Time Series Modeling with Fuzzy Cognitive Maps: Simplification Strategies

The Case of a Posteriori Removal of Nodes and Weights

Wladyslaw Homenda[1], Agnieszka Jastrzebska[1], and Witold Pedrycz[2,3]

[1]Faculty of Mathematics and Information Science, Warsaw University of Technology
ul. Koszykowa 75, 00-662 Warsaw, Poland
[2]Systems Research Institute, Polish Academy of Sciences,
ul. Newelska 6, 01-447 Warsaw, Poland
[3]Department of Electrical & Computer Engineering, University of Alberta,
Edmonton T6R 2G7 AB Canada
{homenda,A.Jastrzebska}@mini.pw.edu.pl,wpedrycz@ualberta.ca

Abstract. The article is focused on the issue of complexity of Fuzzy Cognitive Maps designed to model time series. Large Fuzzy Cognitive Maps are impractical to use. Since Fuzzy Cognitive Maps are graph-based models, when we increase the number of nodes, the number of connections grows quadratically. Therefore, we posed a question how to simplify trained FCM without substantial loss in map's quality. We proposed evaluation of nodes' and weights' relevance based on their influence in the map. The article presents the method first on synthetic time series of different complexity, next on several real-world time series. We illustrate how simplification procedure influences MSE. It turned out that with just a small increase of MSE we can remove up to $\frac{1}{3}$ of nodes and up to $\frac{1}{6}$ of weights for real-world time series. For regular data sets, like the synthetic time series, FCM-based models can be simplified even more.

1 Introduction

Fuzzy Cognitive Maps offer compelling modeling environment of phenomena and relations between phenomena.

Since their definition in 1986, Fuzzy Cognitive Maps have been applied to systems' modeling, to classification and prediction tasks, and also to time series modeling. Our recent research efforts are in the latter direction. During studies and experiments in this area we have faced a vital obstacle: complexity and dimensionality of modeled problems. The size of a model based on Fuzzy Cognitive Map is rapidly growing as we add new nodes. This is because each new node in the map requires taking into account all new connections from it to other nodes and from all other nodes to new node. Complexity is an undesirable feature of Fuzzy Cognitive Maps. Large maps are very difficult to interpret and to use. Moreover, learning large maps consumes a lot of computational resources.

Named inconveniences inclined us towards research on simplification strategies for Fuzzy Cognitive Maps trained to model time series. This topic is our contribution to the area of Fuzzy Cognitive Maps applications and to our best knowledge, it has not been addressed in any previous works in this field.

The objectives of this paper are:

- to discuss Fuzzy Cognitive Maps design process for time series modeling and prediction,
- to introduce Fuzzy Cognitive Maps a posteriori simplification strategies,
- to present experimental evaluation of the proposed methods.

The remainder of this article is organized as follows. Section 2 presents brief literature study in the area of time series modeling with Fuzzy Cognitive Maps. Section 3 discusses time series modeling framework. Section 4 introduces Fuzzy Cognitive Map a posteriori simplification strategies. Section 6 concludes the paper and indicates future research directions.

2 Literature Review

B. Kosko introduced Fuzzy Cognitive Maps (FCMs) in 1986 in [3]. Since then FCMs have been intensively researched. Among successful FCM's applications' area is not only but also time series modeling.

In 2008 in [8] W. Stach, L. Kurgan and W. Pedrycz proposed first widely accepted technique for FCMs-based time series modeling. In brief, their methodology can be decomposed into the following steps: 1. Input fuzzification, 2. FCM training, 3. Modeling/prediction, 4. Defuzzifcation. In this method FCM's nodes represent aggregates that describe a pair of input value $a(t)$ and increment $\delta a(t)$.

There were several attempts to time series modeling similar to the methodology described by Stach et al., but all of them focused only on technical aspects of the original procedure. For example, [5, 6] considered the application of neural networks to facilitate Fuzzy Cognitive Map training procedure.

Alternative approaches to FCM-based time series modeling are related to classification. Published works in this stream of research include for example: [1, 2, 4, 7]. In this method FCM's nodes represent attributes of a multivariate time series. Next, the discussed approach is transferable into a typical FCM-based classification problem.

