5. Hybrid Machine Learning Tools: INSS - A Neuro-Symbolic System for Constructive Machine Learning

Fernando Osório[1], Bernard Amy[2], and Adelmo Cechin[1]

[1] UNISINOS - Computer Science Dept.
Av. Unisinos, 950 - CP 275 - CEP 93022-000
São Leopoldo - RS - BRAZIL
Web : http://www.inf.unisinos.tche.br/
E-mail: {osorio,cechin}@exatas.unisinos.tche.br

[2] Laboratoire LEIBNIZ - IMAG - INPG
46, avenue Felix Viallet 38031
Grenoble Cedex 1 - FRANCE
Web : http://www-leibniz.imag.fr/RESEAUX/
E-mail : amy@imag.fr

1. Introduction

Various Artificial Intelligence methods have been developed to reproduce intelligent human behavior. These methods allow to reproduce some human reasoning process using the available knowledge. Each method has its advantages, but also some drawbacks. Hybrid systems combine different approaches in order to take advantage of their respective strengths. These hybrid intelligent systems also present the ability to acquire new knowledge form different sources and so to improve their application performance.

The main argument, and the most used one, to justify the study and the application of hybrid symboli-connectionist systems is the complementarity of symbolic AI methods and sub-symbolic connectionist methods (Artificial Neural Networks - ANN).

Such a justification is a very general one. And it remains to be more precise about the real contribution of the hybrid approach. What exactly provides the combination of neural networks and knowledge based systems? Researchers claim that hybrid systems take advantage of their respective component strengths. Is it a real property of the existing hybrid neuro-symbolic systems? And what are these advantages?

To validate an hybrid system, one have to answer these questions, and to describe what really can be done with this system which was hardly done with just one of its components. Particularly the system has to be given proof of the following properties:

- Possibility to use and to take into account several kinds of knowledge representation, like empirical data and expert knowledge (examples, production rules and fuzzy rules).

- Best efficiency of the global system when compared to each of its components.
- Strong coupling between the components, leading to an exchange of knowledge between all of these components. This knowledge has to be proved consistent and useful. The best way for implementing such a coupling is to choose the integration mode called co-processing [17, 23], in which the different components of the hybrid system work on the same level and exchange information between both themselves and their environment.
- Possibility of global learning. The whole system is able to adapt its various sets of knowledge to the variations of the data domain. This tuning can be done following two ways : either learning (or forgetting) new examples, or modifying the architecture of the neural component.

In this paper we describe a system, called INSS (Incremental Neuro-Symbolic System [24, 25]), endowed with these properties. In section 2 we explain the origin of our system and the reasons of our choices. Section 3 describes INNS system. Then, in section 4, to validate the system, we present some practical results allowing to show that INSS has the sought properties. Section 5 presents the application of INSS in a medical domain.

2. The co-processing integration mode

In the classification task domain, the hybrid neuro-symbolic systems, such as SYNHESYS [15] and KBANN [35], exploit their capacity to use at the same time theoretical knowledge (set of symbolic rules) and empirical knowledge (set of observed examples). These two systems are significant examples of the coprocessing integration mode in hybrid systems, allowing a bi-directional knowledge transfer between the symbolic and connectionist modules. Figure 2.1 shows the general architecture of this kind of systems.

We chose to base our study on the KBANN model, a well-known hybrid neuro-symbolic system that represents, among others, the state-of-the-art in this domain. This system is able to compile a knowledge base into the form of an ANN. Then, it learns from an example data set, and after that it extracts new rules. This approach allows a refinement of initial knowledge, as we can see in Figure 2.2. Such a system constructs robust networks: the insertion of a priori theoretical knowledge leads to quicker learning; we can use small data sets during the learning phase; all available knowledge about the problem (whether theoretical or empirical) is used; and thus the system is more adapted to process incomplete and/or erroneous data.

However, the KBANN system has some important limitations due to the choice of its ANN model and learning method, the Back-Propagation algorithm [28]:

1. KBANN networks are based on static networks, so it is difficult to change or to add new knowledge. A simple change in old knowledge requires a

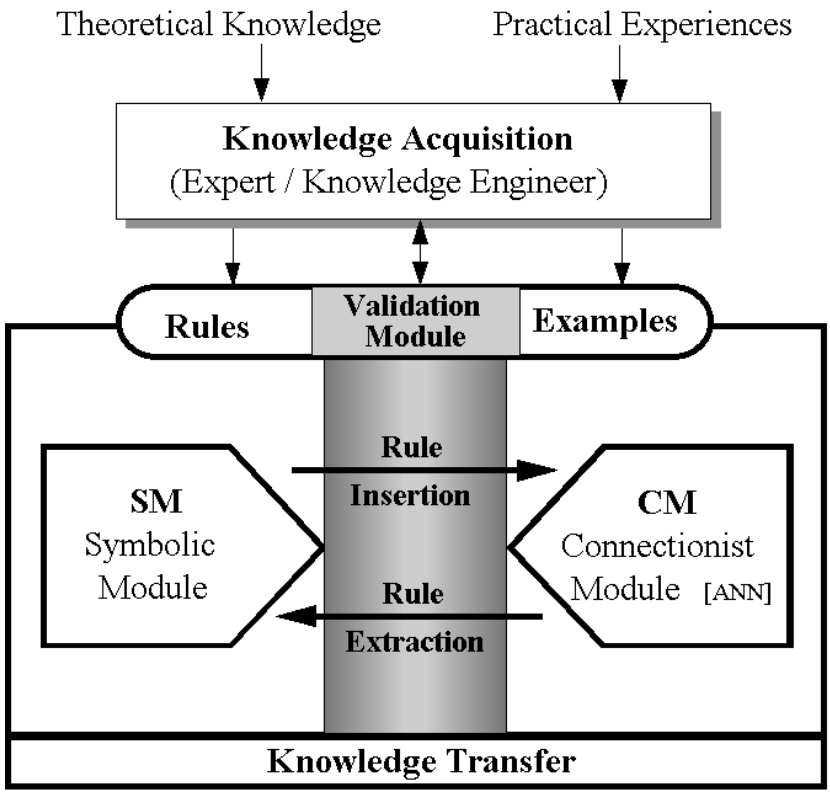


Fig. 2.1. Hybrid neuro-symbolic systems and knowledge transfer

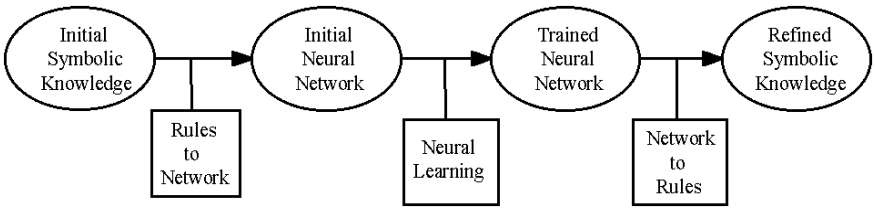


Fig. 2.2. Knowledge refinement using KBANN

retraining of the whole network. Furthermore, if new knowledge is dicovered that turns invalid the old knowledge, it is difficult to correct it;
2. due to the use of the Back-Propagation algorithm, the learning is slow. Back-Propagation as a first order (first derivative) successive approximation algorithm brings with him performance problems (for example, the "flat spot" problem) that can be solved by better training algorithms and strategies like QuickProp, Cascor, RProp, Scaled Conjugate Gradient ALgorithm, etc. [29];
3. in the KBANN since the rule extraction process must consider the whole network after the rule insertion and training, all the rules must be extracted. That is, a fully new knowledge base is created, which may include the old rules or not. The interpretation of this new knowledge base by a human becomes then a time consuming task, since he has to throw away all the previous analysis.

