

Train Fuzzy Cognitive Maps by Gradient Residual Algorithm

Huiliang Zhang¹, Zhiqi Shen², Chunyan Miao¹

¹ School of Computer Engineering,

²School of Electrical and Electronic Engineering
Nanyang Technological University,
Singapore

{PG04043187, ZQShen, ASCYMiao}@ntu.edu.sg

Abstract—Fuzzy Cognitive Maps (FCM) is a popular technique for describing dynamic systems. A FCM for a dynamic system is a signed graph consisted of relevant concepts and causal relationships/weights between the concepts in the system. With suitable weights defined by experts in the related areas, the inference of the FCM can provide meaningful modeling of the system. Thus correctness of the weights is crucial to the success of a FCM system. Normally the weights are set by experts in the related areas. Considering the possible inefficiency and subjectivity of experts when judging the weights, it is an appealing idea to generate weights automatically according to the samples obtained through observation of the system. Some training algorithms were proposed. However, to our best knowledge, few learning algorithm has been reported to generate weight matrix based on sample sequences with continuous values. In this paper, we introduce a new learning algorithm to train the weights of FCM. In the proposed algorithm, the weights are updated by gradient descent on a squared Bellman residual, which is an accepted method in machine learning. The experiment results show that given sufficient training samples, the correct weights can be approximated by the algorithm. The algorithm proposes a new way for FCM research and applications.

Keywords– Fuzzy cognitive maps

I. INTRODUCTION

Fuzzy Cognitive Maps (FCMs) was introduced [1] as an extension of cognitive maps proposed by political scientist Robert Axelrod [2]. A cognitive map is *signed digraph*, in which nodes are variable concepts and edges represent causal connections between the concepts. The positive or negative sign on a directed edge between two concepts indicates whether the change of the first concept causally increases or decreases the second concept. A matrix consisted of the signs of the edges is used to represent the cognitive map. In the matrix, a positive edge is shown as 1, a negative edge as -1 and 0 for unrelated concepts. Kosko proposed to apply fuzzy logic technique to represent the causal relationships between concepts. Fuzzy causal algebra was proposed to infer how a concept will influence another concept. Then Kosko proposed Bidirectional Associative Memories in neural networks [3]. This provides a theoretical base for simple FCM inference algorithm which is widely used in current FCM researches and

applications. FCM inference is a qualitative approach for a dynamic system, where the gross behaviors of a system can be observed quickly and without the services of an operations research expert [4].

Since proposed, FCM has been researched and applied in many areas. The original application of FCM shown by Kosko was to predict political status changes [1]. Later the application of FCM was extended to simulation of virtual worlds. In [5], FCMs are designed for a virtual sea world. The behaviors of dolphin, fish and shark are predicted based on their behavior FCMs. For example, a FCM for the control of a dolphin consists of concepts such as ‘Hunger’, ‘Food Search’, ‘Chase food’ and so on. Through FCM inference, the status changes of the dolphin from an initial status can be observed. The status of the dolphin will finally reach a stable status or repeat for a cycle of limited statuses. The examples show that FCM output states can guide a cartoon of the virtual world. Besides the applications in virtual worlds, FCM technique is applied to solve real-world problems. For example, FCM applications are used in intelligent intrusion detection system [6], the system to control the valves based on the tank height and the gravity of the liquid in the tank [7], the brain tumor characterization system [8], the web surveillance system [9], and so on. FCM shows high efficiency and accuracy in these systems.

In these systems, it is very important to assign suitable weights to the causal relationships between concepts. In the early FCMs designed by Dickerson and Kosko [5], the weights take value as 1, -1 or 0. It is a rather easy job to assign weights by judging the positive, negative or neutral influences from one concept to another. However, in a real system where weights will take continuous values as suggested in [10], assignment of the weights is dependent on experts’ personal knowledge and subjective judgments, which might not always be precise. So it is very appealing if the weights can be generated automatically. Some automatic algorithms have been proposed. We summarize them in the section of related work. No learning algorithm has been proposed to train FCM based on samples with continuous values. In this paper, we propose a gradient residual method to train the weights with some given sample sequences of a dynamic system. The algorithm updates the weights by gradients of squared

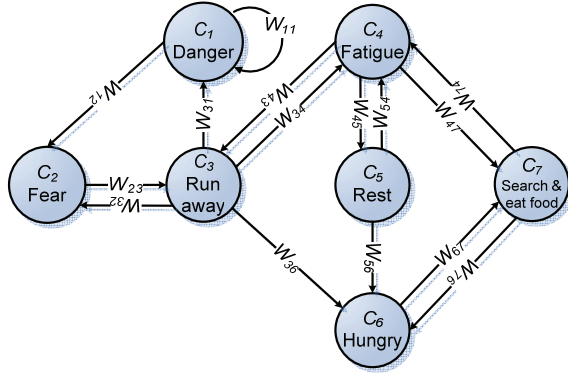


Figure 1. Example of a FCM.

