Computing and Informatics, Vol. 33, 2014, 237–258

DYNAMIC OPTIMAL TRAINING FOR COMPETITIVE NEURAL NETWORKS

Mohammed Madiafi, Abdelaziz Bouroumi

Information Processing Laboratory
Ben M'Sik Faculty of Sciences
Hassan II Mohammedia-Casablanca University
B.P. 7955 Av. Cdt Driss El Harti
20800 Casablanca, Morocco
e-mail: {madiafi.med, a.bouroumi}@gmail.com

Abstract. This paper introduces an unsupervised learning algorithm for optimal training of competitive neural networks. The learning rule of this algorithm is derived from the minimization of a new objective criterion using the gradient descent technique. Its learning rate and competition difficulty are dynamically adjusted throughout iterations. Numerical results that illustrate the performance of this algorithm in unsupervised pattern classification and image compression are also presented, discussed, and compared to those provided by other well-known algorithms for several examples of real test data.

Keywords: Competitive neural networks, unsupervised learning, clustering, pattern classification, image compression

1 INTRODUCTION

Artificial neural networks (ANNs) are heuristic models that try to simulate the human brain capabilities of learning from examples and generalizing the learned skills to new and unseen examples. They have been proved effective in solving many hard real-world problems in different application domains, especially in the field of pattern classification and recognition [1.2].

In practice, ANNs are used as alternatives to traditional deterministic models in order to find, in a reasonable amount of time, approximate yet satisfying solutions to difficult real-world problems that are out of reach for these conventional models. Such problems are often encountered in a variety of fields and applications including medicine, telecommunications, economics, engineering, environment, etc. [1.2].

Technically speaking, the design of an artificial neural networks-based solution to a given practical problem requires three main steps. The first step is the choice of a suitable architecture for the ANN, i.e., the number of neurons or processing elements to use and a suitable way for connecting them in order to build the network. The second step is the choice of an appropriate algorithm for training the network, i.e., a way for adjusting the synaptic weights of the physical connections between pairs of neurons. The third step is the choice of a good learning dataset $X = \{x_1, x_2, \ldots, x_n\} \subset R^p$, i.e., a set of sample examples that can be used during the learning phase as input data for the training algorithm, where each example k is represented by a p-dimensionnal object vector x_k and each component $x_{ik} \in R$ is the numerical value of one of its p features.

Based on the information carried by each of these sample examples, the training process consists in repeatedly adjusting the synaptic weights of the chosen architecture so that the resulting network could be able to recognize, without errors, new and unseen examples when it will be used in the generalization phase. If the elements of X are labeled, i.e., if the original class of each learning example is a priori known, the learning process can be performed in supervised mode. Otherwise we say that the learning process is unsupervised [2].

The most known ANNs are undoubtedly multilayer perceptrons (MLP), which are commonly used in real-world problems for which sufficient amounts of labeled data are available. MLP use a simple architecture with an input layer, an output layer, and one or more hidden layers. They can be efficiently trained using the ubiquitous back-propagation algorithm, which is an optimization procedure aimed at minimizing the global misclassification errors of the network [3.4].

In this paper, we are rather interested in the application of ANN in situations where large amounts of data examples are available but are not labeled, particularly in the field of unsupervised pattern classification and recognition. ANNs that can be used in these situations are called competitive neural networks (CNNs), and their general architecture is depicted in Figure 1.

According to this architecture a CNN can be seen as a particular type of MLP which contains only two layers. The first layer is an input layer whose role is to receive signals representing input data. It contains p neurons, where p is the dimensionality of the data space. The second layer is an output layer, aka competitive layer, whose number of neurons, c, is equal to the number of classes or homogeneous groups supposed present in the learning data base. However, as its learning data are not labeled, a CNN cannot be trained in supervised mode, meaning that the synaptic weights should not be modified in response to desired outputs but rather to the inputs. In this context, the adjective "competitive" means that each time a sample example $x_k (1 \leq k \leq n)$ is presented to the input layer, the c neurons of the output layer compete for benefiting from the information carried by this example in order to adjust their own weights.

Let $W = (w_1, w_2, \ldots, w_c) \subset \mathbf{R}^p$ be the matrix formed by the $c \times p$ weights to be learned, where each vector $w_j \in \mathbf{R}^p$ represents the synaptic weights of the p connections of an output neuron $(1 \leq j \leq c)$; the problem of training the CNN

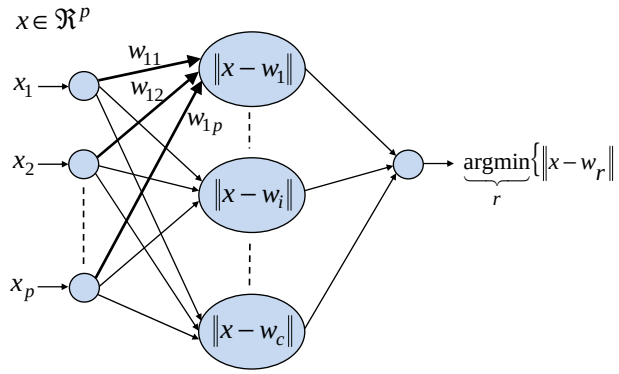

Figure 1. General architecture of a CNN

can be posed in terms of finding the best way to exploit the information carried by the n learning examples in order to find the best value for each of the $c \times p$ synaptic weights. It is a difficult search and optimization problem for which many algorithms have been proposed in the literature, and which still remains an open problem [2, 11].

By remarking that each output vector w_j of the learning phase can also be interpreted as a codebook, or prototype or representative, of the j^{th} class ($1 \leq j \leq c$), CNN can be viewed as prototype generator classifiers. Indeed, what a CNN actually learns are representatives of the c classes supposed present in the learning dataset. In the generalization phase, these representatives can be used, for example, in order to perform the task of recognizing new and unseen examples. A simple way of doing this is by applying the nearest prototype recognition rule (1-np), which consists in assigning each new example x_k to the class i whose prototype is the nearest one to the object-vector associated with that example (Figure 1).

During the last two decades, many learning algorithms have been developed for training CNNs [6, 15]. Based all on the principle of competition, these algorithms differ mainly in the way they manage this competition, i.e., in the learning rule they use for adjusting the synaptic weights of the network. These rules range from the classical winner takes all (WTA) rule, for which at each iteration only one neuron, the winner, benefits from adjusting its weights to different fuzzy learning rules for which all neurons are considered as winners, but to different degrees that fix the extend to which the weights of each neuron should be adjusted [5, 10]. Experimental studies of several variants of these algorithms showed, however, that no one of them is robust enough to perform well for different data sets and different application domains [8, 9].

In this paper, we develop a learning technique for CNN that tries to combine the advantages of some studied methods whilst avoiding their drawbacks. It is an optimal technique whose learning rule is based on the minimization of an objective

criterion. It is also dynamic in that it uses a dynamically adjusted learning rate and a growing competition difficulty.