It is the reason why we developed the new system called INSS to improve KBANN networks and to overcome its main limitations. This new system also authorises insertion, refinement and rule extraction, but, unlike the KBANN system, each process performs incrementally. Moreover, instead of using the Back-Propagation algorithm, based on static networks, INSS uses the Cascade-Correlation learning method [14] which proceeds by adding new units (neurons) during learning. Our approach allows to obtain a constructive network that is able to develop its structure and its knowledge, while keeping unchanged the principal properties of a hybrid neuro-symbolic system. The main feature, that constitutes the originality of our system, is that we are able to perform an incremental rule extraction [11]. The rule extraction process may analyze only the new added units and occurs as new knowledge is acquired. This way, the old knowledge remains intact and is progressively incremented by the new one. Furthermore, the user is able to determine the degree of exactness or of generality of the extracted rules. Less rules give him a overview of the data being modelled but with little exactness. As more rules are extracted from a more refined network, more details can be added to the old ones. We do not know any other neuro-symbolic system able to extract rules in a such incremental way.

3. The INSS system

The *INSS system* is composed of five modules: *Symbolic-Module* (Symbolic Inference Engine), *NeuComp* (Construction of a network from rules), *NeuSim* (ANN learning and recall), *Extract* (Rule extraction), and *Valid* (Validation of acquired knowledge, by means of study of relations between rules and examples). The INSS system components are represented in Figure 2.3.

Our system uses the CLIPS language (*C Language Integrated Production System*) [16], developed by the STB-NASA, as its symbolic module. Our

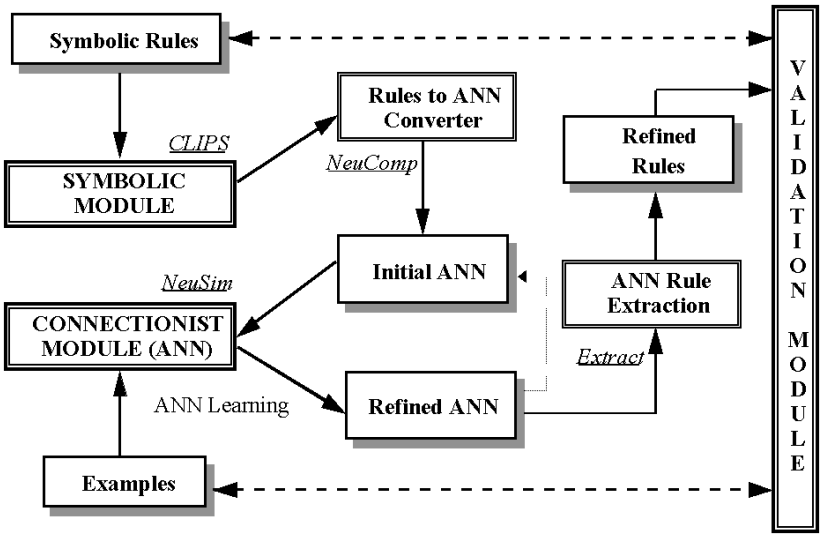


Fig. 2.3. INSS System: Constructive knowledge refinement

system also provides facilities to transfer rules and examples to/from the specific syntax used in this language and the syntax used in our tools (NeuComp/NeuSim/Extract). The NeuSim module can be also used as a forward-chaining inference engine once the symbolic rules have been transferred to the connectionist module.

3.1 Rule Insertion

The NeuComp module can process elementary production rules (simple propositions) which we called "rules of order 0". These rules are equivalent to IF/THEN forms such as:

```
IF <Condition>(TRUE/FALSE) AND/OR
   <Condition> (TRUE/FALSE)...
THEN <Conclusion>
```

The rule compilation follows the method described by Towell [35, 37]. The result of the translation is a network composed of a set of units linked by weighted connections (see Figure 3.1). The activation of this network, before learning, leads exactly to the same results (outputs) as those obtained with the set of rules.

We also extended the rules used by KBANN to "high level rules" [26, 27]: production rules of order 0+, which are rules including value intervals. As application problems, where this type of rules were used, we cite:

– Robotic applications. Ex.: an autonomous robot using left and right sensors can be controlled by rules of the form: "if the left sensor signal is higher

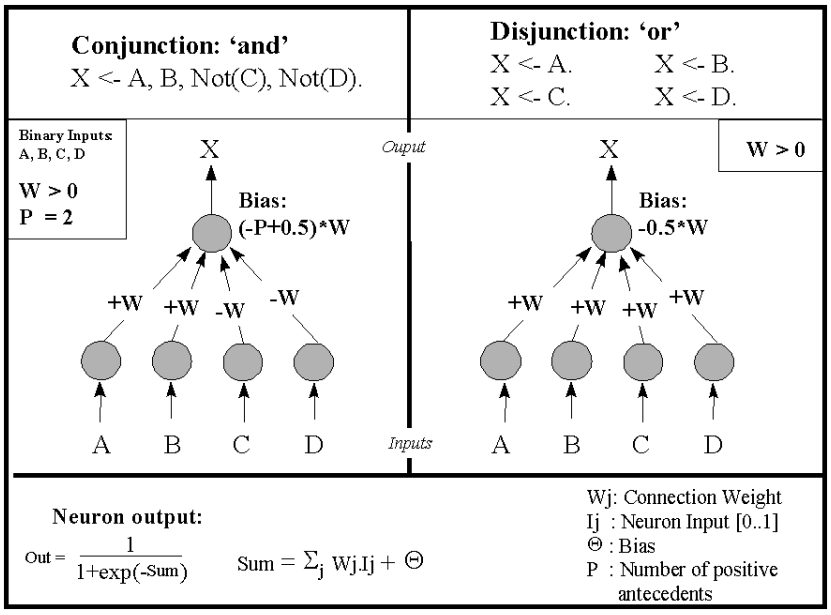


Fig. 3.1. Rules to network translation ("rule insertion")

than the right one, then it is closer to the left wall". Using only boolean operators or boolean logic rules does not allow such a conclusion.
– Medicine. Ex.: an alarm equipment in a hospital can indicate the state of a patient through rules of the form: "if the temperature is higher than 37 degrees or lower than 35 degrees, then please call the nurse".

The introduction of this kind of rules was necessary to express and to introduce the existent knowledge (expert knowledge) to NeuComp in projects conducted by the authors of the paper.

We implemented the usage of comparison functions of the following type:

```
<Operator>(<Feature>, <Value>)    or
<Operator>(<Feature>, <Feature>),
where <Operator> is GreaterThan, LessThan or Equal.
```

Resulting in rules of this kind:

```
IF GreaterThan(Sensor_S1, 1.0) AND
   LessThan(Sensor_S1, Sensor_S2)
THEN Conclusion_C1
```

These rules can be compiled into an ANN composed by simple Perceptron like units (we create feed-forward multi-layer networks with sigmoïd based units). A detailed description of all compilation processes, used within INSS,

