DICTA2002: Digital Image Computing Techniques and Applications, 21--22 January 2002, Melbourne, Australia.

An Evolutionary Connectionist Fuzzy System for Multispectral Data Classification

Khaled M. Mahar
Computer Engineering Department, P.O.Box 1029, AASTMT, Alexandria, Egypt,
Email: khmahar@aast.edu

Abstract

In this paper, numerical exemplars are used in a training method to find the structure of a fuzzy-neural network. After this structure learning, a genetic algorithm is applied to determine the initial weights of the neural network, thereby guiding the neural network to a near-optimal initialization. These well-initialized networks are then trained with backpropagation algorithm. Using this proposed approach, the local minimum phenomenon, which may cause the learning process to stagnate, can be avoided. Overall learning performance is, thus, significantly improved. The proposed method has been implemented and tested on the Thematic Mapper sensor system (TM) data to get the fractional representation of each class within a pixel. Results show the potential of the proposed method for this kind of applications.

1. Introduction

In recent years, several fuzzy-neuron systems have been proposed, and their applications cover a broad range, including pattern classification [1][2][3]. The concepts and techniques of fuzzy-set theory are uniquely helpful in the practice of pattern recognition. One of these concepts is at the class-membership level, where in the fuzzy set approach, a pattern does not necessarily belong to just one class. There is a certain degree of possibility that the pattern might belong to each one of the classes, and the membership functions supply values for these various possibilities [4].

The aim of this paper is to build a fuzzy classifier model with three significant goals. Firstly, the model should take the input pattern in a numeric form and converts it to linguistic form to feed into a fuzzy rule-base, and then outputs the identification vector corresponding to the input. Secondly, the fuzzy rules should be adapted during learning to improve the effect of the input variables on the output. The last objective is during the processing where the fuzzy rules should be matched in parallel to give the consequences due to the applied pattern, thus saving the rules matching time in a traditional fuzzy logic system. These goals can be achieved by combining the properties of artificial neural networks (ANN) with that of a fuzzy-set theory.

In fact, there are many systems that combine the properties of fuzzy set theory and that of ANN [1][2][5][6][7]. Most of these models are based on the multi-layer perceptron structure and the back-propagation learning scheme. However, the most important limitation of the backpropagation training method is that it does not ensure convergence to the global minimum of the error function. It has been shown that the genetic algorithm (GA) [6] with its global optimization capability can solve this problem. The GA is used to determine the link weights of the ANN and then the best resulting weights are used to find a solution via backpropagation. In this way, each algorithm is used to its greatest advantage: the GA -with its global search- determines a sub-optimal weights to solve the problem, and backpropagation -with its local search- seeks the best solution in the area of the weight spaces found by GA.

Also, in this paper, the initial structure of the ANN is built by using an unsupervised algorithm. This algorithm optimizes the number of fuzzy rules necessary for the classification process. Thus avoiding network growth when the fuzzy terms of the input pattern is large in number. In addition, the idea of elastic fuzzy logic (EFL) [8] is used to get adaptable parameters during the learning phase of the ANN. This approach overcomes the problems associated with the most common method which adapts the membership functions of a fuzzy-neuron system. When adapting the membership functions, the purpose is to improve the overall performance of the system. But one of the problems associated with this approach is that if an expert supplies the fuzzy rules, then communication back to the expert is impaired when the definitions of the linguistic variables are changed so that they mean something else from what the expert intended.

The reminder of this paper is organized as follows: in section 2, the proposed fuzzy-neural model is introduced. Then, the learning algorithms are presented in section 3. The application of this model to a multispectral data is given in section 4. Finally, the conclusions are the subject of section 5.

2. A fuzzy neural network system

A fuzzy logic system consists of a set of rules in a simple if-then form. Each rule takes the form, "if x_1 is T_{x1} and x_2 is T_{x2} ... then y is Ty." A membership function $\mu_{Tx}(x_i)$ is used to indicate the degree of applicability of term set Tx to the input linguistic variable x_i. Also a membership function $\mu_{Ty}(\mathbf{y})$, may be supplied to describe how well a given value y fits the output term Ty in rule number r; however, in practice only the center value y_r of the term T_y is sufficient to be supplied. With fuzzy inference and centroid-defuzzification, the rules and the output of the system can be defined as follows

$$R_r = \mu_{T_{x1}^i}(x_1)\mu_{T_{x2}^i}(x_2)\cdots\mu_{T_{xn}^i}(x_p) \quad r = 1,\cdots,N_R \;\&\; i = 1,\cdots,m \tag{1}$$

