Fuzzy Membership Function Elicitation using Plausible Neural Network

Kuo-chen Li and Dar-jen Chang
Department of Computer Engineering and Computer Science, University of Louisville,
Louisville, KY, 40292, USA

Abstract

The elicitation method of the fuzzy membership function depends on the interpretation of the membership function. This paper applies a recently developed neural network framework, Plausible Neural Network (PNN), to generate fuzzy membership functions automatically with or without class labeling based on the similarity and likelihood measurement. The approach combining supervised and unsupervised learning provides a flexible method to generate fuzzy membership functions with semantic or statistical meanings. The results of simulation experiments show that PNN is capable of generating interesting and adaptive fuzzy membership functions.

Keywords: Fuzzy membership function, PNN, SOFM, Feedforward neural network, Mutual information content.

I. INTRODUCTION

A fuzzy logic system (FLS) design usually involves selecting fuzzy rules and membership functions. The selections of fuzzy rules and membership functions directly affect the performance of the FLS. It is argued that the common practice of fuzzy rules design is more concerned with the FLS results than the selection of membership functions [1][2]. However, some studies have shown the selection of membership functions is more important [3]. The main difficulty for the membership function is to interpret the meaning of selected membership function. For example, what does a given membership function mean? Why do we want to use this membership function rather than that to represent the data? How do we obtain suitable membership functions? In [4], elicitation of membership functions is discussed. It shows that elicitation methods cannot be independent of the interpretation.

In this paper, we propose an elicitation method based on the similarity and likelihood measurement. The proposed method can generate membership functions automatically from a data set. Several different existing methods are capable of generating membership functions from a given data set such as fuzzy c-means (FCM) [5], feedforward neural network (FFNN) [6], and self-organizing feature map (SOFM) [7]. FCM and SOFM generate membership functions during the process of clustering. In machine learning terms, FCM and SOFM use an unsupervised leaning process (i.e. without class labeling) to generate membership functions. In contrasts, FFNN, which uses class labels in the output layer, adopts a supervised (e.g. error back-propagation) learning process to tune the weights in order to generate membership functions. We present a new

elicitation method which uses Plausible Neural Network (PNN) to generate membership functions for data with or without class labeling.

PNN was introduced by Chen in 2003 [8][9]. It is a hybrid model of estimating probabilistic and possibilistic inferences. PNN uses mutual information contents to measure the similarity and likelihood among clusters or classes. With the bi-direction feature and the capability of handling different types of attributes, PNN is capable of training with or without class labeling. If the class attribute participates in the training process, PNN takes the information of the class attribute and performs a supervised learning. On the other hand, if the training is done without class attribute, PNN performs an unsupervised learning. As a result, this approach provides a unify framework for generating membership functions with or without class attribute. Furthermore, it provides a mapping from semantic membership functions (based on the given class attribute) to statistical membership functions (based on the similarity clustering), which lessens the difficulty of the interpretation of membership functions.

This paper is organized as follows: Section II presents an overview of PNN. Section III presents how to generate membership functions using PNN. Section IV shows the experiment results of generating membership functions using PNN. Section V concludes the results.

II. PLAUSIBLE NEURAL NETWORK

A basic PNN architecture consists of two layers (input layer and hidden layer) of cooperative and competitive neurons with complete, bidirectional, and symmetric connections. Fig.1 shows the basic architecture for a PNN model. Each input attribute is encoded into a group of competitive neurons which uses winner-take-all (WTA)[10] algorithm to interpret the value of the attribute. The input WTA ensembles cooperate to determine the values of the WTA ensemble in the hidden layer. Each hidden neuron, generally speaking, represents a pattern or cluster for the given training dataset. In other words, while inputting a data sample to the trained PNN, PNN is able to determine which patterns or clusters the data sample might contain or belong to.

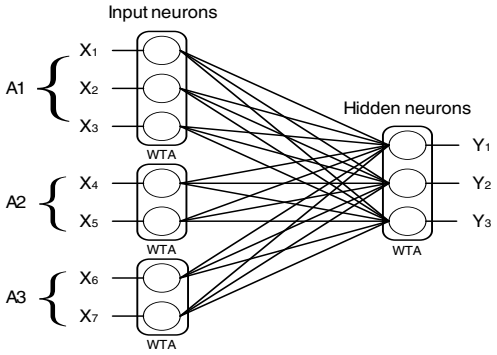

Fig. 1. A general PNN architecture for a dataset contains three attributes (A1, A2, and A3).