Bellman residual. Experimental results show that the algorithm is very effective if sufficient sample sequences are given.

The rest of the paper is organized as following. We first give a brief illustration of FCM in Section II. Then the gradient algorithm to train FCM weights is explained in Section III and experimental results are shown in Section IV. Other FCM training methods are discussed in related work in Section V. Finally, a conclusion is given.

II. FUZZY COGNITIVE MAP

A. Definition

As introduced in last section, FCM is a directed graph which depicts a dynamic system. The nodes represent the concepts in the system. Weight on an edge between two nodes shows how much a concept will affect the next concept. The weight matrix can be shown as:

TABLE I. WEIGHT MATRIX W FOR FCM IN FIGURE 1.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	1	.89	0	0	0	0	0
C_2	0	0	.75	0	0	0	0
C_3	-.67	-.7	0	.9	0	.89	0
C_4	0	0	-.71	0	.94	0	.41
C_5	0	0	0	-.31	0	.07	0
C_6	0	0	0	0	0	0	.99
C_7	0	0	0	.52	0	-.81	0

TABLE II. THE VALUES OF CONCEPTS IN FCM INTERACTION.

t	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0	.6	.3	.3	.6	.7	.7	.7
1	.598	.58	.45	.603	.637	.438	.719
2	.574	.554	.502	.641	.638	.466	.664
3	.559	.54	.49	.645	.646	.488	.674
4	.557	.539	.487	.644	.647	.484	.679
5	.557	.539	.487	.644	.647	.482	.678
6	.557	.539	.487	.643	.647	.482	.677
7	.557	.539	.487	.643	.647	.483	.677
8	.557	.539	.487	.643	.647	.483	.677

$$W = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1n} \\ W_{21} & W_{22} & \dots & W_{2n} \\ \dots & \dots & \dots & \dots \\ W_{n1} & W_{n2} & \dots & W_{nn} \end{bmatrix}$$

where, n is the number of concepts in the FCM.

An example FCM is shown in Figure 1. The FCM consists of seven concepts, which represent the status of a system to control an animal's behaviors. The possible relationships between the concepts are identified and shown in the figure.

In general, a weight W_{ij} takes values in the fuzzy causal interval $[-1, 1]$. A positive value means that the increase (decrease) of concept C_i will increase (decrease) the value of concept C_j . A negative value indicates the value of C_j will change in an opposite direction with C_i . A zero value means that C_i will not affect C_j . Only none-zero weights are shown in Figure 1. For example, we can say that the value of "Hungry" has no effect on "Rest". However, the value of "Hungry" will affect how the animal decides to "Search & eat food".

At a time t , the status vector of the FCM system is represented as $C^{(t)} = (C_1^{(t)}, C_2^{(t)}, \dots, C_n^{(t)})$. The initial status vector of FCM is shown as $C^{(0)}$. The status of a concept C_i in a time t can be calculated by summing the influences from last time. The equation is shown as:

$$C_i^{(t)} = f\left(\sum_{j=1}^n (C_j^{(t-1)} * W_{ji})\right) \quad (1)$$

where, f is the threshold function.

Normally Sigmoid function is used as threshold function for FCM with continuous concept values. It is shown as:

$$f(x) = \left(1 + e^{-\lambda x}\right)^{-1} \quad (2)$$

where, λ is a parameter deciding the width of the sigmoid function. In this paper, λ takes default value 1.

The initial values of FCM concepts are obtained through measuring the concepts in the real system. Then the inference is performed by continuously calculating the status using the equations (1) and (2). The calculation does not stop until the model reaches a limit cycle or exhibits a chaotic change. The outputs of the FCM inference show how the system will change from the initial status.

