A New Fuzzy Learning Scheme
for Competitive Neural Networks

Mohammed Madiafi
Modeling and Instrumentation Laboratory
Hassan II Mohammedia-Casablanca University, BP.7955 Sidi Othmane
Casablanca, 20702, Morocco
madiafi.med@gmail.com

Abdelaziz Bouroumi
Modeling and Instrumentation Laboratory
Hassan II Mohammedia-Casablanca University, BP.7955 Sidi Othmane
Casablanca, 20702, Morocco
a.bouroumi@gmail.com

Abstract
To improve the efficiency of competitive neural network in solving clustering problems, the outline of a new scheme of training is presented in this paper. This technique is based on an unsupervised fuzzy competitive learning, that’s why we name it fuzzy competitive learning or FCL. Results provided by the proposed technique are compared with those obtained by other well known techniques such as LVQ, GLVQ, FLVQ and FCM.

Mathematics Subject Classification: 68T10

Keywords: Pattern recognition, Classification, Neural networks, Unsupervised learning, Fuzzy logic

1 Introduction
The design of an artificial neural network (ANN) based solution to a hard problem requires three steps: 1) the choice of an architecture for the structure of the network, i.e., the number of neurons to use and the way to interconnect them; 2) the choice of a suitable learning algorithm, that means a way to adjust, using examples, the different synaptic connections of neurons in order
to make the network able to achieve the special task for which it is designed; and 3) the choice of a learning database, that means a set of examples to use as input data for the learning algorithm. In this paper, we are interested in step 2 and our work consists in designing a new learning algorithm for competitive neural networks (CNN), which use a particular structure and unlabelled data as learning examples.

The structure of a CNN is fairly simple and composed of two layers: an input layer for receiving data examples of the learning base, and an output layer, or competitive layer, whose neurons represent the different classes supposed present in the learning database by the mean of their synaptic weight vectors [2]. In practice, CNN are used as prototype-generator classifiers and can be very useful in applications where each class or cluster can be represented by its prototype. CNN can be trained using an unsupervised learning mode and an unlabelled data which serve as inputs. Furthermore, no standard learning algorithm exists for this category of ANN and one of the difficulties that can be encountered in applying them is the choice or the design of a suitable learning algorithm.

Many learning techniques have been proposed in the literature. The first one, called learning vector quantization (LVQ), was proposed in 1989 by Kohonen [1,3,4,5]. For each object vector presented at the input layer, LVQ determines a unique neuron of the output layer, called the winner, whose synaptic weights should be adjusted. This is done by minimizing the distance between the synaptic weights vectors of the output layer and the input vector. LVQ suffers from some drawbacks such as the risk that a same neuron dominates the competition and always wins it, and the sensibility to the initialization protocol of the learning process. GLVQ (Generalized Learning Vector Quantization) is a generalization of LVQ that dates back to 1991 [7,8,9,10,11]. This generalization consists in updating not only the winner but all neurons of the output layer using a rule that takes into account the distance of each neuron to the input object. GLVQ gives the same importance to all non-winner neurons and may converge to contradictory situations where non-winner neurons have more importance than the winner (Gonzalez et al., 1995; Karayiannis et al., 1996). Fuzzy Learning Vector Quantization (FLVQ) is a fuzzy generalization of LVQ that was proposed by Tsao in 1994 [12,13,14]. It is a fuzzy unsupervised learning scheme that can be viewed as neural version of the famous algorithm Fuzzy C-Means (FCM) [15,16]. FLVQ consists in iteratively updating the synaptic weights of each neuron according to the membership degrees of input objects to the classes represented by that neuron. FLVQ requires prior availability of all elements of the learning database and cannot be used online, i.e., in situations where learning data are not all available before starting the learning process. FLVQ can also be costly in terms of time processing, especially for large learning databases.
In this paper, we propose a new fuzzy learning technique, called Fuzzy Competitive Learning (FCL), which tries to overcome the main drawbacks of the previous techniques. FCL can be viewed as an intermediary between LVQ and FLVQ in the sense that at each step of the learning process a number of winners is determined that can vary between 1 and the total number of classes. This number is determined using a new parameter we introduced in order to model the difficulty degree of the competition. Initially, this degree is small, which means that all neurons can win it; but as the learning process progresses competition becomes harder and harder causing a decrease of the number of winners. More details of this technique are given in section 4. Examples of results of its application to test data are presented and discussed in section 5, whilst sections 2 and 3 recall, respectively, the architecture of CNN and their first learning algorithm, LVQ. For detailed description of other algorithms we invite the reader to consult the corresponding bibliography.

