World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 1, No. 4, 124-131, 2011

Classification of Uncertain data using Fuzzy Neural Networks

G.V.SURESH
Associate Professor, CSE department

O.Srinivasa Reddy
Associate Professor, CSE department

Shabbeer Shaik
Professor, CSE department

Dr. B. Munibhadrayya
Principal

Abstract— Data mining has emerged to be a very important research area that helps organizations make good use of the tremendous amount of data they have. In this data uncertainty is common in real-world applications due to various causes, including imprecise measurement, network latency, out-dated sources and sampling errors. These kinds of uncertainty have to be handled cautiously, or else the mining results could be unreliable or even wrong. We propose that when data mining is performed on uncertain data, uncertainty has to be considered in order to obtain high quality data mining results. In this paper we explores a fuzzy neural network for classification of uncertain data and suggests how uncertainty can be incorporated in data mining and how to handle data uncertainty in data mining by using a Fuzzy neural network.

Keywords- Neural networks, Fuzzy logic, data classification, uncertain data, fuzzy neural network

I. INTRODUCTION

Data is often associated with uncertainty because of measurement inaccuracy, sampling discrepancy, outdated data sources, or other errors. This is especially true for applications that require interaction with the physical world, such as location-based services [1] and sensor monitoring [3]. For example, in the scenario of moving objects (such as vehicles or people), it is impossible for the database to track the exact locations of all objects at all-time instants. Therefore, the location of each object is associated with uncertainty between updates [4].These various sources of uncertainty have to be considered in order to produce accurate query and mining results. We note that with uncertainty, data values are no longer atomic. To apply traditional data mining techniques, uncertain data has to be summarized into atomic values. Taking moving-object applications as an example again, the location of an object can be summarized either by its last recorded location or by an expected location. Unfortunately, discrepancy in the summarized recorded value and the actual values could seriously affect the quality of the mining results. In recent years, there is significant research interest in data uncertainty management. Data uncertainty can be categorized into two types, namely existential uncertainty and value uncertainty. In the first type it is uncertain whether the object or data tuple exists or not. For example, a tuple in a relational database could be associated with a probability value that indicates the confidence of its presence. In value uncertainty, a data item is modelled as a closed region which bounds its possible values, together with a probability density function of its value. This model can be used to quantify the imprecision of location and sensor data in a constantly-evolving environment.

A. Uncertain data mining

There has been a growing interest in uncertain data mining[1], including clustering[2], [3], [4], [5], classification[6], [7], [8], outlier detection [9], frequent pattern mining [10], [11], streams mining[12] and skyline analysis[13] on uncertain data, etc. An important branch of mining uncertain data is to build classification models on uncertain data. While [6], [7] study the classification of uncertain data using the support vector model, [8] performs classification using decision trees. This paper unprecedentedly explores yet another classification model, a fuzzy Neural Network, and extends them to handle uncertain data. For uncertain classification problems, however, we should learn the class conditional density from uncertain data objects represented by probability distributions.

II. RESEARCH BACKGROUND

A. Neural Networks for Data Mining

Neural networks are suitable in data-rich environments and are typically used for extracting embedded knowledge in the form of rules, quantitative evaluation of these rules, clustering, self-organization, classification and regression, feature evaluation and dimensionality reduction. The following are the major stages in solving a DM problem. The entire DM process is iterative, and the result in each step can be feed back to any of the previous steps for improvement. The loop will continue until a satisfactory result has been obtained. A lot of work in current DM has been done in developing integrated systems to support all 7 stages not only stages 5 and 6 that are typical for NN and machine learning. There are many nice features of

NN, which make them attractive for DM. These features include learning and generalization ability, adaptivity, content addressability, fault tolerance, self-organization, robustness, and simplicity of basic computations. NN are useful especially when there is no a priori knowledge about the analyzed data. They offer a powerful and distributed computing architecture, with significant learning abilities and they are able to represent highly nonlinear and multivariable relationships.

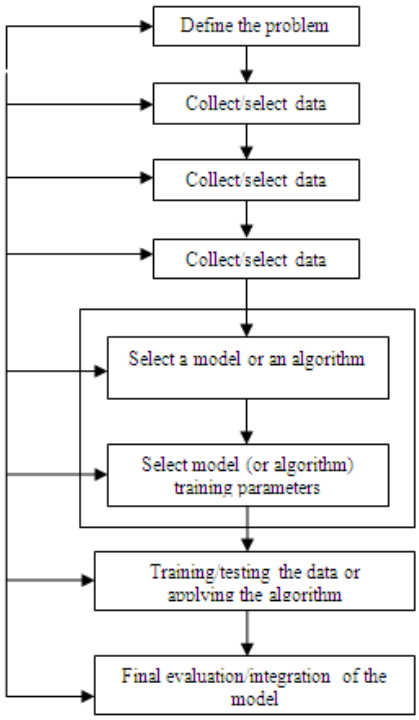


Figure 1. Data modeling process

B. *Fuzzy Neural Network*

A fuzzy neural network (FNN) is a hybrid intelligent system that possesses the capabilities of adjusting adaptively and intelligent information processing. A lot of new concepts, such as innovative architecture. In practice FNN's have found useful in many application fields, for instance, system modelling, system reliability analysis, pattern recognition and knowledge engineering and so on. Based on fuzziness involved in FNN's developed since the late of 1980s, one may broadly classify all FNN models as three main types

