ATLANTIS PRESS
Advances in Computer Science Research (ACSR), volume 72
IV International Research Conference "Information Technologies in Science, Management, Social Sphere and Medicine" (ITSMSSM 2017)

Fuzzy Neural Network Technology Support Decision-Making

Nguyen Anh Tu[1], Anatoliy M. Korikov [2]
[1,2]Dept. of Automation and Computer Systems, [2]Dept. of Automated Control Systems
[1,2]Tomsk Polytechnic University, [2]TUSUR University
Tomsk, Russia
[1]nguyenanhtu@tpu.ru, [2]korikov@asu.tusur.ru

Nguyen Anh Tuan
Faculty of Aerospace Engineering
Le Quy Don Technical University
Ha Noi, Viet Nam
anhtuannguyen2410@gmail.com

***Abstract*—The application of fuzzy neural network models using fuzzy neuron activation functions to solve the problems of clustering the intensity of Markov chains, whose distribution belongs to the exponential family, has been investigated and discussed in this study. The research is carried out with the help of the computer simulation tools of MATLAB software. Markov chains are presented with a ten successive elementary data sets, each of which is characterized by the intensity of arrival of events. Using fuzzy neural networks, the dichotomy problem is solved: the clusterization of the intensity of ten Poisson streams. The simulation results have validated the applicability of fuzzy neural network clustering of Markov chain intensity.**

Keywords— Fuzzy activation functions; Fuzzy neural networks; Markovian arrival processes; intensity clustering

I. INTRODUCTION

Neural network technologies (NNT) occupy leading positions among information technologies and they are successfully applied to solve many problems of science and technology, economics and management, social spheres and medicine. The main advantages of NNT are known and consist in the following [1, 2]: all NNT algorithms are highly parallel, and this is the key to high speed; Neurosystems (NS) can easily be made resistant to interference and destruction; stable and reliable NS can also be created from unreliable elements that have a wide range of parameters. In our previous work [3, 4], the new models of fuzzy NN (FNN) using fuzzy activation functions (AFs) were proposed. In such FNNs, fuzziness is an attribute of neurons and has the role of neural network element. These FNNs were defined in [3, 4] as the FNNs of the second type. These studies also indicated FNNs of the first type. As shown in [2], for this type of FNN, the fuzzy relationships between neurons are similar to those between the elements of NNs. In other word, first-type FNNs utilize fuzzy inference systems by NN methods. It is shown in [3, 4] that the FNN models developed by us and those of the second type can successfully solve the problems of time series prediction. In this paper, we investigate the use of the second type models to solve problems of clustering the parameters of Markov random processes, whose distribution belongs to a certain exponential family. We often face them when solving practical problems.

II. PROBLEM FORMULATION

Let consider a simple model of a random process whose distribution belongs to the exponential family. Let λ(t) - a piecewise constant random process with ten states λ(t) = λ1, λ(t) = λ2, … λ(t) = λ10. We do not observe a random process λ(t) in principle. The duration, in which the process λ(t) takes the value of the i-th state, is a random variable distributed exponentially with the parameters $\lambda_i \ (i=1,2,...,10)$ i.e. according to [5], the current random process expresses the Markovian Arrival Process (MAP) with exponential distribution functions.

$$F_i(t) = 1 - e^{-\lambda_i t}, \ i = 1; 2; ...; 10 \qquad (1)$$

MAPs are often used in Queuing Theory. The pioneer in this research area is the Danish scientist A.K. Erlang, whose scientific results were published in 1909-1917 with the application for telecommunication. The development of Queuing Theory is also related to some practical areas, such as financial management, transportation, communication and computer networks [6]. Markov chains are a particular case of MAP. [7, 8]. These chains can be represented in the form of successively chunks of the simplest (Poisson) flows, each of which is characterized by its arrival intensity λi. In this paper, we study the applicability of the second type FNN to the problems of classification and estimating the intensity λi of the current events. The study is conducted using the MATLAB software .

III. TRAINING DATA FOR FNNS

Experiments on the neural network clustering of Markov chains are performed on the basis of an FNN, whose structure is represented as a multilayer unidirectional network (figure 1). The FNN consists of an input layer, a hidden layer and an output layer. The hidden layer consists of 10 neurons with the second type fuzzy AFs [2, 3].