2 Competitive Neural Networks

Competitive neural networks constitute a particular class of ANN. They are commonly used for solving hard real-world problems such as pattern classification and recognition, image compressing, etc.

![Figure 1: Competitive Neural Network](image)

CNN possess a two-layer architecture. The first layer is composed of \( p \) neurons with \( p \) denoting the number of features per object, i.e., the dimension of the data space. It is an input layer whose role is to receive the \( p \)-dimensional object vectors representing the \( n \) examples of the learning base. The second layer, or output layer, contains \( c \) neurons where \( c \) is the number of classes supposed present in the learning base. The \( p \) synaptic weights of each of these neurons represent the components of vector prototype of a class (figure 1).
As mentioned in the introduction, LVQ was the first algorithm used to train this kind of ANN. LVQ exists in the form of different versions called LVQ1, LVQ2, LVQ3, and LVQ4 [6]. The three first versions use a supervised learning mode, which needs that the data examples are labeled, and the last version uses an unsupervised learning mode for which examples are unlabeled. In the next section we give a more detailed description of different variants of LVQ.

3 Learning Algorithms for CNN

3.1 Learning Vector Quantization (LVQ)

LVQ is an unsupervised learning algorithm aimed at training competitive neural networks. It is based on the idea of competition in the sense that, at each iteration, the \( c \) neurons of the output layer compete for the input sample and only one neuron, the winner, benefits from the adjustment of its synaptic weights. Hence, for each object vector \( x_i = \{x_{i1}, x_{i2}, \ldots, x_{ip}\} \in \mathbb{R}^p \) presented to the network, we locate the neuron \( j \) whose synaptic weights vector \( v_j = \{v_{j1}, v_{j2}, \ldots, v_{jp}\} \in \mathbb{R}^p \) minimizes the distance. This vector is then updated according to the rule:

\[
v_{j,t} = v_{j,t-1} + \eta_{t-1}(x_i - v_{j,t-1})
\]

\( \eta_{t-1} \) is the learning rate which serves to control the convergence of synaptic weights vectors to class prototypes. Starting from an initial value \( \eta_0 \), \( \eta_{t-1} \) is updated, at each iteration \( t \), according to the relation:

\[
\eta_t = \eta_0(1 - \frac{t}{t_{max}})
\]

This operation is repeated until stabilization of synaptic weights vectors or until a maximum number of iterations is reached.

As mentioned before, LVQ suffers from some drawbacks such as its sensitivity to the initialization, the risk of a dominant that always wins the competition, and a bad exploitation of the structural information carried by each data example. Indeed this information is not limited to the distance between the data example and the winner but distributed over the distances to all the \( c \) neurons. To overcome these drawbacks several techniques have been proposed in the literature. The earliest technique was a generalization of LVQ known under the noun: Generalized Learning Vector Quantization (GLVQ).
3.2 Generalized LVQ (GLVQ)

Proposed by Pal, GLVQ is an optimization procedure that tries to minimize the following criterion:

\[ J_i = \sum_{j=1}^{c} \alpha_{ij} \| x_i - v_{j,t-1} \|^2 \]  

with

\[ \alpha_{ij} = \begin{cases} 
1 & \text{if } j = \arg \left( \min_{1 \leq r \leq c} \| x_i - v_{r,t-1} \| \right) \\
\frac{1}{\sum_{r=1}^{c} \| x_i - v_{r,t-1} \|^2} = \frac{1}{D} & \text{otherwise}
\end{cases} \]

This is done by updating the synaptic weights of all neurons of the output layer using the rule:

\[ v_{j,t} = v_{j,t-1} + \eta_{t-1} \frac{\partial J_i}{\partial v_{j,t-1}} \]

that is:

\[ v_{j,t} = v_{j,t-1} + \eta_{t-1} (x_i - v_{j,t-1}) \times \psi_{ij} \]

\[ \psi_{ij} = \begin{cases} 
\frac{D^2 - D + \| x_i - v_{j,t-1} \|^2}{D^2} & \text{if } j = \arg \left( \min_{1 \leq r \leq c} \| x_i - v_{r,t-1} \| \right) \\
\frac{\| x_i - v_{j,t-1} \|^2}{D^2} & \text{otherwise}
\end{cases} \]

By analyzing relation (5) we can see that GLVQ allows all output neurons to be updated; but gives the same importance to all non-winners, which can be inconvenient. In addition, when , non-winner neurons will have more importance than the winner, which is unacceptable.

