Published in the Proceedings of Third Starting AI Researchers' Symposium (STAIRS-2006), L. Ponserini, P. Peppas and A. Perini (Eds), IOS Press, 2006, 96-107.

FUNEUS: A neurofuzzy approach based on fuzzy adaline neurons

Constantinos KOUTSOJANNIS & Ioannis HATZILYGEROUDIS
Department of Computer Engineering & Informatics, School of Engineering, University of Patras, Hellas (Greece)

Abstract. Today hybrid computing is a popular framework for solving complex problems. If we have knowledge expressed in rules, we can build an Expert System, and if we have data, or can learn from stimulation (training) then we can use Artificial Neural Networks. In this paper we present the FUzzy NEUrule System (FUNEUS) which is a Neuro Fuzzy approach based on fuzzy Adaline neurons and uses Differential Evolution for optimization of membership functions. According to previous Neuro-fuzzy approaches and a well-defined hybrid system HYMES, FUNEUS is an attempt to the direction for integration of neural and fuzzy components with Differential evolution. Despite the fact that it remains difficult to compare neurofuzzy systems conceptually and evaluate their performance, early experimental results proved a promising performance and the need for further evaluation in other application domains.

Key words. Hybrid systems, Neurofuzzy architecture, Differential evolution, fuzzy adaline

1. Introduction

Hybrid systems mix different methods of knowledge engineering and make them "work together" to achieve a better solution to a problem, compared to using a single method for the same problem [1, 2]. Hybrid connectionist production systems incorporate artificial neural networks (ANNs) into production rules with respect to approximate reasoning and learning [2, 3]. Fuzzy inference modules are incorporated into production rules in a similar way [4, 5]. Today Neuro-Fuzzy (NF) computing is a popular framework for solving complex problems. If we have knowledge expressed in linguistic rules, we can build a Fuzzy Expert System (FIS), and if we have data, or can learn from stimulation (training) then we can use ANNs. For building a FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge base. Similarly for constructing an ANN for an application the user needs to specify the architecture and learning algorithm. An analysis reveals that the drawbacks pertaining to these approaches seem complementary and therefore it is natural to consider building an integrated system combining the concepts. While the learning capability is an advantage from the viewpoint of FIS, the formation of linguistic rule base will be advantage from the viewpoint of ANN [4, 5]. Interestingly this synergy still becomes a target yet to be satisfied. The essence of the successful synergy relies on the retention of the well-defined identity of the two contributing technologies. In the most systems

[FuzzyCOPE, ANFIS, NEFCLASS, FUNN etc] one of the two technologies becomes predominant resulting a commonly visible accuracy-interpretability trade-off [6, 7, 8]. Generally, because of the approximation abilities are easier to be quantified and eventually to be scientifically or technologically realized, the usual result is a tendency toward for far attention being placed on the neural side of the most NF systems with the approximation capabilities being highly “glorified” even supported with Evolutionary Programming (EP) techniques and the interpretation abilities being quietly reduced (FUZZNET, FNES, FALCON, ANFIS, NEFCON, FINEST, FuNN, NEFCLASS, etc) [4, 3, 5]. As a guideline, for NF systems to be highly intelligent some of the major requirements are: fast learning (memory based - efficient storage and retrieval capacities), on-line adaptability (accommodating new features like inputs, outputs, nodes, connections etc), achievement a low global error rate and computationally inexpensive [3, 4, 7]. The data acquisition and pre-processing training data are also quite important for the success for all NF systems [8, 9]. The underlying conjecture of the previous is that the future NF systems should be constructed on a simple possessing unit [4]: Fuzzy logic Neuron (FN) that include fuzzy data and fuzzy logic operations in its’ unit, the inputs and/or the weights are also expressed in terms of membership functions and whose transparency and learning abilities are accentuated to highest possible level as already proposed in the literature [3, 10].
In this paper we introduce *fuzzy neurules* a kind of rules that incorporate fuzzy Adaline units for training and adaptivity purposes. However, pre-eminence is still given to the symbolic component. Thus, the constructed knowledge base retains the modularity of fuzzy rules, since it consists of autonomous units (*fuzzy neurules*), and their naturalness, since they look much like symbolic rules. In previous papers, we have also described a similar method included in HYMES for generating *neurules* directly from empirical (training) data [2]. In this paper we present and evaluate a new architecture, called FUNEUS, in order to work with fuzzy data. Preliminary experimental results provide promising performance.
The structure of the paper is as follows. Section 2 presents the hybrid connectionist system and the corresponding architecture. Section 3 presents the basic ideas and architecture of fuzzy neurules. In Section 4, the system architecture and parameter adjustment of FUNEUS is described. In Section 5 the hybrid inference mechanism is presented. Section 6 contains experimental results for our model validation. Finally Section 7 discusses related work.