Take the FCM example shown in Figure 1. A weight matrix W is decided by the experts in the domain of animal research and shown in Table I. Now we assume that the system starts from an initial status vector, $C^{(0)} = (0.6, 0.3, 0.3, 0.6, 0.7, 0.7, 0.7)$. For example, the fuzzy value for concept "Fear" is 0.3, which means that the animal is in a status of being a little vigilant. The values of concepts at each step of FCM interaction can be calculated by using equation (1) and (2). The outputs are shown in Table II. It can be seen that the FCM

calculation converges at step 8. This means that without disturbs the animal will exhibit same behaviors after step 7.

B. Generate FCM for a System

A big question of FCM application is how to generate the weight matrix as shown in Table I. The normal method to construct a FCM relies heavily on a group of experts of the system. Each expert evaluates a causal relationship with a different fuzzy value depending on his personal experiences. Then the fuzzy rules are composited and de-composited to generate a fuzzy value which stands for the relationship [8]. It is obvious that the correctness of the weight matrix depends on the experts' personal experience and subjective judgments.

An alternative method is to automatically learn the weight matrix from the model of the dynamic system, which can be observed. Thus it is very attractive if the weight matrix could be generated from observed sample sequences as shown in Table II. Several learning algorithms have been introduced as summarized in Section V. In the next section, we will introduce our training algorithm to generate matrix from sample sequences.

III. GRADIENT FCM TRAINING ALGORITHM

A. Problem Definition

The training objective is to generate weight matrix of FCM based on sample sequences. An illustration of the training system is shown in Figure 2.

An initial weight matrix is input to the training algorithm as shown by arrow 1 in Figure 2. The original weights can be designed to show the basic positive or negative relationships between the concepts. However, there is no demand that the original weight W_{ij} must correctly indicate the causal positive or negative relationship from C_i to C_j . A value which is closer to the correct weight will save the training effort. 0 is assigned to W_{ij} if C_i has no influence to C_j . As we will show in the following algorithm, we must be very careful to set 0 as initial value for a weight because a weight with value 0 will not be updated by the training algorithm. If unsure, a random value other than 0 is suggested to be assigned to W_{ij} .

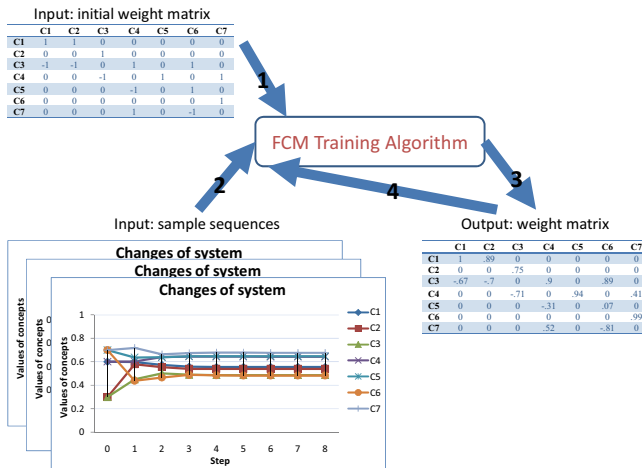


Figure 2. FCM training system.

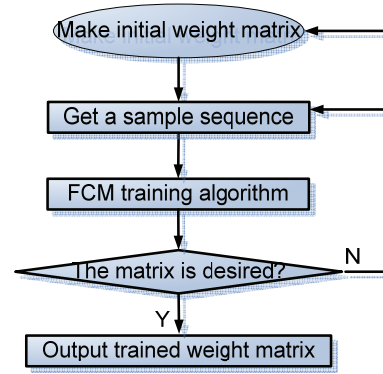


Figure 3. FCM training procedure.

Then some sample sequences are input as indicated by arrow 2 in Figure 2. The FCM training algorithm will train the initial matrix with the first sample sequence. After the training algorithm converges, a resulting weight matrix is output as shown by arrow 3 in Figure 2. The user needs to judge whether the output matrix meets the expectation. If not, the output matrix will be input into the training algorithm for next training as shown with arrow 4 in Figure 2. To be clear, an illustration about this procedure is shown in Figure 3.

Here it must be pointed out that human interference is still very important in the training system. System experts are needed to decide whether the output matrix is reasonable. As we will show later, the gradient training algorithm may be trapped in local minima or return a solution which is not desired. In these cases, a new initial weight matrix may be needed. However, it can be seen that in our training system the experts' jobs to observe system dynamics, make initial weight matrix, and judge whether the output matrix is desirable are easier, which are more objective than the jobs in traditional system to directly figure out the weight values [8]. In fact, as we will discuss in Section V, all other kinds of FCM learning algorithm also need human interference to judge whether their output weight matrix are suitable.

