Pseudoinverse learning of Fuzzy Cognitive Maps for multivariate time series forecasting

Pseudoinverse Learning of Fuzzy Cognitive Maps for Multivariate Time Series Forecasting

Frank Vanhoenshoven[a,*], Gonzalo Nápoles[a,b], Wojciech Froelich[c], Jose L. Salmeron[d,e], Koen Vanhoof[a]

[a] *Faculty of Business Economics, Hasselt University Agoralaan, 3590 Diepenbeek, Belgium*
[b] *Department of Cognitive Science & Artificial Intelligence Tilburg University, Warandelaan 2, 5037 AB, The Netherlands*
[c] *Institute of Computer Science, University of Silesia, Poland*
[d] *Data Science Lab, Universidad Pablo de Olavide, Km. 1 Utrera road, 41013 Seville, Spain*
[e] *Universidad Autónoma de Chile, 5 Poniente, 1670 Talca, Chile*

Abstract

Forecasting multivariate time series is an important problem considered in many real-world scenarios. To deal with that problem, several forecasting models have already been proposed, where Fuzzy Cognitive Maps (FCMs) are proved to be a suitable alternative. The key limitation of the existing FCM-based forecasting models is the lack of time-efficient learning algorithms. In this paper, we plug that gap by proposing a new FCM learning algorithm which is based on Moore-Penrose inverse. Moreover, we propose an innovative approach that equips FCM with long-term, multistep prediction capabilities. A huge advantage of our method is the lack of parameters which in the case of competitive approaches require laborious adjustment or tuning. The other added value of our method is the reduction of the processing time required to train FCM. The performed experiments revealed that FCM trained using our method outperforms the best FCM-based forecasting model reported in the literature.

Keywords: fuzzy cognitive maps, learning, time series, forecasting

1. Introduction

Forecasting time series is a well-known problem addressed for many years by numerous researchers. A specific version of this problem is the forecasting of multivariate time series. In this case, instead of forecasting each time series individually, a vector of their values is subject to forecasting. It is expected that

*Corresponding author
Email addresses: frank.vanhoenshoven@uhasselt.be (Frank Vanhoenshoven), gonzalo.napoles@uhasselt.be (Gonzalo Nápoles), wojciech.froelich@us.edu.pl (Wojciech Froelich), salmeron@acm.org (Jose L. Salmeron)

the constituent individual time series are mutually dependent which should, at least theoretically, help to achieve better forecasting accuracy. On the other hand, due to its increased complexity, the forecasting of multivariate time series is substantially less frequently addressed in the literature when compared with the univariate forecasting scenarios.

Several forecasting models have been proposed to tackle that problem. Most popular are vector autoregression (VAR) [1] and vector autoregression moving-average (VARMA) [2]. For further information about the state-of-the-art methods for multivariate time series forecasting, the reader can refer to [3, 4] which comprise a comprehensive overview of this topic.

A promising and very competitive forecasting model is based on Fuzzy Cognitive Maps (FCMs) [5]. Roughly speaking, an FCM is a signed weighted digraph where the nodes represent neural processing concepts while weighted arcs reflect positive or negative dependencies between them. The direction of an arc distinguishes cause concept from its effect concept. FCMs are nonlinear models because the cumulative impact of causal concepts on the effect concept is transformed by a nonlinear function [6, 7].

FCMs were applied for the forecasting of time series with considerable success. When it comes to the construction of FCM, several approaches are reported in the literature. For example, nodes can be designed to represent information granules [8, 9, 10, 11] that will subsequently be used to perform the forecasting process. Another approach relies on mapping each node to a separate variable from the multivariate time series, so it equips each FCM iteration with a well-defining meaning and interpretation [12, 13, 14].

The learning algorithms used to train FCMs can be summarized as follows. On one hand, we have to do with the fairly effective population-based algorithms [14, 13, 15] that require however much time for training the model and tuning the parameters. On the other hand, there are time-efficient adaptive (Hebbian-based) algorithms that are however ineffective in terms of forecasting accuracy [16].

In this paper, we overcome the above limitations. In a nutshell, our proposal comprises two main theoretical contributions.

1. Firstly, we propose a deterministic learning algorithm that uses the Moore-Penrose inverse to compute the FCM's weights. In our proposal, the FCM weights are allowed to change from one time-step to the following one, which helps to describe the evolution of the modeled system.
2. Secondly, we propose a procedure enabling multistep-ahead forecasts. Our solution helps FCM to converge to an equilibrium point and thus prevents to produce fluctuations in the forecasts.

There are many benefits our approach brings. The most important one is the improved accuracy of the forecasts produced by the FCM trained with the use of our method. The results of the experiments we performed provide solid evidence for the superiority of our method. The high effectiveness of FCM training algorithms reserved until now for the time consuming population-based

algorithms has been ultimately outperformed. Our training algorithm is not only highly efficient in terms of forecasting accuracy but also in terms of the processing time required to achieve those accuracies. The other huge benefit demonstrated by the proposed approach is the lack of laborious adjustment of parameters which is characteristic for all competitive approaches. Especially in the case of population-based algorithms that from their nature consume much time, the tuning of parameters is a huge obstacle faced up by the researchers for years. In our research, we finally overcome that downside.

The rest of this paper is organized as follows. In Section 2 we provide a literature review of the FCM-based forecasting models. Section 3 specifies the addressed time series forecasting problem and provides theoretical preliminaries on fuzzy cognitive maps. Section 4 is devoted to the theoretical contributions of our research. In Section 5 we evaluate the effectiveness of our proposal by using 41 time series datasets. Section 6 provides the final remarks.

2. FCM-based forecasting models - related work

It is possible to distinguish diverse approaches to use FCMs for time series forecasting. Unfortunately, as already mentioned in the introduction, all those methods suffer from the time-consuming population-based training algorithms or alternatively, from the time-effective but poor in terms of the forecasting accuracy, Hebian-like training. Let us present here a brief introduction to the state-of-the-art of the FCM-based methodology.

A popular approach when designing FCM-based forecasting models is to map its concepts to time series. Concepts can play the role of regressors, similarly as in auto-regressive forecasting models [17]. In this case, the accumulated impact of the cause concepts affect the target concept. When applying standard FCMs for time series forecasting, most papers [12, 13, 14] assume that the weights of FCM are adjusted during the training phase and do not change over time when used for forecasting. The above feature is a shortcoming of the approach limiting the use of FCM merely as the first-order forecasting model.

FCMs in which the concepts are the information granules rather than plain variables, turned out to be a very competitive model applied to forecast univariate [18, 19, 20] and multivariate [8, 9, 10, 11] time series. Several contributions have been reported. Song et al. [21] [22] designed an FCM-based network using neural networks and fuzzy sets to predict a chaotic time series. Salmeron et al. [23] proposed an FCM-TOPSIS hybrid methodological support to scenario-based forecasting. TOPSIS [24] stands for Technique for Order of Preference by Similarity to Ideal Solution. More recently, in [25] the authors proposed a procedure to dynamically optimize the length of the learning period and the selection of the transformation function together with its parameters, thus leading to high prediction rates. Although the above approach overcomes to a certain extent the issues related to the first-order FCM model, it still does not solves the problem of the time-consuming training phase. Moreover, the use of a moving window even elevates that problem.

Other examples related to forecasting include the FCM-based prostate cancer prediction model [8]; the introduction of a multistep approach to learning evolutionary FCMs for the prediction of pulmonary infection [10]; or the model devoted to forecasting industrial drying processes [26]. These researches evidence the broad potential behind FCM-based forecasting models.

As previously mentioned, the existing algorithms applied to learn FCMs belong to two main groups, population-based and Hebbian-based methods.

It has been experimentally proved that the population-based algorithms outperform the adaptive ones in terms of forecasting accuracy [27]. Hence, mostly population-based algorithms have been applied for learning FCMs as the forecasting model. These methods involve: Genetic Algorithm (GA) [28, 29], Particle Swarm Optimization (PSO) [18, 30, 31], Memetic Algorithms [26], Artificial Bee Colony (ABC) [32], Modified Asexual Reproduction Optimisation [7] and Differential Evolution (DE) [33].