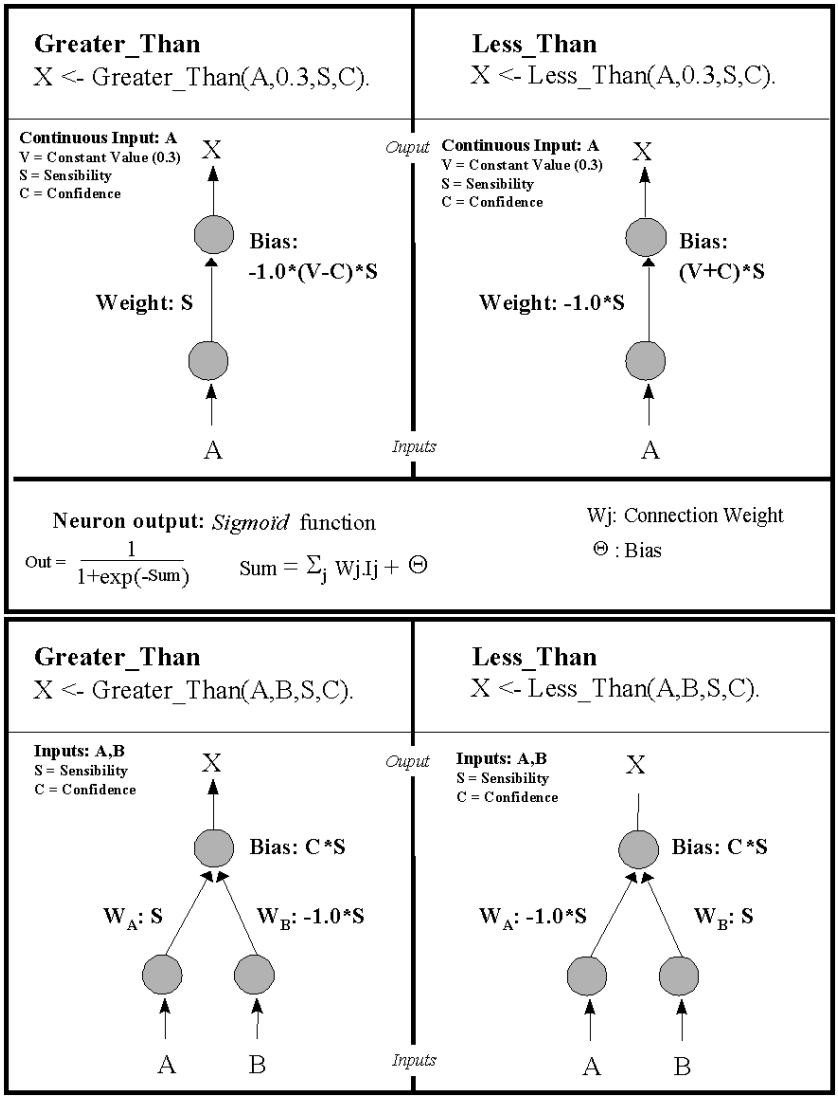


Fig. 3.2. Compiling rules of 0+ order

can be found in [25], and we show a brief description of the compilation process in Figure 3.2. We added two new parameters to our symbolic rules that allow us to specify the "sensibility" (slope of the output curve) and the " confidence" (displacement of the output curve related to the specified activation threshold). So, with these two parameters we are able to compile rules like: less than or equal to X, greater than or equal to X, less than and not equal to X, in range (including or excluding limit values), etc. The "sensibility" parameter acts like the "W" value used with KBANN nets [35].

We show in Figure 3.3 examples of the neurons output. These neurons were obtained using our compilation process of 0+ order rules. We can observe rules that compare one input value with one constant value (Greater_than [Feature, ConstValue]), and rules that compare one input value with another input value (Greater_than [Feature, Feature]).

As the symbolic rules allow to establish some initial knowledge and then give an initial structure to the network, this approach solves two important problems related to Artificial Neural Networks: on one hand this simplifies the choice of the number and distribution of units, on the other hand we obtain a good assignment of initial values to the connection weights.

3.2 Learning

The use of the Cascade-Correlation learning algorithm instead of Back-Propagation, in the NeuSim module, allows a quicker learning [14, 29], with higher performance results [29, 34]. Figure 3.4 shows an example of the network structure evolution when we apply the Cascade-Correlation learning algorithm. It allows especially constructive learning where the initial knowledge is not mixed with the new acquired knowledge. New units are added to the initial network structure in order to correct or complete the initial knowledge.

The importance of such a choice of the learning method is reinforced by studies [30, 31] showing that Cascade-Correlation networks can be used to model some aspects of human cognitive development.

The Cascade-Correlation algorithm developed by Fahlman and Lebiere [14], in contrast to static neural learning algorithms such as Back-Propagation [28], is a generative technique to network construction and learning. Instead of merely adjusting weights in a network of fixed topology, Cascade-Correlation starts with a minimal network of input and output units. During learning, it may add hidden units one at a time, installing each on a separate layer. This is done in the following way: if the net is not reducing error fast enough with its current topology, it will select and install a new hidden unit whose output activations correlate best over all training cases with the existing network error. Once one new unit is installed in the network its weights are frozen, and this unit keeps unchanged its learned weights. So, Cascade-Correlation will reduce step-by-step the network output error by a

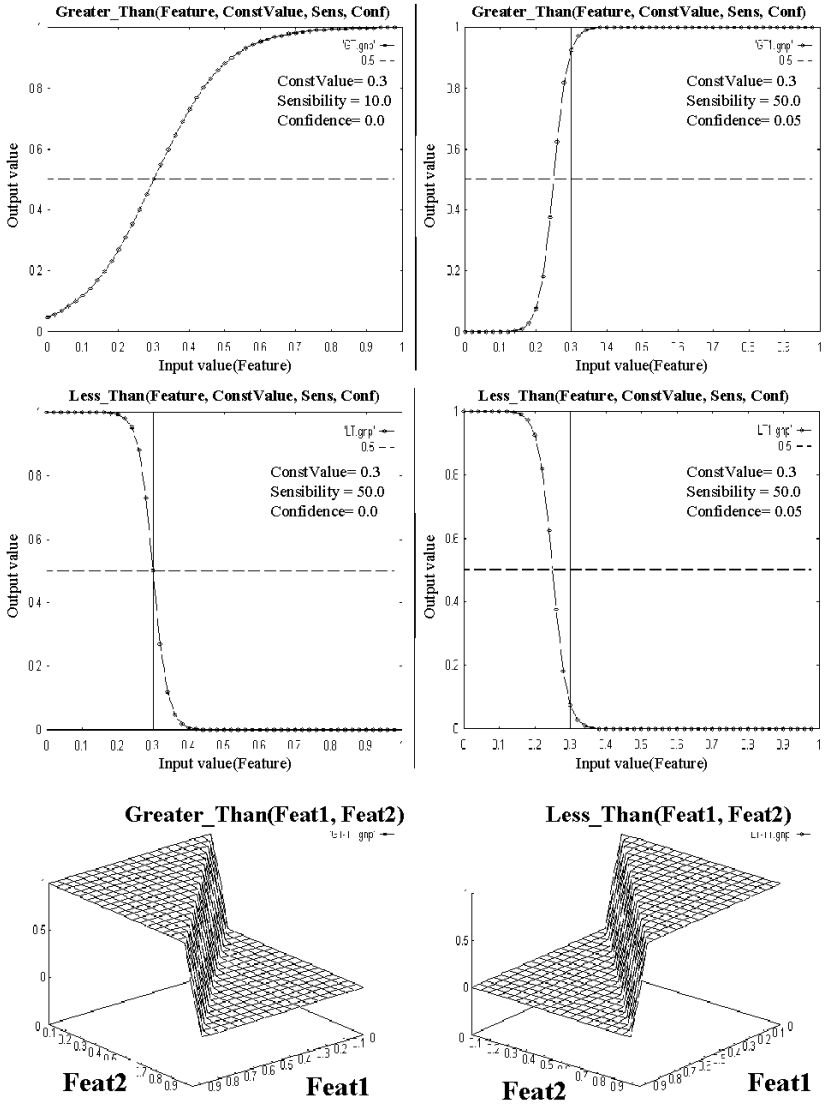


Fig. 3.3. Compiled neurons output - GreaterThan, LessThan

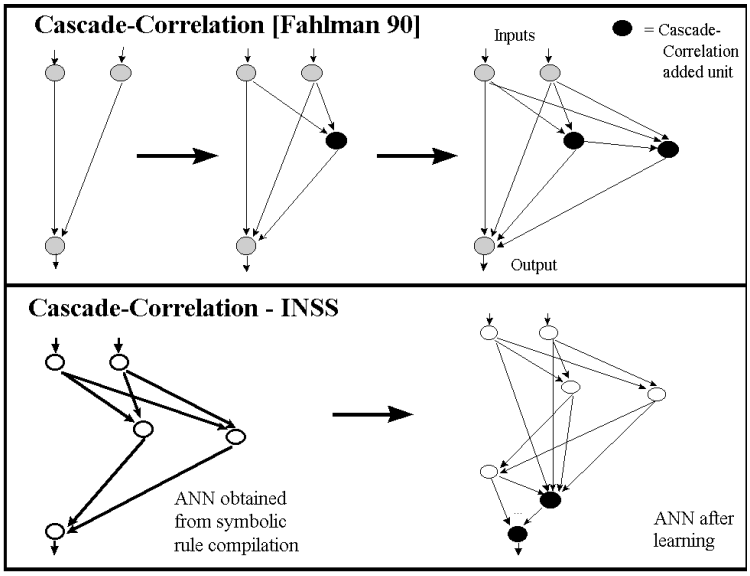


Fig. 3.4. ANN structure evolution using Cascade-Correlation

cyclic process of output units learning and hidden unit addition/learning. In essence, Cascade-Correlation searches not only in weight space but also in the space of network topologies.

