Training Fuzzy Cognitive Maps Using Gradient-Based Supervised Learning

Michal Gregor, Peter Groumpos

Training Fuzzy Cognitive Maps using Gradient-based Supervised Learning

Michal Gregor[1] and Peter P. Groumpos[2]

[1] Department of Control and Information Systems,
Faculty of Electrical Engineering, University of Žilina
michal.gregor@fel.uniza.sk
[2] Laboratory for Automation and Robotics,
Department of Electrical and Computer Engineering, University of Patras
groumpos@ece.upatras.gr

Abstract. The paper considers a novel approach to learning the weight matrix of a fuzzy cognitive map. An overview of the state-of-the-art learning methods is presented with a specific emphasis on methods initially developed for artificial neural networks, and later adapted for FCMs. These have mostly been based on the concept of Hebbian learning. Inspired by the amount of success these methods have faced in the past, the paper proposes a new approach based on the application of the delta rule and the principle of backpropagation, both of which were originally designed for artificial neural networks as well. It is shown by simulation experiments and comparison with the existing approach based on nonlinear Hebbian learning that the proposed approach achieves favourable results, and that these are superior to those of the existing method by several orders of magnitude. Finally, some possible lines of further investigation are suggested.

Keywords: fuzzy cognitive maps, learning, backpropagation.

1 Introduction

In the past several learning methods from the theory of artificial neural networks (ANNs) have been introduced into the theory of fuzzy cognitive maps (FCMs), and have become successful tools for either learning the weight matrix of the FCM from scratch, or tuning an initial weight matrix provided by a group of experts.

Most notable among such approaches were those based on the concept of Hebbian learning. It has been shown before that Hebbian learning can indeed be used to train an FCM from historical data (train it to perform regression). The approach is known as data-driven nonlinear Hebbian learning (DD-NHL). However, Hebbian learning has originally been proposed as an unsupervised learning approach, and it is therefore not necessarily best suited for such application.

In this paper we propose an alternative approach, which too is based on a learning method originally designed for ANNs, that is to say on the delta rule and the backpropagation principle.

The following sections will provide a brief overview of the DD-NHL approach. It will then proceed to give some essentials concerning the theoretical background concerning the delta rule, and backpropagation. The details concerning the method proposed in this paper will be discussed. Finally, experimental results will be presented and evaluated, and a comparison with DD-NHL will be provided.

2 Fuzzy Cognitive Maps

Fuzzy cognitive maps (FCMs) are a symbolic representation for the description and modelling of complex systems [1]. They can be expressed and visualized using a weighted directed graph. The nodes of such graph represent the concepts associated with the modelled system. Every concept C_i is associated with its activation value A_i.

The edges in the graph are directed and weighted. The weights $w_{ij} \in [-1, 1]$ express causal relationships between the concepts. If $w_{ij} > 0$, we say that concept C_i causes C_j. If $w_{ij} < 0$, concept C_i has negative influence on the activation value of C_j. If $w_{ij} = 0$, there is no link.

At every time step activation values are updated. The update is synchronous. The update rule has several distinct forms. We will make use of the most general one proposed in [2] (the notation has been modified for the sake of consistency):

$$A_i^{(k+1)} = f\left(\sum_{j=1}^{N} A_j^{(k)} w_{ji}\right), \tag{1}$$

where N is the number of concepts, $A_i^{(k)}$ is the activation value of concept C_i at time step k. f is the squashing function, which squashes the dot product $\sum_{j=1}^{N} A_j^{(k)} w_{ji}$ into some convenient interval.

Most often, f is either the sigmoid function, which squashes the dot product into interval $[0, 1]$, or the hyperbolic tangent, which yields the interval $[-1, 1]$. The weight matrix of the FCM is usually constructed by experts. There are several approaches which make the task easier and more reliable – for the discussion of these, the reader may refer to [1, 3] for an instance.

