ELSEVIER Fuzzy Sets and Systems 82 (1996) 135–149

FUZZY sets and systems

Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems

Nikola K. Kasabov*

Department of Information Science, University of Otago, P.O. Box 56, Dunedin, New Zealand

Abstract

The paper considers both knowledge acquisition and knowledge interpretation tasks as tightly connected and continuously interacting processes in a contemporary knowledge engineering system. Fuzzy rules are used here as a framework for knowledge representation. An algorithm REFuNN for fuzzy rules extraction from adaptive fuzzy neural networks (FuNN) is proposed. A case study of Iris classification is chosen to illustrate the algorithm. Interpretation of fuzzy rules is possible by using fuzzy neural networks or by using standard fuzzy inference methods. Both approaches are compared in the paper based on the case example. A hybrid environment FuzzyCOPE which facilitates neural network simulation, fuzzy rules extraction from fuzzy neural networks and fuzzy rules interpretation by using different methods for approximate reasoning is briefly described.

Keywords: Learning fuzzy rules; Neural networks; Fuzzy neural networks; Knowledge acquisition; Approximate reasoning

1. Introduction

Two are the major tasks of the contemporary research in knowledge engineering:
- knowledge acquisition/knowledge refinement, and
- knowledge interpretation.

The two tasks are separate steps in building a knowledge-based system, but they are strongly connected. For example, the process starts with initial set of prior or extracted from data rules. The rules are interpreted. During the interpretation new data has been entered and new results have been obtained which should reflect in a refined knowledge and this process is continuous as depicted in Fig. 1.

The above approach of tightening the two processes together in one system is very relevant to solving many AI tasks. It is very much human-like, as humans learn, reason, and explain in a continuous manner over time. Developing knowledge engineering tools which facilitate this approach is a major concern of the research reported here.

The paper has the following organisation. Section 2 presents general issues of approximate reasoning. Section 3 introduces fuzzy neural networks and a model called FuNN for both rules extraction and reasoning. Section 3 discusses issues of connectionist methods for fuzzy rules extraction and also presents an algorithm called REFuNN. The algorithm is illustrated with the well-known Iris database. Section 4 compares results obtained after reasoning in FuNN and by using a standard

*Email: nkasabov@otago.ac.nz.

max–min composition fuzzy inference method. Both produce 100% correct classification on a selected Iris test data set. Section 5 finally suggests a knowledge engineering environment Fuzzy-COPE for building comprehensive AI systems. It facilitates rules extraction and different methods for approximate reasoning. Section 6 gives conclusions and directions for further research.

2. Approximate reasoning – issues and problems

Approximate reasoning is a process of interpretation of knowledge in a presence of uncertainty. The uncertainty can be present in a form of vague and contradictory knowledge, incomplete past data, uncertain new facts, not clear goals, etc. Fig. 2 gives examples of three different knowledge representation schemes which allow for representing uncertainties on a case study of three simple rules. Different representation schemes influence the type of approximate reasoning techniques which can be used. *Simple fuzzy rules* [25, 24, 20, 16, 17]. *weighted production rules* [20, 9, 14] and *generalised fuzzy production rules* [9, 14] are shown there as representation schemes as well as a basis for applying approximate reasoning mechanisms.

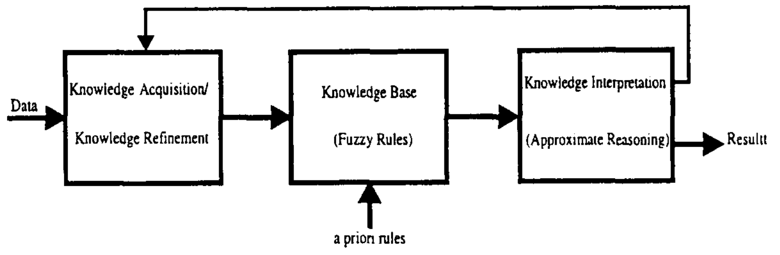

Fig. 1. Knowledge acquisition and knowledge interpretation as two tightly coupled phases in one system.

Simple fuzzy rules

Rule 1: IF x_1 is Medium AND x_2 is Medium THEN y is Medium

Rule 2: IF x_1 is High AND x_2 is High THEN y is High

Rule 3: IF y is Medium THEN z is Medium

Weighted fuzzy production rules

Rule 1: IF x_1 is Medium with $DI_{1,1}=2$ AND x_2 is Medium with $DI_{1,2}=1$ THEN y is Medium ($CF_1=0.8$);

Rule 2: IF x_1 is High with $DI_{2,1}=5$ AND x_2 is High with $DI_{2,2}=2$ THEN y is High ($CF_2=0.6$)

Rule 3:IF y is Medium THEN z is Medium ($CF_3=0.5$),

where: DI_{ij} are degrees of importance attached to the condition elements; CFi are certainty factors attached to the consequent (action) elements;

Generalised fuzzy production rules

Rule 1:IF [x_1 is Medium with $DI_{1,1}=2$ AND x_2 is Medium with $DI_{1,2}=1$] ($NT_1=0.2$, $SF_1=0.6$) THEN y is Medium ($CF_1=0.8$)

Rule 2:IF [x_1 is High with $DI_{2,1}=5$ AND x_2 is High with $DI_{2,2}=2$] ($NT_2=0.2$, $SF_2=0.6$) THEN y is High ($CF_2=0.6$)

Rule 3:IF y is Medium ($NT_3=0.2$, $SF_3=0.6$) THEN z is Medium ($CF_3=0.5$),

where: NT_i are noise tolerance coefficients; SF_i are sensitivity factors; DI_{ij} are degrees of importance; CFiare certainty factors.

Fig. 2. Examples of three different knowledge representation schemes on a case study of three simple rules.

Approximate reasoning methods are concerned with the following general issues:

- How new facts, e.g. x'_1, x'_2 should be *matched* to the condition elements in the rules and how *partial match* should be evaluated;
- How to *combine* the partially matched by the new facts x'_1 and x'_2 condition elements in each of the rules R_1 and R_2 and evaluate the matching of the whole antecedent part in them;
- How to *calculate* the inferred through a rule new fact y'_i for each of the rules R_i;
- How to *aggregate* the inferred values y'_1 and y'_2 in one value y';
- How to *propagate* an inferred fact y' to the next reasoning cycle in the reasoning chain, e.g. how to *match* the inferred value of y' to the third rule R_3 and produce an output value z'.

