

METHODS AND EXPERIMENTS REGARDING LEARNING FUZZY COGNITIVE MAPS (FCM) WEIGHTS USED IN FUPOL ¹

Maria Moise²
Victor Popa³

ABSTRACT

In order to develop a suitable method for computing Fuzzy Cognitive Maps (FCM) weights in FUPOL project, we compare two methods for computing FCM weights using as historical data the data obtained by simulating a simple FCM model. First method Least Square Error (LSE) is based on optimization formalism, the second method Adaptive Neuro Fuzzy System (ANFIS) combines Fuzzy Inference Systems with the optimization approach. Based on this comparison, we can decide if ANFIS method can be used alone or it is necessary to combine it with other methods.

Keywords: FCM, LSE, Identical function, Sigmoid function, Tanh function, ANFIS.

1. DESCRIPTION OF THE METHODS FOR COMPUTING FCM WEIGHTS

For computing FCM weights we propose the following methods:

- Least Squared Error (LSE) based FCM weights computation;
- Adaptive Neuro Fuzzy Inference System (ANFIS) based FCM weights computation.

I. LSE based FCM weights computation

The FCM weights w_{ij} are determined by minimization problem:

$$\min \sum_{t=1}^T \sum_{j=1}^N [(x_j^t - x_j^{\sim t})]^2$$
$$-1 \leq w_{ij} \leq 1 ; i, j = 1, \dots, N$$

where,

- $x_j^{\sim t}$ is training data for concept C_j at time t
- x_j^t is computed value for concept C_j at time t :

$$x_j^t = f\left(\sum_{i=1}^N w_{ij} \cdot x_i^{t-1}\right)$$

- N is number of concepts
- T is number of training data

¹FUPOL project (www.fupol.eu) is a research project, financed by FP7.

²Romanian American University, Bucharest, e-mail:maria.moise@rau.ro

³National Institute for Research & Development in Informatics, Bucharest, e-mail:vpopa@ici.ro

- f = Identical, Sigmoid, Tanh
- ANFIS based FCM weights computation

The ANFIS approach learns the rules and membership functions from data. ANFIS is an adaptive network. An adaptive network is a network of nodes and directional links. It's called adaptive because some, or all, of the nodes have parameters which affect the output of the node. These networks are learning a relationship between inputs and outputs. The basic learning rule for ANFIS is based on gradient descent and chain rule. The combination of a fuzzy model with a training of the outputs has the advantage that the model can be understood by the expert and adapt to the measured values. This leads to a better fit of the model. The ANFIS architecture with two input variables and one output variable is shown in Figure 1. The circular nodes represent nodes that are fixed whereas the square nodes are nodes that have parameters to be learnt.

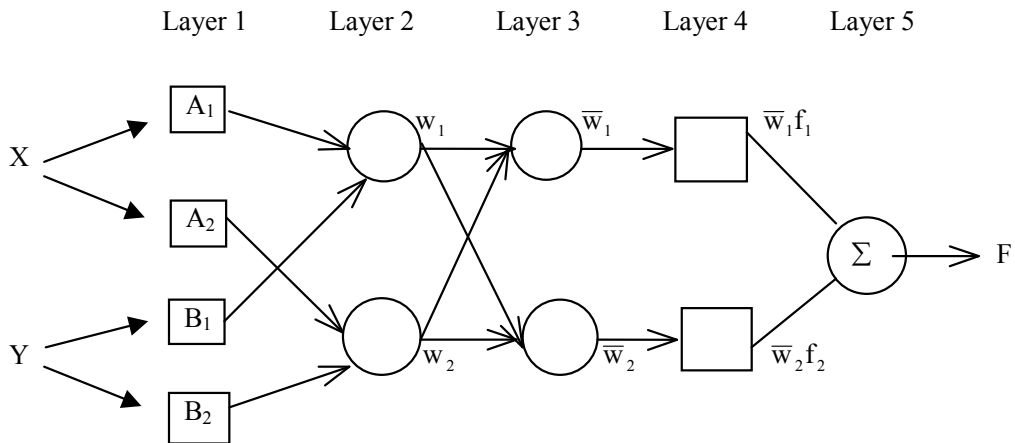


Figure 1: ANFIS Network

ANFIS method is applied to computes FCM weights in 3 steps:

- Generation of fuzzy rules
 - Computing simplified rules
 - Computation of FCM weights
- i. Generation of fuzzy rules**

If FCM model includes N concepts ($C_1, C_2, \dots, C_N$) and for each concept C_i consider M Gaussian membership functions $G_{ij}(m_{ij}, \sigma_{ij})$ $j=1, M$, then ANFIS method generates MN Sugeno rules:

