Learning Fuzzy Cognitive Maps by a Hybrid Method Using Nonlinear Hebbian Learning and Extended Great Deluge Algorithm

Zhaowei Ren

School of Electronic and Computing Systems
University of Cincinnati
2600 Clifton Ave., Cincinnati, OH 45220

Abstract

Fuzzy Cognitive Maps (FCM) is a technique to represent models of causal inference networks. Data driven FCM learning approach is a good way to model FCM. We present a hybrid FCM learning method that combines Nonlinear Hebbian Learning (NHL) and Extended Great Deluge Algorithm (EGDA), which has the efficiency of NHL and global optimization ability of EGDA. We propose using NHL to train FCM at first, in order to get close to optimization, and then using EGDA to make model more accurate. We propose an experiment to test the accuracy and running time of our methods.

Introduction:

Fuzzy Cognitive Maps (FCM) (1) is a modeling methodology that represents graph causal relations of different variables in a system. One way of developing the inferences is by a matrix computation. FCM is a cognitive map with fuzzy logic (2).FCM allows loops in its network, and it can model feedback and discover hidden relations between concepts (3). Another advantage is that Neuron network techniques are used in FCM, e.g. Hebbian learning(4), Genetic Algorithm (GA) (5), Simulated Anealling (SA) (6).

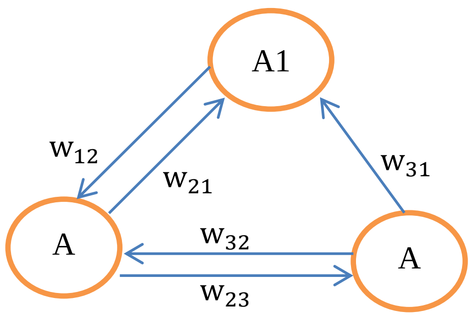

Figure 1 An example of FCM

The structure of FCM is similar to an artificial neuron network, e.g. Figure 1. There are two elements in FCM, concepts and relations. Concepts reflect attributes, qualities and states of system. The value of concepts ranges from 0 to 1. Concepts can reflect both Boolean and quantitative value. For example, a concept can reflect either the state of light (while 0 means off and 1 means on), or water level of a tank. If it reflects a quantitative value, equation [1] can be used for normalization.

$$\text{normalized value} = \frac{A - A_{min}}{A_{max} - A_{min}} \quad [1]$$

where A is the concept value before normalization, and A_{max} and A_{min} are the possible maximum and minimum value of A. Relations reflect causal inference from one concept to another. Relations have direction and weight value. w_{ij} is denoted as the weight value from concept A_i to conceptA_j. For a couple of nodes, there may be two, one or none relations between them. There are three possible types of causal relations:

- $0 < w_{ij} < 1$ the relation from concept A_i to concept A_j is positive. When concept A_i increases (decreases), concept A_j also increases (decreases).
- $-1 < w_{ij} < 0$ the relation from concept A_i to concept A_j is negative. When concept A_i increases (decreases), on the contrary, concept A_j decreases (increases).
- $w_{ij} = 0$ there is no relations between A_i and A_j

When initial state of FCM is given, FCM will converge to a steady state through iteration process. One concept value is computed by the sum of weighted sum of all concepts that may be related to it. In each iteration, concept value is calculated by equation [2].

$$A_i^{k+1} = f\left(A_i^k + \sum_{\substack{j \neq i \\ j=1}}^{N} A_j^k \cdot w_{ji}\right) \quad [2]$$

where A_i^{k+1} is the value of conceptin iteration $k+1$, A_i^k is the value of concept A_i in iteration, and w_{ji}^k is the weight value of the edge from concept A_i to concept A_j. And $f(x) = \frac{1}{1-e^{-\lambda x}}$, which is a transfer function to normalize weight value to [-1,1]. λ is a parameter that determines its steepness.