In addition to the above said, the following issues have to be considered when creating more sophisticated reasoning methods:

- *dynamic fact-changes*, i.e. how to implement partial *'forgetting'* of nonrelevant facts;
- *dynamic rule-changes*, i.e. how rules can change over time, or – as a result of new data and knowledge, or – as a result of changes in the environment; how new rules can be created and old ones – corrected;
- *'communication' between rules* within a knowledge base, i.e. how pieces of knowledge can 'communicate' in order to improve themselves or their performance.

Different techniques offer different solutions to the problems above, none of them so far being able to meet all the requirements. The symbolic methods of AI fail to provide comprehensive approximate reasoning techniques. The well-established methods of probability theory can handle uncertainties when they are strictly represented in the terms of this theory [2, 24]. They are not suitable for chain reasoning or to represent subjective knowledge. *Fuzzy systems* have been widely applied for control and decision making. They use vague, linguistic, fuzzy knowledge and numerical representation (membership functions) to define the fuzzy terms. Some limitations of applying fuzzy logic are experienced in learning and adaptation. These are tasks which *neural networks* and connectionist models can handle. Mixing fuzzy and connectionist models for building approximate reasoning systems is more than a promising approach [3]. In spite of the advances in the area of fuzzy neural networks (FNN) (also called neuro-fuzzy systems [7, 15]) and their applications for learning and adaptation [1, 3, 23], for rules extraction [1, 3, 6, 7, 11, 15], for modelling, control and decision making [7, 1, 4, 9], etc., their potential for learning and reasoning in a hybrid knowledge engineering environment is still to be explored and effectively applied.

3. Fuzzy neural networks. The FuNN architecture for rules extraction and approximate reasoning

3.1. Fuzzy neural networks – a general introduction

A fuzzy neural network (FNN) is a connectionist model for fuzzy rules implementation and inference. There is a big variety of architectures and functionalities of FNN. Adaptive network-based fuzzy inference systems are discussed in [1, 4–7, 15, 18]. Fuzzy neural networks have been implemented and used as reported in [19, 22, 23, 11, 13].

The FNN developed so far differs mainly in the following parameters:

- *Type of fuzzy rules* implemented; this reflects in the connectionist structure used.
- *Type of inference* method implemented; this reflects in the selection of different neural network parameters and neuronal functions, such as summation, activation, output function; it also influences the way the connection weights are initialized before training, and interpreted after training.
- *Mode of operation*; we shall consider here three major modes of operation as suggested in [11].
  - *Fixed mode* – 'fixed membership functions-fixed set of rules', i.e. fixed set of rules is inserted in a network; the network performs inference, but does not change its weights. It cannot learn and adapt. A representative of this type of systems is NPS [9, 14].
  - *Learning mode, i.e.* a neural network is structurally defined to capture knowledge in a certain format, e.g. – some type of fuzzy rules. The network architecture is randomly initialized and

trained with a set of data. Rules are then extracted from the structured network [8]. The rules can be interpreted either in the same network structure or by using other inference methods.

– *Adaptation mode* – A neural network is structurally set according to a set of existing rules, 'hints', heuristics. The network is then trained with new data and then updated rules are extracted from its structure. There are two cases which can be distinguished here: '*fixed membership functions – adaptable rules*' [15] and '*adaptable membership functions – adaptable rules*' [1, 5]. The '*catastrophic forgetting*' phenomenon must be investigated in these cases, i.e. how much the network forgets about previous data after having learned from completely new data without rehearsing the old ones [11].

To summarise the above, FNN have two major aspects:

● *Structural*, i.e. a set of rules is used to define the initial structure of a neural network; two types of neural networks have been mainly used so far:
– multi-layer perceptrons (MLP) [1, 5, 11, 6];
– radial-basis functions networks [11];

● *functional, parametric*, i.e. after having defined the structure of a neural network and possibly having trained it with data, some parameters can be observed which parameters would explain the inference which the network performs [3, 19, 11]. Those parameters can be used to derive a (fuzzy) rule-based system represented in linguistic terms.

3.2. The FuNN model

The FuNN model [11] facilitates *learning from data, fuzzy rules extraction, approximate reasoning.*

FuNN uses a MLP network and a backpropagation training algorithm. It is adaptable FNN where the membership functions of the fuzzy predicates, as well as the fuzzy rules inserted before training (adaption) may adapt and change according to the training data. The general architecture of FuNN consists of the following layers (Fig. 3):

● *Input layer*; a node here represents an input variable.

● *Condition elements layer*; each node here represents a fuzzy predicate of the input variables. The activation values of the nodes represent the membership degrees of the input variables. Different summation function s_C, activation function a_C and output function o_C can be used for the neurons of this layer.

● *Rule layer* – each node in this layer represents either an existing rule, or – an anticipating after training rule. When FuNN is used to implement

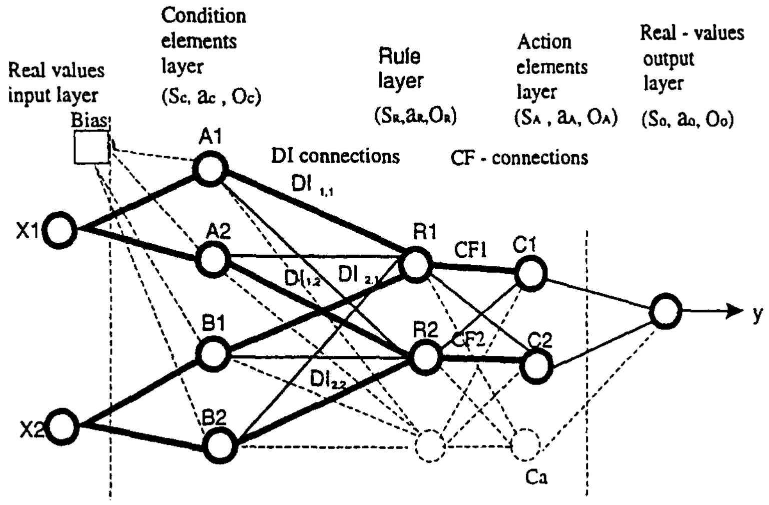

Fig. 3. An exemplar FuNN architecture for a simple set of two fuzzy rules.

initial set of fuzzy rules, then the connections between the condition elements layer and the rule layer are set according to normalized degrees of importance attached to the antecedent elements in the corresponding rules. If degrees of importance are not attached to the condition elements of a rule R_i, then the connection weights w_{ij} to a rule node R_i are uniformly calculated for each connection as

$$w_{ij} = \text{Net}_i/n,$$

where n is the number of condition elements in the rule R_i; Net_i is a constant which defines what the net input to the neuron R_i should be in order to fire the rule. The stronger the rule is as a piece of domain knowledge, the higher Net_i should be, which means a higher contribution of this rule to the output value. For a weak rule R_i, Net_i might take a value of 1, and for a strong rule Net_i might need to be 5, provided a sigmoid activation function is used. The other connection weights are initialised to zero. The following characteristics of this layer define the inference method performed by the FuNN: summation function s_R; activation function a_R; neuronal output function o_R. Additional rule nodes may be pre-set with zero connection weights. This may give the structure more flexibility to adjust the initial rules and the antecedent elements in them and to possibly capture new rules. The way the connection weights are interpreted here is used in the rules extraction algorithm REFuNN presented in the next section.