The advantage of using population-based algorithms is that the learned weights reflect the characteristics of the entire time series used for training. Issues may arise when the training set contains many variables or has a large number of observations. In that case, global optimization made by population-based learning algorithms takes much time. The other issue with the population-based algorithms used for learning FCMs is the high sensitivity to parameters, e.g., population size, the probability of mutation and other [25]. It means in practice, that for every time series to be forecasted the process of tuning parameters requires multiple, even hundreds of learn-and-test trials. In conjunction with a long time needed to run each trial, the optimization of the experimental setup requires considerable effort [25].

Hebbian-based learning algorithms are unsupervised methods based on the Hebbian law [34, 35]. The main idea of these methods is to modify the weights incrementally, which results in forgetting of older weights as they are updated according to the most recent values. To the best of our knowledge, no Hebbian-based method has been used in the context of time series forecasting, perhaps due to their very poor generalization capability.

A survey of FCM learning algorithms can be found in [36].

3. Background

Here we specify the time series forecasting problem that we address in this paper. Later on, we define FCM, the forecasting model we investigate in our research.

3.1. Multivariate time series forecasting

Let $x \in \mathbb{R}$ be a real-valued variable observed over a discrete time scale within the period $t \in \{1, 2, \ldots, T\}$, where $T \in \mathbb{N}$ denotes its length. Thus, a univariate time series is defined as a sequence of observations $\{x^{(t)}\} = \{x^{(1)}, x^{(2)}, \ldots, x^{(T)}\}$. Let us denote by historical time series the one in the period $t \in \{1, 2, \ldots, t_e\}$, where $t_e \leq T$. The goal of forecasting is to predict the next values of the time

series, i.e., $\hat{x}^{(t_e+1)}, \hat{x}^{(t_e+2)}, \ldots, \hat{x}^{(t_e+H)}$, where H is referred to as the *prediction horizon*.

To accomplish the forecasting task, a model $\mathcal{F}$ is required [16]. Assuming single-step forecasting, i.e., for $H = 1$, the model $\mathcal{F}$ is used to calculate $\hat{x}^{(t_e+1)} = \mathcal{F}(x^{(t_e)})$. The problem to be addressed is the construction of $\mathcal{F}$, which is usually not known and has to be discovered using historical records.

For multivariate time series, the vector $S = [S^{(1)}, \ldots, S^{(t)}, \ldots, S^{(T)}]$ comprises a sequence of observations for multiple real-valued variables, such that $S^{(t)} = [x_1^{(t)}, \ldots, x_i^{(t)}, \ldots, x_M^{(t)}]$. The single-step forecasting assumes the form: $\hat{S}^{(t_e+1)} = \mathcal{F}(S^{(t_e)})$, where $\mathcal{F}$ is the forecasting model. In our study, we focus on the FCM-based forecasting model.

3.2. Fuzzy Cognitive Maps

FCM is defined as a 4-tuple $\langle \mathcal{C}, \mathcal{W}, \mathcal{A}, f(\cdot) \rangle$, where $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ is a set of M concepts, each concept being a fuzzy set. $\mathcal{W} : \mathcal{C} \times \mathcal{C} \rightarrow [-1, 1]$ represents the weight matrix. For each pair of concepts (C_i, C_j), the weight $w_{ij} \in [-1, 1]$ defines the causal relationship between the concepts. Visually, w_{ij} is represented as an edge between two nodes in the FCMs network. The value of w_{ij} determines the sign and intensity (magnitude) of the relationship of cause concept C_i on effect concept C_j. The weight w_{ij} should be interpreted as follows:

- $w_{ij} > 0$: If activated, C_i will *excite* C_j. Higher (lower) activation values of C_i in the current iteration will lead to higher (lower) activation values of C_j in the following iteration;
- $w_{ij} < 0$: If activated, C_i will *inhibit* C_j. Higher (lower) activation values of C_i in the current iteration will lead to lower (higher) activation values of C_j in the following iteration.
- $w_{ij} = 0$: C_i will not influence C_j.

The function $\mathcal{A} : \mathcal{C} \times \mathbb{N} \rightarrow A_i^{(t)}$ is used to calculate the activation values of concepts at iteration-step $t \in \{1, 2, \ldots, T\}$. An FCM has a recurrent nature and evolves through different state values in a sequence of iterations. Note that FCM produces a state value at each iteration, thus conferring a recurrent behavior to the reasoning model. The activation rule as proposed by Kosko [5] is depicted in equation (1), with $A^{(t)}$ being the activation vector, $A_i^{(t)}$ denotes the activation value of the concept C_i at the t^{th} iteration,

$$A_i^{(t+1)} = f\left(\sum_{\substack{j=1 \\ i \neq j}}^{M} w_{ji} \cdot A_j^{(t)}\right) \tag{1}$$

where $f : \mathbb{R} \rightarrow I$ is a transfer function [37] that aggregates the impact of multiple causal effects and restricts the activation value into an activation interval, e.g.:

$I = [0, 1]$ or $I = [-1, 1]$. Possible transformation functions include binary, ternary and logistic activators. A sigmoid function (eq:sigmoid) is commonly used as the transfer function.

$$f(x) = \frac{1}{1 + e^{-\lambda \cdot x}} \tag{2}$$

where λ is the parameter determining the gain of the causal impact the function introduces.

4. An innovative FCM-based forecasting model

In this section, we introduce a new approach to forecasting multivariate time series using a novel FCM-based model.

4.1. Dynamic FCM's weights

The model proposed in this paper is based on an FCM equipped with dynamically changing weights. In consequence, the reasoning rule associated with our model assumes the form given by Equation 3:

$$A_i^{(t+1)} = f\left(\sum_{j=1}^{M} w_{ji}^{(t)} A_j^{(t)}\right), \tag{3}$$

where $f(\cdot)$ is the sigmoid transfer function, $t = \{1, 2, \ldots, T\}$ denotes the current iteration, while $A_i^{(t)}$ denotes the activation value for C_i at time t and $w_{ji}^{(t)}$ is the effect of C_j on C_i progressing from iteration t to iteration $t+1$. Finally, M denotes the number of concepts in the FCM model.

4.2. Pseudo-inverse learning of the weight matrix

Let us introduce pseudo-inverse learning of the FCM weight matrix. The peculiarity of this learning method is that it computes a different weight set for each iteration step. In this way, different time-varying pieces of data influence the weights, which change from one iteration to the following.

Aiming at deriving our method from $A_i^{(t+1)} = f\left(\sum_{j=1}^{M} w_{ji}^{(t)} A_j^{(t)}\right)$ (Equation (3)), we formulate a cascade of two-step inference formulas as follows:

$$A_i^{(t+1)} = f\left(\sum_{j=1}^{M}\left(w_{ji}^{(t)} \underbrace{f\Big(\sum_{l=1}^{M} w_{lj}^{(t-1)} A_l^{(t-1)}\Big)}_{A_j^{(t)}}\right)\right) \tag{4}$$

and therefore:

$$A_i^{(t+1)} = \sum_{j=1}^{M}\left(w_{ji}^{(t)} \underbrace{f\Big(\sum_{l=1}^{M} w_{lj}^{(t-1)} A_l^{(t-1)}\Big)}_{A_j^{(t)}}\right) + \xi, \tag{5}$$

where ξ denotes the error caused by suppressing the outer transformation operation. This is formulated in matrix format:

$$\boldsymbol{A}^{(t+1)} = \boldsymbol{W}^{(t)} f\Big(\boldsymbol{W}^{(t-1)} \boldsymbol{A}^{(t-1)}\Big) + \xi, \tag{6}$$

where:

$$\boldsymbol{A}^{(t)} = \begin{bmatrix} A_1^{(t)} & A_2^{(t)} & \dots & A_M^{(t)} \end{bmatrix}^T$$

and

$$\boldsymbol{W}^{(t)} = \begin{bmatrix} w_{11}^{(t)} & w_{12}^{(t)} & \dots & w_{1M}^{(t)} \\ w_{21}^{(t)} & w_{22}^{(t)} & \dots & w_{2M}^{(t)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1}^{(t)} & w_{M2}^{(t)} & \dots & w_{MM}^{(t)} \end{bmatrix}.$$

At this point, we can easily derive the learning rule (7) to compute the target weight matrix $W^{(t)}$, where $(\cdot)^{\ddagger}$ denotes the Moore-Penrose inverse of a given matrix.

$$\boldsymbol{W}^{(t)} \approx \boldsymbol{A}^{(t+1)} \Big(f\big(\boldsymbol{A}^{(t-1)} \boldsymbol{W}^{(t-1)}\big) \Big)^{\ddagger} \tag{7}$$

We adopt the orthogonal projection method to compute the Moore-Penrose pseudoinverse. If the matrix H has linearly independent columns ($H^\top H$ is nonsingular), then $H^{\ddagger} = (H^\top H)^{-1} H^\top$. In contrast, if H has linearly independent rows ($HH^\top$ is nonsingular), then $H^{\ddagger} = H^\top (H^\top H)^{-1}$. The former constitutes a left inverse because of $H^{\ddagger} H = I$, whereas the latter comprises a right inverse because $HH^{\ddagger} = I$.