Before detailing the proposed method in Section 3, in Section 2 we give a brief presentation of the principal other methods studied in this work. Numerical results and discussions will be given in Section 4. Finally, Section 5 contains our conclusion and some perspectives for future work.

2 CNN TRAINING ALGORITHMS

As we mentioned in the above introduction, many unsupervised learning algorithms have been proposed in the literature for training CNN with unlabeled data examples. In the framework of this study, several of these algorithms were studied, programmed and tested on different examples of test datasets. This section gives a brief description of the principal ones among these algorithms. Further information about each presented technique can be found in the corresponding references.

2.1 The Learning Vector Quantization (LVQ)

The first algorithm, known in the literature under the name Learning Vector Quantization (LVQ), was proposed by Kohonen in 1989 [12]. 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 $x_k = \{x_{k1}, x_{k2}, \ldots, x_{kp}\} \in R^p$ presented to the network, we locate the neuron i whose synaptic weights vector $v_i = \{v_{i1}, v_{i2}, \ldots, v_{ip}\} \in R^p$ minimizes the distance $\|x_k - v_i\|$. This vector is then updated according to the rule

$$v_{i,t} = v_{i,t-1} + \eta_{t-1}(x_k - v_{i,t-1}) \tag{1}$$

where η_{t-1} is the learning rate, whose role is to control the convergence of weights vectors to class prototypes. Starting from an initial value η_0. This learning rate is adjusted at each iteration t, according to the relation

$$\eta_t = \eta_0 \left(1 - \frac{t}{t_{tmax}}\right). \tag{2}$$

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

This algorithm suffers from some drawbacks such as its sensitivity to the initialization, the risk of a dominant object that always wins the competition, and a poor 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 is 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 as Generalized Learning Vector Quantization (GLVQ).

2.2 The Generalized Learning Vector Quantization (GLVQ)

Proposed by Pal in 1993 [5], GLVQ is an optimization procedure that tries to minimize the following criterion

$$J_k = \sum_{i=1}^{c} \alpha_{ki} \|x_k - v_{i,t-1}\|^2 \tag{3}$$

with

$$\alpha_{ki} = \begin{cases} 1 & \text{if } i = \arg\min_{1 \le r \le c} \|x_k - v_{r,t-1}\| \\ \frac{1}{\sum_{r=1}^{c} \|x_k - v_{r,t-1}\|^2} = \frac{1}{D} & \text{otherwise.} \end{cases} \tag{4}$$

This is done by updating the synaptic weights of all neurons of the output layer according to the rule

$$v_{i,t} = v_{i,t-1} + \eta_{t-1} \frac{\partial J_k}{\partial v_{i,t-1}} \tag{5}$$

that is

$$v_{i,t} = v_{i,t-1} + \eta_{t-1}(x_k - v_{i,t-1}) \times \psi_{ki} \tag{6}$$

with

$$\psi_{ki} = \begin{cases} \frac{D^2 - D + \|x_k - v_{i,t-1}\|^2}{D^2} & \text{if } i = \arg(\min_{1 \le r \le c} \|x_k - v_{r,t-1}\|) \\ \frac{\|x_k - v_{i,t-1}\|^2}{D^2} & \text{otherwise.} \end{cases} \tag{7}$$

By analyzing the relation (5) we can see that GLVQ allows all output neurons to be updated and gives the same importance to all non-winners, which can be inconvenient. In addition, when $D \in]0, 1[$, non-winner neurons will have more importance than the winner, which is inacceptable. To overcome these drawbacks, different GLVQ versions have been presented in the literature, with different formulations for the criterion to minimize and for the D parameter of Equation (4). Among these GLVQ versions, we mention the revised GLVQ or RGLVQ.

2.3 The Revised Generalized Learning Vector Quantization (RGLVQ)

RGLVQ is an optimal training algorithm for CNN [18], which aims at minimizing the loss fonction $L(x_k, v_i)$, defined for each object vector x_k of the learning dataset and each prototype v_i by

$$\begin{aligned} L(x_k, v_i) &= \|x_k - v_i\|^2 + \frac{\|x_k - v_i\|^2}{D} \sum_{r \neq i}^{c} \|x_k - v_r\|^2 \\ &= 2\|x_k - v_i\|^2 - \frac{\|x_k - v_i\|^4}{D} \end{aligned} \tag{8}$$

with

$$D = \sum_{j}^{c} \|x_k - v_j\|^2. \tag{9}$$

According to this method, the winner is the prototype i that verifies

$$\|x_k - v_i\|^2 \leq \|x_k - v_j\|^2, \quad \forall j \neq i \tag{10}$$

The updating coefficient w_r is determined from the gradient of $L(x_k, v_i)$ as follows:

$$w_r = \begin{cases} 1 - \frac{\|x_k - v_i\|^2}{D} + \frac{\|x_k - v_i\|^4}{2D^2}, & r = i \\ \frac{\|x_k - v_i\|^4}{2D^2}, & r \neq i \end{cases} \tag{11}$$

Despite the modification it brought to GLVQ, RGLVQ suffers from the same drawbacks as GLVQ, especially in the choice of an appropriate mathematical expression for the function D.

Another way to generalize LVQ is by conceiving fuzzy versions inspired by the Bezdek's fuzzy c-means algorithm [6, 13]. In what follows, we give a brief description of the fuzzy c-means algorithm (FCM) and some competitive learning schemes that represent examples of fuzzy versions of LVQ.

2.4 The Fuzzy C-Means Algorithm (FCM)

FCM is a prototype generator classifier designed as an optimization procedure for finding the best fuzzy c-partition of the n elements of X [16, 17, 23]. This procedure tries to minimize the objective function

$$J_m(U, V, X) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m \|x_k - v_i\|^2 \tag{12}$$

where U is the matrix of membership degrees, with u_{ik} denoting the membership degree of the k^{th} object to the i^{th} class; V is the matrix of class prototypes, with v_i representing the i^{th} class prototype; $m > 1$ is a weighting exponent whose role is to control the fuzziness degree of candidate partitions during the learning process.