- *'Action' elements layer*; each node in this layer represents one fuzzy predicate (label) in the 'action' (consequent) elements of the rules. The connections between the rule nodes and the 'action' nodes are set as normalized certainty factors (CF) of the rules. The rest of the connections are set to zero. Again, three functions are defined for these nodes, i.e. summation function s_A, activation function a_A and output function o_A. Additional nodes may be used to capture additional action (conclusion) predicates during training (adaptation).
- *Output variable layer*; it represents the output variables of the system. It is defined by the three functions: summation s_O, activation a_O and output o_O functions.

Fig. 3 depicts a FuNN for the following two rules:

R_1: *IF* x_1 *is* A_1 $(DI_{1,1})$ *and* x_2 *is* B_1 $(DI_{2,1})$
THEN y *is* C_1 (CF_1)
R_2: *IF* x_1 *is* A_2 $(DI_{1,2})$ *and* $x2$ *is* B_2 $(DI_{2,2})$
THEN y *is* C_2 (CF_2)

An algorithm REFuNN for rules extraction from a trained FuNN is presented in the next section. The algorithm uses three layers from the FuNN architecture shown in Fig. 3 as the fuzzy predicates and membership functions are predefined. Fuzzyfication and defuzzification are supposed to be done outside the structure.

4. Connectionist methods for learning fuzzy rules. The REFuNN algorithm

4.1. A general discussion on learning fuzzy rules

Connectionist methods for learning rules from data use a connectionist structure, trained with data, and analyse the connection weights to extract rules. There are different methods which can be applied for rules extraction from a trained neural network architecture. They can be grouped into three major groups:

- *Destructive learning methods*, this involves learning by pruning the neural network structure during the training procedure
- *Nondestructive learning methods*; here the network structure is kept intact during training; it is anlaysed and rules are extracted afterwards.

Learned (or articulated) initial set of rules can be used to '*pre-wire*' a neural network before its training with data as it is the case with the FuNN architecture discussed in the previous section.

Fuzzy rules can be learned based on:

- fuzzified data and pre-defined membership functions, i.e. the data used for training is fuzzified by using pre-defined membership functions for the fuzzy predicates [16, 8, 11];
- crisp data and pre-defined membership functions, i.e. the data used for training is not fuzzified but the membership functions are predefined [1, 5, 11]; these methods allow for tuning the

membership functions during further training or adaptation of the system;
• crisp data and not pre-defined membership functions, i.e. the number and the shape of the membership functions are learned during training [1, 5, 6].

Extracting rules from a trained connectionist structure may subsequently mean loss of information, as the *way* knowledge is extracted restricts the aspects of that knowledge and also directs and biases the knowledge acquisition process. A data set in general, contains much more than it can be extracted from it by using a particular connectionist method. Rules extraction process ends up with an abstract, concise, condensed representation of one aspect only, may be the most important one for a particular application.

4.2. The REFuNN algorithm – Rules extraction from a fuzzy neural network

The REFuNN algorithm, which first version was published in [8], is a simple connectionist method for extracting weighted fuzzy rules and simple fuzzy rules as illustrated in Fig. 2. It is based on training a MLP architecture with fuzzified data. The REFuNN algorithm, outlined below, is based on the following principles [8]:

(1) simple operations are used and a low computational cost achieved;

(2) hidden nodes in a MLP can learn features, rules, groups in the training data;

(3) fuzzy quantization of the input and the output variables are done in advance; the granularity of the fuzzy representation (the number of fuzzy lables used) defines in the end the 'fineness' and the quality of the extracted rules. Standard, uniformly distributed triangular membership functions can be used for both fuzzy input and output labels;

(4) automatically extracted rules may need additional manipulation depending on the reasoning method applied afterwards.

The Algorithm

Step 1: *Initialisation of a FuNN.* A fully connected MLP neural network is constructed as shown in the example in Fig. 3 (the internal structure only between the two dashed vertical lines). The functional parameters of the rule layer and the output fuzzy predicates layer are set as follows: summation input function; sigmoid activation function; direct output function.

Step 2: *Training the FuNN.* Supervised training algorithm is performed for training the network with fuzzified data until convergence. Backpropagation training algorithm can be used.

Step 3: *Extracting initial set of weighted rules.* A set of rules $\{r_j\}$ is extracted from the trained network as follows. All the connections to an action element neuron C_j which contribute significantly to its possible activation (their values, after adding the bias connection weight if such is used, are over a defined threshold $\mathrm{Th_a}$), are picked up and their corresponding hidden nodes R_j, which represent a combination of fuzzy input lables, are analysed further on. Only condition element nodes which support activating the chosen hidden node R_j will be used in the antecedent part of a rule r_j (the connection weights are above a threshold $\mathrm{Th_c}$). The weights of the connections between the condition-element neurons and the rule-nodes are taken as initial relative degrees of importance of the antecedent fuzzy propositions. The weights of the connections between a rule node R_j and an action-element node C_j define initial value for the certainty degree CF_j. The threshold $\mathrm{Th_c}$ can be calculated by using the formula:

$$\mathrm{Th_c} = \mathrm{Net_{max}}/k,$$

where $\mathrm{Net_{max}}$ is the desired value for the net input to a rule neuron to fire the corresponding rule; k is the number of the input variables.

Fig. 6 shows an extracted initial set of weighted rules for 3 fuzzy labels from a FuNN trained with Iris fuzzified data (this is explained later in this section).