Equation (7) provides a deterministic approach to compute the weight matrix at the current iteration. Notice however that although the $\boldsymbol{A}^{(t+1)}$ component is unknown in a forecasting scenario, it is available as part of the historical dataset during the training phase.

To complete the derivation of learning, let us assume a time series $\boldsymbol{X}_{(T)\times M}$, where T is the time series length and M denotes the number of variables in the system. Assuming that we take into account the values of $\boldsymbol{X}$ at time t, $X^{(t)}$, to predict the observed values $\boldsymbol{X}^{(t+1)}$, we can define $\widehat{X}^{(t+1)}$ as the predicted values for time step t.

$$\boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & \dots & x_i^{(1)} & \dots & x_M^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1^{(t)} & \ddots & x_i^{(t)} & \ddots & x_M^{(t)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1^{(T-1)} & \dots & x_i^{(T-1)} & \dots & x_M^{(T-1)} \end{bmatrix}$$

We can easily substitute our time series $\boldsymbol{X}$ into Equation (7) by mapping each $\boldsymbol{X}^{(t)}$ to a vector $\boldsymbol{A}^{(t)}$, as follows:

$$\boldsymbol{W}^{(t)} \approx \boldsymbol{X}^{(t+1)} \big(f(\boldsymbol{W}^{(t-1)} \boldsymbol{X}^{(t-1)}) \big)^{\ddagger}. \tag{8}$$

It should be mentioned that unlike most heuristic learning algorithms, the proposed learning formula does not have an iterative nature. This means that we can compute each weight matrix $\boldsymbol{W}^{(t)}$ in a single step. Nevertheless, this learning process must be repeated for each iteration in the time series, so that each iteration is characterized by a specific weight matrix.

4.3. Learning algorithm

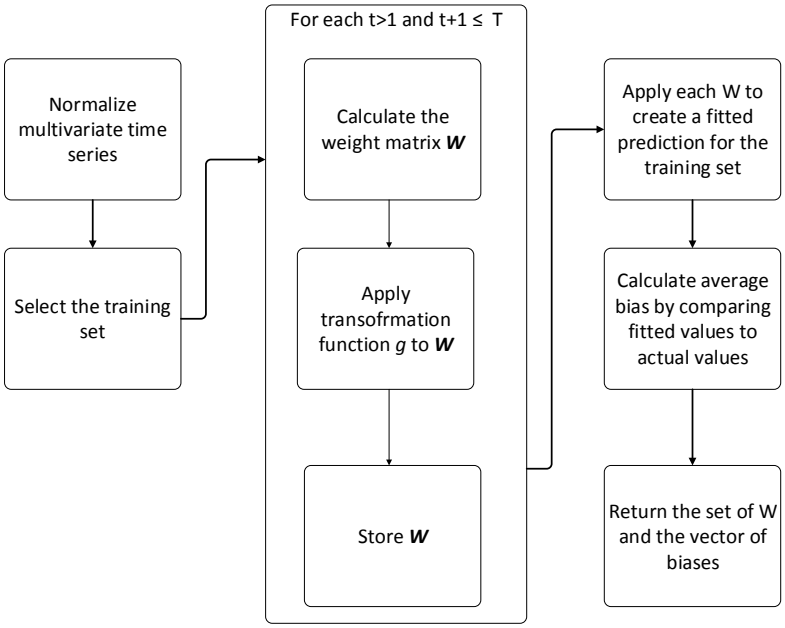

Figure 1: Flow diagram of the high-level method that is followed when learning. More details can be found in Algorithm 1

Figure 1 and Algorithm 1 portray the learning procedure we propose. As can be seen in the high-level overview in figure 1, the learning procedure starts with selecting a training set from a normalized, multivariate time series. During the actual learning phase, the algorithm will iterate over each time step in the dataset and calculate a weight matrix $\boldsymbol{W}$ that describes the transition from step t to step $t-1$. A transformation function can be applied to constrain the individual weights between desired boundaries. Once the set of weight matrices is collected, they will be used to fit the time series. These fitted values are then used to calculate a bias. The learning function will return the set of weight

matrices as well as the bias. A more detailed view of the algorithm is provided in Algorithm 1.

In 1, $\boldsymbol{W}^{(t)}$ represents a weight matrix that is fitted between time step t and $t+1$? whereas $\boldsymbol{X}_{T\times M}$ represent the observed values in a multivariate time series. The vector of observations for each variable at time step t is represented by $\boldsymbol{X}^{(t)}$, whereas $\widehat{\boldsymbol{X}}^{(t)}$ represents the vector of predicted values at time step t.

The main input for the learning procedure is a multivariate time series dataset $\boldsymbol{X}_{T\times M}$, where T denotes its length and M is the number of variables. Other inputs are the transformation function $f(\cdot)$ defined by formula (2) and the weight transformation function $g(\cdot)$, which is defined by the $\tanh(\cdot)$ function. It is worth mentioning that, opposite to the standard FCM, our FCM-MP does not assume that the values of the FCM weights must be within the range $[-1, 1]$.

Algorithm 1 Learning Procedure

1: **function** LEARN WEIGHT MATRICES($\boldsymbol{X}_{T\times M}$, f, g)
2: **for** $t \leftarrow 1, T$ **do**
3: **if** $t = 1$ **then**
4: $\boldsymbol{W}^{(1)} = f\big(\boldsymbol{X}^{(1)}\big)^{\ddagger}\boldsymbol{X}^{(2)}$
5: **else**
6: $\boldsymbol{W}^{(t)} = \boldsymbol{X}^{(t+1)} f\big(\boldsymbol{X}^{(t-1)}\boldsymbol{W}^{(t-1)}\big)^{\ddagger}$
7: **end if**
8: $\boldsymbol{W}^{(t)} = g(\boldsymbol{W}^{(t)})$
9: **end for**
10: **for** $t \leftarrow 1, T$ **do**
11: $\widehat{\boldsymbol{X}}^{(t+1)} = f\big(\boldsymbol{X}^{(t)}\boldsymbol{W}^{(t)}\big)$
12: **end for**
13: $\xi = \frac{\sum_{t=1}^{T-1}\left(\widehat{\boldsymbol{X}}^{(t)} - \boldsymbol{X}^{(t)}\right)}{M(T-1)}$
14: **return** $(\boldsymbol{W}_{T-1\times M^2}, \boldsymbol{\xi}_{M\times 1})$
15: **end function**

Another aspect that deserves attention refers to the simplification of the learning formula depicted in lines 3-7. If we assume that $\boldsymbol{\Psi}^{(t)} = \boldsymbol{X}^{(t)}$ for $t = 1$ and $\boldsymbol{\Psi}^{(t)} = \boldsymbol{W}^{(t-1)}\boldsymbol{X}^{(t-1)}$ for $t > 1$, then Equation (8) can be rewritten as $\boldsymbol{W}^{(t)} \approx \big(f(\boldsymbol{\Psi}^{(t)})\big)^{\ddagger}\boldsymbol{X}^{(t+0)}$, hence being consistent with the notation and the derivation of this pseudoinverse learning.

The dimensions of the returned weight matrix $W_{T-1\times M^2}$ can be explained as follows. Each row represents a transitional step in the training set. For a training set of length T, only $T-1$ transitions can be calculated. Each column represents an individual weight $w_{ij}^{(t)}$. To remodel a full weight matrix $W_{N\times N}$ for each transition t, M^2 individual weights need to be reported.