2.1 Fuzzy Cognitive Maps and Learning

There are two main classes of problems in the theory of FCMs to which the existing learning methods apply: (a) *the regression problem*, that is how an FCM can be trained as a regression model for a given dataset; (b) *the attractor problem*, that is to say given an initial FCM, how can we shift its attractor to a desired point, encode a given limit-cycle, etc.

In this paper we shall focus on the regression problem. The methods that can address the regression problem have very useful applications – given data from a

real system, they can make the FCM automatically learn to model the system. They can also be used to fine-tune an existing model designed by experts.

Several learning principles originally developed for ANNs have previously been applied to FCMs. These approaches were based on the concept of Hebbian learning. They come in several distinct flavours which will be listed hereinafter.

The first among the approaches inspired by Hebbian learning is the so-called *differential Hebbian learning* (DHL) [2, 4]. It has been shown that the rule is able to encode some sequences into the FCM. However it is not capable of encoding an arbitrary sequence. Furthermore only binary (or bipolar) sequences are considered.

In [5] the authors propose the so-called *active Hebbian learning* (AHL) method, which introduces the idea of the sequence of activation. The expert specifies the sequence in which the concepts are activated. The process starts from a concept which activates concepts linked to it, and thus the activation propagates until all the concepts have been activated at which point the simulation cycle stops and a new one starts. A distinct form of the Hebb rule is used to provide learning.

Finally, there are several papers (e.g. [6–9]) discussing the so-called *nonlinear Hebbian learning* (NHL). The learning rule used in this approach is the Oja rule [6], although in [7] several extensions are added to it. The procedure is as follows: An initial FCM is constructed by the experts. This is run using equation (1). In addition, at every step the rule is applied using the current activation values. Thus the NHL method does not simply learn the proper weight matrix, but rather it also helps to drive the process of convergence in an online manner [8].

A more traditional application of the NHL rule is proposed in [9]. In this case, NHL is used to make an FCM with a randomly initialized weight matrix learn the cause-effect relationships from historical data. This approach is called *data-driven NHL* (DD-NHL). Ideally, historical data from a real system should be used, but [9] suggests that we can create another FCM with a random weight matrix, and use that to generate the historical data instead. Such FCM is run from a randomly selected initial state for a predefined number of steps, and the resulting concept sequence is used as historical data.

3 Delta Rule and Backpropagation

This section will set forth some of the theory concerning the delta rule and the principle of backpropagation – methods originally designed for learning in ANNs, which we now propose to apply to the regression problem of FCM learning.

3.1 The Delta Rule

The delta rule is probably the best known approach to learning weights of an artificial neuron. It has been designed for supervised learning – that is to say learning from a dataset consisting of pairs of the following form: *(input, desired output)*. That is to say, for every sample in the dataset the input as well as the

corresponding desired output is specified. Thus it is possible to form an error function [10]:

$$E(W) = \sum_p E^p(W) = \frac{1}{2} \sum_p (D^p - O^p)^2, \tag{2}$$

where W denotes the weight matrix, and D^p and O^p denote the desired and the real output for input pattern p.

The error function can then be minimized using gradient descent, which leads to the following learning rule [10]:

$$\Delta_p w_j = \gamma \delta^p x_j, \tag{3}$$

where $\Delta_p w_j$ denotes the prescribed change of weight w_j due to pattern p, γ is the learning rate, and $\delta^p = (D^p - O^p) f'(u^p)$. $f'(p)$ is the first derivative of the squashing function, and $u^p = \sum_{j=1}^{N} w_j x_j^p$ is the inner potential of the neuron (x_j^p is the i-th input of the neuron with pattern p at the input).

3.2 The Backpropagation Principle

The delta rule cannot by itself be used for learning in multi-layer networks, because only the errors of the output neurons can be computed directly – desired outputs of hidden neurons are unspecified.