Step 4: *Extracting simple fuzzy rules from the set of weighted rules.* The threshold $\mathrm{Th_c}$ used in Step 3 was defined in such a way that all the condition elements in a rule should *collectively* trigger the activation of this rule. This is analogues to an '*AND*' connective. The number of the fuzzy predicates allowed to be represented in the antecedent part of a rule is not more than the number of the input variables (one fuzzy predicate per variable at the most). The initial set of weighted rules can be converted into a set of *simple fuzzy rules* by simply removing the weights from the condition elements. Some antecedent elements however can trigger the rules without any support from the rest of the condition elements, i.e. their degrees of importance $\mathrm{DI}_{ij} = w_{ij}$ (connection weights) are higher than the threshold $\mathrm{Th}_{\cdot\mathrm{OR}\cdot} = \mathrm{Net}_{\max}$. Such condition elements form separate rules which transformation is analogous to a decomposition of rules with OR-connectives into rules with AND-connectives only.

Example. IF there is an initial weighted rule

IF x_1 is $A(8.3)$ and x_2 is $B(1.2)$ THEN y is C,

and a threshold of $\mathrm{Th}_{\cdot\mathrm{OR}\cdot} = 5.0$ is chosen, then two separate simple fuzzy rules will be formed:

IF x_1 is A and x_2 is B THEN y is C,

IF x_1 is A THEN y is C.

The '*AND*' and '*OR*' connectives used here are vague, weak and loosely defined. An '*AND*' connective should rather be expressed as a '*mutual support*' between variables or – *synergism* [19].

Step 5: *Aggregating the initial weighted rules.* All the initial weighted rules $\{r_{i1}, r_{i2} \ldots\}$ which have the same condition elements and the same consequent elements, subject only to different degrees of importance, are aggregated into one rule. The relative degrees of importance DI_{ij} are calculated for every condition element A_{ij} of a rule R_i as a normalized sum of the initial degrees of importance of the corresponding antecedent elements in the initial rules r_{ij}.

An additional option in REFuNN is learning *NOT* connectives in the rules. In this case negative weights which absolute values are above the set thresholds Th_c and Th_a are considered and the input labels corresponding to the connected nodes are included in the formed simple fuzzy rules with a NOT connective in front.

Case example – Iris Classification Problem: As a case example the well understood and widely used in the machine learning community Iris data set is chosen. The whole data set comprises 50 instances of each of the three Iris classes – Setosa, Versicolor and Virginica. The instances are represented by four input attributes as follows: sepal length (SL), sepal width (SW), petal length (PL) and petal width (PW), all measured in cm. 120 examples are used for training and for rule extraction in the experiments below. 30 examples (10 examples of each of the classes) are used for testing the extracted rules. Fig. 4 shows a mapping of the whole Iris data

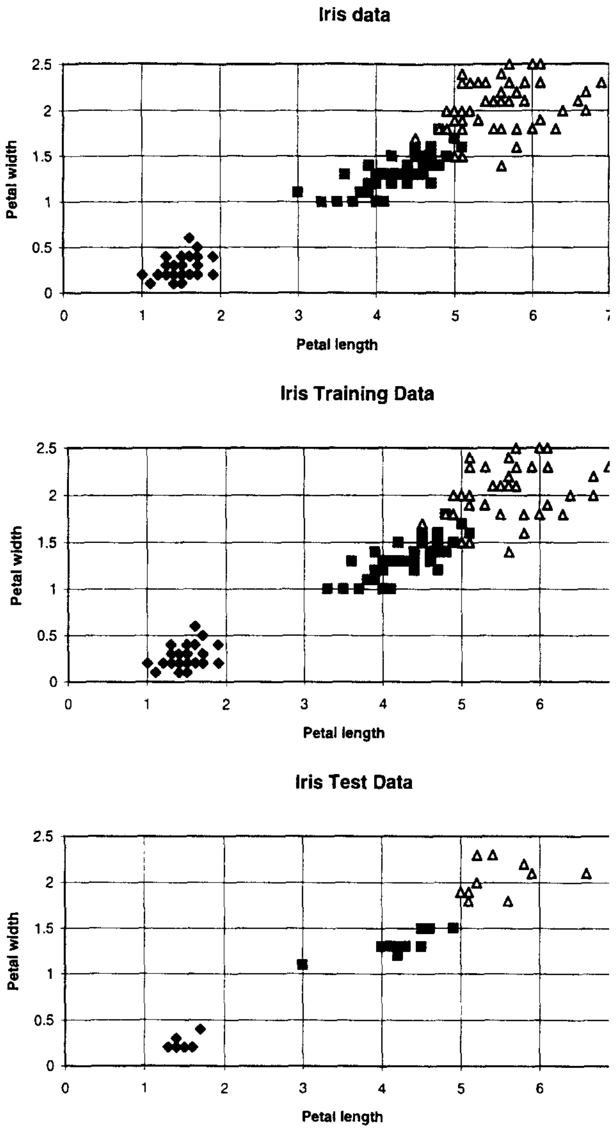

Fig. 4. Iris data sets mapped into the input space of the last two input attributes (diamond – Setosa; square – Versicolor; triangle – Virginica)

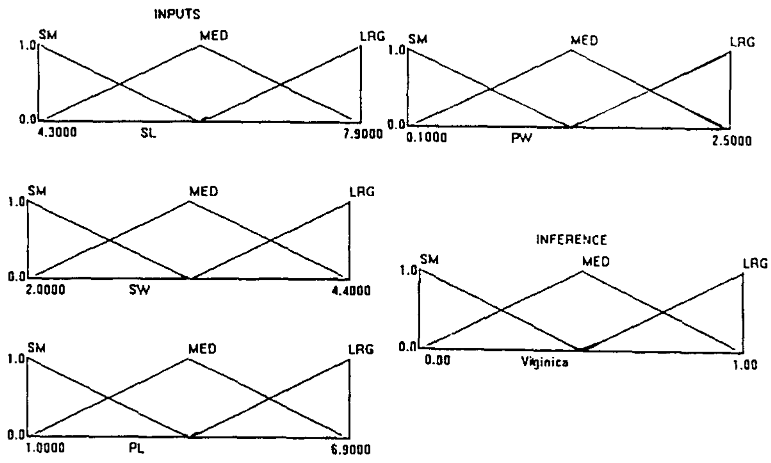

Fig. 5. Three membership functions used to represent each of the input variables and each of the output variables (only Virginica is shown here). The used abbreviations are as follows: SM – small; MED – medium; LRG – large.

set, the training data and the test data into a two-dimensional space of the last two attributes only.

