Data-Driven Nonlinear Hebbian Learning Method for Fuzzy Cognitive Maps

Wojciech Stach, Lukasz Kurgan, and Witold Pedrycz

Abstract— **Fuzzy Cognitive Maps (FCMs) are a convenient tool for modeling of dynamic systems by means of concepts connected by cause-effect relationships. The FCM models can be developed either manually (by the experts) or using an automated learning method (from data). Some of the methods from the latter group, including recently proposed Nonlinear Hebbian Learning (NHL) algorithm, use Hebbian law and a set of conditions imposed on output concepts. In this paper, we propose a novel approach named data-driven NHL (DD-NHL) that extends NHL method by using historical data of the input concepts to provide improved quality of the learned FCMs. DD-NHL is tested on both synthetic and real-life data, and the experiments show that if historical data are available, then the proposed method produces better FCM models when compared with those formed by the generic NHL method.**

I. Introduction

Fuzzy Cognitive Maps (FCMs), introduced by Kosko [1] in 1986, are a convienient conceptual and computing machinery for modeling and simulation of dynamic systems. They represent knowledge in a symbolic manner and relate states, variables, events, outputs and inputs using a cause and effect approach. When compared to other techniques, FCMs exhibit a number of highly appealing properties. In particular, knowledge representation becomes easy and intuitive. One can easily model feedback relationships and capture hidden dependencies between the concepts. [2]. Applications of FCMs are in various areas including engineering [3], [4], medicine [5], political science [6], economics [7], earth and environmental sciences [8], etc.

There are two main groups of approaches to develop Fuzzy Cognitive Maps: (1) manual methods carried out by expert(s) who have knowledge of both FCMs and the domain of application, and (2) automated or semi-automated methods, which use learning algorithms to establish models from historical data (simulations of concept values). The methods from the latter group exhibit numerous advantages over the manual methods, such as independence of the domain of application which may lead to the development of unbiased models [9]. One of the paradigms used to automate development of FCMs stems from the Hebbian law. The first attempt to learn FCMs using this approach was proposed by Dickerson and Kosko in 1994, and was referred to as Differential Hebbian Learning (DHL) [10]. This method was further extended into Nonlinear Hebbian Learning (NHL) [11]. The NHL algorithm learns FCMs from initial expert-derived FCM model and a set of conditions imposed on output concepts. The algorithm does not use historical data and requires an expert to develop an initial map. To this end, we propose a novel extension to NHL method, called data-driven NHL (DD-NHL) which uses historical data to improve the quality of learned FCM models when compared with generic NHL method. It is also worth emphasizing that the proposed method does not rely on some initial, expert-derived FCM model.

The study is organized as follows. Section II presents background information on Fuzzy Cognitive Maps and motivation of this research. In Section III our algorithm, DD-NHL, is introduced. Section IV and Section V describe experiments that have been performed and elaborate on the results, whereas Section VI summarizes this paper.

II. Background and Motivation

A. Fuzzy Cognitive Maps

FCMs define a given dynamic system by means of concepts associated by mutual cause-effect relations. Each relation is described by a number from interval [-1, 1], which corresponds to its strength. Positive values reflect promoting effect, whereas negative values correspond to inhibiting effect. The value of –1 represents full negative, +1 full positive and 0 denotes neutral relation. Other values correspond to different intermediate levels of causal effect. FCMs are conveniently expressed in the form of graphs. In a graph, the nodes correspond to states, and arrows associated with numbers correspond to relations. The graph representation is equivalent to a square matrix, called *connection matrix*, which stores all weight values of edges between corresponding concepts.

Figure 1 shows an example a process control problem [11], which is modeled by the FCM shown in Figure 2.