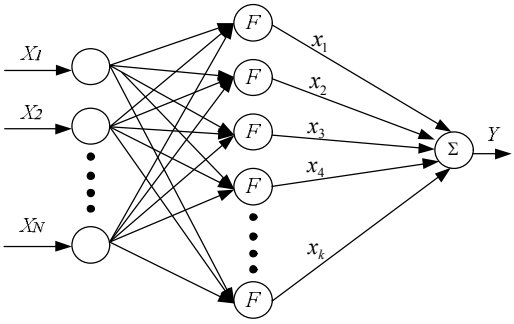

Fig. 1. The structure of the network

The input data consist of 500 random processes with parameters $\lambda_i\ (i = 1, 2, ..., 10)$. Each process contains values corresponding to 1000 time points. Hence, the input data can be given in the form of a 5000×1000 matrix. The output is also a 5000×1 matrix with the values of 1, 2,..., 10 for the parameters $\lambda_1, \lambda_2, ..., \lambda_{10}$, respectively.

IV. ACTIVATION FUNCTIONS

As noted above, the second type fuzzy AF can take one of the four following forms (Fig. 2) [2, 3].

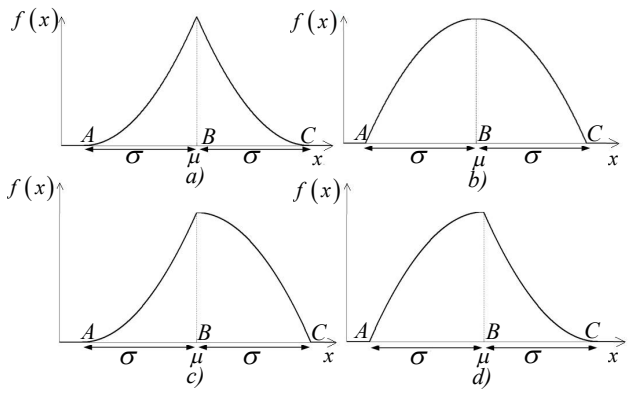

Fig 2. Activation functions

$\langle A, B, C \rangle$ are the characteristic points of the AFs. $f_L(x)$, $f_R(x)$ are second order polynomials and $f_L'(x) = 0$, $f_R'(x) = 0$ at the characteristic points, i.e.

$$f_L(x) = \sum_{i=0}^{2} a_i x^i \qquad f_R(x) = \sum_{i=0}^{2} b_i x^i \qquad (2)$$

For the AF of form 'a', the coefficients of the polynomials $f_L(x)$ and $f_R(x)$ are given by the following expressions:

$$a_0 = \frac{A^2}{(A-B)^2};\ a_1 = \frac{-2A}{(A-B)^2};\ a_2 = \frac{1}{(A-B)^2};$$
$$b_0 = \frac{C^2}{(B-C)^2};\ b_1 = \frac{-2C}{(B-C)^2};\ b_2 = \frac{1}{(B-C)^2}. \qquad (3)$$

Let $AB = BC = \sigma$; and $B = \mu$. Then $f_L(x)$ and $f_R(x)$ are defined as follows:

$$f_L(x) = \frac{1}{\sigma^2}x^2 + \frac{-2(\mu-\sigma)}{\sigma^2}x + \frac{(\mu-\sigma)^2}{\sigma^2},$$
$$\mu - \sigma \le x \le \mu$$
$$f_R(x) = \frac{1}{\sigma^2}x^2 + \frac{-2(\mu+\sigma)}{\sigma^2}x + \frac{(\mu+\sigma)^2}{\sigma^2},$$
$$\mu \le x \le (\mu+\sigma) \qquad (4)$$

where μ and σ denote the center and the width of the AFs.