$$\hat{y}_j = \frac{\sum_{r=1}^{n_R} R_r y_r}{\sum_{i=1}^{n_R} R_r} \qquad j = 1,2,...,q \tag{2}$$

where p is the number of input variables, N_R is the total number of fuzzy rules, m is number of membership functions for each input, n_R is the number of input rules to an output term node, $\hat{y}$ is the crisp output, and q is the number of output variables . Intuitively, R_r represents the degree to which rule number r applies in the current situation x. To adapt the system defined by (1) and (2), Eq. (1) is replaced by:

$$R_r = \omega_r \mu^{\gamma 1}_{T_{x1}^i}(x_1)\mu^{\gamma 2}_{T_{x2}^i}(x_2)\cdots\mu^{\gamma m}_{T_{xn}^i}(x_{n_A}) \quad r = 1,\cdots,n_R \;\&\; i = 1,\cdots,m. \tag{3}$$

where ω and γ parameters are called *elasticities* (hence the term elastic fuzzy logic) and can be adapted by the ANN. They represent the degree of strength of validity of the rule. This is important in practical applications because real numerical data have different reliabilities. For example, bad data will be assigned low elasticities during adaptation, so that its rule could be deleted without affecting the performance of the system.

Figure 1 shows an ANN model which can perform the classification process based on fuzzy logic. Nodes at layers one and three are *term nodes* which act as membership functions to represent the terms of the respective linguistic variables. Each node at layer two is a rule node which represents one fuzzy rule. Thus, all layer-two nodes form a fuzzy rule base. Links at layers two and three function as connections *inference engine* that simulate the rule-matching process [9]. For each node in the network, we can write the output of the node as

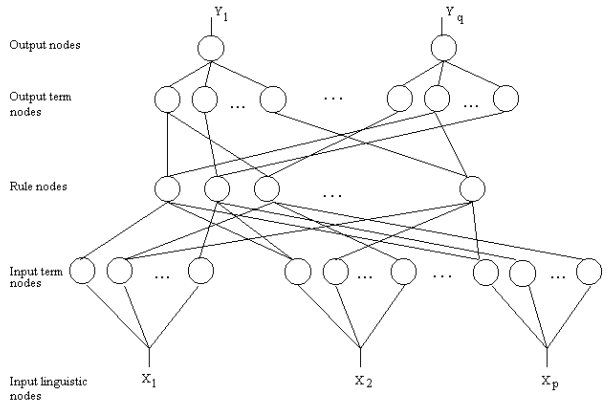

Figure 1. Configuration of fuzzy neural network model.

$$\hat{o} = f(net) = f(U, W) \tag{4}$$

where U is the input vector to the node, the function **f** represents what the node does, and **W** is an adjustable weight vector. The previous equation is defined at each layer as follows:

Layer 1: Each node performs a membership function, then the output of this node should be this membership function. We assign to each term node a bell shape membership function,

$$\mu(u) = e^{-\frac{(u-m)^2}{\sigma^2}} \tag{5}$$

where m and σ are the center and width of the membership function respectively. From (4), the node function is given by

$$net_1 = -\frac{(u-m)^2}{\sigma^2}, f = e^{net_1}. \tag{6}$$

In this study, we consider the nodes in this layer as radial basis function (RBF) which output the membership degree for the input linguistic variables, thus the weight is unity.

Layer 2: The outputs of this layer are the precondition matching of fuzzy logic rules. Hence,

$$net_2 = \prod_{j=1}^{p} w_j u_j, f = net_2 \tag{7}$$

where u_j is the output of the preceding layer, and w_j is a training parameters which can be used as elasticities to reflect the importance of a term in a rule. In this study w_j is put to unity.

Layer 3: the links at this layer represents the elasticities w_r (the degree of importance of the rule), and the function of each node in this layer perform fuzzy OR operation to integrate the fired rules which have the same consequences, thus

$$net_3 = \sum_{j=1}^{n_R} w_j u_j, \; f = \min(1, net_3) \qquad (8)$$

where n_R is the number of input rules to the node, w_j is the elasticities, and u_j is the output from the preceding rule nodes.

Layer 4: This layer simulates the defuzzifier. If m_j and σ_j are the center and width of membership function of term j for an output variable, then the following functions can be used to compute the center of area defuzzification method [9]:

$$net_3 = \sum_{j=1}^{n_R} w_j u_j, \; f = \min(1, net_3). \qquad (9)$$

Here the w_j is $(m_j\sigma_j)$, and u_j is the output from the preceding layer. The f values of this layer represent the crisp values of the classification process.