J_m can be interpreted as a fuzzy measure of the global error incurred in representing the n training vectors by the c prototypes. In 1973, Bezdek proved that in order for FCM to converge to a local minimum of J_m, i.e., to a sub-optimal solution, it is necessary to calculate the components of U and V matrices according to the expressions

$$u_{ik,t} = \left[\sum_{r=1}^{c} \left(\frac{\|x_k - v_{i,t-1}\|^2}{\|x_k - v_{r,t-1}\|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{13}$$

and

$$v_{i,t} = \frac{\sum_{k=1}^{n} (u_{ik,t})^m x_k}{\sum_{k=1}^{n} (u_{ik,t})^m}. \tag{14}$$

Hence, starting from a randomly chosen matrix of initial prototypes $V_0 = \{v_{1.0}, v_{2.0}, \ldots, v_{c,0}\}$, if we repeatedly calculate the components of U and V according to (13) and (14), the iterative process will converge to a point (U^*, V^*) where the c prototypes are stabilized, i.e., a point from which the difference $\|V_t - V_{t-1}\|$ becomes non-significant. $\|V_t - V_{t-1}\|$ is evaluated using the expression

$$\|V_t - V_{t-1}\| = \max_{1\leq i\leq c}\left(\max_{1\leq j\leq p}\left(|v_{ij,t-v_{ij,t-1}}|\right)\right). \tag{15}$$

2.5 Huntsberger and Ajjimarangsee Scheme (HALVQ)

Huntsberger and Ajjimarangsee [7] tried to establish a connection between vector quantization and fuzzy clustering. They proposed a modified version of LVQ inspired by the FCM algorithm, which leads to the following expression

$$v_{r,t} = v_{r,t-1} + \left[\sum_{j=1}^{c}\left(\frac{\|x_i - vr, t-1\|^{\frac{2}{(m-1)}}}{\|x_i - vj, t-1\|^{\frac{2}{(m-1)}}}\right)\right]^2 \times (x_i - v_{r,t-1}). \tag{16}$$

with $1 \leq r \leq c$, $1 \leq i \leq n$ and $1 \leq t \leq t_{max}$

However, this hybrid learning scheme lacked theoretical foundations, formal derivations and clear objectives.

2.6 The Karayiannis Scheme (FGLVQ)

Karayannis et al. [10, 13] proposed another fuzzy version of GLVQ, which consists in using as weights of prototypes the membership degrees produced in developing a fuzzy learning rule by applying the gradient descent technique to the criterion J_k defined by Equation (3).

$$v_{r,t} = v_{r,t-1} - \alpha_t \frac{\partial}{\partial v}\left(\sum_{r=1}^{c} u_{rk}\|x_k - v_{r,t-1}\|^2\right) \tag{17}$$

where

$$u_{rk} = \left[\sum_{j=1}^{c}\left(\frac{\|x_k - v_r\|^{\frac{2}{m-1}}}{\|x_k - v_j\|^{\frac{2}{m-1}}}\right)\right]^{-1}. \tag{18}$$

Karayiannis et al. said that the use of (17) is difficult to justify mathematically and its introduction is based only on intuition hoping that some of the robustness of FCM could be transferred to LVQ by incorporating this relationship. Consequently, FGLVQ may be subject to the same criticism as HALVQ described above.

2.7 The Fuzzy Learning Vector Quantization (FLVQ)

In an attempt to better exploit the structural information carried by each data example, Tsao et al. [6] proposed a variant of LVQ for which all neurons are considered

as winners but with different degrees. This variant is called Fuzzy Learning Vector Quantization (FLVQ) and can be viewed as a neural version of FCM. In fact, like FCM, FLVQ uses the following expressions for calculating membership degrees and prototypes

$$u_{ik,t} = \left[\sum_{r=1}^{c} \left(\frac{\|x_k - v_{i,t-1}\|^2}{\|x_k - v_{r,t-1}\|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{19}$$

$$v_{i,t} = \frac{\sum_{k=1}^{n} (u_{ik,t})^m x_k}{\sum_{k=1}^{n} (u_{ik,t})^m} \tag{20}$$

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: FLVQ↓ and FLVQ↑. In FLVQ↓, m decreases according to the relation

$$m_t = m_{\max} - \frac{t}{t_{\max}} (m_{\max} - m_{\min}) \tag{21}$$

and in FLVQ↑ it increases according to

$$m_t = m_{\min} + \frac{t}{t_{\max}} (m_{\max} - m_{\min}) . \tag{22}$$

In recent years, several improved versions of FLVQ were proposed in the literature, among which we mention the improved batch fuzzy learning vector quantization (IBFLVQ), which has been proposed especially for image compression applications.

2.8 Improved Batch Fuzzy Learning Vector Quantization (IBFLVQ)

IBFLVQ is an optimization algorithm [14], which aims to find prototypes that optimize the objective function

$$J_R(U, V, X) = \theta_t \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik,t} \|x_k - v_{i,t}\|^2 + (1 - \theta_t) \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik,t}^2 \|x_k - v_{i,t}\|^2 \tag{23}$$

where $\theta \in [0, 1[$ is a user defined parameter introduced in order to control the transition from fuzzy to crisp case. The components of the membership degrees matrix $U = \{u_{ik,t}\}_{k=1,\ldots,n;i=1,\ldots,c}$ are calculated according to

$$u_{ik,t} = \frac{2 + (card(T_{k,t}) - 2)\theta_t}{2(1 - \theta_t)} \frac{1}{\sum_{v_j \in T_{k,t}} \left(\frac{\|x_k - v_{i,t}\|}{\|x_k - v_{j,t}\|} \right)^2} - \frac{\theta_t}{2(1 - \theta_t)} \tag{24}$$

with

$$T_{k,t} = \{v_{i,t} \in V_t : u_{ik,t} > 0\} \tag{25}$$

Those of the prototypes matrix $V_t = \{v_{i,t}\}_{i=1,\ldots,c}$ are calculated using the expression

$$v_{i,t} = \frac{\sum_{k=1}^{n}[\theta_t w_{ik,t} + (1-\theta_t)(w_{ik,t})^2]x_k}{\sum_{k=1}^{n}[\theta_t w_{ik,t} + (1-\theta_t)(w_{ik,t})^2]} \tag{26}$$

with

$$w_{ik,t} = \frac{u_{ik,t}}{\sum_{i=1}^{c} u_{ik,t}}, 1 \leq k \leq n, 1 \leq i \leq c. \tag{27}$$

Hence, starting from a randomly chosen matrix of initial prototypes $V_0 = \{v_{1.0}, v_{2.0}, \ldots, v_{c,0}\}$, if we repeatedly calculate the components of U and V according to Equations (21), (23) and (24), the iterative process will converge to a stabilization point.

To close this section we note that, in this work, all the 5 techniques presented above have been coded using C++ language and the resulting software was tested and compared with the one implementing our technique on several data examples. The results of these comparisons will be presented and discussed in Section 4.

3 DYNAMIC OPTIMAL TRAINING FOR COMPETITIVE NEURAL NETWORKS (DOTCNN)

The technique we propose in this work is designed as a fuzzy competitive learning scheme for training competitive neural networks [22]. It is an optimization procedure, aimed at minimizing an objective criterion that can be interpreted as a new error measure. The minimization process is performed using the gradient descent method. It gives rise to a new learning rule, which constitutes the main characteristics of our method.

3.1 The Objective Criterion

It is very difficult to choose a good objective criterion. This difficulty is mainly due to the impossibility of describing all desirable features of a good result by the mean of a unique formula that would be valid for different data structures and different applications. Our experimental results showed, indeed, that no one of the objective criterions used in the studied techniques is universally perfect or universally bad. This means that, in completely unsupervised environment, i.e., in the absence of any prior information about the actual structure of the data to analyze, we have no objective way to definitely adopt a specific criterion. Consequently, it would be better to use a generalized, but simple, expression with more user defined parameters. That would let the user adjust the algorithmic parameters according to the specificities of each application. Numerical results presented in this study confirm the usefulness of this idea.