In this article we present FCM a posteriori simplification procedures for FCMs trained to model and predict time series. The origins of FCM design approach are in the methods fathered by Stach et al. There are several shared elements in ours and the cited FCM design procedure, but there also are several important dissimilarities that differentiate the two. Shared is the general idea, both approaches are based on fzzified data, both train FCMs and require deffuzification at the end. The difference is that we represent the time series in an unprocessed manner, while Stach et al. use time series amplitude and increments. Moreover, the original method stops after the FCM is designed. In contrast, we continue modeling process with a posteriori FCM simplification methods.

3 Time Series Modeling with Fuzzy Cognitive Maps

The proposed modeling framework is based on Fuzzy Cognitive Maps trained to model and forecast future values of scalar time series. Therefore, in this section we present a brief discussion on Fuzzy Cognitive Maps training and we shortly introduce the FCM design approach.

FCMs represent the knowledge with weighted directed graphs. Nodes correspond to phenomena. Labeled arcs join the nodes and inform about relationships between the phenomena. Arc from phenomenon A to phenomenon B with a negative weight informs that a decrease in A results in a decrease in B. Arc from phenomenon A to phenomenon B with a positive weight says that an increase in A results in an increase in B. Arc weighting 0 represents lack of relation.

Numerically Fuzzy Cognitive Map is represented with its weights' matrix $\mathbf{W}$, collecting individual weights, denoted as $w_{ij} \in [-1, 1], i, j = 1, \ldots, n$. n is the number of nodes in the FCM. There are two strategies of obtaining weights' matrix. First, expert(s) can determine weights' matrix. Secondly, weights' matrix can be learned from available training data. In our time series modeling method the second strategy is employed: the shape of weights' matrix is determined with customized learning procedure.

FCM exploration is based on its weights' matrix and it occurs as follows:

- FCM receives i-th input activation: $\mathbf{x}_i = [x_{1i}, x_{2i}, \ldots, x_{ni}]$; $\mathbf{x}_{1i}$ is passed to node 1, $\mathbf{x}_{2i}$ is passed to node 2, $\ldots$, $\mathbf{x}_{ni}$ is passed to n-th node,
- FCM processes input activation according to the following formula:

$$\mathbf{y} = f(\mathbf{W} \cdot \mathbf{x}) \tag{1}$$

where $\mathbf{y}_i = [y_{1i}, y_{2i}, \ldots, y_{ni}]$ denotes map responses, $\cdot$ is matrix product, f is a sigmoid transformation function with parameter τ:

$$f(t) = \frac{1}{1 + exp(-\tau t)} \tag{2}$$

In our experiments the value of τ was set to 5 based on literature review.

Map responses $\mathbf{y}_i$ are the states that we expect to observe in phenomena $1, 2, \ldots, n$ modeled by the map after the influence of an i-th activation. In Fuzzy Cognitive Maps activations and map responses are realized with fuzzy sets, so $x_{ik} \in [0, 1]$ and $y_{ik} \in [0, 1]$, $i = 1, \ldots, n$, $k = 1, \ldots, N$. In general, we are equipped with N activations and we expect to obtain N map responses. Then, the notation looks as follows:

- activations: $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$,
- map responses: $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$.

In the case of time series modeling and forecasting the k-th FCM response ($\mathbf{y}_k$) codes the levels of phenomena predicted for the k-th time point. In order to learn weights' matrix $\mathbf{W}$ we conduct a training procedure. In general, it is realized in a following manner:

– initialize weights' matrix $\mathbf{W}$, random initialization was applied,
– iteratively adjust weights' matrix so that the error between FCM responses $\mathbf{y}$ and goals $\mathbf{g}$ is the smallest.