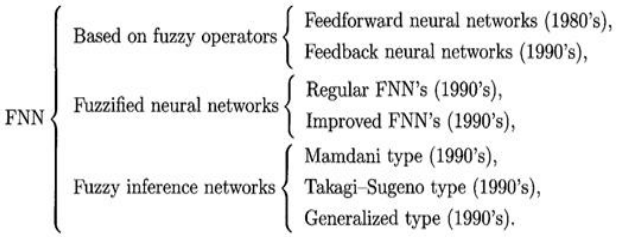


Figure 2. Classification of FNN's

C. *Fuzzy-neural systems*

There are several methods for implementing the neuro-fuzzy modelling technique. An early merging approach was to replace the input-output signals or the weights in neural networks by membership values of fuzzy sets, along with the application of fuzzy neurons. In general, neuro-fuzzy hybridization is done in two ways

- a neural network equipped with the capability of handling fuzzy information processing, termed a fuzzy-neural network (FNN)
- a fuzzy system augmented by neural networks to enhance some of its characteristics, like flexibility, speed, and adaptability, termed a neural-fuzzy system (NFS).

Neural networks with fuzzy neurons are also termed FNN, because they are also capable of processing fuzzy information. A neural-fuzzy system (NFS), on the other hand, is designed to realize the process of fuzzy reasoning, where the connection weights of the network correspond to the parameters of fuzzy reasoning.

The first model consists of a fuzzy inference block, followed by a neural network block, consisting of a multilayer feedforward neural network, the input of which is fed by the inference block (Fuller, 1995). The neural network used can be adapted and adequately trained with training samples to yield the desired outputs.

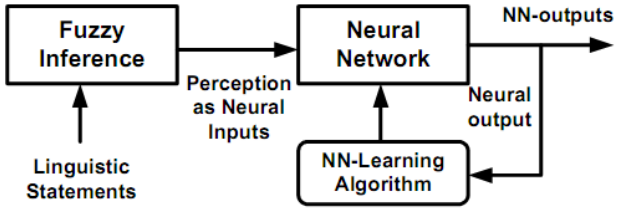


Figure 3. First model Fuzzy-neural system

In the second model the neural network block drives the fuzzy inference system to generate the corresponding decisions. Hence, the first model takes linguistic inputs and generates the numerical outputs, whereas the second model takes numerical inputs and generates the linguistic outputs

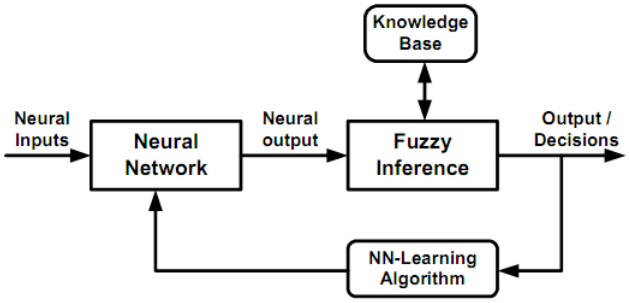


Figure 4. Second model Fuzzy-neural system

Alternatively, the second approach is to use fuzzy membership functions to pre-process or post-process signals with neural networks

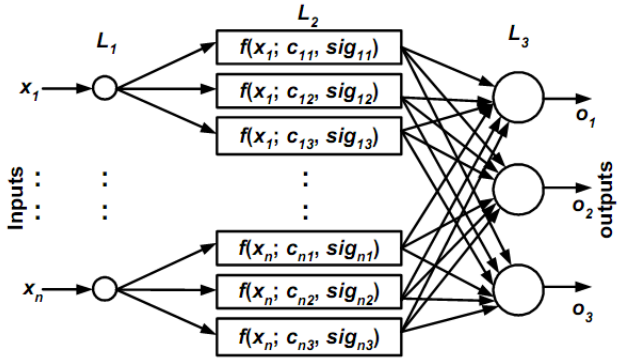


Figure 5. Fuzzy-neural model with tuneable membership function

D. *Fuzzy Neuron*

A Fuzzy Neuron is defined by fuzzifying a crisp neuron, directly. Its structure is shown as Figure 4.1. *d* fuzzy inputs $\tilde{X}_1,....\tilde{X}_d$, and connection weights $\tilde{W}_1,....,\tilde{W}_d$ related are fuzzy numbers in $F_0(\square)$

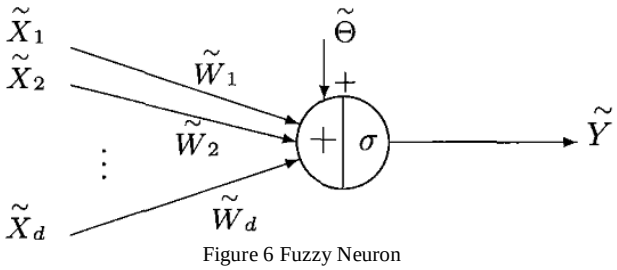


Figure 6 Fuzzy Neuron

The I/O relationship of the fuzzy neuron is determined as follows:

$$\tilde{Y} \square F(\tilde{X}_1,....\tilde{X}_d) = \sigma(\sum_{i=1}^{d} \tilde{X}_i \cdot \tilde{W}_i + \tilde{\Theta}) = \sigma(\langle \tilde{\mathbf{X}}, \tilde{\mathbf{W}} \rangle + \tilde{\Theta}) \quad (1)$$

where $\tilde{\Theta}$ means a fuzzy bias (or a fuzzy threshold), $\tilde{\mathrm{X}}=(\tilde{X}_1,....\tilde{X}_d)$ and $\tilde{\mathrm{W}}=(\tilde{W}_1,....,\tilde{W}_d) \in F_0(\square)^d$ is fuzzy vectors, $\sigma : \square \to \square$ is a transfer function. If $\sigma(x) = x$, the neuron related is called linear. In real, usually a is selected as the following forms (where a > 0 is a constant)

