Behavioral Analysis of Fuzzy Cognitive Maps

Miklós F. Hatwagner

Abstract Fuzzy Cognitive Maps are widely used in decision making. If historical time series data is available, the preferred way of model creation is based on machine learning. Sometimes such data in unavailable, however, and only experts can define the model. These models unavoidably contain more or less subjective elements, but even if the models are created by machine learning, their thorough investigation is recommended before usage. Sometimes a subtle modification of connection weights or a different λ parameter in the threshold function can result in a model that behaves significatly different. This chapter describes a method to automatically investigate a model and detect if some of its components affect its dynamic hahavior significantly.

1 Introduction

Decision support systems and methods are at the center of researcher's attention for a long time [1], because prudent decision making can be really hard in practice, especially if the effect of many related factors with continuously changing states have to be taken into account, and a wrong intervention may cause serious personal injury or damage to property.

Fuzzy Cognitive Maps (FCMs) [10] can be one of the possible tools of a decision maker [17]. They are bipolar fuzzy graphs that are made of nodes interconnected by directed, weighted edges. The various factors, variables of a complex system can be described by the nodes (also called *concepts*) and the strength of causal relations among them can be expressed by edges. Even if an FCM model can characterize the studied system only qualitatively [18], this method is easy to use and provides a transparent, clear description of it. Modeling capabilities and simulations that reveal

Miklós F. Hatwagner
Széchenyi István University, Győr, Hungary e-mail: miklos.hatwagner@sze.hu

the dynamic behavior of the system made FCM an inevitable opportunity in decision support.

There are two main ways of model creation [16]:

1. an expert, or a group of experts can give the description of the system, or
2. based on a long time-series database a suitable machine learning technique may generate the model with more or less support of experts.

In the first case the expert who makes the model unavoidably includes his/her beliefs and subjectivity in the map. This can be decreased if a group of experts works out the model instead of a sole expert [11, 2], but the result will contain some uncertainty for sure. They can agree relatively quickly and easily on the number of concepts, the existence and sign of relationships, but the magnitude of those relationships are much harder to define, especially if an order of importance or strength exists among them. The number of relationships is a quadratic function of the number of concepts, therefore even if the modelers follow Kosko's advice and they do not include self-loops in the FCM, in case of a small model containing only 10 concepts the number of relationships can be up to 90. It will be really hard to define so many weights properly and worse is that the steepness parameter λ of the threshold function does not have any connection with real objects, but its value may strongly influence the results of simulations (number and values of fixed points (FPs), limit cycles (LSs), chaotic behavior (CB)).

That is why the second method, based on machine learning is preferred in most cases. In those cases it can be applied the concepts are still defined by experts, however, and/or the nature of time series data. Sometimes such data is not available and only the first method remains.

No matter how the model is made, it worth investigating the effect of small modifications of connection weights or steepness parameter on model behavior. The sensitive points of the model can be revealed and experts can discuss about the possibilities on making the model more stable and reliable. Unfortunately it cannot be made by hand, following a trial and error approach. In an FCM the connection weights are specified by real numbers, thus the number of possible modified weights are practically infinite. With a reasonable compromise we can say that the weights to try can be limited to the number of used linguistic variables, eg. 5: -1, -0.5, 0, 0.5 and 1. In the already mentioned model containing only 10 concepts, where even 90 relationships may exist, the number of model versions may up to $5^{90} = 8{,}078\times10^{62}$. Obviously the experts are primarily interested only in the effect of slight modifications, thus they would be satisfied with the check of one less and one greater values (or keeping the current one), but it could still mean $3^{90} = 8{,}728 \times 10^{42}$ model versions. That can be even worse if they want to take the effect of different λ values into account and of course, one simulation is not enough to characterize the behavioral effect of a modification, many simulations (depending on the model size to cover all interesting parts of the search space) are needed and these simulations are rather time consuming on their own. There is a clear need for an automated, consistent method for such investigations, and the work on that started in [5, 6] and further developed in [8, 7]. This task is too big even for a computer: an exhaustive search cannot be performed, but an evolutionary search technique, eg. the Bacterial Evolutionary Algorithm is able to guide the directions of search for a slightly modified model that behaves significantly different.

