Journal of Computer Science and Cybernetics, V.24, N.3 (2008), 295–306

A COMBINATION OF CONTEXT-FUZZY CLUSTERING METHOD AND LEARNING WITH FORGETTING ALGORITHM IN A NEURAL NETWORK MODEL TO GENERATING FUZZY RULES

BUI CONG CUONG[1], LE QUANG PHUC[2], NGUYEN THI AN BINH[3]

[1]*Institute of Mathematics, Vietnamese Academy of Science and Technology*
[2]*Tokyo Institute of Technology, Japan; Email:lqpbkhn@yahoo.com*
[3]*Department of Applied Mathematics and Informatics, Hanoi University of Technology;*
Email : haitrungkim02@gmail.com

Abstract. Many researches have been carried out to explore fuzzy rules from a given dataset. We are interested in generating fuzzy rules for the classification problem in contextual situations. In this paper, we present a neural networks model which has predetermined consequent parameters. The antecedent parameters are initiated basing on the result of the context fuzzy clustering method [4]. After structural training phase, rough fuzzy rules with concrete premises and consequent components are achieved. The parameter training phase then refines featured values of each feasible rules. Some initial experiment results for the classification problem of member countries of United Nation Organization (UNO) according the Human Development Index (HDI) [16] are presented. The statistics of UNO in the year 2006 is used.

Tóm tắt. Bài báo giới thiệu một mạng nơron cho phép tạo ra các luật mờ có các tham số được xác định trước. Các tham số của phần tiền tố của luật thu được nhờ một thuật toán phân cụm có ngữ cảnh. Sau pha luyện cấu trúc của mạng sẽ thu được các dạng luật có phần tiền tố và phần hệ quả khởi điểm. Các luật chính xác với các giá trị sẽ thu được sau pha luyện tham số của mạng nơron.

Bài báo cũng công bố một số thử nghiệm tính toán bước đầu cho bài tóan phân lớp các quốc gia theo dữ liệu thông kê năm 2006 của Liên hiệp quốc (UNO) tìm theo ngữ cảnh của chỉ số phát triển con người (HDI).

1. INTRODUCTION

With the growing demands for the exploitation of intelligent and highly autonomous systems, it would be beneficial to combine robust learning capabilities with a high level of knowledge interpretability. Fuzzy neuro computation supports a new paradigm of intelligent information processing [8, 9], in which we are able to achieve this powerful combination. Nowadays, neural network models [2, 3, 11] are developing more and more, and are applied widely in different realms of direct or indirect services and manufactures. It is also considered as a strong aid of rule extraction and data mining from a set of data [2, 5, 6, 7, 8, 9, 10], in which fuzzy factors are really common and rise up various trends to work on.

In this paper, we propose a neuro-fuzzy network to generate fuzzy rules for the classification problem. Given maximum of n input premises (or antecedence) and m output consequences, a fuzzy rule has the form:

If $(X_1 is A_{1i_1})\wedge(X_2 is A_{2i_2})\wedge...(X_n is A_{ni_n})\wedge$ Then $(Y_1 is B_{1k_1})\wedge(Y_2 is B_{2k_2})\wedge...\wedge(Y_m is B_{mk_m})$. in which, $X_1, X_2, ..., X_n$ are n premises, $A_j = \{A_{j1}, A_{j1}, ..., A_{j|A_j|}\}$,with $j = 1, 2, ..., n$ is the set of fuzzy values that j^{th} attribute can have. $i_j \in \{1, 2, ..., |A_j|\}$, $j = 1, 2, ..., n$. Similarly, $Y_1, Y_2, ..., Y_m$ are m consequences, $B_l = \{B_{l1}, B_{l2}, ..., B_{lB_l}\}$, with $l = 1, 2, ..., m$ is the set of fuzzy values that l^{th} consequence can have. $k_l \in \{1, 2, ..., |B_l|\}$, $l = 1, 2, ..., m$.

In former works on fuzzy neural networks, most authors classified and then expressed the range of each variable through a set of fuzzy numbers. Each fuzzy number can be the value of premises or consequent parts of rules. Consider the need of finding all the reasonable rules with the same predetermined consequences and each of which contains at most n premises. On average, each premise can receive k values, so there are $(k+1)^n$ candidates for rules. There are some ways to train structurally a neural network to achieve those reasonable rules and then this network can be used for reasoning.