By using the REFuNN algorithm several sets of fuzzy rules were extracted. First, three fuzzy predicates (Small – Sm, Medium – Med, and Large – Lrg) were used to represent each of the four input attributes and each of the three output variables, the latter representing possibilities for a data example to be classified into one of the three classes as shown in Fig. 5.

A FuNN structure having 12 input nodes, 6 intermediate nodes and 9 output nodes was trained with fuzzified Iris training data for 1000 training cycles, a learning rate of 0.1 and momentum of 0.3 until a RMS error of 0.023. Fuzzy rules were then extracted. Fig. 6 shows the set of initial weighted rules, the sets of simple rules for each of the classes and a set of rules which have the last two input variables only. It can be seen from the list of extracted rules that the most important for the classification task attributes are petal length and petal width.

Another set of rules was extracted when 5 membership functions were used for the input and the

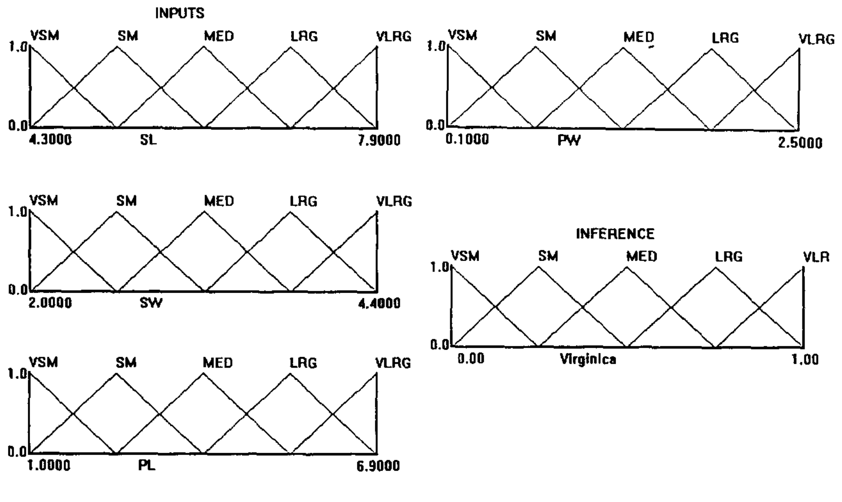

Fig. 7. Five membership functions for the input and the output variables for the Iris data set.

Extracted Iris classification rules for 3 membership functions and thresholds Th_a=Th_c= 2.0
(a) Weghted Rules:
if <SL is Sm 2.5> and <SW is Sm 4.2>and<PL is Lrg 8.8> and <PW is Lrg 10.8> then <Set is Sm 4.7> if <PL is Med 3> and <PW is Med 3.5> then <Set is Sm 2.2>
if <PL is Sm 2.2> and <PW is Sm 2.21> then <Set is Lrg 2.2>
if <PL is Sm 2.3> and <PW is Sm 2.4> then <Versi is Sm 4.9>
if <SL is Sm 2.5> and <SW is Sm 4.2> and <PL is Lrg 8.8> and <PW is Lrg 10.8> then <Versi is Sm 7.8>
if <PL is Sm 2.2> and <PW is Sm 2.2> then <Versi is Sm 4.9>
if <PL is Med 3> and <PW is Med 3.5> then <Versi is Lrg 3>
if <PL is Med 3> and <PW is Med 3> then <Virgi is Sm 3.8>
if <SL is Sm 2.5> and <SW is Sm 4.2>and<PL is Lrg 8.8> and <PW is Lrg 10.8> then <Virgi is Lrg 8.3>

(b) Simple rules for Th_{OR} =5.0
RULES for Setosa
if <SL is Sm> and <SW is Sm> and <PL is Lrg>and<PW is Lrg> then <Set is Sm>
if <PL is Lrg> then <Set is Sm>
if <PW is Lrg> then <Set is Sm)
if <PL is Med> and <PW is Med> then <Set is Sm>
if <PL is Sm> and <PW is Sm> then <Set is Lrg>
RULES for Versicolor
if <PL is Sm> and <PW is Sm> then <Versi is Sm>
if <SL is Sm> and <SW is Sm> and <PL is Lrg>and<PW is Lrg> then <Versi is Sm>
if <PL is Med> and <PW is Med> then <Versi is Lrg>
if <PL is Lrg> then <Versi is Sm>
if <PW is Lrg> then <Versi is Sm>
RULES for Virgiica
if <PL is Med> and <PW is Med> then <Virgi is Sm>
if <SL is Sm> and <SW is Sm> and <PL is Lrg>and<PW is Lrg> then <Virgi is Lrg>
if <PL is Lrg> then <Virgi is Lrg>
if <PW is Lrg> then <Virgi is Lrg>

(c) Sub-set of rules with two attributes only - petal length (PL) and petal width (PW)
if <PL is Lrg> then <Set is Sm>
if <PW is Lrg> then <Set is Sm)
if <PL is Med> and <PW is Med> then <Set is Sm>
if <PL is Sm> and <PW is Sm> then <Set is Lrg>
if <PL is Sm> and <PW is Sm> then <Versi is Sm>
if <PL is Med> and <PW is Med> then <Versi is Lrg>
if <PL is Lrg> then <Versi is Sm>
if <PW is Lrg> then <Versi is Sm>
if <PL is Med> and <PW is Med> then <Virgi is Sm>
if <PL is Lrg> then <Virgi is Lrg>
if <PW is Lrg> then <Virgi is Lrg>

Denotation: SL- sepal length; SW - sepal width; PL - petal length; PW - petal width; Set - Setosa; Versi - Versicolor; Virgi - Virginica; Sm- small; Med - medium; Lrg - large.

Fig. 6. Extracted rules for the Iris classification task when three fuzzy predicates are used for representing the input and the output variables.

output variables as shown in Fig. 7. A MLP having the structure of a 20-20-15 FuNN was trained with fuzzified training data for 1000 cycles, with the same learning rate and momentum values as above. RMS error of 0.014 was achieved this time. Some of the rules are shown in Fig. 8.