In the following, we will focus on the FCM training algorithm. Assume that a sample sequence is defined as: $\{S^{(0)}, S^{(1)}, \dots, S^{(m)}\}$, where m is the number of steps in the sample sequence. Each status vector is defined as $S^{(i)} = (S_1^{(i)}, S_2^{(i)}, \dots, S_n^{(i)})$, where n is the number of concepts. The training algorithm needs to update the initial weight matrix to get a new matrix such that with the output matrix, the FCM inference will be matching the sample sequence. We set $C^{(0)}$ as $S^{(0)}$. Thus the objective is to update each weight, which affects $C_i^{(t)}$, to decrease the difference between the calculated $C_i^{(t)}$ and the expected value $S_i^{(t)}$. The idea of residual algorithm used for function approximation system in machine learning can be exploited. We will first give a simple introduction about the residual algorithm. Then our gradient FCM training algorithm will be explained.

B. Residual Algorithm

Residual algorithm [11] is designed to modulate the weights of neural network which is used for approximating the expected values/rewards in machine learning. In a learning algorithm, Q is a value function to calculate the reward if an

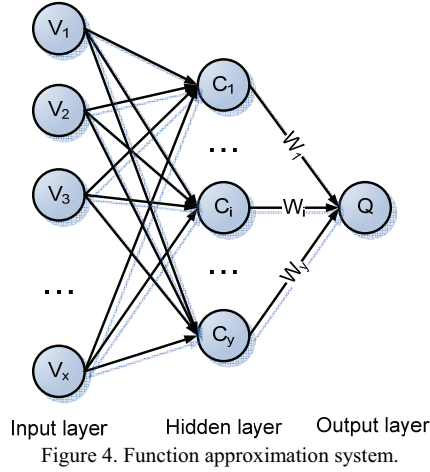


Figure 4. Function approximation system.

action is executed in a state. The objective of residual algorithm is to modulate the parameters in the Q function to make the function better approximate the expected reward.

The mean squared Bellman residual is defined as:

$$E = \frac{1}{|\Sigma|} * \Sigma (R + \gamma * v_{new} - Q)^2 \quad (3)$$

where, R is the immediate reward after an action is executed, γ represents the discount rate of future rewards, v_{new} is the value function for the successor state, Q is the current value function of the current state.

A simple illustration can be seen in Figure 4. Given an input vector $\langle V_1, V_2, \dots, V_x \rangle$, the neural network will output a value. In machine learning, the input variables represent the status and possible actions. The output represents the reward after executing the actions in the status. The residual algorithm will update the weights $\langle W_1, W_2, \dots, W_y \rangle$ to make the output value approximate to the expected value, correct reward. For this example, $E = (R - Q)^2$.

During training, the weights can be updated by performing gradient descent on the mean squared Bellman residual:

$$\Delta W_i = -\alpha * \frac{\partial E}{\partial W_i} \quad (4)$$

where, α is the learning rate.

The gradient residual algorithm guarantees to converge to a minimum, which is sometimes local. Although some mechanisms were proposed to avoid local minimum in gradient learning [12], no algorithm has been able to perfectly solve the problem. To solve this problem is not the intention of this paper. The algorithm proposed in this paper is just an application of the gradient method. In our training system, experts' judgment is needed to decide whether a final solution is suitable. If not, a new training will begin until an acceptable solution is found.

In the following, we will apply the idea of residual algorithm to the problem of FCM training.

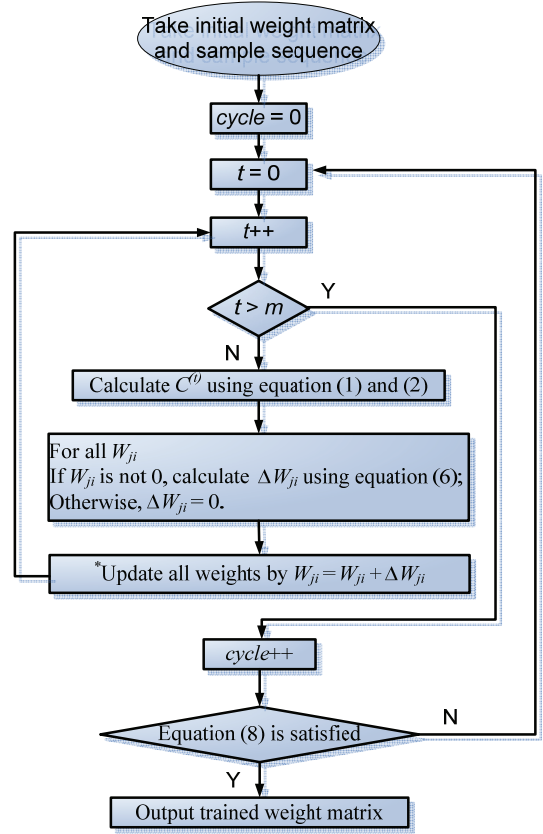


Figure 5. Residual training algorithm.