Rule: if $x_1^t = G^1_{j_1}$ And $x_2^t = G^2_{j_2} \dots \dots \dots$ AND $x_N^t = G^N_{j_N}$



$$\begin{cases} x^{t+1}_1 = a^1_1 \cdot x^t_1 + a^1_2 \cdot x^t_2 + \dots + a^1_N \cdot x^t_N \\ x^{t+1}_2 = b^1_1 \cdot x^t_1 + b^1_2 \cdot x^t_2 + \dots + b^1_N \cdot x^t_N \\ \vdots \\ x^{t+1}_N = z^1_1 \cdot x^t_1 + z^1_2 \cdot x^t_2 + \dots + z^1_N \cdot x^t_N \end{cases}$$

OR

$$\text{if } x^t_1 = G^1_{j1} \text{ AND } x^t_2 = G^2_{j2} \dots \text{ AND } x^t_N = G^N_{jN}$$



$$X^{t+1} = A * X^t$$

where,

- matrix A has elements a_{ij} and $-1 \leq |a_{ij}| \leq 1$

ii. Computing simplified rules

For each generated rule, we construct a simplified rule of form:

$$\mathbf{Rk}_{\text{simplified}}: \text{if } x^t_1 = G^1_{j1} \text{ AND } x^t_2 = G^2_{j2} \dots \text{ AND } x^t_N = G^N_{jN}$$



$$W = A$$

where,

- W = transition matrix from state $\langle x^t_1, x^t_2, \dots, x^t_N \rangle$ to state $\langle x^{t+1}_1, x^{t+2}_2, \dots, x^{t+1}_N \rangle$

We call these simplified rules “FIS”. The FIS system is used in computing FCM rules.

iii. Computation of FCM weights

For each time $t=1, T$ we construct W^t by aggregating simplified rules:

$$W^t = \text{Fis}(\langle x^t_1, x^t_2, \dots, x^t_N \rangle)$$

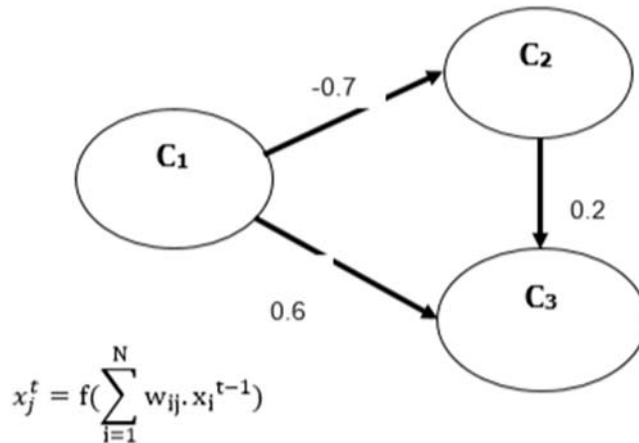
Finally, FCM weight matrix W is computed as average:

$$W = \frac{\sum_t W^t}{T}$$

2. COMPUTING FCM WEIGHTS FROM HISTORICAL DATA

I. Preparing training data

The training data is generated by simulating a simple FCM model presented below.



As shown in the figure above there are some relations between causal concepts and effect concepts, and they can be presented in a square matrix with zero on the diagonal (Table 1).

	C ₁	C ₂	C ₃
C ₁	0	-0.7	0.6
C ₂		0	0.2
C ₃			0

Table 1: Relations between causal concepts and effect concept

The simulation will be done considering 3 cases: **f = Identical function**, **f = Sigmoid function**, **f= Tanh function**.

Case 1 (f = Identical function)

Running the FCM model in this case we obtain the data listed in Table 2 and shown in graphical representation (Figure 2).

t	C ₁	C ₂	C ₃
0	1	2	3
1	1	-0.7	1
2	1	-0.7	0.46
3	1	-0.7	0.46

Table 2: Training Data based on Identical function

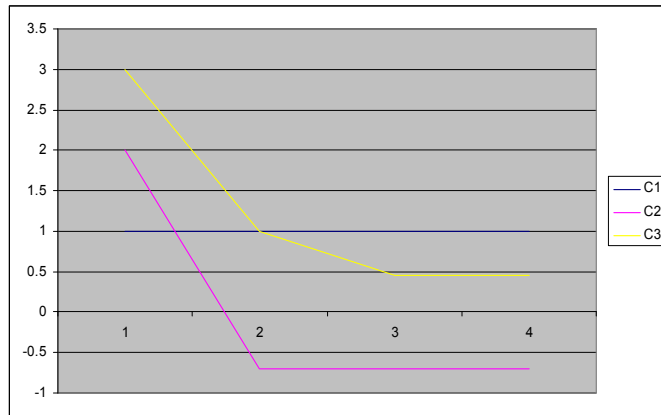


Figure 2: Concept Graphics (Identical function)

Case 2 ($f = \text{Sigmoid}$)

Running the FCM model in this case we obtain the data listed in Table 3 and shown in graphical representation (Figure 3).

t	C ₁	C ₂	C ₃
0	1	2	3
1	1	0.331812227831834	0.731058578630005
2	1	0.331812227831834	0.660688172023353
3	1	0.331812227831834	0.660688172023353