Goals, denoted as $\mathbf{g}$, are real, observed states of modeled phenomena. The learning procedure minimizes the expression:

$$error(\mathbf{y}, \mathbf{g}) \tag{3}$$

where error is a measure of difference between map responses $\mathbf{y}$ and observed states $\mathbf{g}$. In our programs we use Mean Squared Error (MSE):

$$MSE = \frac{1}{n \cdot N} \cdot \sum_{j=1}^{N} \sum_{i=1}^{n} \left(y_{ij} - g_{ij}\right)^2 \tag{4}$$

Training procedure stops when error is smaller than an ϵ threshold or after a fixed number of optimization algorithm's iterations has been exceeded. Weights' matrix adjustment can be performed, for example, with the use of search heuristics, such as Particle Swarm Optimization or Evolutionary Algorithms. In this study Particle Swarm optimization was arbitrarily chosen as a tool for solving an optimization problem. Our tests showed that other search methods give qualitatively similar results, what is sufficient for this study. Therefore, we do not discuss this topics in details.

We continue to use MSE for model quality evaluation throughout the article as well though we are aware that it is sensitive to occasional large error. Due to space limitations we do not elaborate on other error measures. Note, that this paper focuses on our method and we do not compare obtained results with other methods. In order to fully compare two different models in our future works we plan to extend the statistics to Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE).

In the proposed time series modeling method we distinguish following steps:

– data division into train/test. Input time series comprises of $a_0, a_1, \ldots, a_N$, $a_{N+1}, \ldots, a_M$ observations. It is divided into training dataset: $a_0, a_1, \ldots, a_N$ and testing dataset: $a_{N+1}, \ldots, a_M$. In our experiments we split data in proportions 70% for training and 30% for testing,
– model building phase. With training data we learn the weights' matrix $\mathbf{W}$. In order to proceed to this phase we have to transform scalar time series so that the FCM, which is a model on a higher level of abstraction can process it,
– model tuning phase,
– FCM exploration. FCM responses corresponding to first $0, \ldots, N$ activations model time series. FCM responses for $N+1, \ldots, M$ activations are one-step-ahead forecasts for the time series.

Experiments presented in this article follow this convention. As for technical details, we aimed at comparability, so each experiment has been conducted in

the same environment. FCM learning procedure has been written in R and it uses Particle Swarm Optimization "psoptim" function from "pso" package to minimize the Mean Squared Error between map responses and goals. All parameters are default, listed under: [9]. The number of algorithm repetitions was 1000 (also default).

Let us discuss how based on any scalar time series one can design a Fuzzy Cognitive Map on an example of two synthetic time series. The synthetic time series are used for illustrative purposes. First synthetic time series was constructed based on sequence (2,5,8,5,8,2) replicated 500 times, what gave 3000 data points. The second was constructed based on sequence (1,5,7,3,9,3,9,7,5,1,1,7,3,5,9) repeated 200 times, also total of 3000 data points. The original data points were subsequently distorted by adding random number from normal distribution with mean 0 and standard deviation 0.7. First synthetic time series is based on 3 numbers: $\{2, 5, 8\}$, second on 5 numbers: $\{1, 3, 5, 7, 9\}$. The shorter one has a period of 6 values, the longer of 15 values. Hence, the first time series can be considered as easier to learn, the second as harder.

Over the scalar time series we form concepts that aggregate the underlying data set. Formation of concepts is data-driven and aided with Fuzzy C-Means algorithm. We extract arbitrary number of $2, 3, \ldots, u$ cluster centers (k-th cluster center is denoted as v_k). Each cluster becomes a new concept. In this article we set $u = 3$ in each experiment. A posteriori simplification strategy behaves similarly for other architectures (for different u's) so we chose $u = 3$ as it is a good fit for clear illustration. It is worth to notice that other number of concepts still exhibit similar behavior with regard to properties outlined in this study. Of course, studying other properties requires more detailed discussion on valuing this parameter.

Extracted 1-dimensional concepts are described with linguistic variables. Examples of linguistic terms: Small, Moderate, Moderately High, High, etc. As an output of clustering procedure we obtain also membership values for time series data points: $a_0, a_1, \ldots, a_N$ to fuzzy concepts v_1, v_2, v_3.

