Expert Systems with Applications 39 (2012) 10620–10629

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Towards Hebbian learning of Fuzzy Cognitive Maps in pattern classification problems

G.A. Papakostas *, D.E. Koulouriotis, A.S. Polydoros, V.D. Tourassis

Democritus University of Thrace, Department of Production Engineering and Management, 67100 Xanthi, Greece

ARTICLE INFO

Keywords:
Fuzzy Cognitive Maps
Hebbian learning
Pattern classification
Classifier
Training
Soft computing

ABSTRACT

A detailed comparative analysis of the Hebbian-like learning algorithms applied to train Fuzzy Cognitive Maps (FCMs) operating as pattern classifiers, is presented in this paper. These algorithms aim to find appropriate weights between the concepts of the FCM classifier so it equilibrates to a desired state (class mapping). For these purposes, six different types of Hebbian learning algorithms from the literature have been selected and studied in this work. Along with the theoretical description of these algorithms and the analysis of their performance in classifying known patterns, a sensitivity analysis of the applied classification scheme, regarding some configuration parameters have taken place. It is worth noting that the algorithms are studied in a comparative fashion, under common configurations for several benchmark pattern classification datasets, by resulting to useful conclusions about their training capabilities.

1. Introduction

An increased interest about the theory and applications of the Fuzzy Cognitive Maps (FCMs) (Glykas, 2010) in engineering science is noted recently. FCMs are characterized by their ability to model the dynamics of complex systems by incorporating the casual relationships between the main concepts that describe the system. They have been used initially to model complex social and financial systems (Aguilar, 2005; Koulouriotis, Diakoulakis, Emiris, & Zopounidis, 2005), where analytical descriptions do not exist or can not be derived and recently as an inference mechanism of modern decision support systems (Azadeh, Ziaei, & Moghaddam, 2012; Büyüközkan & Vardaloğlu, 2012; Ghaderi, Azadeh, Nokhandan, & Fathi, 2012; Lee & Lee, 2012).

The design of the FCMs is highly dependent on the knowledge of the experts, which define the model's concepts and their interconnections among them. This dependency may significantly affect the modeling accuracy due to the imprecise expert's knowledge.

In order to reduce this dependency and to enhance the knowledge storing capabilities of the FCM models, learning algorithms that search for the best weight values of the concepts' interconnections have been proposed. Mainly, there are two categories of learning algorithms: (1) algorithms that use the Hebbian adaptation rule (Hebbian-like) (Dickerson & Kosko, 1994; Huerga, 2002; Li & Shen, 2004; Papageorgiou & Groumpos, 2004; Papageorgiou, Stylios, & Groumpos, 2003, 2004, 2006; Stach, Kurgan, & Pedrycz, 2008) and (2) algorithms that use evolutionary optimization mechanisms (Evolutionary-like) (Papageorgiou & Groumpos, 2005; Papageorgiou, Parsopoulos, Stylios, Groumpos, & Vrahatis, 2005; Papakostas, Boutalis, Koulouriotis, & Mertzios, 2008; Stach, Kurgan, Pedrycz, & Reformat, 2005).

Although the algorithms of the latter category ensure near-optimum solutions in the weights search space, the Hebbian-like algorithms of the former category are studied in this work, due to their popularity and their quick convergence to desirable FCM states.

The first usage of the Hebbian rule in learning FCMs was performed by Dickerson and Kosko (1994), who proposed the Differential Hebbian Learning (DHL) algorithm by using the generalized Hebbian rule introduced by Oja (1989). Since then, five other types of Hebbian learning algorithms were proposed, the Balanced DHL (BDHL) (Huerga, 2002), the Nonlinear Hebbian Learning (NHL) (Papageorgiou & Groumpos, 2004; Papageorgiou et al., 2003), the Improved NHL (INHL) (Li & Shen, 2004), the Active Hebbian Learning (AHL) (Papageorgiou et al., 2004, 2006) and the Data-Driven Nonlinear Hebbian Learning (DD-NHL) (Stach et al., 2008) ones.

Although the above Hebbian-like algorithms have been proposed as learning procedures of the FCM model for system modeling tasks, and in decision support systems, less work that investigates their performance in training an FCM as pattern classifier is reported in the literature.

Therefore, despite the review of the Hebbian-like learning algorithms from the literature, the main contribution of this work is the comparison of these algorithms in training an FCM pattern classifier under the same conditions. Moreover, the fundamental operational principles of the pattern classification scheme using

* Corresponding author.
E-mail addresses: gpapakos@ee.duth.gr (G.A. Papakostas), jimk@pme.duth.gr (D.E. Koulouriotis), athapoly@pme.duth.gr (A.S. Polydoros), vtourasi@pme.duth.gr (V.D. Tourassis).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.02.148

the FCM classifier, regarding its structure, the class mapping, the weight initialization, the training data ordering and the need for hidden nodes, are also investigated. Furthermore, the observed performance of the Hebbian-like algorithms is also compared with that of a typical Evolutionary-like algorithm that uses a Genetic Algorithm in order to find the appropriate weight set of the FCM classifier.

It is worth pointing out that in discussing the influence of the previously mentioned classification principles, the FCMs become more similar to Neural Networks (NNs) models. However, their main difference, the rule according to which the model's output is generated, still remains. The incorporation of common to NNs classification principles in FCMs operation is studied herein in order to improve their knowledge storing capabilities needed in demanding classification applications.

The paper is organized, by describing shortly the fundamental theory of the FCMs in Section 2 and by discussing the Hebbian-like learning algorithms in Section 3. In Section 4 the application of the FCMs as pattern classifiers is concerned and a detailed sensitivity analysis of the pattern classification scheme has taken place. An extensive experimental study, with well-known from the literature pattern classification benchmarks, is presented in Section 5, while the performance of the most efficient Hebbian-like algorithm is compared with that of an Evolutionary-like algorithm in Section 6. Finally, Section 7 summarizes the main conclusions drawn from the previous experimental study and future research directions are determined.

2. Fuzzy Cognitive Maps (FCMs)

Fuzzy Cognitive Maps (FCMs) are fuzzy signed directed graphs with feedback. They were proposed by Kosko (1986) as a modeling methodology of complex systems, able to describe the casual relationships between the main factors (concepts) that determine the dynamic behavior of a system.

In their graphical presentation, FCMs have the form of a diagram with nodes having interconnections between them, as illustrated in the following Fig. 1. As it can be seen from Fig. 1, an FCM consists of nodes (concepts), C_i, $i = 1,2,3,\ldots,N$, where N is the total number of concepts, which are characteristics, main factors, or properties of the system being modeled. The concepts are interconnected through arcs having weights that denote the cause and effect relationship that a concept has on the others. There are three possible types of casual relationships between two concepts C_i and C_j: (i) positive, meaning that an increase or decrease in the value of a cause concept results in the effect concept moving in the same direction and it is described by a positive weight W_{ij}, (ii) negative, meaning that changes in the cause and effect concepts take place in opposite directions, and the weight W_{ij} has negative sign and (iii) no relation, with the interconnection weight being zero. The weight value, e.g. W_{ij}, denotes the strength of how casual concept C_i causes C_j and takes values in the range $[-1,1]$.