Let $a = (\mu - \sigma)$ if $\mu - \sigma \le x \le \mu$ and $a = (\mu + \sigma)$ if $\mu \le x \le (\mu + \sigma)$, then we can obtain the general expression of the form 'a' AF:

$$f(x) = \frac{1}{\sigma^2}x^2 + \frac{-2a}{\sigma^2}x + \frac{a^2}{\sigma^2} \qquad (5)$$

Similarly, for the AF of form b, $f_L(x)$ and $f_R(x)$ are defined as follows:

$$f_L(x) = \frac{-1}{\sigma^2}x^2 + \frac{2\mu}{\sigma^2}x + \frac{(\mu-\sigma)^2 - 2\mu(\mu-\sigma)}{\sigma^2},$$
$$\mu - \sigma \le x \le \mu$$
$$f_R(x) = \frac{-1}{\sigma^2}x^2 + \frac{2\mu}{\sigma^2}x + \frac{(\mu+\sigma)^2 - 2\mu(\mu+\sigma)}{\sigma^2},$$
$$\mu \le x \le (\mu+\sigma) \qquad (6)$$

We can obtain the general expression of the AF in form b:

$$f(x) = \frac{-1}{\sigma^2}x^2 + \frac{2\mu}{\sigma^2}x + \frac{a^2 - 2\mu a}{\sigma^2} \qquad (7)$$

where $a = (\mu - \sigma)$ if $\mu - \sigma \le x \le \mu$ and $a = (\mu + \sigma)$ if $\mu \le x \le (\mu + \sigma)$.

For form c, $f_L(x)$ and $f_R(x)$ are defined as follows:

$$f_L(x) = \frac{1}{\sigma^2}x^2 + \frac{-2(\mu-\sigma)}{\sigma^2}x + \frac{(\mu-\sigma)^2}{\sigma^2},$$
$$\mu - \sigma \le x \le \mu$$
$$f_R(x) = \frac{-1}{\sigma^2}x^2 + \frac{2\mu}{\sigma^2}x + \frac{(\mu+\sigma)^2 - 2\mu(\mu+\sigma)}{\sigma^2},$$
$$\mu \le x \le (\mu+\sigma) \qquad (8)$$

Then, the general expression of the AF is:

$$f(x) = k_1\left(\frac{1}{\sigma^2}x^2 + \frac{-2a}{\sigma^2}x + \frac{a^2}{\sigma^2}\right) + k_2\left(\frac{-1}{\sigma^2}x^2 + \frac{2\mu}{\sigma^2}x + \frac{a^2 - 2\mu a}{\sigma^2}\right) \quad (9)$$

where $a = (\mu - \sigma)$, $k_1 = 1$, $k_2 = 0$ if $\mu - \sigma \le x \le \mu$ and $a = (\mu + \sigma)$, $k_1 = 0$, $k_2 = 1$ if $\mu \le x \le (\mu + \sigma)$.

Similarly, for the AF of form d, the general expression of the function is

$$f(x) = k_1\left(\frac{1}{\sigma^2}x^2 + \frac{-2a}{\sigma^2}x + \frac{a^2}{\sigma^2}\right) + k_2\left(\frac{-1}{\sigma^2}x^2 + \frac{2\mu}{\sigma^2}x + \frac{a^2 - 2\mu a}{\sigma^2}\right) \quad (10)$$

where $a = (\mu + \sigma)$, $k_1 = 0$, $k_2 = 1$ if $\mu - \sigma \le x \le \mu$ and $a = (\mu - \sigma)$, $k_1 = 1$, $k_2 = 0$ if $\mu \le x \le (\mu + \sigma)$.

Expressions (5), (7), (9), (10) are the activation functions used in the neuron kernels of the hidden layer illustrated in Fig. 1.

V. TRAINING FNN

During the training process, we have to select the values of the center, the width of each neuron, and the output weights, which are located between the neurons and the output nodes.

The center and distribution of activation functions should have characteristics similar to the data. One of the approaches to create the set of centers is to use K-means clustering [9] and the cluster centers μ. Each cluster corresponds to the kernel of the activation functions. A brief description of the K-means algorithm is expressed as follows:

Step 1: randomly select the number of initial cluster centers from the data sets presented above $\mu_1, \mu_2, ..., \mu_k$, where k is the number of initial centers of the cluster. The lager value of k is corresponding to the more accurate results and longer processing time.