In the example of the two synthesized time series Fuzzy C-Means extracted the following cluster centers:

- first time series: 1.95, 5.00, 8.06
- second time series: 1.56, 5.04, 8.52

To the concepts above we attach following linguistic interpretation: Small for 1.95 and 1.56, Moderate for 5.00 and 5.04, High for 8.06 and 8.52, abbreviated as S, M and H respectively. Because of properties of the two synthetic time series clustering into 3 concepts fits better to the first example, which is based on 3 numbers - the same as u. Clustering into 3 concepts is qualitatively in disagreement with character of the second synthetic time series based on 5 numbers. Figure 1 illustrates first 150 data points of the first (left plot) and of the second (right plot) time series and extracted 3 concepts with linguistic terms attached. Plots are drawn in 1-dimensional space of time series values. Determined concepts with their linguistic variables are marked with squares.

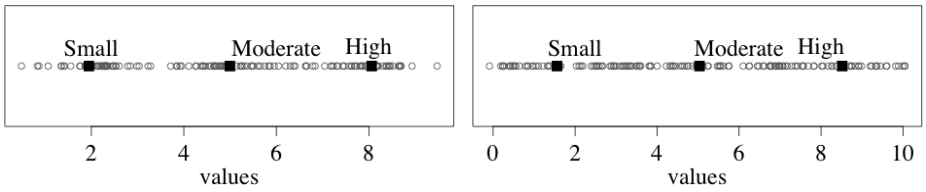

Fig. 1. First 150 data points of two synthetic time series with extracted concepts. Left: (2,5,8,5,8,2)-based time series. Right: (1,5,7,3,9,3,9,7,5,1,1,7,3,5,9)-based time series.

Subsequently, we move from 1-dimensional space of time series values (a_0, a_1, $a_2, \ldots, a_N$) into 3-dimensional space of: current value, past value, and before past value. In such system each observation is characterized with 3 values representing its history. We can represent any time series in this manner as follows: $((a_2, a_1, a_0), (a_3, a_2, a_1), \ldots, (a_{N-2}, a_{N-1}, a_N))$. Interpretation of such triple is the following (current value, past value, before past value).

Next, 1-dimensional concepts are elevated to 3-dimensional concepts. Coordinates of new concepts' centers are determined by applying Cartesian product to the extracted values. Each new concept is characterized with 3 linguistic terms describing current, past and before past values. Examples of 3-dimensional concepts for the first synthetic time series are: (1.95, 1.95, 8.06), (1.95, 1.95, 5.00), (1.95, 5.00, 8.06). By analogy, concept with coordinates (1.95, 5.00, 8.06) is described linguistically as (Small, Moderate, High). 3-dimensional concepts form a sort of lattice points in the system of time series current values, past values and before past values. As a result we get u^3 3-dimensional concepts.

Figure 2 illustrates synthetic time series transformed to the 3-dimensional space of current, past and before past values. 3-dimensional concepts centers are marked with squares.

The synthetic time series form easily separable clouds of points. The first synthetic time series was distributed into 6 clouds, second into 15. Clouds for the second time series overlap more than in the first case. Several concepts proposed for the first time series match its character. Observe, that there is one concept falling to each cloud. There are some potentially redundant concepts as well. In contrast, concepts proposed for the second time series do not fit this well. We have used two distinct examples on purpose, to clearly illustrate the developed method. Broadly speaking the simplification procedure will take advantage from the fact that not all concepts and not all connections fit time series.

In order to relate each time series triple (current, past, before past) with the new fuzzy concepts we have to calculate level of membership for each such triple to the concepts. Membership value is at the same time the level of activation for a given triple. We compute level of activation for $\mathbf{a}_i$-th observation to j-th concept with the standard Fuzzy C-Means objective function:

$$\mathbf{x}_{ij} = \frac{1}{\sum_{k=1}^{n} \left(\frac{||\mathbf{a}_i - \mathbf{v}_j||}{||\ \mathbf{a}_i - \mathbf{v}_k||} \right)^{2/(m-1)}} \tag{5}$$