With the proposed neural network model we are interested in generating fuzzy rules for the classification problem with context factors. We triy on a model of neural network having predetermined consequent parameters. After structural training phase, rules may have the number of premises less than the size of neural network inputs.

Moreover, sometimes as in our experiment, we can see that the same set of figures on, for instant, economy and society, can be interpreted differently, depending on different situations such as regions or countries. That means we can make use of the new definition of context variable [5, 6, −7] in our network to solve the problem. To say a little bit about our experiment, considering more than 100 variables in 31 fields reported by the United Nations in 2006 [16], we choose GDPPC (Gross Domestic Products per capita) the context variable as it is one of the most important factors to the calculation of HDI (Human Development Index). This index also makes up the relative concept of "groups of countries", which can be interpreted as "underdeveloped countries", "developing countries", "developed countries", and "super or strong developed countries". From those observations, we have pre-analyzed the data set with that way, we use the conditional fuzzy clustering method [4] to initiate neurons at the context layer, and following by using the fuzzy clustering method FCM [1] to initiate consequences at the output layer. This model is expected to achieve some feasible rules depending on contexts after being trained.

In addition, we also integrate the algorithm "learning with forgetting" supposed by Kasabov to optimize the number of nodes and connections in the neural network. In this paper, we tried on a model of neural network having predetermined consequent parameters. After structural training phase, rules may have the number of premises less than the size of neural network inputs.

The paper is organized as follows: The second section reviews the context fuzzy C-means method which was introduced by Pedrycz [4, 6]. The Section 3 states briefly about the algorithm "learning with forgetting" which will be applied in the parameter training phase of the neural network. The fourth section represents the neural networks structure to draw out rules from data base .The context fuzzy c-mean will be used in training process. Section 4.3 is dedicated to a structure learning algorithm. After fuzzy classification, each fuzzy center in each context forms one node in the context layer or premise layer. The number of nodes in input layer and output layer are already preset. Weight on each connection between two nodes of adjacent layers has the meaning of certainty or importance, according to specific cases that will be described later. The phase of structure training establishes the rule layer and related connections. In Section 4.4 we present two parameter learning algorithms. The second one applies the learning with forgetting algorithm. Weights on connections and memories of some nodes are refined in this phase. Redundant connections and nodes may also be omitted after calculation. Finally, some initial experiment results using the UNO data are given in Section 5 [16] will be presented.

2. CONTEXT FUZZY C-MEANS METHOD

Giving a dataset of n attributes, suppose that missing data have been processed, our purpose is to classify them into C clusters. There are many criteria to execute this. As usual, consider each set of n values corresponding to n attributes a data-point (or shortly a point) in n-dimension space. Applying crisp C-means method to the above problem, each point belongs to one and only one category and does not belong to any other one at all. All categories contain at least one point. The algorithm stops when the target function $J = \sum_{i=1}^{C} \sum_{k=1}^{N} \|x_k - v_i\|^2$ achieves the minimum, e.g., the sum of squared distances from each point to the center of the category that it belongs to reaches the least. Here, x_k is the k^{th} point, v_i is the center of i^{th} cluster, J is the error function or the lost function.

A better and more common method is Fuzzy C-means, which allows every point to belong to all clusters with memberships between 0 and 1, as long as the sum of membership measures of one point to all clusters is equal to 1. If some point belongs to some group with the membership measure 0, it actually does not belong to that group at all. Therefore, the above target function is adjusted to $J_m = \sum_{i=1}^{C} \sum_{k=1}^{N} u_{ik}^m \|x_k - v_i\|^2$, where m is a coefficient of fuzziness, u_{ij} is an element of distribution matrix U defined as follows:

$$U = \{u_{ik} \in [0,1] : \sum_{i=1}^{C} u_{ik} = 1 \text{ for } k = 1, 2, ..., N,\ 0 < \sum_{k=1}^{N} u_{ik} < N \text{ for } i = 1, 2, ..., C\},$$

u_{ik} is the membership value of the k^{th} point to the i^{th} cluster.