To encode the attribute value into a WTA ensemble, first, each WTA ensemble has to be under one condition: for a WTA ensemble $(X_1, X_2, \ldots, X_k)$ where

$$\sum_{i=1}^{k} X_i = 1 \quad \text{and} \qquad 0 \leq \quad X_i \quad \leq 1 \text{ for all } X_i$$

In this manner, for a categorical attribute, we can use one neuron to represent one categorical value intuitively. For example, an attribute, tallness, with three possible values (*short*, *average*, and

tall) can be expressed by a WTA ensemble with three neurons (X1, X2, X3). As to the continuous attribute, a fuzzy set coding fits in perfectly for the WTA ensemble. An initial fuzzy coding is applied to the input WTA ensemble. Each neuron represents a membership function for the fuzzy coding.

The weight definition in PNN is based on the mutual information content which can determine the strength of the relationship between an input neuron and a hidden neuron. Consider two neurons, x and y, the weight between two neurons is given by the mutual information content.

$$\omega_{ij} = \ln\left(\frac{P(X_i, Y_j)}{P(X_i)P(Y_j)}\right) \tag{1}$$

Based on the given weight definition, if X_i, Y_j are independent, $P(X_i,Y_j) = P(X_i)*P(Y_j)$ and we can compute W_{ij}=0. It shows there is no relationship between X_i and Y_j. On the other hand, if the calculated weight is positive, it is called positively associated. It indicates that neuron Y_j is more likely fire if neuron X_i fires. If the weight is negative, it is called negatively associated. That means neuron Y_j is more likely rest if neuron X_i fires.

Through the WTA algorithm and the competition method, the output of hidden neuron (y_j) for the input $\boldsymbol{X}$ can be represented by the following:

$$y_j = S(\sum_i w_{ij} x_i), \forall j, e^{\sum_i w_{ij} X_i} / \sup_j e^{\sum_i w_{ij} x_i} > \alpha \tag{2}$$

$$y_j = 0, otherwise$$

where S(t_j) is the normalization function

$$S(t_j) = e^{\kappa t_j} / \sum_j e^{\kappa t_j} \tag{3}$$

In (2), α is a threshold value to cut the weak signal. In (3), κ is the temperature argument that can amplify the signals. The default setting for κ is usually 1. Since PNN is complete, bidirectional, and symmetric, (2)(3) can be used for both directions between input layer and hidden layer. Forward firing and reverse firing are defined based on the different directions of firing.

The learning method is based on the computation of belief. Each training procedure measures the action potentials for hidden neurons. First, PNN fires hidden neurons randomly for each training sample in order to produce an initial fire table. Based on the initial fire table and input training samples, PNN calculate the weight tables for each weight connection between input neurons and hidden neurons using estimated weight:

$$W_{xy} = \ln\left(\frac{n\sum x_k y_k}{\sum x_k \sum y_k}\right) \tag{4}$$

After a new weight table estimated, PNN fires each training sample using firing method to get a new fire table. The next step is to compare two fire tables. If two fire tables are identical, that means weight table hasn't changed and PNN is stable. Otherwise, based on the new fire table, estimate the new weight table and repeat previous steps until PNN is stable.

III. GENERATING MEMBERSHIP FUNCTION WITH PNN

The PNN implementation of generating membership functions without class attribute is illustrated in Fig. 2, where for convenience the input data represent person height. In our experience with PNN, for membership functions generation, two complimentary WTA neurons for data coding are sufficient to get good results. Each hidden neuron in Fig. 2 represents a membership function. The concept of generating membership functions is similar to that of FCM and SOFM. Membership functions are generated by an unsupervised clustering process. After the PNN training, for an input value, the degree of likelihood of this value belonging to each cluster (hidden neuron) can be estimated through the network computation. The degree of likelihood of belonging to a cluster, in the fuzzy logic terms, matches the degree of the membership function represented by the cluster. In addition, the WTA hidden-neuron ensemble always normalized their represented membership functions. This normality property of a set of membership functions is important for some fuzzy logic system applications [12].

