9	1
	.8			.2	.3	.4	.5	.6	.7	.8	.9	1
	.9		.1	.2	.3	.4	.5	.6	.7	.8	.9	1
1	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1	

Table 1. Table showing Lee's fuzzy inference

Heuristic building of the network

Using a neural network for a particular application is not as easy as one might think. Indeed, today there are neither rules nor formulae which allow the best possible selection of the numerous parameters of a network: input and output coding, network structure in terms of the number of units and number of layers, type of model, selection of the learning and test examples, etc. Everyone must make the most of the experience gained through developing previous applications, use his intuition and proceed cautiously through trial and error.

Coding of the uncertainty coefficients

The input layer codes the UCs of both the n premises and of the rule. The output pattern represents the UC of the conclusion. The UCs are located in the interval $[0,1]$. With the aim to simplify, all uncertainty coefficient values are chosen discrete. The UCs belong to the following set $\{0, 0.1, 0.2, \dots, 1\}$. The fuzzy inference then becomes a function $f: \{0, 0.1, 0.2, \dots, 1\}^{n+1} \rightarrow \{0, 0.1, 0.2, \dots, 1\}$.

There are numerous coding possibilities for the UCs on the input layer [Handcock 88]. The one we have selected is a thermometer coding for all values between $[0, 1]$ and a special code for \emptyset (distant in the sense of the Hamming distance). It should be noted that this coding minimizes the Hamming distance between two neighbouring values. For instance, codes for 0.2 and 0.3 differ by only one unit value (the third from the left). On the contrary, distant values such as 0.2 and 0.9 differ by a larger number of cells (7). This coding is shown in table 2.

UV	Codage
0	1 1 1 1 1 1 1 1 1 1 1 1 1
.1	1 1 1 1 1 1 1 1 1 1 1 1 -1
.2	1 1 1 1 1 1 1 1 1 1 1 -1 -1
.3	1 1 1 1 1 1 1 1 1 1 -1 -1 -1
.4	1 1 1 1 1 1 1 1 1 -1 -1 -1 -1
	...
.9	1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1
1	1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
\emptyset	-1 1 1 1 1 1 1 1 1 1 1 1 1

Table 2. Uncertainty coefficient coding

Structure

A priori, the number of hidden layers is not specified. Theoretical results impose one hidden layer, however it is sometimes easier to obtain a solution using two or more hidden layer networks. On the other hand, if the number of units in the input and output layers is defined by the coding, it is not the case for the hidden layers. Due to our previous experience in that domain, we have obtained a satisfactory solution after a certain number of attempts. Resulting from this more or less intuitive process, the selected model is a 66-neuron, 1080-synapses, three-layer network. There are 24 cells in the input layer, 12 in the output layer and 30 cells in the hidden layer.

Learning examples

The network is trained using a set of examples which are presented repeatedly. A learning example is a pair, the uncertainty coefficients of the premises and the rule and the associated UC of the conclusion. Learning examples are randomly selected from all possible values. There are 121 examples that we distribute arbitrarily between the learning set and the test set. It is extremely important that the learning example represent the problem and, for this, be selected at random with a fixed probability density.

Table 3 shows a learning set and its associated test set.

UV(Q)		UV(P->Q)											
		0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1	
UV(P)	0	■	□	■	□	□	□	□	□	■	□	□	1
	.1	□	■	□	■	□	■	■	□	■	□	.9	1
	.2	□	□	□	□	■	□	■	□	.8	.9	1	1
	.3	■	□	∅	■	□	■	□	.7	.8	.9	1	1
	.4	□	□	□	□	■	□	.6	.7	.8	.9	1	1
	.5	□	■	□	■	□	.5	.6	.7	.8	.9	1	1
	.6	□	□	■	□	.4	.5	.6	.7	.8	.9	1	1
	.7	■	■	□	.3	.4	.5	.6	.7	.8	.9	1	1
	.8	□	□	.2	.3	.4	.5	.6	.7	.8	.9	1	1
	.9	■	.1	.2	.3	.4	.5	.6	.7	.8	.9	1	1
1	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1	1	

■ : Learning examples (52), □ : Test examples (69)

Table 3. Learning and testing example bases

Learning

Before learning, network weights are set at random in the interval $[-.3, .3]$. The network response to an input vector (v_1, v_2) is of no importance. The gradient backpropagation algorithm modifies the connection weights iteratively so as to establish the mapping between the input vector and the desired output. For each presentation of an input vector, the network computes by propagation its output cell values. An error representing the difference between the obtained and the desired value is measured. This error is backpropagated and allows the algorithm to modify the weights in order to reduce the error.