The fuzzy c-means algorithm has four steps:

1. Initiate the matrix U.
2. Re-calculate centers of each groups according to:

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^m x_k}{\sum_{k=1}^{N} u_{ik}^m} \text{ for } i = 1, ..., C.$$

3. Re- calculate matrix U as follow:

$$u_{ik} = \frac{1}{\sum_{j=1}^{C} (\|x_k - v_i\| / \|x_k - v_j\|)^{2/(m-1)}}, \quad \text{for } i = 1, ..., C,\ k = 1, ..., N.$$

4. If the error of distribution matrix $\|U(k+1) - U(k)\|$, defined through some analysis normal, is less than threshold δ, then the algorithm stops, else return to Step 2.

This algorithm is also proved to be converged to the minimum or the addle point of the target function [1]. Besides, the velocity of convergence varies depending on different ways of initiation, although they can converge to the same result after quite different numbers of steps.

We have just reviewed the development of ideas from the k-means method to the fuzzy c-means method in a classification issue. When putting them into the practice, scientists have innovated so that it meets the real requirements of the fact. Finding that the common fuzzy c-means has the trend of uniformly evaluation in the space of dataset, W.Pedrycz [4] ameliorated the algorithm at this point [5]. As presented in [5], if we can narrow the origin dataset under some conditions of certain dimensions, the velocity and efficiency of classification can be improved considerably, and the result focuses on the area that really has many relevant points. To achieve that improvement, we define a new variable Y called the context variable. Its determination is stated through the map:

$$A : Y \to [0,1]$$
$$y_k \mapsto f_k = A(y_k),$$

where Y is the subspace of origin n-dimension-space. From now on, we resign the size of dataset with the letter N. f_k value can be understood as the representation for the level of relation of the k^{th} point to the supposed context Y. There are some way of defining the relation between f_k and the membership of k^{th} point to the i^{th} cluster, for instant, using the sum operator or maximum operator. Then, the FCM must be adjusted as follows. The distribution matrix definition would be define as:

$$U = \{u_{ik} \in [0,1] : \sum_{i=1}^{C} u_{ik} = f_k \text{ for } k = 1,2,...,N,\ 0 < \sum_{k=1}^{N} u_{ik} < N \text{ for } i = 1,2,...,C\}.$$

And the formula for recalculating the membership value of k^{th} point to the i^{th} group is defined as:

$$u_{ik} = \frac{f_k}{\sum_{j=1}^{C} (\frac{\|x_k - v_i\|}{\|x_k - v_j\|})^{\frac{2}{(m-1)}}} \quad \text{for } i = 1,...,C,\ k = 1,...,N.$$

That means we only need to change the total membership of each point to all the groups. The sum is not necessary equal to 1, but it can vary from 0 to 1. It is obvious from those formulas that, if a point has no meanings in a certain context, its contextual value f_k will be equal to 0, and it plays no role in re-manipulating the positions of centers and the membership measures. The target function of the algorithm remains unchanged.

We have seen that standard FCM is a direction-free construction, which means it is regardless if a dimension is an input or an output variable. This may lead to a not very reasonable distribution of prototypes or centers of groups in that the algorithm sweeps over even unrelated areas in data space. In a specific case, the context-sensitive FCM allows us to concentrate the classification into a subspace due to conditions of some dimensions showed in defined context. Only a subset of origin dataset which has considerable meaning to the context is helpful in the algorithm.

3. "LEARNING WITH FORGETTING" METHOD

Some research have declared that most of neural networks, especially fuzzy neural networks have a large amount of abundant connections. This fact is one of the major reasons that cause the ineffectiveness of the networks operation. "Learning with forgetting" method has a purpose of optimizing the number of connection and nodes in a neural network. Its idea is that: a connection will be reduced gradually after each loop until it is "recalled" by the inversely propagated error signal. Consequently, a connection will be reduced till very close to zero if it is "forgot" too much, else it will be updated to a value quite far from zero if it is often received adjusted signals. The algorithm is based on the standard BP (back propagation) as shown in the following formulas:

$$J = \sum_{i}(y_i - y_i^*)^2 + \varepsilon' \sum_{i,j} |w_{ij}|.$$

This is the formula of error function, in which the first term itself is the error function value of the standard BP. The second term represents the innovated idea. λ is the learning rate. ε is the forgetting rate. And $\varepsilon' = \varepsilon\lambda$ is also a constant. Then, the weight is recalculated as:

