Learning Algorithms For Fuzzy Cognitive Maps

Elpiniki Papageorgiou
Laboratory for Automation& Robotics, Dep. of Electrical and Computer Engineering, University of Patras, 26500 Rion, Patras, GREECE
epapageo@ee.upatras.gr

Chrysostomos Stylios
Laboratory for Automation & Robotics,
Dep. of Electrical and Computer Engineering, University of Patras, 26500 Rion, Patras, GREECE
stylios@ee.upatras.gr

Peter Groumpos
Laboratory for Automation & Robotics,
Dep. of Electrical and Computer Engineering, University of Patras, 26500 Rion, Patras, GREECE
groumpos@ee.upatras.gr

Abstract

Fuzzy Cognitive Maps have been introduced as a combination of Fuzzy logic and Neural Networks. In this paper a new learning rule based on unsupervised Hebbian learning and a new training algorithm based on Hopfield nets are introduced and are compared for the training of Fuzzy Cognitive Maps.

Keywords: Fuzzy Cognitive Maps, Learning algorithms, Hopfield Networks, Unsupervised Hebbian learning law.

1 Introduction

Fuzzy Cognitive Map (FCM) is a soft computing modeling methodology for complex systems, which originated from the combination of fuzzy logic and neural networks. FCM is a symbolic representation for the description and modeling of the behavior and operation of a system. FCMs consist of concepts, that illustrate different aspects in the behavior of the system and these concepts interact each other and are interconnected with weighted arcs showing the dynamics of the system. Human experts, who have knowledge and experience in the operation of the system construct and develop the FCM, experts define the number and kind of concepts and the weighted interconnections between them [9]. But, the human knowledge and experience aren't always reliable. Therefore, learning algorithms are proposed to training the FCM and select the appropriate weights [6,8].

Fuzzy Cognitive Maps are a combination of techniques from fuzzy logic theory and neural networks. So, it is expected to have learning capabilities. The learning rule for a FCM is a procedure for modifying its weight matrix in order to train the FCM to model the behavior of a system. The Differential Hebbian learning algorithm has been considered as the more appropriate for Fuzzy Cognitive Maps [6] and here it is proposed the use of Hopfield learning algorithm.

2 Differential Hebbian Learning

The unsupervised Hebbian learning rule have been proposed to be implemented in training FCMs. Hebb' s postulate of learning states the following:

1. If two neurons on either side of a synapse (connection) are activated simultaneously (i.e., synchronously), then the strength of that synapse is selectively increased.

2. If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

An initial FCM is constructed with arbitrary weight values. It is then trained to make predictions of future average value of concepts, the FCM runs through the historical data set one state at a time. For

comparing the FCM's output with the expected output provided in the data. Weights are adjusted when error is identified. The data set is cycled until the error has been reduced sufficiently for no more changes in weights to occur. If a correlated change between two concepts is observed, then a causal relation between the two is likely and the strength of this relationship should depend on the rate of the correlated change. This proposition forms the basis of the Differential Hebbian Learning (DHL). Kosko discusses the use of DHL as a form of unsupervised learning for FCMs [7].

DHL can be used to make an FCM adapt causal link strengths as part of an unsupervised training with a sequence of state vectors. However, there is no guarantee that DHL will encode the sequence into the FCM. It has been proposed a potential of DHL by creating a bivalent simple FCM, which is fed with stimulus vectors to obtain some limit cycles. Using these limit cycles as training data, DHL is then used to create a new FCM. It has been showed that the new FCM tends to learn the same attractors (limit cycles) as those in the initial FCM.

The general unsupervised Hebbian rule is :

$$e_{ij}(t+1) = e_{ij}(t) + \mu_t \left[\Delta C_i(t) \cdot \Delta C_j(t) - e_{ij}(t)\right] \quad (1)$$

where ÄC$_i$ is the change in concept i and ÄC$_i$(t)=C$_i$(t)-C$_i$(t-1). The learning coefficient μ_t decreases slowly over time. Kosko [6] suggested the coefficient :

$$\mu_t = 0.1\left[1 - \frac{t}{1.1N}\right] \quad (2)$$

The constant N ensures the learning coefficient C_t never becomes negative.

So, at each time step t, the value for e_{ij}, the edge linking concept i and j, is given by the discrete version of the DHL law from equation (1).