Table 3: Training Data based on Sigmoid function

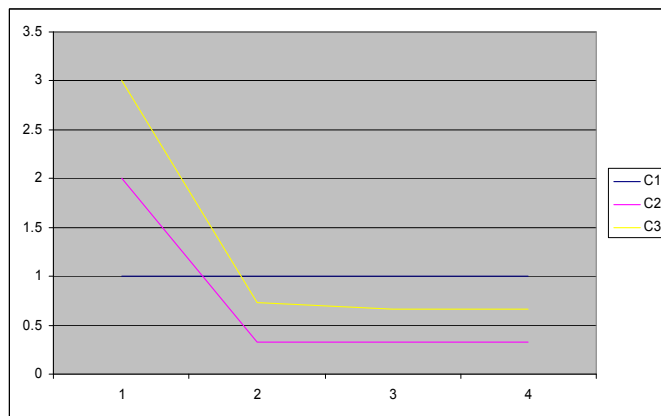


Figure 3: Concept Graphics (Sigmoid function)

Case 3 ($f = \text{Tanh}$)

Running the FCM model in this case we obtain the data listed listed in Table 4 and shown in graphical representation (Figure 4).

t	C ₁	C ₂	C ₃
0	1	2	3
1	1	-0.604367777117164	0.761594155955765
2	1	-0.604367777117164	0.445543736205845
3	1	-0.604367777117164	0.445543736205845

Table 4: Training Data based on Tanh function

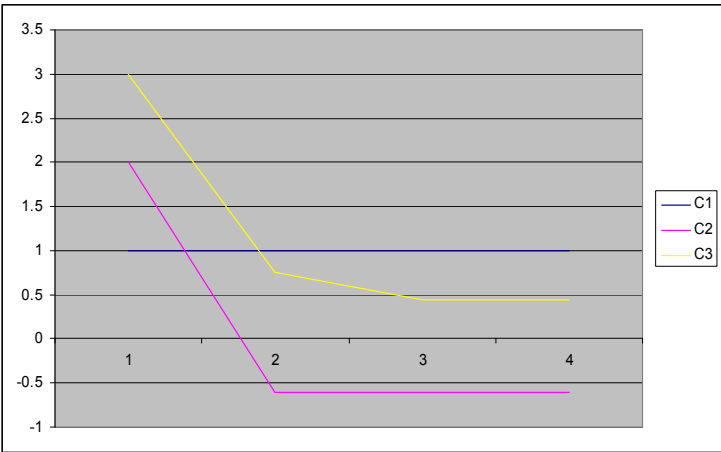


Figure 4: Concept Graphics (Tanh function)

II. Computing FCM weights

A. LSE Method and Experiments

LSE method for computing FCM weights needs Training data and Relationship matrix as inputs and FCM Weight Matrix is output result, as is illustrated in Figure 5.

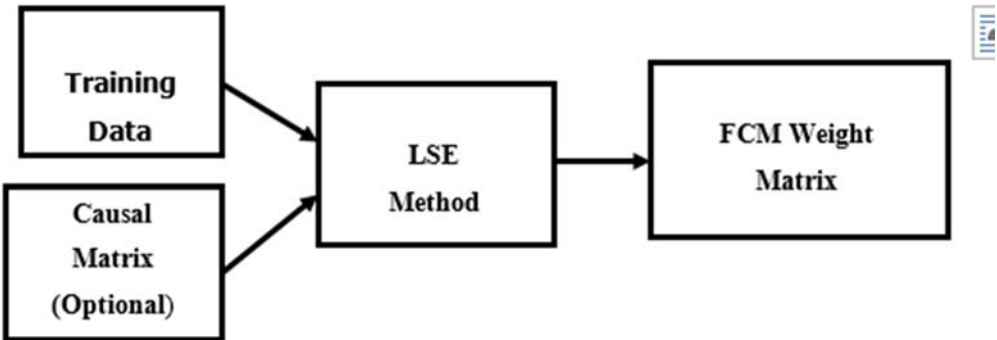


Figure 1: LSE Architecture

Relationship matrix indicates if concept C_i influences concept C_j . It contains values 0 and 1. Value 1 indicates that concept C_i influences concept C_j . If matrix is not specified the LSE method considers all values 1.

The LSE method will be running in three cases:

- Training Data = Table 2
- Training Data = Table 3
- Training Data = Table 4

Case 1 (Training Data = Table 2)

In this case the input file and relationship matrix are constructed as bellow:

3
1 2 3
1 -0.7 1
1 -0.7 0.46
1 -0.7 0.46

a. Input file

0	1	1
0	0	1
0	0	0

b. Relationship matrix

The LSE method is running in the following sub cases:

- f= identical function
- $f = \frac{1}{1+e^{-cx}} ; c=1$
- $f = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

- Running the method in the first sub case we obtain the FCM weights presented in Table 5.

C ₁	C ₂	C ₃
0	-0.6999999999999999	0.5999999999999994
0	0	0.2000000000000001
0	0	0

Table 5: Calculated FCM weights using Training Data (Table 2) and LSE (Identical function)

Comparing the initial weights with weights obtained by LSE method we notice a good approximation as is illustrated in Figure 6.