3.3 Fuzzy Learning Vector Quantization (FLVQ)

In attempt to better exploit the structural information carried by each data example, Tsao et al. proposed a variant of LVQ for which all neurons are declared winners but with different degrees. This variant is called Fuzzy Learning Vector Quantization and can be viewed as a neural version of the famous algorithm Fuzzy C-Means (FCM). In fact, like FCM, FLVQ uses the following expressions for calculating membership degrees and prototypes vectors:

\[ u_{ji,t} = \left[ \sum_{r=1}^{c} \left( \frac{\| x_i - v_{j,t-1} \|^2}{\| x_i - v_{r,t-1} \|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \]

\[ v_{j,t} = \frac{\sum_{k=1}^{n} (u_{jk,t})^m x_k}{\sum_{k=1}^{n} (u_{jk,t})^m} \]
The difference between FCM and FLVQ concerns the \( m \) parameter, which is constant for FCM but variable for FLVQ. Depending on the way \( m \) varies throughout iterations two versions of FLVQ have been developed: \( \downarrow \)FLVQ and \( \uparrow \)FLVQ. In \( \downarrow \)FLVQ, \( m \) decreases according to the relation:

\[
mt = m_{\max} - \frac{t}{t_{\max}} (m_{\max} - m_{\min})
\]

and in \( \uparrow \)FLVQ it increases according to:

\[
mt = m_{\min} + \frac{t}{t_{\max}} (m_{\max} - m_{\min})
\]

Unlike LVQ and GLVQ, FLVQ and FCM use a learning mode that requires the prior availability of the totality of data examples before starting the learning phase. This means that FLVQ and FCM cannot be used online, i.e., in situations where data are continually arriving.

4 Fuzzy Competitive Learning (FCL)

In this section we present a new technique called Fuzzy Competitive Learning that we have designed in order to remedy to the drawbacks of previously described methods. FCL is an optimization procedure that seeks to minimize the following criterion:

\[
E_{i,t} = \sum_{j=1}^{c} u_{ji,t} \| x_i - v_{j,t-1} \| \tag{10}
\]

where \( u_{ji,t} \) denotes a similarity measure between the object \( x_i \) and the prototype \( v_j \) which represents the \( j \)th class. \( \| x_i - v_{j,t-1} \| = \sum_{q=1}^{p} (x_{iq} - v_{jq,t-1})^2 \) is the distance between \( x_i \) and \( v_j \).

\( E_{i,t} \) can be interpreted as the global error incurred when we replace each object by the prototype of the class to which it belongs.

As a measure of similarity we used the expression:

\[
u_{ji,t} = \frac{\| x_i - v_{ji,t-1} \|^{-1}}{\sum_{r=1}^{c} \| x_i - v_{r,t-1} \|^{-1}} \tag{11}
\]

From equation (11) we can easily see that \( u_{ji,t} \) verifies the three properties:

1) \( 0 < u_{ji,t} < 1 \), 2) \( \sum_{j=1}^{c} u_{ji,t} = 1 \), 3) \( \sum_{i=1}^{n} u_{ji,t} \geq 0 \), which allows us to interpret \( u_{ji,t} \) as a measure of the membership degree of \( x_i \) to the \( j \)th class.

To obtain the rule of adjusting the synaptic weights of output neurons we calculate the derivative of (10) according to the gradient descent rule:

\[
v_{j,t} = v_{j,t-1} + \eta_{t-1} \frac{\partial E_{i,t}}{\partial v_{j,t-1}} \tag{12}
\]
FCL
Given a set of unlabeled data \( X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^{np} \)

Step 1: Chose
- The number of neurons of the output layer, \( c \) ;
- A maximal number of iterations \( t_{\text{max}} \);
- A tolerable threshold for the variation of the weights matrix between two consecutive iterations \( \varepsilon > 0 \);