However, the delta rule can be further generalized to multi-layer networks using the so-called backpropagation principle – in which case the error is propagated back from the output layer to hidden layers. Again, the full derivation of the rule can be found in [10], and we will only state the resulting rule:

$$\delta_h^p = f'(u_h^p) \sum_{o=1}^{N_o} \delta_o^p w_{ho}, \tag{4}$$

where h refers to a neuron of the hidden layer, and o refers to neurons of the output layer. N_o is the number of neurons in the output layer. If there are several hidden layers, the principle can be applied recursively.

The backpropagation principle has further been extended to perform learning in recurrent neural networks (RNNs) – the approach is known as *backpropagation through time* (BPTT). The idea is that an RNN can be unwrapped in time into a feedforward ANN, and then trained using backpropagation. (For additional details and precise mathematical and algorithmic formulations the reader may refer to [11].)

4 The Proposed Approach

Let us now briefly discuss how we propose to apply the above-mentioned principles to solve the *regression problem* of FCM learning.

4.1 One-step Delta Rule

First of all, we can directly use the delta rule as given above. The FCM can be considered as a single-layer network, and thus in this version we do not need to employ the backpropagation principle. Also, since we will be learning from historical data where values of concepts are provided for every time step, we should need to do no BPTT either. We will hereinafter refer to this baseline approach as the *one-step delta rule approach* (OSDR). The results, comparison, and evaluation will follow in a separate section.

4.2 Every-step Delta Rule with Windowed BPTT

A further approach is proposed and studied by the authors: the *every-step delta rule with windowed BPTT* (ESWB) approach. In this case the sequence of historical data is cut into windows of a given size and with a given overlap. For the sake of brevity we will use the notation $w[\text{window size}, \text{overlap size}]$ to describe windowing. The window size is understood to be the number of samples the window contains, and the overlap size is the number of samples which the window shares with the following one. Thus, windowing of $w[5, 4]$ refers to windowing with window size of 5 samples and overlap of 4 samples.

In the ESWB approach we take the first sample and use that as the input of the FCM. The FCM is then run for $window\ size - 1$ steps. Concept values from the last step are compared to the last sample in the window. Delta rule is applied to compute the error and to compute weight updates. The updates are stored in a separate vector – they are not applied to the FCM directly.

BPTT is then applied to propagate the error from the last step back in time. In addition to this backpropagated error, we also compute the error using the corresponding samples from the window. These two errors are added together and used to compute weight updates. Weight updates from all steps are accumulated in the same vector and once all steps have been considered, they are applied to the FCM as a batch.

Afterwards the algorithm moves to another window. In this way we make use of both – the BPTT *and* the error computed for that particular sample.

4.3 One-step Delta Rule with Windowed BPTT

The final approach is proposed mainly for comparison with the one-step delta rule with windowed BPTT (OSWB) approach. This approach is closely related to ESWB except that once the errors are computed using the last sample in the window, *only BPTT* is used to compute weight updates. The other samples from the window are not used to compute error.

5 Simulation Experiments

We conducted several simulation experiments. These were laid out in such manner as to make the results easily comparable to those presented in [9].

Similarly to their work, we first generate an FCM with a random weight matrix. This FCM is run for 20 steps so as to generate the historical data. The data is afterwards used to form *(input, desired output)* pairs for delta rule learning. Learning proceeds on this data for the maximum of 100 epochs (it will also stop if the error goes below 1.10^{-4}). The learning rate is fixed to 0.2.

Testing is first done on the same 20 data steps used in training: these results are used to compute the *in-sample* mean square error (MSE). In addition to that, 10 random initial states are generated and the FCMs are run from each of these for 20 steps. The difference between the outputs of the original and the trained FCM is measured and used to compute the *out-sample* MSE (the procedure taken in [9]).

5.1 Simulation Setup

Unless said otherwise, for every single setting the learning was tested with FCMs of several sizes – with 5 concepts, 10 concepts, and 20 concepts – and with the connection density of 20% and 40% (again the procedure from [9]). The whole process was repeated for 20 independent runs each time. The results were averaged across the runs.