2. Low level Hybrid Systems and Fuzzy Neurules

A hybrid system is a mixture of methods and tools used in one system which are loosely [1, 4] or tightly coupled [1, 4] from the functional point of view and hierarchical [3], flat [3], sequential [3, 4] or parallel [3, 4] from the architectural point of view. Additionally there are systems that are blended at a low structural and functional level that they are not separable from functional and structural point of view [3, 4, 6]. The most of Low level Hybrid Connectionist Systems models use variants of McCulloch and Pitt’s neurons to build a network.

2.1 Fuzzy Neurules

2.1.1 Symbolic Rules in Neural Networks (NN): Connectionist Expert Systems

Building a connectionist rule base is possible not only by training a neural network with a set of data examples but by inserting existing rules into a neural network structure [5, 7]. This approach brings advantages of connectionism, that are: learning, generalization, modularity, robustness, massive parallelism etc., to the elegant methods for symbolic processing, logical inferences and global-driven reasoning. Both paradigms can be blended at a low, neuronal level and structural knowledge can built up in a neuron and in a NN realized as a connectionist rule-based system [3. 4].

2.1.1.1 Representing symbolic knowledge as NN

Representation of symbolic knowledge in the form of production rules, in a NN structure requires appropriate structuring of the NN and special methods. A NN may have fixed connections, that is that NN cannot learn and improve its knowledge or adaptable connections, that is that NN can learn in addition to its inserted structured knowledge. A great advantage to using NNs for implementing rule-based systems is the capacity that they provide for approximate reasoning. It is true only if the neurons used in the network allow grading. If, they are binary, only exact reasoning is possible [8, 9].

2.1.1.2 Neurons and NNs that represent Simple Symbolic Rules

A Boolean propositional rule of the form of:

$$IF\ x_1\ and\ x_2\ and\ \dots\ x_n\ THEN\ y$$

Where x_i (i=1,2,…, n) and y are Boolean propositions, can be represented in a binary input - binary output neuron which has a simple summation input function and an activation threshold function f. Similarly, the Boolean propositional rule:

$$IF\ x_1\ or\ x_2\ or\ \dots\ x_n\ THEN\ y$$

will be realized in a similar binary neuron but with different connection weights and thresholds [7]. The neurons cannot learn. These two simple neurons can be used for building NNs that represent a whole set of rules, but which are not adaptable. Symbolic rules that contain different types of uncertainties can also be realized in a connectionist structure. These include rules where uncertainty is expressed by probabilities and in this case is set in such a way that it calculates conditional probabilities as well as rules with confidence factors that is:

$$IF\ x_1\ is\ A_1\ and\ x_2\ is\ A_2\ and\ \dots\ x_n\ is\ An\ THEN\ B\ (Cf)$$

Can be realized either by [5]:

1. Inserting the rule into the connections of n-input, one output neuron or
2. Applying a training procedure to e neuron with training examples, whose input and output values represent certainties for existing facts to match the condition elements and confidence for the inferred conclusion.

Realizing that more than one rule in a single neuron of the Perceptron-type may not be appropriate, minded the restrictions of Perceptron neurons pointed out by a number of authors [5, 7], *neurules* presented in [2] have been developed as a kind of a connectionist production system that has incorporated Adaline-type neurons to represent sets of simple symbolic rules. In the next sections we describe *fuzzy neurules* that are a kind of fuzzy rules in a fuzzy expert system incorporated Fuzzy-Adaline neurons to represent sets of simple fuzzy rules.

2.2 Integrating Fuzzy Neurules with Fuzzy ADALINE neurons

2.2.1 The Fuzzy Logic neuron

A fuzzy neuron has the following feature, which distinguish it from the ordinary types of neurons [4, 5]:

- The inputs of the neuron $x_1,\ x_2,\ \dots\ x_n$ represent fuzzy labels of the fuzzy input variables.

- The weight are replaced by the membership functions μ_i of the fuzzy labels x_i *(i=1,2,…, n)*
- Excitatory connections are represented by MIN operation and inhibitory connections by fuzzy logic complements followed by MIN operation.
- A threshold level is not assigned.

In fuzzy neuron there is no learning. The membership functions attached to the synaptic connections do not change. *Neo-fuzzy neuron* [10] that is a further development of the fuzzy neuron, with the new features of:

a. the incorporated additional weights which are subject to change during training and

b. it works with standard triangular membership functions and thus only two membership functions are activated simultaneously by an input, and consequently

c. have proved faster in training and better in accuracy even than a three-layered feedforward NN with backpropagation algorithm.