As the method depicted in Equation (8) discards the effects of a transformation function

$$f(\cdot)$$

in the tuning process, there will be a bias in the results. This bias is estimated by the parameter ξ which is calculated as the mean error, mean$(\widehat{\boldsymbol{X}} - \boldsymbol{X})$, in the training set. Besides the set of weight matrices $W_{T-1\times M^2}$, the trained model will also return $\boldsymbol{\xi}_{M\times 1}$. This allows applying the same correction to the test data during the execution phase. The algorithm returns a weight matrix $W_{1\times M^2}$ for each time step t in the training set $\boldsymbol{X}$.

4.4. From short-term to long-term forecasts

In this section, we propose two techniques for time series forecasting that is based on the same learning mechanism. More explicitly, a model that can be applied for short-term predictions (one-step ahead) is described in section 4.4.1, while subsection 4.4.2 explains a model that can be used for long-term, multiple steps ahead, predictions.

4.4.1. Single-step-ahead forecasts

Performing single-step-ahead predictions is straightforward and requires the reasoning process to be performed for only a single test instance. In this setting, our FCM-MP uses the weight matrix fitted between time steps $T-1$ and T to make the one-step-ahead prediction for $T+1$. Equation (9) displays how to calculate the next predicted value $\widehat{\boldsymbol{X}}^{(T+1)}$ using the reasoning formula in its matrix form.

$$\widehat{\boldsymbol{X}}^{(T+1)} = f\big(\boldsymbol{W}^{(T)}\boldsymbol{\Psi}^{(\boldsymbol{T})}\big), \tag{9}$$

where: $\Psi^{(T)}$ is initialized with the test instance (i.e., the activation vector associated with the instance) and $\boldsymbol{W}^{(T)}$ is the weight matrix fitted to calculate $\boldsymbol{X}^{(T)}$ from $\boldsymbol{X}^{(T-1)}$ during the training phase.

4.4.2. Multistep-ahead forecasts

The continuous (fragmentary) learning proposed in the previous section assumes that the weight matrix $\boldsymbol{W}^{(T)}$ is a better estimate than $\boldsymbol{W}^{(T-1)}$. However, this approach is not feasible for long-term forecasts. If we would incrementally apply our forecasting approach for multistep-ahead predictions — using a predicted set $\widehat{\boldsymbol{Y}}^{(t)}$ to forecast $\widehat{\boldsymbol{Y}}^{(t+1)}$ — the same weight matrix $\boldsymbol{W}$ would be used to perform each next forecast. This may cause the model to converge to a fixed-point attractor, thus suppressing its capability to capture variations in the time series.

When applying our FCM-MP for multistep-ahead forecasting (see Algorithm 2) we attempt to select a weight matrix from the adjusted ones that better fits the current situation and use that matrix to make each next step prediction. Figure 2 displays the blueprint of the long-term forecasting procedure. The process starts with the initial values $\boldsymbol{A}^{(0)} = \boldsymbol{X}^{(0)}$ that are deemed the starting point for the multistep forecast. Each subsequent prediction will be defined as $\boldsymbol{A}^{(t)} = \widehat{\boldsymbol{Y}}^{(t-1)}$. Predictions will be made for H steps, resulting in a long-term forecast output $\boldsymbol{A}_{H\times M}$.

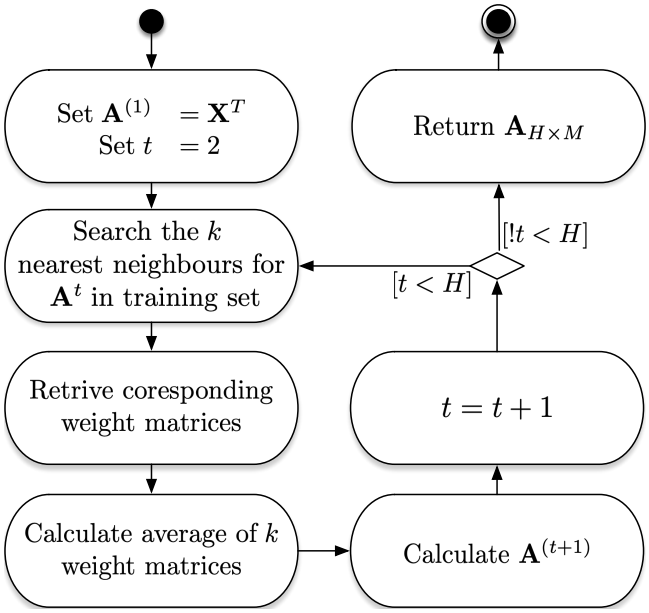

Figure 2: Flow diagram of the proposed forecasting approach.

To predict each value $\widehat{\boldsymbol{Y}}^{(t)}$, our model scan the training set to detect a moment in time that is most similar to $\boldsymbol{Y}^{(t-1)}$. A parameter $w \in \mathbb{N} : w < T$ is defined to control whether we want to compare against a single data point or a moving time windows of size w. This heuristic is similar to applying a k-NN approach to determine the more suitable weight matrix when making each prediction in the long-term context.

The algorithm detects the k most resembling periods from the training history and returns the respective weight sets that have been fitted to calculate the $t + 1$ forecast. The average of these k weight matrices is adopted to make a prediction $\widehat{\boldsymbol{Y}}^{(t+1)}$ from $\widehat{\boldsymbol{Y}}^{(t)}$. This prediction is adjusted by adding the bias ξ, which is determined during the learning stage. The process is repeated until H time steps have been predicted. Algorithm 2 provides a more technical description of the proposed procedure.

To detect the most resembling periods from training history, a distance function is adopted. For $w = 1$, the Euclidean distance is calculated between each multivariate time observation. For $w > 1$, the model calculates a rolling cross-correlation for each variable and then calculates the average cross-correlation across all dimensions. The highest value denotes the most resembling time window. Equation (10) displays the cross-correlation for a dimension n and a time window w, between the recent series X and any historical series Y having the same rolling window,

Algorithm 2 Multistep-ahead forecasts

function PREDICT($\boldsymbol{Y}_{1\times M}$, $\boldsymbol{X}_{T\times M}$, $\boldsymbol{W}_{T\times M^2}$, ξ, h, w, k)
  $\boldsymbol{A}_{h+1\times M} = \boldsymbol{0}_{h+1\times M}$
  $\boldsymbol{A}^{(0)} = \boldsymbol{Y}$
  for $t \leftarrow 1, h$ **do**
    if $w = 1$ **then**
      for $i \leftarrow 1, T$ **do**
        $d_i = ||\boldsymbol{A}^{(t-1)}, \boldsymbol{X}^{(i)}||_2$
      end for
    else
      $d = dtw(\{\boldsymbol{A}^{(t-w)}, \ldots, \boldsymbol{A}^{(t-1)}\}, \boldsymbol{X})$
    end if
    sort(d)
    select ordered indices b of k most similar periods in $\boldsymbol{X}$
    $\widehat{\boldsymbol{W}}^{(t)}_{M\times M} = \frac{\sum_{c=1}^{k} \boldsymbol{W}^{(b)}}{k}$
    $\boldsymbol{A}^{(t)} = f\big(\boldsymbol{A}^{(t-1)}\widehat{\boldsymbol{W}}^{(t)}\big) + \xi$
  end for
  return $\widehat{\boldsymbol{Y}} = \boldsymbol{A}_{h+1\times M}$
end function

$$\rho^{(n)}(X,Y) = \frac{1}{w} \cdot \left(\sum_{i=1}^{w} \frac{(x_i^{(n)} - \bar{x}^{(n)})(y_i^{(n)} - \bar{y}^{(n)})}{\sigma_{x^{(n)}}\sigma_{y^{(n)}}} \right), \tag{10}$$

where X and Y are univariate time series of length w, $\bar{x}^{(n)}$ is the median associated with the specified dimension (variable) while $\sigma_{x^{(n)}}$ is the standard deviation. The calculations are performed for each rolling window w in the training dataset, resulting in $N - w$ different average cross-correlation measures. It should be mentioned that, for long time windows, more elaborate measures to compare time series (e.g., the *Dynamic Time Warping* [38, 39]) would lead to better results. In this paper, however, we adopted a simpler strategy to keep the training time as low as possible.