To develop a new objective criterion, we start from a very simple objective criterion used to express the error $E_{k,t}^{crisp}$ incurred when we replace each object x_k

with the prototype of the class it belongs to, i.e.

$$E_{k,t}^{\text{classic}} = \sum_{i=1}^{c} g_{ik,t-1} \|x_k - w_{i,t-1}\|^2 \tag{28}$$

with

$$g_{ik,t-1} = \begin{cases} 1 & \text{if } i = \arg\min_{1 \le i \le c}\{\|x_k - w_{i,t-1}\|\} \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

This criterion is valid in the crisp case where the membership of any object x_k to any class C_j is equal to either 1 or 0 (29).

Since our method is a fuzzy learning scheme, where each object belongs to all classes with different degrees, it is necessary to provide a fuzzy generalization of $E_{k,t}^{classic}$ that can be used as an objective criterion. The mathematical expression we propose for this is

$$E_{k,t} = \sum_{i=1}^{c} u_{ik,t-1}^{m} \|x_k - w_{i,t-1}\|^{\alpha} \tag{30}$$

where $u_{ik,t-1}$ denotes the membership degree of x_k to the i^{th} class, whose prototype at iteration $t-1$ is $w_{i,t-1}$. $u_{ik,t-1}$ also represents the synaptic weights vector of the i^{th} neuron of the output layer; m is a positive parameter adjustable by the user, which controls the fuzziness of the partition generated by DOTCNN.

$E_{k,t}$ can be interpreted as a fuzzy measure of the global error incurred in replacing each input vector x_k with its nearest prototype.

To evaluate $u_{ik,t-1}$, we use the following expression

$$u_{ik,t-1} = \frac{\|x_k - w_{i,t-1}\|^{-\alpha}}{\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}} \tag{31}$$

where $\|x_k - w_{i,t-1}\|$ denotes the Euclidean distance between x_k and $w_{i,t-1}$ at iteration $t-1$, i.e.

$$\|x_k - w_{i,t-1}\| = \sqrt{\sum_{q=1}^{p} (x_{kq} - w_{iq,t-1})^2} \tag{32}$$

where α is a user defined parameter whose values are heuristically fixed by the user such as $\alpha \geq 1$.

3.2 The Learning Rule

The learning rule is the protocol that fixes the adjustment $\Delta w_{rqk,t}$ that should be applied to each component $w_{rq,t-1}$ of each weights vector $w_{r,t-1}$ of each neuron r at each iteration t. Its general form is given by the expression

$$w_{rq,t} = w_{rq,t-1} + \Delta w_{rqk,t} \tag{33}$$

for $r = 1, \ldots, c$; $q = 1, \ldots, p$; and $k = 1, \ldots, n$.

To calculate $\Delta w_{rqk,t}$, we use the gradient descent method

$$\Delta w_{rqk,t} = -\eta_t \frac{\partial E_{k,t}}{\partial w_{rq,t-1}} \tag{34}$$

which gives

$$\begin{aligned}
\Delta w_{rqk,t} &= -\eta \tfrac{\partial}{\partial w_{rq,t-1}} \left(\textstyle\sum_{i=1}^{c} u_{ik,t-1}^{m} \|x_k - w_{i,t-1}\|^{\alpha} \right) \\
&= -\eta_t \textstyle\sum_{i=1}^{c} \left(\|x_k - w_{i,t-1}\|^{\alpha} \frac{\partial u_{ik,t-1}^{m}}{\partial w_{rq,t-1}} + u_{ik,t-1}^{m} \frac{\partial \|x_k - w_{i,t-1}\|^{\alpha}}{\partial w_{rq,t-1}} \right) \\
&= -\eta_t \textstyle\sum_{i=1}^{r-1} \left(\|x_k - w_{i,t-1}\|^{\alpha} \left[\frac{\partial u_{ik,t-1}^{m}}{\partial w_{rq,t-1}} \right]_{i \neq r} \right) - \eta_t \|x_k - w_{r,t-1}\|^{\alpha} \frac{\partial u_{rk,t-1}^{m}}{\partial w_{rq,t-1}} \\
&\quad -\eta_t \textstyle\sum_{i=r+1}^{c} \left(\|x_k - w_{i,t-1}\|^{\alpha} \left[\frac{\partial u_{ik,t-1}^{m}}{\partial w_{rq,t-1}} \right]_{i \neq r} \right) - \eta_t u_{rk,t-1}^{m} \frac{\partial \|x_k - w_{r,t-1}\|^{\alpha}}{\partial w_{rq,t-1}}
\end{aligned} \tag{35}$$

with

$$\begin{aligned}
\left[\frac{\partial u_{ik,t-1}}{\partial w_{rq,t-1}} \right]_{i \neq r} &= \frac{\partial \left[\frac{\|x_k - w_{i,t-1}\|^{-\alpha}}{\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}} \right]_{i \neq r}}{\partial w_{rq,t-1}} = \frac{-\|x_k - w_{i,t-1}\|^{-\alpha} \frac{\partial \|x_k - w_{r,t-1}\|^{-\alpha}}{\partial w_{rq,t-1}}}{\left(\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha} \right)^2} \\
&= \frac{-\alpha \left(x_{kq} - w_{rq,t-1} \right) \|x_k - w_{r,t-1}\|^{-\alpha-2} \|x_k - w_{i,t-1}\|^{-\alpha}}{\left(\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha} \right)^2}
\end{aligned} \tag{36}$$

$$\begin{aligned}
\frac{\partial u_{rk,t-1}}{\partial w_{rq,t-1}} &= \frac{\partial \left[\frac{\|x_k - w_{r,t-1}\|^{-\alpha}}{\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}} \right]}{\partial w_{rq,t-1}} \\
&= \frac{\frac{\partial \|x_k - w_{i,t-1}\|^{-\alpha}}{\partial w_{rq,t-1}}}{\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}} + \frac{-\|x_k - w_{i,t-1}\|^{-\alpha} \frac{\partial \|x_k - w_{r,t-1}\|^{-\alpha}}{\partial w_{rq,t-1}}}{\left(\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha} \right)^2} \\
&= \frac{-\alpha \left(x_{kq} - w_{rq,t-1} \right) \|x_k - w_{r,t-1}\|^{-\alpha-2} \|x_k - w_{i,t-1}\|^{-\alpha}}{\left(\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha} \right)^2} \\
&\quad + \frac{\alpha \left(x_{kq} - w_{rq,t-1} \right) \|x_k - w_{r,t-1}\|^{-\alpha-2}}{\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}}
\end{aligned} \tag{37}$$

$$\frac{\partial \|x_k - w_{r,t-1}\|^{\alpha}}{\partial w_{rq,t-1}} = -\alpha \left(x_{kq} - w_{rq} \right) \|x_k - w_{r,t-1}\|^{\alpha-2}. \tag{38}$$