(i) Piecewise linear function: $\sigma(x) = \frac{|1+\alpha x|+|1-\alpha x|}{2}$

(ii) Hard judgment function: $\sigma(x) = \mathrm{sign}(x)$

(iii) S type function: $\sigma(x) = \frac{1}{1+\exp(-\alpha x)} (\alpha > 0)$

(iv) Radial basis function: $\sigma(x) = \exp(-\alpha x)$

Related to (1) an obvious fact holds, that is, the I/O relationship $F(\square)$ is monotone, i.e. for given $\tilde{X}_1,....\tilde{X}_d, \tilde{Z}_1,...., \tilde{Z}_d \in F_0(\square)$,

$$\tilde{X}_i \subset \tilde{Z}_i (i = 1,...., d) \Rightarrow F(\tilde{X}_1,....\tilde{X}_d) \subset F(\tilde{Z}_1,...., \tilde{Z}_d) \quad (2)$$

Fuzzy neurons can process all kinds of fuzzy information, efficiently. It includes crisp neurons as special cases

III. RELATED WORKS

In this paper we present an approach to neuro-fuzzy uncertain data analysis. The goal is to derive fuzzy rules from a set of data that can be separated in different crisp classes. The fuzziness involved is due to an imperfect or incomplete measurement of features thus rendering it difficult to assign a pattern to the correct category. The hybrid approach is to interpret the rule base of the fuzzy system in terms of a neural network. The fuzzy sets can be seen as weights and the input and output variables and the rules can be interpreted as neurons. This way a fuzzy system can be interpreted as a special neural network. The learning algorithm works by modifying the structure and/or the parameters that means the inclusion or deletion of neurons or adaption of the weights

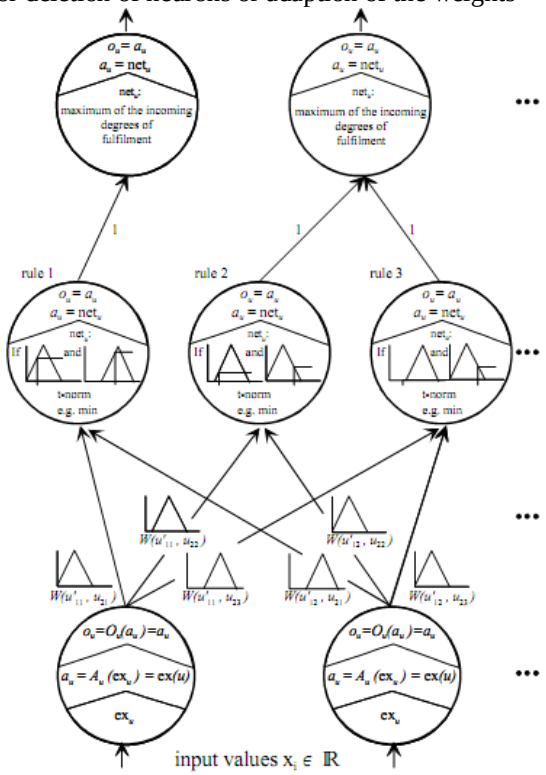


Figure 7 A neural network view of a neuro-fuzzy classifier

A. *Fuzzy Neural Network Design*

The task of fuzzy neural network subsystem is training the weighted fuzzy production rules by modifying the weight so that the rule can classify data with higher accuracy. Through the literature review we know that the fuzzy neural network generally has the input membership layer, fuzzy rule layer and output layer. The activation function for the input membership layer is the trapezoid membership function; the activation for the fuzzy rule layer is the fuzzy rule, and the activate function for the output layer is the logic operation function. In this paper we suggest a generic model for neural fuzzy systems based on a 3 - layer fuzzy perceptron by using it to derive neural fuzzy systems for special domains it would be possible to evaluate these different neuro-fuzzy approaches by means of the same underlying model

B. Fuzzy Perceptron

A fuzzy perceptron has the architecture of a usual multi-layer perceptron, but the weights are modelled as fuzzy sets and the activation, output, and propagation functions are changed accordingly. The intention of this model is to be interpretable in form of linguistic rules and to be able to use prior rule based knowledge, so the learning has not to start from scratch. In we suggested a generic model for fuzzy neural networks based on a 3-layer fuzzy perceptron .By using it to derive neural fuzzy systems for special domains, it would be possible to evaluate these different neuro-fuzzy approaches by mean s of the same underlying model.