The underlying conjecture of the previous is that the new NF systems should be constructed in a simple possessing unit: Fuzzy logic Neuron (FN) that include fuzzy data and fuzzy logic operations in its' unit, the inputs and/or the weights are also expressed in terms of membership functions and whose transparency and learning abilities are accentuated to highest possible level [4]. Types of fuzzy neurons have been successfully applied to prediction and classification problems [5, 9].

2.2.2 The Fuzzy ADALINE neuron

The Wirdow-Hoff's ADALINE can be thought of as the smallest, linear building block of the artificial neural networks. This element has been extensively used in science, statistics (in the linear regression analysis), engineering (the adaptive signal processing, control systems), and so on. Recently Fuzzy ADALINE neurons were developed introducing the following modifications to ADALINEs [6]:

1. The signum-type non-linearity has been replaced by a two-level clipper.
2. The position of the non-linear function block has been shifted inside the loop, unlike the case of Widrow's model, where it was on the forward path outside the loop.
3. A provision for stretching the linear portion of the non-linearity has been incorporated in the model.

In the proposed model, with the input and output constrained in the range [-1, +1], the unity slope in the nonlinearity of the neuron will fail to yield the desired output whenever the target signal is higher than the average of the inputs. In order to circumvent this situation, an adaptive algorithm for adjustment of the slope of the linear portion in the non-linear clipper function have been developed.

In order *fuzzy rules* to work in fuzzy environment we incorporated the a "fuzzy" neuron to produce sets of rules called *fuzzy neurules*. Each *fuzzy neurule* is now considered as a fuzzy adaline unit that has replaced a number of fuzzy rules in the rule base of a fuzzy system. However, pre-eminence is still given to the fuzzy component. Thus, the constructed knowledge base retains the modularity of production rules, since it consists of autonomous units (*fuzzy neurules*), and their naturalness, since fuzzy *neurules* look much like fuzzy rules.

3. Fuzzy neural Networks

A Fuzzy Neural Network (FNN) is a connectionist model for fuzzy rules implementation and inference. There is a great variety of architectures and functionalities of FNN. The FNNs developed so far differ mainly in the following parameters:

- *Type of fuzzy rules* implemented that affects the connectionist structure used.
- *Type of inference* method implemented that affects the selection of different neural network parameters and neuronal functions, such as summation, activation and output function. It also affects the way the connection weights are initialized before training, and interpreted after training
- *Mode of operation:* that can have one of the three modes a) *Fixed mode* with fixed membership functions-fixed set of rules, that is a fixed set of rules inserted in a network, which performs the inference, but does not change its weights, resulting to learning and adaptation [5], b) *Learning mode* with the network is structurally defined to capture knowledge in a format of fuzzy rules, and after random initialization and training with a set of data the set of fuzzy rules are finally extracted from the structured network and c) *Adaptation mode* where the network is structurally set according to a set of fuzzy rules and heuristics and then after training with a set of data updated rules are extracted with: 1. *fixed membership functions – adaptable rules* and 2. *adaptable membership functions-adaptable rules*.

FNNs have two major aspects: a. *the structural* that refers to the type of neurons that are used i.e. the multilayer perceptrons (MLPs) and the radial-basis functions. FuNN is a characteristic example of adaptable FNNs that uses MLP or backpropagation training algorithm with adaptable membership functions of the fuzzy predicates and adaptable fuzzy rules [5].

4. FUNEUS

4.1 FUNEUS: A Hybrid Fuzzy Connectionist Production System

4.1.1 System Structural Components

4.1.1.1 The Fuzzy Neural Network

NF systems could be broadly classified in two types: a) weakly coupled systems and b) tightly coupled systems [6].

