Decomposition Method for Neural Multiclass Classification Problem

H. El Ayech, and A. Trabelsi

***Abstract*—**In this article we are going to discuss the improvement of the multi classes' classification problem using multi layer Perceptron. The considered approach consists in breaking down the n-class problem into two-classes' subproblems. The training of each two-class subproblem is made independently; as for the phase of test, we are going to confront a vector that we want to classify to all two classes' models, the elected class will be the strongest one that won't lose any competition with the other classes. Rates of recognition gotten with the multi class's approach by two-class's decomposition are clearly better that those gotten by the simple multi class's approach.

***Keywords*—**Artificial neural network, letter-recognition, Multi class Classification, Multi Layer Perceptron.

I. Introduction

THE improvement of training techniques has been a continuous process since the use of certain learning models, such as artificial neural networks, the hidden markov models, Bayesian networks and more lately the support vector machines etc... We are going to get especially interested in the artificial neural network and more precisely to the multi layer Perceptron (MLP) architecture. The improvement of the learning process through the MLP has taken several aspects.

Several works are directed towards the optimization of the model's architecture and its internal parameters as well as the training algorithms; among these works we find the genetic algorithms that consist in finding the appropriated architecture for a problem given among others by modifying the model's configuration. Other aspects indicate the importance of the pre processing phase by trying to valorise the good choice of characteristic vectors that presents at best the data of a problem.

In general the neural networks are flexible models that permit to treat the multiclasses problems in a direct way, however the SVM (support vector machines) models can not treat this task in a straight way as they are two-classes models by necessity, but it can approach the multiclasses problems by virtue of some techniques of which we inspire the idea for this paper. Among the techniques adopted for the SVMs, we find the technique one against all formulated by Vapnik [1] and one against one, called pairwise classification again [2].

Manuscript received August 31, 2006.

Authors are with the Business & Economic STatistics MODeling Laboratory (BESTMOD), 41, Rue de la liberté Cité Bouchoucha, 2000, Le Bardo, Tunis, Tunisia (e-mail: hafedh.elayech@edunet.tn, Abdel.trabelsi@isg.rnu.tn).

In this paper we will try to improve the learning multi classes' process by the choice of the decomposition strategy in two classes's subproblems.

In the first section we will present the multi classes' learning problem, the second section will be dedicated to the presentation of the decomposition method. The third section will show the results gotten with a comparative survey.

II. Problem of Multi Classes Classification

A. The Architecture of Neural Network

A neural network is an oriented and weighted graph. The nodes of this graph are simple automatons named formal neurons possessing an internal state that represents its activation and by which they influence the other neurons of the network. This activity propagates itself in the graph along weighted bows called synaptic links (or weights).

The state of the whole network is composed of its neurons' activation, and by the matrixes of the synaptic weights joining a layer to the following, each one is a matrix in which we find the weights of the links, the answer o_j of a neuron number j is given like follows:

$$o_j = f(net_j) = f(\sum_{i=1}^{n} x_i w_{ij}) \quad (1)$$

$$\text{with } f(x) = \frac{1}{1+e^{-x}} \quad (2)$$

We can categorize the neurons of the network in three sets: I, O and H. The I set corresponds to the input neurons. The input neurons are the neurons that receive the data of the problem [3]. The O set is the set of the output neurons: it is a subset of neurons whose activation will be interpreted as the answer of the network. The H set is the set of the hidden neurons. Hidden neurons are those that are not present in the input nor in the output of the network. The hidden neurons don't have any direct links with the outside and act therefore through other neurons. A network having hidden neurons, is often more powerful than a network without hidden neurons.

Although a variety of architectures and of connections' topologies exists, we are essentially interested in the multi layered architecture named Multi Layer Perceptron (MLP) and in the generalized structure called non recurrent feed-forward. Neurons are arranged by layers. There is no connection between neurons of a same layer and connections settle only with the neurons of the following layer. Every neuron of a layer is connected to all neurons of the following layer and to only this one, although we can find, in the generalized networks, some connections that can directly jump from the

input layer to the output one. We will adopt the following notation to designate the architecture of a MLP: MLP($nlayer_1$, $nlayer_2$,....., $nlayer_i$,... $nlayer_l$), $nlayer_i$ is the number of neurons for the layer number i.

