An Approach of Neuro-Fuzzy Classifiers Using the Example Dependent Costs

Alina Constantinescu

***Abstract*—The current research from the field of economic sciences uses the neural and neuro-fuzzy networks procedures as an alternative to classical methods. This paper deals with neuro-fuzzy classifiers and presents a new neuro-fuzzy classifier which considers the example dependent costs in the inductive construction of classifiers.**

***Index Terms*—artificial learning, neuro-fuzzy classifier, costs.**

I. Introduction

During the last decade, there has been an increase in the integration of neural networks (NN) and fuzzy logic. Many NN algorithms have been successfully adapted to neuro-fuzzzy networks (see [1], [2], [3]). So, the classification problems make use of several new neuro-fuzzy classifiers which can solve them (see [5], [6], [7], [9]).

Some of the most important applications of the classification theory are especially in the economic field. At present, the economists do a lot of work to develop models that take into account the subjective and imprecise character of human thinking and behaviour. On the other hand, there are many classification economic applications which need to be treated using the example dependent costs (see [4], [8]). These arguments lead to the conclusion that we should use the costs in neuro-fuzzy training algorithm.

In this article we start from a neuro-fuzzy classifier (NEFC) proposed by Taur and Tao in [10], then we design a new NEFC that considers the cost functions in the inductive construction of the classifier. This approach makes the example dependent costs of the teaching values have their contribution when the error is computed.

In the following section we give a brief description of the Taur and Tao's training algorithm, and then Section 3 presents our new approach. Finally, in Section 4 we summarize the discussion from this paper.

II. NEURO-FUZZY CLASSIFIER

In [10], [11] Taur and Tao define a neuro-fuzzy classifier with adjustable rule importance. Here we present their NEFC for two classes.

Let $x = (x_1,...,x_N)$ an N -dimensional input pattern, N_x the number of the input patterns, $C = 2$ the number of class and R_i the number of rules in class i.

For two classes and for the input pattern $x = (x_1,...,x_N)$, the fuzzy rules of classifier are:

Positive Rule: If $x_i = r_i$, $i = 1,..,N$ then x is in class i

Negative Rule: If $x_i = r_i$, $i = 1,..,N$ then x is not in class k

The training algorithm is globally supervised and locally unsupervised. The initialization of values for a class is obtained using a clustering unsupervised method, then the classifier is modified using a supervised method with a training set composed of the training patterns.

The network is trained with the reinforced and anti-reinforced learning rules.

For a noise tolerance level $(1-n_t)$, the errors are:

$$e_i = t_i - f_i^* \tag{1}$$

Where $f_i^* = \dfrac{\alpha_i}{\beta_i}$ is the output for class i and represents the crisp defuzzification for the i_{th} output with:

$$\alpha_i = \sum_{r=1}^{R_i} (+1) y_r^{ii} z_r^i + \sum_{i \neq j, j=1}^{2} \sum_{r=1}^{R_j} (-1) y_r^{ij} z_r^j \tag{2}$$

$$\beta_i = \sum_{j=1}^{2} \sum_{r=1}^{R_j} y_r^{ij} z_r^j \tag{3}$$

where the matching degree for the if-part, r_{th} rule and class j is

$$z_r^j = \exp\left(- \sum_{n=1}^{N} \frac{(x_n - m_{rn}^j)^2}{(\sigma_{rn}^j)^2} \right) \tag{4}$$

In the above formula, m_{rn}^j and σ_{rn}^j are the mean and the variance for the n_{th} membership function corresponding to j_{th} class and r_{th} rule.

Manuscript received March 1, 2009.

A. Constantinescu is with the Department of Science, Valahia University, Targoviste, Romania (e-mail: alinaconsta@yahoo.com).

The value t_i is the i_{th} component of the learning vector $t^j = (t_1, .t_2)$ and has the value: $t_i = -1$ if $i \neq j$ or $t_i = 1$ if $i = j$.

In the classification procedure, after the outputs of each class f_i^* are computed, is established the class p which has the maximum output using the following formula:

$$p = \arg\left(\max_i f_i^*\right) \tag{5}$$

The confidence measure for a pattern from class i is:

$$C_m^i = \frac{\sum_{r=1}^{R_i} y_r^{ii} z_r^i}{\max_{r,i,j} y_r^{ij}} \tag{6}$$

and confidence measure average is:

$$\frac{1}{N_x} \sum_{i=1}^{C} \sum_{x \in class\ i} C_m^i \tag{7}$$

where N_x is the total number of patterns.