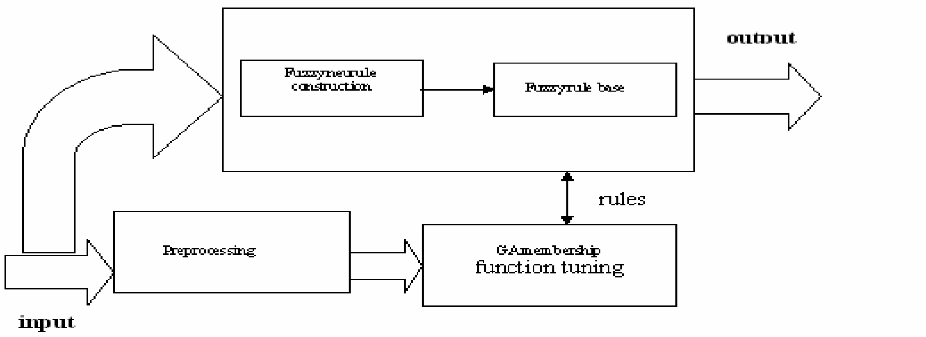

Figure 1: A general diagram of FUNEUS

A weakly coupled NF system employs both a neural net unit and a fuzzy system unit in cascade. In a tightly coupled system neurons the basic elements of a NN are constructed amalgamating the composite characteristics of both a neuronal element and fuzzy logic [4, 11].

4.1.1.2 The GA component

Additionally a Fuzzy-Genetic algorithm synergism is used for membership functions optimization, that are intuitively chosen in a fuzzy system [6]. Given that the optimization of fuzzy membership functions may involve many changes to many different functions, and that a change to one function may effect others, the large possible solution space for this problem is a natural candidate for a GA based approach despite many NF approaches that use a gradient descent-learning algorithm to fine-tune the parameters of the fuzzy systems. GA module for adapting the membership function parameters acts as a stand-alone system that already have the if-then rules. GA optimizes the antecedent and consequent membership functions. Differential Evolution (DE) algorithm here used is a very simple population based, stochastic function minimizer which is very powerful at the same time turned out to be the best genetic type of algorithm for solving the real-valued test functions [14].

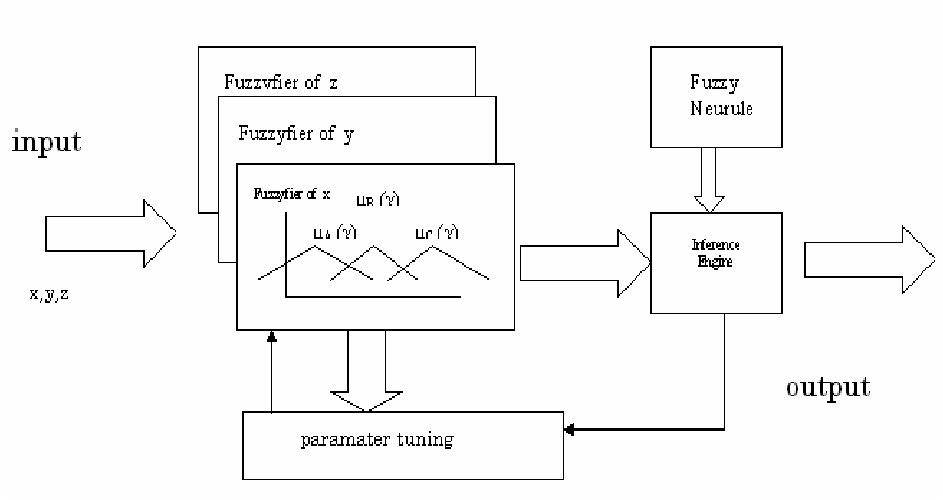

Figure 2. Membership functions adjustment by GA

The crucial idea behind DE is a scheme for generating trial parameter vectors. For this architecture evolution of membership functions proceeds at a faster time scale in an environment usually decided by the problem, the architecture and the inference system. Here we use evolution for triangular membership functions tuning using a mutation function as:

$$X_{new} - X_{old} = F(y+z)$$

where x, y, z and F random numbers (with fuzzy or crisp values) and a mutation function for curves with representation (x (a), y (b)).

The GA used in the system is, in essence, the same as DE genetic algorithm, with the important exception that the chromosomes are represented as strings of floating point numbers, rather than strings of bits. According to this NF-GA synergism let x, y and z be three fuzzy variables and μ_A (x), μ_B (x), μ_C (x) are the three fuzzy membership functions of the fuzzy variable x with respect to fuzzy sets A, B and C respectively. Similarly we have the membership functions μ_A (y), μ_B (y), μ_C (y) and μ_A (z), μ_B (z), μ_C (z) for variables y and z respectively. These functions have been chosen intuitively. Now for optimization of the membership curves that are isosceles triangles we denote each side with (x (a) ,y (b)) for each variable. The chromosomes in the present context

thus has 18 fields, two for each membership curve from the input side of the system and the 1 more field from the desired output side of the system. The crossover and mutation operations in the present context are realized in convenient way
in correspondence to the specified mutation function, that is not presented here in details because it is usually a comparison between the system responses F(x,y,z) with the desired system responses F_d(x,y,z) specified from the application (Fig. 8).

