The Labeled Classification and its Application

M. Nemissi, H. Seridi, and H. Akdag

Abstract—This paper presents and evaluates a new classification method that aims to improve classifiers performances and speed up their training process. The proposed approach, called labeled classification, seeks to improve convergence of the BP (Back propagation) algorithm through the addition of an extra feature (labels) to all training examples. To classify every new example, tests will be carried out each label. The simplicity of implementation is the main advantage of this approach because no modifications are required in the training algorithms. Therefore, it can be used with other techniques of acceleration and stabilization. In this work, two models of the labeled classification are proposed: the LMLP (Labeled Multi Layered Perceptron) and the LNFC (Labeled Neuro Fuzzy Classifier). These models are tested using Iris, wine, texture and human thigh databases to evaluate their performances.


I. INTRODUCTION

In the past few years, ANNs (Artificial Neural Networks) have been widely used in several application of the pattern recognition. They have been employed as powerful classifiers for the reason of their capacities of learning and generalizing. But among their disadvantage is the slowness of their training process. Avoiding this problem is the main goal of the labeled classification.

On the other hand, ANNs and FIS (Fuzzy Inference Systems) are appropriate to describe complex systems where it is difficult to give mathematical description. Moreover, a combination of these complementary methods allows having more robust systems [1] [2]. For example, ANNs are not interpretable, so they are not able to represent knowledge explicitly while a fuzzy system can do it by fuzzy if-then rules [1]. Furthermore, implementation of FIS necessitates tuning of the membership function parameters that can be automatically updated in the case of Neuro-Fuzzy Systems [2].

The BP (Back Propagation) algorithm [3] is a useful algorithm in many applications. It is widely used for training the ANNs and the Neuro-Fuzzy Classifiers, but its convergence rate is relatively slow [4] and it is an unreliable algorithm [5] [6]. In order to improve performances of the BP, several approach have been proposed. Jacob et al. [7] presented an approach based on the use of a different step gains for each weight and the updating of theses term iteratively. Li et al. [8], Russo [9], Nguyen et al. [10] and Yam et al. [11] presented methods based on the weight initialization, Lee et al. [12] studied the effect of initial weight on premature saturation and Kamarthi et al. [13] introduced an algorithm based on extrapolation of each weight to accelerate the BP algorithm. Zurada [14] and Chandra et al. [15] proposed methods which are based on the adapting of the activation functions parameters. Rumelhart et al. [3] added an extra term (the momentum) and Zweiri et al. [4] [16] introduced a third term (proportional factor) in addition to the learning rate and the momentum. The problem of the local minima was studied by Ampazis et al. [17], Phansalkar et al. [18] and Vitela et al. [19]. Cho et al. [20] proposed an approach based on the least-squares method to improve the BP convergence and Wang et al. [21] proposed a modified error function by adding one term to the conventional function.

The labeled classification approach proposed in this work is different from these methods in the sense that our proposition aims to improve the training process by changing the representation of examples instead of modifying the training algorithm. Indeed, this method seeks to speed up the BP convergence by adding an extra feature (labels).

Our approach can be used with several models trained by the BP, and it can be used with different techniques of acceleration and stabilization. In this work, two models of the labeled classification are proposed: LMLP and LNFC. The first model is based on the use of a neural architecture while the second is based on the use of Neuro-Fuzzy architecture. In section 2, we describe the proposed approach. In sections 3, we present the first model and we discuss its classification

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>ADVANTAGES AND DISADVANTAGES OF ANN AND FIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advantage</td>
<td>Disadvantage</td>
</tr>
<tr>
<td>ANN</td>
<td>Parallel computing</td>
</tr>
<tr>
<td></td>
<td>Capacity of generalization</td>
</tr>
<tr>
<td>Self-adaptation</td>
<td></td>
</tr>
<tr>
<td>FIS</td>
<td>Possibility to use a prior knowledge</td>
</tr>
</tbody>
</table>
performances on Iris, wine, human thigh and texture databases. The second model and its performances are presented in section 4. Finally, conclusions are given in section 5.