For example, figure 2 is It is a problem an industrial process control problem (8). There is a tank with two valves where liquids flow into the tank. These two liquid had reaction in this tank. There is another valve which empties the fluid in the tank. There is also a sensor to gauge the gravity of produced liquids (proportional to the rate of reaction) in tank. As described in the figure below

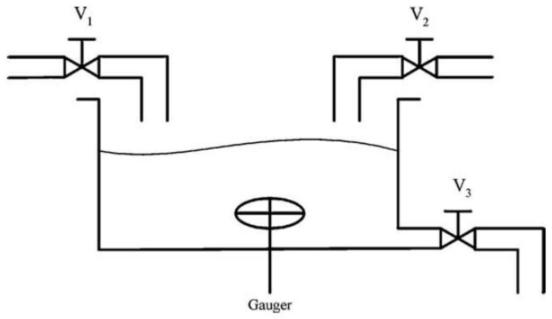

Figure 2 An industrial control problem

There are two constraints of this problem. The first one is to maintain value of G in a particular range, and the second one is to keep height of liquids (T) in a range. Parsopoulos et al. (8) proposes that there should be five concepts: (a) height of liquid in the tank, (b) the state of valve 1, (c) the state of valve 2, (d) the state of valve 3, and (e) the gravity of produced liquid in the tank. Our aim is to find out the causal inference value from one concept to another one.

There are mainly two strategies to learn an FCM. One is to exploit expert domain knowledge and formulate a specific application's FCM (7), this can be used when there is no good automated or semi-automated methods to build this model. If there are multiple domain experts available, each expert choose a value (e.g. very weak, weak, medium, strong, very strong) for the causal effect from one concept to another one; then the values are quantified and combined together into one value between -1 and 1. This strategy has its own shortage: when the problem is complex and need a large number of concepts to describe a system, the cost of expert strategy is very high; moreover, it is difficult to discover new hidden relations by this strategy. Another strategy is to develop a data driven learning method. Input, output and state of a system are recorded when it is running, and these records are used as a neuron network training dataset.

Background

One branch of Fuzzy Cognitive map (FCM) learning is Hebbian learning. Different Hebbian learning has been proposed, for example, Differential Hebbian Learning (DHL)(4), and its modified version Banlanced Differential Hebbian Learning (BDHL)(9). DHL changes weight matrix by the difference of two records, but it did not consider the scenario that multiple concepts have effect on one mutually. BDHL covers this situation, but it is costly owe to lack of optimization. The two Hebbian learning methods that have been used in real world is Active Hebbian learning (AHL) (10) and Nonlinear Hebbian Learning(NHL) (11), and both of them require expert knowledge before computation. AHL explores a method to determine the sequence of active concepts. For each concept, only concepts that may affect it are activated. AHL is fast but requires expert intervention. Experts should determine the desired set of concepts and initial structure of FCM. NHL is a nonlinear extension of DHL. In NHL, before iteration starts,experts have to indicate an initial structure and sign of each non-zero weight. Weight values are updated synchronously, and only with concepts that experts indicate.

Another branch of learning FCM structure is population-based method. Koulouriotis (12)proposes evolution strategies to train fuzzy cognitive maps. In 2007 Ghazanfari et al. (6) proposes using Simulated Annealing (SA) to learn FCM, and he compared Genetic Algorithm (GA)(5) and SA. They concluded that when there are more concepts in the network, SA has a better performance than genetic algorithm. In 2011, Baykasoglu and Adil (13) proposed an algorithm called extended great deluge algorithm (EGDA) to train FCM. EGDA is quite similar to SA, but it demands smaller number of parameters than SA. Population-based method is capable to reach global optimization even if the initial weight matrix is not good, but it is usually computationally costly, especially when the initial weight matrix is far from optimal position. Moreover, population-based methods have many parameters that have to be set before processing. The parameters are set usually by experiences, and then duplicated experiments with different parameters should be made to get better performance. Hebbian learning methods are relatively fast, but their performance depends on initial weight matrix and predefined FCM structure very much. Expert intervention is usually essential. Experts need to indicate a structure before experiments.