4.1.2 Fuzzy Neurule Base Construction

We use the simple and straightforward method [2] and the proposed method by Wang and Mendel [16] for generating fuzzy rules from numerical input-output training data. The task here is to generate a set of fuzzy rules from the desired input-output pairs and then use these fuzzy rules to determine the complete structure of the rule base. The algorithm for constructing a hybrid rule base from training data is outlined below:

1. *Determine the input and output variables (fuzzy) and use dependency information to construct an initial fuzzy rule for each intermediate and output variable.*
2. *Determine the for each initial fuzzy rule from the training data pairs, train each initial fuzzy rule using its training input-output set and produce the corresponding fuzzy neurule(s).*
3. *Put the produced fuzzy neurules into the fuzzy neurule base.*

In the sequel, we elaborate on each of the first two steps of the algorithm.

4.1.3 Constructing the initial fuzzy neurules

To construct *initial neurules*, first we need to know or determine the input, intermediate and output variables. Then, we need *dependency information*. Dependency information indicates which intermediate variables (concepts) and output variables (concepts) depend on. If dependency information is missing, then output variables depend only on input variables, as indicated by the *training data*.

In constructing an initial fuzzy neurule, all the conditions including the input, intermediate and the output variables that contribute in drawing a conclusion (which includes an intermediate or an output variable) constitute the inputs of the initial *fuzzy neurule* and the conclusion its output. So, a *fuzzy neurule* has as many conditions as the possible input, intermediate and output variable-value pairs. Also, one has to produce as many initial *fuzzy neurules* as the different intermediate and output variable-value pairs specified. Each fuzzy neurule is a *fuzzy Adaline* neuron with inputs the fuzzified values, weights the membership functions and additional weights with initial values of '1'.

4.1.4 Training the initial neurules

From the initial training data, we extract as many (sub)sets as the initial fuzzy neurules. Each such set, called a *training set*, contains *training examples* in the form [x_1 x_2 … x_n *d*], where x_i, *i*= 1, …,*n* are their *component values*, which correspond to the *n* inputs of the fuzzy neurule, and *d* is the *desired output*. Each training set is used to train the corresponding initial neurule and calculate its additional weights that are already set as '1'. So, step 2 of the algorithm for each initial *fuzzy neurule* is analyzed as follows:

2.1 From the initial training data, produce as many initial training sets (x_1 x_2 … x_n *d*) as the number of the initial *fuzzy neurules* .

2.2 We assign the (x_1 x_2 … x_n d) to the region that has maximum degree resulting one value with one membership function for each input parameter

2.3 For each set of desired input-output we obtain do the following:

2.3.1 Obtain one fuzzy neurule i.e. *If x_1 is low and x_2 high …. then d is normal*

2.3.2 Assign an additional weight for each *fuzzy neurule*. The rule weight is defined as $CF_i = \mu_A(\chi_1)\mu_B(\chi_2)...\mu_C(d)$. This step is further performed to delete redundant rules, and therefore obtain a concise *fuzzy neurule base*.

2.3.3 Produce the corresponding fuzzy neurule that is like *If x_1 is A and x_2 B …. then d is C (CF)*

If two or more generated fuzzy neurules have the same conditions and consequents, then the rule that has maximum degree in the desired output is used. In this way, assigning the additional weigth to each rule, the fuzzy rule base can be adapted or updated by the relative weighting strategy: *the more task related the rule becomes, the more weight degree the rule gains*. As a result, not only is the conflict problem resolved, but also the number of rules is reduced significantly [2].

Suppose for example that we are given the following set of desired input -(x_1,x_2) output (y) data pairs (x_1,x_2,y): (0.6, 0.2; 0.2), (0.4, 0.3; 0.4). In our system, input variable *fever used* has a degree of 0.8 in *low*, a degree of 0.2 in *low.* Similarly, input variable *itching* has degree of 0.6 in *low* and of 0.3 in *medium*. Secondly, assign $x_1^{\ i}$, $x_2^{\ i}$, and $y^{\ 1}$ to a region that has maximum degree. Finally, obtain one rule from one pair of desired input-output data, for example,

$$(x_1^{\ 1}, x_2^{1}, y^{1}) \Rightarrow [x_1^{1}\ (0.8 \text{ in low}),\ x_2^{\ 1}\ (0.2 \text{ in low}), y^{\ 1}\ (0.6 \text{ in normal})],$$

• ***R*1:** *if x_1 is low and x_2 is medium, then y is normal;*

$$(x_1^{\ 2}, x_2^{\ 2}, y^{\ 2}), \Rightarrow [x_1(0.8 \text{ in low}), x_2\ (0.6 \text{ in medium}), y^{\ 2}\ (0.8 \text{ normal})],$$