C. Gradient Algorithm for FCM Training

In FCM training, for each step t , the squared Bellman residual for a concept C_i is defined as:

$$E_i^{(t)} = (C_i^{(t)} - S_i^{(t)})^2 \quad (5)$$

where, $C_i^{(t)}$ is the value of concept calculated by using equations (1) and (2) with the input value of $S^{(t-1)}$.

According to the residual algorithm, the update for W_{ji} , which affects $C_i^{(t)}$, is defined as:

$$\Delta W_{ji} = -\eta * [C_i^{(t)} - S_i^{(t)}] * \frac{\partial C_i^{(t)}}{\partial W_{ji}} \quad (6)$$

where, η is the learning rate.

With the equations (1) and (2), the gradient is calculated by:

$$\frac{\partial C_i^{(t)}}{\partial W_{ji}} = \frac{e^{-\lambda * \sum_{k=1}^n (W_{ki} * S_k^{(t-1)})} * \lambda * S_j^{(t-1)}}{\left[1 + e^{-\lambda * \sum_{k=1}^n (W_{ki} * S_k^{(t-1)})} \right]^2} \quad (7)$$

The weights will be updated at each step and the updated weight matrix will be used in next step.

The training algorithm is shown in Figure 5. Variable *cycle* is used to count how many cycles the FCM is trained. It is noticed that a standard is needed to judge whether the training converges. In this paper, a standard equation is designed as:

$$\sum_{i=1}^n \sum_{j=1}^n \left| \sum_{l=1}^m \Delta W_{ij}^{(l)} \right| < threshold \quad (8)$$

where, the *threshold* value is used to decide whether the sum of changes of weights has been small enough.

It should be pointed out that if W_{ji} is updated to 0 from a non-zero value during the training, a very small value should be assigned to W_{ji} , instead of 0. This will ensure that W_{ji} is updated in future. This also means that if a user is not sure about a weight, a small value other than 0 should be used in the initial weight matrix.

Different training samples may produce different output matrices. However, all the output matrices should be able to satisfy the criteria in Equation (8) when being used to approximate the specific training samples. When more training samples are used to refine the output matrix, the final matrix to satisfy all the training samples will be produced.

We will show the experiment results of this algorithm in next section.

IV. EXPERIMENT

We realized the above training algorithm using Visual J#. The program is run on a desktop PC with Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz, 3.37GB of RAM. The program is single thread and just one core of the CPU is utilized. To show the result of the training algorithm, the FCM system shown in Figure 1 is used as an example. We designed 3

TABLE III. INITIAL WEIGHT MATRIX W FOR TRAINING FCM.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	1	1	0	0	0	0	0
C_2	0	0	1	0	0	0	0
C_3	-1	-1	0	1	0	1	0
C_4	0	0	-1	0	1	0	1
C_5	0	0	0	-1	0	1	0
C_6	0	0	0	0	0	0	1
C_7	0	0	0	1	0	-1	0

TABLE IV. THE OUTPUT WEIGHT MATRIX W .

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	.999	.881	0	0	0	0	0
C_2	0	0	.752	0	0	0	0
C_3	-.671	-.687	0	.859	0	.909	0
C_4	0	0	-.71	0	.94	0	.411
C_5	0	0	0	-.581	0	.311	0
C_6	0	0	0	0	0	0	.99
C_7	0	0	0	.806	0	-1.052	0

experiments with different initial weight matrix.

A. Experiment 1

In the first case, the initial weight matrix is made based on the causal relationships between the identified seven concepts. A positive relationship is assigned a weight, 1. A negative relationship is assigned a weight, -1. Otherwise, 0 is assigned. The initial weight matrix we created is shown in Table III. The sample values shown in Table II was used as the first training sample sequence.

Using parameters η as 0.1 and *threshold* as 0.00001, the training algorithm converges after 90,938 cycles in 2.39 seconds. The output matrix is shown in Table IV. We can see that the output matrix is quite close to the original matrix shown in Table I except the weights associated with C_4 and C_6 . The matrix in Table IV is returned instead of the matrix in Table I because the training algorithm stops at a local minimum before the global optimal solution is found.