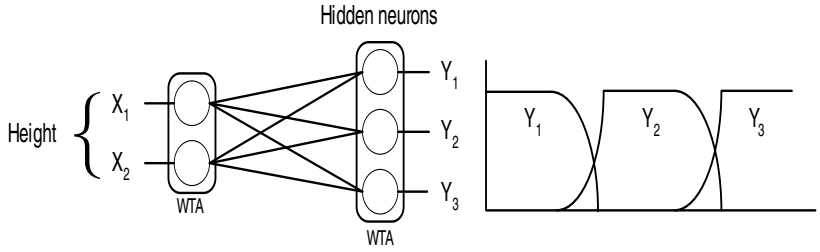

Fig. 2 PNN implementation of generating membership functions without class attribute.

For the PNN implementation of generating membership function with class attribute, the same framework is used except that the class attribute is added to the network input layer for training. Fig. 3 shows such a PNN topology for generating the semantic fuzzy membership functions representing the "tallness" linguist term (e.g. *short, average, or tall*) from the continuous attribute height. Since the class attribute participates in the training process, it is considered as a supervised learning. After the training process, to elicit the membership functions, we can either elicit the membership functions from the hidden layer or from the class attribute (shown in Fig. 3).

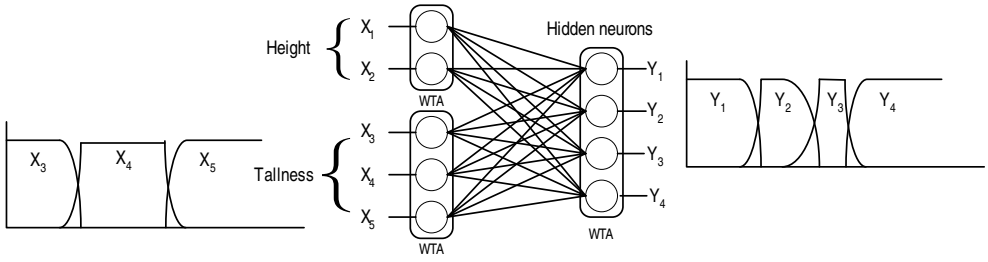

Fig. 3 PNN implementation of generating membership function with class attribute.

To generate the membership functions from the class attribute, after training, a height value along with an unknown class value is fired to the PNN to trigger the hidden neurons. And then, the output of hidden neurons is fired back to the class attribute to get the degree of likelihood for each class value, which corresponds to the membership function value. Note Fig.3 shows that the membership function for the class value, *average*, is the union of the membership functions of cluster 2 and 3.

IV. EXPERIMENTS

Three artificial datasets are created to test the PNN implementation. The first dataset consists of 100 random samples, 50 of which are taken using a normal distribution with mean 30 and variance 10 and the other 50 of which are taken using a normal distribution with mean 90 and variance 10. Fig. 4(a) shows the distribution of the dataset. Using the dataset, Fig. 4(b) and 4(c) show the membership functions generated by two PNNs with two and four hidden neurons, respectively. Next, the same dataset adding two-value class attribute as shown in Fig. 4(d) are used in a PNN with four hidden neurons to generate membership functions. Fig. 4(e) shows the membership functions generated by reverse-firing to the class attribute. Note that only three of four hidden neurons contain membership functions as shown in Fig. 4(f) since the class attribute contribute the information to merge two neighboring clusters.