$$\Delta w_{ij} = \Delta w'_{ij} - \varepsilon sgn(w_{ij}).$$

These improvements have some advantages. First, the error function will not reduce as fast as that of the standard BP; however, in the opposite aspect, it may fall into the "hill-climbing" situation if it decreases too slowly. This depends on the value of chosen forgetting rate constant. Second, it allows changing the networks structure, and therefore, decrease the overtraining risk caused by redundant connections. Nevertheless, the computation time increases, so, this algorithm has higher complexity compared to the standard BP. This weak point has been solved by some enhances such as zeroing method which is also introduced by Kasabov. In this application, we do not focus on that requirement. The main effect we have to pay attention here is the chose of forgetting constant. If it is too large, the network may become rigid because the connections are decreased and omitted too fast. In that case, the network will lack the flexibility and accuracy. Vice verse, if the forgetting rate is too small, the forgetting effect will be too slow and it will take rather long for the network to train. More details and examples of the excellent advantage of this method can be found in the following.

4. NEURAL NETWORK ARCHITECTURE TO GENERATING FUZZY RULES

4.1. Statement of the problem

There are many ways to construct a neural network model and to find rules from a database, depending on specific tasks of building rules. In this study, we combine the CFCM into the initiate stage of the neural network. Suppose that the context variables have been defined and denoted by $CT = \{CT_1, CT_2, ..., CT_t\}, t \in N^*$. This means there are t contexts we are interested in. Denote $x_1, x_2, ..., x_n$ are n input variable, each of which represents one attribute that can appear in the premise sets of rules. Assume there are m fuzzy values $y_1, y_2, ..., y_n$ can be assigned to the consequences of one rule. Without losing the generalization, suppose that we classify the origindata into C fuzzy clusters according to each context. The model of neural network has four layers as below and illustrated by the Figure 1.

4.2. A neural networks model

The concrete model is described as follows:

- Input layer: Each neuron has only one input signal. Aggregation function and transfer function are homogeneous. The number of neurons is n. Links between layer 1 and layer 2 are full connected, and the weights are identically equal to 1.

- Premise layer: There are t groups of neurons corresponding to t contexts. Each group contains $(n+1)$ neurons (1 context neuron and n attribute neurons), in which the first one, colored in black in the figure, is always the context neuron. It receives enough n signals from all n input-layer-neurons. This neuron then calculates the maximum level that the inputs can active one of C classes in the same context. The next n neurons, denoted by red nodes in the figure, in the same group represent for n input variables. Each of them only receives one signal from the input neuron at the same dimension. Then it works out the maximum level that the single input activates fuzzy value corresponding to that dimension of centers in the current context. So there are 2 types of neurons at this layer. All the outputs of neurons at layer 2 are real numbers in interval $[0, 1]$. We also set a threshold at each neuron, so that, if some input signal is too weak, for instant, it will be omitted from prefixes set of generated rules.

- Rule layer: each node represents a rule found out. The number of neurons is limited by a predetermined value and by t times the size of training dataset, to ensure that it is not too large (one pattern may generate at most t new rules). By the way we will train structurally the network; the number of incoming signals to one neuron is the number of premises of that rule, including one special signal. This special signal is propagated on the link from one context neuron at layer 2 to that rule, showing the level that context is activated by inputs of neural network. Thus, the number of links from layer 2 to one rule at layer 3 is various from 2 (if only 1 premise is logged) to $(n+1)$ (if all n input become n premises). Output of one neuron at this layer indicates the aggregated activation level for that rule.

- Output layer: the number of neurons is m. In general case, it is the total fuzzy levels of all consequent variables. There are full connections from layer 3 to layer 4. That means each link from all the layer-3-neurons to one output layer neuron represents the manipulated certainty of the consequence for each rules. Every rule will lead to all consequences with different certainty between 0 and 1. Each output neuron is based on the incoming signals and weights on the same link to determine the last certainty of that consequence.

In addition, we can also add one more layer as a defuzzification phase for the neural network as usual. But we do not focus on that phase at this rule searching problem.

4.3. A structure learning algorithm

Before training the network, we do context fuzzy classification for each of four contexts, each of which into three fuzzy classes. After fuzzifying those $T \times C = 12$ clear prototypes, we gain fuzzy prototypes of sets of $T \times C$-dimension-triangular figures. These fuzzy figures are used to initiate nodes at layer 2 of the network.