The extracted rules can be used for reasoning with new data which issue is discussed in the next

Iris classification rules: 5 membership functions; thresholds: Th_c =1.2; $Th_{'OR'}$=5
RULES for Setosa
if <SL is Med> and < SW is Sm> and <PL is VRlrg>and <PW is VRlrg> then <Set is VSm>
if <PL is Med> then <Set is VSm>
if <PL is Med> and <PW is Med> then <Set is VSm>
if <PW is Med> then <Set is VSm>
if <PL is Lrg> then <Set is VSm>
if <SL is Med> and < SW is Sm> and <PL is Lrg> and<PW is VRlrg> then <Set is VSm>
if <SL is Med> and < SW is Sm> and <PL is VRlrg> and<PW is VRlrg> then <Set is VSm>
if < SW is Sm> then <Set is VSm>
if <SL is VRlrg> and < SW is Sm> and <PL is Lrg> and<PW is VRlrg> then <Set is VSm>
if <SL is VRlrg> and < SW is Sm> and <PL is VRlrg> and<PW is VRlrg> then <Set is VSm>
if <PL is VSm> then <Set is VRlrg>
if <PL is VSm> and <PW is VSm> then <Set is VRlrg>
RULES for Versicolor
if <SL is VSm> and < SW is VSm> and <PL is VSm>and<PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is VSm> and <PL is VSm>and<PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is VSm> and <PL is Lrg>and<PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is VSm> and <PL is Lrg>and<PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is VSm> and <PL is VRlrg>and<PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is VSm> and <PL is VRlrg>and<PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is Med> and <PL is VSm>and<PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is Med> and <PL is VSm> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is Med> and <PL is Lrg> and <PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is Med> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is Med> and <PL is VRlrg> and <PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is Med> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is Lrg> and <PL is VSm> and <PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is Lrg> and <PL is VSm> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is Lrg> and <PL is Lrg> and <PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is Lrg> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is Lrg> and <PL is VRlrg> and <PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is Lrg> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is VRlrg> and <PL is VSm> and <PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is VRlrg> and <PL is VSm> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is VRlrg> and <PL is Lrg> and <PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is VRlrg> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is VRlrg> and <PL is VRlrg> and <PW is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is VRlrg> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is VSm> and <PL is VSm> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is VSm> and <PL is VSm> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is VSm> and <PL is Lrg> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is VSm> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is VSm> and <PL is VRlrg> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is VSm> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Med> and <PL is VSm> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is Med> and <PL is VSm> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Med> and <PL is Lrg> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is Med> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Med> and <PL is VRlrg> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is Med> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Lrg> and <PL is VSm> and <PW is VSm> then <Versi is VSm>
(continuous to the next page)

Fig. 8. Extracted rules for the Iris classification task when five fuzzy predicates are used for representing the input and the output variables.

(continuation from the previous page)
if <SL is VRlrg> and < SW is Lrg> and <PL is VSm> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Lrg> and <PL is Lrg> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is Lrg> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Lrg> and <PL is VRlrg> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is Lrg> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is VRlrg> and <PL is VSm> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is VRlrg> and <PL is VSm> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is VRlrg> and <PL is Lrg> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is VRlrg> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is VRlrg> and <PL is VRlrg> and <PW is VSm> then <Versi is VSm>
if <SL is VRlrg> and < SW is VRlrg> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <PL is VSm> then <Versi is VSm>
if <SL is VSm> and < SW is Sm> and <PL is Lrg> and <PW is Lrg> then <Versi is VSm>
if <SL is VSm> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is Sm> and <PL is VRlrg> and <PW is Lrg> then <Versi is VSm>
if <SL is VSm> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is Sm> and < SW is Sm> and <PL is Lrg> and <PW is Lrg> then <Versi is VSm>
if <SL is Sm> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is Sm> and < SW is Sm> and <PL is VRlrg> and <PW is Lrg> then <Versi is VSm>
if <SL is Sm> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Sm> and <PL is Lrg> and <PW is Lrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Sm> and <PL is VRlrg> and <PW is Lrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is VSm> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VSm> and < SW is VSm> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is Med> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is Med> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Versi is VSm>
if <SL is VRlrg> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Versi is VSm>
if <PL is Med> then <Versi is VRlrg>
if <PW is Med> then <Versi is VRlrg>
if <PL is Lrg> then <Versi is VSm>
RULES for Virginica
if <PL is Med> then <Virgi is VSm>
if <PL is VSm> then <Virgi is VSm>
if < SW is Sm> and <PL is VSm> then <Virgi is VSm>
if <PL is VSm> and <PW is VSm> then <Virgi is VSm>
if <PW is Med> then <Virgi is VSm>
if <SL is VSm> and < SW is Sm> and <PL is Lrg> and <PW is Lrg> then <Virgi is VRlrg>
if <SL is VSm> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is VSm> and < SW is Sm> and <PL is VRlrg> and <PW is Lrg> then <Virgi is VRlrg>
if <SL is VSm> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is Sm> and < SW is Sm> and <PL is Lrg> and <PW is Lrg> then <Virgi is VRlrg>
if <SL is Sm> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is Sm> and < SW is Sm> and <PL is VRlrg> and <PW is Lrg> then <Virgi is VRlrg>
if <SL is Sm> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is VRlrg> and < SW is Sm> and <PL is Lrg> and <PW is Lrg> then <Virgi is VRlrg>
if <SL is VRlrg> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is VRlrg> and < SW is Sm> and <PL is VRlrg> and <PW is Lrg> then <Virgi is VRlrg>
(continuous to the next page)

Fig. 8. Continued.

(continuation from the previous page)
if <SL is VRlrg> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is Med> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is Med> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is VRlrg> and < SW is Sm> and <PL is Lrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SL is VRlrg> and < SW is Sm> and <PL is VRlrg> and <PW is VRlrg> then <Virgi is VRlrg>
if <SW is Sm> then <Virgi is VRlrg>

Denotation: SL- sepal length; SW - sepal width; PL - petal length; PW - petal width; Set - Setosa; Versi - Versicolor; Virgi - Virginica; VSm- very small; Sm- small; Med- medium; Lrg - large; VLgr - very large.

Fig. 8. Continued.

	Setosa	Versicolor	Virginica
Setosa	10		
Versicolor		10	
Virginica			10

(a)

	Setosa	Versicolor	Virginica
Setosa	10		
Versicolor		10	
Virginica		1 (2)	7 (2)

(b)

	Setosa	Versicolor	Virginica
Setosa	10		
Versicolor		10	
Virginica		2 (1)	7 (1)

(c)

Fig. 9. Classification results for the Iris test data when 3 fuzzy labels are used: (a) fuzzy neural network FuNN; (b) max–min composition fuzzy inference with centroid defuzzification over the full set of extracted simple fuzzy rules; (c) the same method applied on a subset of simple rules (for the last two input variables only). The figures in brackets show the number of ambiguously classified instances.

section. As it is shown there, both FuNN produce correct classification for the test data. When simple rules are used a better classification is obtained for the rules having five membership functions. This can be explained as follows: the loss of information when 'cutting the weights' for the purpose of simple rules extraction, needs to be compensated by a higher granularity in the fuzzy representation.