2.1 Proposed Hebbian law for FCMs.

Here it is suggested a new training algorithm, based on the above unsupervised Hebbian learning rule:

$$w_{ij}(t+1) = w_{ij}(t) + a \cdot \left(C_i(t) \cdot C_j(t) - w_{ij}(t)\right) \quad (3)$$

where $C_i(t)$ is the value of concept i at time step t, and a is the learning coefficient. After simulation experiments the most appropriate value of a is equal to 0,2, which gives better and faster results.

The general update rule for each concept ina a FCM is:

$$A_i(t+1) = f(A_i(t) + \sum_{j \neq i} w_{ji}(t+1) \cdot A_j(t)) \quad (4)$$

In section 5 the suggested learning rule is implemented for a simple example.

3 Hopfield Neural Networks

The Hopfield network is a single-layer feedback Neural Network as shown in Figure 1. When operated in discrete-time fashion, it is called a discrete Hopfield network and its structure is termed as recurrent [4].

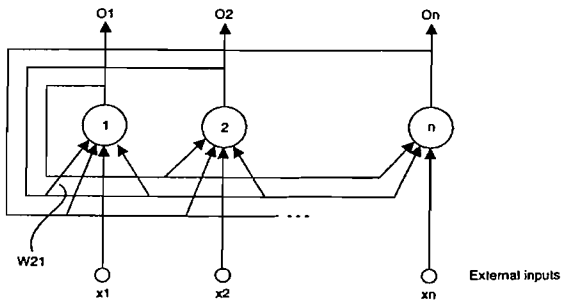

Figure 1: Hopfield auto associative neural network

The neurons in Hopfield networks are characterized with: binary or bivalent input signals, binary or bivalent output signals, simple summation function, and hard-limited threshold activation function. Every neuron j, j=1,2,...n, in the network is connected back to every other one, except itself. Input patterns x_j are supplied as external inputs and cause activation of the external outputs. The response of such a network, when an input vector is supplied during the recall procedure, is dynamic, that is, after supplying the new input pattern, the network calculates the outputs and then feeds back to the neurons; new output values are then calculated and so on, until an equilibrium point is reached.

In this research work a new learning rule for weights of a FCM is proposed, based on the theory of Hopfield networks. A part of the discrete Hopfield algorithm is adopted and it is adjusted in the framework of FCMs, considering updated values of

concepts due to feedback. The transition process continues until no new updated responses are produced, the values of concepts are the Desired Values of Concepts (DVC) and the FCM has reached an equilibrium point.

4. FCM training based on Hopfield NN

Hopfield considers that each node has an external input xj and a threshold Èj, where j=1,..n (figure 1). It is important to point out that there is no self-feedback in a Hopfield network. The same is also considered in FCMs. The j-th node output is connected to each 7other nodes' inputs through a multiplicative weight w_{ij} for i=1,...,n, i≠j; that is w_{ii}=0 for i=1,..n. Furthermore, it is required that the network weights be symmetric, that is, w_{ji}=w_{ij}, i,j=1,2,...n. the evolving rule (or the update rule) for each node in a discrete Hopfield network is:

$$y_i^{(k+1)} = \text{sgn}\left(\sum_{j=1}^{n} w_{ij} \cdot y_i^{(k)} + x_i - \theta_i \right) \quad (5)$$

where sgn( ) is the signum function defined by the equation: sgn (f) = {1, if f≥0 and –1, if f<0} and the superscript k denotes the index of recursive update.

Asynchronous fashion means that for a given time only a single node is allowed to update its output. The next update on a randomly chosen node in a series uses the already updated output. In other words, under asynchronous operation of the network, each output node is updated separately, while taking into account the most recent values that have already been updated.

The training procedure for the Hopfield network is reduced to the simple calculation of the weights wij on the basis of the training examples with the use of the formula:

$$w_{ji} = \sum_{p=1,m} \left(2x_j^{(p)} - 1\right) \cdot \left(2x_i^{(p)} - 1\right) \quad (6)$$

where the summation is held for all the training patterns $x^{(p)}$, and the expressions in parentheses can be only 1 or 0 according to the value of the input pattern.

In FCM case, it is considered that there are not input values in each node and the thresholds $\dot{E}_i$ have constant values for each concept, suggested by us. Each value of concept will be updated, due to the training procedure, and it is named as Desired value of Concept (DVC).

The following update rule for each DVC in a FCM is proposed:

$$DVC_i(t+1) = f\{DVC_i(t) + \sum_j W_{ji}(t+1) \cdot DVC_j(t)\} \quad (7)$$

where f is the activation function given f(x)=1/(1+exp(-x)), and t denotes the recursive update (iteration).