On the other hand, doing free context fuzzy classification for the origin dataset according to HDI, we get the result to build links from layer 3 to layer 4 of the network. After initializing a network with determined number of neurons at layer 1, layer 2 and layer 4, we can apply the following algorithm to train the structure and set links from layer 2 to layer 3 as well as the temporary values on weights from layer 3 to layer 4 of the network.

Algorithm

1. Read one data point from dataset, propagate through layer 1.
2. Get the output from layer 2. Compare this to the set of premises of each existed rules.

If it coincides with none of premise sets of all existed rules, generate new rule and rule node at layer 3.

Else

Suppose that it coincides with that of rule noted R^*. Compare the value of layer 2 outputs with corresponding premise components of R^*. If they are completely the same on all dimensions, that means the input data point activates exactly an existed rule R^*. Then, skip Step 3 and execute Step 4, then return Step 1; otherwise, proceed Step 2 for the next existed rule until all rules are compared.

3. If all the existed rules are compared and no rule has the same values with that of corresponding dimension of layer 2 output, generate new rule and rule node at layer 3. Only meaningful components will have connections to the new rule.
4. Adjust the weight on the link from activated rule to the m neurons at output layers. We use the mean values here to re-calculate each weights.
5. If the number of rule nodes reaches the preset limitation or all the data points have been read, stop the procedure; otherwise return to Step 1.

Besides this, all the connections from layer 1 to layer 2, and from layer 2 to layer 3 are set equal to 1.

4.4. A parameter learning algorithm

In fact, the links from the input layer to context neurons at premise layer can be interpreted as the importance of correspondent dimensions in each context. The weights on connections from rule layer to output layer make a sense of the confidence index of each rule. From this point of view, it would be better to integrate already knowledge of fuzzy combination rules into the training procedure. However, in simpler cases, we can gather difficulties of parameter training process into two positions on the network. One of them is the real weights on the connections from rule neurons to output neurons. The other is the fuzzy numbers memorized on neurons at layer 2. In this paper, we install the simulating program with learning parameters using back propagation algorithm as follows.

Algorithm 1

For each pattern or data point from the training dataset:

1. Propagate input signals through the network and get the outputs, which mark m rules that give the maximum possibilities of m consequences.
2. Use the gradient reduction method to determine adjusting value on each links from rule neurons to output neurons.
3. Use the back propagation formulas to calculate total adjusting values on features parameters of each fuzzy number in the memories of layer-2-nodes. These changes also have the same meanings with adjusting weights on links from layer-2-nodes to layer-3-nodes.
4. Read the next pattern from dataset or stop if there is no pattern remains.

To enhance the parameter training process, we combine the learning-with-forgetting method with the above procedure to make a subloop. By that way, after each subloop, the structure of network can be "shrink" in the sense crossing away abundant connections. Then, the main loop is described as follows.

Algorithm 2

1. Execute exactly the whole loops with epochs as in the previous parameter learning algorithm, except for using the formulas of learning-with-forgetting instead of that of standard BP.
2. Reject connections having weight smaller than a preset threshold θ. If some node has no incoming connections and outcoming connections, reject it too. Increase the number of main loop by 1.
3. Repeat 1 and 2 until error function is small enough or after upper limitation value of the number of main loops

There is an important procedure in the Step 2. Suppose that there exists a connection to and from one rule corresponding to the same attribute, such as variable of dimension i. So, one of these or both have to be crossed away. Remind that, the weight value on a connection to one rule shows the importance of corresponding variable to that rule, and the weight value on a connection from one rule represents the certainty of corresponding consequence if that rule is activated. Thus, if both of them are larger than the threshold, the connection with smaller weight will be rejected.

Formula of error value on a connection from neuron i to neuron j of the output layer can be as follows:

$\Delta w_{ij} = -\eta[d_j - a(net_j)]f(net_j)o_i$, where η is learning rate constant, d_j is the task output of neuron j get from the dataset; o_j is the real output of the neuron j; In case $a(net_j)$ is the maximum function, we choose $f(net_j) = o_k$ if $a(net_j) = \max_i\{o_i \times w_{ji}\} = o_k \times w_{jk}$. And $f(net_j) = 0$ for all $i \neq k$. In case $a(net_j)$ is a linear combination function, we choose $f(net_j)$ the derivative function of $a(net_j)$.