Fuzzy perceptron i s a 3-layerfeedforward neural network (*U, W, NET, A, O, ex*) with the following specifications:

(i) $U = \bigcup U_i (i \in \{1,2,3\})$ *is a non-empty set of units. For all i,* $j \in \{1,2,3\}, U_i \neq \theta$ *and* $U_i \cap U_j = \theta$ *with* $i \neq j$ *holds.* U_1 *is called input layer* U_2 *rule layer (hidden layer), and* U_3 *output layer*

(ii) The structure of the network (connections) is defined as $W : U \times U \to F(\square)$, *such that there are only connections W (u, v) with* $u \in U_i, v \in U_{i+1} (i \in \{1,2\})$ ($F(\square)$ *is the set of all fuzzy subsets of* $\square$)

(iii) A defines an activation function A_u *for each* $u \in U$ *to calculate the activation* a_u

a) for input and rule units $u \in U_1 \bigcup U_2$: $A_u : \square \to \square$,

$a_u = A_u(net_u) = net_u$,

b) for output units $u \in U_3$: $A_u : F(\square) \to F(\square)$,

$a_u = A_u(net_u) = net_u$

(iv) O defines for each $u \in U$ *an output function* O_u *to calculate the output* o_u

a) for input and rule units $u \in U_1 \bigcup U_2$: $O_u : \square \to \square$,

$o_u = O_u(a_u) = a_u$,

b) for output units $u \in U_3 : O_u : F(\square) \to F(\square)$,

$o_u = O_u(net_u) = DEFUZZ_u(net_u)$ *where* $DEFUZZ_u$ *is a suitable defuzzification function*

(v) NET defines for each unit $u \in U$ a *propagation function* NET_u *to calculate the net input* net_u

a) for input units $u \in U_1 : NET_u : \square \to \square$, $net_u = ex_u$,

b) for input units $u \in U_2 : NET_u : (\square \times F(\square))^{U_1} \to [0,1]$,

$\top_{u' \in U_1} (W(u',u)(o_{u'}))$, *Where* $\top$ *is a t -norm,*

c) for output units $u \in U_3 : NET_u : ([0,1] \times F(\square))^{U_2}$

$\to F(\square)$, $net_u : \square \to [0,1]$,

$net_u(x) = \perp_{u' \in U_2} (\top(o_{u'}, W(u',u)(x)))$ *Where* $\perp$ *is a t -conorm. If the fuzzy sets* $W(u',u), u' \in U_2, u \in U_3$ *are monotonic on their support, and* $W^{-1}(u',u)(\tau) = x \in \square$ *such that* $W^{-1}(u',u)(x) = \tau$ *holds, then the propagation function* net_u *of an output unit* $u \in U_3$ *can alternatively be defined as*

$$net_u(x) = \begin{cases} 1 \text{ if } x = \dfrac{\sum_{u' \in U_2} o_{u'} \square m(o_{u'})}{\sum_{u' \in U_2} o_{u'}} \\ 0 \quad \text{otherwise} \end{cases}$$

with $m(o_{u'}) = W^{-1}(u',u)(o_{u'})$ *to calculate the output* o_u *in this case* $o_u = x$,*with* $net_u(x) = 1$ *is used*

(vi) $ex : U_1 \to \square$ *defines for each input unit* $u \in U_1$ its *external input* $ex(u) = ex_u$ *.for all other units ex is not defined*

A fuzzy perceptron is like a usual perceptron used for function approximation. The advantage is to interpret its structure in the form of linguistic rules, because the fuzzy weights can be associated with linguistic terms. The network can also be created partly or in the whole out of fuzzy if-then rules.

C. Fuzzy neural network design

The Fuzzy neural network model is derived from the generic fuzzy perceptron. The learning algorithm is a special version of fuzzy error backpropagation and uses a reinforcement procedure that allows on-line learning.

A Fuzzy neural network system is a special 3-layer fuzzy perceptron with the following specifications:

(i)The input units are denoted as $\xi_1 \ldots, \xi_n$ *the hidden rule units are denoted as* $R_1 \ldots, R_k$ *and the single output unit is denoted as* η

(ii) Each connect ion between units ξ_i *and* R_r *is labelled with a linguistic term* $A_{j_r}^{(i)}$ ($j_r \in (\{1 \ldots\ldots, p_i\})$.

(iii) Each connection between units R_r *and the output unit* η *is labelled with a linguistic term* B_{j_r} ($j_r \in (\{1 \ldots\ldots, q\})$.

(iv) Connect ions coming from the same input unit ξ_i and *having identical labels, bear the same fuzzy weight at all times. These connect ions are called linked connections. An analogous condition holds for the connections leading to the output unit* η

(v) Let $L_{\xi,R}$ denote the label of the connection between the input unit ξ_i and the rule unit R. For all rule units R, R'

$$(\forall \xi\ L_{\xi,R} = L_{\xi,R'}) \Rightarrow R = R'\ \textit{holds.}$$

This definition makes it possible to interpret a Fuzzy neural network system in terms of a fuzzy controller each hidden unit represents a fuzzy if-then rule. Condition (iv) specifies that there have to be shared weights. If this feature is missing, it would be possible for fuzzy weights representing identical linguistic terms to evolve differently during the learning process. .Condit ion (v) determines that there are no rules with identical antecedents.