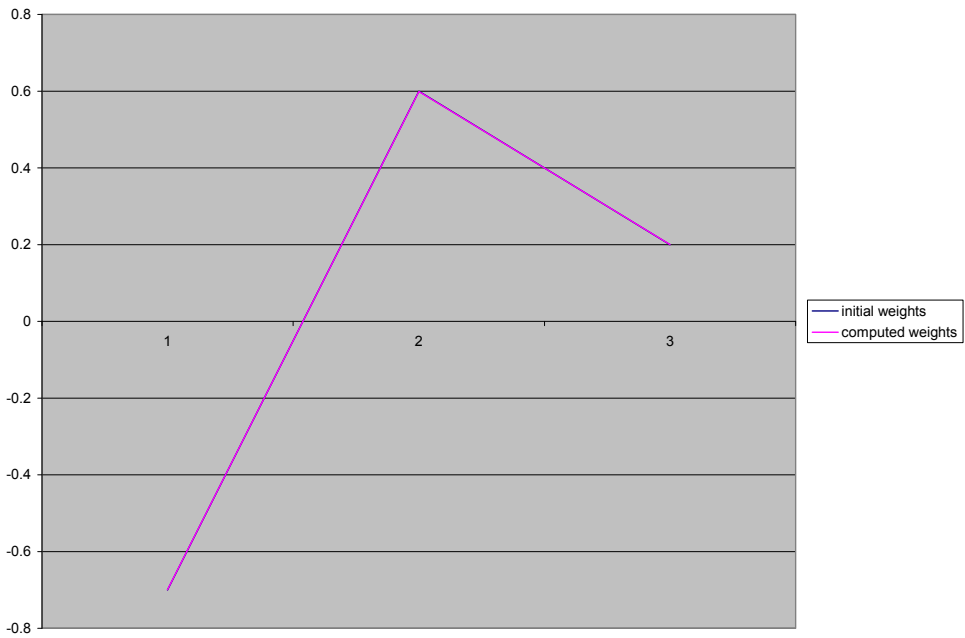


Figure 6: Computed weights versus initial weights (using Table 2 and based on Identical function)

If relation matrix is not specified, the LSE method considers that exist causality relations between any nodes (Table 6):

1	1	1
1	1	1
1	1	1

Table 6: Causal Matrix

Running the LSE method, we obtain the FCM weights presented in Table 7.

C_1	C_2	C_3
0.999999999995465	-0.699999999996825	0.599999999996705
-2.87027118985772E-12	2.00932028902595E-12	0.199999999997914
3.39377560655695E-12	-2.37584436425974E-12	2.46593789026427E-12

Table 7: Calculated FCM weights using Training Data (Table 2) and LSE (Identical function), when there are causality relations between any nodes

We notice that only weight $w_{1,1}$ has changed strongly.

- b. Running the LSE method in the second sub case we obtain the FCM weights presented in Table 8.

C_1	C_2	C_3
0	-1	0.577074262749158
0	0	1
0	0	0

Table 8: Calculated FCM weights using Training Data (Table 2) and LSE (Sigmoid function)

Comparing the initial weights of FCM model with weights obtained by LSE method we notice a bad approximation, as is illustrated in Figure 7.

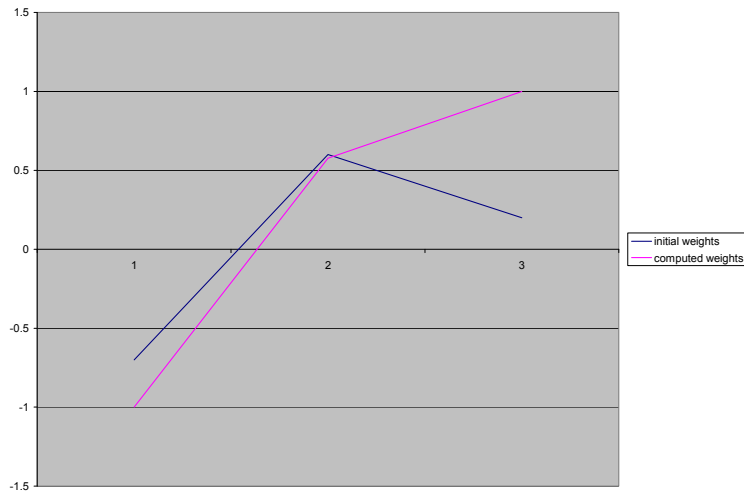


Figure 7: Computed weights versus initial weights (Table 2 and based on Sigmoid function)

- c. Running the LSE method in the third sub case we obtain the FCM weights presented in Table 9.

C_1	C_2	C_3
0	-0.867300527693412	1
0	0	0.719611093166366
0	0	0

Table 9: Calculated FCM weights using Training Data (Table 2) and LSE (Tanh function)

Comparing the initial weights of FCM model with weights obtained by LSE method we notice a bad approximation, as is illustrated in Figure 8.