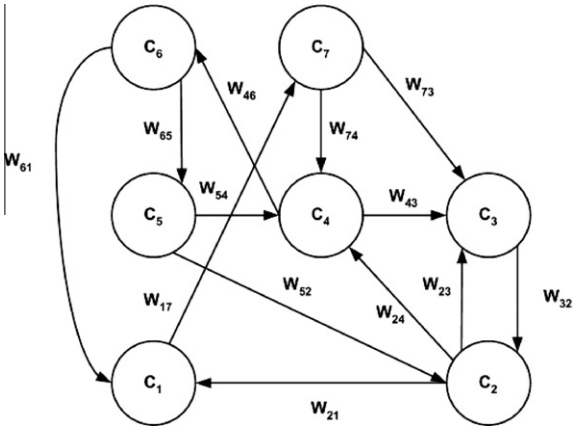

Fig. 1. A simple Fuzzy Cognitive Map.

At each time, the value A_i of each concept C_i is calculated, by summarizing the influences of all the other concepts to this and by squashing the overall impact using a barrier function f, according to the following rule:

$$A_i^{(t+1)} = f\left(A_i^{(t)} + \sum_{i=1, i\neq j}^{N} W_{ji} A_j^{(t)}\right) \tag{1}$$

where A_i^{t+1} and A_i^t are the values of concept C_i at times $t+1$ and t, respectively, A_i^t the value of concept C_j at time t, W_{ji} the weight value of the interconnection with direction from concept C_j to concept C_i, and f the barrier function used to restrict the concept value into a specific range, usually in the range $[0,1]$ or $[-1,1]$.

In each step, a new state of the concepts is derived according to (1) and after a number of iterations, the FCM may arrive in one of the following states: (1) a fixed equilibrium point, (2) a limited cycle and (3) a chaotic behavior.

When the FCM arrives in a fixed equilibrium point, we can conclude that the FCM has converged and the final state corresponds to the actual system state in which the system concludes, when the initial values of concepts are applied.

The design of the FCMs is significantly based on the experience of some experts, who have the knowledge of the system being modeled and provide the weight values for the interconnection between concepts. In more flexible FCM structures these weights are being found during a learning procedure, in a similar way as in the case of neural networks training. For this reason many researchers have borrowed and adjusted algorithms from the neural network theory and proposed new learning procedures.

In the present paper, six Hebbian-like learning algorithms (Dickerson & Kosko, 1994; Huerga, 2002; Li & Shen, 2004; Papageorgiou & Groumpos, 2004; Papageorgiou et al., 2003, 2004, 2006; Stach et al., 2008) are used in order to find the optimal weight set of the FCMs when they operate as pattern classifiers and their performance is compared to each other and with that of a typical Evolutionary-like learning algorithm.

3. Hebbian-like learning algorithms

Hebbian learning constitutes an unsupervised technique initially applied on the training of artificial neural networks (Haykin, 1999). The main feature of this learning rule is that the change of a synaptic is computed by taking into account the presynaptic and postsynaptic signals flow towards each processing unit (neuron) of a neural network and is defined by the following Eq. (2):

$$\Delta w_{ij} = \eta y_i(n) x_j \tag{2}$$

where η is a positive constant that determines the rate of learning and x_j and y_i is the presynaptic and postsynaptic signals. The above rule has been modified by Oja (1989) in order to solve the stability problems imposed in Eq. (2), by introducing a generalized form of the Hebbian rule called Oja's rule, having the following form:

$$\Delta w_{ij} = \eta y_i(n)\left(x_j(n) - y_i(n) w_{ij}(n)\right) \tag{3}$$

Based on the Oja's rule, several Hebbian-like learning algorithms have been proposed in the literature, initially for the case of neural networks and recently, after appropriate modifications, for the training of FCMs.

The common processing steps for the Hebbian-like algorithms are as follows:

Step 1: For a given initial state A_0 and weight set W_0.
Step 2: For each iteration k.
Calculate the concept values A_k by using Eq. (1).
Update weights W_k by using the weight updating rule of the algorithm.
Until some termination criteria are met.
Step 3: Return the final weights W_{final}.

A short description of the Hebbian-like algorithms, having the form of Eq. (2), proposed as learning procedures that find an appropriate set of interconnection weights (connection matrix) subject to a system's requirements is presented hereafter in chronological order.

3.1. Differential Hebbian Learning (DHL)

The DHL algorithm was the first Hebbian learning algorithm proposed by Dickerson and Kosko (1994) in order to find better interconnection weights than those provided by the experts participated to the FCM design. This algorithm correlates the changes in the concept nodes of the model in order to modify their weights, according to the following Eq. (4):

$$w_{ij}^{(t+1)} = \begin{cases} w_{ij}^{(t)} + \mu_t\left(\Delta A_i^{(t)}\Delta A_j^{(t)} - w_{ij}^{(t)}\right), & \Delta A_i^{(t)} \neq 0 \\ w_{ij}^{(t)}, & \Delta A_i^{(t)} = 0 \end{cases} \tag{4}$$

where

$$\Delta A_i^{(t)} = A_i^{(t)} - A_i^{(t-1)} \tag{5}$$

Moreover, parameter μ_t refers to the learning rate which decreases with the algorithm's iterations as follows:

$$\mu_t = 0.1\left[1 - \frac{t}{1.1N}\right] \tag{6}$$

where t is the current iteration and N is a constant that ensures the learning rate not to take negative values. An acceptable value of this constant is the maximum possible number of the algorithm's iterations.

3.2. Balanced Differential Hebbian Learning (BDHL)

Another Hebbian-like learning algorithm proposed by Huerga (2002), the Balanced DHL (BDHL), as a way to improve the knowledge representation of the FCM model. This algorithm is described by a weight updating rule according to each weight is updated by considering not only the concepts that it connects, but also all the other concepts of the FCM. In this way the BDHL algorithm, assumes that the cause-effect relations between the concepts are distributed among all the concepts of the map and only between the connected ones. The weight updating rule of the BDHL is as follows:

$$w_{ij}^{(t+1)} = \begin{cases} w_{ij}^{(t)} + A_i^t/N & i = j \\ w_{ij}^{(t)} + \mu_t\left[\dfrac{\Delta A_i^{(t)}/\Delta A_j^{(t)}}{\sum_{k=1}^{n}{}_{\Delta A_i^{(t)}\Delta A_j^{(t)}>0} \Delta A_i^{(t)}/\Delta A_k^{(t)}} - w_{ij}^{(t)}\right] & i \neq j \quad \Delta A_i^{(t)}\Delta A_j^{(t)} > 0 \\ w_{ij}^{(t)} + \mu_t\left[\dfrac{-\Delta A_i^{(t)}/\Delta A_j^{(t)}}{\sum_{k=1}^{n}{}_{\Delta A_i^{(t)}\Delta A_j^{(t)}>0} \Delta A_i^{(t)}/\Delta A_k^{(t)}} - w_{ij}^{(t)}\right] & i \neq j \quad \Delta A_i^{(t)}\Delta A_j^{(t)} < 0 \end{cases} \tag{7}$$

In the above rule μ_t is the learning rate of Eq. (6) and N the same parameter as in the case of the DHL algorithm.