• ***R*2:** *if x1 is low and x2 is high, then y is medium;*

Assign a degree to each rule. To resolve a possible conflict problem, i.e. rules having the same antecedent but a different consequent, and to reduce the number of rules, we assign a degree to each rule generated from data pairs and accept only the rule from a conflict group that has *a maximum degree*. In other words, this step is performed to delete redundant rules, and therefore obtain a concise fuzzy neurule base. The following product strategy is used to assign a degree to each rule. The degree of the rule de-noted by

$$R_i : \text{if } x_1 \text{ is } A \text{ and } x_2 \text{ is } B, \text{ then } y \text{ is } C(w_i),$$

The rule additional weight is defined as

$$wi = \mu_A(x_l)\mu_B(x_2)\mu_c(y)$$

For example of our example

R1 has a degree of

W1 = $\mu_{lowf}(x1)\mu_{\ medium}\ (x2)\mu_{\ normal}\ (y) = 0.8 \text{ x } 0.2 \text{ x } 0.6 = 0.096,$

and R2 has a degree of

W2 = $\mu_{half}(x1)\mu_{\ high}(x2)\mu_{\ normal}\ (y) = 0.8 \text{ x } 0.6 \text{ x } 0.8 = 0.384$

Note, that if two or more generated fuzzy rules have the same preconditions and consequents, then the rule that has maximum degree is used. In this way, assigning the degree to each rule, the fuzzy rule base can be adapted or updated by the relative weighting strategy: the more task related the rule becomes, the more weight degree the rule gains. As a result, not only is the conflict problem resolved, but also the number of rules is reduced significantly. After the structure-learning phase *(if-then* rules), the whole network structure is established, and the network enters the second learning phase to optimally adjust the parameters of the membership functions using the GA algorithm to minimise the error function.

5. Inference through FUNEUS

A functional block diagram of the FUNEUS model consists of two phases of learning processes: a) The first phase is the structure-learning *(if-then* rules) phase using the knowledge acquisition modul*e* producing the *fuzzy neurule base. b)* The second phase is the parameter-learning phase for tuning membership functions to achieve a desired level of performance with the use of a Genetic Algorithm to tune membership functions to fine-tune the parameters of the fuzzy membership functions.

In the connectionist structure, the input and output nodes represent the input states and output decision signals, respectively, and in the hidden layers, there are nodes functioning as quantification of membership functions (MFs) and *if-then* rules. As soon as the initial input data set is given and put in the Working Memory (WM), the resulting output *fuzzy neurules* are considered for evaluation. One of them is selected for evaluation. Selection is based on textual order. A rule succeeds if the output of the corresponding fuzzy adaline unit is computed to be '1', after evaluation of its conditions.

A condition evaluates to 'true', if it matches a fact in the WM, where there is a fact with the same variable, predicate and value. A condition evaluates to 'unknown', if there is a fact with the same variable, predicate and 'unknown' as its value. A condition cannot be evaluated if there is no fact in the WM with the same variable. In this case, either a question is made to the user to provide data for the variable, in case of an input variable, or an intermediate fuzzy neurule in Fuzzy Neurule Base (FNRB) with a conclusion containing that variable is examined, in case of an intermediate variable. A condition with an input variable evaluates to 'false', if there is a fact in the WM with the same variable, predicate and different value. A condition with an intermediate variable evaluates to 'false' if additionally to the latter there is no unevaluated intermediate fuzzy neurule in the FNRB that has a conclusion with the same variable. Inference stops either when one or more output fuzzy neurules are fired (success) or there is no further action (failure). In the training phase input membership functions and desired output values of the training data-set are used in a text-form to fine tune the parameters of the fuzzy sets with the offline use of the GA component.

6. Model Validation Using Testing Data Sets

6.1 Coronary Heart Disease Development (crisp and fuzzy data)

The experimental data set was taken from Takumi Ichimura and Katsumi Yoshida [11] that prepared a medical database named Coronary Heart Disease DataBase (CHD_DB), which makes it possible to assess the effectiveness of classification methods in medical data. The CHD_DB is based on actual measurements of the Framingham Heart Study - one of the most famous prospective studies of cardiovascular disease. It includes more than 10,000 records related to the development of coronary heart disease (CHD). We have proved it enough valid by statistical analyses.

We used FUNEUS to develop a diagnostic system that outputs whether each record is a non-CHD case or a CHD case.