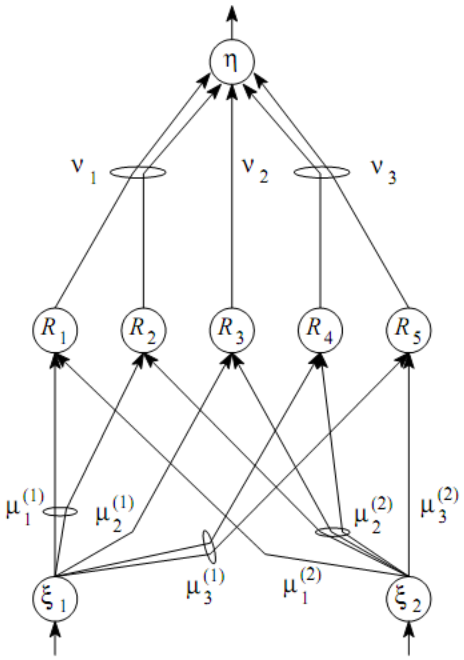


Figure 8 Architecture of Fuzzy Neural Network with two input variables

D. Learning Algorithm

The definition of a learning algorithm for the fuzzy perceptron is not as straightforward as for its neural counterpart because of the usually non differentiable t-norms and t-conorms used as activation functions for the units. By using fuzzy sets as weights it is possible to derive the following simple heuristic as learning

Consider a 3-layer fuzzy perceptron with n input and m output units. Let L be a learning task consisting of sample patterns $p = (i^{(p)}, t^{(p)})$ where $i^{(p)} \in \square^n$ and $t^{(p)} \in \square^m$ denote the input and output vectors of pattern $p \in L$. Let $u \in U_3$ and $t_u^{(p)}$ denote the desired output of the output unit u given the input vector $i^{(p)}$ and let $o_u^{(p)}$ be the actual output value. Also let $range_u$ be the difference between the maximal and minimal output values for unit u. The fuzzy error $E_u^{(p)}$ of u given p is defined as

$$E_u^{(p)} = 1\text{-}exp\left(\beta\left(\frac{t_u^{(p)} - o_u^{(p)}}{range_u}\right)^2\right)$$

where $\beta \in \square$ is a scaling factor.

The scaling factor β is used to adjust the sensitivity of the fuzzy error tolerating more or less large differences between the target and the output values. This β is useful if e.g. a fuzzy classification is sufficient, and there is no need to obtain exact values to classify a given input pattern

Consider a 3-layerfuzzy perceptron and a learning task L.The generic fuzzy error backpropagation algorithm is defined as follows

(i) Choose any $p \in L$ and propagate the input vector $i^{(p)}$

(ii) Determine

$$\delta_u^{(p)} = \begin{cases} \operatorname{sgn}(t_u^{(p)} - o_u^{(p)}).E_u^{(p)} & \text{for } u \in U_3 \\ \sum_{u \in U_3} a_u^{(p)}.\ \delta_u^{(p)} & \text{for } u \in U_3 \end{cases}$$

(iii)Determine

$$\Delta_p W(u.v) = f(\delta_u^{(p)}, a_u^{(p)}, net_u^{(p)}), (u \in U_i, v \in U_j, i, j \in M, j = i+1)$$

Repeat these steps for all $p \in L$ until the overall error

$$E = \sum_{p \in L} \sum_{u \in U_3} E_u^{(p)}$$

is suffiently small after an epoch is complete.

(iv) Determine the parameter changes for the fuzzy weights $\nu_{j,k}$ with learning rates $\sigma_1, \sigma_c, \sigma_r \in \square$

$$\Delta_p l_{j,k} = \sigma_l \cdot \delta_{u_k} \cdot (c_{j,k} - l_{j,k}),$$

$$\Delta_c c_{j,k} = \sigma_c \cdot \delta_{u_k} \cdot (r_{j,k} - l_{j,k}),$$

$$\Delta_p r_{j,k} = \sigma_r \cdot \delta_{u_k} \cdot (r_{j,k} - c_{j,k}),$$

and for all $\mu_{i,j}$ with learning rates $\eta_1, \eta_c, \eta_r \in \square$

$$\Delta_p l_{i,j} = -\eta_l \cdot \delta_{u_j} \cdot (c_{j,k} - l_{i,j}),$$

$$\Delta_c c_{i,j} = \eta_c \cdot \delta_{u_j} \cdot (a_{u_i} - c_{i,j}),$$

$$\Delta_p r_{i,j} = \eta_r \cdot \delta_{u_k} \cdot (r_{i,j} - c_{j,k})$$

The algorithm stops, when E is sufficiently small after an epoch has been completed, otherwise it has to be repeated for another epoch.

The change in the fuzzy weight W (u, v) defined in step (iii) of the above defined algorithm depends on the δ -signal of the unit *v*, and it may also be depend on the activation and net input. This dependency has to be specified according to the actual fuzzy perceptron that is in use .Usually the fuzzy sets will be defined by parameterized membership functions, so that the changes have to be specified in terms of these parameters.