2 Bacterial Evolutionary Algorithm

The thorough description of Bacterial Evolutionary Algorithm (BEA) can be found eg. in [15, 13, 12, 14], only a short summary is given here to prove the applicability and usefulness of this algorithm. BEA is an evolutionary algorithm and as such able to solve any non-linear, highly modal, high dimensional global optimization problems. Another advantage is that it uses only a few parameters to set by the researcher. Despite its efficiency, BEA has high computational demand. It can be speed up by parallel execution [4] or by a problem specific local search operator [9], converting BEA to Bacterial Memetic Algorithm.

BEA works with a collection of possible solutions, which is also called *population*. The algorithm continuously improves the members of the population, which are also called *bacteria*, because the method imitates the reproduction and evolution of bacteria in the nature. Two operators are used to achieve this goal: *bacterial mutation* is responsible for the exploration of the search space, and *gene transfer* exploits the genetic information stored in bacteria in order to create new, hopefully fitter, more viable bacteria. The algorithm uses these operators alternately until the termination criterion is fulfilled, eg. a specified fitness value is achieved, a predefined number of generations are generated, the convergence slowed down, etc. Usually the best bacterium of the last population is considered result.

Bacterial mutation develops all bacteria at the same time, independently from each other. First, some identical copies of a bacterium are created, they are the *clones* of the original bacterium. Then every alleles of the clones are mutated, changed in random order. If any of these changes lead to a better objective value, the old allele is replaced by the new one in the original bacterium and in all the clones. This operator is elitist by its nature because it keeps the genetic information of the old bacterium if newer bacteria are less fit.

Gene transfer starts with the sort of bacteria according to their fitness values. The bacteria with better fitness go in the *superior* set, the remaining bacteria are locked up in the *inferior* set. Next, the operator selects a bacterium randomly from the superior set, and another one from the inferior set. At least one allele of the better bacterium is transferred to the worse bacterium. Then the modified bacterium have to be evaluated again, and if it has a better objective value, it has a chance to migrate into the superior set and pass on its genetic data to other bacteria during the consecutive gene transfers.

3 Details of the behavior analysis

The goal of behavior analysis is to modify a model only slightly but causing significant impact on its behavior at the same time. This can be an increased number of FPs, the presence of new LCs or CB. If such modifications are possible, they have to be further analysed by experts before the model is applied in decision making.

The search for small but significant modifications are controlled by BEA. In order to decrease the search space and accelerate the process, only 9 of the possible different weight values were tried in the last, most advanced version of the method [7], according to the applied linguistic variables $(-1, -0.75, -0.5, ..., +1)$. In our experience, experts reliably recognize the lack of causal relationships, therefore zero weight connections were left untouched. It also helps to speed up the search process.

The bacteria of BEA encodes a possible, common λ value in the $[0.1, 10]$ interval for all concepts, and all non-zero connection weights in a fixed order.

The behavioral properties of modified models were tested with several simulations started with different inital state vectors. Only 5 activation values were used: 0, 0.25, 0.5, 0.75 and 1. 1000 unique scenarios were generated randomly, and the same set was used to test all modified models for comparability. The program was prepared to automatically detect FPs, and count the number of different initial state vectors that resulted in the same FP. The simulations were definitely stopped after 100 iterations. In most cases, FCM converge very quickly to a FP. In the rare remaining cases the program searched for repeating sequences of state vectors to detect LCs. If it was not successful, the outcome of the simulation was considered a CB.

The state of a model was considered stable only if the state of all concepts were stable, viz. their values changed by at most 0.001 in the last five consecutive time steps. Unfortunately the stable states are not exactly the same in many cases due to rounding errors of floating point arithmetic or other issues. That is why the similar final states were clustered by K-Means algorithm [3] and the cluster centers were considered the common, stable, final states. K-Means needs to know the number of clusters in advance, however. It was automatically estimated by gap statistics [19].