This work was supported in part by the Alberta Ingenuity, and by the Natural Sciences & Engineering Research Council of Canada (NSERC)

W. Stach, L. Kurgan, and W. Pedrycz are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada (e-mail: {wstach, lkurgan, pedrycz}@ece.ualberta.ca)

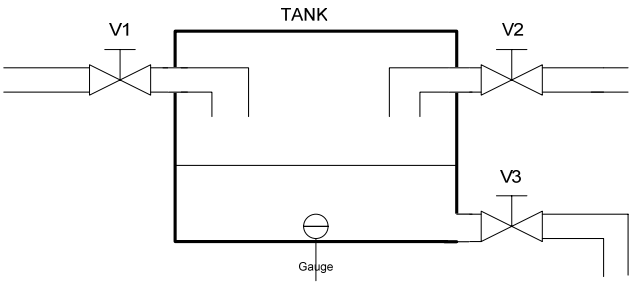
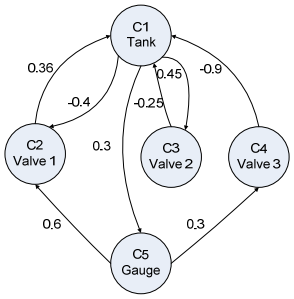


Fig. 1. Simple process control problem

Valve 1 and valve 2 provide two different liquids into the tank. The liquids are mixed and a chemical reaction takes place. The control objective is to maintain the desired level of liquid and its specific gravity. Valve 3 is used to drain liquid from the tank.

Three experts have developed the initial FCM model for this system [11]. It consists of five concepts that are defined as follows:

C1 – the amount of the liquid in the tank
C2 – the state of Valve 1
C3 – the state of Valve 2
C4 – the state of Valve 3
C5 – the specific gravity of the liquid into the tank



	C1	C2	C3	C4	C5
C1	0	-0.4	-0.25	0	0.3
C2	0.36	0	0	0	0
C3	0.45	0	0	0	0
C4	-0.9	0	0	0	0
C5	0	0.6	0	0.3	0

Fig. 2. Example of an FCM graph along with its connection matrix

In FCMs, each concept has a value that reflects the degree to which the concept is active in the system at a particular iteration (discrete time moment). This value, called *activation level*, is a floating-point number between 0 (inactive) and 1 (active). For the above example, activation level of each valve determines degree to which it is open. The value of 0 means that a given valve is closed, value of 1 means that it is fully opened, and other values represent partially opened valve. Similarly, values of the two remaining concepts correspond to different amount of the liquid and its gravity.

Once an FCM has been formed and initial values of all concepts were determined as initial state of the entire system, the model can be simulated. Simulation boils down to calculating future values of concepts at discrete time points based on equation (1), which takes into account the activation levels at the previous iteration and the connection matrix

$$\forall j \in \{1,...,N\}, C_j(t+1) = f\left(C_j(t) + \sum_{i=1}^{N} e_{ij} C_i(t)\right) \quad (1)$$

where: $C_j(t)$ – activation level of concept j^{th} at iteration t
e_{ij} – strength of relation from concept C_i to concept C_j
f – transformation function

The transformation function is used to maintain the values of the weighted sum within a certain range. The normalization hinders quantitative analysis, but, at the same time, it allows to compare activation levels of different concepts.

A snapshot of activation levels of all nodes at a particular iteration defines the system state. It can be conveniently represented by a *state vector*, which consists of the nodes' activation values. *Initial state vector* refers to the system state at the first iteration. Successive states are calculated by iterative application of the formula (1).

Figure 3 shows a sample simulation result of the model from Figure 2 started from initial state vector suggested in [11], i.e. $\mathbf{C}(0) = [0.4, 0.708, 0.612, 0.717, 0.3]$

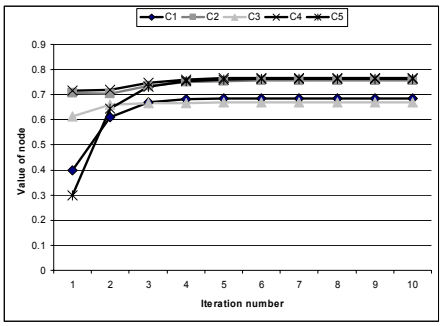


Fig. 3. Sample FCM simulation result

The values of all concepts stabilize after a few iteration steps, and they correspond to a stable state of the system. Particularly interesting are values of concepts C1 and C5 since they represent target variables in this system.