For every configuration the mean square error (MSE) is reported, and also the MSE_{attr}, which specifies how precisely the stable state (the attractor) of the learning FCM corresponds to that of the original FCM – again by giving the mean square error for that. It should be noted, that we compare the values of all concepts – not just of several randomly selected concepts as done in [9]. Also, we did not do restarts in cases when the algorithm did not converge – the algorithm converged to an acceptable solution every time.

5.2 One-step Delta Rule

The results achieved using one-step delta rule (OSDR) follow in Table 1.

Table 1. Errors using the one-step delta rule.

		IN-SAMPLE		OUT-SAMPLE	
Size	**Density**	**MSE**	**MSE_{attr}**	**MSE**	**MSE_{attr}**
5	20%	9.96 **E-5**	3.88 **E-6**	2.51 **E-4**	3.88 **E-6**
	40%	2.14 **E-4**	8.69 **E-6**	3.77 **E-4**	8.69 **E-6**
10	20%	2.38 **E-4**	7.23 **E-6**	1.09 **E-3**	7.23 **E-6**
	40%	3.77 **E-4**	1.89 **E-5**	1.23 **E-3**	1.89 **E-5**
20	20%	4.13 **E-4**	2.68 **E-5**	2.89 **E-3**	2.68 **E-5**
	40%	6.79 **E-4**	2.15 **E-4**	2.22 **E-3**	2.15 **E-4**

We also include results achieved DD-NHL [9] for comparison (Table 2). However, it is difficult to give adequate interpretation to some of the results reported there. 100% accuracy is reported, by which it is understood that none of the target concepts differs from its desired value by more than 0.1 once the stable state

is reached. However, the policy used by the authors is to restart the algorithm from a new randomly-generated initial weight matrix unless it achieves the accuracy of 100% after a predefined number of iterations. Thus, learning is bound to either achieve 100% accuracy at some point, or else to go on indefinitely. It would be of interest to learn how many restarts were required under any given scenario.

Table 2. Errors using DD-NHL.

		IN-SAMPLE		OUT-SAMPLE	
Size	**Density**	**MSE**	**$\mathbf{MSE_{attr}}$**	**MSE**	**$\mathbf{MSE_{attr}}$**
5	20%	0.129	**?**	0.129	**?**
	40%	0.129	**?**	0.129	**?**
10	20%	0.176	**?**	0.175	**?**
	40%	0.180	**?**	0.180	**?**
20	20%	0.180	**?**	0.180	**?**
	40%	0.207	**?**	0.207	**?**

In any case, the comparison shows that the results achieved using OSDR are more precise than those achieved using DD-NHL – and that by several orders of magnitude.

We may also note that we have experimented with several levels of connection density for the initial weight matrix of the learning FCM, but this did not appear to make any considerable difference.

5.3 Every-step Delta Rule with Windowed BPTT

The next experiment was carried out using the ESWB approach. Windowing of $w[5, 4]$ was used to cut the signal up. The results are presented in Table 3.

When we compare the results with those achieved using one-step delta rule (OSDR; Table 1), we must conclude that the results achieved using ESWB seem better – except those for the out-sample MSE, which means that generalization has regressed a little.

Table 3. Errors using ESWB with $w[5, 4]$.

		IN-SAMPLE		OUT-SAMPLE	
Size	**Density**	**MSE**	**$\mathbf{MSE_{attr}}$**	**MSE**	**$\mathbf{MSE_{attr}}$**
5	20%	9.08 **E-5**	1.45 **E-7**	2.53 **E-4**	1.45 **E-7**
	40%	1.88 **E-4**	2.50 **E-7**	3.86 **E-4**	2.50 **E-7**
10	20%	2.77 **E-4**	2.42 **E-7**	1.01 **E-3**	2.42 **E-7**
	40%	3.16 **E-4**	1.01 **E-6**	1.15 **E-3**	1.01 **E-6**
20	20%	3.38 **E-4**	1.60 **E-6**	2.86 **E-3**	1.60 **E-6**
	40%	5.72 **E-4**	3.05 **E-5**	1.98 **E-3**	3.05 **E-5**

What we cannot say with certainty yet is whether the difference results from using ESWB, or whether it is simply the effect of doing more training (OSDR is now done several times for most steps due to the overlapping). To ascertain how much effect should be ascribed to that we present Table 4, which provides results for OSDR with the maximum number of epochs set to 500 instead of 100 (because most samples now form part of 5 windows instead of one).