II. THE LABELED CLASSIFICATION

An important reason of the BP slowness is the saturation behavior of the activation function used in different layer [4]. In fact, when a sigmoid has a slope near a zero, a weight point may enter the saturation region of the weight space [5]. In such situation, the weight increment remains little even if the error is relatively large. The basic idea of our approach is to make the training example linearly separable by adding an extra feature (labels). The labels must be identical for examples belonging to the same class to ensure the linear separation and reduce the possibility of entering the saturation region. After the training process, every test example will be classified according to the classifier’s outputs with all labels (Fig. 1).

Recall that a conventional classifier maps any input vector \(X(x_1, x_2, ... x_N)\) into an output vector \(Z(z_1, z_2, ... z_K)\) corresponding to the class \(C_i\) of the example represented by \(X\). Fig. 1 shows a labeled classifier with two input and three classes. It is based on the adding of labels at the input of the used classifier and giving a decision according to the classifier outputs.

A. Methodology

The labeled classification is performed in two stages:

1. Addition of labels for all training examples and performing training of the used classifier (according to two modes).
2. Carrying out tests with these labels to classify every novel example (Fig. 2).

Therefore, for each class \(C_i\), corresponds a label \(L_i\). Every training example \(X(x_1, x_2, ... x_N)\) of \(C_i\) is represented by \(X(x_1, x_2, ... x_N, L_i)\). The representations of all training examples are modified by the same manner. After the training process, every new example \((X)\) will be tested with all labels and it will be classified according to the following decision rule:

\[X \in C_i \text{ if } E_{r_1}(X) = \min \{E_{r_1}(X), E_{r_2}(X), ... E_{r_K}(X)\}\]

where \(E_{r_i}\) is the sum-squared error between the target \(T_i\) corresponding to the class \(C_i\) and the calculated output \(Z_i\) using the label \(L_i\). \(E_{r_i}\) is defined by:

\[E_{r_i} = \|T_i - Z_i\|\]

B. Training in the Labeled Classification

The labeled classification is used with the classifiers trained by the BP for the reason that our decision rule is based on the sum-squared-error between target and classifier output. Training in the labeled classification is performed using two modes:

1) First Mode: Simple Training
The first mode consists in carrying out the training normally. No modifications are involved in the algorithm and the cost function is the total sum-squared-error, which is given by:

\[TSSS = \sum_{q=1}^{Q} \|T^{(q)} - Z^{(q)}\|\]

where \(T^{(q)}\) and \(Z^{(q)}\) are the target and the classifier output of the \(q^{th}\) example.
2) Second Mode: Full Training
In the second mode, the training is performed by minimizing the sum-squared errors between the target and the classifier outputs obtained with each label (Fig. 3). That is to say, for every presented example, the classifier must be simulated and updated for all labels. The cost function becomes:

\[ E = \sum_{q=1}^{Q} \sum_{i=1}^{K} \left\| T^{(q)} - Z_i^{(q)} \right\| \]

where \( T^{(q)} \) is the target and \( Z_i^{(q)} \) is the calculated output of the \( q \)th example using the label \( L_i \).

Fig. 4 represents a LMLP with \( N \) neurons at the input layer, \( M \) neurons at the hidden layer and \( J \) neurons at the output layer.

The second part of the above equation shows the consequence of adding labels at the network outputs.

\[ y_m = h\left( \sum_{n=1}^{N} x_n W_{nm} + L_i w_{N+1,m} \right) \]

The \( j \)th network output becomes:

\[ z_j = g\left( \sum_{m=1}^{M} h\left( \sum_{n=1}^{N} x_n W_{nm} \right) u_{mj} \right) + g\left( \sum_{m=1}^{M} L_i w_{N+1,m} u_{mj} \right) \]

where \( z_j \) is the output of the \( j \)th neurons, \( x_n \) is the \( n \)th input, \( h \) and \( g \) are the activation function (sigmoid). The second part of the above equation shows the consequence of adding labels at the network outputs.