Data in coronary heart diseases database are divided into two classes: non-oronary heart disease cases (non-CHD) and coronary heart disease cases (CHD). Each patient's disorder is diagnosed according to the results of eight test items. The eight items tested are Cholesterol (TC), Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), Drinking (ALCOHOL) that are used as inputs with fuzzy values, Left Ventricular Hypertrophy (LVH), Origin (ORIGIN), Education (EDUCATE), Smoking (TABACCO), and that used as inputs with non-fuzzy value. The fuzzy inference system was created using the FUNEUS. We used triangular membership functions and each input variable were assigned three MFs. Ten fuzzy rules were created using the methodology mentioned in Section 3 (Rule format: IF (…) THEN CHD) :

Rule 1 : *(SBP high)*
…
Rule 3 : *(TC high)*
Rule 4 : *(SBP medium) AND (DBP > high) AND (LVH = 0)AND (EDUCATE < 2) AND (TABACCO > 0) AND (TABACCO < 3)*
…
Rule 6 : *(TC medium) AND (DBP high) AND (ORIGIN = 1) AND (TABACCO > 0) AND (ALCOHOL low)*
Rule 7 : *(TC medium) AND (SBP high) AND (DBP medium) AND (EDUCATE < 1) AND (TABACCO < 2)*
…
Rule 10 : *(TC high) AND (SBP high) AND (DBP > high) AND (LVH = 0) AND (TABACCO > 0)AND (TABACCO < 3) AND (ALCOHOL medium)*

We also explored the fine-tuning of membership functions using DE algorithm. We started with a population size 10, tournament selection strategy, mutation rate 0.01 and implemented a one point crossover operator. After a trial and error approach by increasing the population size and the number of iterations (generations), we finalized the population size and number of iterations as 50. Figure 3 demonstrates the effect of parameter tuning of membership functions (before and after evolutionary learning) for the input variable ALCOHOL *use*d.

6.2 Evaluation results

We used Train_Z, set which is consisted of 400 CHD cases, and 3600 non-CHD cases. The judgment accuracy for 4000 training data was as follows: results were correct for 310 cases of 400 CHD cases, and were false for 2683 of 3600 non-CHD cases. Therefore, the recognition rate to the set of training data was 75.0%, comparable with machine learning method used in [15].

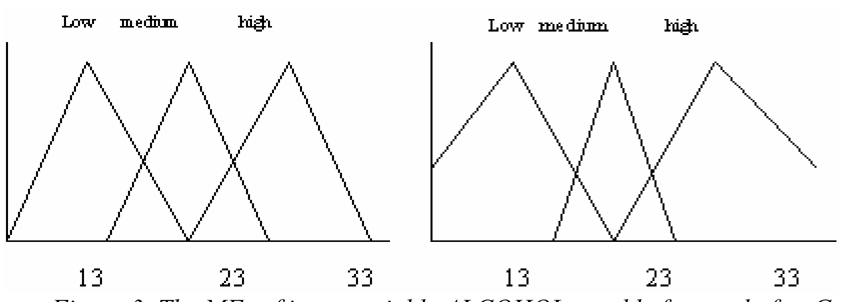

Figure 3. The MFs of input variable ALCOHOL used before and after GA learning

7. Discussion and Related Work

A number of NF systems have used for various real life problems. Among others, three neuro-fuzzy systems that use hybrid learning for rules generation and parameter tuning like FUNEUS presented in this paper have already been described [8]. The first one is called NEFCON (NEuro-Fuzzy CONtroller) and used for control applications. The next one is NEFCLASS (NEuro-Fuzzy CLASSifier) and used for classification problems and pattern recognition [5]. The third one is NEFPROX (Neuro-Fuzzy function for approximation) and used for function approximation. The three systems are based on generic fuzzy perceptron which is a kind of FNN. Each of these systems can learn a rule-base and then tune the parameters for the membership functions. Perceptron is used in order to provide a framework for learning algorithms to be interpreted as a system of linguistic rules and to be able to use prior knowledge in the form of fuzzy IF-THEN rules. Additionally fuzzy weights are associated with linguistic terms. The fuzzy perseptron is composed from an input layer, in a hidden layer and an output layer. Connections between them are weighted with fuzzy sets instead real numbers. In NEFCON inputs are state variables and the only one output neuron outputs control action applied to a technical system. In NEFCLASS the rule-base approximates a function for a classification problem and maps an input pattern to proper class using no membership functions in the rule's consequents [8]. There are also examples of NF systems that do not employ any algorithms for rule generating, so the rule base must be known in advance. They can only adjust parameters of the antecedent and consequent fuzzy sets. The most popular system of this kind is ANFIS (Adaptive-Network-based Fuzzy Inference System) that is of the first hybrid NF systems for function approximation. The architecture is five-layer feed-forward implementing Tagaki-Sugeno type of rules [11]. According to Abraham [3] Sugeno type systems are high performers but often require complicated learning procedures and computational expensive. An optional design of a NF system can only be achieved by the adaptive evolution of membership functions, rule base and learning rules. For each architecture evolution of membership functions proceeds at a faster time scale in an environment decided by the problem, the architecture and the inference system. Thus, global search of fuzzy rules and membership functions provide the fastest possible time scale [3]. The main problem here is that the resulting rules are usually not accepted form the expert knowledge because they are mechanically derived and not related with real world. So it is desirable to tune a set of predefined rules rather to produce a set from data [15]. In this paper we present FUNEUS which is a NF system based on fuzzy Adaline neurons and uses Genetic Algorithms for optimization of membership functions. Taking account the previous approaches and that it remains difficult to compare NF systems conceptually and evaluate their performance FUNEUS is an attempt to the direction of integrating the best components of such approaches after our experience with a well-defined hybrid model so-called HYMES. Experimental results proved acceptable performance of the NF need to be evaluated in more domains as Medical Diagnosis [17] and Intelligent Educational Systems [18]. Further work will provide additionally experimental results compared with other hybrid approaches.