5. Approximate reasoning with extracted fuzzy rules

Extracting *weighted fuzzy rules* and *simple fuzzy rules* in the REFuNN algorithm allows for different fuzzy inference techniques to be tried on the extracted rules as explained below.

(i) *Using the existing trained FuNN*. The trained FuNN, already used for the rules extraction procedure, can be used for reasoning over new data. When the two FuNN (12-6-9 and 20-20-15) from the previous section were tested with the Iris test data, all the test examples were classified correctly.

(ii) *Inserting weighted fuzzy rules after their extraction from FuNN, in a new FuNN.* Weighted rules can be inserted in a new FuNN architecture and the network then can be used for approximate reasoning or for further training. All the connections which do not appear in any of the weighted rules are set to 0 [11].

(iii) *Using standard fuzzy inference methods with extracted simple fuzzy rules.* Extracted simple rules can be used for inference when different standard fuzzy inference methods are applied [24, 21, 16]. In our experiment the *max–min composition fuzzy inference method* is used with a centroid defuzzification applied [20, 17 16]. The classification results for a full set of extracted rules and the set of the two-input variables simple rules for the 3-membership functions FuNN (see Fig. 6) are shown in Fig. 9. The figures in brackets show the number of ambiguously classified instances (two classes equally 'win' the classification). When the simple fuzzy rules for 5 fuzzy labels (Fig. 8) were used with the same method, *all* the test examples were classified correctly.

Rules extraction and approximate reasoning modules should be tightly connected in a knowledge engineering environment as it was pointed out in the introduction to this paper (see Fig. 1). The next section introduces such an environment.

6. FuzzyCOPE – a Fuzzy COnnectionist Production system Environment

FuzzyCOPE [10, 13] is a knowledge engineering environment which is based on five main modules as shown in Fig. 10 and explained below.

- *Rules extraction module* – it includes methods for rules extraction from fuzzy neural networks.
- *Fuzzy inference module* – it offers different fuzzy inference methods for experimenting with while tuning the fuzzy reasoning over a set simple fuzzy rules. Compositional inference methods as well as decompositional methods for reasoning with fuzzy

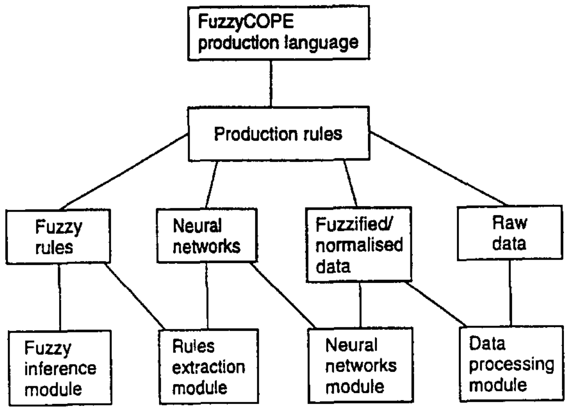

Fig. 10. A block architecture of FuzzyCOPE.

inputs and fuzzy outputs are implemented [24,20, 17]. This module processes a set of fuzzy rules either designed by the users, or by the experts, or automatically extracted from raw or pre-processed data through the rules extraction module.

- *Neural network module* – it consists of several neural network simulators which can be initialised, trained with raw or fuzzified data, and tested either separately, or as a part of production rules written and executed at a higher level. Fuzzy neural networks can be created and used. Fuzzification of raw data can be done in the data processing module.
- *Data processing module* – it does operations over raw data, such as fuzzification, normalization, clusterization. The output of this module can be used as an input to other modules in the environment.
- *FuzzyCOPE production system module* – it is a rule-based production system [2] extended with functions to call fuzzy inference methods, to call rules extraction methods, to call neural networks and to call data processing methods, all of them being available for experimenting with in the corresponding modules of the environment. A user program written in FuzzyCOPE can realize a hybrid multi-modular and multi-paradigm system.

A major characteristic of FuzzyCOPE is that the different AI paradigms, realised as different modules, can be either used as separate tools, or mixed in one production system. For mixing them in one system all of the available functions in the different modules can be called from a production

```
;; A program in FuzzyCOPE for training a fuzzy neural network,
;; for fuzzy rules extraction and approximate reasoning with new data
( defrule initialisation_of_a_FuNN(fuzzy_neural_network)
  (start)
=>
  ( bind $?status (newbp "iris.wgt" 12 6 9) )
  ( assert (FuNNexists) ))
( defrule fuzzify_Iris_data
  (start)
=>
  ( bind $?status (fuzzify "iris.trn" "iris-fz.trn" 3))
  ( assert (DataFuzzified)))
( defrule Training_FuNN
  (FuNNexists)
  (DataFuzzified)
=>
  ( bind $?error (trainbp "iris.wgt" "iris-fz.trn" 1000 0.1 0.3 0.001))
  ( assert (FuNNtrained)))
( defrule Fuzzy_rules-extraction_from_FuNN
  (FuNNtrained)
=>
  ( bind $?status (extract "iris.wgt" "iris-fz.rul" 2 2.0))
;; (defrule input-new-data ;this rule is only commented here
;; this rule reads input data for a new instance and asserts
;; a fact (F $?new_iris) in the working memory

(defrule reasoning_in_FuNN
   (F $?new_iris)
=>
   (bind $?results1 (recallbp "iris.wgt" $?new_iris))
   (printout t $?results1))

;;; separation of the set of fuzzy rules "iris-fz.rul" into three sets of
;;; rules - "iris_se.rul", "iris_ve.rul" and "iris_vi.rul" has to be done
;;; before the following rule is fired

(defrule one_fuzzy_inference_method_for_approximate_reasoning
    (F $?new_iris)
=>
    (bind ?se (fidfuzzy "c:iris_se.rul" $?new_iris ))
    (bind ?ve (fidfuzzy "c:iris_ve.rul" $?new_iris ))
    (bind ?vi (fidfuzzy "c:iris_vi.rul" $?new_iris ))
    (bind $?results2 ?se ?ve ?vi) (printout t $?results2 crlf))
```

Fig. 11. A FuzzyCOPE program for: fuzzification of training data, initialisation of a fuzzy neural network, training the network with fuzzified data, rules extraction, reasoning in the fuzzy neural network and in a standard fuzzy inference engine.

rule in the production system module. Fig. 11 shows an example of a FuzzyCOPE program for fuzzification of training data, initialisation of a FuNN structure, training the network with fuzzified data, rules extraction, approximate reasoning in the trained FuNN and approximate reasoning with a standard fuzzy inference method when new data has been entered.