B. LMLP Training

1) Simple Training
In this mode, labels are treated as the other features. The LMLP is trained as any conventional MLP and the BP is used without any change. The adaptation task is to minimize the total sum-squared error (TSSE) between the classifier outputs and the targets.

2) Full Training
In this mode, for every presented example, the network outputs must be determined with all labels. The weights are adjusted according to these outputs. The function cost
becomes the sum-squared errors between targets and the classifier outputs obtained with different labels. Therefore, at
every iteration, a training example \( X(q) \) is presented to the classifier with a label \( L_i \). The adaptation task is to minimize
the partial sum-squared error (\( PE \)) between the classifier output and the target.

\[
PE = \| T^{(q)} - Z_i^{(q)} \|
\]

where \( T^{(q)} \) is the target and \( Z_i^{(q)} \) is the calculated output of the \( q \)-th example using the label \( L_i \).

C. Choice of Labels

In this process, the choice of labels is very important because they influence directly the classification
performances. We suggest choosing values around 0.5 with a small deference (\( \delta \)) between them. For example, in the case of
three classes: \( L_1=0.5- \delta \), \( L_2=0.5 \) and \( L_3=0.5+ \delta \).

D. Iris Database Classification

To appreciate the proposed model, tests are carried out on the Iris database using an MLP and a LMLP. The classification performances of this database using MLP depend strongly on the initial weight. In some cases, the weight point enters the saturation region and the MLP makes a large number of iterations to escape from this region, or escape may never be achieved. An example of such situation is presented in Fig. 5. This figure indicates the evolution of the classification rate during the training stage of an MLP and a LMLP. Both models have the same architecture (8 hidden neurons), they are initialized by the same weights and the training parameters are the same (step gains and momentum). The used labels are \( L_1=0.475 \), \( L_2=0.5 \) and \( L_3=0.525 \). So, with \( \delta=0.025 \), LMLP1 denotes LMLP with simple training and LMLP2 denotes LMLP with full training. The graphs showed on this figure indicate the improvements obtained by the labeled classification. It allows obtaining a classification rate equal to 98 % after less than 200 iterations while the MLP permits obtaining this rate after more than 750 iterations.

E. Wine Database Classification

This database was obtained after a chemical analysis of wines grown in the same region but derived from three
different cultivars. It contains 178 examples with 13 features belonging to 3 classes. The first one has 59 examples, the second has 71 and the third has 48. We utilize the training data-itself-as-testing method.

Fig. 8 indicates the evolution of the classification rate during the training stage of an MLP and a LMLP. Both models have the same architecture (6 hidden neurons), initialization weights and parameters. The used labels are
$L_1=0.49$, $L_2=0.5$ and $L_3=0.51$. So, with $\delta=0.01$, LMLP1 denotes LMLP with simple training and LMLP2 denotes LMLP with full training. The graphs showed on this figure indicate the improvements obtained by the labeled classification (with full training). It allows obtaining a classification rate equal to 100% after 35 iterations while the MLP permits obtaining this rate after 50 iterations.

![Fig. 8 Wine classification using the MLP and the LMLP](image)

Fig. 8 Wine classification using the MLP and the LMLP

Fig. 9 shows the effect of labels in the case of the simple training. Graph A corresponds to $\delta=0.05$ ($L_1=0.45$, $L_2=0.5$ and $L_3=0.55$), graph B corresponds to $\delta=0.025$ and graph C corresponds to $\delta=0.01$. We can note that the LMLP with simple training gives acceptable results for $\delta \leq 0.025$.

Fig. 10 shows the effect of labels in the full training. The LMLP gives the same results for these different labels.

![Fig. 10 Wine classification using the LMLP (full training) with different labels](image)

Fig. 10 Wine classification using the LMLP (full training) with different labels

F. Human Thigh Database Classification

The image of Fig. 11 is acquired by cryosection color photography. A manual classification was made by an expert and four components were identified (grease, bone, marrow and muscle). Each one of these components corresponds to a class and a file of 300 pixels representing each one. The obtained sample consists of 1200 pixels, 300 pixels of each class. The addition of components X and Y (to locate geometrical position of a pixel and to take account of its vicinity) improves the classification performances.