B. Development of Fuzzy Cognitive Maps from Data

Differential Hebbian Learning (DHL) [10] is an automated method for learning FCMs, in particular learning the connection matrices, which is based on the law which correlates changes of causal concepts.

$$de_{ij} = -e_{ij} + dC_i dC_j \quad (2)$$

where de_{ij} is the change of weight between concept i^{th} and j^{th} e_{ij} is the current value of this weight, and dC_i, dC_j are changes in concepts i^{th} and j^{th} values, respectively.

The learning process iteratively updates values of all weights from the FCM graph until the desired structure (connection matrix) is found. Nevertheless, the results of experiments performed with this learning method were inconclusive. As a result, several other learning approaches based on the Hebbian principle have been proposed. One of them, the Nonlinear Hebbian Learning algorithm (NHL) that was recently introduced by Papageorgiou and colleagues [11], [2] is used in this paper. This method adjusts the weights based on initial experts' knowledge, i.e. initial sketch of the map, and additional information on the

modeled system expressed by restrictions imposed on some concepts, to derive the connection matrix. Therefore, the main application of NHL method is to fine-tune the initial model.

The NHL algorithm is based on the nonlinear Hebbian-type learning rule that was introduced for Artificial Neural Networks [12], [13]. More precisely, it uses Oja learning rule originally introduced to learn weights of neurons. Adapted to Fuzzy Cognitive Maps, the rule is expressed as follows

$$e_{ij}(k) = e_{ij}(k-1) + \eta C_j \left(C_i - \mathrm{sgn}(e_{ij}) C_j e_{ij}(k-1) \right) \quad (3)$$

where: C_i, C_j are the current activation values of the concept i^{th} and j^{th} calculated for each iteration according to the formula (1), $e_{ij}(k)$ is the weight value of the relation between concept i^{th} and j^{th} at the iteration k, and η is the learning coefficient.

Hebbian learning principle assumes that the update of the weight e_{ij} is proportional to the product of the C_i and C_j concepts activations. However, this may lead to infinite growth of the weight value. The key idea behind the Oja learning rule is to avoid this effect by using *forgetting* term, which is subtracted from the right hand side. In this particular method, the forgetting term is proportional not only to the value of the weight, but also to the square of the value of the target concept (for e_{ij} C_j is the target concept).

NHL method assumes that all the concepts are synchronously triggered at each iteration and that they synchronously change their values. The learning algorithm takes the initial FCM and initial values of all concepts and iteratively updates the model until the desired map is found.

This method has two termination conditions. The first one utilizes information on desired values of some concepts, which come from expert knowledge or problem specification. These concepts are called Desired Output Concepts (DOCs) and they usually have predefined range of desired values. Depending on the particular problem, learning may terminate when all the desired concepts reach the desired activation levels or when they are close enough to these levels. For the example shown in Figure 2, concepts C1 and C5 have been determined as DOCs. The second condition takes into consideration the variation of the subsequent values of the DOCs and is held if all of them changes less than predefined very small constant e. When the variation of the DOCs is smaller than e it is pointless for the algorithm to continue the learning process.

C. Motivation

The NHL algorithm learns FCMs from initial expert knowledge and a set of conditions imposed on Desired Output Concepts. It does not exploit any additional information that could improve learning and generate more accurate models. We note that historical data that describe given system (a simulation of concept values) are often available. Since the NHL algorithm does not take advantage of these data to improve the learning, in this paper we propose a novel extension to NHL method that utilizes the historical data to develop models of better quality than the models learned using the generic NHL.

The two main objectives of the study can be outlined as follows:

- to propose a novel approach to learn FCMs from data based on NHL algorithm, which is called data-driven nonlinear Hebbian Learning method (DD-NHL)
- to test the proposed method by comparing the quality of learned maps obtained from DD-NHL to the maps obtained from the generic NHL.