The third branch is hybrid method, which takes both the effectiveness of Hebbian learning and global search capability of population-based methods. Papageorgiou and Groumpos (14) proposed a hybrid learning method that combines NHL and Differential Evolution algorithm (DE). First, NHL is used to learn FCM, and then its result is feed to DE algorithm. This method makes uses of both the effectiveness of Hebbian learning and the global search ability of population-based method. The three experiments they did show this hybrid method is capable to train FCM effectively. Zhu et.al(15) proposes another hybrid method which combines NHL and Real-coded Genetic Algorithm (RCGA)

Here I suggest a hybrid method combing NHL and EGDA. EGDA has global search ability and relatively less demand of parameters. If its initial weight matrix is close to optimal condition, it will save much computing expense. Here we use NHL to train FCM roughly first, and then feed its result to EGDA. NHL is picked because it is simple and fast, and it can deal with continuous range of value of concepts

Hybrid Method Using NHL and EGDA

This hybrid method is processed by two stages.

Stage 1 use nonlinear Hebbian learning (NHL) (11)to train FCM

Step 1: Initialize weight matrix W^0 with help of experts and read input concept A^0. We feed the initial weight matrix to feed W^0

Step2:

Calculate A^1 (concept value in iteration 1.Initial values can be denoted as values in iteration 0) by the equation [3]

$$A_i^{k+1} = f(A_i^k + \sum_{\substack{j \neq i \\ j=1}}^{N} A_j^k \cdot w_{ji}^k) \quad [3]$$

where $f(x) = \frac{1}{1-e^{-\lambda x}}$. λ is a parameter that determines increasing rate of curve. It is a transfer function. When x changes from -∞ to ∞, f(x) changes from 0 to 1. Therefore, final result of concept value is still from zero to one.

Step 3:

Use equation [4] to update weights,

$$w_{ji}^k = w_{ji}^{k-1} + \eta_k A_j \ (A_i^k - A_j \ w_{ji}^{k-1}) \quad [4]$$

where η_k is learning rate function, and it decreases as k increases.

Step 4: At the end of each updating, the error function is computed as equation [5]

$$J = \sum_{i=1}^{N} (A_i^{k+1} - A_i^k)^2 \quad [5]$$

where k is the iteration number. There are two termination conditions. One is that value of error function [3] is below a threshold, and the other is there are enough times of iterations. If one of the termination conditions is reached, the iteration ends. Otherwise, go on the next iteration.

For example, now we have time series data of each concept value as Table 1

A1	A2	A3
0.5	0.5	0.1
0.6	0.4	0.2
0.5	0.3	0.3

Table 1 Concept value record

Each tuple is a record of three concept value.

Initial weight matrix is predicted by experts or generated randomly. Here it is as Table 2

W	1	2	3
1	N/A	0.3	0
2	0.7	N/A	0.2
3	-0.6	-0.3	N/A

Table 2 Initial weight matrix

w_{ij} (the weight from concept I to concept j) is the value in line I and column j. For example, $w_{21} = 0.7$

For example, we want to update w_{21} and w_{31} using the first tuple of data. First, we use equation [3] to calculate A_1^1. λ is set to 1 here.

$$A_1^1 = f(A_1^0 + w_{21}^0 A_2^0 + w_{31}^0 A_3^0)$$
$$= f(0.5 + 0.7 * 0.5 + (-0.6) * 0.1)$$
$$= f(0.74)$$
$$= 0.677$$

Then we use equation [2] to update w_{21} and w_{31}.Here the learning rate $\eta = 0.5$

$$w_{21}^1 = w_{21}^0 + \eta * (A_1^0 - A_2 \, w_{21}^0)$$
$$= 0.7 + 0.5 * (0.5 - 0.5*0.7)$$
$$= 0.707$$
$$w_{31}^1 = w_{31}^0 + \eta * (A_1^0 - A_3 w_{31}^0)$$
$$= -0.6 + 0.5 * \big(0.5 - 0.1 * (-0.6)\big)$$
$$= -0.32$$

Other weights are updated as above. Then we got the new weight matrix as below

W	1	2	3
1	N/A	0.475	0
2	0.707	N/A	0.317
3	-0.32	0.565	N/A

And A1=0.677, A2= 0.65, A3=0.55

$$J = (0.677 - 0.5)^2 + (0.65 - 0.5)^2 + (0.55 - 0.1)^2$$
$$= 0.2563$$

If J is larger than termination threshold, then go to step 2. Otherwise, terminate this algorithm and got to stage 2.