3.3. Nonlinear Hebbian Learning (NHL)

The next Hebbian learning algorithm that mostly looks like the Oja's rule is the Nonlinear Hebbian Learning (NHL) proposed by Papageorgiou et al. (2003), Papageorgiou and Groumpos (2004), defined as:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \eta A_j^{(t)}\left(A_i^{(t)} - A_j^{(t)}w_{ij}^{(t)}\right) \tag{8}$$

where η is the learning rate. Besides the above basic formulation of the NHL algorithm two more extensions were introduced by Papageorgiou et al. (2006), having the following forms:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \eta A_j^{(t)}\left(A_i^{(t)} - \mathrm{sgn}\left(w_{ij}^{(t)}\right)A_j^{(t)}w_{ij}^{(t)}\right) \tag{9}$$

where sgn() corresponds to the sign function that returns the sign of a quantity and it is used to maintain the sign of the corresponding weight. The last NHL form is derived by adding a weight decay parameter γ in the update rule, as follows:

$$w_{ij}^{(t+1)} = \gamma w_{ij}^{(t)} + \eta A_j^{(t)}\left(A_i^{(t)} - \mathrm{sgn}\left(w_{ij}^{(t)}\right)A_j^{(t)}w_{ij}^{(t)}\right) \tag{10}$$

In order to decrease the number of the free parameters, only the NHL version of Eq. (9) will be considered in the experimental section of this work.

Despite the new weight update rule of the NHL algorithm, the usage of two termination criteria (Papageorgiou et al., 2006) of the learning procedure was also a significant novel feature.

These two criteria have the following form:

$$F_1 = \sqrt{\sum_{i=1}^{n}\left(OC_i^{(t)} - T_i\right)^2} \tag{11}$$

$$F_2 = \left|OC_i^{(t+1)} - OC_i^{(t)}\right| < e \tag{12}$$

where $OC_i^{(t)}$ is the ith output concept at iteration t and T_i the target value of the output concept, which for the case of a constrained concept in the range $[T_i^{\min}, T_i^{\max}]$, is equal to $T_i = \frac{T_i^{\min}+T_i^{\max}}{2}$. The objective of the NHL learning algorithm is to find a set of interconnection weights that minimizes the cost functions F_1 and F_2 (subject to a predefined tolerance e), so the FCM model can reach a desired state. While the first criterion shows the error of the output concepts from the desired values, the second one ensures that the FCM model equilibrates to a final state.

3.4. Improved Nonlinear Hebbian Learning (INHL)

An Improved Nonlinear Hebbian Learning algorithm was proposed by Li and Shen (2004), by adding an additional quantity in the weight updating rule, called *impulse term* ($\alpha^{(t+1)}\Delta w_{ij}^{(t)}$) in order to avoid possible trap to local minima in regions where the error surface is flat. The weight updating rule in this case takes the following form:

$$\Delta w_{ij}^{(t+1)} = \alpha^{(t+1)}\Delta w_{ij}^{(t)} + \eta^{(t+1)}\left(z^2\right)^{(t)}\left(1 - z^{(t)}\right)\left(A_j^{(t)} - A_i^{(t)}w_{ij}^{(t)}\right) \tag{13}$$

where z is equal to

$$z^{(t)} = \frac{1}{\left(1 + e^{-A_i^{(t)}}\right)} \tag{14}$$

Additionally, $\alpha \in [0, 1)$ is a small positive scalar called *impulse constant*, which reflects the acceleration on descending the error surface.

3.5. Active Hebbian Learning (AHL)

A common feature of the previous learning algorithms is that in each iteration the concepts of the FCM model are updated by using Eq. (1), synchronously. The Active Hebbian Learning (AHL) algorithm considers that the concept nodes are activated asynchronously by a specific sequence. In this way the equilibrium point is reached by considering different times of nodes' activation, a mechanism that is useful in systems where the concepts are activated based on a specific activation sequence.

Moreover, unlike the previous algorithms where only the non-zero weights are being updated, in the case of the AHL algorithm all weights except those of the diagonal are updated.

Based on the AHL algorithm, the nodes of the FCM model is distinguished to *Activated* and *Activation concepts*, by the former ones being the nodes that firstly activated by causing the activation of the latter. In this case the activated concepts are updated through the following rule:

$$A_i^{(t+1)} = f\left(A_i^{(t)} + \sum_{i=1, i\neq j}^{N} W_{ji} A_j^{act(t)}\right) \tag{15}$$

where A_i are the value of the C_i activated concept and A_j^{act} the values of the activation concepts that influence the concept C_i. The weights are updated according to the following rule:

$$w_{ij}^{(t+1)} = (1-\gamma^{(t)}) w_{ij}^{(t)} + \eta^{(t)} A_i^{(t)} \left(A_j^{(t)} - w_{ij}^{(t)} A_i^{act(t)}\right) \tag{16}$$

where the learning rate η and the weight decay γ are decreased exponentially with the algorithm's iteration number (activation–simulation cycles) as follows:

$$\begin{aligned} \eta^{(t)} &= b_1 e^{(-\lambda_1 t)} \\ \gamma^{(t)} &= b_2 e^{(-\lambda_2 t)} \end{aligned} \tag{17}$$

where $0.01 < b_1 < 0.09$, $0.1 < \lambda_1 < 1$ and b_2, λ_2 are positive constants determined by trial and error. Furthermore, the two termination criterion of the NHL is also applied in the case of AHL in order to derive a desired equilibrium point of the FCM model.

3.6. Data-Drive Nonlinear Hebbian Learning (DD-NHL)

A modified version of the NHL algorithm was proposed by Stach et al. (2008), which incorporated historical data of the model into the learning procedure. This algorithm called Data-Driven Nonlinear Hebbian Learning (DD-NHL) and differs to the original NHL in two steps:

Difference 1: Instead of computing the concept values in each iteration by using Eq. (1), the concept values are provided by the historical data. These data is generated by applying the NHL algorithm in a first stage.
Difference 2: The termination criterion F_1 is replaced by measuring the output concepts that lie inside the desired target values. More precisely, the candidate weight set is used to simulate the FCM model and if the output concepts equilibrated to a desired state the learning is completed (with simultaneous satisfaction of the F_2 criterion), otherwise the procedure continues.

4. Pattern classifier – An FCM approach

Apart from the application of the FCMs in modeling complex systems, recently they have been applied as decision makers (Azadeh et al., 2012; Büyüközkan & Vardaloğlu, 2012; Ghaderi et al., 2012; Lee & Lee, 2012) and pattern classifiers (Papakostas & Koulouriotis, 2010, chap. 12; Papakostas et al., 2008; Peng, Yang, & Fang, 2008), in difficult problems of the engineering life, where a decision about the identity of some input data, need to be drawn.

However, the incorporation of the FCM models in pattern classification tasks needs some modifications regarding the structure and the operational principles according to the applied pattern classification scheme.

The basic principles that describe the main operation of an FCM classifier called FCMper, has been demonstrated in a past work of the authors' (Papakostas et al., 2008). Based on those definitions, the following one is of major importance since it shows the modification to the operation of the conventional FCM in modeling:

Definition 1. If a common set of interconnection weights can be found, which can lead the FCM to specific equilibrium states for different initial set of input concepts, then this model can map the patterns to the classes they belong to; in other words it can recognize them.