III. Data-driven Nonlinear Hebbian Learning method

The generic Nonlinear Hebbian Algorithm (NHL) for learning Fuzzy Cognitive Maps consists of seven steps [2], [11]

NHL Algorithm

STEP 1. Given: values of concepts $\boldsymbol{C}(0)$ and initial connection matrix $\boldsymbol{E}(0)$, and restrictions imposed on desired values of DOCs in the form of inequalities $C_j^{MIN} \le C_j \le C_j^{MAX}$

STEP 2. For each iteration step k

STEP 3. Update the weights according to equation (3)

STEP 4. Calculate $\boldsymbol{C}(k)$ for each concept according to (1)

STEP 5. Evaluate termination conditions using $\boldsymbol{C}(k)$ from STEP 4, $\boldsymbol{E}(k)$, and $\boldsymbol{E}(k\text{-}1)$.

STEP 6. Until both termination conditions are met, go to STEP 2

STEP 7. Return the final connection matrix W_{FINAL}

The two conditions from STEP 6 can be expressed as follows

CONDITION 1 (minimizing the cost function F)

STEP 1. Calculate cost function for each $F = \sqrt{\sum_{DOC_j} \left\| C_j(k) - T_j \right\|^2}$ where T_j is the mean target value of the concept C_j, i.e. $T_j = \dfrac{C_j^{MAX} - C_j^{MIN}}{2}$

The objective of the training process is to determine the set of weights that minimize function F.

CONDITION 2 (terminate algorithm after a limited number of steps)

STEP 1. Calculate the maximum difference e_{max} between $e_{ij}(k)$ and $e_{ij}(k\text{-}1)$

STEP 2. If the absolute value of e_{max} is less than ε return TRUE, otherwise return FALSE

Now, let us assume that historical data are available for the given system. They form a matrix D, where d_{ij} corresponds to the value of i^{th} concept at the j^{th} time point. In

other words, values of the concepts are expressed as time-series. The size of D is KxN where K is the number of available data points and N is the number of concepts in the modeled system. For instance, the data shown in Figure 3 would form a matrix D of size 10x5.

Data-driven Nonlinear Hebbian Algorithm (DD-NHL) utilizes available historical data in STEP 4, which is modified to the following form:

STEP 4. Assign C(k) with the next row of matrix D

Thus, the matrix update is carried out based on the available data, which are used at each iteration step of the algorithm. In case all data are points exploited and the termination conditions are not satisfied, we start using the same data points again (first row of matrix D).

Basically, DD-NHL still uses the Oja learning rule, but instead generating data used for learning only from the current model, it takes advantage of data available for a given system. DD-NHL still needs the initial connection matrix but instead of expert-generated map, it can use a randomly generated initial map.

Since the problem of learning in this case is to obtain an FCM model, which if started from a given initial state vector converges to a state that fulfills certain conditions we also updated the first termination condition. After each iteration of weight update (learning), current FCM is simulated from the initial state vector until a stable state is reached. Then, the values of DOCs from this state are compared with the desired values of DOCs. This procedure guarantees that if the solution is found, it would meet all the learning requirements. Therefore, the Condition 1 is now expressed in the following way:

CONDITION 1 (checking conditions imposed on DOCs)

STEP 1. Simulate the current FCM defined by $\boldsymbol{E}(k)$ starting from the initial condition $\boldsymbol{C}(0)$ until a fixed state is reached

STEP 2. For each C_j that has been defined as DOC_j check whether the fixed value $C_j(n)$ meet the restriction $C_j^{MIN} \le C_j(n) \le C_j^{MAX}$

STEP 3. If there is at least one C_j that does not meet the restriction from STEP 2, return FALSE

STEP 4. Otherwise, return TRUE

IV. Evaluation

In this paper, we used both synthetic and real-life data to evaluate the DD-NHL algorithm. The goal of the experiments is to compare quality of the solutions found by NHL algorithm with DD-NHL method.

A. Data Sets

1) Synthetic data

The synthetic data used in experiments were obtained by generating random FCMs along with random initial vectors. Three groups of data have been prepared with maps that consist of 5, 10, and 20 concepts, respectively. As noted in [14], in practice FCMs are usually relatively small, and typically involve 5–10 nodes. Additionally, for each group we generated two subsets with different map densities (defined as the ratio of the non-zero weights to the total number of weights) equal to 20% and 40%, respectively. Again, our choice is motivated by the results of analysis presented in [14], which reveals that the typical density of FCMs is in the range of 20–30%. Consequently, for the experiments we formed six different setups. In addition, ten independent maps were generated for each setup to assure statistical validity of the results.