Stage 2: use extended great deluge algorithm (EGDA)(13) to train FCM.

Step 1: Initialize the weight matrix with the suggested value from stage 1. The output of step 1 is feed to this step. Assume the weight matrix we got from last step is as Table 3

W	1	2	3
1	N/A	0.3	0
2	0.6	N/A	0.1
3	-0.4	-0.3	N/A

Table 3 Weight matrix after stage 1

Step 2: find a new neighbor of the current weight matrix. For each non-zero weight (because the edge with zero weight does not exist by expert prediction)in the matrix, use the equation below to generate their neighbor.

$$w_{ij}^* = w_{ij} + (2 * \text{random}(\ \ ) - 1) * \text{step}_k \quad [6]$$

where random( ) is a function to generate random value from 0 to 1, and then $2 * \text{random}(\ \ ) - 1$ is a function to generate random value from -1 to 1. step_k is a step size of moving. It is gradually decreased so this algorithm can have a more detailed search during the end of the search.

Step 3: Use equation [1] and new weight matrix to calculate estimated concept value. Then calculate fitness function to determine if new configuration is better than current one. Here we use the total error to be fitness function. The equation is as below

$$\text{Total error} = \frac{1}{KN} \sum_{i=1}^{N} |\text{output}_i^{\text{estimated}} - \text{output}_i^{\text{real}} \quad [7]$$

where *K* is the number of iteration, and *N* is the number of concepts.

Step 4: If the fitness function value of the neighbor configuration is better than tolerance, it is picked as current configuration. And then go to step 5, otherwise, go to step 2. Then reduce the tolerance.

Step 5: If the value of fitness function is better than best condition, update best condition.

For example: First we find a new neighbor for this. step_1 is set as 0.2.

$$w_{12}^{*} = 0.3 + (2 * \mathrm{random}(\ \ ) - 1) * 0.2$$
$$= 0.3 + (2 * 0.5478 - 1)*0.2$$
$$=0.3191$$

(random()=0.5478, generated randomly by matlab)

Using the same equation, we could get new weight matrix

W	1	2	3
1		0.3197	0
2	0.5482		0.2945
3	-0.2453	-0.4583	

Then calculate the concept value $A_1^{\mathrm{estimated}}, A_2^{\mathrm{estimated}}, A_3^{\mathrm{estimated}}$. In this example we use the record below

A1	A2	A3
0.5	0.5	0.1

$$A_1^{\mathrm{estimated}} = f\big(0.5 + 0.5 * 0.5482 + 0.1 * (-0.2453)\big) = 0.6981$$
$$A_2^{\mathrm{estimated}} = f\big(0.5 + 0.5 * 0.3197 + 0.1 * (-0.4583)\big) = 0.5985$$
$$A_3^{\mathrm{estimatd}} = f(0.1 + 0.5 * 0 + 0.5 * 0.2945) = 0.6351$$
$$\text{Total error} = \frac{1}{1 * 3}(|0.6981 - 0.5| + |0.5985 - 0.5| + |0.6351 - 0.1|) = 0.2772$$

If this value is above a tolerance, it is denoted as current configuration, and then it is compared with best configuration to see if it is the best so far. If the new neighbor is not below tolerance, find another neighbor near current one. Reduce tolerance after each search. If the total error is below a threshold or there is enough number of iteration, then this algorithm terminates.

Experiment Design:

There are two steps of our experiment. First we are going to test our method by simulated data, and try to find out the scenario that our method can be most efficient and accurate. On the second step, we will use our method in a real application.