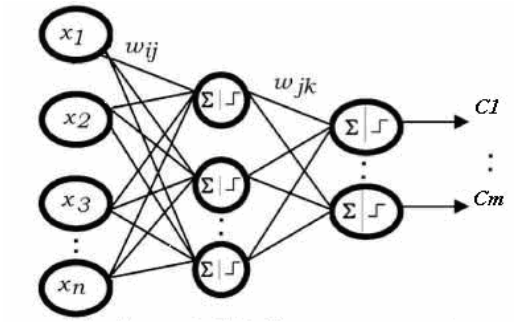

Fig. 1 MLP with 3 layers

B. Multi Classes Classification with the Multi Layer Perceptron

We approach a supervised training's problem (where the network must know the answer that it should have given) that consists in constructing (or in estimating) a function $f(x)$ from the observed data $(x_1,y_1),...,(x_n,y_n)$, so that $f(x)$ is a good approximation of the desired answer Y.

We often choose f in order to minimize the sum of the quadratic errors (in this case, it is the cost function or Objective function)

$$F = \sum_t (Y_t - f(X_t))^2 \quad (3)$$

The back propagation's algorithm based on the technique of the descent gradient is well adapted to the resolution of this problem. The task of this type of application is aiming to associate the label of a known pattern to a given input vector. The input neurons can be real or binary values; as for the output neurons that present the classes, the desired values are often strictly binary (the answers given by the network are contained between 0 and 1). When we present an input vector (representative of features) in the training phase, we want to have only one unit (class) that will have the value of 1 and the other units (classes) will have the value of 0 at the level of the output layer. Once the network is trained (the values of synapses already adjusted), when we want to recognize the class of an input vector of feature, we are going to have real values in the interval [0,1] (it depends on the function of transfer at the level of the output layer : in this case, it is the sigmoid function), the likeliest class is the one that includes the value the more close to 1.

C. Learning Process of a MLP

Let's recall that it is about reaching optimal weight values that minimize the objective function, for this reason we use the back propagation's algorithm that requires an architecture having at least one hidden layer; moreover, the function of transfer that transforms the activation in answer at the level of a hidden layer must be non linear.

The MLP proceeds by a training with known examples. The set of vectors of training is defined by $\{(x_p,y_p)|p=1,2,...,P\}$, the vector x_p is presented through the input layer. The network will calculate its output vector after the passage of the stimulus through the hidden layers of the network, the vector op is compared to the one that we want to get yp, and we deduce the vector of errors y_p - o_p. The criteria that we must minimize is the objective function (or cost function) that is the sum of the squared errors (SSE: Sum of Squared Errors). It is expressed like follows:

$$F = \sum_p^P F_p = \frac{1}{2}\sum_p^P \sum_k^N (y_{pk} - o_{pk})^2 \quad (4)$$

where p is the index of the training vector, o_{pk} is the calculated response of the neuron k corresponding to the vector p, P is the total number of training vectors and N is the number of output neurons.

The calculated error will be back propagated through the network, and the weights w_{ij} will be modified; the back propagation algorithm uses the technique of the gradient descent to minimize the distance between the wanted output and the output gotten by the network:

$$\Delta w_{ij} = -\eta \frac{\partial F_p}{\partial w_{ij}} \quad (5)$$

The updating formula of the (synaptic) weights will be:

$$w_{ij}^{p+1} = w_{ij}^p + \Delta w_{ij} \quad (6)$$

The back propagation algorithm can be summarized in the following steps after having fixed the architecture of the network (the number of layers, the number of neurons in every layer, the parameters of training as the choice of the transfer function for the neurons of the hidden layers and output layer as well as the parameter η called learning rate):

1) Initialize randomly the weights of the network and initialize the number of n iteration to zero.
2) Present firstly the input vectors from the training dataset in the network
3) Send the input vector p through the network to get an output

$$o_{pk} = f(\sum_j w_{jk} f(\sum_m w_{mj} f(....f\sum_i w_{il} x_i)))$$

4) Calculate a signal of errors between the real output and the desired output, calculate the sum squared errors (the objective function F to minimize) and increment n.
5) Send the error signal backwards through the network
   For every unit of output k, calculate, $\delta_k = (o_k - y_k) f'(net_k)$
   For every hidden unit j, calculate, $\delta_j = f'(net_j)\sum_k \delta_k w_{jk}$
6) Correct the weights to minimize the error by the following updating: $\Delta w_{ij}^{(n+1)} = \eta \delta_j o_i + \alpha \Delta w_{ij}^{(n)}$ where α is a constant called momentum which serves to improve the learning process.