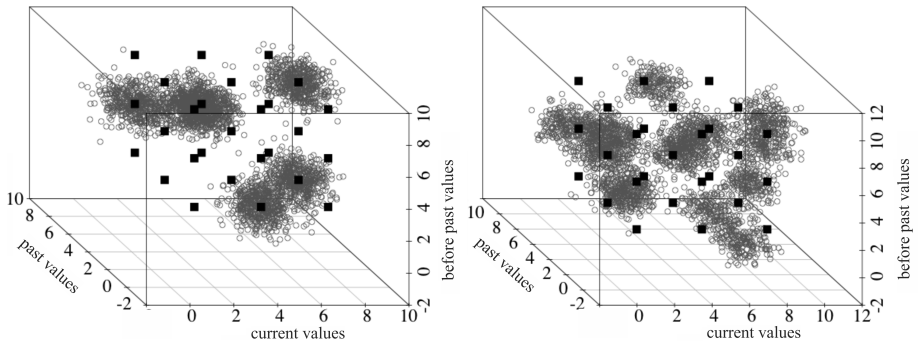

Fig. 2. Synthetic time series in 3-dimensional space of current, past and before past values with extracted 27 concepts (black squares). Left: (2,5,8,5,8,2)-based time series. Right: (1,5,7,3,9,3,9,7,5,1,1,7,3,5,9)-based time series.

m is the fuzzification coefficient ($m > 1$) and $||\cdot||$ is Euclidean norm. In the sequel, we arrange the all activations into $u^3 \times N$ matrix, where membership values are activations and goals are activations shifter forward by 1. The number of concepts is the number of nodes in the map.

To sum up, in this article we consider Fuzzy Cognitive Maps designed with a procedure discussed above. As a result we get a FCM with $n = 27$ nodes and 729 connections (weights).

4 Fuzzy Cognitive Maps a Posteriori Simplification Strategies

The key element of the research investigated in this paper are a posteriori FCM simplification methods. Scheme of the simplification procedure is the following:

- design and train a full FCM-based model.
- simplify the model.

Simplification occurs after we train the full map, this is why we call it a posteriori. In contrast, we can also simplify a FCM before we train it (a priori simplification). We have discussed this alternative method in our previous publications. The both strategies are not exclusive, they are complementary.

Why do we need to investigate FCM simplification strategies? Large models are difficult to understand and to use. Therefore, we agree that at the cost of accuracy we simplify models to make them more usable. In the case of a posteriori simplification we have to acknowledge that the more we simplify, the bigger the error, but for practical reasons we consider this as a reasonable trade-off. In order to simplify the map we can:

- remove weak nodes,
- prune weak weights.

The justification for such simplification strategies is that not all concepts proposed with the Cartesian product are supported with experimental evidence. It was shown in Figure 2, that along with potentially good concepts, the design procedure proposes also potentially redundant concepts. The proportions of useful/unfit concepts depends on the time series. Complexity of modeled data makes FCM design a challenging task. Therefore, the design procedure that produces an overflow of concepts can be tuned by simplification procedures.

Experiments presented in this paper are for time series models and predictions (one-step-ahead forecasts). Model data is called train, because it was used to train the FCM. Part of data for which we made predictions is called test. Proposed method was extensively tested on a series of synthetic time series, where we were able to investigate its properties. Next, we have applied it on several real-world time series. In this paper we present selection of results both on synthetic and real data.

4.1 FCM Simplification by Removing Weak Nodes

In order to determine the quality of nodes after FCM training procedure we evaluate how particular concepts/nodes interact with other concepts/nodes. In other words, we assesses how strongly nodes influence other nodes. Weak nodes are candidates for removal. The following index:

$$\sum_{j=1}^{N} |w_{ji}| \tag{6}$$

can be used to evaluate influence of i-th node on other nodes in the map. The stronger the node, the greater the sum of absolute values of it's outgoing weights.

We pose a question, how removing weak nodes affects model's accuracy. Figure 3 illustrates in detail how this simplification strategy influences MSE on the examples of two synthetic time series.