The equilibrium states referred to the above definition, correspond to the classes' labels.

Furthermore, most of the learning algorithms applied so far (Hebbian-like and Evolutionary-like) search a weight set which can lead the FCM from a specific initial state to a desired equilibrium point. However, in the case of patterns classification the learning algorithm should work as following:

Definition 2. The learning algorithm for the case of FCMper, has to find a common weight set which, for different initial state of the input concepts, the FCMper equilibrates to different states (classes/categories).

In order to analyze the behavior of the FCM models in classifying known patterns, a sensitivity analysis regarding the influence some configuration parameters have to the classification accuracy of the FCMper has to be conducted. Through this study an appropriate pattern classification scheme is expected to be designed and its free configuration parameters to be decided.

For the needs of the sensitive analysis presented in this section, three well-known pattern classification datasets (UCI Machine Learning Repository., xxxx) have been selected and their properties are illustrated in the following Table 1.

Each one of the above datasets describes data that belong to three classes, while the number of discrimination features used as classifier inputs in order to distinguish each class (attributes) and the number of data samples (instances) varies from dataset. Noted that in all the following classification experiments the NHL learning algorithm is applied and the 75% of the datasets' data is used to train the classifiers, while the remaining 25% is used to test them. Finally, due to the random choice of the weights the experiments have been executed 10 times and statistical measurements (minimum, maximum, mean, standard deviation) are used to describe the behavior of each model. Finally, the "*correct classified samples to total number of samples ratio*" called *classification rate* (CRate) is used as performance index in order to describe quantitatively the classifiers classification capabilities.

4.1. Classifier structure

Before studying some alternative FCM classifier's structures and the influence of the structure on classifier's accuracy, it is useful to introduce a new representation of the bidirectional casual relation-

Table 1
Datasets' properties.

Dataset	Instances	Attributes	Classes
Wine	178	13	3
Iris	150	4	3
Thyroid	216	5	3

ship between two concepts of an FCM model. On the left side of Fig. 2, the conventional representation of the bidirectional casual relationship between the two concepts is replaced by the one presented on the right side.

In the Fig. 2, the weights W_{12} and W_{21} represent the strength of how C_1 influences C_2 and vice versa. This representation will facilitate the illustration of the structures that follow.

In all the proposed structures the concepts are divided into two types, the input and output concepts, (ICs) and (OCs), respectively. The input concepts are considered to be steady state concepts, meaning that their values remain unchanged, they are not updated according to the inference rule Eq. (1), but they just contribute to the updating procedure of the rest concepts.

The investigated structures of the FCM classifier are depicted in Fig. 3, where N is the number of ICs and M the number of OCs.

As it can be realized from this figure, these three structures differ as to whether there are connections between the output concepts (Fig. 3(b)) or input concpets (Fig. 3(c)) or anywhere in both (Fig. 3(a)).

Moreover, it is worth mentioning that in both structures Fig. 3(a) and (b) the input concepts are steady, while in the case of the third structure these concepts need to be updated (so the connections be acceptable) based on Eq. (1), although their values during the inference process are of no interest.

The classification performance of the three classifier structures for the case of the datasets of Table 1, are summarized in Table 2 that follows.

Each experiment has been executed 10 times and the minimum (Min.), maximum (Max.), mean classification rates (Crates) and standard deviation (Stdev.) have been calculated and presented in Table 2. Despite the type of concepts interconnections, the studied structures have input concepts equal to the number of datasets' attributes while the number of output concepts is equal to the number of datasets' classes.

From the classification results of Table 2, it can be concluded that the three structures show similar performance, with the first one (Fig. 3(a)) being the most advantageous due to its simplicity. The addition of more interconnections between the input or output concepts, seems to not improve the overall classification performance of the FCM classifier, although more free adjustable parameters (weights) need to be found.

It has to be noted that in all the experiments the classifiers' weights are randomly initialized (a common policy in neural classifiers), the training data are presented by their class order, while the classes are decided by clustering the values of the output concepts.

4.2. Class mapping

A common way to map the outputs of the classifier to a problem's classes is to assign a specific class to a single output (*Class per Output – CpO*), that with the highest value. However, this procedure assumes that the number of classifier outputs is equal to the number of problem's classes and the classifier is supervised trained.

Although the Hebbian-like algorithms discussed in the previous section constitute unsupervised learning methods, the existence of the criterion F_1 (Eq. (11)), where a specific target need to be achieved, allows the application of such class mapping.

The classification performance of this class mapping, for 10 executions of each experiment, is presented in Table 3.

A short look at these results leads to the conclusion that such a class mapping approach is not effective, since the classification rate is very low in all the cases. This is due to the fact that there is not any connection between the criterion F_1 and the weight updating rule of the Hebbian-like algorithms, as it might exist in the case of a supervised training.

An alternative class mapping approach (*By Threshold – ByT*), commonly used in unsupervised learning procedures, is the extraction of specific output thresholds that separate the problem's classes to each other. This approach has been extensively applied in decision support systems (Azadeh et al., 2012; Büyüközkan & Vardaloğlu, 2012; Ghaderi et al., 2012), where the number of classes is quite small and only one output concept is considered. However, if the number of problem's classes is big, for an N-class problem $N-1$ thresholds have to be found, the process of threshold extraction can be a laborious task.

This deficiency of the *ByT* class mapping can be overcome by applying a more reliable procedure consisting of clustering the values of the output concepts (*Class of Minimum Distance – CoMD*) and finding the centers of the clusters during the training phase. In the testing, a test sample is decided to belong to a specific class whose output has the less distance from classes' centers.

The *ByT* and *CoMD* class mappings have been applied to the three benchmark datasets by using the classifier structure of Fig. 3(a), while the remaining configuration parameters are kept the same with the previous subsection and their classification performance is presented in Table 4.

The results show that the proposed *CoMD* class mapping approach has a slight superiority to the *ByT* one, since the former classifies the patterns more correctly (about 2%) than the latter. While this superiority is marginal, the simplicity of the *CoMD* when compared with that of the *ByT* makes it an effective class mapping procedure.

4.3. Weight initialization

While for the case of system modeling the initial weights of the interconnections between the concepts are provided by the experts during the system design phase, in the case of pattern classification applications the weights have to be initialized appropriately in order to boost the training of the classifier.

The issue of weights initialization has occupied researchers in neural computation for many years, by proposing several initialization methods (Papakostas, Boutalis, Samartzidis, Karras, & Mertzios, 2008). In this work, a short study about the usefulness of the random selection of the initial weights is taking place. In this way, the classification performance of an FCM classifier with ran-

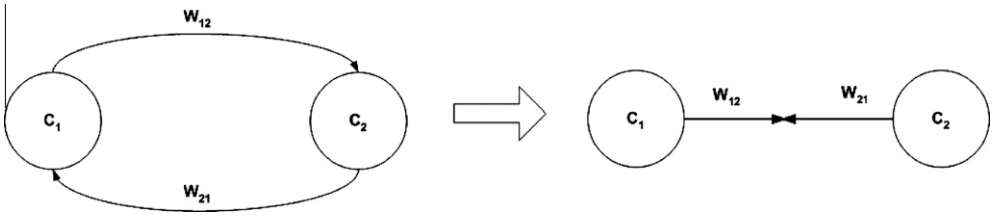

Fig. 2. Bidirectional casual relationship between concepts C_1 and C_2.