7) Repeat the steps 2-6 with the next input vector until the error becomes sufficiently small or until a maximal number of iterations fixed in advance is reached.

III. Decomposition Towards Two Classes' Problems

A.. Learning Phase:

The idea is to explode a unique multi classes' problem in several two classes' problems. Instead of making learning for a unique MLP multi classes, we are going to make learn several two class MLPs, the number of MLP models gotten by this decomposition is given by the following formula:

$$total\_number\_of\_mlps = \frac{(number\_of\_classes \times (number\_of\_classes - 1))}{2}$$ For example**:** if we have a problem of classification to 4 classes, we will get 6 sub problems of classification to 2 classes.

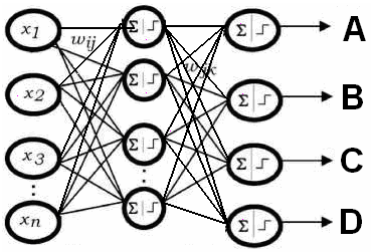

Fig. 2 The model initial to 4 classes

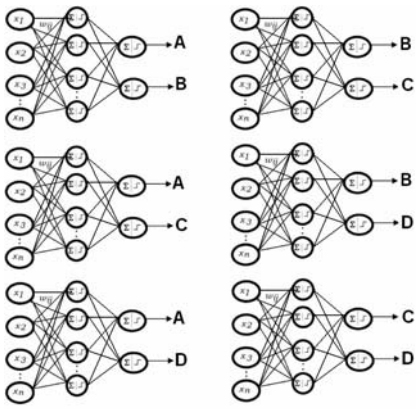

Fig. 3 The final model including 6 two class MLPs

B. Test Phase

The Principle of classification of a vector with two class competition:

At the end of the learning phase of all MLPS, our global model including the two-class MLP models becomes able to classify any characteristic vector presented in input. The basic idea is to confront a vector to classify to every two-class MLP. At every confrontation to every MLP (propagation of the signal in the network to get the response vector at the output), we will have one winning class and the other is a losing class. The winning class is the class that has had the most maximal value answer and the nearest to 1. After every competition we will add every winning class to a list of winners and every losing class to a list of losers. After having finished all competitions, that is to say after having confronted the input vector to all MLPs, we are going to eliminate all losing classes existing in the winners' list, it will remain only one class in the winners' list since it has not been found in the losers' list and it hasn't lost any competition. This class is the pattern that it should be recognized by the global model and that represents the good classification of the vector presented in input.

IV. Application to the Problem of Characters' Recognition

A. Description of Data

We used the famous training dataset used by Slate and Frey [4]; this dataset includes 20000 input vectors with their desired answers. Every input vector includes 16 attributes representing features of the picture that it should be recognized. We had the choice to do two types of normalization of values of input vectors. The goal of this normalization is to reduce the interval of the incoming absolute values between zero and one. If we want to normalize data towards the interval [0,1], we will use the following formula: $x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$

If we want to normalize data towards the interval [-1,1], we will use the formula:

$$x' = \frac{2x - x_{\max} - x_{\min}}{x_{\max} - x_{\min}}$$

x_{max} and x_{min} are respectively the most maximal and the smallest value in the training dataset.

We have 26 classes to recognize, this implies that we will have 325 two class MLP models to make learn.

B. Selection of the Architecture of Artificial Neural Network

We already have the possibility to decide the number of input and output neurons: the number of neurons in the input layer is the number of attributes of a characteristic vector; the number of neurons in the output layer is the classes' number of the problem. We haven't decided yet the choice of the number of hidden layers and the number of neurons for every hidden layer. We will tempt several possible configurations and we will choose one according to criteria of performance measure. The first criteria is the value of the objective function, the second is to test the network which is already trained on a sample of test and then we will note the percentage of successful recognition for every class and the average of recognition percentages for the different classes.

Finally we are also based on the Akaike information criteria and on the Bayesian information criteria known in the literature in order to value the performance of a model: the more these criteria give small values, the more the model is efficient.

$$AIC(p) = n \log\left(\frac{SSE}{n}\right) + 2p$$

$$BIC(p) = n \log\left(\frac{SSE}{n}\right) + p + p \log n$$

where p represents the number of parameters that it should be estimated (in our case, p represents the number of synapses connecting neurons between them), SSE is the sum of squared

errors and *n* corresponds to the number of observations used to adjust the model.