Figure 3 shows the MSE for model and for forecasts for the synthetic time series for increasing number # rej of rejected nodes. Removing weak nodes makes the MSE grow, but there is an inflection point, up to which the growth is very slow. For FCM architectures, which fit time series well we are able to detect and remove more redundant nodes. This is the case of time series based on 3 numbers divided into 3 concepts (top plot Figure 3), where we have reduced FCM size from 27 to 9 at relatively low cost. In this case for FCM with $n = 27$ MSE on train is 0.001. After simplification procedure by nodes removal for FCM with $n = 8$ we get MSE equal 0.009. At the same time reducing FCM dimensionality from 27 to 9 caused also elimination of weights from 729 down to 81. One can observe that such substantial simplification came at reasonable cost.

For time series models that do not fit this well the underlying data the improvement ranks are smaller, but still impressive. This is the case of the second synthetic time series, bottom plot Figure 3. In comparison, we were able to decrease FCM dimensionality from $n = 27$ with MSE equal 0.001 to $n = 11$ with MSE equal 0.009.

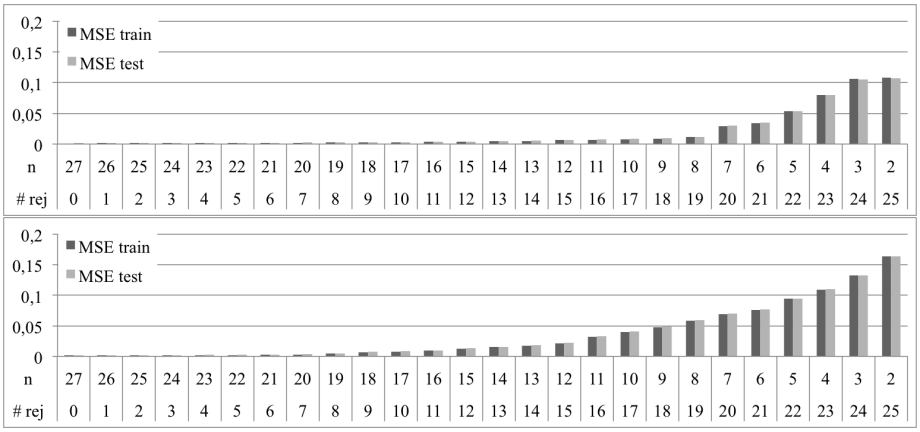

Fig. 3. MSE on time series model (train) and predictions (test) after a posteriori FCM simplification by removing #rej weak nodes. Top: (2,5,8,5,8,2)-based time series. Bottom: (1,5,7,3,9,3,9,7,5,1,1,7,3,5,9)-based time series.

4.2 FCM Simplification by Pruning Weights

Second approach to FCM simplification is based on evaluation of significance of weights in the trained FCM. The weaker is the absolute value of given weight, the less influence it has on the model. Hence, we can simplify the map by pruning such relations. A criterion here is the absolute strength of the weight. We set a threshold *thresh*, below which we consider the weight as insignificant and set its value to 0 indicating no relation.

Figure 4 shows how synthetic time series react to this simplification strategy. It illustrates change in MSE for increasing value of weights rejection threshold *thresh*. When we prune weights weaker than 0.5 the increase in MSE is not significant. Conclusion is as before. For well-fitted models we are able to reduce dimensionality more.

To conclude, the proposed technique of a posteriori FCM simplification allows reducing the complexity of the model at a reasonable cost in the form of the increase of the MSE. The better is FCM initial design, the more we can simplify it, but even not fortunate FCM designs have high capacity for simplification.

Figure 5 completes this topic it illustrates how many weights, measured as % share, we have removed for the thresholds *thresh* of 0, 0.1, ..., 0.9.