Learning in a network by adding new units allows to complete, to change, or to refine the initial knowledge. In INSS, using the Extract module, one can be able to analyse only the new added units and the modified output units. The old units always keep their function and their meaning in comparison with the initial rules introduced into the network. As we can preserve unchanged the initial knowledge acquired, this technique makes the main difference of our system in comparison with the KBANN system [11, 12, 25].

3.3 Rule Extraction

The Extract module [11, 12] implements an improved version of the SUBSET algorithm [2, 35, 36] of rule extraction from neural networks. This algorithm was improved in two ways. First, the extraction process is a lot simpler and quicker since we look only at a small part of the network. We do not need to extract all network knowledge, but just the new acquired knowledge. Second, we developed heuristic methods for network simplification (remove less significant units and links), used before extraction. The use of a simplified network helps us to reduce the complexity of the extraction procedure.

We included in our system the use of expert and fuzzy rules, since a) the inputs of a neural network can be interpreted as the state of the system; b) the mapping performed by the network of the system state to an output can

be interpreted as the inference mechanism performed by production systems; c) the output of the network can be interpreted as the action to be taken on the system.

The use of fuzzy rules in this context is mainly due to:

- for some kinds of neural networks (RBF networks), there is a proof of equivalence between the neural network and fuzzy inference systems ([19];
- the smooth continuous mapping implemented by some feedforward neural networks (like MLP) can be easily represented by fuzzy rules;
- *counterpropagation networks* use proximity concepts, which are very similar to membership functions of the fuzzy techniques.

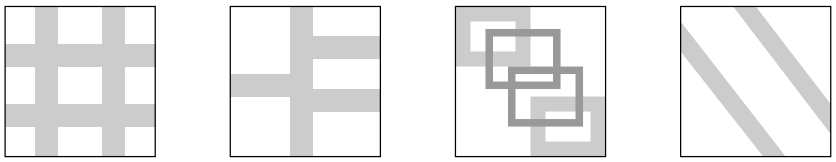
The main disadvantage in the use of fuzzy rules is that one rule interferes with the others making the rules less modular. This disadvantage is partly suppressed by the use of mutually exclusive rules.

Unfortunately most of the extraction algorithms work only with a dedicated structure, where the network has special neurons with special activation functions connected in a special way. The result is a mirroring of the fuzzy algorithm in form of a network. There are fuzzyfication neurons, inference neurons and defuzzyfication neurons. A simple look at the network reveals the fuzzyfication, inference and defuzzyfication methods of the corresponding fuzzy inference system. So we should limit ourselves to work only with MLP networks.

Two problems arise when studying the diverse architectures and working with transparent networks (networks offering the possibility to extract knowledge in the form of rules):

1. How to obtain an initial input space partition to be expressed in form of fuzzy sets (see figure 3.5): there are 4 methods to partition the input space: grid partition, tree partition, scatter partition [18] and linear partition. These fuzzy sets are used then to form the rules. The grid partition is commonly used: divide each input variable range in 3 fuzzy sets with linguistic values *low*, *medium* and *high* and then combine them with the intersection (see figure 3.5(a)). The disadvantage of the grid partition is that the number of fuzzy sets increases exponentially with the number of input variables. The number of fuzzy sets is important because this number corresponds to the number of rules, which is the most important index about the comprehensibility of a fuzzy inference system. The tree and scatter partitions increase the comprehensibility of rules, but have the disadvantage that, depending on the application and on the distribution of the data, many of them may be required to cover the training set. For example, when the data is distributed diagonally to the input variables, or when there are dependencies among the input variables. The scatter partition has the disadvantage that the partitions are not mutually exclusive. The linear partition has the advantage that

it produces the smallest number of rules but the membership functions are more difficult to comprehend.
2. How to tune or to train the fuzzy inference system: the rules alone are not enough to implement a good mapping from the inputs to the outputs. A training with data is necessary to adapt the parameters of the rules.



(a) (b) (c) (d)

Fig. 3.5. Various methods for partitioning the input space: grid partition (a); tree partition (b); scatter partition (c); linear partition (d).

These two problems are very similar to the problems of determining the number of hidden units in a sigmoid neural network and determining the structure of the network.

Therefore, the most common strategies to solve the problems above were identified and listed below.

1. The use of special structures representing the structure of the fuzzy inference system: this facilitates the introduction of fuzzy rules in the network and the solution of the problem of the input partition. At the same time, this facilitates the extraction of the rules.
2. The use of special units representing the elementary operations of a fuzzy inference systems: min, max, multiplication, division.
3. The use of gradient descent algorithms to solve the problem of parameter tuning: most algorithms are a modification of Backpropagation adapted to the new units and architectures. The problem with such a generalization in the use of gradient descent algorithms is that some care must be taken with units whose nonlinear function has a zero derivative somewhere or units, whose effect on the learning is to adapt the wrong parameters of the network.
4. The use of a partition aligned to the input variable axis and mutually exclusive (see figure 3.5(a) and (b)).
5. The use of a partition defined by the user: this is considered by many authors as an advantage, since knowledge about the problem can be introduced into the network.
6. The computation of all combinations of the linguistic values of the input variables in the premises: this simplifies the algorithm but increases the number of rules generated.

7. The use of reinforcement learning or stochastic search to solve the control problem. Both methods require the possibility to experiment with the process or with some model of the process. The stochastic search requires also an objective function to be minimized.

Furthermore the different rule extraction systems generate two types of fuzzy rules: Mamdani or Sugeno fuzzy rules.

A **Mamdani Fuzzy Inference System** with two inputs x_1 and x_2 and one output y contains a set of Mamdani Rules with the form [20]:

$$\texttt{IF } x_1 \in F_1 \texttt{ AND } x_2 \in F_2 \texttt{ THEN } y = F \tag{3.1}$$

where F_1, F_2 and F are fuzzy sets. Their membership functions μ_{F_1}, μ_{F_2} and μ_F have normally the form of triangles ($\mu_{F_1}(x_1) = 1 - \min(k_1|x_1 - k_2|,1)$), where k_1 defines the steepness and k_2 is the center of the triangle. The membership functions of a fuzzy inference system constructed with Mamdani rules should cover the input space, at least the points which can occur in practice. The fuzzy logic operation AND is implemented with the min operator, the rule of inference is computed using the min operator and the composition is computed with the max operator.

The final result is a fuzzy set. A real value must still be computed from the membership function using some defuzzyfication method.

A **Sugeno Fuzzy Inference System** with input variables $x_1, \cdots, x_n$ and output variable y contains a system of Sugeno rules with the form [33]:

$$\texttt{IF } f(x_1 \; is \; F_1, \cdots, x_i \; is \; F_i, \cdots, x_n \; is \; F_n) \texttt{ THEN } y = g(x_1, ..., x_n) \tag{3.2}$$

where $F_1, \cdots, F_n$ are fuzzy sets representing the region of the input space where the rule is valid, f is the fuzzy logic operation (AND, OR) that connects the propositions ($x_i \; is \; F_i$) in the premise and g is the function which computes the value of y when $x_1, \cdots, x_n$ satisfies the premise.