Step 2: Initialize
- The counter of iterations \( t = 0 \);
- The prototypes matrix \( V_0 = \{v_{1,0}, v_{2,0}, \ldots, v_{c,0}\} \in \mathbb{R}^{cp} \);
- The learning rate \( 0 < \eta_0 < 1 \);

Step 3: While (\( \|V_t - V_{t-1}\| < \varepsilon \) and \( t < t_{\text{max}} \)) do:
  - \( t = t + 1 \);
  - Evaluate \( \eta_t \) using (2)
  - Evaluate \( \xi \) using (14)
  - For each \( x_i \in X \) do:
    - Calculate \( u_{ji,t} \) using (11) with \( 1 \leq j \leq c \)
    - Adjust the synaptic weights of all neurones \( n_j \) that verify \( u_{ji,t} \geq \xi \) using (13)
    - Evaluate the variation: \( \|V_t - V_{t-1}\| = \max_{1 \leq j \leq c} \left( \max_{1 \leq i \leq p} (|v_{ji,t} - v_{ji,t-1}|) \right) \)

That means:
\[
v_{ji,t} = v_{ji,t-1} + c\eta_{t-1} u_{ji,t}^2 \frac{(x_i - v_{ji,t-1})}{\|x_i - v_{ji,t-1}\|} \tag{13}
\]

\( \eta_{t-1} \) is the learning rate whose initial value \( \eta_0 \) fixed by the user.

Hence, for each objet \( x_i = \{x_{i1}, x_{i2}, \ldots, x_{ip}\} \in \mathbb{R}^p \) of the learning base, we can use (11) to calculate the membership degree of \( x_i \) to each class and then adjust the prototype of this class using (13). In this case, all prototypes, including those who are very far from \( x_i \), are considered as winners and benefit from the adjustment of their components. To avoid unnecessarily update far prototypes we have introduced a new parameter \( \xi_t \in [0,1] \) which serves as a measure of the difficulty degree of the competition. Using this parameter we can control the number of winners at each iteration and limit the updating process to prototypes that present a sufficient similarity with the input datum.

Hence, in order to be considered as a winner each neuron \( j \) should verify the condition \( u_{ji,t} > \xi_t \). Initially \( \xi_t \approx 0 \), meaning that the competition is supposed easy and all prototypes have a chance to be adjusted. But as the learning process progresses the competition becomes more and more difficult, \( \xi_t \) increases, and the number of winners decreases. The variation of \( \xi_t \) throughout iterations was heuristically determined and the mathematical expression we
adopted for this study is:

\[ \xi_t = \left( \frac{t}{t_{\text{max}}} \right)^2 \]  

(14)

A more formal description of FCL is given in “figure 2”.

5 Experimental Results and Discussion

In this section, we present typical examples of results provided by the proposed algorithm for a variety of real test data, commonly used in the literature as benchmarks to test and compare algorithms, and we compare these results to those provided by the other studied algorithms. For this, four well-known data sets have been considered:

<table>
<thead>
<tr>
<th>Database</th>
<th>LVQ</th>
<th>GLVQ</th>
<th>↓FLVQ</th>
<th>↑FLVQ</th>
<th>FCM</th>
<th>FCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>10.666</td>
<td>11.333</td>
<td>11.333</td>
<td>10.666</td>
<td>10.666</td>
<td>10</td>
</tr>
<tr>
<td>Yeast</td>
<td>70.081</td>
<td>-</td>
<td>71.024</td>
<td>67.52</td>
<td>60.714</td>
<td>59.299</td>
</tr>
<tr>
<td>Spect</td>
<td>43.82</td>
<td>-</td>
<td>39.7</td>
<td>39.7</td>
<td>39.7</td>
<td>32.209</td>
</tr>
</tbody>
</table>

Table 1: Miscalified Error Rates.

<table>
<thead>
<tr>
<th>Initialisation</th>
<th>LVQ</th>
<th>GLVQ</th>
<th>↓FLVQ</th>
<th>↑FLVQ</th>
<th>FCM</th>
<th>FCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode1</td>
<td>10.666</td>
<td>11.333</td>
<td>11.333</td>
<td>10.666</td>
<td>10.666</td>
<td>10</td>
</tr>
<tr>
<td>Mode3</td>
<td>70.081</td>
<td>-</td>
<td>71.024</td>
<td>67.52</td>
<td>60.714</td>
<td>59.299</td>
</tr>
</tbody>
</table>

Table 2: Miscalified Error Rates for the Three Studied Initialization Modes with Iris-Data.