In this experiment, data is generated by a random process. First the number of concepts and density of relations are set. We can try different number of concepts, from small to large, in order to test the performance of this method in network with different complexity. Density represents how many percent of edges exist in a network. It is defined as equation [8].

$$density = \frac{edges\ that\ exist}{all\ edges\ that\ may\ connects\ two\ concepts} = \frac{edges}{number\ of\ nodes(number\ of\ nodes-1)} \quad [8]$$

For example, if we set number of concepts as 5, and density as 0.4, number of edges is computed as below

$$number\ of\ edges = 5 \times (5-1) \times 0.4 = 8$$

then there would be eight edges in this network.

After number of concepts and edges are set, a model can be generated with random weight, and we name it original model. Then random data is generated, and they are fed to equation [1] iteratively, until it reaches a steady state (the error in equation [7] is lower than threshold). The steady state would be a record for simulated data. After a certain time of iteration, if it still cannot reach steady state, a new tuple of data would be generated randomly and fed to equation [1]. After hundreds of times, we will have a series of data as training set. This data is used to learn FCM by our method. The weight matrix we get would be compared with the original model. The error is calculated as equation [9]

$$\text{error} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ i\neq j}}^{N} (w_{ij} - \overline{w_{ij}})^2 \quad [9]$$

where *N* is the number of concepts in this model.

Some other methods (NHL, EGDA, SA) can also be programmed, and compared with this method. These methods will be compared in accuracy and running time, in several conditions.

After simulated experiment, based on the best conditions for our method, we will apply it on a real practical problem.

Conclusion

We propose a hybrid method to learn FCM. Our method has taken advantages of fast speed of NHL and global search ability of EGDA. Moreover, we propose an experiment to test our algorithm, and try to apply it into practice.

Reference

1. Bart and Kosko. Fuzzy cognitive maps. International Journal of Man-Machine Studies 1986; 24: 65.
2. Kosko B. Fuzzy engineering. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. 1997: .
3. Papageorgiou EI, Stylios C, and Groumpos PP. Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links. International Journal of Human-Computer Studies 2006; 64: 727.
4. Dickerson JA, Kosko B. Virtual worlds as fuzzy cognitive maps. Virtual Reality Annual International Symposium, 1993 , 1993 IEEE 1993; 471-477.
5. Stach W, Kurgan L, Pedrycz W, and Reformat M. Genetic learning of fuzzy cognitive maps. Fuzzy Sets Syst 2005; 153: 371-401.
6. Ghazanfari M, Alizadeh S, Fathian M, and Koulouriotis DE. Comparing simulated annealing and genetic algorithm in learning FCM. Applied Mathematics and Computation 2007; 192: 56.

7. Khan MS, Quaddus M. Group decision support using fuzzy cognitive maps for causal reasoning. Group Decis Negotiation 2004; 13: 463-480.

8. Parsopoulos KE, Papageorgiou EI, Groumpos PP, and Vrahatis MN. A first study of fuzzy cognitive maps learning using particle swarm optimization. 2003; 2: 1440.

9. Huerga AV. A balanced differential learning algorithm in fuzzy cognitive maps. 2002; .

10. Papageorgiou EI, Stylios CD, and Groumpos PP. Active hebbian learning algorithm to train fuzzy cognitive maps. International Journal of Approximate Reasoning 2004; 37: 219.

11. Papageorgiou E, Stylios C, and Groumpos P. Fuzzy Cognitive Map Learning Based on Nonlinear Hebbian Rule. In: Gedeon T and Fung L eds. AI 2003: Advances in Artificial Intelligence. Springer Berlin / Heidelberg, 2003: 256-268.

12. Koulouriotis DE, Diakoulakis IE, and Emiris DM. Learning fuzzy cognitive maps using evolution strategies: A novel schema for modeling and simulating high-level behavior. 2001; 1: 364.

13. Baykasoglu A, Durmusoglu ZDU, and Kaplanoglu V. Training fuzzy cognitive maps via extended great deluge algorithm with applications. Comput Ind 2011; 62: 187.

14. Papageorgiou EI, Groumpos PP. A new hybrid method using evolutionary algorithms to train fuzzy cognitive maps. Applied Soft Computing 2005; 5: 409.

15. Yanchun Z, Wei Z. An integrated framework for learning fuzzy cognitive map using RCGA and NHL algorithm. 2008; 1.