3. Generating and tuning fuzzy neural network

Three steps are established in this study to generate the fuzzy rules. The first step finds the parameters of membership function by unsupervised clustering algorithm. The second generates the fuzzy rules from a number of training samples, while the last step trains the ANN in a supervised manner to estimate the elasticities of the fuzzy rules. The following subsections illustrate the function of each step extensively.

3.1 Finding membership function parameters

Before the training of the ANN, the membership functions and fuzzy rules must be known, so the structure of the ANN can be constructed. The centers and widths of the membership functions are determined by unsupervised learning algorithm. The input in this learning phase is the training data x_i, i=1,2,...,n, and the number of fuzzy partitions $|T(x)|$ with their initial centers. The output is the optimal representation of the input data through the learned centers. This places the domains of membership functions to cover only those regions of the input /output spaces where the data are present. Kohonen's feature-maps algorithm [10] is used here to find the center m_i of the membership function:

$$\|x(k) - m_{closest}(k)\| = \min_{1 \le i \le k} \{\|x(k) - m_i(k)\|\} \qquad (10)$$

$$m_{closest}(k+1) = m_{closest}(k) + \alpha(k)[x(k) - m_{closest}(k)] \qquad (11)$$

$$m_i(k+1) = m_i(k) \;\; \text{for } m_i \neq m_{closest} \qquad (12)$$

where $\alpha(k)$ is a monotonically decreasing scalar learning rate. This adaptive formulation runs independently for each input and output linguistic variable. Once the centers of membership functions are found, their width can be simply found by first-nearest-neighbor as

$$\sigma_i = \frac{|m_i - m_{closest}|}{a} \qquad (13)$$

where *a* is an overlap parameter.

3.2 Generating fuzzy rules from numerical data

After the parameters of membership functions have been found, the fuzzy rules can also be generated from the numerical data [11]. Suppose we are given a set of desired input-output data pairs $(x_1^i, x_2^i; y_1^i)$, i=1,2,...,n. the following two subsections generate the fuzzy rules.

3.2.1 Generate fuzzy rules from given data pairs. First, determine the degrees of a given x_1^i, x_2^i, and y_1^i in different terms. Second, assign the given x_1^i, x_2^i, and y_1^i to the region with maximum degree. Finally, obtain one rule from one pair of desired input-output data.

3.2.2 Assign a degree to each rule. Since there are usually lots of data pairs, and each data pair generates one rule, it is highly probable that there will be some conflicting rules, i.e. rules that have the same IF part but a different THEN part. One way to solve this conflict is to assign a degree to each rule, and accept only the rule from the conflict group that has maximum degree. In this way not only the conflict is resolved, but also the number of rules is greatly reduced. The degree of the rules is computed as in (1).

3.3 Adapting the elasticities of the fuzzy rules

After the fuzzy rules have been identified, the network structure is established, and a supervised training is applied to adapt the elasticities of the fuzzy rules. The purpose of this phase is the changing of the elasticity parameters to minimize the error between the actual output of the ANN and the desired output over the training set. First a GA is used to determine sub-optimal elasticities. Then, the best elasticities developed by the GA are used to find a solution via backpropagation. In this way, the algorithm avoids the local minima convergence of the backpropagation.

3.3.1 Genetic algorithm. In GA [11] the search space of the problem is represented as a collection of individuals. The individuals are represented by character strings, which are often referred to as chromosomes. The purpose of the use of GA is to find the individual from the search space with the best genetic material. The quality of an individual is measured with an objective function.

Various schemes for augmenting GAs and ANNs have been proposed in recent years. The simplest scheme uses a GA as a stand-alone learning algorithm for ANN [12]. Another scheme where a network is represented as a genotype that has six kinds of genes was proposed in [13]. The genes are a learning rate, a slant of sigmoid function, a coefficient of momentum term, an initializing weights range, the number of layers and the unit numbers of each layer. Genetic operators affect populations of these genotypes to produce adaptive networks with higher fitness values. Ignizio [15] proposed other genotype ANN. He used a GA for simultaneous design and training of an ANN classifier. To do so, a GA which codes both the weights as well as the number of masks that separate the different categories was used. A small-space problem was tested by this model and a pleasing results were obtained. Larranga *et al.* [15] presented an approach to structure learning in the field of Bayesian networks. They tackled the problem of the search for the best Bayesian network structure, given a database of cases, using a GA philosophy for searching among alternative structure. Another idea of augmenting GA with ANN was presented in [16] where a GA searches among candidate solutions of the problem, while the ANN provides the objective function value of each candidate solution. This method is suitable for problems where the evaluation of the objective function is computationally time consuming and may seem ill-fitted. Still another method which presented in [6] is a hybrid algorithm that combines the modified quasi-Newton method and the GA. It exploits the high convergence rate of the quasi-Newton method in the trust region to obtain a local minimum, and it uses the GA to search for a better offspring from the local minimum point.