Generalization : After the learning phase, we test the ability of the neural network to generalize, i. e. to give the correct answer for an unknown input vector by referring to the learning examples.

Results

The network has been implemented using the software SACREN [Touzet 89]. It is a general event-driven neural network simulator for SUN or Apollo workstations. This simulator is written in C and Pascal and allows the description of all neural models. Procedures have been written to facilitate the learning process and the statistical analysis of performance. The network response is thresholded: all positive or zero values of an output cell are considered equal to 1 and all negative values are -1.

A learning trial is the presentation of all the learning examples to the network and the modification of all the weights by the learning algorithm. After 25 learning trials, a minimum has been obtained for the error value: all learning examples produce the desired output.

After the learning phase, the 81 unknown examples of the test base are presented to the network so as to measure the generalization performance. Figure 3 shows this performance in relation to the number of learning examples.

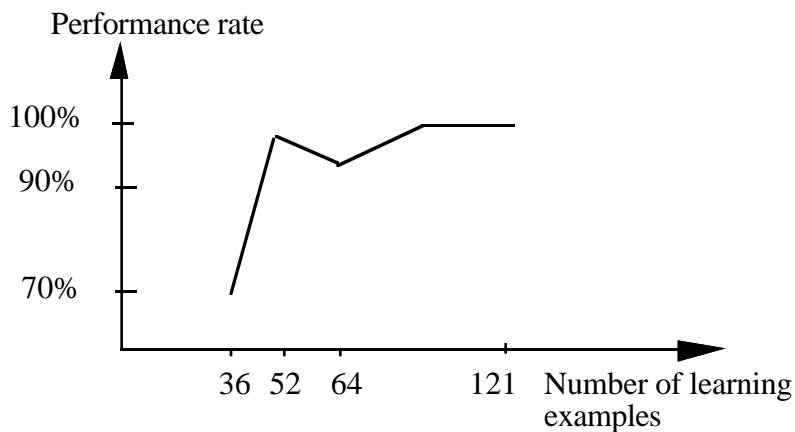


Figure 3. Impact of the number of learning examples on test performance

Generalization performance is deemed 72% effective when using 40 learning examples if the network response is considered true when equal to the correct value. If we accept an error value of 0.1 on the network response, then the generalization performance is 100%.

6 Fuzzy inference used in MYCIN

A second example which illustrates the capabilities of the connectionist approach concerns using examples to find the formula used in the well-known MYCIN system [Shortliffe 76] for the propagation of UCs. This fuzzy inference is described by:

Let $R = F_1 \& F_2 \dots \& F_n \rightarrow F$ be a coefficient rule $UV(R)$; the coefficient associated to F is then defined by:

$$UV(F) = UV(R) \times v$$

with :

$$v = \begin{cases} w + UV_p(F) - w \times UV_p(F) & \text{if } w \geq 0 \text{ and } UV_p(F) \geq 0 \\ w + UV_p(F) + w \times UV_p(F) & \text{if } w \leq 0 \text{ and } UV_p(F) \leq 0 \\ (w + UV_p(F)) / (1 - \text{MIN}(w, UV_p(F))) & \text{if } -1 < w \times UV_p(F) < 0 \\ 1 & \text{otherwise} \end{cases}$$

$$w = \begin{cases} \text{MAX}(UV(F_1), \dots, UV(F_n)) & \text{if } UV(F_i) \leq -0.2 \text{ " } i \\ \text{MIN}(UV(F_1), \dots, UV(F_n)) & \text{if } UV(F_i) \geq +0.2 \text{ " } i \\ 0 & \text{otherwise} \end{cases}$$

$$UV_p(F) = \begin{cases} \text{Previous coefficient of } F & \text{if } F \text{ is already deduced} \\ 0 & \text{otherwise} \end{cases}$$