IV. EXPERIMENTS

Uncertain values are common in many practical settings. It is not always possible to observe all features of a pattern. This can be due to high costs, faulty sensors, errors in recording, etc. If a feature is sometimes measured and sometimes not, we can use the cases for which it has been measured to learn to predict its values when it is not. Another approach to learning in the presence of unobserved variables is the EM algorithm. The EM algorithm searches for a maximum likelihood hypothesis by repeatedly re-estimating the expected values of the unobserved variables given the current hypothesis, then recalculating the maximum likelihood hypothesis using these expected values. If a feature is uncertain, we do not make any assumptions about its value but assume that any value may be possible. Based on this assumption we do not want to restrict the application of a fuzzy rule to a pattern with uncertain features. This means a uncertain value will not influence the computation of the degree of fulfilment of a rule. This can be done by assigning 1.0 as the degree of membership to the uncertain feature, i.e. a uncertain value has a degree of membership of 1.0 with any fuzzy set. A pattern where all features are uncertain would then fulfill any rule of the fuzzy rule base with a degree of 1.0, i.e. any class would be possible for such a pattern. We denote a pattern with missing values by $p = (x, ?)$ .we compute the degree of fulfilment μ_r of some rule R_r by:

$$\mu_r(x, ?) = \min_{x_i}\{\mu_r^{(i)}(x_i), 1\} = \min_{x_i}\{\mu_r^{(i)}(x_i)\}$$

Rule learning consists of three steps:

1. Determine all possible antecedents;

2.Create an initial rule base by finding an appropriate consequent for each antecedent;

3.Select a final rule base from the initial rule base by computing the performance of each rule.

If we encounter an uncertain value, any fuzzy set can be included in the antecedent for the corresponding variable. Therefore we create all combinations of fuzzy sets that are possible for the current training

A. Results (Example 1)

We use three fuzzy sets to partition each variable. If we obtain the input pattern $(x, ?)$ as shown in Figure 8 we can assign the fuzzy set large to the first feature, because it yields the largest degree of membership for x_0.As the value for *y* is uncertain, any fuzzy set is possible for *y*. Therefore we create the antecedents (*x is large and y is small*), (*x is large and y is medium*), and (*x is large and y is large*). In step (2) of the rule-learning algorithm appropriate consequents will be determined for these antecedents, depending on all training patterns. In step (3) the rules with the highest performance will be selected. After a rule base was created, the membership functions are trained by classifier. If a uncertain value is encountered, then for the corresponding fuzzy set simply no training signal will be generated from this pattern. As a simple example to demonstrate the capabilities of handling uncertain values we used the Iris data set. The Iris data is a well-known benchmark for classification approaches, although it represents a very simple classification problem. The data set consists of 150 patterns with 4 features. There are 3 classes (*iris setosa, iris virginica and iris versicolour*) that each contain 50 patterns. The first and second class are linearly separable to each other, and the second and third class slightly overlap. The results are given in Table 1. For the first experiment we randomly deleted one feature in 30 randomly selected cases. For the second experiment we randomly deleted 85 values from 70 randomly selected cases in the data set, i.e. now there are also cases where more than one feature is uncertain

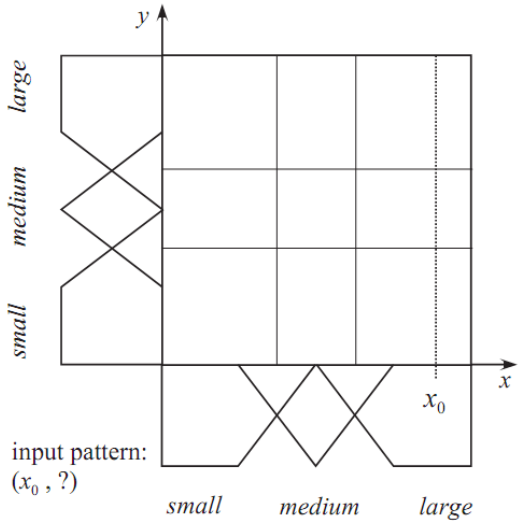


Figure 9 Rule learning with missing values: three rules are created by the pattern *(x0,?)*, because for *y* three fuzzy sets are possible

TABLE I. RESULTS FOR THE IRIS DATA WITH DIFFERENT NUMBERS OF UNCERTAIN VALUES

Missing Values	*Rules*	*Error Training*	*Error Test*
0	9	3	2
30	6	4	2
85	7	4	5

For learning, the data set was randomly split in two stratified samples each containing 75 patterns. One sample was used for training and the other for validation. The rule-learning procedure automatically determined the number of rules and continued to train the membership functions until no further improvement on the validation set was observed (stop training method).As we can see in Table 1 the number of classification errors slightly increases with the number of uncertain values.