The values of concepts through FCM interaction are calculated using the output matrix in Table IV and shown in Table V. Although with the differences in the weights associated with C_4 and C_6 , the calculated values are still very close to the original input sample sequence in Table II.

Compared to Table I, it is obvious that the output matrix in Table IV is not desired. Then we train the output matrix in Table IV with a new sample sequence as shown in Table VI. With the new training sample sequence, the learning algorithm converges in 7,289 cycles in 0.15625 seconds. The output matrix is shown in Table VII. This new output matrix is close to the matrix shown in Table I with negligible differences.

By this example, we can see that given sufficient training samples, the gradient learning algorithm is able to generate weight matrix for FCM system very precisely.

B. Experiment 2

In the second experiment, we will show how the initial input weight matrix will affect the learning. Like Experiment 1, the sequence in Table II is used as the first training sample. A new initial weight matrix is designed by setting all causal relationships in Table III as -1.

This time the learning algorithm converges in 54,787 cycles in 1.46875 seconds. The output matrix is shown in Table VIII. It is noticed that this output matrix is different from the matrix in Table IV.

The FCM inference results using this output weight matrix are very close to the original training sequence in Table II. Using this output weight matrix in Table VIII and the second training sample sequence in Table VI, the training algorithm converges after 7,333 cycles in 0.140625 seconds. The output weight matrix is shown in Table IX. The resultant output weight matrix is very close to the original weight matrix in Table I.

It is shown in Tables VII and IX the first weight of the output matrices is 1.001, which exceeds the range for weights, [-1, 1]. In these cases, the upper bound 1 is used to replace

TABLE V. THE VALUES OF CONCEPTS IN FCM INTERACTION.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0	.6	.3	.3	.6	.7	.7	.7
1	.598	.58	.45	.602	.637	.439	.719
2	.573	.554	.502	.645	.638	.463	.664
3	.559	.54	.49	.645	.647	.489	.673
4	.557	.539	.487	.643	.647	.485	.679
5	.557	.539	.487	.643	.647	.482	.678
6	.557	.539	.487	.643	.647	.483	.677
7	.557	.539	.487	.643	.647	.483	.678
8	.557	.539	.487	.643	.647	.483	.678

TABLE VI. THE NEW TRAINING SAMPLE SEQUENCE.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0	.3	.6	.3	.8	.2	.6	.7
1	.525	.514	.471	.639	.68	.429	.715
2	.552	.534	.483	.642	.646	.472	.665
3	.557	.538	.486	.641	.646	.484	.675
4	.558	.539	.487	.643	.646	.483	.677
5	.558	.539	.487	.643	.647	.483	.677
6	.558	.539	.487	.643	.647	.483	.677

TABLE VII. THE NEW OUTPUT WEIGHT MATRIX W .

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	1.001	.881	0	0	0	0	0
C_2	0	0	.745	0	0	0	0
C_3	-.669	-.69	0	.887	0	.886	0
C_4	0	0	-.706	0	.941	0	.413
C_5	0	0	0	-.309	0	.076	0
C_6	0	0	0	0	0	0	.983
C_7	0	0	0	.527	0	-.811	0

TABLE VIII. THE OUTPUT WEIGHT MATRIX W .

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	.999	.88	0	0	0	0	0
C_2	0	0	.751	0	0	0	0
C_3	-.671	-.686	0	.925	0	.844	0
C_4	0	0	-.71	0	.94	0	.409
C_5	0	0	0	-.053	0	-.235	0
C_6	0	0	0	0	0	0	.991
C_7	0	0	0	.257	0	-.488	0

TABLE IX. THE OUTPUT WEIGHT MATRIX W .

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	1.001	.88	.0	.0	.0	.0	.0
C_2	.0	.0	.748	.0	.0	.0	.0
C_3	-.669	-.689	.0	.899	.0	.877	.0
C_4	.0	.0	-.708	.0	.941	.0	.407
C_5	.0	.0	.0	-.305	.0	.074	.0
C_6	.0	.0	.0	.0	.0	.0	.991
C_7	.0	.0	.0	.515	.0	-.804	.0

1.001 in the final matrix. The same method applies in other similar cases.