Hence

$$\begin{aligned}
\Delta w_{rqk,t} &= -\eta_t \sum_{i=1}^{c} \left(\frac{-\alpha\ m\ u_{ik,t-1}^{m-1} \left(x_{kq} - w_{rq}\right) \|x_k - w_{r,t-1}\|^{-\alpha-2}}{\left(\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}\right)^2} \right) \\
&\quad -\eta_t \frac{\alpha\ m\ u_{rk,t-1}^{m-1} \left(x_{kq} - w_{rq}\right) \|x_k - w_{r,t-1}\|^{-2}}{\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}} \\
&\quad + \eta_t \alpha\ u_{rk,t-1}^{m} \left(x_k - w_{rq,t-1}\right) \|x_k - w_{r,t-1}\|^{\alpha-2} \\
&= -\eta_t \sum_{i=1}^{c} \frac{\left(-\alpha\ m\ u_{ik,t-1}^{m-1}\right) \|x_k - w_{r,t-1}\|^{-2} \|x_k - w_{r,t-1}\|^{\alpha} \|x_k - w_{r,t-1}\|^{-2\alpha}}{\left(\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}\right)^2} \\
&\quad -\eta_t \frac{\alpha\ m\ u_{rk,t-1}^{m-1} \left(x_{kq} - wrq,t-1\right) \|x_k - w_{r,t-1}\|^{-2} \|x_k - w_{r,t-1}\|^{\alpha} \|x_k - w_{r,t-1}\|^{-\alpha}}{\sum_{j=1}^{c} \|x_k - w_{j,t-1}\|^{-\alpha}} \qquad (39) \\
&\quad + \eta_t\ \alpha\ m\ u_{rk,t-1}^{m} \left(x_{kq} - w_{rq}\right) \|x_k - w_{r,t-1}\|^{\alpha-2} \\
&= -\eta_t \sum_{i=1}^{c} \left(-\alpha\ m\ u_{ik,t-1}^{m-1} \left(x_{kq} - w_{rq,t-1}\right) \|x_k - w_{r,t-1}\|^{\alpha-2} u_{rk,t-1}^{2}\right) \\
&\quad -\eta_t\ \alpha\ m\ u_{rk,t-1}^{m-1} \left(x_k q - w_{rq,t-1}\right) \|x_k - w_{r,t-1}\|^{\alpha-2} u_{rk,t-1} \\
&\quad + \eta_t\ \alpha\ u_{rk,t-1}^{m} \left(x_{kq} - w_{rq,t-1}\right) \|x_k - w_{r,t-1}\|^{\alpha-2} \\
&= \eta_t\ \alpha\ u_{rk,t-1}^{2} \frac{\left(x_{kq} - w_{rq,t-1}\right)}{\|x_k - w_{r,t-1}\|^{-\alpha+2}} \left[(1-m)\, u_{rk,t-1}^{m-2} + \sum_{i=1}^{c} m\ u_{ik,t-1}^{m-1}\right].
\end{aligned}$$

The resulting learning rule can be expressed in the form:

$$w_{rq,t} = w_{rq,t-1} + \eta_t\ \psi_{rk,t-1}\ \ \left(x_{kq} - w_{rq}\right) \qquad (40)$$

with

$$\psi_{rk,t-1} = \alpha\ u_{rk,t-1}^{2} \|x_k - w_{r,t-1}\|^{\alpha-2} \left[(1-m)\, u_{rk,t-1}^{m-2} + \sum_{i=1}^{c} m\ u_{ik,t-1}^{m-1}\right] \qquad (41)$$

where η_t denotes the learning rate at iteration t. This rate decreases during iterations according to the expression

$$\eta_t = \eta_{t-1} \left(1 - \frac{t}{t_{max}}\right). \qquad (42)$$

As to the learning process, it operates as follows: for each training vector x_k, we calculate the c membership degrees $\{u_{ik,t}\}_{i=1,\ldots,c}$. Neurons qualified to compete are then selected and all winners benefit from the adjustment of their weight vectors according to (40) and (41). Qualified neurons are those which verify the condition

$$u_{ik,t} \geq \zeta_t \qquad (43)$$

where ζ_t is a measure of the degree of difficulty of the competition at iteration t. The role of this measure is to control the difficulty of the competition throughout

iterations in order to dynamically fix the number of winners. For this, ζ_t is varied according to the heuristically determined expression

$$\zeta_t = \left(\frac{t}{t_{max}}\right)^{\beta} \tag{44}$$

where β is a positive parameter adjustable by the user.

Hence, at the beginning of the learning process, $\zeta_{t=0} = 0$, the competition is easy and all neurons are qualified for competition. However, as iterations continue, the competition becomes ever harder and few neurons would be qualified. This means that the proposed learning rule verifies three main conditions:

1. it is optimal because it is based on the minimization of an objective criterion,
2. it is fuzzy because at each iteration many neurons can be considered as winners and their synaptic weights can be adjusted to different degrees, and
3. it is dynamic because the difficulty of competition varies throughout iterations, leading to a natural decrease of the number of winners.

A more formal description of this unsupervised learning algorithm is given by the following pseudo-code:
Given a set of unlabeled data $X = \{x_1, x_2, \ldots, x_n\} \in R^{n \times p}$

Step 1: Chose

- The number of neurons of the output layer, c;
- A maximal number of iterations $t_{\max}$;
- A tolerable threshold $\varepsilon > 0$ for the variation of the weights matrix between two consecutive iterations;

Step 2: Initialize

- The counter of iterations $t = 0$;
- The prototypes matrix $W_0 = \{w_{1.0}, w_{2.0}, \ldots, w_{c,0}\} \in R^{c \times p}$;
- The learning rate $0 < \eta_0 < 1$;

Step 3: do {

- $t = t + 1$;
- Evaluate η_t using (42)
- Evaluate ζ using (44)
- For each $x_k \in X$ do {
  - Calculate $u_{ik,t}$ using (11) with $1 \leq i \leq c$
  - Adjust the synaptic weights of all neurons n_i that verify (43) using (40) and (41).

  }

- Evaluate the variation:
  $\|W_t - W_{t-1}\| = \max_{1\leq i\leq c}(\max_{1\leq j\leq p}(|w_{ij,t} - w_{ij,t-1}|))$

} While ($\|W_t - W_{t-1}\| < \varepsilon$ and $t < t_{\max}$);

4 NUMERICAL RESULTS AND DISCUSSION

In this section, we present and compare typical examples of numerical results obtained by applying the proposed technique, and the other techniques described in the previous sections to two kinds of problems: clustering and compression.

4.1 Clustering

A clustering problem can be formulated mathematically as follows: given a set of n unlabeled feature vectors $X = \{x_1, x_2, \ldots, x_n\} \subset R^p$, where $x_k \in R^p$ represents a sample object ($1 \leq k \leq n$) and $x_{kj} \in R$ the numerical value of its j^{th} feature, find the best way to organize the n objects into c natural groups or clusters such that objects within each group will be more similar to each other than are objects belonging to different groups [17, 23, 24, 25].