In this paper, instead of finding a local minimum and ensure that it is a global minimum or to find offspring from a local point, the GA is used as an initial learning algorithm for the ANN. It searches the error-space to find a near-optimum value which is then considered a seed for the backpropogation algorithm. This is accomplished by means of a chromosome of the following form:

$$\gamma_1\gamma_2\gamma_3\cdots\gamma_n \tag{14}$$

where γ_i represent the *elasticities* of the fired rule number i (the weights of layer three), and n is the number of rule nodes. It should be noted that a floating-point numbers are employed rather than binary strings for the representation of each element of the chromosome. The mutation method applied in this work is to add a delta to the genes by means of a mutation probability. The other operators are employed as in conventional GA. The goal is to minimize the classification error of the system so that the function which measures the fitness of a chromosome is the squared error function of the ANN defined by:

$$E = \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2 \tag{15}$$

where y_i is the desired output, and $\hat{y}_i$ is the actual output of training sample number i.

3.3.2 Application of backpropagation algorithm. After the best-GA elasticities have been found, the backpropagation algorithm is used to find the optimum elasticity values. The goal is to minimize the error function defined in (15). For each training data set, starting at the input nodes, a forward pass is used to compute the activity levels of all the nodes in the network. Then starting at the output nodes, a backward pass is used to compute $\frac{\partial E}{\partial y}$ for layer three nodes which contains the adaptable weights or the elasticity parameters. The general learning rule used is

$$w_{k+1} = w_k + \eta(-\frac{\partial E}{\partial w}) \tag{16}$$

where η is the learning rate, and from which we get:

$$\frac{\partial E}{\partial w_{ij3}} = -u_{i3}\sum_{k=1}^{K}(y_k - \hat{y}_k)\frac{m_{jk}\sigma_{jk}(\sum_\ell \sigma_{\ell k}u_{\ell k}) - (\sum_\ell m_{\ell k}\sigma_{\ell k}u_{\ell k})\sigma_{jk}}{(\sum_\ell \sigma_{\ell k}u_{\ell k})^2} \tag{17}$$

4. Application to multispectral image classification

In remote sensing images most pixel reflectance value results from the spectral mixture of a number of classes that constituting the sensed surface area. However, in most studies, classification algorithms are used to assign a pixel to a unique ground cover class. An alternative is to unmix the pixel to get the fractional representation of each class within the pixel. The applicability of the proposed system in unmixing pixels is tested using data taken from the image of Landsat-5, TM for path (165) and row (045) for al-Kharj area, south east of Riyadh, Saudi Arabia of (1990). The data used for training and test consist of 50 pixels of pure vegetation, 50 pixels of soil, and another 50 pixels of water from other area. These pixels were used to create a 1000 mixed-pixel for training the system and to test the ability of the system in the classification. Since the TM system has 6 usable bands [17], there are six input linguistic variables representing these bands. While the output linguistic variables represent the fractional representation of each category in a pixel.

Figure 2 shows the learned membership functions for the six bands. Note that each band (variable) associates with three terms only. More terms produce more fuzzy rules but in our problem, we found that three terms were adequate. Note also that the membership functions cover only the usable range of each band, thus ensuring the resolution of the fuzzy rules.