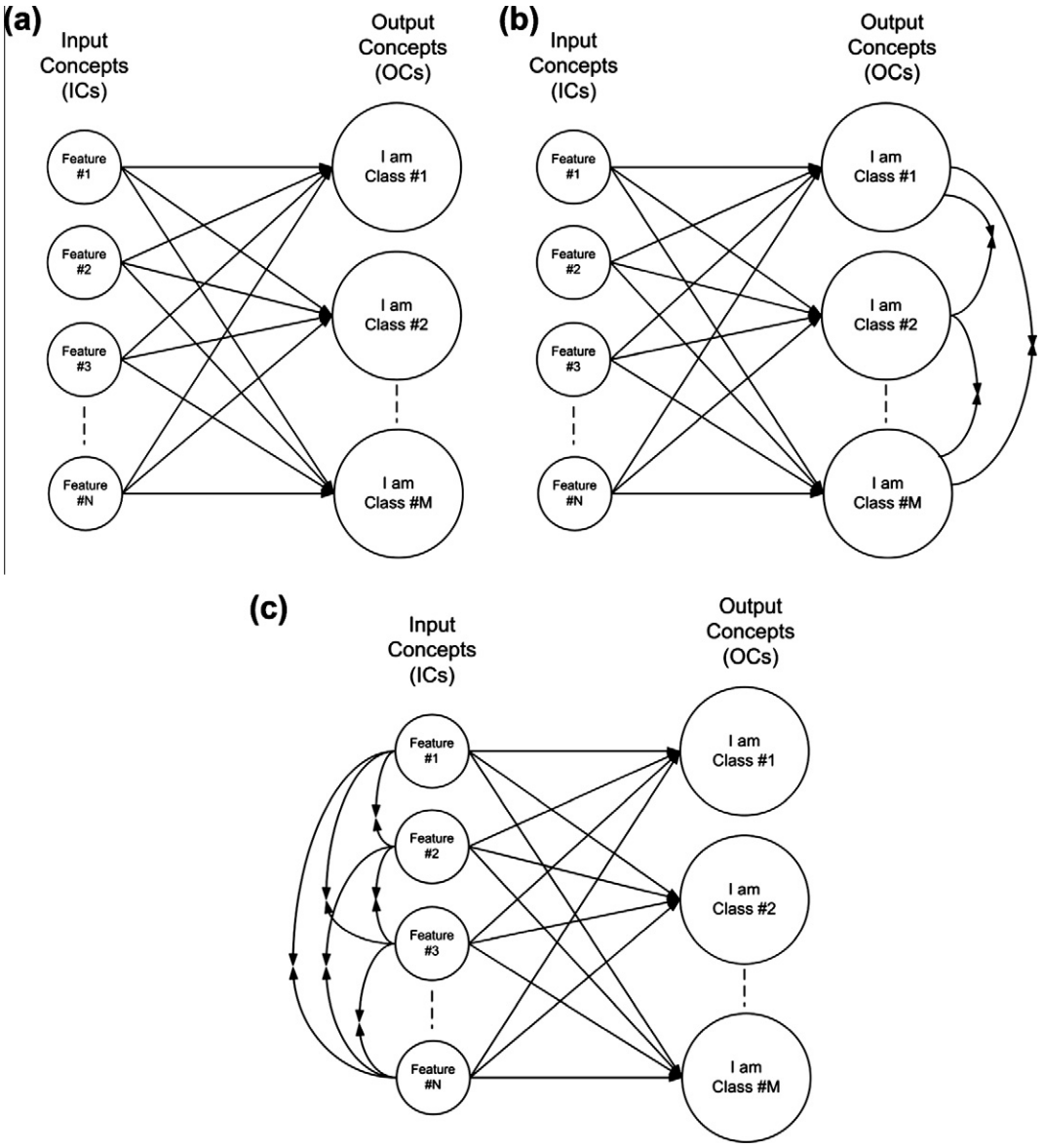

Fig. 3. Structures of the FCM classifier.

Table 2
Classification results of the three FCM classifier structures.

Dataset	Structure	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.
Wine	Fig. 3(a)	67.42	68.54	69.10	0.66
	Fig. 3(b)	67.42	68.43	69.10	0.66
	Fig. 3(c)	67.42	67.98	68.54	0.25
Iris	Fig. 3(a)	72.00	72.80	74.67	1.22
	Fig. 3(b)	70.67	71.87	74.67	1.11
	Fig. 3(c)	70.67	74.27	74.67	1.20
Thyroid	Fig. 3(a)	61.86	64.28	66.51	1.48
	Fig. 3(b)	61.86	63.40	65.58	1.32
	Fig. 3(c)	54.88	63.58	64.65	2.91

domly initialized weights is compared with that of a classifier having zero initial weights and the results are summarized in Table 5.

From the results of Table 5, it is obvious that the random initialization of the FCM classifier's weights constitute a good choice as compared to the zeroed initial weights, since it helps the learning algorithm to find a more optimal weight set, able to improve the classification accuracy of the classifier.

However, more analysis regarding the influence of the weights initialization procedure has to be performed in the future, by applying several initialization methods (Papakostas et al., 2008) already tested in neural classifiers with significant results.

Table 3
Classification results of the *CpO* class mapping approach.

Dataset	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.
Wine	26.96	32.69	39.88	5.35
Iris	33.33	33.33	33.33	0.00
Thyroid	13.95	15.58	16.28	1.06

4.4. Training data ordering

Another configuration parameter of the pattern classification scheme that needs to be specified is the order according to which the training data are provided to the input concepts of the FCM classifier. Two different strategies can be followed regarding the

Table 4
Classification results of the *ByT* and *CoMD* class mapping approaches.

Dataset	Class mapping	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.
Wine	*CoMD*	65.38	77.00	90.77	6.82
	ByT	54.61	74.00	90.77	12.36
Iris	*CoMD*	52.00	80.70	90.00	10.94
	ByT	56.00	78.80	88.00	13.19
Thyroid	*CoMD*	49.23	72.31	86.00	14.33
	ByT	49.23	71.69	84.62	12.57

Table 5
Classification performance of the weights initialization procedures.

Dataset	Weight initialization	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.
Wine	*Random*	67.42	67.75	68.54	0.44
	Zero	67.42	67.42	67.42	0.00
Iris	*Random*	64.00	69.26	86.66	8.96
	Zero	33.33	33.33	33.33	0.00
Thyroid	*Random*	61.86	63.67	65.58	1.53
	Zero	49.76	49.76	49.76	0.00

training data ordering, the *shuffle ordering*, where the data are presented randomly and the *class ordering*, where the data are presented class by class.

The following Table 6 summarizes the classification results of the FCM classifier having the structure of Fig. 3(a), the *CoMD* as class mapping procedure and random initial weights.

A short look at the above results one can conclude that the data ordering does not affect significantly the classification performance of the FCM classifier, although the shuffle ordering shows quite better behavior.