The training procedure uses the following updating rule:

$$\omega_j^{k+1} = \omega_j^p - \sum_{i=1}^{C} g_{ij} \gamma(i, j, p) \tag{8}$$

For a pattern from class p , Taur and Tao compute three types of training gradient information, taking into account the node which is in consideration and the class which the updating rule belongs to:

$$\begin{aligned} &1.\ g_{ij},\ j = p \quad \gamma = \eta_1 \\ &2.\ g_{ij}, i = p,\ j \neq p \quad \gamma = \eta_2 \\ &\quad g_{ij},\ i \neq p,\ j \neq p,\ \ i = j \quad \gamma = \eta_2 \\ &3.\ g_{ij},\ i \neq p,\ j \neq p,\ i \neq j \quad \gamma = \eta_3 \end{aligned} \tag{9}$$

They call the first type ‘reinforced learning’, the second one ‘anti-reinforced learning’ and the third is called ‘cross-reinforced learning’. Thus, the weighting factors $\gamma(i, j, p)$ can be η_1, η_2 or η_3 and they increase or decrease the absolute value of the outputs of the rule nodes and the absolute values of the importance weighting factors for the rules from the same class.

III. NEURO-FUZZY CLASSIFIER WHICH CONSIDERS THE EXAMPLE DEPENDENT COSTS

A frequent economic problem is to classify an individual in one of the two classes when a certain cost is defined for each class. Usually this cost represents the loss which is registered when an individual is incorrectly classified. The aim is to minimize the total loss, that means to design a good classifier with a minimum classification error see [4].

Our NEFC uses for the supervised method a learning algorithm which considers the criterion function depending on costs.

For two classes C_{+1} , C_{-1} and for example dependent cost function c_{+1}, c_{-1}, we define the criterion function as follows:

$$J(w) = \frac{1}{N}\left[\sum_{x \in C_{+1}} c_{+1}(x)(t_i - f_i^*)\sigma(f_i^* - t_i) + \sum_{x \in C_{-1}} c_{-1}(x)(f_i^* - t_i)\sigma(t_i - f_i^*)\right] \tag{10}$$

where σ is:

$$\sigma(a) = \begin{cases} 1 & for \quad a \geq 0 \\ 0 & for \quad a < 0 \end{cases} \tag{11}$$

Function $c_{+1}(x)$ represents the cost of an incorrect classification in class C_{-1} when feature vector x is from class C_{+1}. Function $c_{-1}(x)$ represents the analogue cost for class C_{-1}. In many practical cases these functions are known apriori (see [4]).

Let T_n, T_c, T_a the training thresholds, where T_n is the desired noise tolerance threshold, and η_1, η_2, η_3 the weighting factors for the first, second and third type $(\eta_3 \leq \eta_2 \leq \eta_1)$.

The gradient of J which corresponds to the i_{th} output node has the components:

$$\begin{aligned} g_{ij} &= 0 \ \text{ if abs}(e_i) < 1 - n_t \\ g_{ij} &= \frac{\partial J_i}{\partial \omega_j} \ \text{ if abs}(e_i) \geq 1 - n_t \end{aligned} \tag{12}$$

The variable ω_j represents one of the parameters $y_r^{ij}, m_{rl}^j, \sigma_{rl}^j$.

A) *The computing formula*

The computing formula for g_{ij} is:

$$\frac{\partial J_1}{\partial \omega_j} = \frac{1}{N}\left[\sum_{x \in C_{+1}} -c_{+1}(x)\frac{\partial f_1^*}{\partial \omega_j}\sigma(t_1 - f_1^*) + \sum_{x \in C_{-1}} c_{-1}(x)\frac{\partial f_1^*}{\partial \omega_j}\sigma(f_1^* - t_1)\right.$$

When the parameter ω_j is y_r^{1j}:

$$\frac{\partial f_1^*}{\partial y_r^{1j}} = \frac{\partial \alpha_1}{\partial y_r^{1j}} \beta_1^{-1} - \alpha_1 \beta_1^{-2} \frac{\partial \beta_1}{\partial y_r^{1j}} \qquad (13)$$

But for two-class situation:

$$\alpha_1 = \sum_{r=1}^{R_1} (+1) y_r^{11} z_r^1 + \sum_{r=1}^{R_2} (-1) y_r^{12} z_r^2 \quad \text{and} \qquad (14)$$

$$\beta_1 = \sum_{r=1}^{R_1} y_r^{11} z_r^1 + \sum_{r=1}^{R_2} y_r^{12} z_r^2$$

With two previous formulas for α_1 and β_1 we obtain:

$$\frac{\partial f_1^*}{\partial y_r^{11}} = z_r^1 \beta_1^{-1} - \alpha_1 \beta_1^{-2} z_r^1 \quad \text{and}$$

$$\frac{\partial f_1^*}{\partial y_r^{12}} = -z_r^2 \beta_1^{-1} - \alpha_1 \beta_1^{-2} z_r^2 =$$

$$= -z_r^2 \beta_1^{-1} - f_1^* \beta_1^{-1} z_r^2 .$$

When the parameter ω_j is m_{rl}^j, (13) becomes:

$$\frac{\partial f_1^*}{\partial m_{rl}^j} = \frac{\partial f_1^*}{\partial z_r^j} \frac{\partial z_r^j}{\partial m_{rl}^j}$$