5. Experiments

In this section, we provide empirical evidence for the high effectiveness of the proposed approach.

5.1. Experimental setup

For the experiments, we used 42 time series datasets taken from the web site: http://faculty.chicagobooth.edu/ruey.tsay/teaching/mtsbk. The repository comprises a wide variety of time series with diverse characteristics. The descriptive summary of all datasets is given in Table 1, where: N denotes the number of variables, T is the number of observations. The minimum, median

and maximum of the absolute values of Pearson correlations are denoted as $min(|\rho|)$, $med(|\rho|)$, $max(|\rho|)$ respectively.

We selected mean squared error (MSE) as a benchmark for the evaluation of the performance of each forecasting model. Formula (11) formalizes this measure, where $Y_i^{(t)}$ denotes the observed value for concept i at time t in the time series, $\widehat{Y}_i^{(t)}$ is the predicted value of concept i for that time step, M represents the number of concepts, or variables, in the multivariate time series, whereas T is the time series length or the number of observations. If the MSE is calculated for the training set, only the training instances are taken into account and thus T would represent the number of observations in the training set. Similarly, when calculating the MSE for validation set and test set, T will represent the length of those sets only.

$$MSE = \frac{\sum_{i=1}^{M} \sum_{t=1}^{T} \left(Y_i^{(t)} - \widehat{Y}_i^{(t)}\right)^2}{(T\dot{M}} \tag{11}$$

To perform learn-and-test experiments, each considered time series was split into three sets, namely: the training set, the validation set, and the test set. The training set contained the first 80% of observations and was used to build the model. The next 10% of the time series represented the validation set which was used to tune parameters of each given model. Parameters were tuned by running all different configurations and capturing an MSE for the validation set. Parameter settings with the lowest MSE for the validation set are used to evaluate each model. Finally, the test set comprises the last 10% observations from every time series.

For comparison purposes we use the following models:

- Vector autoregression (VAR) [1]. The order of VAR was tested in range $p \in \{1, 2, 3, 4\}$.
- Vector autoregression moving-average (VARMA) [2]. VARMA was tuned with any combination of $p \in \{0, 1, 2\}$ and $q \in \{0, 1, 2\}$, where p is the order of the autoregressive part and q is the order of the moving average part.
- Multilayer presentations (MLPs). For MLP, the number of hidden nodes was tuned based on the number of concepts N in each given dataset. Any number in the range $[N, 2N]$ was tested.
- FCM-GA trained by the genetic algorithm. For the FCM-GA, the cardinality of the initial population is equal to T, the number of instances in the dataset whereas the maximum number of generations is 100. The Rank wheel is used as a crossover selection method. The probabilities of mutation $[0.01, 0.05, 0.1]$ and crossover $[0.8, 0.9]$ were tuned on a test-and-error basis individually for every considered dataset. Given the non-deterministic nature of GA, multiple runs are required to obtain statistically significant results. For each configuration, the results of 10 independent runs have been aggregated by calculating the average MSE. For both

Table 1: Overview of datasets

ID	dataset	N	T	$min(\|\rho\|)$	$med(\|\rho\|)$	$max(\|\rho\|)$
1	a-rgdp-14	14	52	0.9067	0.9905	0.9990
2	a-rgdp-per-4	4	52	0.9659	0.9903	0.9942
3	aa-3rv	3	339	0.4713	0.8783	0.8783
4	d-bhpvale-0206	2	946	0.9894	0.9947	0.9947
5	d-vix-0412	4	2177	0.9878	0.9944	0.9949
6	d-xomspaapl	3	1280	0.4712	0.8194	0.8194
7	flourc	3	99	0.9509	0.9899	0.9899
8	m-3state-un	3	407	0.9204	0.9614	0.9639
9	m-amdur	4	246	0.6468	0.8476	0.9753
10	m-bnd	2	608	0.9956	0.9978	0.9978
11	m-bndpgsp-6211	3	600	0.0189	0.4875	0.4875
12	m-bndpgspabt	6	600	0.0492	0.3108	0.9645
13	m-cpitb3m	2	792	0.0775	0.5388	0.5388
14	m-dec125910-6111	5	609	0.6654	0.9231	0.9673
15	m-dec15678-6111	5	612	0.7808	0.9753	0.9826
16	m-excess-10c-9003	10	168	0.0497	0.2623	0.7357
17	m-gasoil	4	226	0.5299	0.9842	0.9916
18	m-houst-nsa	4	648	0.4676	0.7095	0.8210
19	m-hsmort7112	2	492	0.0419	0.5209	0.5209
20	m-hsoldhst6312	2	595	0.6814	0.8407	0.8407
21	m-hstarts-3divisions	3	198	0.8022	0.8859	0.9004
22	m-ibmko-0111	2	132	0.1477	0.5739	0.5739
23	m-ibmsp-6111	2	612	0.5998	0.7999	0.7999
24	m-ibmspko-6111	3	612	0.2582	0.5998	0.6481
25	m-ip3comp	3	589	0.9431	0.9936	0.9936
26	m-ip4comp	4	792	0.9173	0.9800	0.9897
27	m-napm-8812	4	300	0.3872	0.6576	0.9416
28	m-pgspabt-6211	3	600	0.4287	0.5024	0.6510
29	m-pgspabt	3	600	0.4287	0.5024	0.6510
30	m-tenstocks	10	132	0.0282	0.4168	0.7497
31	m-unemp-states	50	429	0.0342	0.7100	0.9741
32	m-unempstatesAdj	50	416	0.0041	0.6918	0.9744
33	m-unippmitcu-6712	4	552	0.0473	0.3875	0.6889
34	q-4macro	4	214	0.2882	0.6854	0.9686
35	q-fdebt	3	171	0.9352	0.9727	0.9727
36	q-gdp-ukcaus	3	126	0.9960	0.9978	0.9978
37	q-gdpunemp	2	256	0.2665	0.6332	0.6332
38	q-gpsavedi	2	263	0.9478	0.9739	0.9739
39	q-rgdp-brkris	3	62	0.1045	0.4395	0.5292
40	q-ungdp-4812	2	258	0.3413	0.6706	0.6706
41	ushog	5	82	0.0607	0.6753	0.9174

FCM-GA and FCM-MP, we tested different configurations for the sigmoid transfer function (see Equation (2)) by changing the slope parameter in the set $\lambda \in \{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0\}$.

- FCM-MP trained by the approach proposed in this paper. FCM-MP was trained by a deterministic method, therefore only a single execution was required.

5.2. One-step-ahead forecasting

Table 2 provides a summary of the one-step-ahead predictions. Values in bold denote the lowest errors (i.e., the highest performance) as defined in Equation (11). The table shows the best-performing model concerning the parameter settings, which are estimated by an automated trial-and-error procedure. Note that occasionally, some of the methods could not be applied due to singularity constraints, which indicates a high degree of covariance within the dataset.

Using the obtained results we performed Friedman's rank-sum test which produced a p-value of 1.493E−12, which is a strong indication that there was a performance difference between the considered methods.

The superiority of FCM-MP over its direct competitor FCM-GA can be recognized visually when looking at the error values given in Table 2. The reader can notice that FCM-GA never turned out to be the best, whereas our FCM-MP won the competition 17 times.

The paired Wilcoxon signed-rank test, as displayed in Table 3, allows for a deeper analysis of the differences in performance. According to this test, our FCM-MP outperformed standard FCM-GA at level $p < 0.005$. Let us remind you that the proposed learning rule bears some additional advantages. Firstly, it is not required to perform multiple runs to obtain a credible solution. Secondly, our FCM-MP is computationally faster than standard FCM-GA which uses a population-based metaheuristic for training. These experiments support our theoretical hypothesis of the superiority of the FCM-MP over FCM-GA for one-step-ahead forecasting.

When comparing the effectiveness of FCMs with competitive methods, the results turned out to be less unequivocal. Based on the obtained p-values of the Wilcoxon test, our FCM-MP was superior to VARMA at significance level $p < 0.0001$ and superior to MLP at a significance level $p < 0.001$. However, when comparing FCM-MP with VAR, we obtained no significant performance differences. For this reason, for single-step-ahead forecasting of multivariate time series we recommend to select alternatively FCM-MP or VAR. The final decision should be based on testing, separately for every of the considered time series.