Table 4. Errors using OSDR; and max. of 500 epochs.

		IN-SAMPLE		OUT-SAMPLE	
Size	**Density**	**MSE**	**$\mathbf{MSE_{attr}}$**	**MSE**	**$\mathbf{MSE_{attr}}$**
5	20%	1.01 **E-4**	1.31 **E-7**	2.06 **E-4**	1.31 **E-7**
	40%	1.42 **E-4**	1.99 **E-7**	3.75 **E-4**	1.99 **E-7**
10	20%	2.28 **E-4**	2.38 **E-7**	9.99 **E-4**	2.38 **E-7**
	40%	3.24 **E-4**	9.50 **E-7**	1.17 **E-3**	9.50 **E-7**
20	20%	3.82 **E-4**	1.56 **E-6**	2.96 **E-3**	1.55 **E-6**
	40%	5.34 **E-4**	2.67 **E-5**	1.96 **E-3**	2.67 **E-5**

The results for in-sample MSE and out-sample are rather mixed in this case – none of the two approaches seems to be decisively the better. Therefore we may conclude that combining the delta rule with BPTT as ESWB approach suggests does not produce any significant improvement.

5.4 One-step Delta Rule with Windowed BPTT

Finally, let us present the results achieved using the OSWB approach. Although the results of the ESWB approach were not very encouraging, the results of OSWB will be of some theoretical interest even if they prove to be only comparable to those of OSDR with 500 epochs (Table 4) – this will indicate BPTT can efficiently be applied to FCMs, and it even to a certain extent able to supply for computing the actual difference between the desired and the real output at some steps. This property may be useful when data for some of the concepts is not available for all steps, or is not available at all.

The results follow in Table 5. Windowing of $w[5, 4]$ was applied.

Table 5. Errors using OSWB with $w[5, 4]$.

		IN-SAMPLE		OUT-SAMPLE	
Size	**Density**	**MSE**	**$\mathbf{MSE_{attr}}$**	**MSE**	**$\mathbf{MSE_{attr}}$**
5	20%	1.06 **E-4**	1.07 **E-7**	2.72 **E-4**	1.08 **E-7**
	40%	1.89 **E-4**	2.23 **E-7**	4.12 **E-4**	2.23 **E-7**
10	20%	2.13 **E-4**	1.48 **E-7**	9.41 **E-4**	1.48 **E-7**
	40%	3.17 **E-4**	1.38 **E-6**	1.05 **E-3**	1.38 **E-6**
20	20%	3.77 **E-4**	1.79 **E-6**	2.95 **E-3**	1.79 **E-6**
	40%	5.35 **E-4**	2.67 **E-5**	1.96 **E-3**	2.67 **E-5**

We conclude that the results are indeed comparable to those achieved using the 500 epoch OSDR, and using ESWB. In fact, in some cases OSWB even achieves better results than ESWB.

6 Further Work

Several lines of future investigation may be suggested. There is little doubt that learning can be made faster and more precise yet by using some of the more advanced learning methods based on the backpropagation principle, such as Quickprop, Rprop, or by the Levenberg-Marquardt algorithm. To ascertain how much effect such methods will have on learning FCMs may form part of future work.

Also, generalization could be improved by using historical data starting from several initial states instead of just one sequence of data. It is obvious that one sequence may not contain all the data required to learn the corresponding matrix accurately. In cases when more data is available, generalization may be improved considerably.