1) IRIS data is a set of 150 4-dimensional vectors representing each the measures in cm of the length and width of sepal and petal of an iris flower. The dataset consists of 50 samples from each of three different classes: Setosa, Versicolor and Virginia. One of the main characteristics of this example is that
one of the three classes is well separated from the other two which present an important overlapping, making it difficult to separate them,

2) BCW data set contains 699 vectors of 9 dimensions, originated from two classes of different size. The first class contains 458 samples and the second 241. These are numerical data extracted from medical images related to breast cancer,

3) Yeast data set is related protein localization and contains 1484 8-dimensional vectors, distributed over 10 different classes of different size,

4) Spect data set is a medical database of 267 vectors of 22 dimensions originated from two different classes of heart-disease. A first comparison is based on the misclassification error rate defined by (15). Misclassification error rates of each of the six studied learning algorithms are reported, for each data set, on Table 1.

\[
e = \frac{\text{Number of miscalified objects}}{\text{Number of objects}}
\]  

The second comparison is based on the running time of each method. The dataset used for this comparison is a 129x129 IRM image originated from the McConnell cerebral imagery center (figure 3.a). “figure 4” depicts the variation of running time of each algorithm with the number of prototypes.

The third comparison concerns the sensitivity of each method to the prototypes initialization technique. It is based on the results obtained for the Iris data. For this, three different initialization modes were studied :

1) Random initialization, which consists in choosing random initial components for each prototype,

2) Initialization of each component by a random value comprised between two limits that ensure that initial prototypes belong to the data space,

<table>
<thead>
<tr>
<th>Initialisation</th>
<th>LVQ</th>
<th>FCM</th>
<th>FCL</th>
</tr>
</thead>
</table>
| Model1         | \[
\begin{bmatrix}
50 & 0 & 0 \\
50 & 0 & 0 \\
50 & 0 & 0 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
50 & 0 & 0 \\
0 & 47 & 3 \\
0 & 13 & 37 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
50 & 0 & 0 \\
0 & 47 & 3 \\
0 & 13 & 37 \\
\end{bmatrix}
\] |
| Model2         | \[
\begin{bmatrix}
50 & 0 & 0 \\
0 & 47 & 3 \\
0 & 13 & 37 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
50 & 0 & 0 \\
0 & 47 & 3 \\
0 & 13 & 37 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
50 & 0 & 0 \\
0 & 48 & 2 \\
0 & 13 & 37 \\
\end{bmatrix}
\] |
| Model3         | \[
\begin{bmatrix}
50 & 0 & 0 \\
0 & 47 & 3 \\
0 & 13 & 37 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
50 & 0 & 0 \\
0 & 47 & 3 \\
0 & 13 & 37 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
50 & 0 & 0 \\
0 & 48 & 2 \\
0 & 13 & 37 \\
\end{bmatrix}
\] |

Table 3: Confusion Matrices.
3) Each prototype is initialized using an object vector of the learning database. Results of this part are reported on Table 2 and Table 3. For each initialization mode and each learning algorithm Table 2 shows the misclassification error rate, while Table 3 gives the confusion matrix.

The previous results show that, globally, the performances of the proposed method (FCL) are better than those of other well-known methods. Indeed, as we can easily see both the misclassification error rate and the running time of FCL are less than those observed for all the other methods. Another advantage of FCL is its ability to converge to the best prototypes for different initialization modes, which is not the case for other algorithms.

Finally, in “figure 5” we present the evolution of running time and error rate with the learning rate $\eta$, which is one of the most important parameters of our method. As we can see, the choice of $\eta$ can influence both the running time and error rate.

6 Conclusion

In this paper, we presented a new unsupervised learning algorithm for competitive neural networks, called Fuzzy Competitive Learning (FCL). This algorithm has been applied to different test data sets, including images data,
A new fuzzy learning scheme for CNN

Figure 4: Evolution of the running time in seconds with the number of prototypes

Figure 5: Evolution of the Running Time (a) and the Error Rate (b) with the Learning Rate

and its results favorably compared to those produced, for the same data, by other well-known algorithms including LVQ, GLVQ, FLVQ and FCM. These encouraging results justify the continuation of this study in order, for example, to avoid the sensitivity to initialization, which remains a common problem for many problems.

References


Received: January, 2012