To illustrate the efficacy of the proposed technique in clustering, in this section we present the numerical results of its application to a series of well-known real data sets, which are publicly available on the machine learning repository of the University of California at Irvine [19]. These examples concern five popular datasets:

1. Bcw: 699 object vectors of 9 components, which are calculated from a scanned image in relation with breast cancer. The objects described in this dataset are divided into 2 classes of different sizes. The first class contains 458 samples and the second 241 samples.
2. Yeast: 1484 amino acid sequences of 8 components. These sequences are divided into 10 classes of 463, 429, 244, 163, 51, 44, 37, 30, 20 and 5 samples, respectively. Yeast dataset is used to test systems conceived for cellular localization sites of proteins.
3. Spect: 267 object vectors of 22 components, which are calculated from images of SPECT (Single Proton Emission Computed Tomography) concerning cardiac analysis. The objects of this dataset are divided into 2 classes with 212 and 55 samples.
4. Wine: 178 object vectors of 13 components, which represent a set of measures in relation to 3 varieties of Italian wine, with 59, 79, and 48 samples per variety, respectively.
5. FGlass: comes from forensic testing of 214 glass fragments of 13 components. The objects described in this dataset are divided into 3 classes of different sizes, with 87, 76 and 51 samples per class, respectively.

All these datasets are used as bases of unlabeled examples to train our unsupervised neural network. The name tags of the objects described in these datasets are used to evaluate the misclassification error rate, e_c, according to the expression

$$e_c = \frac{n^*}{n} \tag{45}$$

with n^* being the number of misclassified objects and n the total number of objects in the dataset.

The first study of this part aims at applying the proposed method, DOTCNN, and the other techniques discussed in this paper to the five test data. Then we evaluate the misclassification error rate for each technique. We do this in three steps:

1. We start by finding the prototypes of the c clusters assumed in each dataset; for example, the application of DOTCNN to BCW dataset gives the two prototypes

$$\begin{aligned} w_1 &= \{2.654; 1.137; 1.276; 1.238; 2.081; 1.107; 1.571; 1.129; 1.201\} \\ w_2 &= \{6.670; 7.836; 7.566; 6.674; 5.198; 7.743; 7.252; 6.403; 2.360\}. \end{aligned}$$

2. We apply the nearest prototype rule to completely assign each object x_k to a unique cluster according to the decision rule $x_k \in C_j$ if $j = \arg\min_{1 \le j \le c}\{\|x_k - w_j\|\}$, with C_j being the j^{th} cluster.
3. We evaluate the number of misclassified objects n^*. In this step, we must have a prior knowledge about the membership of each object. The results of this study are shown in Table 1.

Figure 2. From left to right: Lena; Fishing-Boat; Baboon; Peppers

The second study in this part consists of measuring the quality of the partition generated by each fuzzy technique viewed in this paper. The quality of the partition $P(X)$ is described by the partition coefficient PC(U) expressed mathematically as follows:

$$PC(U) = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^2 \tag{46}$$

where U is the matrix of membership degrees, u_{ik} is the membership degree of x_k to the i^{th} cluster.

Figure 3. From left to right: Compressed by LVQ$\{c = 512\}$; Compressed by FLVQ$\{c = 512\}$; Compressed by IFLVQ$\{c = 512\}$; Compressed by DOTCNN$\{c = 512\}$

Figure 4. From left to right: Compressed by LVQ$\{c = 512\}$; Compressed by FLVQ$\{c = 512\}$; Compressed by IFLVQ$\{c = 512\}$; Compressed by DOTCNN$\{c = 512\}$

Figure 5. From left to right: Compressed by LVQ$\{c = 512\}$; Compressed by FLVQ$\{c = 512\}$; Compressed by IFLVQ$\{c = 512\}$; Compressed by DOTCNN$\{c = 512\}$

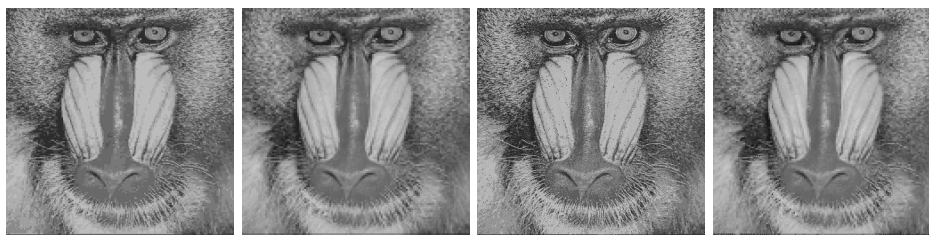

Figure 6. From left to right: Compressed by LVQ$\{c = 512\}$; Compressed by FLVQ$\{c = 512\}$; Compressed by IFLVQ$\{c = 512\}$; Compressed by DOTCNN$\{c = 512\}$

Technique	Misclassification error rate (%) for each dataset				
	Spect	Wine	FGlass	BCW	Yeast
FCM	39.7 ± 1.5	31.46 ± 1.12	71.68 ± 1.40	4.39 ± 0.72	69.18 ± 0.54
FGLVQ	79.4 ± 0.0	31.46 ± 2.25	77.94 ± 0.93	3.95 ± 0.36	68.8 ± 1.01
FLVQ↓	79.4 ± 0.0	31.46 ± 1.12	76.54 ± 0.47	4.25 ± 0.18	73.52 ± 0.54
FLVQ↑	79.4 ± 0.0	30.34 ± 1.12	68.32 ± 0.47	4.25 ± 0.18	65.84 ± 0.61
GLVQ	79.4 ± 0.0	35.06 ± 2.25	64.39 ± 0.93	3.95 ± 0.36	68.8 ± 1.28
HALVQ	79.4 ± 0.0	29.21 ± 1.69	67.29 ± 0.70	34.99 ± 0.3	68.8 ± 1.15
LVQ	43.4 ± 3.3	34.94 ± 3.93	67.1 ± 3.74	3.95 ± 1.29	69.74 ± 1.55
RGLVQ	42.7 ± 2.2	36.24 ± 2.25	64.02 ± 2.34	4.1 ± 0.80	67.72 ± 1.15
DOTCNN	22.8 ± 1.8	28.71 ± 2.25	58.32 ± 2.10	3.81 ± 0.73	62.71 ± 0.67

Table 1. Misclassification error rate for each technique viewed in this paper

Technique	Spect	Wine	FGlass	BCW	Yeast
FCM	0.51 ± 0.03	0.79 ± 0.01	0.44 ± 0.08	0.84 ± 0.03	0.13 ± 0.02
FLVQ↓	0.51 ± 0.04	0.67 ± 0.07	0.35 ± 0.10	0.76 ± 0.08	0.10 ± 0.03
FLVQ↑	0.50 ± 0.03	0.67 ± 0.03	0.31 ± 0.07	0.76 ± 0.03	0.11 ± 0.03
HALVQ	0.50 ± 0.06	0.71 ± 0.05	0.14 ± 0.05	0.50 ± 0.05	0.10 ± 0.06
DOTCNN	0.59 ± 0.07	0.81 ± 0.04	0.55 ± 0.08	0.85 ± 0.05	0.16 ± 0.02

Table 2. Partition coefficient for each fuzzy technique viewed in this paper

The higher is the value of $PC(U)$, the higher is the quality of the partition $P(U)$. The results of this study are shown in Table 2.