5.3. Multistep-ahead forecasting

Table 4 shows the results related to the multistep-ahead forecasting. Friedman's rank-sum test produced a p-value of 5.053e−07, indicating a significant difference between the performances of the different methods.

Table 2: MSE for one-step-ahead predictions

ID	dataset	VAR	VARMA	MLP	FCM-GA	FCM-MP
1	a-rgdp-14	**0.0027**	NA	0.0263	0.0055	0.0037
2	a-rgdp-per-4	**0.0031**	NA	0.0197	0.0043	0.0904
3	aa-3rv	0.0205	0.0201	0.0200	0.0115	**0.0028**
4	d-bhpvale-0206	**0.0004**	0.0611	0.045	0.0026	0.0027
5	d-vix-0412	**0.0004**	0.0060	**0.0004**	0.0008	**0.0004**
6	d-xomspaapl	0.0021	0.0020	0.0021	0.0095	**0.0013**
7	flourc	**0.0030**	0.0339	**0.003**	0.0066	0.0401
8	m-3state-un	**0.0006**	0.0637	0.0036	0.0035	0.0015
9	m-amdur	**0.0012**	0.8734	0.0165	0.0106	0.0056
10	m-bnd	**0.0002**	0.0104	0.0008	0.0018	0.0016
11	m-bndpgsp-6211	0.0091	0.0089	0.0089	0.0112	**0.0025**
12	m-bndpgspabt	0.0067	0.0114	0.0078	0.0125	**0.0020**
13	m-cpitb3m	**0.0002**	0.0323	0.0155	0.0058	0.0140
14	m-dec125910-6111	**0.0158**	0.0164	**0.0154**	0.0171	0.1354
15	m-dec15678-6111	0.0175	0.0171	**0.0167**	0.0178	0.1411
16	m-excess-10c-9003	0.0389	0.0321	0.0335	0.0367	**0.0052**
17	m-gasoil	**0.0066**	0.1084	0.0503	0.0095	0.0084
18	m-houst-nsa	NA	NA	0.0086	0.0153	**0.0027**
19	m-hsmort7112	**0.0010**	0.1563	0.0105	0.0077	0.0029
20	m-hsoldhst6312	**0.0027**	0.1787	0.0070	0.0066	0.0086
21	m-hstarts-3divisions	0.0081	0.1573	0.0108	0.0104	**0.0026**
22	m-ibmko-0111	0.0085	0.0066	**0.0063**	0.0409	0.0085
23	m-ibmsp-6111	0.0139	0.0143	0.0143	0.0200	**0.0035**
24	m-ibmspko-6111	0.0122	0.0124	0.0124	0.0193	**0.0032**
25	m-ip3comp	**0.0002**	0.0264	0.0052	0.0044	0.0823
26	m-ip4comp	**0.0003**	0.2580	0.0088	0.0042	0.0023
27	m-napm-8812	0.0085	0.0255	0.0119	0.0096	**0.0028**
28	m-pgspabt	0.0112	0.0111	0.0111	0.0164	**0.0027**
29	m-pgspabt-6211	0.0112	0.0111	0.0110	0.0165	**0.0033**
30	m-tenstocks	0.0243	NA	0.0222	0.0376	**0.0036**
31	m-unemp-states	0.0119	NA	0.0185	NA	**0.0049**
32	m-unempstatesAdj	0.0050	NA	0.0141	NA	**0.0048**
33	m-unippmitcu-6712	**0.0011**	0.1190	0.0180	0.0222	0.0080
34	q-4macro	**0.0010**	0.0516	0.0207	0.0099	0.0071
35	q-fdebt	**0.0032**	0.1289	0.1700	0.0046	0.0039
36	q-gdp-ukcaus	**0.0003**	0.0292	0.0180	0.0038	0.0040
37	q-gdpunemp	**0.0007**	0.0459	0.0215	0.0093	0.0046
38	q-gpsavedi	**0.0013**	0.0428	0.0556	0.0067	0.0786
39	q-rgdp-brkris	**0.0072**	0.0272	0.0177	0.0255	0.0077
40	q-ungdp-4812	**0.0006**	0.0413	0.0180	0.0071	0.0045
41	ushog	0.0097	0.1165	0.0184	0.0162	**0.0040**

Table 3: Wilcoxon Rank-Sum Test for one-step-ahead prediction methods. The adjusted Holm, Hommel, Hochberg and Bonferroni p-values are indicated as well. The $p-$values indicate whether the performance of the models to the left is significantly different from our FCM-MP model. Commonly, a threshold of $p < 0.05$ is used to declare the difference in prediction accuracy significant.

	FCM-MP				
	Wilcoxon	Bonferroni	Hochberg	Holm	Hommel
VAR	0.5563	1.0000	0.5563	0.5563	0.5563
VARMA	0.0000	0.0000	0.0000	0.0000	0.0000
MLP	0.0001	0.0008	0.0002	0.0005	0.0005
FCM-GA	0.0003	0.0029	0.0004	0.0012	0.0012

It can be noticed that FCM-MP outperforms the FCM-GA variant (our method was the best alternative for 15 datasets, whereas FCM-GA reported superior accuracy for a single problem). The adjusted p-values of a paired Wilcoxon Rank-Sum test (see Table 5) supports the superiority of our approach with respect to FCM-GA in terms of MSE ($p < 0.005$) when making long-term predictions for multivariate time series.

Further calculations of a paired Wilcoxon Rank-Sum test did not provide sufficient evidence to claim superiority in any other pairwise comparison. However, you can notice in Table 4 that the VAR model reported the lowest errors for 15 datasets. Hence, VAR and FCM-MP turned out to be the two most competitive methods in our study.

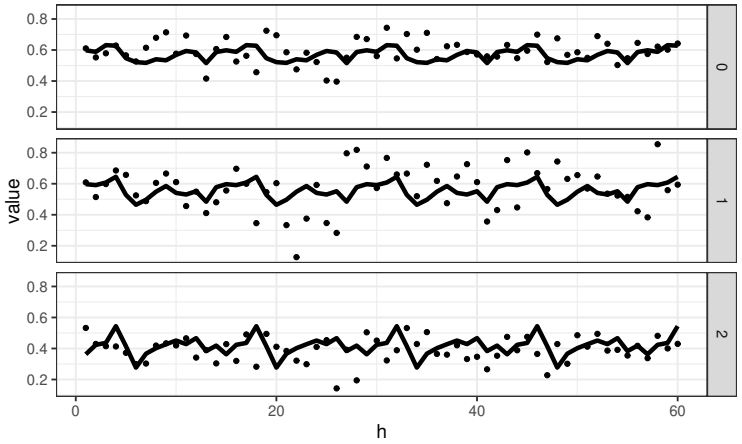

Figure 3: Predictions made by FCM-MP for the pgspabt dataset. The x-axis represents the timeline with the prediction horizon as the maximum value. The graph plots prediction and actual value for each forecasted step h. The full line represents the predicted values for the test set; Actual values are plotted as dots. The better the line matches the dots, the better the fit. The graph demonstrates the ability of FCM-MP to produce a recurring pattern.