To evaluate generalization capacities of our model, a cross validation of order 4 is used. Four datasets are obtained. Each base contains 900 training examples and 300 test examples.

![Fig. 11 Image of human thigh cryosection](image)

Fig. 11 Image of human thigh cryosection

As in [23], the used MLP is composed of 5 neurons at the input, 8 hidden neurons and 4 outputs neurons. The MLP and the LMLP are initialized by the same weights. In table (2), the obtained results are showed. This results are the average of the 4 data sets obtained by the cross validation.
TABLE II

RESULTS OF THE HUMANHTIGH CLASSIFICATION USING MLP AND LMLP

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Labels</th>
<th>Classification rate (Test datasets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td></td>
<td>98.17</td>
</tr>
<tr>
<td>LMLP (Simple training)</td>
<td>0.425 0.475 0.525 0.575 (δ = 0.050)</td>
<td>97.83</td>
</tr>
<tr>
<td>LMLP (Full training)</td>
<td>0.485 0.495 0.505 0.515 (δ = 0.010)</td>
<td>97.92</td>
</tr>
</tbody>
</table>

G. Texture Database Classification

Fig. 12 shows an image constituted of two different microtextures. A pretreatment (calculation of different local correlations) of the initial image allows obtaining a series of 8 images. Each one is the detection result of a particular attribute. Every pixel is then described by a vector of 8 attributes.

Fig. 12 Image of texture

Thus, the image is represented by two classes, which are described by eight files containing each one the pixel values. The obtained sample consists of 400 pixels of each class. A cross validation of order 4 allows obtaining four training datasets containing each one 600 pixels and 4 test datasets of 200 pixels for each one.

For the comparison, the used MLP is composed of 5 neurons at the input, 8 hidden neurons and 2 outputs neurons. Table III indicates the obtained results.

IV. THE LNFC

A. Presentation

The implementation of Neuro-Fuzzy Systems aims to combine proprieties and advantages of ANNs and FIS. In these systems, every layer of ANN performs a different function of a FIS: Fuzzification, Inference and Defuzzification. The NFCs (Neuro-Fuzzy Classifiers) [1] have a Neuro-Fuzzy architecture that can incorporate in its structure fuzzy if-then rule of the form:

If \( x_1 \) is ‘small’ and \( x_2 \) is ‘big’ then \( X \) belongs to \( C_k \)

The conception of the LNFC aims to exploit and improve proprieties of the NFCs. The use of LNFC leads to replace the above rules by rules of the form [24]:

If \( x_1 \) is ‘small’ and \( x_2 \) is ‘big’ and its label is \( L_1 \) then this example belongs to \( C_k \)

B. Architecture

The labeled classification consists essentially in adding labels to all training examples. Consequently, a neuron is added at the first layer and \( K \) neurons at the second (\( K \) is the number of classes). Every neuron added to the second layer corresponds to the membership function of a label.

Fig. 13 shows an example of LNFC with two input variables \( (x_1, x_2) \) and two output variables \( (z_1, z_2) \). Every input is represented by two linguistics variables. In the first layer, every neuron corresponds to a linguistic variable. Neurons of the second layer send the product of the incoming signals and every neuron of the third layer corresponds to a class. The output of the \( m^{th} \) output of the third layer is:

\[
y_m = \prod_{n=1}^{N} \mu_{mn}(x_n)
\]