The two main advantages of the Sugeno rules in relation to the Mamdani ones are: first to solve the problem of the need of too many implications to cover the whole input space and second to simplify the computation effort of the defuzzyfication process. If the input has many components, then the space is too large to be covered by the rules. The functional relations in the consequence part between inputs and outputs make an additional dependence between the two spaces. This turns the premises valid in a larger set and then the number of implications needed is reduced. Second, the defuzzyfication includes normally the computation of some integrals. This is a laborious computation which takes a lot of time and can be a problem in real-time applications.

Although the function g was generically defined, Takagi and Sugeno report only the use of a linear function $g(x_1, ..., x_n) = k_0 + k_1 x_1 + \cdots + k_i x_i + \cdots + k_n x_n$ where k_i are constants.

Based on the last analysis of different fuzzy inference systems, we present now the idea for the extraction of fuzzy rules from a neural network of the type MLP. Consider the network with the structure as in figure 3.6 (left). This figure shows that the threshold unit separates the input space into two mutually exclusive regions. Each region is described by two pieces of information: where it is located (its region) and the corresponding mapping equation performed by the network. Its region can be expressed in form of an inequality: $a_3 > 0$ for the grey region and $a_3 \leq 0$ for the white region, where a_3 is the activation of the unit 3.

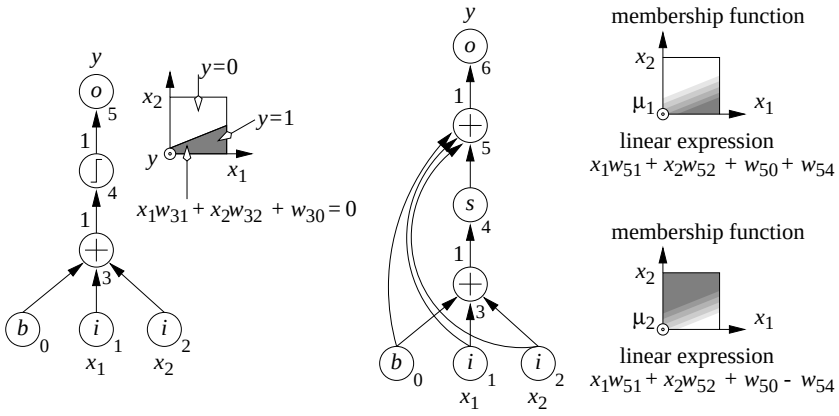


Fig. 3.6. Neural network with one *threshold* unit (marked with an s) (left). Neural network with one *sigmoid* unit (right). w_{ij} is the weight of the connection from the j-th to the i-th unit. The symbol $\odot$ is to be understood as an arrow pointing to the reader.

Consider now the network in figure 3.6 (right) with shortcut connections, *sigmoid* units instead of *threshold* units and *add* unit at the network output. Since the sigmoid function is continuous there is no sharp limit separating the white from the grey region as in figure 3.6 (left). In this case, the relations $a_3 > 0$ and $a_3 \leq 0$ must be generalized to a fuzzy relation with defining membership functions μ_1 and μ_2.

Now, associated with each region there is a membership function (for example, $\mu_1(x_1, x_2)$) and a linear equation (for example, $y = x_1 w_{51} + x_2 w_{52} + (w_{50} + w_{54})$) expressing a dependance of the network output on the network inputs in this region. The computation of the output of the network can be performed multiplying (implementation of a fuzzy inference) the membership function by the corresponding linear expression and afterwards added up (implementation of a fuzzy composition):

$$y = \mu_1(x_1, x_2)(x_1 w_{51} + x_2 w_{52} + w_{50} + w_{54}) + \mu_2(x_1, x_2)(x_1 w_{51} + x_2 w_{52} + w_{50} - w_{54})$$

or expressed in form of Sugeno fuzzy rules:

$$\texttt{IF } (x_1, x_2) \textit{ is } G_1 \texttt{ THEN } y = x_1 w_{51} + x_2 w_{52} + w_{50} + w_{54}$$
$$\texttt{IF } (x_1, x_2) \textit{ is } G_2 \texttt{ THEN } y = x_1 w_{51} + x_2 w_{52} + w_{50} - w_{54}$$

where G_1 and G_2 are the fuzzy sets with membership functions μ_1 and μ_2, respectively.

This idea can be expanded for MLPs with short-cut connections and any number of hidden layers. This was developed in the FAGNIS (Fuzzy Atomatically Generated Neural Inferred System) system ([5] [6]). In [5] a proof of the equivalence between the neural network and the generated fuzzy inference system was performed, in dependence only of the choice for the membership functions. In that work an exact definition of the extraction problem is shown with theorems, which define the limits and conditions for the rule extraction.

To maintain an easy interpretability of the membership functions associated with the fuzzy rules generated some constraints were imposed on them:

1. The fuzzy set $F_i(x)$ is a convex fuzzy set, whose membership function is the highest at only one point with value 1. This way, it is guaranteed that the degree of membership decreases as the input values get farther from this point.
2. **Mutual exclusiveness of the fuzzy sets.** Mathematically, this constraint is expressed by $\sum_{i=1}^{n} F_i(x) \leq 1$. The system of fuzzy rules shall not have more than one rule completely active at a certain state.
3. **Region of influence constraint.** The region of influence (where the membership function has a value different of 0) should be occupied by as few membership functions as possible. For the one-dimensional case this number should not be greater than two.

The FAGNIS system of rule extraction was already tested on some applications. First in the control area [8]. These ideas were applied to the control of the angle and position of the inverted pendulum with a specialized network. There, FAGNIS was compared with a Deadbeat controller (linear state controller). These ideas were adapted later to treat a second problem, the control of an interferometer in the MIPAS (Michelson Interferometer for Passive Atmospheric Sounding [22]) experiment at the Nuclear Research Center Karlsruhe, which was investigated and published in [7] [13].

A second application of FAGNIS was in the chemistry area where it was used to interpret neural networks trained to predict separation factors in the gas chromatography ([6][9][10]).

3.4 System Improvements

Since we can not be totally sure if all symbolic knowledge (rules) are really perfect with no inconsistencies, and also we can not be totally sure if all examples in learning database are really perfect, so we need to check the

validity of our old rules against the new symbolic rules (obtained by rule extraction) and the available learning examples. The Valid module [3] finds out the probably incorrect rules and examples. Thus, we will need to submit these inconsistencies to an expert analysis, or to increase our learning and rule databases with more informations. This module is very important and complex and it is under development.

In summary, the INSS system presents some important advantages over its predecessor, KBANN. Our system improvements allow us to eliminates some drawbacks of KBANN nets:

- The INSS constructive neural architecture allows to work with incomplete symbolic rule sets and also with incorrect symbolic rule sets. Our system can easily add new rules (neurons) or even make broader changes in the existing ones. The KBANN networks, as they use static networks, restrict learning to less important changes to the rule set. If we need to 'learn' a new rule from examples in KBANN we should add manually specific units for this purpose.
- The KBANN network algorithm tries not to change unit meaning, and tries to keep the symbolic label significance associated to them. We can not be sure that, during the KBANN learning process, its units will not suffer a meaning shift. The Cascade-Correlation, used within INSS, keeps unchanged the initial acquired knowledge (compiled rules) by freezing the network connection weights, and does not have any problem of meaning shift.
- The learning algorithm used in INSS is faster than KBANN's Back-Propagation based algorithm. Besides, this algorithm allows an incremental network construction, by improving the connection weights as well as the network topology.
- Our rule extraction algorithm does not need to analyse all the ANN structure, but instead we just consider the new acquired network knowledge by analysing the new added units. This leads to an important reduction of the rule extraction process complexity.
- We are not restricted to using binary inputs (rules of order 0), nor obligated to pre-process continuous inputs in order to discretizate them. Our system allows symbolic rule compilation of proposition rules of order 0+.
- The integration of efficient fuzzy rule extraction algorithms into the INSS system due to its modular construction expands its capabilities. This represents an improvement in the knowledge representation power of the generated rules and so in the range of applications of the system.