For the illustration of the obtained results, we present some plots that vi-

Table 4: Testing error for multistep-ahead predictions

ID	dataset	VAR	VARMA	MLP	FCM-GA	FCM-MP
1	a-rgdp-14	0.0080	-	0.0516	0.3289	**0.0072**
2	a-rgdp-per-4	0.0280	-	0.0414	0.2694	**0.0024**
3	aa-3rv	**0.0200**	0.0201	0.0201	0.0832	0.0231
4	d-bhpvale-0206	0.0587	0.0611	0.1040	0.1556	**0.0300**
5	d-vix-0412	0.0061	**0.0060**	0.0063	0.0416	0.0081
6	d-xomspaapl	**0.0020**	**0.0020**	**0.0020**	0.0053	0.0023
7	flourc	0.0283	0.0339	0.0797	0.0754	**0.0061**
8	m-3state-un	0.0637	0.0637	0.0674	**0.0563**	0.0713
9	m-amdur	0.5776	0.7266	**0.0212**	0.1553	0.0558
10	m-bnd	0.0329	0.0104	0.0192	0.0301	**0.0074**
11	m-bndpgsp-6211	**0.0089**	**0.0089**	0.0090	0.0129	0.0116
12	m-bndpgspabt	0.0124	0.0114	**0.0094**	0.0226	0.0124
13	m-cpitb3m	**0.0323**	**0.0323**	0.0563	0.1904	0.1825
14	m-dec125910-6111	**0.0164**	**0.0164**	**0.0164**	0.0174	0.0219
15	m-dec15678-6111	**0.0171**	**0.0171**	0.0172	0.0193	0.0302
16	m-excess-10c-9003	**0.0321**	**0.0321**	0.0325	0.0360	0.0401
17	m-gasoil	0.1084	0.1084	0.0988	0.1303	**0.0747**
18	m-houst-nsa	0.1028	0.1083	0.1251	0.1072	**0.0809**
19	m-hsmort7112	0.1487	0.1563	0.2079	0.1521	**0.1437**
20	m-hsoldhst6312	**0.0985**	0.1787	0.2553	0.1434	0.2588
21	m-hstarts-3divisions	0.1262	0.0873	**0.0264**	0.1071	0.1194
22	m-ibmko-0111	0.0067	**0.0066**	**0.0066**	0.0091	0.0111
23	m-ibmsp-6111	**0.0143**	**0.0143**	0.0144	0.0177	0.0196
24	m-ibmspko-6111	0.0124	0.0124	**0.0123**	0.0145	0.0144
25	m-ip3comp	0.0263	0.0264	0.0140	0.2012	**0.0055**
26	m-ip4comp	0.1747	0.2580	0.0204	0.2417	**0.0070**
27	m-napm-8812	0.0255	**0.0244**	0.0297	0.0307	0.0269
28	m-pgspabt	0.0111	0.0111	**0.0110**	0.0137	0.0134
29	m-pgspabt-6211	**0.0111**	**0.0111**	**0.0111**	0.0141	0.0134
30	m-tenstocks	**0.0218**	-	0.0217	0.0245	0.0222
31	m-unemp-states	0.1560	-	0.1885	-	**0.0561**
32	m-unempstatesAdj	**0.0662**	-	0.1056	-	0.1027
33	m-unippmitcu-6712	0.1098	0.1190	**0.0668**	0.1179	**0.0683**
34	q-4macro	0.0560	**0.0255**	0.0515	0.1869	0.1258
35	q-fdebt	0.1391	0.1289	0.2869	0.1483	**0.0914**
36	q-gdp-ukcaus	0.0292	0.0222	0.0280	0.2510	**0.0014**
37	q-gdpunemp	**0.0421**	0.0459	0.1093	0.1955	0.0928
38	q-gpsavedi	**0.0428**	**0.0428**	0.0951	0.1885	0.0464
39	q-rgdp-brkris	**0.0221**	0.0272	0.0251	0.0496	0.0224
40	q-ungdp-4812	0.0672	**0.0457**	0.1111	0.1568	0.0732
41	ushog	0.0808	0.1165	0.1084	0.1470	**0.0385**

Table 5: Wilcoxon Rank-Sum Test. Adjusted Holm, Hommel, Hochberg and Bonferroni p-values indicated as well. The $p-$values indicate whether the performance of the models to the left is significantly different from our FCM-MP model. Commonly, a threshold of $p < 0.05$ is used to declare the difference in prediction accuracy significant.

	FCM-MP				
	Wilcoxon	Bonferroni	Hochberg	Holm	Hommel
VAR	0.2751	1.0000	0.4585	1.0000	0.8253
VARMA	0.1423	1.0000	0.2847	0.8540	0.7280
MLP	0.3662	1.0000	0.5232	1.0000	0.8545
FCM-GA	0.0003	0.0026	0.0026	0.0026	0.0026

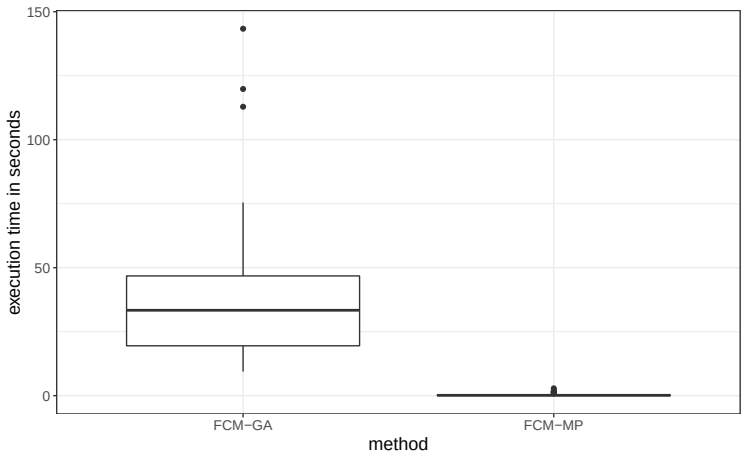

Figure 4: Speed comparison (in seconds) between FCM-GA and FCM-MP.

sualize the prediction accuracy of our model. The graphs depict the test set of the given dataset. Model predictions are plotted as a full line; observations are plotted as points. Despite there being no explicit built-in mechanisms for recurrent behavior, the reader can notice that the FCM-MP method is able to produce cyclic behavior as shown in Figure 3.

5.4. Comparison of time-efficiency

Figure 4 compares FCM-GA with FCM-MP in terms of the learning time. The graph gives a clear visual representation of the superiority of FCM-MP over FCM-GA. This speed advantage is amplified when taking into account that the deterministic nature of our learning method only requires a single execution whereas the stochastic nature attached to the FCM-GA variant demands multiple runs with the same parameter settings.

5.5. Effect of training ratio on accuracy

We examine the effect that the size of the training dataset has on algorithm accuracy by comparing the accuracy for training ratios of 70%, 80%, and 90%. As in the previous simulations, the remaining part is split equally between a validation set and a test set in which the validation set is used for parameter tuning.

As can be seen in Figure 5, the algorithm delivers a comparable performance across training rates. Wilcoxon signed-rank test (Table 6 concludes that there is indeed no proof that there is a difference between those distributions.

5.6. Transparency of the model

Compared to the single $N \times N$ weight matrix of a classical fuzzy cognitive map, our model produces a $N \times N$ for each time step in the training set. This

Table 6: Pairwise Wilcoxon Signed Rank test.

	80%	90%
70%	0.4240	0.7306
80%		1.0000

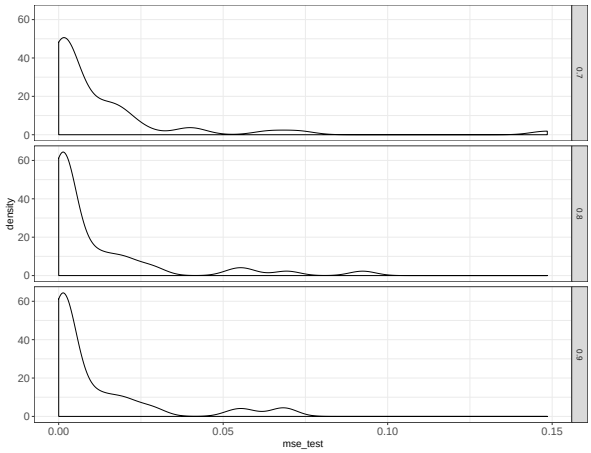

Figure 5: MSE across datasets for different training set sizes.

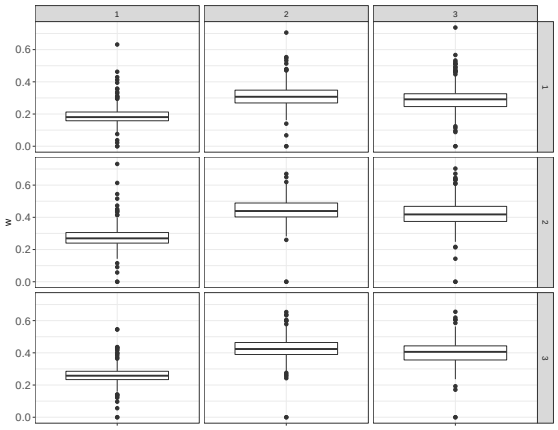

Figure 6: Visualized weights for the m-bndpgsp-6211 dataset, where w_{ij} is represented in the $i-$th row ad the $j-$th column.

reduces the transparency of the model, However, it can be partially recuperated by depicting the discovered weights as a graph. An example for the bndpgsp dataset can be seen in Figure 6.