5 Experiments on Real-world Time Series

In this section we present a series of experiments on 6 real-world time series. Modeling and prediction quality is assessed with MSE on train (model) and test (prediction) data. Time series illustrated in this section were downloaded from [10, 11]. The selection of time series was based on their character. We tested

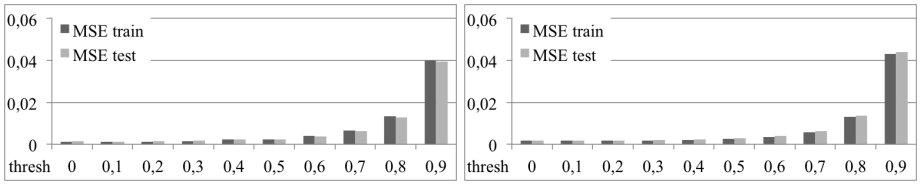

Fig. 4. The influence of weak weights removal from full $n = 27$ FCM on the MSE for thresholds of 0, 0.1, ..., 0.9. Left: (2,5,8,5,8,2)-based time series. Right: (1,5,7,3,9,3,9,7,5,1,1,7,3,5,9)-based time series.

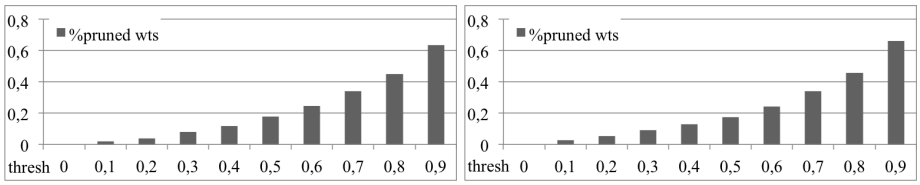

Fig. 5. Percentage of removed weights from full $n = 27$ FCM for thresholds of 0, 0.1, ..., 0.9. Left: (2,5,8,5,8,2)-based time series. Right: (1,5,7,3,9,3,9,7,5,1,1,7,3,5,9)-based time series.

different data sets, with various characteristics (seasonality, trends, length, etc.) Due to space limitations we do not elaborate more on selected time series.

Table 1 illustrates change in MSE for the 6 real-world time series models and predictions for a posteriori simplification procedure of weak nodes removal. Names of time series are the same as in the source repositories. Proposed procedure maintains the same properties for the synthetic and real data sets. We can remove over $\frac{1}{3}$ nodes with very little negative impact on the MSE.

The same observation concerns one-step-ahead predictions for the real-world time series, see lower part of Table 1. The more we simplify the map, the higher the error, but the increase has an inflection point. We can significantly improve time series model without a substantial decrease in its quality.

Table 2 illustrates the influence of a posteriori simplification by weights' pruning on MSE. In the experiment we have successively removed weights that were weaker than threshold $thresh = 0.2, 0.4, 0.6$. Table 2 illustrates the results for the 6 real-world time series. It was confirmed that weak weights can be removed without significant impact on map's quality. For real-world time series the simplification by removing weak weights does not allow to remove as many arcs as for very regular, synthetic data without a big increase in MSE. Nevertheless, maps can be simplified to some extent in this manner as well. We can drop around $\frac{1}{6}$ of weights without a substantial loss in FCM quality.

Table 1. 100·MSE on train (model) and test (one-step-ahead predictions) after removing a posteriori #rej weakest nodes.

#rej	0	2	4	6	8	10	14	18	22
series	train								
Bicup2006	0.07	0.08	0.09	0.10	0.12	0.15	0.36	0.84	2.91
DailyIBM	0.01	0.02	0.02	0.02	0.05	0.08	0.43	0.85	2.86
Kobe	0.11	0.12	0.14	0.17	0.28	0.43	0.83	2.50	5.75
Nv515	0.26	0.29	0.35	0.44	0.71	0.92	1.75	4.87	9.28
Sunspots	0.09	0.10	0.11	0.13	0.15	0.20	0.78	2.35	9.27
Wave2	0.03	0.04	0.06	0.09	0.12	0.16	0.46	1.26	4.31
series	test								
Bicup2006	0.10	0.11	0.12	0.14	0.17	0.20	0.37	0.99	3.28
DailyIBM	0.01	0.02	0.02	0.03	0.07	0.12	0.58	1.22	4.04
Kobe	0.12	0.14	0.16	0.20	0.33	0.47	0.95	2.80	7.28
Nv515	0.25	0.28	0.34	0.45	0.73	0.96	1.72	4.75	9.22
Sunspots	0.11	0.12	0.13	0.15	0.18	0.23	0.98	2.56	9.97
Wave2	0.03	0.04	0.06	0.08	0.12	0.16	0.43	1.17	4.02