From this experiment, we can see that the output weight matrix will not be affected by initial weight matrix given

TABLE X. INITIAL WEIGHT MATRIX W FOR TRAINING FCM.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	.5	.1	0	0	0	0	0
C_2	0	0	.6	0	0	0	0
C_3	-.2	-.4	0	.7	0	.3	0
C_4	0	0	.2	0	-.5	0	.8
C_5	0	0	0	.1	0	.7	0
C_6	0	0	0	0	0	0	.3
C_7	0	0	0	1	0	-.2	0

TABLE XI. THE OUTPUT WEIGHT MATRIX W .

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	.993	.875	.0	.0	.0	.0	.0
C_2	.0	.0	.748	.0	.0	.0	.0
C_3	-.664	-.68	.0	.887	.0	.88	.0
C_4	.0	.0	-.707	.0	.94	.0	.418
C_5	.0	.0	.0	-.332	.0	.075	.0
C_6	.0	.0	.0	.0	.0	.0	.98
C_7	.0	.0	.0	.549	.0	-.807	.0

sufficient training sample sequences. However, we still suggest adopting good initial weight matrix considering existence of local minima in the learning algorithm.

C. Experiment 3

In this time, we assign random values to the initial weights. The initial weight matrix is shown in Table X.

We use the first sample sequence to train the weight matrix. The training algorithm converges after 26,730 cycles in 0.71875 seconds. The output matrix is shown in Table XI. We could see that this matrix is very close to the matrix shown in Table I.

More experiments with different initial weight matrixes which are randomly assigned are not shown. In these experiments, the gradient algorithm is also able to generate solution weight matrixes successfully.

V. RELATED WORKS

The idea to automatically generate weights matrix for FCM has been researched by many researchers. The proposed algorithms mostly fall into two groups, Hebbian learning algorithm and evolutionary algorithm.

Hebbian learning algorithm was first applied in training FCM by Kosko [13]. Before that, Kosko proposed a temporal associative memories (TAMs) algorithm [14]. TAMs can store only a few patterns. Then Kosko proposed to use the Differential Hebbian Law (DHL) to encode the usual binary changes in concepts. The experiment result in [5] shows that the algorithm was quite good at generating FCM weights to replicate the usual binary changes. Then an extension of DHL, Balanced Differential Learning (BDL) algorithm, was proposed in [15]. The new algorithm considered more factors while deciding how to update weights. Experiment results show that this algorithm is better to learn patterns. These two algorithms aim to learn the binary changes of concepts. Two more

extensions of DHL were also proposed to learn the weights of systems with continuous values [7], active hebbian learning (AHL) algorithm and nonlinear hebbian learning (NHL) algorithm. Different to other algorithms, the inputs of AHL and NHL are desirable regions of output concepts. These two algorithms need many user interferences to ensure that proper weights are trained. To the best of our knowledge, no learning algorithm which generates weight matrices based on sample sequences with continuous values has been reported.

Evolutionary algorithm is another frequently used algorithm when learning FCM weights matrix. The general idea is to search the weights matrix which can best replicate the training samples [16]. Red-coded genetic algorithms (RCGA) adds a floating-point extension to basic genetic algorithm, which makes genetic algorithm more effective for training weights with continuous values [17]. Another improvement of genetic algorithm can be seen in [18]. Tabu search method was applied to avoid local optima in genetic algorithms. The comparison results show that Tabu search is faster and can find solutions with better fitness than traditional genetic algorithms. Particle swarm optimization (PSO) is another kind of evolutionary algorithm which comes from swarm intelligence algorithms [19]. Besides these, simulated annealing method for training FCM is proposed recently [20]. Simulated annealing makes a serial of moves from an initial solution before it finds solution or is frozen in local optima. Evolutionary algorithm is an applicable way to find good weight matrix for FCM. However, like in our algorithm, human interference is needed to judge whether an output matrix meets the expectation.

VI. CONCLUSION

In this paper, we propose a gradient residual training algorithm for generating weights of FCM. The algorithm updates weights by trying to decrease the squared Bellman residual. The equations to update the weights and the training system are demonstrated. Then the experiments using the algorithm are shown. The results show that with sufficient training samples, the learning algorithm can generate output weight matrix which is very close to the desired matrix. It is an efficient algorithm for generating FCM weight matrix of a dynamic system given the change models of the system. The observations of system models are normally easier than to estimate the possible influences between the concepts. Thus the algorithm proposes a new direction for FCM research and applications. In future, more experiments will be performed in real applications to test the training system.