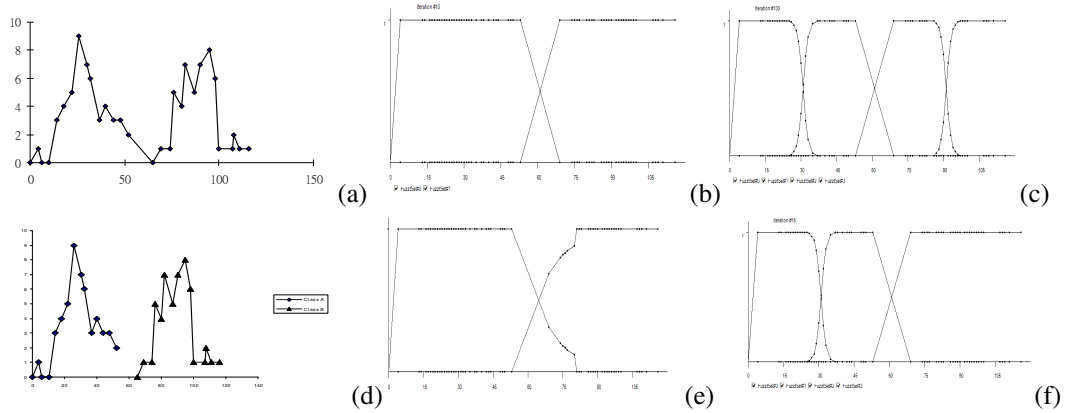

Fig. 4 Generating membership functions using PNN with or without class labeling

The second dataset which contains 150 samples from two distributions of the same configuration as the first dataset, 50 from the first and 100 from the second distribution (Fig. 5(a)), is used in the same PNNs of the previous test to generate membership functions. The results from these two tests are very similar (please compare Figure 5 with figure 4). This suggests that the number of samples for different classes barely affects the results of PNN. The PNN technique is robust relative to other elicitation methods which are affected by the imbalance of data class labeling. Thus the PNN method has this nice property: the frequency of the occurrence of the elements (number of samples for a class) is independent of the form of the membership functions as described in [11].

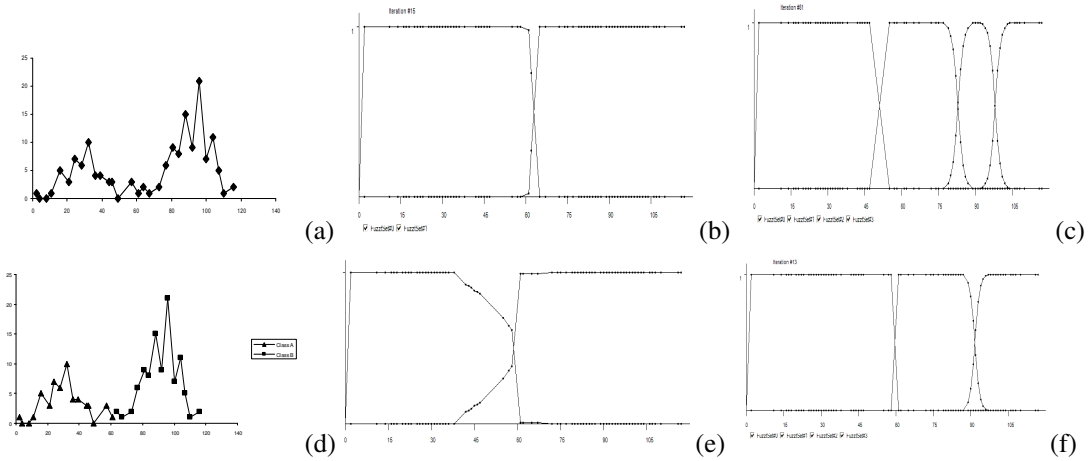

Fig. 5 Generating membership functions from unbalanced dataset using PNN

The third dataset is created with two overlapped normal distributions (mean 30 and 50, respectively). Using this dataset, the same PNNs as created before stabilize in dozens of iterations. Comparing Fig. 6(b) (without class labeling) with (e) (with the class labeling), the membership functions generated by reverse-firing combine the semantic class labeling and the statistical characteristics of the dataset. The fuzziness of the overlapping area in the class membership (see Fig. 6(d)) is reflected in generated membership functions (see Fig.6(e)) using the PNN technique.