The training algorithm for the weights is:

$$w_{ij}(t+1) = w_{ij}(t) + (2DVC_i(t) - 1) \cdot (2DVC_j(t) - 1) \quad (8)$$

It is assumed that w_{ij}=w_{ji} for i≠j due to symmetry suggested by Hopfield and w_{ii}=0 because there is no feedback from the DVC to itself.

Due to asynchronous updating, only a concept is allowed to update its value every step. The next update on a randomly chosen node in a series uses the already updated concept, DVC.

It is defined that the FCM reaches an equilibrium region after t steps when no new updated responses are produced, the DVCs are constant and the same for the weights.

4.1 The energy function

Hopfield also defined a parameter E, called the energy of the network, which is a dynamical parameter and which can be calculated at any moment t (step) as follows:

$$E_{tot} = -\frac{1}{2} \cdot \sum_{j}^{N} \sum_{i \neq j}^{N} w_{ji} \cdot OUT_i \cdot OUT_j \quad (9)$$

The energy for DVCs of FCM is defined:

$$E_{tot} = -\frac{1}{2} \cdot \sum_{j}^{N} \sum_{i \neq j}^{N} w_{ji} \cdot DVC_i \cdot DVC_j \quad (10)$$

We introduce the concept of energy in the FCMs and we calculate the total energy of the interesting concept at the final state and if this energy is minimum at the same time with the stable state of the outputs, then we conclude that the FCM has reached an equilibrium point.

The energy E, suggested by Hopfield can be illustrated as a surface in n-dimensional space. The

equilibrium point in a trained Hopfield network can be explained by *the attractor principle.*

During training the network "sets" some *basins* of attraction, which are stable states for the network corresponding to the training patterns. When a new vector x' is supplied for a recall, the network will eventually rest after some cycles in an attractor, thus associating the new vector with one of the known attractors (figure 3).

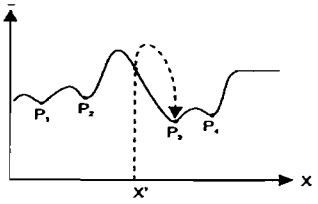

Figure 2: When a new input pattern X' is presented the network relaxes in an equilibrium state, "a basin of attraction" where the energy E is minimum, thus associating the new pattern with class pattern P_3. [1]

In an extreme case, the number of basins of attraction is equal to the number of training patterns. During the recall procedure, when a new input pattern is applied, the network tries to relax into the nearest basin of attraction, thus finding the right pattern to associate the new one with. The recall process is the process of relaxation.

The main property of the energy function is that it always decreases or stays constant as the network cycles. In the following it is easy to show mathematically why this function always decreases at each time step.

Suppose that at time t, the state of the jth concept is changed. If the change in the state of the ith concept is:

$\Delta C_i(t+1) = C_i(t+1) - C_i(t)$, then we have $\ddot{A}C_i(t+1)=0$ for $i\neq j$, and else $\ddot{A}C_j(t+1)\neq 0$.

The change in the energy caused by the change in the state of the jth concept is:

$$\Delta E = -\frac{1}{2}\sum_i\sum_j w_{ij}\left(C_i(t+1)\cdot C_j(t+1) - C_i(t)\cdot C_j(t)\right)$$

Adding and subtracting $C_i(t+1)C_j(t+1)$, and simplifying, we get:

$$\Delta E = -\frac{1}{2}\sum_i\sum_j w_{ij}\left(C_i(t+1)\cdot \Delta C_j(t+1) + \Delta C_i(t)\cdot C_j(t)\right)$$

which can be reduced to:

$$\Delta E = -\Delta C_j \cdot \left(\sum_{i\neq j} w_{ij}\cdot C_i\right)$$

using the facts that $\ddot{A}C_i(t+1)=0$ for $i\neq j$ and $w_{ij}=w_{ji}$.

If the term in brackets is positive, the value of $\ddot{A}C_j(t+1)$ is also positive, so ÄE is negative. If the value in brackets is negative, the same is $\ddot{A}C_j(t+1)$, and again ÄE is negative. Hence each change of state brought about by the asynchronous updating method reduces the value of the energy function.