Finally, we are interested in calculating the entropy $H(U)$ of the results obtained by DOTCNN and fuzzy techniques studied in this paper. The partition entropy is a global measure of fuzziness in the partition defined by the matrix of membership degrees generated by the algorithm:

$$H(U) = \frac{-1}{n} \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} \log (u_{ik}) \tag{47}$$

The higher is the entropy, the more ambiguous is the situation when we want to completely assign each object to a unique cluster. The results of this study are shown in Table 3.

Technique	Spect	Wine	FGlass	BCW	Yeast
FCM	0.68 ± 0.09	0.38 ± 0.07	1.15 ± 0.06	0.27 ± 0.04	2.17 ± 0.01
FLVQ↓	0.69 ± 0.02	0.58 ± 0.04	1.37 ± 0.07	0.39 ± 0.09	2.28 ± 0.03
FLVQ↑	0.69 ± 0.06	0.58 ± 0.01	1.48 ± 0.09	0.39 ± 0.09	2.27 ± 0.08
HALVQ	0.69 ± 0.04	0.52 ± 0.08	1.95 ± 0.07	0.69 ± 0.04	2.3 ± 0.01
DOTCNN	0.58 ± 0.08	0.36 ± 0.04	0.94 ± 0.01	0.26 ± 0.08	2.03 ± 0.01

Table 3. Entropy of the fuzzy partition produced by each technique for each dataset

Technique	Codebook size			
	64	128	256	512
LVQ	20.445 ± 0.719	19.731 ± 0.41	21.721 ± 1.781	21.436 ± 1.56
FLVQ	19.825 ± 0.401	19.762 ± 0.392	18.831 ± 0.519	19.563 ± 0.415
IBFLVQ	32.679 ± 0.873	42.936 ± 0.835	34.658 ± 0.771	31.561 ± 0.995
DOTCNN	18.852 ± 0.402	19.181 ± 0.369	18.971 ± 0.539	18.397 ± 0.623

Table 4. Distortion between original "Lena" image and reconstructed image for each technique

Technique	Codebook size			
	64	128	256	512
LVQ	25.596 ± 0.964	25.513 ± 0.456	32.849 ± 0.776	33.282 ± 0.858
FLVQ	24.913 ± 0.706	24.916 ± 0.98	25.145 ± 0.49	24.347 ± 0.37
IBFLVQ	38.814 ± 0.931	42.183 ± 0.62	41.217 ± 0.696	39.472 ± 0.263
DOTCNN	24.877 ± 0.702	24.455 ± 0.385	24.182 ± 0.694	24.051 ± 0.445

Table 5. Distortion between original "Fishing-Boat" image and reconstructed image for each technique

Numerical results presented in Table 1 show the effectiveness of the proposed technique by comparison to other techniques commonly cited in the literature. These results are reinforced by those presented in Table 2, which show that the quality of the partitions generated by DOTCNN is generally better than that generated by other fuzzy techniques. Indeed, with DOTCNN, it is less ambiguous to assign each object to its cluster, as we can see by comparing the values of the entropy of the partitions generated by DOTCNN to those produced by the fuzzy techniques studied in this paper (Table 3).

4.2 Application to Image Compression

Image compression is a particular kind of image processing, aimed at reducing the number of bits required for representing the information contained in digital im-

Technique	Codebook size			
	64	128	256	512
LVQ	21.976 ± 0.279	22.823 ± 0.667	21.620 ± 0.884	21.010 ± 0.505
FLVQ	22.799 ± 0.538	20.954 ± 0.708	22.804 ± 0.982	21.379 ± 0.259
IBFLVQ	43.506 ± 0.332	41.626 ± 0.319	37.329 ± 0.788	41.365 ± 0.797
DOTCNN	20.274 ± 0.801	20.145 ± 0.98	21.266 ± 0.337	19.397 ± 0.931

Table 6. Distortion between original "Baboon" image and reconstructed image for each technique

Technique	Codebook size			
	64	128	256	512
LVQ	39.619 ± 0.979	39.711 ± 0.314	35.760 ± 0.852	38.230 ± 0.639
FLVQ	34.878 ± 0.338	29.791 ± 0.303	32.186 ± 0.741	27.756 ± 0.725
IBFLVQ	54.728 ± 0.982	61.165 ± 0.271	51.109 ± 0.612	50.919 ± 0.28
DOTCNN	30.847 ± 0.585	27.634 ± 0.957	33.371 ± 0.982	25.013 ± 0.258

Table 7. Distortion between original "Peppers" image and reconstructed image for each technique

ages. For each image I received as input data, an image compression method provides, as output, a shorter representation of I, called $C(I)$. The reverse processing, known as image decompression, takes $C(I)$ as input and generates a reconstructed image $I^{'}$ as output. If $I^{'}$ is an exact replica of I, we say that the compression method is lossless. Otherwise, the compression method is said to be lossy [14, 21]. In practice, however, lossy methods are generally preferred because of their higher compression ratios. Each image I to be compressed is called a training dataset, and each pixel $I(x, y)$ of I is called a training vector. Compressing I consists then in finding the best c-partition of the set of these training vectors. Each cluster i of this partition being represented by a prototype or codeword $v_i \in R^p$, and the set $V = \{v_1, v_2, \ldots, v_c\} \subset R^p$ of all prototypes is called codebook. To obtain the reconstructed image $I^{'}$, we replace each training vector by its closest codeword [14, 21].

In this part, we use the four 512×512 images presented in Figure 2 [20]. To obtain the training vectors that represent each of these images, we decompose it into 16 384 rectangular blocks of 4×4 pixels, and we consider each block as a training vector.

The experimental work conducted on these examples consists in applying LVQ, FLVQ, IBFLVQ and the proposed technique to the set of training vectors representing each image. The aim is to find the c code-book vectors representing the pixels of each image. These code-book vectors were used to reconstruct each image. Then, the distortion between each reconstructed image and the corresponding original one was calculated, for each algorithm, using the expression

$$D = \frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{p} (x_{kj} - x_{kj}^{'})^2 \tag{48}$$

where x_{kj} denotes the j^{th} component of the training vector x_k, and $x_{kj}^{'}$ the j^{th} component of the k^{th} pixel of the reconstructed image.

Numerical results of these experiments, for different values of the codebook size, are summarized in Tables 4–7.

A brief examination of these tables shows that the observed distortion for the proposed method is always smaller than those corresponding to the other methods.

This means that the quality of the codebook generated by the proposed technique is better than the quality of those generated by LVQ, FLVQ and IBFLVQ.

The reconstructed images are depicted in Figures 3–6 for each of the four compared methods, respectively. All of these figures show a clear difference between the qualities of the reconstructed images that is in favor of the proposed method.