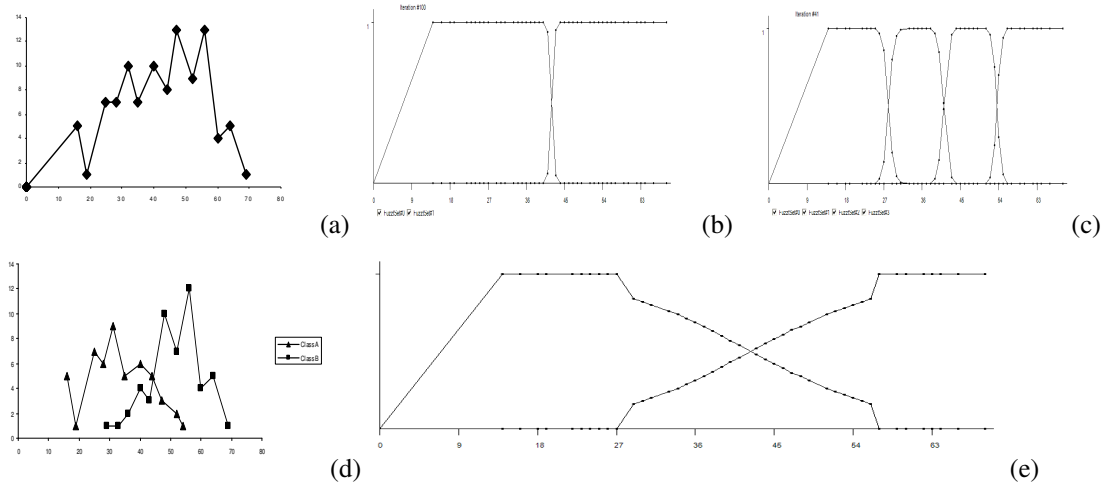

Fig. 6 Generating membership functions from overlapped dataset using PNN

V. CONCLUSIONS

Different elicitation methods for the fuzzy membership functions vary the interpretation of fuzzy membership functions. The proposed PNN technique generates membership functions which represent both semantic class labeling and statistical characteristic which are inherent in the data set. Furthermore, the unified framework can be used not only to generate one-attribute membership functions with or without class labeling but also to generate multi-dimensional membership functions given multiple-attribute data.

PNN combines the fuzzy logic and the neural work architecture in a single inference framework. The fuzzy logic system can be implemented in the framework with the automatically generated membership functions. This simplifies the design of the fuzzy logic system. Another advantage of using the proposed technique is the fast convergence of the PNN training. Instead of using the error back-propagation method to adjust weights in most FFNN, PNN estimates mutual information content used as the synaptic weights. Experiments show that the PNN training converges to the satisfying results in dozens of iterations. The research can be expended by implementing multi-dimensional membership function for pattern recognition or rule discovery.

References

[1] P. Xian-Tu, "Generating Rules for Fuzzy Logic Controllers by Functions," Fuzzy Sets and Systems 36 (1990) 83-89.

[2] O. Cordon, F. Herrera, L. Magdalena, and P. Villar, "A Genetic Learning Process for the Scaling Factors, Granularity
and Contexts of the Fuzzy Rule-Based System Data Base," Information Sciences 136 (2001) 85-107.

[3] O. Cordon, F. Herrera, and P. Villar, "Analysis and Guidelines to Obtain a Good Uniform Fuzzy Partition Granularity
for Fuzzy Rule-Based Systems Using Simulated Annealing," International Journal of Approximate Reasoning 25 (2000)

[4] T. Bilgiç and I. B. Türksen "Elicitation of Membership Functions: How far can theory take us?", Proceedings of FUZZIEEE '97, Barcelona July 1997.

[5] J.C. Bezdek, " Pattern recognition with fuzzy objective function algorithm, " Plenum Press, New York, 1981.

[6] H. Takagi and I. Hayashi, "NN-driven Fuzzy Reasoning," Int'l Journal of Approximate Reasoning (Special Issue of IIZUKA'88), Vol.5, No.3, pp.191-213 (1991)

[7] Kohonen, T. "The self-organizing map." Proceedings of the IEEE, 78(9):1464-1480,1990.

[8] Y. Y. Chen, "Plausible neural networks, " Advance in Neural Networks World, A. Grmela and N. E. Mastorakis, Ed. WSEAS Press 2002, pp. 180-185.

[9] Y. Y. Chen, "Plausible neural network with supervised and unsupervised cluster analysis," U.S. Patent 20030140020, July 24, 2003.

[10] W. Maass, "On the computational power with winner-take-all, " *Neural Computation*, 12 (11), 2000,pp. 2519-2536.

[11] Hersh, H., Carmazza, A. & Brownell, H. H, " Effects of context on fuzzy membership functions, " *in* M. M. Gupta, R. M. Ragade & R. R. Yager (eds), *Advances in Fuzzy Set Theory*, NorthHolland, Amsterdam, pp. 389–408, 1979.

[12] D. Simon, "Sum normal optimization of fuzzy membership functions, " International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems, vol. 10, pp. 363-384, August 2002.