Behavioral analysis is a multiobjective optimization problem. The sometimes conflicting goals were as follows:

1. maximize the number of FPs,
2. maximize the number of LCs,
3. maximize the number of CBs,
4. maximize the similarity of original and modified model versions.

The similarity of models was measured by Eq. 1.

$$d = \sum_{i=1}^{N} \sum_{j=1}^{N} (o_{ij} - m_{ij})^2 \tag{1}$$

where d is the difference of models, N is the number of concepts, o_{ij} is the weight of connection in the original, and m_{ij} is the weight of connection in the modified weight matrix. The fitness values of bacteria were defined in a Pareto-optimal manner. The Pareto-front of all bacteria constituted the members of the set of bacteria with highest fitness value, the Pareto-front of the remaining bacteria were the members of the set of second best bacteria, and so on.

4 Case study

A real life problem of a bank was chosen in [7, 8] to demonstrate the usefulness and operability of the developed program. The concepts are collected and interpreted in Table 1, the causal relationships among concepts are given by Table 2.

Table 1 Concept IDs, names and categories of the investigated model.

Concept ID	Concept name	Category
C1	Clients	Assets
C2	Rules & regulations	
C3	New IT solutions	
C4	Funding	Money
C5	Cost reduction	
C6	Profit/loss	Financials
C7	Investments	
C8	Staff	Human resources
C9	New services	Product and process development
C10	Quality	
C11	Client development	Output measures
C12	Service development	
C13	Productivity	

First, the behavior of the original model was analyzed. The λ steepness parameter of the threshold function used in the inference equation was set to 5. Two FPs were found. 23.1% of the investigated cases resulted in the first, and the remaining 76.9% in the second FP. These FPs are pretty similar, only the values of concepts C6 and C8 differentiates them. C4 is an input concept and it was left out from Table 3.

Second, the behavioral analysis of modified models followed. The model contains 13 concepts, but the density of it is only $\approx$39 %. It still meant a 62 variable optimization problem. The population of BEA consisted of 10 bacteria, 3 clones were created for each bacterium, and also 3 gene transfers were applied in the consecutive iterations. The optimization was stopped after the 10th generation. The connection matrices of the two best modified models (they are the members of the first Pareto-

Table 2 Connection matrix of the FCM model.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
C1	0	0	0.5	0	0	0.5	1	0.5	0	0.5	1	0.5	0
C2	1	0	0.5	1	0	0	1	1	0.5	0	1	1	0
C3	1	0.5	0	0	0	-1	0	-1	1	0	1	1	1
C4	0	0	0	0	0	0	0	0	0	0	0	0	0
C5	0	0	1	-0.5	0	0	0	-1	0	0	0	1	0
C6	0	0	0	0	-0.5	0	0	0	0	0	0	0	0
C7	0.5	0	0.5	1	0	0.5	0	0	0	-0.5	0	0	0
C8	0	0	0	0	0	-0.5	0	0	0	0.5	0	0	-0.5
C9	0	0	0	1	0	0.5	0.5	0.5	0	-0.5	0	0.5	0
C10	0.5	0	0	0	0	0	0.5	0.5	0.5	0	1	0	0
C11	0	0	0.5	0.5	0	0	0	0	0.5	0.5	0	0	1
C12	0	0	0.5	0.5	0	0	1	0	0.5	0	0.5	0	-0.5
C13	0	0	1	0	0	0.5	0	0	0	0	1	0	0

Table 3 Fixed-point attractors of the model.

Concepts	C1-C3, C5, C7, C9-C13	C6	C8
FP#1	1.000	0.150	0.990
FP#2	1.000	0.855	0.922

front, see Fig. 1) are given by Tables 4 and 5. The main properties of these models are collected in Table 6.