The \( j^{th} \) network output is:

\[
z_j = g\left( \sum_{m=1}^{M} \left( \prod_{n=1}^{N} \mu_{mn}(x_n) \right) w_{mj} \right)
\]

where \( \mu_{mn} \) is the \( m^{th} \) membership function, \( x_n \) is the \( n^{th} \) input and \( w_{mj} \) is the weight between the \( m^{th} \) hidden neuron and the \( j^{th} \) output neuron. In the version of Jang [2], he used sigmoid function at the output layer. Using his model, the \( j^{th} \) network output becomes:

\[
z_j = g \left( \sum_{m=1}^{M} \left( \prod_{n=1}^{N} \mu_{mn}(x_n) \right) w_{mj} \right)
\]

where \( g \) is the activation function of the output layer.
C. Training

The back propagation algorithm is widely used for neuro-fuzzy systems training, for example in [1] [24] [25] [26]. The adaptation task is to minimize the total sum-squared error between the classifier outputs and the targets. Our model is trained using this algorithm. The training of LNFC is performed without tuning the membership functions, which allows:

1. Obtaining a simplified training process because only the output weights are updated.
2. Keeping the original linguistic meaning of the membership functions.
3. Changing the T-norm operator (used in neurons of the third layer) without changing the training algorithm.
4. Changing the membership functions without changing the training algorithm.

As for the LMLP, the training can be performed using two modes: simple training and full training. In both cases, the update expression at the \((i+1)\) iteration of \(u_{mj}\) is:

\[
u_{mj}^{(i+1)} = u_{mj}^{(i)} + \eta_i \left(t_j - z_j\right) g'(z_j) y_m\]

where \(\eta_i\) is the step gain, \(t_j\) is the \(j^{th}\) component of the target, \(z_j\) is the \(j^{th}\) output and \(g'\) is the derivate of \(g\) (activation function of the output layer).

D. Choice of Labels

The premises of the Fuzzy rules established by the third layer depend on the membership functions of labels instead by the labels themselves. That is to say, contrary to the case of the LMLP where the choice of labels values influences directly the classification performances.

E. Iris Database Classification

To appreciate the LNFC, it is compared with a conventional NFC using Iris database. In both cases, three linguistic variables are used for the fuzzification (Fig. 14).

Fig. 15 shows the evolution of the classification rate during the training stage of the NFC and the LNFC. Both models are initialized by the same weights and the training parameters are the same. The used membership function of labels are \(\mu_i(Li)=1\) and \(\mu_i(Lj)=0.85\). This figure indicates the improvements obtained by the labeled classification. It allows obtaining a classification rate equal to 99.33 % after 60 iterations using LNFC with full training and 98.67 using LNFC with simple training while the NFC permits obtaining this rate after 80 iterations.

Fig. 16 shows the effect of the membership function of labels in the case of the simple training. Graph A corresponds to \(\mu_i(Li)=1\) and \(\mu_i(Lj)=0.9\), graph B corresponds to \(\mu_i(Li)=1\) and \(\mu_i(Lj)=0.8\) and graph C corresponds to \(\mu_i(Li)=1\) and

Fig. 15 Iris classification using the NFC and the LNFC

Fig. 16 shows the effect of the membership function of labels in the case of the simple training. Graph A corresponds to \(\mu_i(Li)=1\) and \(\mu_i(Lj)=0.9\), graph B corresponds to \(\mu_i(Li)=1\) and \(\mu_i(Lj)=0.8\) and graph C corresponds to \(\mu_i(Li)=1\) and
\[ \mu(L_j) = 0.7. \] We can note that the LNFC with simple training gives an acceptable classification rate (98.67\%) for \( \mu(L_j) \geq 0.8. \)

On the other hand, Fig. 17 indicates the effect of labels in the case of the full training: the LNFC allows obtaining a classification rate equal to 99.33 \% for \( \mu(L_j) \geq 0.8. \)

F. Wine Database Classification

The LNFC is compared with NCF using wine database. In both case, two linguistic variables are used for the fuzzification (Fig. 18).

The evolution of the classification rate during the training stage of the NFC and the LNFC are showed in Fig. 19. In both case the initial weights and the training parameters are the same. The used membership function of labels are \( \mu(L_i) = 1 \) and \( \mu(L_j) = 0.85. \) The graphs showed on this figure indicate the improvements obtained by the labeled classification. It allows obtaining a classification rate equal to 100 \% after 10 iterations using LNFC with full training and after 40 iterations using LNFC with simple training while the NFC permits obtaining this rate after 80 iterations.