The example FCM in this paper contains seven concepts. The number of time steps is thus short (e.g. 8 steps). Some large FCMs with many concepts may generate infinite steps. The residual algorithm works quite well with such cases in our other experiments. In these experiments, the algorithm works well even for a FCM with 80 concepts. Due to the page limit, we cannot display many large matrices. Another point is that in the training, not all steps before convergence are required to train a weight matrix. The benefit of providing more steps is to generate matrix closer to the original matrix. More experiments

and comparative analysis of residual algorithm and the GA algorithm by Stach [21] will be reported soon.

ACKNOWLEDGMENT

This research is financially supported by Singapore National Research Fund.

REFERENCES

- [1] B. Kosko, "Fuzzy Cognitive Maps," *Int. J. Man-Machine Studies*, vol. 24, pp. 65-75, 1986.
- [2] R. M. Axelrod, *Structure of Decision: The Cognitive Maps of Political Elites*: Princeton University Press, 1976.
- [3] B. Kosko, "bidirectional associative memories," *IEEE transactions Systems, Man, and Cybernetics*, vol. 18, pp. 49-60, 1988.
- [4] J. Aguilar, "A survey about fuzzy cognitive maps papers," *International Journal of Computational Cognition*, vol. 3, no. 2, pp. 27-33, 2005.
- [5] J. A. Dickerson, and B. Kosko, "Virtual worlds as fuzzy cognitive maps." pp. 471-477.
- [6] C.-P. Mu, H.-K. Huang, and S.-F. Tian, "Fuzzy cognitive maps for decision support in an automatic intrusion response mechanism." pp. 1789-1794.
- [7] E. I. Papageorgiou, C. Stylios, and P. P. Groumpos, "Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links," *International Journal of Human-Computer Studies*, vol. 64, no. 8, pp. 727-743, Aug, 2006.
- [8] E. I. Papageorgiou, P. Spyridonos, D. Glotsos *et al.*, "Brain Tumor Characterization using the Soft Computing Technique of Fuzzy Cognitive Maps," *Applied Soft Computing*, vol. 8, pp. 820-828, 2008.
- [9] G. Meghabghab, "Mining User's Web Searching Skills through Fuzzy Cognitive State Map vs. Markovian modeling," *International Journal of Computational Cognition*, vol. 1, no. 3, pp. 51-92, September, 2003.
- [10] Y. Miao, Z.-Q. Liu, C. K. Siew *et al.*, "Dynamical cognitive network - an extension of fuzzy cognitive map," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 5, pp. 760-770, Oct, 2001.
- [11] L. C. Baird III, "Reinforcement Learning Through Gradient Descent," School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1999.
- [12] M. Gori, and A. Tesi, "On The Problem Of Local Minima In Backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76-86, January, 1992.
- [13] B. Kosko, *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*: Prentice Hall; Har/Dis edition, 1992.
- [14] B. Kosko, "Hidden Patterns in Combined and Adaptive Knowledge Networks," *International Journal of Approximate Reasoning*, vol. 2, pp. 337-393, 1988.
- [15] A. V. Hueriga, "A balanced differential learning algorithm in fuzzy cognitive maps," *the 16th International Workshop on Qualitative Reasoning*, 2002.
- [16] D. E. Koulouriotis, I. E. Diakoulakis, and D. M. Emiris, "Learning fuzzy cognitive maps using evolution strategies: a novelschema for modeling and simulating high-level behavior." pp. 364-371.
- [17] W. Stach, L. Kurgan, and W. Pedrycz, "Parallel Learning of Large Fuzzy Cognitive Maps." pp. 1584-1589.
- [18] S. Alizadeh, M. Ghazanfari, M. Jafari *et al.*, "Learning FCM by Tabu Search," *International Journal of Computer Science*, vol. 2, no. 2, pp. 142-149, 2007.
- [19] E. I. Papageorgiou, K. E. Parsopoulos, P. P. G. C. D. Stylios *et al.*, "Fuzzy Cognitive Maps Learning Using Particle Swarm Optimization," *International Journal of Intelligent Information Systems*, vol. 25, no. 1, pp. 95-121, July, 2005.
- [20] S. Alizadeh, and M. Ghazanfari, "Learning FCM by chaotic simulated annealing," *Chaos, Solitons & Fractals*, June 2008.
- [21] W. Stach, L. Kurgan, W. Pedrycz *et al.*, "Learning Fuzzy Cognitive Maps with Required Precision Using Genetic Algorithm Approach," *Electronics Letters*, vol. 40, no. 24, pp. 1519-1520, 2004.