It should be noted that in this formula:

- The computation of w attempts to establish the MIN of the UCs of the premises in absolute values (if the values are of the same sign and superior to a threshold of 0.2).
- The computation of v tries to reinforce the uncertainty coefficient of a fact if this fact is involved in several rules.
- The coefficient associated with the conclusion is something like:

$$UV(R) \times \text{MIN}(UV(F_i)).$$

With regards to the learning of this formula and for reasons of clarity, we will restrict the problem to the following case:

- Rules do not allow more than 2 premises: $n \leq 2$.
- Conclusions are inferred for the first time: $UV_p(F) = 0$.

With these restrictions in place, the formula becomes:

$$UV(F) = UV(R) \times v$$

$$v = w = \begin{cases} \text{MAX}(UV(F_1), UV(F_2)) & \text{if } UV(F_1) \leq -0.2 \text{ and } UV(F_2) \leq -0.2 \\ \text{MIN}(UV(F_1), UV(F_2)) & \text{if } UV(F_1) \geq +0.2 \text{ and } UV(F_2) \geq +0.2 \\ 0 & \text{otherwise} \end{cases}$$

Network structure

The selected network is a 69-neuron, 1100-synapses, three-layer network. There are 33 cells in the input layer, 11 in the output layer and 25 cells in the hidden layer.

Example base

The uncertainty coefficients belong to the set $\{-1, -.6, -.3, -.2, -.1, 0, .1, .2, .3, .6, 1\}$, so there are 1331 possible behavioural examples of the fuzzy inference rule. Learning examples are chosen at random.

Results

Figure 4 shows performance in relation to the number of learning examples (40 trials).

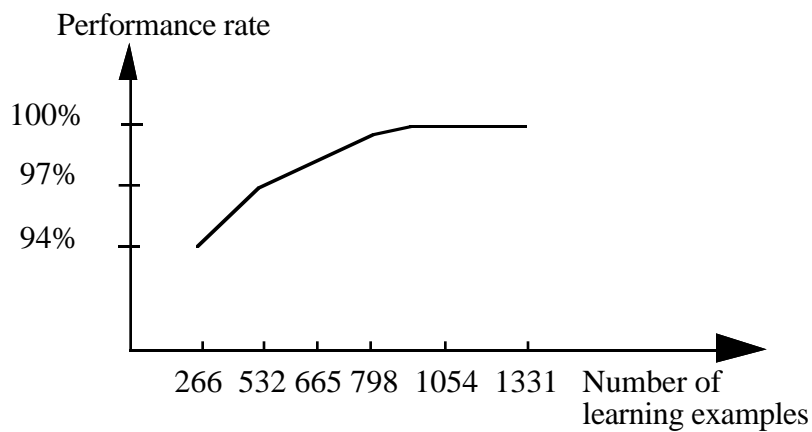


Figure 4. Impact of the number of learning examples on the performance of the MYCIN fuzzy inference

665 learning examples are sufficient to allow a 100% performance rate when a deviation of 0.1 from the exact value of the output is tolerated.

7 Discussion

Performances have been presented in terms of the number of learning examples, with the goal being to illustrate the generalization capabilities of a neural network. We are seeking the minimum

density of learning examples which allow a good performance in generalization. It is clear that this value is strongly connected to the nature of the function described by the examples (in our case, fuzzy inferences: Lee or MYCIN).

Only the case of rules with one or two premises has been processed here. Practical problems alone limit the application to multi-premis rules:

- 1) The number of input neurons increases in relation to the number of premises.
- 2) The size of the learning base grows in relation to the number of premises n and the selected precision p of a factor equal to n/p .
- 3) Increasing the precision is also expressed by an increase in the number of neurons, more or less linearly.

As a matter of fact, the practical implementation of our connectionist module of uncertainty propagation in a real system comes up against the problem which is posed by the variable number of premises. Even if we fix a priori an upper limit to the number of premises, how should rules having a variable number of premises be handled? An expensive solution, in terms of the neuron number, consists in building a network for each case.