$\mu_a = \{x_{b1}, x_{b2}, ..., x_{bN}\}$ where $a = 1; 2...; k$ and b is a random number ranging from 1 to *L*. *L* is the number of input data, *N* is the time numbers observing the input data.

Step 2: Define the Euclidean distance from x_i, $(i = 1; 2...; L)$ to cluster C_j $(j = 1; 2; ...; k)$ if $\|x_i - z_j\| < \|x_i - z_p\|$, $p = 1; 2; ...; k$, $j \ne q$ then x_i is located in cluster C_j.

Step 3: the new cluster centers μ_i^* are calculated by

$$\mu_i^* = \frac{1}{n_i}\sum_{x_j \in C_i} x_j, \ (j = 1; 2; ...; k), \quad (11)$$

where n_i is the amount of distributions belonging to the cluster C_i.

Step 4: if $\mu_i^* = \mu_i$ $(i = 1; 2; ...; k)$ then the program is terminated. Otherwise, go to step 2.

If there are time constraints and if the clustering process does not end in step 4, then it is executed for the given maximum number of iterations *ki*. The lager value of *ki* is, the more accurately the centers of clusters are determined.

Adjusting the width of the kernel when building the FNN is not an easy task. If the width of the core is too large, then the estimated probability density could be smoothed out. Conversely, when it is too small, there may be an excessive adaptation to a particular set of data. The width of the kernel is set to the average distance between the data of the corresponding cluster:

$$\sigma_i = \sqrt{\frac{1}{p}\sum_{j=1}^{p}(X_j - \mu_i)^2} \quad (12)$$

where σ_i is the width of the *i*-th hidden fuzzy neuron, μ_i is the center of the *i*-th hidden fuzzy neuron, p is the amount of functional data in this cluster. X_j is the *j*-th data in the *i*-th cluster. The introduction of the width's kernel (12) can ensure that the individual activation functions are not too sharp or / and not too flat.

The final set of parameters used for the training process is the output weights. When constructing the FNN, the task of adjusting the network weights between the hidden and the output layers can be regarded as an optimization problem solved by the gradient descent method (also known as the least mean squares). In this paper, the weights of the FNN between the hidden layer and the output layer are calculated using the pseudo-inversion technique [10, 11]. This algorithm is capable of overcoming many problems that occur in traditional gradient algorithms, such as the criteria for the program termination, learning speed, the number of epochs, and local minima. Due to the shorter learning time and the ability to generalize, this approach is more suitable for real-time applications.

In Fig. 1, X represents the sets of input data on the spatial input attributes. The output y of the FNN at the observed time j ($j = 1; 2; ...; N$, N is the number of observed times) has the following form:

$$y_j = \sum_{i=1}^{k} w_i f(x_i), \ j = 1; 2; ...; N, \quad (13)$$

where w_i is the weight between the output neuron and the ith hidden neuron; $f(x)$ is the generalized AF; *k* is the number of neurons in the hidden layer.

Equation (12) can be written in the form

$$Y = \Phi w, \quad (14)$$

where

$$\mathbf{\Phi}(\mu,\sigma,X)=\begin{bmatrix}\varphi_1(\mu_1,\sigma_1,X_1) & \dots & \varphi_K(\mu_K,\sigma_K,X_1)\\ \dots & \dots & \dots\\ \varphi_1(\mu_1,\sigma_1,X_v) & \dots & \varphi_K(\mu_K,\sigma_K,X_v)\\ \dots & \dots & \dots\\ \varphi_1(\mu_1,\sigma_1,X_N) & \dots & \varphi_K(\mu_K,\sigma_K,X_N)\end{bmatrix},$$

$$v=1;\ 2;\dots;\ N \qquad (15)$$

φ - Activation function. The matrix $\mathbf{\Phi}$ (dimension $N\times K$) is called the output matrix in the hidden layer; The *i*-th row is the *i*-th hidden output neuron with respect to the inputs $X_1, X_2....X_N$.

Let $\mathbf{T}$ - the matrix of the output data used for training. If $\mathbf{\Phi}w=\mathbf{T}$, then the output weights are calculated by the formula

$$w=(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{T}\ , \qquad (16)$$

where $(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}$ is a generalized pseudo- Moore-Penrose inverse output matrix of the hidden layer.