Table 4 Connection matrix of the 1st model variant.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
C1	0.0	0.0	0.75	0.0	0.0	0.5	1.0	0.25	0.0	0.25	0.75	0.5	0.0
C2	1.0	0.0	0.5	1.0	0.0	0.0	0.5	0.75	0.75	0.0	0.75	1.0	0.0
C3	0.25	0.0	0.0	0.0	0.0	-1.0	0.0	-0.75	0.5	0.0	0.5	0.75	1.0
C4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C5	0.0	0.0	1.0	-0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0
C6	0.0	0.0	0.0	0.0	-0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C7	0.75	0.0	-0.25	1.0	0.0	0.25	0.0	0.0	0.0	1.0	0.0	0.0	0.0
C8	0.0	0.0	0.0	0.0	0.0	0.25	0.0	0.0	0.0	0.75	0.0	0.0	-0.5
C9	0.0	0.0	0.0	1.0	0.0	0.25	-0.75	0.25	0.0	0.25	0.0	0.75	0.0
C10	-0.75	0.0	0.0	0.0	0.0	0.0	0.25	-0.5	0.25	0.0	-0.25	0.0	0.0
C11	0.0	0.0	0.75	0.5	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	1.0
C12	0.0	0.0	0.5	0.5	0.0	0.0	0.25	0.0	0.25	0.0	-0.75	0.0	-0.5
C13	0.0	0.0	0.75	0.0	0.0	0.75	0.0	0.0	0.0	0.0	0.75	0.0	0.0

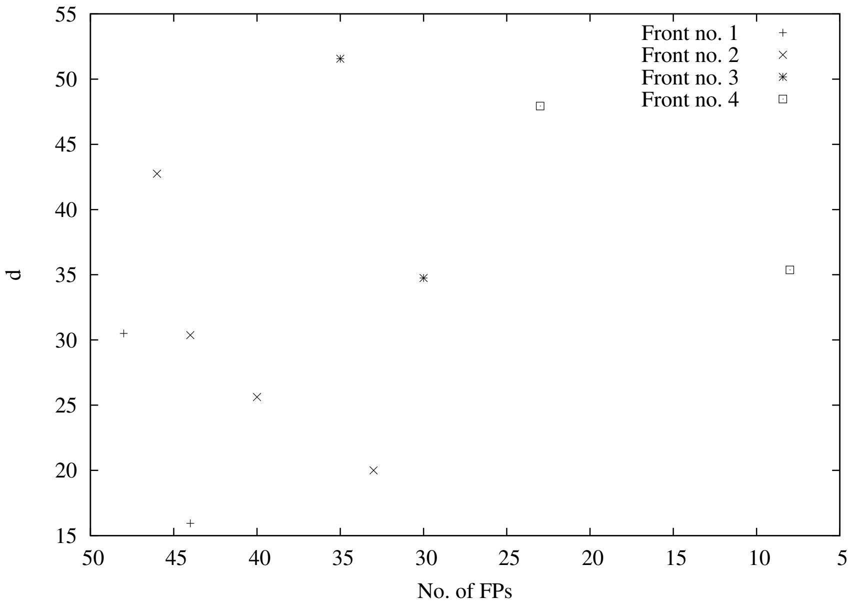

Fig. 1 Bacteria of Pareto-fronts in the last generation.

Table 5 Connection matrix of the 2nd model variant.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
C1	0.0	0.0	0.5	0.0	0.0	0.5	1.0	1.0	0.0	1.0	0.75	0.5	0.0
C2	-0.75	0.0	0.0	1.0	0.0	0.0	0.5	0.0	0.5	0.0	0.0	0.5	0.0
C3	0.25	0.25	0.0	0.0	0.0	-0.75	0.0	-1.0	0.0	0.0	0.5	-0.5	0.0
C4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C5	0.0	0.0	-0.5	-0.5	0.0	0.0	0.0	0.25	0.0	0.0	0.0	0.5	0.0
C6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C7	0.5	0.0	0.0	1.0	0.0	0.75	0.0	0.0	0.0	-0.5	0.0	0.0	0.0
C8	0.0	0.0	0.0	0.0	0.0	-1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.5
C9	0.0	0.0	0.0	1.0	0.0	0.0	0.75	1.0	0.0	-0.75	0.0	0.75	0.0
C10	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	-0.25	0.0	-0.75	0.0	0.0
C11	0.0	0.0	-1.0	0.5	0.0	0.0	0.0	0.0	0.25	1.0	0.0	0.0	-1.0
C12	0.0	0.0	0.0	0.5	0.0	0.0	1.0	0.0	0.5	0.0	-0.25	0.0	0.25
C13	0.0	0.0	1.0	0.0	0.0	0.75	0.0	0.0	0.0	0.0	0.5	0.0	0.0

Table 6 Main properties of the modified model variants.