The change in E, as based on a change in a single desired value of concept $\ddot{A}DVC_i$, can be expressed as

$$\Delta E_j = -\tfrac{1}{2}\cdot \Delta DVC_j \sum_j^N \sum_{i\neq j}^N w_{ji}\cdot DVC_i \qquad (11)$$

This is a steadily decreasing function as the network evolves. During the recall procedure, the system continues to calculate desired values of concepts DVCs until it reaches a minimum of the energy function E at the same time, which is an equilibrium state.

4.2 Proposed training FCM algorithm

Here it is proposed a training algorithm for FCM.

There are presented two ways of calculating new values of weights, one based on Hopfield nets that described in section 3, and the other based on Hebbian learning that described in section 2.1.

There are two methods for updating the states of the concepts. In the asynchronous updating method, the states of the concepts are changed one at a time, in random order. In the simultaneous method, the states of the concepts are all changed at the same time, on the basis of their states after the previous update.

We suggest here the asynchronous updating for the concepts of FCM.

1. The FCM is developed [9] and the weights w_{ij} take their initial values.
2. Randomly initial values Aj(0) are assigned to the concepts j, j=1,2,...n
3. Due to asynchronous updating only one of the concepts C_j is selected as the first updated concept (DVCj), and the other concepts DVC_i of FCM are updated randomly at next steps t.

4. The new values of the updated concept, $DVC_j(1)$ for j=1,2,...n, is calculated using the equations (8) and (9):

$$DVC_i(t+1) = f\{DVC_i(t) + \sum_j w_{ij}(t+1) \cdot DVC_j(t)\} \quad (12)$$

where

$$w_{ij}(t+1) = w_{ij}(t) + (2DVC_i(t) - 1) \cdot (2DVC_j(t) - 1) \quad (13)$$

in the case of Hopfield training algorithm, and

$$w_{ij}(t+1) = w_{ij}(t) + 0.2(DVC_i(t)DVC_j(t) - w_{ij}(t)) \quad (14)$$

in the case of unsupervised Hebbian training algorithm, f is the activation function and $w_{ij}(t+1)$ is the value of weight at step t+1.

5. The updated value DVC_i of concept from the selected node C_i becomes the new updated input through the feedback connections to the other nodes C_j.

6. The transition process continues until no new updated responses are produced and the system-FCM has reached its equilibrium.

7. Ôhe total energy E(t) of the system at every step is also calculated. When the energy function E reaches a minimum value comparing with the values of energy at next and previous steps, the transition process stops.

When all the outputs-DVCs, no longer change their values for at least three consecutive cycles and at the same time the total energy E is minimum then the FCM reaches an equilibrium region.

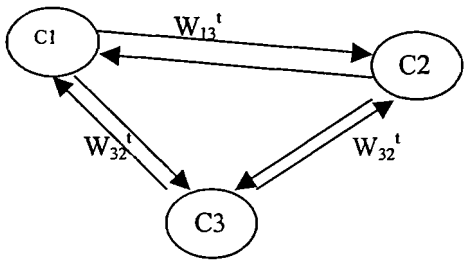

Figure 3. A simple FCM of 3 concepts where concept C_3 is a feedback node and W_{31}^t , W_{32}^t are the trained weighted arcs.

5. Example and simulation results.

We consider an FCM consisted of 3 concepts without inputs to nodes, assuming all the values of thresholds $\grave{E}_i$ equal to zeros, (Figure 3).

The initial values of concepts and weights are the following:

C^0=[0.32, 0.4, 0.45] and

$$W^0 = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

For the case of the Hopfield algorithm the equation (13) is used to calculate the updated weights and equation (12) to calculate the updated Desired Values of Concepts (DVCs), for t time steps.

The weights take their final updated values after two simulation steps:

$$W^1 = \begin{bmatrix} 0 & 0.329 & -0.717 \\ 0.329 & 0 & 0.0058 \\ -0.717 & 0.0058 & 0 \end{bmatrix}$$

$$W^2 = \begin{bmatrix} 0 & 0.38 & -0.66 \\ 0.38 & 0 & 0.02 \\ -0.66 & 0.02 & 0 \end{bmatrix}$$

At the next steps the values of weights do not longer change. The DVCs take their constant values after 8 simulation steps, as shown in Table 1.

Table 1. The values of DVCs for 9 simulation steps.