So, the main parameters of the FNN are determined by calculating the centers and width using the K-means clustering algorithm, and determining the weights of the output layer of the FNN using the pseudo-inversion method.

VI. TEST RESULTS

The experiments were performed using MATLAB R2013a and a Core i5-4460 3.2 GHz processor with 8 GB of RAM. In the experiment, the test data were generated randomly. The results of the experiment are shown in Fig. 3.

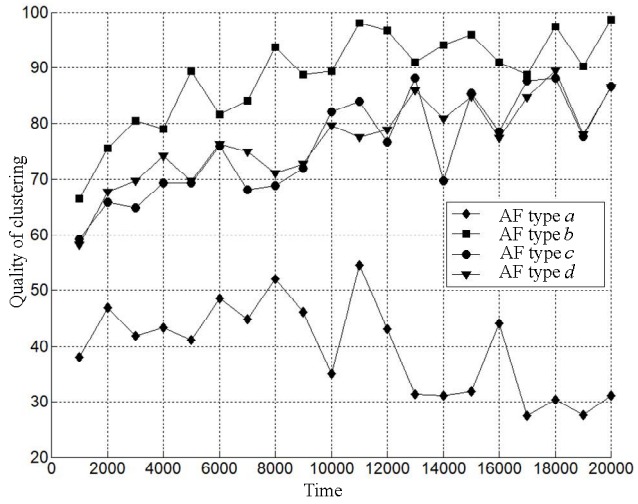

Fig. 3. Test results

Analyzing the experimental results, it is found that the FNN models developed in this work can successfully solve the problems of clustering in terms of the intensity of Markov chains. The FNNs with the AF of type *b* has the highest efficiency.

VII. CONCLUSION

The results of the study have confirmed the applicability of FNN models, in which fuzzy membership functions are used as activation functions to solve the problems of clustering Markov chain intensity. The study results have shown that it is possible to create a fuzzy neural network clusterizer for the intensity of Markov chains to resolve support decision-making problems in many practical fields, such as transportation, communication and computer networks.

REFERENCES

[1] S. Russell and P. Norvig , Artificial Intelligence: A Modern Approach,Third Edition , New Jersey: Prentice Hall, 1153 p., 2010.

[2] N. G. Yarushkina, “Lectures on Neuroinformatics,” Moscow: Moscow Engineering Physics Institute, part 1, pp. 151-199, 2004.

[3] A. T. Nguyen and A. M. Korikov, “Neural network model with fuzzy activation functions for time series predictions,” in Journal Proceedings of TUSUR, vol. 4, pp. 49- 51, 2016.

[4] A. T. Nguyen and A. M. Korikov, “ Neural network model with fuzzy activation functions,” in IOP Conference Series: Materials Science and Engineering, vol. 177, 2017.

[5] A. N. Dumim and V. I. Climenok, “Queuing systems with correlated flows,” Minsk: Buryat State University, 175 p., 2005.

[6] B. V. Gnedenko and I. N. Kovalenko, “Introduction to Queuing Theory , Second Edition,” Boston: Birkhauser, 314 p. 1989.

[7] A. M. Gortsev and L. A. Nezhel'skaya, “About connection of Markov chain and MAP,” in Tomsk State University Journal, vol. 1, pp. 13- 21, 2011.

[8] E. N. Bekkerman, S. G. Kataev, S. S. Kataev and D. U. Kuznetsov, “Approximation of a real event flow by the Markov chain,” in Tomsk State University Journal, № 14, pp. 2458- 253, 2005.

[9] L. Pierson, “Data science for dummies,” Published by: John Wiley & Sons, Inc., New Jersey, 367 p., March 2015.

[10] D. Wettscherch and T. Dietterich, “Improving the performance of radial basis function networks by learning center locations,” in Electronic Proceedings of the Neural Information Processing Systems Conference, № 4, pp. 1133- 1140, 1991.

[11] P. V. Saraev, “Using pseudo-inversion in training artificial neural networks,” in Electronic Journal "Investigated in Russia", № 29, pp. 308- 317, 2001.