Table 2. 100·MSE for model (train) and predictions (test) and the number of pruned edges (#pr) for different weight's thresholds.

threshold	0.2			0.4			0.6		
series	train	test	#pr	train	test	#pr	train	test	#pr
Bicup2006	0.08	0.11	36	0.08	0.11	78	0.31	0.35	149
DailyIBM	0.01	0.01	37	0.02	0.03	87	0.35	0.22	164
Kobe	0.11	0.12	28	0.15	0.14	72	0.19	0.20	167
Nv5150	0.26	0.25	52	0.29	0.27	111	0.52	0.48	217
Sunspots 0	0.09	0.11	38	0.10	0.12	81	0.26	0.26	156
Wave2	0.03	0.03	40	0.03	0.04	81	0.19	0.22	138

6 Conclusion

In this article we have investigated the issue of complexity and proposed simplification strategies in the context of the design method for time series modeling and prediction with Fuzzy Cognitive Maps.

Fuzzy Cognitive Map design procedure for time series modeling and prediction by assumption extracts an overflow of concepts. Having this in mind, we have proposed appropriate a posteriori FCM simplification strategies. Discussed methods evaluate nodes and weights by their influence in the map. Weak

nodes and weights are candidates for removal. We have extensively tested this approach. First, on very uniform and regular synthetic time series. Next, on several real-world time series. Experiments show that we can significantly simplify models with only a slight increase of Mean Squared Error.

In future research we will continue the research on time series modeling and prediction with Fuzzy Cognitive Maps. We plan to focus on interpretation of trained Fuzzy Cognitive Maps.

Acknowledgment

The research is partially supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing. Project financed from The European Union within the Innovative Economy Operational Programme (2007-2013) and European Regional Development Fund.

The research is partially supported by the National Science Center, grant No 2011/01/B/ST6/06478.

References

1. W. Froelich, E.I. Papageorgiou, Extended Evolutionary Learning of Fuzzy Cognitive Maps for the Prediction of Multivariate Time-Series, in: Fuzzy Cognitive Maps for Applied Sciences and Engineering, 2014, pp. 121-131.
2. W. Froelich W., E.I. Papageorgiou, M. Samarinasc, K. Skriapasc, Application of evolutionary fuzzy cognitive maps to the long-term prediction of prostate cancer, in: Applied Soft Computing 12, 2012, pp. 3810-3817.
3. B. Kosko, Fuzzy cognitive maps. in: International Journal of Man-Machine Studies 7, 1986, pp. 65-75.
4. W. Lu, J. Yang, X. Liu, The Linguistic Forecasting of Time Series based on Fuzzy Cognitive Maps, in: Proc. of IFSA/NAFIPS, 2013, pp. 649 - 654.
5. H. Song, C.Y. Miao, Z.Q. Shen, W. Roel, D.H. Maja, C. Francky, Design of fuzzy cognitive maps using neural networks for predicting chaotic time series, in: Neural Networks, Vol. 23, Issue 10, 2010, pp. 12641275.
6. H. Song, C. Miao, W. Roel, Z. Shen, Implementation of Fuzzy Cognitive Maps Based on Fuzzy Neural Network and Application in Prediction of Time Series, in: IEEE Transactions on Fuzzy Systems, Vol. 18, No. 2, 2010, pp. 233-250.
7. H.J. Song, C.Y. Miao, R. Wuyts, Z.Q. Shen, M. DHondt, An Extension to Fuzzy Cognitive Maps for Classification and Prediction, in: IEEE Transactions on Fuzzy Systems, Vol. 19, No. 1, 2011, pp. 116-135.
8. W. Stach, L. Kurgan, W. Pedrycz, Numerical and Linguistic Prediction of Time Series, in: IEEE Transactions on Fuzzy Systems, 16(1), 2008, pp. 61-72.
9. http://cran.r-project.org/web/packages/pso/pso.pdf
10. http://lib.stat.cmu.edu
11. http://robjhyndman.com/tsdldata.