It should also be noted that the backpropagation algorithm could be used to learn even in cases where the activation values of some of the concepts remain unknown. Simulation results achieved using the OSWB method indicate that backpropagation and BPTT in particular can be used effectively in FCM learning. On the other hand, however, if several concepts are left unspecified, the learning algorithm will not be able to discriminate between them as it has no innate understanding of their meaning whatsoever. Therefore this issue will need some further investigation.

It also remains to be shown how well the learning method will perform when some of the weights are forced to remain fixed to their initial values.

7 Conclusion

The paper has presented an overview of the state-of-the-art methods for learning weights of a fuzzy cognitive map. Special emphasis has been put on methods based on Hebbian learning, which has originally been designed for artificial neural networks.

Inspired by the success of these approaches, we have proposed and presented a new method based on the delta rule, and the backpropagation principle, which has also originally been designed for neural networks. We have given a detailed description of our approach, and of its several varieties. These have been discussed, tested by simulation experiments, and compared with data-driven nonlinear Hebbian learning.

It has been shown that the results achieved using the proposed method surpass the accuracy of nonlinear Hebbian learning by several orders of magnitude.

All the varieties of the approach have been tested in turn. The results seem to indicate that the principle of backpropagation, and especially that of backpropagation through time can be used effectively in FCM learning. This should

allow us to train the FCM even in cases, where the activation values of certain concept are not know, or even in cases where values for certain time steps are missing. However, these ideas need further investigation.

In addition to this, several other potential lines of future research and development have been indicated.

References

1. Groumpos, P.P.: Fuzzy Cognitive Maps: Basic Theories and Their Application to Complex Systems. In: Glykas, M., ed. Fuzzy Cognitive Maps. Berlin: Springer-Verlag, pp. 1–22. (2010)
2. Dickerson, J.A. Kosko, B.: Virtual Worlds as Fuzzy Cognitive Maps. Virtual Reality Annual International Symposium, pp. 471–477. (1993)
3. Stylios, C.D., Christova, N., Groumpos, P.P.: A Hierarchical Modeling Technique of Industrial Plants using Multimodel Approach. Proceeding of 10th IEEE Mediterranean Conference on Control and Automation, Lisbon, Portugal. (2002)
4. Huerga, A.V.: A Balanced Differential Learning Algorithm in Fuzzy Cognitive Maps. Proceedings of the Sixteenth International Workshop on Qualitative Reasoning, pp. 10–12. (2002)
5. Papageorgiou, E.I., Stylios, C.D., Groumpos, P.P.: Active Hebbian Learning Algorithm to Train Fuzzy Cognitive Maps. International Journal of Approximate Reasoning, vol. 37, no. 3, pp. 219–249. (2004)
6. Papageorgiou, E.I., Stylios, C.D., Groumpos, P.P.: Fuzzy Cognitive Map Learning based on Nonlinear Hebbian Rule. AI 2003: Advances in Artificial Intelligence, pp. 256–268. (2003)
7. Papageorgiou, E.I., Groumpos, P.P.: A Weight Adaptation Method for Fuzzy Cognitive Map Learning. Soft Computing, vol. 9, no. 11, pp. 846–857. (2005)
8. Papageorgiou, E.I., Stylios, C.D., Groumpos, P.P.: Unsupervised Learning Techniques for Fine-tuning Fuzzy Cognitive Map Causal Links. International Journal of Human-Computer Studies vol. 64, no. 8, pp. 727–743. (2006)
9. Stach, W., Kurgan, L., Pedrycz, W.: Data-driven Nonlinear Hebbian Learning Method for Fuzzy Cognitive Maps. IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2008 (IEEE World Congress on Computational Intelligence), pp. 1975–1981. (2008)
10. Krose, B., Smagt, P.V.D.: An Introduction to Neural Networks. University of Amsterdam. Available from: `http://www.cs.unibo.it/babaoglu/courses/cas/resources/tutorials/Neural_Nets.pdf` [Accessed 10 May 2013]. (1996)
11. Werbos, P.J.: Backpropagation Through Time: What It Does and How to Do It. Proceedings of the IEEE vol. 78, no. 10, pp. 1550–1560. (1990)