For other connections, the formula is $\Delta w_{ji} = \sum_{k=1}^{|l|} \Delta w_{kj} \times w_{kj} \times f(net_j) \times o_i$, where $|l|$ is the number of neurons at layer l to which there has been a connection from neuron j at layer $_{(l-2)}$; o_i is the output of neuron i at layer $_{(l-2)}$.

We adjust the support of the triangular fuzzy numbers stored or "memorized" in neuron i according to the following:

$$\delta = \Delta w_{ji} \times (1 - o_i^j),$$

$$\Delta w_{ji}^b = \eta \times \delta_i \times sgn(\min_j |d_j - o_j^*|)(w_{ji}^c - w_{ji}^a),$$

$$\Delta w_{ji}^c = \eta \times \delta_i \times (w_{ji}^c - w_{ji}^a) + \Delta w_{ji}^b,$$

$$\Delta w_{ji}^a = -\eta \times \delta_i \times (w_{ji}^c - w_{ji}^a) + \Delta w_{ji}^b,$$

where $sgn(\min_j |d_j - o_j^*|)$ is the sign of the minimum difference of all the pair real output and task output at output neurons of the network.

4.5. Other notations and options

In the training algorithm above, we have to calculate the membership degree of one fuzzy number to another one, and of one fuzzy vector to another fuzzy vector. In the formula to manipulate the membership degree of one fuzzy vector to another, we can set different weights for each variable, or set all to 1 if all the variables should be evaluated equally. In case of discrete data and continuous data appear in one pattern, we have some ways to process the similarities as in many publishes, such as the following way, which calculates similarity between two n-dimension-vectors x and v :

$$s(x,v)=\frac{\sum\limits_{q=1}^{n} s_q(x,v)}{\sum\limits_{q=1}^{n} w_q},$$

where $s_q(x,v)=1-\dfrac{|x_q-v_q|}{r_q}$ is the similarity between component q of vector x and component q of vector v., r_q is an optional scale coefficient. We can set $r_q=v_q$. Another way to choose is that, in the formula $s_q(x,v)$ above, we take $|x_q+v_q|$ in stead of r_q to dispose unit of component q, $w_q=0$ if the component q of at least x or v are undefined or component q is a binary variable and they are both equal to 1 in x and v, $w_q=1$.

On the other hand, after context fuzzy classification, we execute the fuzzification by establishing adjacent triangular fuzzy numbers which have the minimum and maximum points attached to adjacent centers. In the case of the first and last fuzzy numbers, we plus or minus an interval equal the largest distance between any two centers in that dimension. By that way, we get the corresponding triangular fuzzy figures.

Furthermore, in comparison with the well-known ANFIS [17, 18], the model here pays more attention to different contexts. Therefore, fuzzy numbers stored in premise neurons may coincide partly because they are correspondent to different contexts. In addition, one pattern may generate at most T new rules if it activates all T contexts with the activation greater than given threshold, which is different to the mechanism in ANFIS structural training. That is the basic dissimilarity between the two fuzzy inference system models.

5. SOME EXPERIMENT RESULTS

In this study, we illustrate the model using the statistics of the United Nation Organization in order to evaluate the comprehensive development level of member countries, especially the Human development index (HDI). That means, in a specific case, consequences of a rule are preset to fuzzy levels of HDI attributes. The theory of a neural network model to find rules with the consequences unpreset has been introduced in the Section 4. However, its simulating program is still on the tip of completion.

Every year, UNO collects more than 100 figures of 31 fields from 177 countries. The highest aim of all the effects and activities of this organization is to upgrade the quality of human livings. The most important general index to appraise that progress is the HDI. This index is calculated through analytic functions of longevity index, education index and GDP (Gross Domestic Products) index and many others parameters. In this study, we try to rule out some relations showing the affection of 5 out of 100 parameters that we most appreciate to HDI. They are GDPPC, Life expectancy, Enrollment ratio, Number of telephones every 1000 people, and Public expenditure on health. In fact, these values are mutually dependent on each others and on other factors of social economy and major forms of services. However, these 5 parameters say differently in different groups of countries. This is the point that we can exploit to apply "context" concept. As mentioned in the Introduction section, "Groups of countries" are defined relatively, basing mainly on annual GDPPC or GDP. For example, if the ratio of illiteracy is "medium", level of internet popularization is "low", etc...., and the expense for public education and public health are "medium", then, in the context of a developed country, HDI is surely not "high". Its reason is may be that the country does not attend to the real quality of livings, or gets high incomes just due to extra interest from possession of certain resources. But if a developing or underdeveloped country has the same set of statistics, its HDI may be "high". As usual, HDI is scaled to the unit interval $[0, 1]$.