Such knowledge engineering environments bring all the benefits of the symbolic AI systems, the connectionist systems and the fuzzy systems into one comprehensive hybrid system.

7. Conclusions and directions for further research

Issues of knowledge acquisition and approximate reasoning in hybrid neuro-fuzzy systems are discussed in the paper. Fuzzy neural networks and one particular architecture called FuNN are introduced for realising this approach.

An algorithm for rules extraction from FuNN, called REFuNN, is proposed in the paper. Weighted fuzzy rules as well as simple fuzzy rules can be extracted and used further for approximate reasoning either in FuNN or in a fuzzy inference engine. This is illustrated with the Iris classification problem.

Environments which facilitate different rule extraction methods and different approximate reasoning methods are needed to implement the main idea of this paper. Such an environment FuzzyCOPE is introduced and illustrated in the last section of the paper.

Further research can be done in the following directions:

- developing new rules extraction algorithms from FuNN, e.g. extracting *generalised fuzzy production rules* (see Fig. 2);
- developing new algorithms for training and adaptation of FuNN which allow for '*growing*' and '*shrinking*' of the initial FuNN structure;
- further development of FuzzyCOPE-like integrated knowledge engineering environments by introducing more paradigms in them such as genetic algorithms, chaotic data analysis, etc.

References

[1] T. Furuhashi, T. Hasegawa, S. Horikawa, and Y. Uchikawa, An adaptive fuzzy controller using fuzzy neural networks, in: *Proc. 5th IFSA World Congr.* (1993) 769–772.

[2] J. Giarratano and G. Riley, *Expert Systems. Principles and Programming* (PWS Publ., Boston, 1989).

[3] M.M. Gupta and D.H. Rao, On the principles of fuzzy neural networks, *Fuzzy Sets and Systems* **61** (1994) 1–18.

[4] T. Hashiyama, T. Furuhashi and Y. Uchikawa, A decision making model using a fuzzy neural network, in: *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan, (1992) 1057–1060.

[5] W. Hauptmann and K. Heesche, A neural net topology for bidirectional fuzzy-neuro transformation, in: *Proc. FUZZ-IEEE/IFES*, Yokohama, Japan (IEEE Press, New York, 1995) 1511–1518.

[6] H. Ishibuchi, H. Tanaka and H. Okada, Interpolation of fuzzy if–then rules by neural networks, *Int. J. Approximate Reasoning* **10** (1994) 3–27.

[7] J.S.R. Jang and C.T. Sun, Neuro-fuzzy modelling and control, *Proc. IEEE* (1995) to appear.

[8] N. Kasabov, Learning fuzzy production rules for approximate reasoning, in: S. Gielen and B. Kappen, Eds., Connectionist production systems, *Proc. Internat. Conf. on Artificial Neural Networks ICANN '93*, Amsterdam, 1993 (Springer, Berlin, 1993) 337–342.

[9] N. Kasabov, Connectionist Fuzzy Production Systems, Lecture Notes in Artificial Intelligence, 847 – A. Ralescu, Ed., Fuzzy Logic in Artificial Intelligence (Springer, Berlin 1994) 114–128.

[10] N. Kasabov, Hybrid connectionist fuzzy production systems – towards building comprehensive AI, *Internat. J. Intelligent Automation and Soft Computing*, AutoSoft (1995) to appear.

[11] N. Kasabov, Adaptable Neuro Production Systems, to appear in *Neurocomputing* (Elsevier, Amsterdam, 1995).

[12] N. Kasabov, Hybrid Connectionist Fuzzy Rule-based Systems for Speech Recognition, to appear in: Lecture Notes in Computer Science/Artificial Intelligence (Springer, Berlin 1995)

[13] N. Kasabov, *Neural Networks, Fuzzy Systems and Knowledge Engineering* (MIT Press, Cambridge, 1996) to appear.

[14] N. Kasabov and S. Shishkov, A connectionist production system with partial match and its use for approximate reasoning, *Connection Sci.* **5** (1993) 275–305.

[15] A. Kawamura, N. Watanabe, H. Okada and K. Asakawa, A prototype of neuro-fuzzy cooperation system, in: *Proc. 1st IEEE Conf. on Fuzzy Systems* (1992) 1275–1280.

[16] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence* (Prentice-Hall, Englewood Cliffs, NJ, 1992).

[17] M. Mizumoto and H. Zimmermann, Comparison of fuzzy reasoning methods. *Fuzzy Sets and Systems* (1982) 253–283.

[18] K. Nakamura, T. Fujimaki, R. Horikawa and Y. Ageishi, Fuzzy network production system, in: *Proc. 2nd Internat. Conf. on Fuzzy Logic & Neural Networks*, Iizuka, Japan (1992) 127–130.

[19] S. Tano, T. Oyama, T. Arnould and A. Bastian, Definition and tuning of unit-based fuzzy systems in FINEST, IEEE, 0-7803-1896-X/94 (1994) 436–440.

[20] T. Terano, K. Asai and M. Sugeno, *Fuzzy Systems Theory and Its Applications* (Academic Press, New York, 1992).

[21] T. Terano, Long-term view on fuzzy technology and LIFE projects, *LIFE Tech. News*, **4** (November 1993) 1–10.

[22] R. Yager, Modelling and formulating fuzzy knowledge bases using neural networks, *Neural Networks* **7** (1994) 1273–1283.

[23] T. Yamakawa, H. Kusanagi, E. Uchino and T. Miki, A new effective algorithm for neofuzzy neuron model, in: *Proc. 5th IFSA World Congr.* (1993) 1017–1020.

[24] L. Zadeh, The role of fuzzy logic in the management of uncertainty in expert systems, in: M. Gupta, A. Kandel, W. Bandler and J. Kiszka, Eds., *Approximate Reasoning in Expert Systems* (North-Holland, Amsterdam, 1985).

[25] L. Zadeh, Fuzzy sets, *Inform. and Control* **8** (1965) 338–353.