B. Results (Example 2)

The Wisconsin Brest Cancer (WBC) data set contains 683 cases. In the following some information and statistics about the WBC dataset are described

TABLE II. STATISTICS ABOUT THE WBC DATA

Number of input features	***Table column subhead***
Number of input features	9
Names of input features	clumb_thickness uniformity_of _cell_size uniformity_of_cell_shape marginal_adhesion single_epithelial_cell_size bare_nuclei bland_chromatin normal_nucleoli mitoses
Minimum value	1.00
Maximum value	10.00
Defined minimum value	0.00
Defined maximum value	11.00
Number of classes:	2
Names of classes	malign benign
Number of cases:	683

The classifier is created with the 'automatically determine the rule base' option in order to find a rule base that covers the whole data set. After training the classifier the following the rule base ends up with 5 rules as follows:

- if uniformity_of _cell_size is small and uniformity_of _cell _shape is small and bare_nuclei is small and then benign
- if uniformity_of_cell_size is large and uniformity_of_cell_shape is small and bare_nuclei is large and then malign
- if uniformity_of_cell_size is small and uniformity_of_cell_shape is small and bare_nuclei is large and then malign
- if uniformity_of_cell_size is large and uniformity_of_cell_shape is large and bare_nuclei is small and then malign
- if uniformity_of_cell_size is large and uniformity_of_cell_shape is large and bare_nuclei is large and then malign

The classification result is:

- 32 misclassifications
- error = 53.190468

TABLE III. PERFORMANCE ON TRAINING DATA

	Predicted Class							
	0		***1***		***n.c***		***sum***	
0	224	32.80%	15	2.20%	0	0.00%	239	34.99%
1	17	2.49%	427	62.52%	0	0.00%	444	65.01%
sum	241	35.29%	442	64.71%	0	0.00%	683	100.00%

0: malign Correct: 651((95.31%)
1: benign Misclassified: 32((4.69%)
n.c.: not classified

TABLE IV. RESULT OF THE CROSS VALIDATION

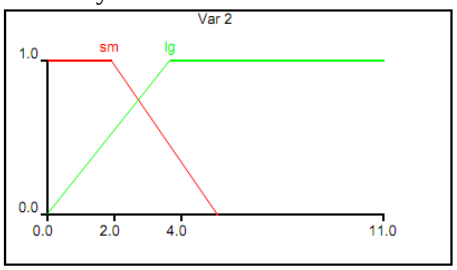
Result of the Cross Validation	
Number of patterns	683
Estimated number of misclassifications per pattern for unseen data:	mean = 0.036637 standarddeviation= 0.015091 n = 10
99% confidence interval for the error estimation:	0.036637 ± 0.012979
Estimated error value	3,7% ± 1,3%

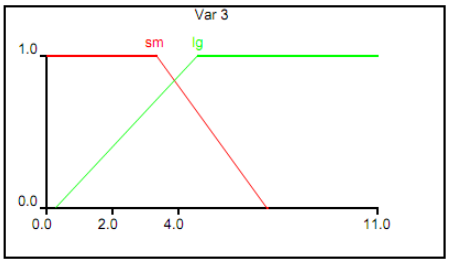
The classifier resulting from the creation process described in the experiments produces classification results which can be interpreted very well because the rule base consists of only two short rules. Under this constraint an estimated error value of 3,7% ± 1,3% is a really good classification result.

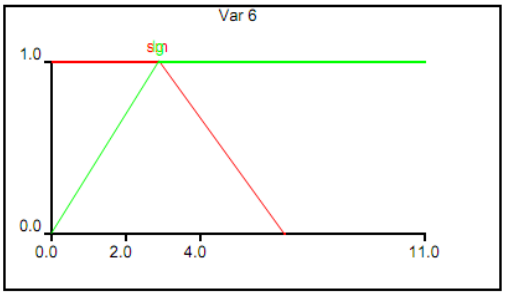
The Rules

- *if uniformity_of_cell_size is small and bare_nuclei is small then benign*
- *if uniformity_of_cell_size is large and uniformity_of_cell_shape is large and bare_nuclei is large then malign*

The Fuzzy Sets







V. CONCLUSION

In this paper, we propose a a neuro-fuzzy strategy for classifying and predicting uncertain data. Fuzzy systems are used to exploit the tolerance for imprecise solutions. Fuzzy systems should be used because they are easy to implement, easy to handle and easy to understand. A learning algorithm to create a fuzzy system from data also should have these

features. The new philosophy in training the rule base automatically was successful, so users who have no experience with training methods can create a compact interpretable classifier. The validation option that offers cross validation as well as single test enables the user to compare the quality of the classification results very easily to other approaches. No further computation with statistic tools is needed. We plan to explore more classification approaches for various uncertainty models and find more efficient training algorithms in the future

REFERENCES

[1] Aggarwal, C.C.: A Survey of Uncertain Data Algorithms and Applications. IEEE Transactions on Knowledge and Data Engineering 21(5) (2009)

[2] Cormode, G., McGregor, A.: Approximation algorithms for clustering uncertain data. In: Principle of Data base System, PODS (2008)

[3] Aggarwal, C.C., Yu, P.: A framework for clustering uncertain data streams. In: IEEE International Conference on Data Engineering, ICDE (2008).

[4] Singh, S., Mayfield, C., Prabhakar, S., Shah, R., Hambrusch, S.: Indexing categorical data with uncertainty. In: IEEE International Conference on Data Engineering (ICDE), pp. 616–625 (2007)

[5] Kriegel, H., Pfeifle, M.: Density-based clustering of uncertain data. In: ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 672–677 (2005)

[6] Aggarwal, C.C.: Managing and Mining Uncertain Data. Springer, Heidelberg (2009)

[7] Nauck D., Klawonn F. and Kruse R.: 'Foundations of Neuro-Fuzzy Systems'. Chich-ester, Wiley (1997). 139, 140, 143, 146

[8] Quinonero Candela, J. & Girard, A. (2002) Prediction at an Uncertain Input for Gaussian Processes and Relevance Vector Machines - Application to Multiple-Step Ahead Time-Series Forecasting. Technical Report, IMM, Danish Technical University.