In fact, strong developed countries generally have great GDP, and hence have virtual conditions to upgrade the human life. Countries with high GDP and GDP per capita often have high HDI. However, the range of GDP per capita is rather wide; the minimum is 561 USD, while the maximum reach 69961 USD. This is a very important index and is the most dispersive attribute among indices. Thus, we can consider GDP per capita (GDPPC) our context variable.

Chose t is equal to 4, C is equal to 3, n is equal to 5, and m is equal to 4. We wish to gain 3 fuzzy classes in each context because this seems reasonable for our small size of dataset. Then, extracting from the dataset of UNO 2006 [16], we have 177 sets of figures (or records), each of which considers a 6-dimension-vector (including the HDI index also).

Divide this into four levels so that we can relatively classify 177 countries according to four level of HDI: "very high", "high", "medium", and "low", which are defined by trapezoidal or triangular fuzzy number as following:

- 4 contexts: $T1(150, 450, 650, 1050)$ mainly includes backward, very poor or underdeveloped countries; $T2(950, 4000, 7000, 10000)$ almost consists of low developed, rather poor or average developing countries; $T3(9500, 13000, 18000, 25000)$ consists of most of fast developing countries; $T4(24500, 30000, 35000, 40000)$ contains most of strong developed and superpower countries. (UNO also has notice that the highest value is considered 40000 due to the comfort calculation).

- 4 levels of HDI: $T1(0; 0.5; 0.61)$ for low HDI; medium HDI defined by $T2(0.6; 0.7; 0.81)$, fair HDI by $T3(0.8; 0.85; 0.91)$, and $T4(0.9; 0.93; 1)$ for the high HDI.

- The dataset includes statistics in 2006 for 177 countries, classified into 3 groups each context.

Linguistic meanings of reasoning in these rules can be imagined like stating in this example: according to rule number 60, if the ratio of illiterate is "medium", popularization of internet is "medium", etc, and expanse for public health is "high", then in the context of "low" GDPPC, HDI is "very likely" "high", in which all the degree words in the quotes " " are marked by weights from variable neurons at layer 2 to rule neuron at layer 3 for the premises, and marked by output signals between [0,1] of the output layer for the possibilities such as "very likely".

We divide the original dataset into two parts, one consists of 110 records for training, and the other for testing. On the CPU socket 486, Ram 256MB, Windows Vista OS, the time of structural training is 0.5 second, and about 42 minutes to train parameters in 1000 epochs, and the upper limitation of rules is equal to 50. The maximum number of rules can be discovered is 73 (in case setting upper limitation of rules to any value larger than 73).

When testing by the second part of dataset, there are 9 patterns are misclassified. Besides the reason of imperfect trained network, another reason is that there are still many missing values in the dataset. We have to replace them by average values at the same dimension and this causes error as well. Another reason for this not high accuracy of the network is that the size of dataset is not large enough. In the future, we will lengthen the table of dataset and train again to get the highest feasible accuracy of the network.

Moreover, these 50 rules can be categorized into 22 different structures. Different structures are featured by the appearance or disappearance of premises for rules. Let see an example: with the input (77.9; 99%; 496/1000; 38827 USD; 5.8%) (Ireland), output of the network show that these inputs activates rule 12 with degree 0.2114, 0.181 for rule 37, 0.343 for rule 30 and 0.686 for rule number 60. That means it makes the most sense according to rule 60: in the context 3 with "high" GDPPC, enrollment ratio is "high", the number of telephone per 1000 people is "low", "low" expenditure on health, HDI index is "very high" with the possibility of 0.9878 (this figure itself is the weight on the link from the rule node 60 to the output neuron corresponding to the "high" HDI index).