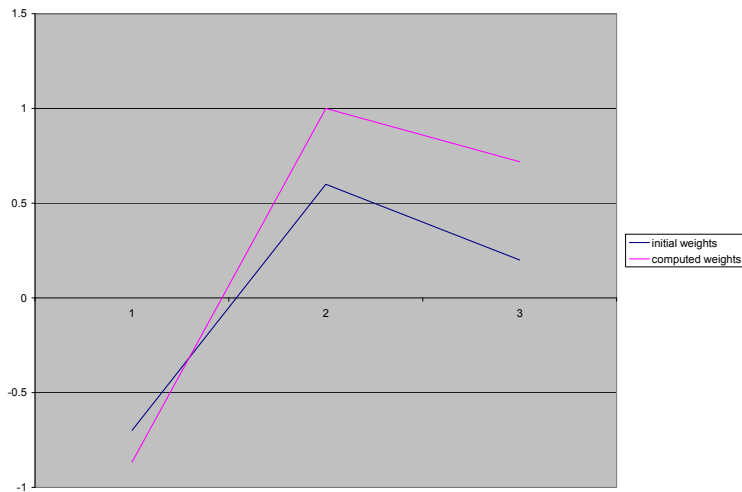


Figure 8: Computed weights versus initial weights (using Table 2 and based on Tanh function)

Case 2 (Training Data =Table 3)

Running the LSE method in the first sub case we obtain the FCM weights presented in Table 10.

C_1	C_2	C_3
0	0.331812227819927	0.646691089812599
0	0	0.0421837444087769
0	0	0

Table 10: Calculated FCM weights using Training Data (Table 3) and LSE (Identical function)

Comparing the initial weights of FCM model with weights obtained by LSE method we notice a bad approximation, as is illustrated in Figure 9.

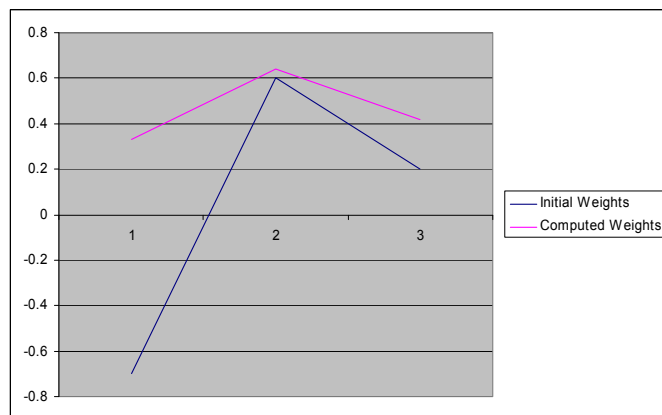


Figure 9: Computed weights versus initial weights (using Table 3 and based on Identical function)

Running the LSE model in the second sub case we obtain the FCM weights presented in Table 11.

C_1	C_2	C_3
0	-0.699999999999993	0.600000000015048
0	0	0.199999999971495
0	0	0

Table 11: Calculated FCM weights using Training Data (Table 3) and LSE (Sigmoid function)

Comparing the initial weights of FCM model with weights obtained by LSE method we notice a good approximation, as is illustrated in Figure 10.

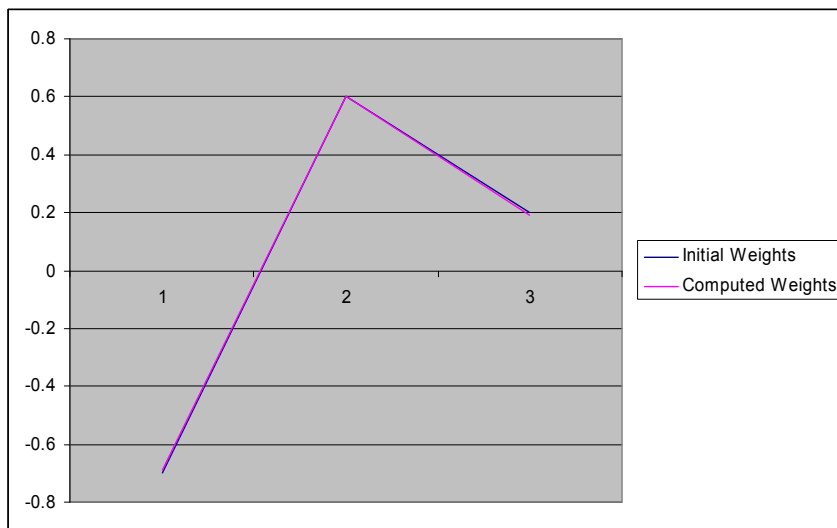


Figure 10: Computed weights versus initial weights (using Table 3 and based on Sigmoid function)

Running the LSE model in the third sub case, we obtain the FCM weights presented in Table 12.