5 CONCLUSION

In this paper, we introduced an unsupervised learning procedure for optimal and dynamic training of competitive neural networks. The training process is optimal in the sense that the procedure minimizes an objective function that represents an error criterion. It is dynamic because the learning rate and the competition difficulty are continously adjusted throughout iterations. Competition difficulty is introduced as a new concept by the procedure, as well as some algorithmic parameters that are heuristically initialized and adjusted. This procedure was successfully applied to two different application domains: pattern classification and image compression. For both applications, several examples of test data were used and the corresponding numerical results were favorably compared to those provided by other well-known algorithms. Further work on this research project will be concentrated on the robustness of the proposed method, mainly by finding more appropriate methods for initializing and adjusting algorithmic parameters.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments, as well as Miss Rkia Fajr and Mr. Youssef Safi, Ph. D. candidates and members of the "Soft Computing and Intelligent Systems" team of Information Processing Laboratory, Hassan Mohammedia-Casablanca University, for their helpful suggestions for improving the manuscipt.

REFERENCES

[1] Lu, C. C.—Shin, Y. H.: Neural Networks for Classified Vector Quantization of Images. Eng. Appl. Artif. Intell., Vol. 5, 1992, No. 5, pp. 451–456.

[2] Haykin, S.: Neural Networks: a Comprehensive Foundation. Prentice Hall, 1999.

[3] Lee, C.-J.—Hsiung, T. K.: Sensitivity Analysis on a Multilayer Perceptron Model for Recognizing Liquefaction Cases. Computers and Geotechnics, Vol. 36, 2009, No. 7, pp. 1157–1163.

[4] Lee, C. M.—Yang, S. S.—Ho, C. L.: Modified Back-Propagation Algorithm Applied to Decision-Feedback Equalisation. Vision, Image and Signal Processing, Vol. 153, 2006, No. 6, pp. 805–809.

[5] Pal, N. R.—Bezdek, J. C.—Tsao, E. C.-K.: Generalized Clustering Networks and Kohonen's Self-Organizing Scheme. IEEE Trans. Neural Networks, Vol. 4, 1993, pp. 549–557.

[6] Tsao, E. C.-K.—Bezdek, J. C.—Pal, N. R.: Fuzzy Kohonen Clustering Networks. Pattern Recognition, Vol. 27, 1994, No. 5, pp. 757–764.

[7] Huntsberger, T.—Ajjimarangsee, P.: Parallel Self-Organizing Feature Maps for Unsupervised Pattern Recognition. Int. J. General Syst., Vol. 16, 1989, pp. 357–372.

[8] Jain, A. K.—Murty, M. N.—Flynn, P. J.: Data Clustering: A Review. ACM computing surveys, Vol. 31, 1999, No. 3, pp. 264–323.

[9] Jain, A. K.—Dubes, R. C.: Algorithms for Clustering Data. Englewood Cliffs, N. J.: Prentice Hall, 1988.

[10] Karayiannis, N. B.—Bezdek, J. C.—Pal, N. R.—Hathaway, R. J.—Pai, P.-I.: Repairs to GLVQ: A New Family of Competitive Learning Schemes. IEEE Trans. Neural Networks, Vol. 7, pp. 1062–1071.

[11] Ahalt, S. C.—Krishnamurthy, A. K.—Chen, P.—Melton, D. E.: Competitive Learning Algorithms for Vector Quantization. Neural Networks, Vol. 3, 1990, pp. 227–290.

[12] Kohonen, T.: Self-Organization and Associative Memory. 3rd ed., Berlin: Springer-Verlag, 1989.

[13] Karayiannis, N. B.—Bezdek, J. C.: An Integrated Approach to Fuzzy Learning Vector Quantization and Fuzzy c-Means Clustering. IEEE Trans Fuzzy Syst., 1997.

[14] Tsekouras, G. E.—Antonios, M.—Anagnostopoulos, C.—Gavalas, D.—Economou, D.: Improved Batch Fuzzy Learning Vector Quantization for Image Compression. Information Sciences, Vol. 178, 2008, pp. 3895–3907.

[15] Biehl, M.—Ghosha, A.—Hammer, B.: Learning Vector Quantization: The Dynamics of Winner-Takes-all Algorithms. Neurocomputing, Vol. 69, 2006, pp. 660–670.

[16] Bezdek, J. C.: Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Plenum, 1981.

[17] Bouroumi, A.—Limouri, M.—Essad, A.: Unsupervised Fuzzy Learning and Cluster Seeking. Intelligent Data Analysis, Vol. 4, 2000, No. 3-4, pp. 241–253.

[18] Shui-Shenga, Z.—Wei-Wei, W.—Li-Huab, Z.: A New Technique for Generalized Learning Vector Quantization Algorithm. Image and Vision Computing, Vol. 24, 2006, pp. 649–655.

[19] UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2007. Available: `http://archive.ics.uci.edu/ml/`.

[20] `http://www.petitcolas.net/fabien/watermarking/image_database/`.

[21] Acharya, T.—Tsai, P. S.: JPEG2000 Standard for Image Compression Concepts. Algorithms and VLSI Architectures, Wiley Interscience, 2005.

[22] Madiafi, M.—Bouroumi, A.: A New Fuzzy Learning Scheme for Competitive Neural Networks. Applied Mathematical Sciences, Vol. 6, 2012, No. 63, pp. 3133–3144.

[23] Chattopadhyay, S.—Kumar Pratihar, D.—Chandra de Sarkar, S.: A Comparative Study of Fuzzy C-Means Algorithm and Entropy-Based Fuzzy Clustering Algorithms. Computing and Informatics, Vol. 30, 2011, No. 4, pp. 701–720.

[24] Hoonlor, A.—Szymanski, B. K.—Zaki, M. J.—Chaoji, V.: Document Clustering with Bursty Information. Computing and Informatics, Vol. 31, 2012, No. 6+, pp. 1533–1555.

[25] Chen, X.—Huo, M.—Liu, Y.: MST-Based Semi-Supervised Clustering Using M-Labeled Objects. Computing and Informatics, Vol. 31, 2012, No. 6+, pp. 1557–1574.

Mohammed Madiafi received the B. Sc. degree in electrical engineering and M. Sc. degree in information processing in 2006 and 2008, respectively, both from the Hassan II Mohammedia-Casablanca University, Morocco. He is a Ph. D. student in Modeling and Simulation Laboratory of Ben M'Sik Faculty of Sciences. His current research interests include artificial neural networks, fuzzy and intelligent systems, evolutionary algorithms, pattern classification and recognition, unsupervised learning, and data compression.

Abdelaziz Bouroumi received the "Doctorat d'Etat" degree from Mohammed V-Agdal University, Rabat, Morocco, in 2000. He is a Professor of information processing and computer science at the Hassan II Mohammedia-Casablanca University. His current research interests include fuzzy and intelligent systems, artificial neural networks, evolutionary algorithms, pattern classification and recognition, unsupervised learning, collaborative learning, and e-learning.