References

[1] Bonissone PP, Chen YT, Goebel T, and Khedkar PS.: Hybrid Soft Computing Systems: Industrial and Commercial Applications, *Proceedings of the IEE*E, 87(9), 1641-1667, (1999).

[2] Hatzilygeroudis I., Prentzas, J.: HYMES: A HYbrid Modular Expert System with Efficient Inference and Explanation. Proceedings of the 8th Panhellenic Conference on Informatics, Nicosia, Cyprus, Vol.1 (2001) 422-431

[3] Abraham A. and Nath B.: Hybrid Intelligent Systems Design-A Review of a Decade of Research,

[4] Pedrycz W.: Heterogeneous Fuzzy Logic Networks: Fundamentals and Development Studies IEEE Transactions on Neural Networks, vol. 15, 6:1466- 81 (2004)

[5] Kasabov N.: Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, MIT Press (1996)

[6] Konar A.: Computational Intelligence, Springer-Verlag (2005).

[7] Gallant S.I., Neural Network Learning and Expert Systems, MIT Press (1993).

[8] Shavlik J.: Combining Symbolic and Neural Learning, Machine Learning, 14, 321-331 (1994)

[9] Rutkowska D.: Neuro-Fuzzy Architecture and Hybrid Learning, Physica-Verlag Press (2002)

[10] Yamakawa T. Pattern recognition hardware system employing fuzzy neuron, In Proceedings of the International Conf of Fuzzy Logic and Neural Networks , Lizuka Japan, July 1990, 943-948.

[11] Pal SK, Mitra S.: Neuro-Fuzzy Pattern Recognition, John Wiley & Sons, NY (1999).

[12] Jang J.: ANFIS: Adaptive-Network-based-Fuzzy-Inference System, IEEE Trans on Systems, Man and Cybernetics, 23:665-685.

[13] Suka M , Ichimura T, and Yoshida K.: Development of Coronary Heart Disease Database, KES 2004, LNAI 3214:1081–1088, 2004.

[14] Storn, R., "System Design by Constraint Adaptation and Differential Evolution", IEEE Trans. on Evolutionary Computation, 1999, Vol. 3, No. 1, pp. 22 - 34.

[15] Hara A., Ichimura T.: Extraction of Rules from Coronary Heart Disease Database Using Automatically Defined Groups, M.Gh. Negoita et al. (Eds.): KES 2004, LNAI 3214:1089–1096, (2004).

[16] Wang L. X. & Mendel J. M., Generating Fuzzy Rules by Learning from Examples, IEEETranscation on System, Man and Cybernetics, Vol. 22, Issue 6, pp. 1414-1427, 1992.

[17] Georgios D. Dounias and Derek A. Linkens (Eds.), (2001), Adaptive Systems and Hybrid Computational Intelligence in Medicine, Special Session Proceedings of the EUNITE 2001 Symposium, Tenerife, Spain, December 13-14, 2001, A Publication of the University of the Aegean, ISBN 960-7475-19-4

[18] I. Hatzilygeroudis, C. Giannoulis and C. Koutsojannis, Combining Expert Systems and Adaptive Hypermedia Technologies in a Web Based Educational System, ICALT 2005 (Kaohsiung,Taiwan, July 5-8,2005).