4. Validation of INSS : practical results

The possibility to use and to take into account several kinds of knowledge representation appears clearly in the description of the functionning of INSS.

It is the same for the possibility of global learning. INSS can not only adapt its various sets of knowledge to the variation of the data domain, but also to learn by modifications of its architecture.

It remains to show that the hybridisation increases the efficiency of the system, and that the knowledge extracted by INSS is a "good" knowledge. With this aim in view we have applied INSS on a relatively simple application, the *Monk's Problem* [34]. This problem is a set of tests developed for performance comparison of different learning algorithms. There are three Monk's problem data sets. Here we will discuss only the results we obtained within the first one, the Monk1 problem, although our tests cover all three problem data sets.

Table 4.1. Monk1: Description of the symbolic rule set

Input Features:
HEAD_SHAPE = { ROUND, SQUARE, OCTAGON }
BODY_SHAPE = { ROUND, SQUARE, OCTAGON }
IS_SMILING = { YES, NO }
HOLDING = { SWORD, BALLOON, FLAG }
JACKET_COLOUR = { RED, YELLOW, GREEN, BLUE }
HAS_TIE = { YES, NO }
Symbolic Rules:
(1) Monk1 ← HEAD_SHAPE = ROUND, BODY_SHAPE = ROUND
(2) Monk1 ← HEAD_SHAPE = SQUARE, BODY_SHAPE = SQUARE
(3) Monk1 ← HEAD_SHAPE = OCTAGON, BODY_SHAPE = OCTAGON
(4) Monk1 ← JACKET_COLOUR = RED

The Monk1 problem data set is composed by one set of four symbolic rules (see table 1 for the complete domain theory), by one generalization test set of 432 examples (covering all the input space), and by one learning set of 124 examples. The examples are exactly those available in the original data [38]. In our experiments we used portions of the rule set and the examples set in order to study the generalization capacity of our system. Just the learning set and the rule set were partitioned, for its part, the generalization test set was preserved unchanged in all experiments.

4.1 First experiment : validity of the extracted knowledge

This experiment aims at verifiying if the system is able to find again the complete rule set from a partial set of knowledge. This is accomplished by means of learning an example base built up with the complete set.

In a first test, we created a network by compiling 75% of the rules (3 among the 4 available rules). Then we applied the rule extraction method. The extraction process has been applied only on two units, the output and one hidden unit, because one unit only has been added to the network during

the learning period. We repeated such a test for all the configurations of the incomplete rule set : one rule eleminated among four available rules. In any case, the extraction method allowed to retrieve the rule removed from the initial set.

In a second test, we used another incomplete rule set constructed by suppressing 50% of the rules contained into the complete set. As in the first test, we refined this initial knowledge by using the original learning data set. The result we obtained is the same one: *we rediscovered all the rules eliminated from the original rule set.*

This set of experiments leads to two remarks :

- In any case, the retrieved rules were found by rule extraction from the ANN added units. That shows the process of modification of the network architecture is consistent.
- The fact we rediscover the eliminated rule means the removed rule was implicitly present in the examples learned by the neural network. The extracted knowledge is sound and not in contradiction to the example set.

Table 4.2. Monk1 problem: Using rules and examples to improve generalization

Portion of Rule Set	Portion of Examples Set	Generalization using INSS	Generalization Just rules	ANN Generaliz. Just examples
-	100%	100%	-	100%
-	75%	89.21%	-	89.21%
-	50%	70.92%	-	70.92%
100%	-	100%	100%	-
75%	-	83.33%	83.33%	-
50%	-	72.22%	72.22%	-
75%	100%	100%	83.33%	100%
50%	100%	100%	72.22%	100%
75%	75%	100%	83.33%	89.21%
75%	50%	100%	83.33%	70.92%
50%	75%	100%	72.22%	89.21%
50%	50%	89.86%	72.22%	70.92%

∗ Generalization scores represents the average obtained from 5 different runs
+ Our system and the data used in these tests are available for comparisons

4.2 Second experiment : efficiency of the hybridisation

The results obtained (see table 2) show that INSS is able to treat this problem using all available learning examples, or using a combination of the theoretical knowledge (rules) and empirical knowledge (examples). We showed that we always obtain a superior generalization rate when we use at the same time rules and examples. Lower generalization rates are obtained when we used just one information source at the same time.

5. Medical Diagnosis and other applications

In order to study the behavior of INSS on a real application, the system has been also tested on a medical diagnosis application : diagnosis of toxic coma. When a comatose patient is admitted in an emergency care unit, the clinician makes an early tentative diagnosis by collecting clinical and biological parameters. The diagnosis may be later confirmed or rejected by toxicological analysis. So, for the initial therapeutic action to be as adequate as possible, there is a need for an accurate prediction of the toxic cause, without waiting for the toxicological analysis. The use of an intelligent automated system to help in this diagnosis task seems to be very useful. Until now, there is no complete model for describing this knowledge by means of rules.

Our goal was to use INSS to aid to identify the causes of a psychotrope induced coma. We have available a case base of 505 pre-analysed examples of patients. Each example is described with 13 parameters or symptoms obtained directly when the patient is admitted, without waiting for the toxicological analysis. The diagnosis should aid to identify the presence or absence of each one of the 7 individual toxic causes (Alcohol, ADT, Benzodiazepines, Barbiturates, Carbamates, Morphine or Phenothiazines). A more detailed description of this problem can be found in the technical report of the Esprit MIX Project [1].

Table 5.1 reproduces the results we obtained comparing INSS to other machine learning systems applied to this medical diagnosis problem. All the systems were tested with exactly the same learning and testing data sets, and the results expressed in this table are the average of 10 different runs. The systems we compared with INSS are described in the MIX report [1].

The scores showed in Table 5.1, related to the other methods (K-PPV, C4.5, and ProBis), were reproduced from the results obtained by other researchers [21]. As we were constrained to use the same experiment protocol in order to be able to compare these different methods, we show here just a brief performance comparison. Although we published in [1] a more detailed list of the results obtained with INSS related to this problem.

Table 5.1. Comparison of the generalisation test rate after learning

Method → Class ↓	K-PPV	C4.5	ProBIS	INSS
Alcohol -E	66.56%	65.40%	68.94%	**74.50%**
ADT - a	55.39%	55.26%	57.63%	**60.79%**
Barbituriques - B	65.65%	63.32%	64.60%	**82.45%**
Benzodiazepines-b	62.37%	64.34%	63.95%	**83.37%**
Carbamates - c	81.58%	**87.64%**	84.87%	87.28%
Morphine - m	97.23%	97.50%	**97.97%**	97.88%
Phenothiazines - p	66.45%	71.26%	68.95%	**75.36%**

As we can observe from Table 5.1, the INSS system shows a remarkable performance in this task compared with the other techniques. In some classes the percentage differences between INSS and other methods are quite small, but INSS is always near to the best performance obtained. However we have to remark that in some classes we get relatively poor classification results for all methods. That is due to the intrinsic complexity of this problem and the strong overlapping of the different classes : this kind of complexity is a typical feature of the medical diagnosis.

We also tried to extract rules from the trained ANN. The extracted rules were presented to one expert of this domain, and he immediately recognised them as " valid rules". He also noted that a great part of the rules had captured important relations between certain input features and the presence of one specific toxic substance (e.g., intermediate pupil size, normal eye movement, low core temperature, and prolonged cardiological QT interval, are factors that indicate the possible presence of ADT).

This research resulted in the development of an experimental user interface to give access to our system through the WWW (World-Wide-Web). This program allows the consultation of the INSS system for toxic coma diagnosis. The user can fill-in a form with the patient's clinical and biological parameters and get back the ANN answer indicating the possible toxic substances absorbed. Presently, the system answer is based upon an ANN trained with the 505 cases database. However, this small number of available cases has proved to be insufficient for a good diagnosis of all toxic substances.