2) Real-life data

For real-life experiments we chose an FCM model for a process control problem that was introduced in [11] and already shown in Figure 1 and 2. The model has 5 concepts and the density of the map is 32%. We decided to use a real FCM model rather than raw data since the NHL method requires an initial connection matrix as an input. This way we could investigate the impact of providing the true or a randomized matrix as the input.

B. Experiments

1) Synthetic data

For each setup, we took the generated models and initial vectors and performed simulation until convergence was reached. Then, we arbitrary chose 40% of concepts to be DOCs (this value was selected to be consistent with the real-life model reported in [11]). It corresponds to 2, 4, and 8 DOCs for maps that include of 5, 10, and 20 nodes, respectively. For the selected concepts the desired range of values were established as ±0.1 of the stable value. It means that if a given concept C_i stabilized after the simulation at the activation level of a, $\min C_i$ and $\max C_i$ has been calculated as $a-0.1$, and $a+0.1$, respectively. These values were used as the first termination criterion in both NHL and DD-NHL methods. The second criterion e was set to 0.02 as suggested in [11].

The learning procedure for both NHL and DD-NHL was carried out as follows. Firstly, we randomly generated initial map. Next, we performed learning using both NHL and DD-NHL. Additionally, if the termination criteria could not be met after 10000 iterations, the procedure was restarted with a new initial map.

2) Real-life data

The model of a process control system from [11] was used to perform two groups of experiments:

1. In the first one, we took the initial model proposed by experts (Figure 2) and performed learning with NHL method. The map that has been established met all the restrictions defined in [11], i.e. $0.68 \le N1 \le 0.70$ and $0.74 \le N5 \le 0.80$, and, therefore, was considered as the desired model of this system for the remaining experiments. Next, we generated data by simulating this

model, and performed learning using DD-NHL method.

2. In the second group of experiments, we randomly chose initial matrix and carried out learning using both NHL and DD-NHL using the same procedure as for single experiment with synthetic data.

C. Evaluation Criteria

We considered three evaluation measures to assess the performance of the proposed method. They are consistent with the FCM evaluation criteria proposed in [14] and they have been used with experiments on both synthetic and real-life data:

- *in-sample error* – difference between the available data, and data generated by simulating the learned model from the same initial vector. The criterion is defined as a normalized average error between corresponding concept values at each iteration between the two state vector sequences.

$$error\_initial = \frac{1}{(K-1)\cdot N}\sum_{t=1}^{K-1}\sum_{n=1}^{N}\left|C_n(t)-\hat{C}_n(t)\right| \quad (4)$$

where $C_n(t)$ is the value of a node n at iteration t in the input data, $\hat{C}_n(t)$ is the value of a node n at iteration t from simulation of the learned model, K is the input data length, and N is the number of concepts.

- *out-of-sample error* – evaluation of the generalization capabilities of the learned FCM. To compute this criterion, both the desired and learned models are simulated from ten randomly chosen initial state vectors. Subsequently, the value of the measure error_initial is computed for each of the simulations to compare state vector sequences generated by these models, and an average of these values is computed.

$$error\_behavior = \frac{1}{P\cdot(K-1)\cdot N}\sum_{p=1}^{P}\sum_{t=1}^{K-1}\sum_{n=1}^{N}\left|C_n^p(t)-\hat{C}_n^p(t)\right| \quad (5)$$

where $C_n^p(t)$ is the value of a node n at iteration t for data generated by desired model started from p^{th} initial state vector, $\hat{C}_n^p(t)$ is the value of a node n at iteration t for data generated by learned model started from p^{th} initial state vector, K is the input data length, N is the number of concepts, and P is the number of random initial state vector.