[9] Buckley J. J. and Hayashi Y.: 'Neural networks for fuzzy systems'. Fuzzy Sets and Systems, Vol. 71, pp. 265–276 (1995)

[10] Halgamuge S. K. and Glesner M.: 'Neural networks in designing fuzzy systems forreal world applications'. Fuzzy Sets and Systems, Vol. 65, pp. 1–12 (1994).

[11] Takagi and I. Hayashi (1 991). NN {Driven Fuzzy Reasoning. Int. J. Approximate Reasoning, 5:1 91-21 2

[12] Shigeo Abe and Ming-ShongLan. A method for fuzzy rules extraction directly from numerical data and its application to pat tern classification. IEEE Trans.Fuzzy Systems, 3 (1):18{28, February 1995.

[13] Shigeo Abe, Ming-ShongLan, and Ruck Thawonmas. Tuning of a fuzzy classifier derived from data. In t. J.Approximate Reasoning, 14 (1):1{24, January 1996

[14] Detlef Nauck and Rudo lf Kruse. NEFCLASS { a neuro{fuzzy approach for the classification of data. In K.M. George, Jan ice H. Carrol, Ed Deaton , Dave Oppenheim , and Jim Hightower , editors , Applied Com puting 1995. Proc. of the 1 995 ACM Symposium on Applied Computing, Nashville, Feb. 26-28, pages 461-465. ACM Press, New York, February 1995

[15] Tschichold Gurman N.: 'Generation and improvement of fuzzy classifiers with incremental learning using fuzzy rulenet'. In 'Applied Computing 1995. Proc. 1995ACM Symposium on Applied Computing, Nashville, Feb. 26–28', K. M. George ,J.H.Carrol, E.Deaton, D.Oppenheim, and J.Hightower, Eds.(New York:ACMPress), pp. 466–470 (1995).

[16] Kruse R., Schwecke E. and Heinsohn J.: 'Uncertainty and Vagueness in Knowledge-Based Systems: Numerical Methods'. Berlin, Springer-Verlag (1991).

[17] Gebhardt J. and Kruse R.: 'Learning possibilistic networks from data'. In 'Proc. ofFifth Int. Workshop on Artificial Intelligence and Statistics' (Fort Lauderdale,Florida), pp. 233–244 (1994).

[18] R. Kruse, J. Gebhardt and F. Klawonn (1994). Foundations of Fuzzy Systems. Wiley, Chichester

[19] D. Nauck and R. Kruse (1998). New Learning Strategies forNEFCLASS. In Proc. of the Seventh International Fuzzy Systems Association WorldCongress, IFSA'97, Vol. IV. Prague

[20] D. Nauck and R. Kruse (1998). NEFCLASS-X: A Soft Computing Tool to build Readable Fuzzy Classifiers. In BT Technology Journal, Vol. 16, No. 3, July 1998.Martelsham Heath Ipswich.

[21] W.H. Wolberg and O.L. Mangasarian. Multi surface method of pattern separation for medical diagnosis applied to breast cytology. Proc. National Academy of Sciences, 87:9193-9196, December 1990

[22] Nauck and R. Kruse (1998). How the Learning of Rule WeightsAffects the Interpretability of Fuzzy Systems. In Proc. IEEE Int. Conf. Fuzzy Systems (FUZZIEEE'98). Anchorage, Alaska.

[23] S.M Sulzberger, N.N. Tchichold-Gürman and S.J. Vestli (1993).FUN: Optimization of Fuzzy Rule Based Systems Using Neural Networks. In Proc. IEEE Int. Conf. on Neural Networks 1993, pp. 1022-1027. San Francisco

[24] Vladimir N. Vapnik. Statistical Learning Theory. John Wiley & Sons, NY, 1998

[25] UCI Machine Learning Repository, *www.ics.**uci**.edu/~mlearn*

AUTHORS PROFILE

G.V.Suresh received his B.Tech and M.Tech from JNT University Hyderabad, India in 2004 and 2008 respectively and currently working toward PhD degree from JNT University. His research interests are in Data Mining Neural Networks, and Networks. He is a life member for Computer Society of India (CSI). He has authored a good number of research papers published in International Journals and Conference Proceedings

O.Srinivasa Reddy has done his M.Tech from Nagarjuna University India in 2009. His research interests are in Image processing, Networks. He is a life member for Computer Society of India (CSI). Currently he is working as Assistant Professor for Computer Science and Engineering Department, at Universal college of Engineering &Tech, Guntur. He has coauthored a good number of research papers published in International Journals and Conference Proceedings

Shabbeer Shaik has done his M.Tech from Nagarjuna University India in 2004. His research interests are in Image processing ,Networks. He is a life member for Computer Society of India (CSI). Currently he is working as Professor for Computer Science and Engineering Deapertment, at Nalanda Institute of Technology(NIT),Guntur.He has coauthored a good number of research papers published in International Journals and Conference Proceedings

Dr. B. Munibhadrayya has done his PhD. His research interests are in Image processing He is a life member for Computer Society of India (CSI). Currently he is working as Principal at Nalanda Institute of Engineering and Technology(NIET) Guntur.