8 Conclusion

We hope that we have provided some answers to the questions put forth in the introduction of this paper. Concerning the usefulness of neural networks, we have described an application which allows fuzzy expert systems to learn by experience and thus to be better adapted to the application domain. A method based on connectionist networks and gradient backpropagation has been proposed to set up the computational formula for the uncertainty coefficient of the conclusion from the uncertainty coefficients of the premise and the rule. Starting from examples of fuzzy inferences,

the network learns to behave as according to the inference rule, despite the fact that no rule has been explicitly programmed in the neural network. A three-layer network is able to learn how to compute the uncertainty coefficient of the conclusion precisely (with respect to level of discreteness adopted) for learned examples. In the case of unknown examples, this network is able to generalize correctly.

The proposed solution can be envisaged in two cases:

- If the knowledge engineer (+ possibly the expert) is able to find by trial and error the uncertainty propagation formula (as we suppose was the case for the MYCIN system), we can then foresee determining a formula by learning which is reasonably close in nature and free the knowledge engineer from this problem.
- If the knowledge engineer (+ possibly the expert) is not able to find such a formula, we can try to find it by learning from examples formulated by the expert.

With the same aim of evaluating the usefulness of neural networks, we can cite the coupling between a connectionist model and a classical theorem demonstrator that makes it all the more pertinent [Suttner & al. 91], in this work.

9 References

[Amy & al. 90]

B. Amy, A. Giacometti & A. Gut, "Modèles connexionnistes de l'expertise," Neuro-Nimes 90, Nimes, France, EC2 Eds., pp 99-119, November 1990.

[DARPA 88]

"DARPA Neural network Study," AFCEA International Press, 1988.

[Dayhoff 90]

J. Dayhoff, "Neural Networks Architectures," Van Nostrand Reinhold, 1990.

[Giambiasi & al. 89]

N. Giambiasi, R. Lbath & C. Touzet, "Une approche connexionniste pour calculer l'implication floue dans les systèmes à base de règles", Neuro-Nîmes 89, Nîmes, France, EC2 Eds, November 1989.

[Handcock 88]

P. Handcock, "Data representation in neural nets: an empirical study," Proc. of the 1988 Connectionist Models Summer School, D. Touretzky & al. Eds, Morgan Kaufmann Publishers, 1988.

[Hebb 49]

D. O. Hebb, The Organization of Behaviour," New-York : Wiley, 1949.

[Hecht-Nielsen 90]

R. Hecht-Nielsen, "Neurocomputing," Addison-Wesley, pp. 132, 1989.

[Karna & al. 89]

K. Karna and D. Breen, "An artificial neural networks tutorial: part 1-Basics," The International Journal of Neural Networks Research and Applications, Vol. 1, No. 1, pp 4-23, January 1989.

[Kaufmann 87]

A. Kaufmann, "Nouvelles logiques pour l'Intelligence Artificielle", Ed. Hermès, 1987.

[Kohonen 88]

T. Kohonen, "An introduction to neural computing", Neural Networks, Vol. 1, No. 1, pp 3-16, 1988.

[Le Cun 87]

Y. Le Cun, "Modèles connexionnistes de l'apprentissage," Thèse de Doctorat, Université Paris 6, July 1987.

McCulloch & Pitts 43]

W. S. McCulloch & W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics 5, pp 115-133, 1943.

[Minsky & al. 88]

M. L. Minsky & S.A. Papert, "Perceptrons," Expanded Edition, MIT Press, 1988.

[Suttner & al. 91]

C. Suttner & W. Ertel, "Using backpropagation for guiding the search of a theorem prover," The International Journal of Neural Networks Research and Applications, Vol. 2, No. 1, pp 3-16, March 1991.

[Shortliffe 76]

E. Shortliffe, "Computer-Based Medical Consultations : MYCIN", American Elsevier, N.Y., 1976.

[Touzet 89]

C. Touzet, SACREN : Système d'Aide au Choix d'un Réseau de Neurones, Rapport final du contrat ANVAR n° A8801006, juillet 1989.