- *final-state accuracy* – evaluation of meeting the restrictions on DOCs. The model is simulated from the initial condition until fixed state is reached, and then the following formula is used

$$ACC = \frac{\overline{DOC_i}}{DOC_i}\cdot 100\% \quad (6)$$

where $\overline{DOC_i}$ is the number of DOCs that meet the restrictions after simulating corresponding model from the initial vector, and DOC_i is the total number of DOCs. This measure is calculated for both in-sample and out-of-sample experiments.

TABLE I
IN-SAMPLE ERROR RESULTS FOR SYNTHETIC DATA

		NHL		DD-NHL	
SIZE	DENSITY	IN-SAMPLE	ACC	IN-SAMPLE	ACC
5	20%	0.153 (0.015)	70	0.129 (0.013)	100
	40%	0.149 (0.014)	80	0.129 (0.008)	100
10	20%	0.182 (0.017)	60	0.176 (0.015)	100
	40%	0.193 (0.009)	70	0.180 (0.016)	100
20	20%	0.213 (0.015)	80	0.180 (0.018)	100
	40%	0.210 (0.015)	60	0.207 (0.014)	100

TABLE II
OUT-OF-SAMPLE ERROR RESULTS FOR SYNTHETIC DATA

		NHL		DD-NHL	
SIZE	DENSITY	OUT-OF-SAMPLE	ACC	OUT-OF-SAMPLE	ACC
5	20%	0.152 (0.007)	70	0.129 (0.011)	100
	40%	0.149 (0.011)	80	0.129 (0.015)	100
10	20%	0.182 (0.014)	60	0.175 (0.010)	100
	40%	0.193 (0.016)	70	0.180 (0.011)	100
20	20%	0.213 (0.014)	80	0.180 (0.014)	100
	40%	0.210 (0.017)	60	0.207 (0.016)	100

V. RESULTS

A. Synthetic Data

Table I and Table II summarize the experimental results for the synthetic data. Reported values have been calculated as averages obtained from 10 independent experiments (with different models) for each setup. The rows correspond to different experimental setups in terms of maps' sizes (5, 10, and 20) and densities (20% and 40%), and the value in each cell expresses the average value of a corresponding criterion. For in-sample and out-of-sample errors the average values are followed by standard deviations (in brackets) across all ten runs.

Experimental results show that the data-driven approach is on average better when it comes to the in-sample errors, i.e., average errors equal 0.183 for NHL vs. 0.167 for data-driven NHL. This is because we use historical data for learning data-driven NHL, whereas the generic NHL algorithm takes into consideration only the final state. However, more important are results included in Table II, as they determine how well a given model captures (generalizes) the knowledge of the target domain. This is because out-of-sample experiments are performed on

previously unseen data. They show that DD-NHL consistently, over different map sizes and densities, produces better FCM models when compared with NHL.

The quality of learning decreases slightly with the increase the map size for both methods. The out-of-sample errors are 40% worse for 20 nodes maps when compared with errors for 5 nodes in case of NHL, and 49% worse in case of DD-NHL. Also, we note that the map density does not have significant influence on the learning process.

Taking into consideration the last criterion, i.e. final-state accuracy, the advantage of using DD-NHL method becomes evident. The solutions found by NHL method do not meet the learning objective in 20-40% of experiments. Our method is guaranteed to meet the conditions for DOCs for the in-sample experiments. However, it turns out that the DOC conditions were also satisfied in all out-of-sample experiments performed with DD-NHL method.

Table III covers the results of statistical analysis of the differences between the results produced by the NHL and DD-NHL We performed paired t-test at 95% confidence between the corresponding pairs of 10 experiments performed with NHL and DD-NHL methods.

TABLE III
OUT-OF-SAMPLE STATISTICAL RESULTS COMPARISON THROUGH PAIR T-TESTS

		NHL vs. DD-NHL
SIZE	DENSITY	T-VALUE
5	20%	5.43
	40%	5.52
10	20%	5.63
	40%	5.08
20	20%	7.27
	40%	1.93

Since the *critical t-value* at 95% confidence equals 2.26, the results show that the differences are statistically significant, i.e., the DD-NHL method provides statistically significantly lower error rate, for 5 out of 6 different setups. The difference is not statistically significant only for the largest map with the high desitiy.