We are currently using the INSS system in two other domains: autonomous robot control and models of human cognitive development (e.g., balance scale problem [30, 31]). A description of our preliminary results obtained with these applications can be found in [25].

6. Conclusion

The INSS system presented here offers many advantages compared to the KBANN system by which it was inspired. This system has a better performance and allows incremental acquisition/extraction of network knowledge. Furthermore, it is based upon an incremental learning method already used to model human cognitive development. This learning method allowed us to develop a system perfectly adapted to the concepts proposed in the framework of constructive machine learning systems. The system was tested on different applications (classification tasks, medical diagnosis, autonomous robot control) obtaining satisfactory results. Actually our main goals are to develop a deeper study of the real-world applications of INSS, as well as to study the aspects related to the constructive acquisition of knowledge.

Our future work is the implementation of a complete hybrid machine learning tool based on ANN allowing diverse input and output forms of knowledge. Figure 6.1 shows the structure of such a system for introduction and

extraction of different forms of knowledge. In the future with the addition of interface routines for format conversion, such a system could process other forms of knowledge transforming it into and from Neural Networks.

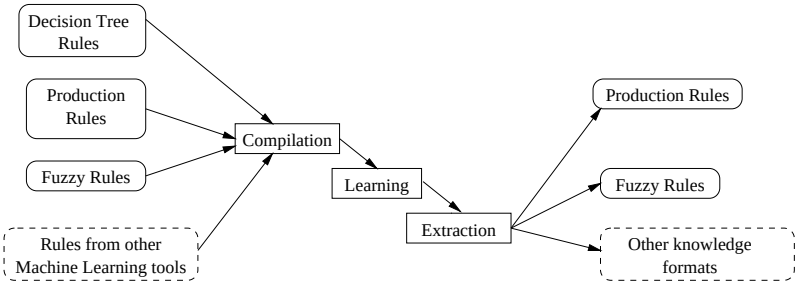


Fig. 6.1. Generic system for introduction and extraction of knowledge

Acknowledgements: Thanks to Scott Fahlman, Geoffrey Towell, Vicent Danel and Daniel Memmi for their contribution to our research and/or to the production of this article. We would like also to thank Prof. Wolfgang Rosenstiel and to the "Neuro-Group" at the Tübingen University for the enlightening discussions and support. This research was supported in part by a fellowship from CAPES-Brazil and Cofecub-France, by a fellowship from CNPq and by the german Government (DFG project).

References

1. Amy, B. et al. (1997) *MIX Medical Application (Final Report - Deliverable D16).* Technical Report, Project Esprit- BRA 'MIX' - European Community (May 1997). Web: http://www-leibniz.imag.fr/RESEAUX/public.html . Ftp: ftp://-ftp.imag.fr/pub/LEIBNIZ/RESEAUX-D-AUTOMATES/MIX-FMR.ps.gz
2. Andrews, R.; Diederich, J.; Tickle, A. B. (1995) *A Survey And Critique of Techniques For Extracting Rules From Trained ANN.* Technical Report - Neurocomputing Research Centre - QUT (Queensland University of Technology, Brisbane - Australia, January 1995). Also published in: Knowledge-Based Systems 8(6), p.378-38 (1995). Web: http://www.fit.qut.edu.au/NRC/ Ftp:ftp://-ftp.fit.qut.edu.au/pub/NRC/tr/ps/QUTNRC-95-01-02.ps.Z
3. Blond, P.-Y. (1996) *Validation de Connaissances dans un Système Hybride Neuro-Symbolique à Apprentissage Incrémental.* Rapport du D.E.A. en Sciences Cognitives (Lab. LEIBNIZ - IMAG, Grenoble - France, 1996).
4. Boz, Olcay (1995) *Knowledge Integration and Rule Extraction in Neural Networks.* Technical Report, EECS, Lehigh University. (October 1995). Web: http:// www.lehigh.edu/ ob00/ integrated.html
5. Cechin, A.: *The Extraction of Fuzzy Rules from Neural Networks.* Fakultät für Informatik der Eberhard-Karls-Universität zu Tübingen, Shaker Verlag, Aachen, ISBN 3-8265-3541-3, 1998.

6. Cechin, A., Epperlein, U., Koppenhoefer, B. and Rosenstiel, W.: *The Extraction of Sugeno Fuzzy Rules from Neural Networks.* in Michel Verleysen (editor): *Proceedings of the European Symposium on Artificial Neural Networks*, 49–54, Brussels, Belgium, 1996. D facto publications.
7. Cechin, A. and Eppler, W.: *Automatic Design of a Fuzzy Controller from a Neural Process Modell.* in H.-J. Zimmermann (editor): *Proceedings of the 2nd European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany, 1994.
8. Cechin, A. and Eppler, W.: *Automatischer Entwurf eines Fuzzy-Reglers aus einem neuronalen Prozeßmodell.* in B. Reusch (editor): *4. Dortmunder Fuzzy-Tage*, Dortmund, Germany, 1994. Poster.
9. Cechin, A., Epperlein, U., Koppenhoefer, B. and Rosenstiel, W.: *The Extraction of Sugeno Fuzzy Rules from Neural Networks.* in Robert Andrews and Joachim Diederich (editors): *Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop*, 16–24, Brighton, United Kingdom, 1996.
10. Cechin, A., Epperlein, U., Koppenhoefer, B. and Rosenstiel, W.: *Fuzzy Rules Extraction from Neural Networks Applied to Molecular Recognition.* in A.B. Bulsari, S. Kallio and D.T. Tsaptsinos (editors): *Proceedings of the International Conference on Engineering Applications of Neural Networks*, 107–110, London, United Kingdom, 1996.
11. Decloedt, L., Osorio, F., Amy, B. (1996) RULE_OUT Method: A New Approach for Knowledge Explicitation from Trained Artificial Neural Networks. *Proceedings of the AISB'96 – Workshop on Rule Extraction from Trained Neural Nets, p.34-42.* (QUT - Andrews and Diederich Eds. 1996). Web: http:// www-leibniz.imag.fr/ RESEAUX/ public.html Ftp: ftp:// ftp.imag.fr/ pub/ LEIBNIZ/ RESEAUX-D-AUTOMATES/ osorio.aisb96.ps.gz
12. Decloedt, Loïc (1995) *Explicitation de Connaissances dans un Système Hybride d'Intelligence Artificielle.* Rapport du D.E.A. en Informatique, Laboratoire LIFIA – IMAG, INPG (Grenoble – France, Juin 1995). Web: http:// www-leibniz.imag.fr/ RESEAUX/ public. html Ftp: ftp:// ftp.imag.fr/ pub/ LEIBNIZ/ RESEAUX–D–AUTOMATES/ DEA/ decloedt.dea.ps.gz
13. Eppler, W., Gemmeke, H. and Cechin, A.: *Stabilization of a Stratospheric Balloon Experiment by a New Fuzzy Controller with a Neural Process Model.* in *IEEE Nuclear Science Symposium*, 422–426, Norfolk, USA, 1994.
14. Fahlman, S. E., Lebiere, C. (1990) *The Cascade-Correlation Learning Architecture.* Carnegie Mellon Un., Technical Report - CMU-CS-90-100. (1990) Web: http:// www.cs.cmu.edu/ Reports/ index.html . Ftp: ftp:// archive.cis.ohio-state.edu/ pub/ neuroprose/ fahlman.cascor-tr.ps.Z
15. Giacometti, A. (1992) *Modèles hybrides de l'expertise.* Thèse de Doctorat, LIFIA - IMAG (Grenoble – France, 1992). Web: http:// www-leibniz.imag.fr/ RESEAUX/ public.html . Ftp: ftp:// ftp.imag.fr/ pub/ LEIBNIZ/ RESEAUX-D-AUTOMATES/ giacometti. these. ps.tar.gz
16. Giarratano, Joseph C. (1993) *CLIPS User's Guide - Version 6.0.* Lyndon B. Johnson Space Center, Software Technology Branch, NASA (U.S.A., 1993). Web: http:// www.jsc.nasa.gov/ clips/ CLIPS.html or http:// home.haley.com/ clips.html Ftp: ftp:// ftp.cs.cmu.edu/ afs/cs/ project/ ai-repository/ ai/areas/ expert/ systems/ clips/ 0.html
17. Hilario, M. (1996) An Overview of Strategies for Neurosymbolic Integration. In: *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches.* Ron Sun (Ed.) - Chapter 2. (Kluwer Academic Publishers, 1996). Ftp: ftp://-cui.unige.ch/AI/
18. Jang, J.-S.R. and Sun, C.T.: *Neuro-Fuzzy Modeling and Control.* Proceedings of the IEEE, 83:378–406, 1995.