As perspective, the lengthen table of dataset, other kinds of aggregation operators, and activation functions at each neuron can be used to achieve the highest feasible accuracy of the network.

6. CONCLUSION

In this paper, we have proposed a model of four-layer neural network. We briefly introduced a model of four-layer-neural network to rule out automatically fuzzy rules after structural training phase. This model can also makes use of both training data and expert knowledge in that to build the structure of four-layer neural network, we even can initiate rule nodes before training structures. The mechanism of its work is like a process of applying known rules to reasoning. Propagating given full n inputs or even missing some inputs, the output of neuron will indicates the highest possibilities of the conclusion of consequence levels, along with the rule index giving that result. Moreover, from the list of rules worked out, we can evaluate the relative importance of some input variables to rules in each context. For instant, the absence of factor "telephone per 1000 people" is more often than other factors may imply that we should replace this factor by some other more important factors from more than 100 attributes of each pattern from UNO. In this research, the set of premises of a rule is flexible. The generalization of the proposed four-layer neural network is done by making the set of consequences of a rule is changeable.

In the future, we expect to discover other methods to upgrade the effectiveness of finding such generated rules.

REFERENCES

1. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.
2. G. Bortolan, An architecture for fuzzy neural networks for linguistic processing, *Fuzzy Sets and Systems* **100** (1998) 197–215.

3. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Second Edition, Prentice Hall, New Jersey, 1999.
4. W. Pedrycz, Conditional fuzzy ***C***-mean, *Pattern Recognition Lett.* **17** (1996) 625–632.
5. G. Bortolan, W. Pedrycz, Linguistic neurocoomputing: the design of neural networks in the framework of fuzzy sets, *Fuzzy Sets and Systems* **128** (2002) 389–412.
6. W. Pedrycz, Fuzzy set technology in data mining and knowledge discovery, *Fuzzy Sets and Systems* (3) (1998) 279–290.
7. W. Pedrycz, A. Rocha, Fuzzy-set based models of neuron and knowledge-based networks, *IEEE Trans. Fuzzy Systems* **1** (1993) 254–266.
8. W. Pedrycz, Heterogeneous fuzzy logic networks: fundamentals and developments studies, *IEEE Trans. Neural Networks* **15** (2004) 1466–1481.
9. A. F. Gobi, W. Pedrycz, The potential of fuzzy neural networks in the realization of approximation reasoning engines, *Fuzzy Sets and Systems* **157** (2006) 2954–2973.
10. L. A. Zadeh, Fuzzy logic, neural networks, and soft computing, *Comm. ACM* **37** (1994) 77–84.
11. B. C. Cuong, N. D. Phuoc, *Fuzzy Systems, Neural Networks and Applications*, Second Ed., Science and Technology Pub., Hanoi, 2006.
12. B. C.Cuong , "Fuzzy logic and applications", Lecture at the Center for Talent Engineers, Hanoi University of Technology, 2003.
13. N. K. Kasabov, R.Kozma, M. J. Watts, *Phoneme-based Speech Recognition Via Fuzzy Neural Networks Modeling and Learning*, 1997.
14. D. Nauck, R. Kruse, Neuro-fuzzy methods in fuzzy rule generation, *Fuzzy Sets in Approximate Reasoning and Information Systems*, Eds. J. C.Bezdek, D. Dubois, and H. Prade, Kluwer Acad. Pub., London, 1999 (305–334).
15. E. H. Ruspini, P. P. Bosnissone, W. Pedrycz, *Handbook of Fuzzy Computation*, Eds. Enrique H.Ruspini, Piero P. Bosnissone, and Witold Pedrycz, Institute of Physics Pub., Briston, 1998.
16. http://hdr.undp.org/en/reports/global/hdr2006
17. Estimation Régionale des débits, De crues par la méthode ANFIS, *Rapport de recherche No R-930*, Par Chang Shu, Taha B.M.J.Ouarda, Avril 2007, ISBN 978-2-89146-542-7.
18. R. Ghosh, "A Novel Hybrid Learning Algorithm for Artificial Neural Networks", Ranadhir Ghosh, BE (Comp.Sc.), Phd. Thesis, MIT, Submitted in fulfillment of the requirements of the degree of Doctor of Philosophy, December, 2002.

Received on September, 29 - 2008