B. Real-life Data

Tables IV and V show the results for the first experiment. We note that the NHL method was used to generate the desired model, and therefore all the error measures are equal 0 in this case.

TABLE IV
IN-SAMPLE ERROR RESULTS FOR THE FIRST EXPERIMENT WITH REAL MODEL

		NHL		DD-NHL	
SIZE	DENSITY	IN-SAMPLE	ACC	IN-SAMPLE	ACC
5	20.6%	0	100	0.087	100

TABLE V
OUR-OF-SAMPLE ERROR RESULTS FOR THE FIRST EXPERIMENT WITH REAL MODEL

		NHL		DD-NHL	
SIZE	DENSITY	OUT-OF-SAMPLE	ACC	OUT-OF-SAMPLE	ACC
5	20.6%	0	100	0.091	100

	C1	C2	C3	C4	C5
C1	0.00	0.40	0.48	-0.76	0.07
C2	-0.31	0.00	0.07	0.08	**0.63**
C3	-0.17	0.07	0.00	0.07	0.07
C4	0.07	0.08	0.07	0.00	0.35
C5	0.35	0.08	0.07	0.08	0.00

	C1	C2	C3	C4	C5
C1	0.00	0.34	0.47	-0.87	0.21
C2	-0.22	0.00	0.03	0.05	**0.36**
C3	-0.18	0.03	0.00	0.03	0.03
C4	0.00	0.04	0.03	0.00	0.18
C5	0.50	-0.03	0.04	-0.02	0.00

Fig. 4. NHL vs. DD-NHL models for the first experiment

The DD-NHL learning performed by starting from the initial matrix generated by experts (Figure 2) resulted in better learning quality when compared to results on synthetic data. This is because the initial map was similar to the desired map. Figure 4 illustrates connection matrices of the models found by NHL and DD-NHL. Bolded are values that differ by more that 0.25 between the two solutions. The two models are very similar and differ by more than 0.25 just for one weight. On average, the difference is 7%.

Tables V and VI report experimental results for the second experiment. Similarly to the experiments for synthetic data, 10 independent experiments were carried out and the average values are shown in the tables.

TABLE VI
IN-SAMPLE ERROR RESULTS FOR THE SECOND EXPERIMENT WITH REAL MODEL

		NHL		DD-NHL	
SIZE	DENSITY	IN-SAMPLE	ACC	IN-SAMPLE	ACC
5	20.6%	0.158 (0.009)	60	0.135 (0.011)	100

TABLE VII
OUR-OF-SAMPLE ERROR RESULTS FOR THE SECOND EXPERIMENT WITH REAL MODEL

		NHL		DD-NHL	
SIZE	DENSITY	OUT-OF-SAMPLE	ACC	OUT-OF-SAMPLE	ACC
5	20.6%	0.160 (0.010)	70	0.137 (0.012)	100

The tests show that DD-NHL produces FCMs that are 14% better (they reduce the corresponding error rates by (0.158-0.135)/0.158=14.6% for in-sample and (0.160-0.137)/0.160=14.4%) when compared with maps generated by NHL algorithm for both in-sample and out-of-sample measures. The ACC criterion show that the DD-NHL method is capable to find models that fulfill the learning objectives (restrictions imposed on DOCs), whereas the NHL approach satisfies the objectives in 60-70% of

experiments. Similarly to results from Table III, the statistical analysis of the results was performed. The paired t-test value between NHL and DD-NHL was equal to 5.72, which means that the difference is statistically significant at 95% confidence.

VI. Conclusions

In this paper, a novel method for automated learning of FCMs from data, named DD-NHL, is introduced and experimented with. The proposed method applies a non-linear Hebbian principle and available data to generate FCM models. The main idea behind the DD-NHL method was to use historical data to improve learning quality. When compared to NHL method for learning FCMs, in DD-NHL the stopping criterion was modified to achieve models that fulfill the initial requirements on certain, predefined concepts called Desired Output Concepts (DOCs).