Property	1st variant	2nd variant
λ value	2.366	2.070
Number of FPs	44	48
Number of LCs	0	0
Number of chaotic behavior	0	0
Difference from orig. model (d)	15.938	30.500

References

1. Jerome R Busemeyer. Dynamic decision making, 1999.
2. Peter P Groumpos. Fuzzy cognitive maps: Basic theories and their application to complex systems. In *Fuzzy cognitive maps*, pages 1–22. Springer, 2010.
3. John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
4. M Hatwagner and A Horvath. Parallel gene transfer operations for the bacterial evolutionary algorithm. *Acta Technica Jaurinensis*, 4(1):89–111, 2011.
5. Miklós F Hatwágner and László T Kóczy. Uncertainty tolerance and behavioral stability analysis of fixed structure fuzzy cognitive maps. *ESCIM 2016*, page 15, 2016.
6. Miklós F Hatwágner, Vesa A Niskanen, and László T Kóczy. Behavioral analysis of fuzzy cognitive map models by simulation. In *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*, pages 1–6. IEEE, 2017.
7. Miklós F Hatwagner, Gyula Vastag, Vesa A Niskanen, and László T Kóczy. Improved behavioral analysis of fuzzy cognitive map models. In *International Conference on Artificial Intelligence and Soft Computing*, pages 630–641. Springer, 2018.
8. Miklós F Hatwágner, Gyula Vastag, Vesa A Niskanen, and László T Kóczy. Banking applications of fcm models. In *Trends in Mathematics and Computational Intelligence*, pages 61–72. Springer, 2019.
9. László T Kóczy, Péter Földesi, and Boldizsár Tüű-Szabó. Enhanced discrete bacterial memetic evolutionary algorithm-an efficacious metaheuristic for the traveling salesman optimization. *Information Sciences*, 460:389–400, 2018.
10. B. Kosko. Fuzzy cognitive maps. *Int. J. Man-Machine Studies*, 24:65–75, 1986.
11. Bart Kosko. Hidden patterns in combined and adaptive knowledge networks. *International Journal of Approximate Reasoning*, 2(4):377–393, 1988.
12. Norberto Eiji Nawa and Takeshi Furuhashi. Bacterial evolutionary algorithm for fuzzy system design. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 3, pages 2424–2429. IEEE, 1998.
13. Norberto Eiji Nawa and Takeshi Furuhashi. A study on the effect of transfer of genes for the bacterial evolutionary algorithm. In *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES'98. 1998 Second International Conference on*, volume 3, pages 585–590. IEEE, 1998.
14. Norberto Eiji Nawa and Takeshi Furuhashi. Fuzzy system parameters discovery by bacterial evolutionary algorithm. *IEEE Transactions on Fuzzy Systems*, 7(5):608–616, 1999.
15. Norberto Eiji Nawa, Tomonori Hashiyama, Takeshi Furuhashi, and Yoshiki Uchikawa. A study on fuzzy rules discovery using pseudo-bacterial genetic algorithm with adaptive operator. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 589–593. IEEE, 1997.
16. Elpiniki I Papageorgiou. Learning algorithms for fuzzy cognitive maps—a review study. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(2):150–163, 2012.

17. Elpiniki I Papageorgiou. *Fuzzy cognitive maps for applied sciences and engineering: from fundamentals to extensions and learning algorithms*, volume 54. Springer Science & Business Media, 2013.
18. J. L. Salmeron. Supporting decision makers with fuzzy cognitive maps. *Research-Technology Management*, 52(3):53–59, 2009.
19. Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.