Fig. 20 shows the effect of the membership function of labels in the case of the simple training. Graph A corresponds to \( \mu(L_i) = 1 \) and \( \mu(L_j) = 0.9, \) graph B corresponds to \( \mu(L_i) = 1 \) and \( \mu(L_j) = 0.8 \) and graph C corresponds to \( \mu(L_i) = 1 \) and
\[ \mu(L_j) = 0.7. \] We can note that the LNFC with simple training gives acceptable results for \( \mu(L_j) \geq 0.8 \).

On the other hand, Fig. 21 indicates the effect of labels in the full training; the LNFC gives the same results for these labels.

**G. Human Thigh Database Classification**

The NFC and the LNFC are tested using human thigh database. Three linguistic variables are used for the fuzzification. Table IV illustrates the obtained results. This results are the average of the 4 data sets obtained by the cross validation.

**TABLE IV**

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Labels</th>
<th>Classification rate (Test datasets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFC</td>
<td>( \mu(L_i) = 1 ), ( \mu(L_j) = 0.7 )</td>
<td>97.92</td>
</tr>
<tr>
<td>LNFC (simple training)</td>
<td>( \mu(L_i) = 1 ), ( \mu(L_j) = 0.9 )</td>
<td>98.08</td>
</tr>
<tr>
<td>LNFC (full training)</td>
<td>( \mu(L_i) = 1 ), ( \mu(L_j) = 0.7 )</td>
<td>98.08</td>
</tr>
<tr>
<td>LNFC (full training)</td>
<td>( \mu(L_i) = 1 ), ( \mu(L_j) = 0.9 )</td>
<td>98.08</td>
</tr>
</tbody>
</table>

**H. Texture Database Classification**

The NFC and the LNFC are also evaluated using texture database. Two linguistic variables are used for the fuzzification. The obtained results are showed on Table V.

**TABLE V**

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Labels</th>
<th>Classification rate (Test datasets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFC</td>
<td>( \mu(L_i) = 1 ), ( \mu(L_j) = 0.8 )</td>
<td>99.13</td>
</tr>
<tr>
<td>LNFC (simple training)</td>
<td>( \mu(L_i) = 1 ), ( \mu(L_j) = 0.9 )</td>
<td>99.25</td>
</tr>
<tr>
<td>LNFC (full training)</td>
<td>( \mu(L_i) = 1 ), ( \mu(L_j) = 0.7 )</td>
<td>99.25</td>
</tr>
<tr>
<td>LNFC (full training)</td>
<td>( \mu(L_i) = 1 ), ( \mu(L_j) = 0.9 )</td>
<td>99.25</td>
</tr>
</tbody>
</table>

**V. CONCLUSION**

In this paper, a new classification method is presented. Two models obtained by the use of this method are proposed: the labeled MLP and the labeled NFC. To evaluate the performances established by this development, the LMLP and the LNFC are compared respectively with conventional NFC and MLP.

Four databases are used for evaluation of these networks. Therefore, our models are examined by different type of features: length and width measure (in Iris database), pixel value and their position (in the human thigh database), local correlations components (in the texture database) and chemical features (in the wine database).

The obtained results on these databases show that the proposed approach improve performances of the MLP and the NFC except in the case of human thigh classification using MLP.

The NFC is more stable than the MLP. The LNFC provides also this property because its conception does not require any modification in the structure and the training algorithm of the NFC.

The training of the LNFC is performed without modifying the membership functions parameters, which leads obtaining simple training process and not loosing the original linguistic meaning of the membership functions. We can also change the T-norme operator (used in neurons of the third layer) and the membership functions without modifying the training algorithm.

The training in the proposed approach can be performed using two modes: the simple training and the full training.
According to our experiments, we can note that the first one is more simple but it depends strongly on labels values while the second provides more flexibility in the choice of labels but its training process is relatively complex.

REFERENCES