Experimental results for both synthetic and real-life data show that DD-NHL is capable of forming better quality FCMs when compared with the constructs resulting from the NHL algorithm. Results for synthetic data are on average 9% better for DD-NHL for the out-of-sample tests. Both methods produce slightly worse solutions for larger maps with both errors growing linearly. Experiments with real-life data show that DD-NHL method provides satisfactory solutions when started from a matrix that is predefined by the expert and close to the desired solution. Comparative analysis shows the advantages of using DD-NHL over NHL method, in terms of both lower in-sample and out-of-sample errors as well as better ability to satisfy the conditions set on DOCs; the results are consistent with these obtained for synthetic data.

To sum up, the proposed DD-NHL algorithm learns better models than the generic NHL method. However, this improved quality requires the availability of historical data. In contrast, NHL algorithm does not require historical data, but it relies on expert-derived initial map and a set of conditions on DOCs.

References

[1] B. Kosko, "Fuzzy cognitive maps", *International Journal of Man-Machine Studies*, vol. 24, pp. 65-75, 1986

[2] E. I. Papageorgiou, C. Stylios, P. P. Groumpos, "Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links", *International Journal of Human-Computer Studies,* vol. 64, pp. 727-743, 2006

[3] W. Stach, L. Kurgan, W. Pedrycz, and M. Reformat, "Parallel fuzzy cognitive maps as a tool for modeling software development project," *North American Fuzzy Information Processing Society Conference (NAFIPS'04)*, pp. 28–33, 2004

[4] M. A. Styblinski, and B. D. Meyer, "Signal flow graphs versus fuzzy cognitive maps in application to qualitative circuit analysis," *International Journal of Man–Machine Studies*, vol. 35, pp. 175–186, 1991

[5] P. R. Innocent, and R. I. John, "Computer aided fuzzy medical diagnosis," *Information Sciences*, vol. 162, no. 2, pp. 81–104, 2004

[6] M. Khan, and M. Quaddus, "Group decision support using fuzzy cognitive maps for causal reasoning," *Group Decision and Negotiation Journal*, vol. 13, no. 5, pp. 463–480, 2004

[7] D. Kardaras, and G. Mentzas, "Using fuzzy cognitive maps to model and analyse business performance assessment," in *Advances in Industrial Engineering Applications and Practice II*, J. Chen, and A. Mital, (Eds), pp. 63–68, 1997

[8] R. Giordano, G. Passarella, V. F. Uricchio, and M. Vurro, "Fuzzy cognitive maps for issue identification in a water resources conflict resolution system," *Physics and Chemistry of the Earth*, vol. 30, no. 6–7 (Special Issue), pp. 463–469, 2005

[9] W. Stach, L. A. Kurgan, and W. Pedrycz, "A survey of fuzzy cognitive map learning methods," In: P. Grzegorzewski, M. Krawczak, and S. Zadrozny, (Eds.), *Issues in Soft Computing: Theory and Applications*, Exit, pp. 71–84, 2005

[10] J. A., Dickerson, and B. Kosko, "Virtual worlds as fuzzy cognitive maps", *Presence*, vol. 3, no. 2, pp. 173-189, 1994

[11] E. I. Papageorgiou, C. D. Stylios, and P. P. Groumpos, "Fuzzy cognitive map learning based on nonlinear Hebbian rule," In: T. D. Gedeon, and L. C. C. Fung, (Eds.), *Lecture Notes in Artificial Intelligence*, Springer–Verlag, vol. 2903, pp. 254–266, 2003.

[12] E. Oja, "Neural networks, principal components and subspaces", *International Journal of Neural Systems,* vol. 1, pp. 61-68, 1989

[13] E. Oja, H. Ogawa, J. Wangviwattana , "Learning in nonlinear constrained Hebbian networks", In: Kohonen T., et al. (Eds.), Artificial Neural Networks, North-Holland, pp. 385-390, 1991

[14] W. Stach, L. Kurgan, W. Pedrycz, and M. Reformat, "Genetic learning of fuzzy cognitive maps," *Fuzzy Sets and Systems*, vol. 153, no. 3, pp. 371–401, 2005