4.5. Hidden nodes

In the theory of neural networks, it has been proved that the addition of hidden neurons in several hidden layers, can lead the network to high classification rates, although the exact number of needed neurons is not a priori known. This happens due to the fact that the addition of hidden neurons increases the number of the adjustable weights and therefore the knowledge capacity of the network is significantly increased. In the same way as in the case of neural classifiers, there is a need to examine the dependency of the FCM classifier's performance to the number of the hidden concepts. In this case the hidden concepts do not have any physical meaning, but just change the casual effect relations of the useful concepts.

The general form of an FCM classifier with K hidden concepts is illustrated in Fig. 4.

Moreover, the classification performance of the classifier for various hidden concepts, for the case of the three datasets is plotted in Fig. 5. It is worth mentioning that the hidden concepts takes

Table 6
Classification performance of the ordering procedures.

Dataset	Training data ordering	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.
Wine	*Shuffle ordering*	67.41	67.97	68.53	0.43
	Class ordering	67.42	67.81	68.54	0.44
Iris	*Shuffle ordering*	72.00	73.60	74.67	1.31
	Class ordering	70.67	71.6	72.00	0.61
Thyroid	*Shuffle ordering*	61.86	63.67	65.58	1.53
	Class ordering	61.86	63.58	65.58	1.48

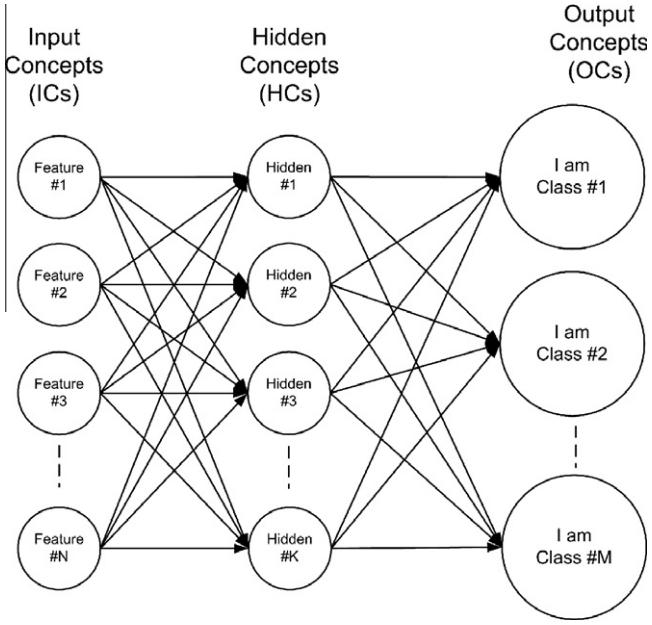

Fig. 4. An FCM classifier with K hidden concepts.

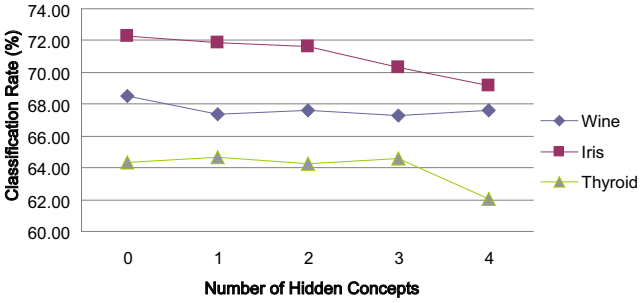

Fig. 5. Classification performance versus number of hidden concepts.

zero initial values and are updated by applying Eq. (1), if they were normal data concepts.

As it can be seen from the above plots, the additional hidden concepts do not improve the classification accuracy of the FCM classifier, but rather they adversely affect it. Therefore, the hidden concepts seem not to increase the knowledge capacity of the model as they do in neural networks.

5. Algorithms' performance analysis

While the performance of the Hebbian-like learning algorithms has been studied in the past by the authors (Papakostas, Polydoros, Koulouriotis, & Tourassis, 2011) for the case of system modeling, their performance in pattern classification problems, is examined in this section. For this purpose a set of appropriate experiments has been arranged, by using the pattern classification scheme derived by the previous sensitivity analysis. Therefore, the FCM classifier applied in this section has the structure of Fig. 3(a), the *CoMD* method is applied to map the outputs to specific classes, the weights are initialized randomly, and the training data are provided with shuffle, while no hidden concepts are inserted into the FCM model.

The configuration parameters of the Hebbian-like algorithms are summarized in the following Table 7.

Table 7
Algorithms' settings.

Algorithm	Parameters
DHL	$N = 100$, $e = 0.005$
BDHL	$N = 100$, $e = 0.005$
NHL	$\eta = 0.03$, $\gamma = 0$, $e = 0.005$
INHL	$\eta = 0.03$, $\alpha = 0.005$, $e = 0.005$
AHL	$b_1 = 0.01$, $\lambda_1 = 0.1$, $\gamma = 0$, $e = 0.005$
DD-NHL	$\eta = 0.03$, $\gamma = 0$, $e = 0.005$

Along with the three datasets used in the previous investigation, five more datasets are selected from UCI-Machine Learning Repository (xxxx) in order to compare the performance of the Hebbian-like algorithms with the configuration parameters derived from the preceding section. The main properties of all the datasets used in this experimental study are summarized in the following Table 8, while the algorithms' classification performance is presented in Tables 9–16.

Table 8
Datasets' properties.

Dataset	Instances	Attributes	Classes
Pima	768	8	2
Hepatitis	155	19	2
Parkinson	195	22	2
Wine	178	13	3
Iris	150	4	3
Thyroid	216	5	3
Vehicles	188	18	4
Glass	214	9	6

Table 9
Algorithms' performance for the case of Pima dataset.

Algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.	Iterations
Dataset – Pima					
DHL	55.10	55.10	55.10	0.00	12
BDHL	73.44	73.44	73.44	0.00	12
NHL	70.31	71.72	73.44	1.23	12
INHL	59.38	59.38	59.38	0.00	12
AHL	52.60	54.43	56.77	1.28	9
DD-NHL	65.10	68.75	75.52	2.54	12

Table 10
Algorithms' performance for the case of Hepatitis dataset.

Algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.	Iterations
Dataset – Hepatitis					
DHL	41.03	41.03	41.03	0.00	21
BDHL	41.03	41.03	41.03	0.00	23
NHL	23.08	25.38	25.64	0.77	22
INHL	25.64	25.64	25.64	0.00	24
AHL	25.64	41.03	58.97	10.32	6
DD-NHL	25.64	28.46	46.15	6.33	22

Table 11
Algorithms' performance for the case of Parkinson dataset.

Algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.	Iterations
Dataset – Parkinson					
DHL	55.10	55.10	55.10	0.00	27
BDHL	57.14	57.14	57.14	0.00	26
NHL	40.82	43.88	55.10	5.02	27
INHL	81.63	81.63	81.63	0.00	10
AHL	75.51	80.82	85.71	3.32	7
DD-NHL	28.57	57.35	75.51	21.86	27

Table 12
Algorithms' performance for the case of Wine dataset.

Algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.	Iterations
Dataset – Wine					
DHL	60.00	60.00	60.00	0.00	16
BDHL	60.00	60.00	60.00	0.00	16
NHL	53.33	53.78	57.78	1.33	19
INHL	40.00	40.00	40.00	0.00	18
AHL	53.33	64.89	75.56	6.80	7
DD-NHL	53.33	57.33	64.44	4.07	19

Table 13
Algorithms' performance for the case of Iris dataset.

Algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.	Iterations
Dataset – Iris					
DHL	83.33	83.33	83.33	0.00	9
BDHL	83.33	83.33	83.33	0.00	9
NHL	80.56	81.11	83.33	1.11	9
INHL	69.44	69.44	69.44	0.00	9
AHL	80.56	87.78	97.22	5.58	10
DD-NHL	69.44	72.22	77.78	2.15	10

Table 14
Algorithms' performance for the case of Thyroid dataset.

Algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.	Iterations
Dataset – Thyroid					
DHL	58.49	58.49	58.49	0.00	10
BDHL	58.49	58.49	58.49	0.00	10
NHL	58.49	60.75	64.15	2.77	10
INHL	58.49	58.49	58.49	0.00	10
AHL	49.06	66.79	77.36	7.41	11
DD-NHL	16.98	44.53	58.49	15.24	10

Table 15
Algorithms' performance for the case of Vehicles dataset.

Algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.	Iterations
Dataset – Vehicles					
DHL	32.61	32.61	32.61	0.00	24
BDHL	32.61	32.61	32.61	0.00	24
NHL	30.43	33.70	43.48	3.54	24
INHL	26.09	26.09	26.09	0.00	22
AHL	26.09	30.22	36.96	3.14	6
DD-NHL	28.26	31.96	41.30	4.96	24

Table 16
Algorithms' performance for the case of Glass dataset.

Algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.	Iterations
Dataset – Glass					
DHL	1.89	1.89	1.89	0.00	18
BDHL	1.89	1.89	1.89	0.00	17
NHL	13.21	15.47	16.98	1.85	17
INHL	20.75	20.75	20.75	0.00	17
AHL	24.53	34.53	45.28	5.91	9
DD-NHL	1.89	4.91	5.66	1.25	17

A careful study of the above tables depicting the algorithms' classification accuracy, can lead to the conclusion that the AHL algorithm shows a remarkable performance. This can also be derived easily, by measuring the algorithms' ranking by taking into

Table 17
Algorithms' ranking.

Dataset	DHL	BDHL	NHL	INHL	AHL	DD-NHL
Pima	5	1	2	4	6	3
Hepatitis	1	1	4	3	1	2
Parkinson	5	4	6	1	2	3
Wine	2	2	4	5	1	3
Iris	2	2	3	5	1	4
Thyroid	3	3	2	3	1	4
Vehicles	2	2	1	5	4	3
Glass	5	5	3	2	1	4
Total scores	25	20	25	28	17	26

account the mean classification rate. In this way the number 1 is assigned to the algorithm with the best rate, the number 2 to the next algorithm with the highest rate and so on, as Table 17 shows.

It is obvious that the algorithm with the lowest total score is the most efficient one, which in our case is the AHL one having the highest classification rate in 5/8 datasets. Moreover, this algorithm converges to an equilibrium point 3–4 times faster than the other ones, but it is sensitive to the initial weights since its standard deviation is quite high.

The second most efficient algorithm is the BDHL, has the highest classification rate in 2/8, while the rest ones give similar results in the same number of iterations.

6. Hebbian-learning Vs GA-learning

In order to better evaluate the learning capabilities of the Hebbian-like algorithms, which determine the classification performance of the FCM classifiers, it is useful to examine the classification scores of the classifiers, trained by a Genetic algorithm (GA-learning).

For this purpose, two different implementations of the GA-learning are studied in this section: the first one (GA1) uses the *CoMD* and the second (GA2) the *CpO* class mapping methods discussed in Section 4.2.

The GA configuration parameters are presented in Table 18, while the objective function being optimized (maximized) is the classification rate (CRate) defined in Section 4.

In the following Table 19, the classification statistics of both GA1 and GA2 algorithms are compared with those of the AHL algorithm, since based on the previous analysis; it is the most effective one among all the Hebbian-like algorithms.

By studying the above results, two conclusions of major importance can be drawn. Firstly, it is deduced that the training of the FCM classifier, by using the genetic algorithm and the *CpO* mapping method, shows superior performance against the Hebbian-like algorithms (NHL – Table 3). This outcome is justified by the supervised nature of the GA-learning, according to which the objective function evaluation is used to find more optimal weights. This kind of system feedback does not exist in the Hebbian-like algorithms which work in an unsupervised fashion.

Secondly, the results highlight the superiority of the GA1 algorithm not only over the GA2 but also over the AHL algorithms. This

Table 18
Genetic algorithm's settings.

Parameter	Value
Population size	10
Maximum generations	100
Crossover probability	0.8
Mutation probability	0.01
Selection method	Stochastic universal approximation (SUS)
Crossover points	2 Points

Table 19
Classification performance of the GA-learning algorithms and the AHL one.

Dataset	Learning algorithm	Min. CRate (%)	Mean CRate (%)	Max. CRate (%)	Stdev.
Pima	AHL	52.60	54.43	56.77	1.28
	GA1	67.19	72.40	77.60	3.65
	GA2	59.38	63.75	67.19	2.89
Hepatitis	AHL	25.64	41.03	58.97	10.32
	GA1	71.79	79.23	84.62	3.21
	GA2	71.79	77.69	84.62	2.67
Parkinson	AHL	75.51	80.82	85.71	3.32
	GA1	65.31	74.49	81.63	5.54
	GA2	73.47	81.22	87.76	6.23
Wine	AHL	53.33	64.89	75.56	6.80
	GA1	88.89	92.00	95.56	3.82
	GA2	44.44	64.00	84.44	10.12
Iris	AHL	80.56	87.78	97.22	5.58
	GA1	97.22	99.17	100.00	0.87
	GA2	58.33	69.44	83.33	7.54
Thyroid	AHL	49.06	66.79	77.36	7.41
	GA1	94.34	94.53	96.23	0.93
	GA2	81.13	85.28	90.57	2.65
Vehicles	AHL	26.09	30.22	36.96	3.14
	GA1	39.13	59.78	69.57	8.98
	GA2	30.43	39.13	54.35	6.64
Glass	AHL	24.53	34.53	45.28	5.91
	GA1	50.94	57.74	67.92	3.87
	GA2	35.85	46.42	54.72	4.13

fact implies that the genetic algorithms as a global optimizer with less possibility to converge to local optima, finds a good solution on the weight search space, while the Hebbian-like algorithms are possibly trapped in local optima.

However, although the GA settings (Table 18) are not optimal (they found by trial and error) and the GA-learning approach is time consuming it significantly dominates over the Hebbian-like algorithms (Froelich, 2009), makes them a good choice for FCM classifier training.

7. Conclusion

A short review of the Hebbian-like algorithms by highlighting their main differences and similarities has been taken in the first part of this work, in order to analyze the algorithms' theoretical aspects that make them appropriate for the training of the FCMs. While these algorithms have been proposed to find the appropriate interconnection weights of the FCMs for system modeling tasks, their performance in pattern classification problems was investigated in the main part of this study.