C_1	C_2	C_3
0	0.344863320485106	0.766791080148257
0	0	0.082103160940454
0	0	0

Table 12: Calculated FCM weights using Training Data (Table 3) and LSE (Tanh function)

Comparing the initial weights of FCM model with weights obtained by LSE method we notice a bad approximation, as is illustrated in Figure 11.

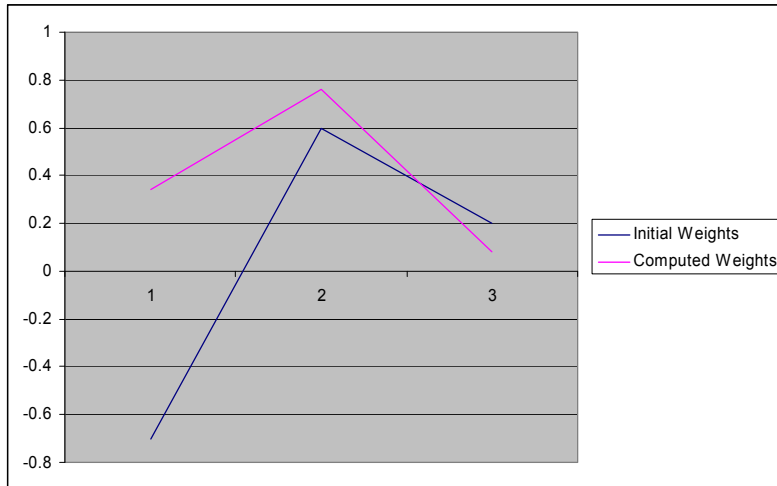


Figure 11: Computed weights versus initial weights (using Table 3 and based on Tanh function)

Case 3 (Training Data= Table 4)

Running the LSE method in the first sub case, we obtain the FCM weights presented in Table 13.

C_1	C_2	C_3
0	-0.604367777117163	0.518886177054436
0	0	0.121353989450664
0	0	0

Table 13: Calculated FCM weights using Training Data (Table 4) and LSE (Identical function)

Comparing the initial weights of FCM model with weights obtained by LSE method we notice a bad approximation, as is illustrated in Figure 12.

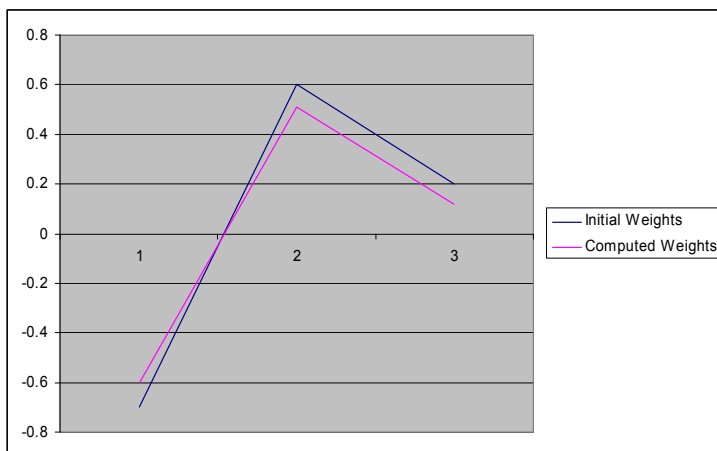


Figure 12: Computed weights versus initial weights (using Table 4 and based on Identical function)

Running the LSE model in the second sub case, we obtain the FCM weights presented in Table 14.

C_1	C_2	C_3
0	-1	0.101579930841112
0	0	0.529929715349475
0	0	0

Table 14: Calculated FCM weights using Training Data (Table 4) and LSE (Sigmoid function)

Comparing the initial weights of FCM model with weights obtained by LSE method we notice a bad approximation, as is illustrated in Figure 13.

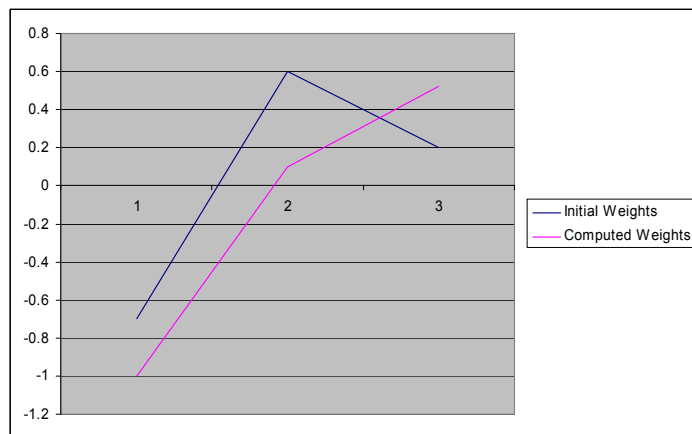


Figure 13: Computed weights versus initial weights (using Table 4 and based on Sigmoid function)

Running the LSE model in the third sub case, we obtain the FCM weights presented in Table 15.