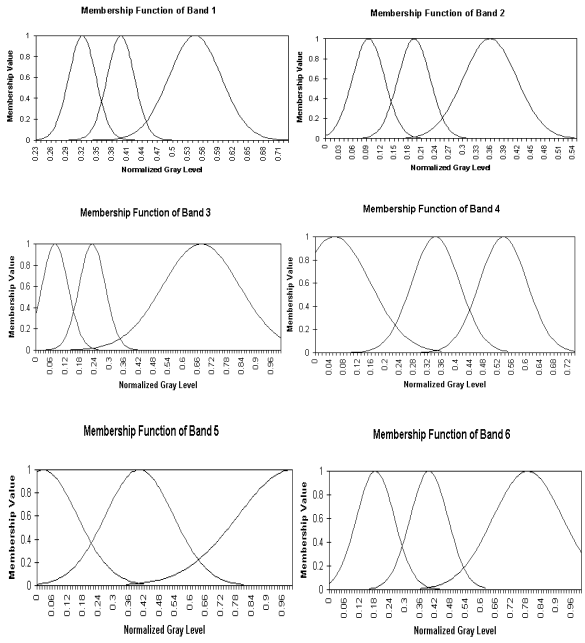

Figure 2. Learned membership functions for Band-1to Band-6.

Figure 3 shows the minimization of the error function. It can be seen from figure 3-a that GA does the most work, while the backpropagation (figure 3-b) improves the result. Note that in the GA, the offspring do not necessary better than their parent, this can be seen from the wiggling of the curve. But to overcome this problem, our method selects the better chromosome from the entire generations.

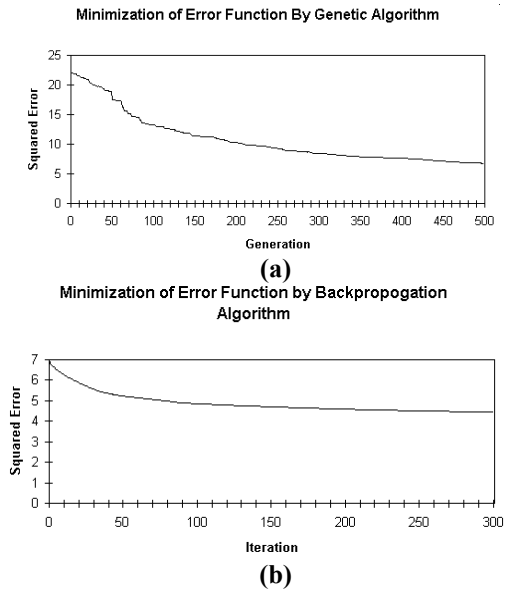

Figure 3. The minimization of error function (a) with genetic algorithm, (b) with backpropagation algorithm.

Table (1) shows the result of classification for some pixels. The first six rows of the table contain pure pixels for three different categories. These are two rows of vegetation, two of soil, and the last two of water.

The target data columns are computed by using a linear unmixing model, while the proposed model computes the output columns. The classification process for these pixels performed very well, although there is a small fraction of error in some rows. This may be happened if the selected pixel is on the border between two different categories. As can be seen from the table for other pixels, the biggest RMS (root mean square) error is found in row number seventeen, and is less than 0.09.

Comparing these results with that of the neuro-fuzzy system reported in [3], we find that the results of the classification are agreed. However, the result of proposed model is accepted because each band is partitioned into three terms only, while in [3] each band is associated with ten terms. Thus the result is satisfied and can be improved if we increase the number of partitions for each input variable. But this increasing is proportional with the training time which is already large.

Table 1. Representative results from the fuzzy-neural model.

P#	Contents						
	Vegetation		Soil		Water		RMS
	T	A	T	A	T	A	
1	1.00	1.00	0.00	0.00	0.00	0.00	0.0000
2	1.00	0.99	0.00	0.00	0.00	0.01	0.0082
3	0.00	0.00	1.00	1.00	0.00	0.00	0.0000
4	0.01	0.00	0.99	1.00	0.00	0.00	0.0082
5	0.01	0.02	0.00	0.00	0.99	0.98	0.0082
6	0.00	0.03	0.00	0.00	1.00	0.97	0.0024
7	0.12	0.16	0.83	0.82	0.05	0.02	0.0294
8	0.01	0.09	0.14	0.00	0.85	0.90	0.0097
9	0.51	0.57	0.49	0.43	0.00	0.00	0.0490
10	0.60	0.56	0.38	0.37	0.02	0.07	0.0374
11	0.32	0.34	0.28	0.25	0.40	0.41	0.0216
12	0.05	0.18	0.70	0.66	0.25	0.17	0.0911
13	0.46	0.47	0.54	0.53	0.00	0.00	0.0082
14	0.91	0.96	0.09	0.03	0.00	0.01	0.0455
15	0.64	0.66	0.36	0.32	0.00	0.02	0.0283
16	0.74	0.73	0.10	0.09	0.16	0.18	0.0141
17	0.64	0.53	0.36	0.47	0.00	0.00	0.0898
18	0.29	0.30	0.54	0.52	0.17	0.18	0.0141
19	0.07	0.16	0.70	0.67	0.22	0.17	0.0619
20	0.40	0.35	0.48	0.52	0.12	0.13	0.0374