So, $$\frac{\partial f_1^*}{\partial m_{rl}^1} = \left(y_r^{11} \beta_1^{-1} - \alpha_1 \beta_1^{-1} y_r^{11} \right) \cdot \left(\frac{2\left(x_l - m_{rl}^1\right) z_r^1}{\left(\sigma_{rl}^1\right)^2} \right)$$

$$\frac{\partial f_1^*}{\partial m_{rl}^2} = \left(- y_r^{12} \beta_1^{-1} - \alpha_1 \beta_1^{-2} y_r^{12} \right) \cdot \left(\frac{2\left(x_l - m_{rl}^2\right) z_r^2}{\left(\sigma_{rl}^2\right)^2} \right)$$

When the parameter ω_j is σ_{rl}^j:

$$\frac{\partial f_1^*}{\partial \sigma_{rl}^j} = \frac{\partial f_1^*}{\partial z_r^j} \frac{\partial z_r^j}{\partial \sigma_{rl}^j}$$

So, $$\frac{\partial f_1^*}{\partial \sigma_{rl}^1} = \left(y_r^{11} \beta_1^{-1} - \alpha_1 \beta_1^{-1} y_r^{11} \right) \cdot \left(\frac{2\left(x_l - m_{rl}^1\right)^2 z_r^1}{\left(\sigma_{rl}^1\right)^3} \right)$$

$$\frac{\partial f_1^*}{\partial \sigma_{rl}^2} = \left(- y_r^{12} \beta_1^{-1} - \alpha_1 \beta_1^{-2} y_r^{12} \right) \cdot \left(\frac{2\left(x_l - m_{rl}^2\right)^2 z_r^2}{\left(\sigma_{rl}^2\right)^3} \right)$$

B) *Training procedure for NEFC*

Step 1: Initialize the parameters and thresholds.

Step 2: Initialize a noise tolerance level 1-n_t.

Step3: If $n_t < T_n$ then go to Step 3, otherwise increase n_t and go to Step 2.

Step 4: Compute the outputs and C_m^i for each class i using Eqs. 2-6 and for each pattern x in class i. If $C_m^i > T_c$ then go to Step 5, otherwise go to Step 6.

Step 5: If $\text{abs}(e_i) \geq 1 - n_t$ then update parameters using the updating rule from the Eqs 8, 12, otherwise go to Step 6.

Step 6: If $C_{ma} < T_a$ then STOP otherwise go to Step 7.

Step7: If iteration > max-iteration, increase η_2 and η_3 and save the current parameters, otherwise go to Step 2.

IV. CONCLUSIONS

In this article we considered extension of the Taur and Tao's classifier for which we incorporated the example dependent costs. Thus, many economic classification problems in which the costs are important, can be solved using our new classifier.

REFERENCES

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press: Oxford, 1995.

[2] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons: New York, 1973.

[3] I. Gath, A.B. Geva, *Unsupervised Optimal Fuzzy Clustering*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 11, no. 7, 1989, pp. 773-781.

[4] P. Geibel, F. Wysotzki, *Learning Perceptrons and Piecewise Linear Classifier Sensitive to Example Dependent Costs*, Applied Intelligence 21, pp. 45-56, 2004.

[5] S. Knerrr, L. Personnaz, G. Dreyfus, *Handwritten digit recognition by neural networks with single-layer raining*, IEEE Transaction on Neural Networks, vol. 3, no. 6, pp. 962-968, 1992.

[6] S. Y. Kung, J. S. Taur, *Decision –based Neural Networks with Signal/Image Classification Application*, IEEE Trans. On Neural Networks, vol. 6, no. 1, 1995, pp. 170-181.

[7] S. Lawrence, C. L. Giles, A.C. Tsoi, A. D. Back, *Face Recognition: A Convolutional Neural Network Approach*, IEEE Trans. On Neural Networks, vol. 8, no. 1, pp. 98-113.

[8] A. Lenarcik, Z. Piasta, *Rough classifier sensitive to costs varying from object to object*, Proceedings of the 1 st International Conference on Rough Sets and Current Trends in Computing (RSCTC-98), edited by

L. Polkowski and A. Skowron. Vol. 1424 of LNAI, Springer, Berlin, pp. 222-230, 1998.

[9] D. Michie, D. H. Spiegelhalter, C. C. Taylor, *Machine Learning, Neural and Statistical Classification,* Series in Artificial Intelligence, Ellis Horwood, 1994

[10] J. S. Taur, C. W. Tao, *A New Neuro-Fuzzy Classifier with Application to On-Line Face Detection and Recognition,* Journal of VLSI Signal Processing 26, pp. 397-409, 2000.

[11] J. S. Taur, C. W. Tao, *An On-line Face Detection and Recognition System Using Neuro-Fuzzy Classifier*, Proceedings International Symposium on Multimedia Information Processing, Dec. 1999, pp. 297-302.