C_1	C_2	C_3
0	-0.6999999999999999	0.5999999999999991
0	0	0.19999999999999988
0	0	0

Table 15: Calculated FCM weights using Training Data (Table 4) and LSE (Tanh function)

Comparing the initial weights of FCM model with weights obtained by LSE method we notice a good approximation, as is illustrated in Figure 14.

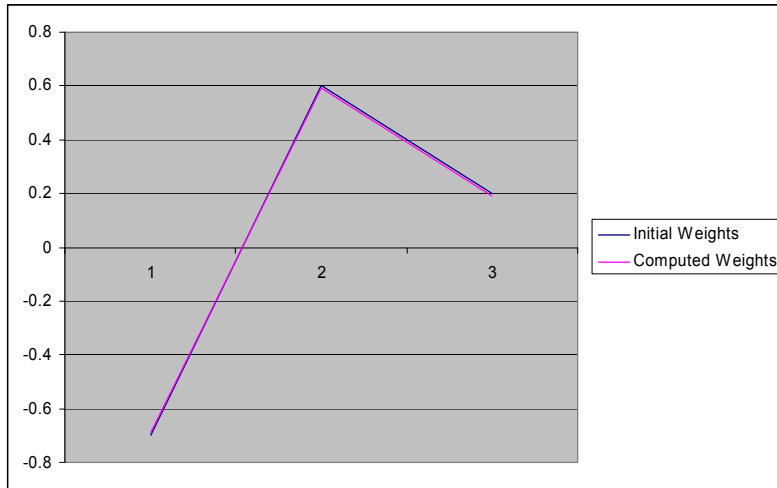


Figure 14: Computed weights versus initial weights (using Table 4 and based on Tanh function)

In conclusion, the weights computed by LSE method are good if LSE method and FCM model use same normalization function.

B. ANFIS Method and Experiments

ANFIS method for computing FCM weights needs Training data and Relationship matrix as inputs. FCM Weight Matrix is output result, as is illustrated in Figure 15.

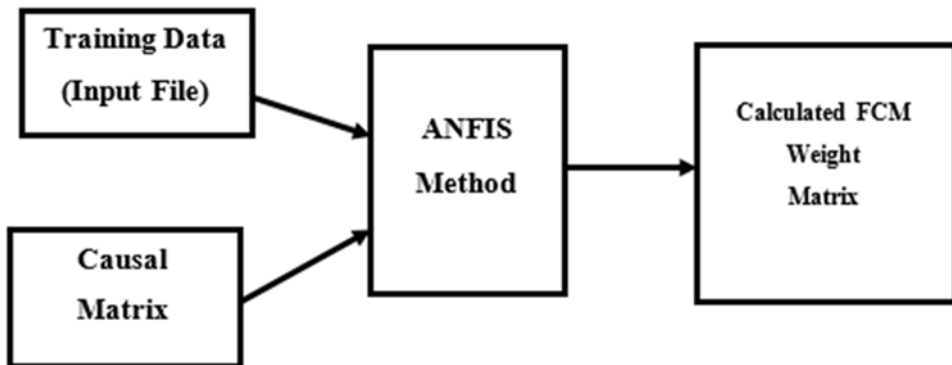


Figure 2: ANFIS Architecture for FCM weights computation

Input file has the following structure:

- First line contains the number of input variables and the number of output variables
- Next lines contain the pairs $\langle X, Y \rangle$ where X is input vector and Y is output vector
- The elements of input/output vectors are separated by one blank.

Relationship matrix indicates if concept C_i influences concept C_j . It contains values 0 and 1. Value 1 indicates that concept C_i influences concept C_j . If matrix is not specified the LSE method considers all values 1.

The ANFIS method will be running in three cases:

- Training Data = Table 2
- Training Data = Table 3
- Training Data = Table 4

Input file is constructed as bellow:

3	3
1	2 3 1 -0.7 1
1	-0.7 1 1 -0.7 0.46
1.03	-0.7 0.46 1 -0.7 0.46

Relation Matrix is constructed as bellow:

0	1	1
0	0	1
0	0	0

Running ANFIS method in first case we obtain the 27 fuzzy rules, 27 simplified rules, 3 FCM weight Matrixes as bellow: If values of concepts C_1 , C_2 , C_3 are x , y , respectively z and number of membership functions $M=3$ then generated rules (27 rules) are:

Generated rules:

Rules: if $x^t = G^{j_1}$ And $y^t = G^{j_2}$ And $z^t = G^{j_3}$



$$\begin{cases} x^{t+1} = a_1 \cdot x^t + a_2 \cdot y^t + a_3 \cdot z^t \\ y^{t+1} = b_1 \cdot x^t + b_2 \cdot y^t + b_3 \cdot z^t \\ z^{t+1} = c_1 \cdot x^t + c_2 \cdot y^t + c_3 \cdot z^t \end{cases}$$

$$j_1=1, 2, 3; \quad j_2=1, 2, 3; \quad j_3=1, 2, 3$$

$$-1 \leq |a_i| \leq 1; \quad -1 \leq |b_i| \leq 1; \quad -1 \leq |c_i| \leq 1$$

From generated rules ANFIS method constructs the following 27 simplified rules:

Simplified rules: if $x^t = G^{j1}$ And $y^t = G^{j2}$ And $z^t = G^{j3}$

$$W = \begin{matrix} & a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{matrix}$$

$j_1=1, 2, 3$

Using FIS function we compute FCM weight matrix (W) as:

$W^0 = \text{FIS}(<1,2,3>)$

$W^1 = \text{FIS}(<1,-0.7,1>)$

$W^2 = \text{FIS}(<1,-0.7,1>)$

$W^3 = \text{FIS}(<1,-0.7,0.46>)$

$W = (W^0 + W^1 + W^2 + W^3)/4$

The FCM matrixes (W^0, W^1, W^2) are listed below:

Time:0		
0	0.000570098968961402	0.100034813585322
0	-0.123418718581486	-0.374072473904171
0	0	0
Time:1		
0	0.0159741610461311	0.0768773957279217
0	-0.0396837846073909	-0.356521313591737
0	0	0
Time:2		
0	-0.0491533379180282	0.109413932237374
0	-0.0256194623915574	-0.380693729046574
0	0	0

Computing arithmetic average of these matrices and comparing it with initial FCM weights we observe a bad approximation.

3. CONCLUSIONS

In conclusion, we notice that our researches based on the LSE method concludes that:

- Good approximation of FCM weights computation when the LSE method and FCM model use the same normalization function;
- Sufficient approximation of FCM weights computation, when FCM model uses Tanh function and LSE method uses the identical function

Next experiments will be done in FUPOL project regarding LSE method, by using real experimental data (Vodno Mountain model/Yantai or Barnsley model). Regarding ANFIS

method, we will make some complex experiments in order to see if this method is suitable for computing FCM weights or it is necessary to combine both LSE and ANFIS methods.

4. REFERENCES

- [1] E. I. Papageorgiou, C. D. Stylios, P. P. Groumpos, Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links, *International Journal of Human-Computer Studies* 2006, 64: 727-743.
- [2] E. I. Papageorgiou, C. D. Stylios, P. P. Groumpos, Active Hebbian learning algorithm to train fuzzy cognitive maps, *International Journal of Approximate Reasoning*, 2004, 37:219-249.
- [3] S. Renals, R. Rohwer, A study of network dynamics, *J. Statist. Phys.*, 1990, 58:825-848.
- [4] C. D. Stylios, P. P. Groumpos, Fuzzy Cognitive Maps: a model for intelligent supervisory control systems, *Computers in Industry*, 1999, 39: 229-238
- [5] C. D. Stylios and P. P. Groumpos, Modeling complex systems using fuzzy cognitive maps, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 2004, 34(1): 155-162.
- [6] A. K. Tsadiras, K. Margaritis, An experimental study of the dynamics of the certainty neuron fuzzy cognitive maps, *Neurocomputing*, 1999, 24: 95-116.
- [7] Massimo Bertolini, Assessment of human reliability factors: A fuzzy cognitive maps approach, *International Journal of Industrial Ergonomics*, 2007, 37: 405–413.
- [8] E.I. Papageorgiou, P.P. Spyridonos, D. Th. Glotsos, C.D. Stylios, P. Ravazoula d, G.N. Nikiforidis, P.P. Groumpos, Brain tumor characterization using the soft computing technique of fuzzy cognitive maps, *Applied Soft Computing*, 2008, 8: 820–828.
- [9] Luis Rodriguez-Repiso, Rossitza Setchi, Jose L. Salmeron, Modelling IT projects success with Fuzzy Cognitive Maps, *Expert Systems with Applications*, 2007, 32: 543–559.
- [10] Moon-Chan Kim, Chang Ouk Kim, Seong Rok Hong, Ick-Hyun Kwon, Forward–backward analysis of RFID-enabled supply chain using fuzzy cognitive map and genetic algorithm, *Expert Systems with Applications*, 2007.
- [11] T. Kottas, Y. Boutalis, and M. Christodoulou, A new method for weight updating in Fuzzy cognitive maps using system feedback, in *Proc. 2nd ICINCO*, Barcelona, Spain, September 13-17, 2005, pp. 202-209.
- [12] Xiaodong Liu and Yanli Zhang, Numerical dynamic modeling and data driven control via least square techniques and hebbian learning algorithm, *International Journal of Numerical Analysis and Modeling*, Volume 7, Number 1, pp. 66–86.
- [13] Luciana S. Solera, Kasper Kokb, Gilberto Camaraa and Antoine Veldkampc, Using fuzzy cognitive maps to describe current system dynamics and develop land cover scenarios: a case study in the Brazilian Amazon, *Journal of Land Use Science* First, 2011, 1–27.
- [14] Jang J. R., ANFIS: Adaptive network based fuzzy inference system, *IEEE Transactions on systems man and cybernetics*, 23(3), pp 665-685, 1993.