P#: Pixel number T: target A: Actual

5. Conclusions

A general neural network based on a fuzzy logic model was proposed. The model is trained by genetic algorithm in the first phase and then by backpropagation algorithm in the second phase to avoid the local minimum convergence. The proposed model uses the idea of elastic fuzzy logic to adapt the fuzzy rules during the supervised learning of the neural network. Thus making the whole data more effective in computing the consequences of the fuzzy rules. It is noted that the number of fuzzy rules depends on the number of terms of both the input and output linguistic variables. In other words, as the number of terms increase, the number of generated fuzzy rules is also increased, so the performance in the classification is enhanced. The method of generating fuzzy rules which used in this study reduces the number of rules during generation, thus avoiding the large memory requirement before learning as in other methods. Although the use of genetic algorithm in training the system avoid the local minima problem, the genetic algorithm performance is sensitive to its parameters especially the mutation and crossover probabilities. The result of the proposed system is satisfied and can be improved by increasing the number of terms for each variable.

References

[1] Pal, S. K., and Mitra, S., "Multilayer perceptron, fuzzy set, and classification," *IEEE Transactions on Neural Networks*, vol. 3, 1992, pp. 683-697.

[2] Pedrycz, W., "Fuzzy neural networks with reference neurons as pattern classifiers," *IEEE Transactions on Neural Networks*, vol. 3, 1992, pp. 770-775.

[3] Mahar K., *Multispectral image identification using clustering, linear analysis, and neural- network-based fuzzy logic system*, Ph.D. Thesis, Electrical Engineering Department, Cairo University, Giza, Egypt. 1996.

[4] Pao, Y., H., *Adaptive pattern recognition and neural networks*, (MA:Addison-Wesley), 1989.

[5] Kuo, Y., Kao, C., and Chen, J., " A fuzzy neural network model and its hardware implementation," *IEEE Transactions on Fuzzy Systems*, vol 1, 1993, pp. 171-183.

[6] L. Zhang, Y. Li, and H Chen, "A new global optimization algorithm for fuzzy neural networks," Int. J. electronics, vol. 80, No. 3, 1996, pp. 393-403.

[7] Detlef N., Radolf K., "How the learning of rule weights affects the interpretability of fuzzy system," Proceeding IEEE international conference on fuzzy systems, Anchorage, Ak, May 1998, pp. 1235-1240.

[8] Werbos, P., "Neurocontrol and Elastic fuzzy logic: capabilities, concepts, and applications," *IEEE Transactions on Industrial Electronics*, vol 40, 1993, pp. 170-180.

[9] Lin, C. T., and Lee, C. S. G., "Neural-network-based fuzzy logic control and decision system," IEEE Transactions on Computer, vol 40, 1991, pp. 1320-1336.

[10] Kohonen T., self-Organization and Associative Memory. Berlin, Germany: Springer-Verlag, 1988.

[11] Holland, J. H., Adaptation in N natural and Artificial Systems, Ann Arbor, Mich.: The University or Michaigan press, 1975.

[12] Janson D.J. and Frenzel J.F., "Application of genetic algorithms to the training of higher order neural networks," Journal of Systems Engineering, vol. 2, no. 4, pp. 272-276, 1992.

[13] Takahashi, H., Agui, T., and Nagahashi, H.D, 1993 Designing adaptive neural network architectures and their learning parameters using genetic algorithms. Proceedings of the SPIE - The International Society for Optical engineering, no. 1966, 1993, pp. 208-215.

[14] Ignizio, J. P. and Soltys J. R., Simultaneous design and training of ontogenic neural network classifiers. Computer Ops Res., vol 23, 1996, pp. 535-546.

[15] Larranaga P., Poza M., Yurramendi Y., Murga R., and Kuijpers C., "Structure Learning of Bayesian Networks by Genetic Alorithms: A performance analysis of control parameters," IEEE Transaction on pattern analysis and Machine intelligence, vol 18, 1996, pp. 912-925.

[16] Coit D., W. and Smith A. E., "Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. Computer Ops Res., vol 23, 1996, pp. 515-526.

[17] Lillesand T. M. and Kiefer R. W., Remote sensing and image interpretation, New York: Wiley, 1987, pp. 570-580.