Steps	DVC1	DVC2	DVC3
1	0.32	0.4	0.5646
2	0.5	0.5	0.5646
3	0.5	0.5	0.5527
4	0.5	0.5	0.5527
5	0.5	0.5	0.5498
6	0.5	0.5	0.5498
7	0.5	0.5	0,5491
8	0.5	0.5	0.5491
9	0.5	0.5	0.5491

At each simulation step the total energy is calculated according to the equation (10) for the Desired Value of Concept.

The energy function decreases and at the 7^{th} step stays constant for the consecutive cycles. At this step we take the minimum value of energy (equals to

0,0895) and the change in energy ÄE is always negative, as shown in the previous section.

For the case of Differential Unsupervised Hebbian learning algorithm, the equation (14) is used to calculate the updated weights and equation (12) to calculate the updated DVCs.

After 30 simulation steps the values of weights remain unchanged and do the updated DVCs.

The final updated values of weights after the proposed training procedure are:

$$W^{29}=W^{30}=\begin{bmatrix} 0 & 0,2502 & 0,3778 \\ 0,2495 & 0 & 0,3786 \\ 0,3789 & 0,3786 & 0 \end{bmatrix}$$

The DVCs stay constant after 28 time steps. Table 2 gives the DVCs for 30 consecutive cycles:

Steps	DVC1	DVC2	DVC3
1	0.32	0.4	0.6347
2	0.5	0.5	0.6347
3	0.5	0.5	0.780
4	0.5	0.5	0.780
5	0.5	0.5	0.7899
6	0.5	0.5	0.7899
7	0.5	0.5	0,7826
8	0.5	0.5	0.7826
9	0.5	0.5	0.7751
10	0.5	0.5	0.7751
11	0.5	0.5	0.7695
12	0.5	0.5	0.7695
13	0.5	0.5	0.7656
14	0.5	0.5	0.7656
15	0.5	0.5	0.761
16	0.5	0.5	0.761
17	0.5	0.5	0.7584
18	0.5	0.5	0.7584
19	0.5	0.5	0.7577
20	0.5	0.5	0.7577
21	0.5	0.5	0.7572
22	0.5	0.5	0.7571
23	0.5	0.5	0.7571
24	0.5	0.5	0.7570
25	0.5	0.5	0.7570
26	0.5	0.5	0.7570
27	0.5	0.5	0.7569
28	0.5	0.5	0.7569
29	0.5	0.5	0.7569
30	0.5	0.5	0.7569

As we observe from the above, the DVC for the 3rd concept is reached the desired value 0.7569 and the other two DVCs are reached the value 0.5, as it was expected

So, from the results of Hebbian learning, for three or more consecutive steps the updated weights and the Desired Values of Concepts (DVCs) no more change their values and calculating the energy we observe that it always decreases taking at 29th time step a minimum value. At this step the FCM reaches an equilibrium region.

6. Conclusions

Two methods for training FCM based on Hopfield Networks and Hebbian learning law was presented. The Hebbian learning algorithm gives better results comparing with the Hopfield algorithm.

Acknowledgements

This research was partially supported by the Greek GSRT and European Social Fund under PENED'99 project 99ED514

References

[1] M. Hagan, H. Demuth, M. Beale (1996). *Neural Network Design*. PWS Publishing Company.

[2] S. Haykin (1994). *Neural Networks: A comprehensive Foundation*, Prentice Hall, Inc.

[3] J. Hopfield (1982). Neural Networks and Physical Systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci. USA* Volume 79 pages 2554-2558.

[4] J. Hopfield and D. Tank (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics* Volume 52 pages 142-152.

[5] N. Kasabov (1996). *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. MIT Press Cambridge, London.

[6] B. Kosko (1992). *Neural Networks and Fuzzy Systems*. Prent.Hall, Inc. Englewood Cliffs, N. Jersey, USA

[7] B. Kosko (1997). *Fuzzy Engineering*. Prentice Hall, Inc. Upper Saddle River, N. Jersey, USA

[8] C.D. Stylios and P.P. Groumpos (1999). Mathematical formulation of Fuzzy Cognitive Maps. In: *Proc. of the 7th IEEE Med. Conference on Control and Automation*, CD, Haifa, Israel, June 1999.

[9] C. D. Stylios and P.P. Groumpos (2000). Fuzzy Cognitive Maps: A soft computing technique for Intelligent Control. In *Proc. of the 15th IEEE Int.Symp.on Intelligent Control, pages 67-72 Patras*, Greece, July 2000.

[10] P. Vuorimaa (1994). Fuzzy self-organizing Map", FSS 66, 223-231.