For the multi layer Perceptron, the number of *p* parameters is given as follows:

$$p = \sum_{i=1}^{l-1} (number\_of\_neurons_i \times number\_of\_neurons_{i+1})$$

where *l* is the total number of layers in the network.

C. Empirical Results

We used 15000 vectors for the training phase, the remaining 5000 will serve for the test. The criteria of stop of the training has been fixed to 500 epochs, the epoch term means the number of times that the training dataset has been completely browsed. Results are impressive and the ability of classification of our model is very powerful.

Results gotten by a simple multi class MLP are shown in the table 1, the average recognition rate is around 82,54%. The table 2 gives results gotten by the global model with bi class decomposition, the average recognition rate is about 90,35%. We notice that the global model with two class decomposition is distinctly better that a simple multi class MLP model. Here are the comparative results for both models after test.

TABLE I
RECOGNITION WITH SIMPLE MULTI CLASS MLP MODEL (16,26,26,26)

A 91,26	O 80,11
B 84,39	P 89,37
C 76,60	Q 72,35
D 81,48	R 88,94
E 84,81	S 76,26
F 86,59	T 80,97
G 72,59	U 89,30
H 58,42	V 83,92
I 85,36	W 90,41
J 90,71	X 89,24
K 80,22	Y 78,14
L 81,95	Z 76,28
M 92,51	**Average :**
N 83,83	**82,54%**

TABLE II
RECOGNITION WITH TWO CLASS COMPETITION OF 325 MLPS(16,16,16,2)

A 93,20 %	O 84,53 %
B 90,75 %	P 90,33 %
C 92,98 %	Q 94,0092%
D 90,27 %	R 74,51 %
E 90,05 %	S 94,94 %
F 93,81 %	T 91,84 %
G 82,21 %	U 93,48 %
H 80,33 %	V 91,07 %
I 90,73 %	W 97,005%
J 91,80 %	X 96,23 %
K 89,26 %	Y 91,25 %
L 88,29 %	Z 93,81 %
M 95,18 %	**Average :**
N 87,37 %	**90,35 %**

V. CONCLUSION AND PERSPECTIVES

We have studied the effect of decomposition of a learning multi classes' problem in sub two class problems on the performance of classification. We have obtained some encouraging results. This survey remains in the experimental frame, thus we should construct a theoretical frame proving and explaining why the method adopted in this paper has generated this considerable success.

REFERENCES

[1] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, London, UK,1995.

[2] U. H.-G. Kreßel. "Pairwise classification and support vector machines". In B. Scholkopf,C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 255–268. The MIT Press, Cambridge, MA, 1999.

[3] J. F. Jodouin, *Les réseaux de neurones, principes et définitions*. *Hermès*, 1994.

[4] P.W. Frey and D.J. Slate "Letter Recognition Using Holland-style Adaptive Classifiers". *Machine Learning Vol 6 #2,* 1991.

[5] H. White, "Connectionist nonparametric regression: Multilayered feedforward networks can learn arbitrary mapping", *Neural Networks*, 3, pp.535-550, 1990.

[6] Z. Tang, P. Fishwick "Feed-forward Neural Nets as Models for Time Series Forecasting", *ORSA Journal of computing* 5 (4) pp 374-386, 1993.

[7] W. CHEN, S. CHEN, C. LIN, "A speech recognition method based on the sequential multi-layer perceptrons" *Neural Networks,* Vol. 9, No. 4, pp. 655-669,1996.

[8] H. Abdi, *Les réseaux de neurones.* Sciences et technologies de la connaissance. Presse Universitaire de Grenoble, France, 1994.

[9] V. Kecman. *Learning and Soft Computing, Support Vector Machines, Neural Networks, and Fuzzy Logic Models.* England : The MIT Press, 2001.

[10] A. Cornuéjols, L. Miclet, Y. Kodratoff. *Apprentissage artificiel Concepts et algorithmes*. France : Editions Eyrolles, 2003.

[11] A. Shigeo "Analysis of Multiclass Support Vector Machines", Graduate School of Science and Technology Kobe University, Kobe, Japan, Available: www2.kobe-u.ac.jp/~abe/pdf/cimca2003.pdf

[12] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. "Large margin DAGs for multiclass classification". *In S. A. Solla, T. K. Leen, and K.-R.M¨uller, editors, Advances in Neural InformationProcessing Systems 12, pages 547–553*. The MIT Press, 2000.