Moreover, an important sensitivity analysis of the main configuration parameters that influence the pattern recognition scheme incorporating the FCMs has been presented, by concluding to an appropriate framework that ensures high classification accuracy of the FCM classifier.

As a result of the algorithms' comparison, the AHL algorithm has shown the best performance, when it works in collaboration with the *CoMD* mapping methodology introduced herein.

Although it is worth giving attention to the training results of the AHL algorithm, its performance is inferior to that of the GA-learning under the same conditions. This deficiency of the Hebbian-like algorithms indicates the case of trapping to local optima, in contrast to the global searching mechanism of the genetic algorithms.

By processing the results of the present analysis, several future research directions can be scheduled, mainly regarding the weight initialization procedure, the hidden concepts addition, and the

development of a pattern classification oriented learning algorithm with improved generalization abilities.

References

Aguilar, J. (2005). A survey about fuzzy cognitive maps papers. *International Journal of Computational Cognition, 3*(2), 22 (Invited paper).

Azadeh, A., Ziaei, B., & Moghaddam, M. (2012). A hybrid fuzzy regression-fuzzy cognitive map algorithm for forecasting and optimization of housing market fluctuations. *Expert Systems with Applications, 39*(1), 298–315.

Büyüközkan, G., Vardaloğlu, Z. 2012. Analyzing of CPFR success factors using fuzzy cognitive maps in retail industry. Expert Systems with Applications. http://dx.doi.org/10.1016/j.eswa.2012.02.014.

Dickerson, J. A., & Kosko, B. (1994). Virtual worlds as fuzzy cognitive maps. *Presence, 3*(2), 173–189.

Froelich, W., & Juszczuk, P. (2009). Predictive capabilities of adaptive and evolutionary fuzzy cognitive maps – A comparative study. In N. T. Nguyen, E. Szczerbicki (Eds.), *Intelligent systems for knowledge management, Studies in computational intelligence* (Vol. 252, pp. 153–174).

Ghaderi, S. F., Azadeh, A., Nokhandan, B. P., & Fathi, E. (2012). Behavioral simulation and optimization of generation companies in electricity markets by fuzzy cognitive map. *Expert Systems with Applications, 39*(5), 4635–4646.

Glykas, M. (2010). *Fuzzy cognitive maps: Advances in theory, methodologies, tools and applications*. *Studies in fuzziness and soft computing* (Vol. 247). Springer-Verlag.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Prentice-Hall.

Huerga, A. V. (2002). A balanced differential learning algorithm in fuzzy cognitive maps. In *International workshop on qualitative reasoning, 2002*.

Kosko, B. (1986). Fuzzy Cognitive Maps. *International Journal Man–Machine Studies, 24*, 65–75.

Koulouriotis, D. E., Diakoulakis, I. E., Emiris, D. M., & Zopounidis, C. D. (2005). Development of dynamic cognitive networks as complex systems approximators: Validation in financial time series. *Applied Soft Computing, 5*(2), 157–179.

Lee, K. C., Lee, S. 2012. A causal knowledge-based expert system for planning an Internet-based stock trading system. *Expert Systems with Applications*. http://dx.doi.org/10.1016/j.eswa.2012.01.191.

Li, S. J., & Shen, R. M. (2004). Fuzzy cognitive map learning based on improved nonlinear Hebbian rule. In *3rd International conference on machine learning and cybernetics, 26–29 August, 2004* (pp. 2301–2306). Shanghai.

Oja, E. (1989). Neural networks, principal components and subspaces. *International Journal of Neural Systems, 1*, 61–68.

Papageorgiou, E. I., & Groumpos, P. P. (2004). A weight adaptation method for fuzzy cognitive maps to a process control problem. In: M. Bubak, G. van Albada, P. Sloot, J. Dongarra (Eds.), *ICCS 2004, LNCS* (Vol. 3037, pp. 515–522).

Papageorgiou, E. I., Stylios, C. D., & Groumpos, P. P. (2003). Fuzzy cognitive map learning based on nonlinear Hebbian rule. In: T. D. Gedeon, & L. Fung (Eds.), *AI 2003: Advances in artificial intelligence, LNCS* (Vol. 2903, pp. 256–268).

Papageorgiou, E. I., & Groumpos, P. P. (2005). A new hybrid method using evolutionary algorithms to train Fuzzy Cognitive Maps. *Applied Soft Computing, 5*(4), 409–431.

Papageorgiou, E. I., Parsopoulos, K. E., Stylios, C. S., Groumpos, P. P., & Vrahatis, M. N. (2005). Fuzzy cognitive maps learning using particle swarm optimization. *Journal of Intelligent Information Systems, 25*(1), 95–121.

Papageorgiou, E. I., Stylios, C. D., & Groumpos, P. P. (2004). Active Hebbian learning algorithm to train fuzzy cognitive maps. *International Journal of Approximate Reasoning, 37*(3), 219–249.

Papageorgiou, E. I., Stylios, C. D., & Groumpos, P. P. (2006). Unsupervised learning techniques for fine-tuning fuzzy cognitive map casual links. *International Journal of Human–Computer Studies, 64*(8), 727–743.

Papakostas, G. A., Boutalis, Y. S., Koulouriotis, D. E., & Mertzios, B. G. (2008). Fuzzy cognitive maps for pattern recognition applications. *International Journal of Pattern Recognition and Artificial Intelligence, 22*(8), 1461–1468.

Papakostas, G. A., Boutalis, Y. S., Samartzidis, S. T., Karras, D. A., & Mertzios, B. G. (2008). Two-stage hybrid tuning algorithm for training neural networks in image vision applications. *International Journal of Signal and Imaging Systems Engineering, 1*(1), 58–67.

Papakostas, G. A., Polydoros, A. S., Koulouriotis, D. E., & Tourassis, V. D. (2011). Training fuzzy cognitive maps by using Hebbian learning algorithms: A comparative study. In *IEEE international conference on fuzzy systems (FUZZ–IEEE 2011)*, 27–30 June, 2011 (pp. 851–858). Taipei, Taiwan.

Papakostas, G. A., & Koulouriotis, D. E. (2010). Classifying patterns using fuzzy cognitive maps. In M. Glykas (Ed.), *Fuzzy cognitive maps: Advances in theory, methodologies, tools and applications* (pp. 291–306). Springer. ISBN: 978-3-642-03219-6.

Peng, Z., Yang, B., & Fang, W. (2008). A learning algorithm of Fuzzy Cognitive Map in document classification. In *International conference on fuzzy systems and knowledge discovery* (pp. 501–504).

Stach, W., Kurgan, L., & Pedrycz, W. (2008). Data-driven nonlinear Hebbian learning method for fuzzy cognitive maps. In *IEEE international conference on fuzzy systems (FUZZ–IEEE)* (pp. 1975–1981).

Stach, W., Kurgan, L., Pedrycz, W., & Reformat, M. (2005). Genetic learning of fuzzy cognitive maps. *Fuzzy Sets and Systems, 153*(3), 371–401.

UCI Machine Learning Repository. Irvine. (xxxx). <http://archive.ics.uci.edu/ml/datasets.html>.