19. Jang, J.-S.R. and Sun, C.T.: *Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems.* IEEE Transactions on Neural Networks, 4:156–159, 1993.
20. Kruse, R., Gebhardt, J. and Klawonn, F.: *Fuzzy-Systeme.* B.G.Teubner, Stuttgart, 1993.
21. Malek, Maria. (1996) *Un modèle hybride de mémoire pour le raisonnement à partir de cas.* Thèse de Doctorat, UJF - LEIBNIZ (Grenoble - France, 1996). Web: http:// www-leibniz.imag.fr/ RESEAUX/ public.html . Ftp: ftp:// ftp.imag.fr/ pub/ LEIBNIZ/ RESEAUX-D-AUTOMATES/ malek.these.ps.gz
22. Oelhaf, H., Clarmann, T., Fergg, F., Fischer, H., Friedl-Vallon, F., Fritzsche, C., Piesch, C., Rabus, D., Seefeldner, M. and Völker, W.: *Remote Sensing of Trace Gases with a Ballon Borne Version of the Michelson Interferometer for Passive Atmospheric Sounding (MIPAS).* in *Proceedings of the 10th ESA-Symposium on European Rocket and Balloon Programmes,* 207–213, Mandelieu-Cannes, France, 1991.
23. Orsier, Bruno (1995) *Etude et application de systèmes hybrides neuro-symboliques.* Thèse de Doctorat, UJF - LIFIA (Grenoble - France, 1995). Web: http:// www-leibniz.imag.fr/ RESEAUX/ public.html . Ftp: ftp:// ftp.imag.fr/ pub/ LEIBNIZ/ RESEAUX-D-AUTOMATES/ orsier.these.ps.gz
24. Osorio, F. S. & Amy, B. (1995) *INSS: A Hybrid Symboli-Connectionist System that Learns from Rules and Examples (text in portuguese).* Panel'95 - XXI Latin-American Conference on Computer Science (Canela, Brazil, August 1995). Web: http:// www-leibniz.imag.fr/ RESEAUX/ osorio/ papers/ diret.html
25. Osorio, F. S. (1998) *INSS: Un Système Hybride Neuro-Symbolique pour l'Apprentissage Automatique Constructif.* Thèse de Doctorat, INPG - Laboratoire LEIBNIZ - IMAG (Grenoble, France, 1998). Web: http:// www-leibniz.imag.fr/ RESEAUX/ public.html . Ftp: ftp:// ftp.imag.fr/ pub/ LEIBNIZ/ RESEAUX-D-AUTOMATES/ osorio.these.ps.gz
26. Reyes, Gerardo (1997) *Etude des Connaissances dans les Réseaux de Neurones Artificiels : Représentation et Explicitation de Règles d'Haut Niveau.* Rapport du D.E.A. en Sciences Cognitives, LEIBNIZ - IMAG. (Grenoble - France, 1997).
27. Reyes, G.; Osorio, F.; Amy, B. (1997) Neural Learning of "High Level Rules": The Balance Scale Problem. In: *HELNET'97 International Workshop on Neural Networks.* (Montreux, Swiss, October 1997). Web: http:// www-leibniz.imag.fr/ RESEAUX/ public.html . Ftp: ftp:// ftp.imag.fr/ pub/ LEIBNIZ/ RESEAUX-D-AUTOMATES/ osorio.helnet97.ps.gz
28. Rumelhart, D., Hinton, G., Willians, R. (1986) Learning Internal Representations by Error Propagation. In: *Parallel Distributed Processing - Explorations in the Microstructure of Cognition,* V1. (Cambridge - MIT Press, 1986).
29. Schiffmann, W.; Joost, M. & Werner, R. (1993) Comparison of Optimized Backpropagation Algorithms. In: *Proceedings of the European Symposium on Artificial Neural Networks, ESANN'93.* p.97-104. (Brussels, 1993) Web: http:// www.uni-koblenz.de/ schiff/ publications.html
30. Shultz, T. R., Schmidt, W. C. (1991) A Cascade-Correlation Model of Balance Scale Phenomena. In: *Proceedings of the Thirteenth Annual Conf. of the Cognitive Science Society.* p.635-640. (Hillsdale, NJ - Erlbaum, 1991). Web: http:// www.psych.mcgill.ca/ labs/ lnsc/ html/ Lab-Home.html (or Pub-cog-dev.html). Ftp: ftp:// ego.psych.mcgill.ca/ pub/ shultz/ balcog.ps.gz
31. Shultz, T. R., Mareschal, D., Schmidt, W. (1994) Modeling Cognitive Development on Balance Scale Phenomena. In: *Machine Learning.* No.16, p.57-86. (Kluwer Publishers, 1994). Web: http:// www.psych.mcgill.ca/ labs/ lnsc/

html/ Lab-Home.html (or Pub-cog-dev.html). Ftp: ftp:// ego.psych.mcgill.ca/ pub/ shultz/ balml.ps.gz
32. Sun, Ron & Alexandre, Frederic (1997) *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches.* (Lawrence Erlbaum Associates, 1997).
33. Takagi T. and Sugeno, M.: *Fuzzy Identification of Systems and Its Application to Modeling and Control.* IEEE Transactions on Systems, Man, and Cybernetics, 15:116–132, 1985.
34. Thrun, S. B. et al. (1991) *The Monk's Problem - A Performance Comparison of Different Learning Algorithm.* Carnegie Mellon University, Technical Report CMU-CS-91-197. (1991). Web: http:// www.cs.cmu.edu/ thrun/ . Ftp: ftp:// archive.cis.ohio-state.edu/ pub/ neuroprose/
35. Towell, G. (1991) *Symbolic Knowledge and Neural Networks: Insertion, Refinement and Extraction.* Ph.D. Thesis, University of Wisconsin-Madison - Computer. Science Dept. (1991). Web: http:// www.cs.wisc.edu/ shavlik/ uwml.html Ftp: ftp:// ftp.cs.wisc.edu/ machine-learning/ shavlik-group/ (towell.thesis.*.ps)
36. Towell, G. & Shavlik, J. (1993) Extracting Refined Rules From Knowledge-Based Neural Nets. In: *Machine Learning.* pp.71-101, 13. (Kluwer Academic Publishers - Boston, 1993). Web: http:// www.cs.wisc.edu/ shavlik/ uwml.html . Ftp: ftp:// ftp.cs.wisc.edu/ machine-learning/ shavlik-group/ towell.mlj93.ps
37. Towell, G. & Shavlik, J. (1994) Knowledge-Based Neural Nets. In: *Artificial Intelligence.* pp.119-165, 70. (1994). Web: http:// www.cs.wisc.edu/ shavlik/ uwml.html . Ftp: ftp:// ftp.cs.wisc.edu/ machine-learning/ shavlik-group/ towell.aij94.ps
38. UCI-ML Repository (1997) *UCI - University of California, Irvine Machine Learning Databases and Domain Theories.* (1997). Web: http:// www.ics.uci.edu/ mlearn/ MLRepository.html . Ftp: ftp:// ftp.ics.uci.edu/ pub/ machine-learning-databases/ README