The boxplots in Figure 6 represents each weight $<_{ij}$ in the $i-$th row and the $j-$th column. From the graph, we can derive that w_{32} is generally higher than w_{31}, indicating that the influence of C_3 is larger on C_2 then it is on C_1.

Assuming that domain knowledge on the dataset is available, a judgment from a domain expert could lend additional credit to the predictions of the model.

By benchmarking the performance of the model on a set of benchmark datasets, we are able to demonstrate the prediction accuracy of the model. However, its transparency can only be fully explored in applications where domain knowledge is available and the dataset contains concepts that have somewhat understood, and preferably causal, relationships.

6. Concluding remarks

In this research, we have presented an FCM-based solution to forecast multivariate time series. The main achievement of our proposal is a new approach to learning FCM models. By using the Moore-Penrose inverse we developed the FCM learning method which works substantially faster when compared with the genetic learning approach. Moreover, the simulation results have shown that our proposal is able to significantly outperform standard FCMs applied to single-step-ahead and multiple-step-ahead forecasting of multivariate time series. The forecasting accuracy of the proposed model also turned out to be very competitive compared to traditional techniques for time series forecasting.

The deterministic nature of the learning method provides added value compared to stochastic methods, who typically need multiple executions of the learning phase in order to obtain a statistically trustworthy result.

An FCM is characterized by a transparent and understandable topology in which nodes and edges have a specific interpretation that can be useful to decision-makers. Our FCM-based model aims at competitive prediction accuracy without sacrificing the transparent nature of the FCM. The concepts in our model map directly to a variable in the dataset and do not represent information granules or other forms of aggregated or derived information. Interpretation of a node is a prerequisite for the interpretation of edges.

On the one hand, given the dynamic behavior of weights over time, our model cannot lay claim to a fully transparent weight matrix. However, the changing weights provide the opportunity to understand how a system evolves, which is an added value when an FCM iteration has a semantic interpretation.

We encourage future research to enhance the current proposal towards a fully competitive and transparent FCM for time series prediction. Any approach preserving the understandability of a single weight matrix in time series forecasting would be a useful addition to the field. Other potential improvements include, but are not limited to, the incorporation of a dynamic time lag in during predictions and dynamic extraction of seasonal effects.

References

[1] E. Zivot, J. Wang, Vector Autoregressive Models for Multivariate Time Series, Springer New York, New York, NY, 2006, pp. 385–429.

[2] P. Aboagye-Sarfo, Q. Mai, F. M. Sanfilippo, D. B. Preen, L. M. Stewart, D. M. Fatovich, A comparison of multivariate and univariate time series approaches to modelling and forecasting emergency department demand in western australia, Journal of Biomedical Informatics 57 (Supplement C) (2015) 62 – 73.

[3] H. Lütkepohl, New Introduction to Multiple Time Series Analysis, Springer-Verlag, 2007.

[4] R. S. Tsay, Multivariate Time Series Analysis with R and Financial Applications, John Wiley, 2014.

[5] B. Kosko, Fuzzy cognitive maps, International Journal of man-machine studies 24 (1) (1986) 65–75.

[6] E. I. Papageorgiou, Review Study on Fuzzy Cognitive Maps and Their Applications during the Last Decade, in: M. Glykas (Ed.), Business Process Management, no. 444 in Studies in Computational Intelligence, Springer Berlin Heidelberg, 2013, pp. 281–298.

[7] J. L. Salmeron, T. Mansouri, M. R. S. Moghadam, A. Mardani, Learning fuzzy cognitive maps with modified asexual reproduction optimization algorithm, Knowledge-Based Systems (163) (2019) 723–735.

[8] W. Froelich, E. I. Papageorgiou, M. Samarinas, K. Skriapas, Application of evolutionary fuzzy cognitive maps to the long-term prediction of prostate cancer, Appl. Soft Comput. 12 (12) (2012) 3810–3817.

[9] E. I. Papageorgiou, W. Froelich, Application of evolutionary fuzzy cognitive maps for prediction of pulmonary infections, IEEE Transactions on Information Technology in Biomedicine 16 (1) (2012) 143–149.

[10] E. I. Papageorgiou, W. Froelich, Multi-step prediction of pulmonary infection with the use of evolutionary fuzzy cognitive maps, Neurocomputing 92 (2012) 28–35.

[11] W. Stach, L. Kurgan, W. Pedrycz, M. Reformat, Genetic learning of fuzzy cognitive maps, Fuzzy Sets and Systems 153 (3) (2005) 371–401.

[12] K. Poczeta, A. Yastrebov, Analysis of fuzzy cognitive maps with multi-step learning algorithms in valuation of owner-occupied homes, in: 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2014, pp. 1029–1035.

[13] E. I. Papageorgiou, K. Poczeta, C. Laspidou, Application of Fuzzy Cognitive Maps to water demand prediction, in: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2015, pp. 1–8.

[14] K. Poczeta, A. Yastrebov, E. I. Papageorgiou, Learning fuzzy cognitive maps using Structure Optimization Genetic Algorithm, in: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), 2015, pp. 547–554.

[15] K. Poczeta, Ł. Kubuś, A. Yastrebov, Structure Optimization and Learning of Fuzzy Cognitive Map with the Use of Evolutionary Algorithm and Graph Theory Metrics, in: S. Fidanova (Ed.), Recent Advances in Computational Optimization: Results of the Workshop on Computational Optimization WCO 2017, Studies in Computational Intelligence, Springer International Publishing, Cham, 2019, pp. 131–147.

[16] G. Felix, G. Nápoles, R. Falcon, W. Froelich, K. Vanhoof, R. Bello, A review on methods and software for fuzzy cognitive maps, Artificial Intelligence Review (2019) 1707–1737.

[17] R. H. Shumway, D. S. Stoffer, Time Series Analysis and Its Applications, Springer, 2000.

[18] W. Homenda, A. Jastrzebska, W. Pedrycz, Joining concept's based fuzzy cognitive map model with moving window technique for time series modeling, in: Computer Information Systems and Industrial Management, 2014, pp. 397–408.

[19] W. Homenda, A. Jastrzebska, W. Pedrycz, Time series modeling with fuzzy cognitive maps: Simplification strategies - the case of a posteriori removal of nodes and weights, in: Computer Information Systems and Industrial Management, 2014, pp. 409–420.

[20] X. L. Wei Lu, Jianhua Yang, The hybrids algorithm based on fuzzy cognitive map for fuzzy time series prediction, Journal of Information and Computational Science 11 (2) (2014) 357–366.

[21] H. Song, C. Miao, W. Roel, Z. Shen, F. Catthoor, Implementation of fuzzy cognitive maps based on fuzzy neural network and application in prediction of time series, IEEE Transactions on Fuzzy Systems 18 (2) (2010) 233–250.

[22] H. Song, C. Miao, Z. Shen, W. Roel, D. Maja, C. Francky, Design of fuzzy cognitive maps using neural networks for predicting chaotic time series, Neural Networks 23 (10) (2010) 1264–1275.

[23] J. L. Salmeron, R. Vidal, A. Mena, Ranking fuzzy cognitive map based scenarios with topsis, Expert Systems with Applications 39 (3) (2012) 2443–2450.

[24] C.-L. Hwang, K. Yoon, Methods for Multiple Attribute Decision Making, in: C.-L. Hwang, K. Yoon (Eds.), Multiple Attribute Decision Making: Methods and Applications A State-of-the-Art Survey, Lecture Notes in Economics and Mathematical Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 1981, pp. 58–191.

[25] J. L. Salmeron, W. Froelich, Dynamic optimization of fuzzy cognitive maps for time series forecasting, Knowledge-Based Systems 105 (2016) 29 – 37.

[26] J. L. Salmeron, A. Ruiz-Celma, A. Mena, Learning fcms with multi-local and balanced memetic algorithms for forecasting drying processes, Neurocomputing 232 (2017) 52–57.

[27] W. Froelich, P. Juszczuk, Predictive Capabilities of Adaptive and Evolutionary Fuzzy Cognitive Maps - A Comparative Study, Springer Berlin Heidelberg, 2009, pp. 153–174.

[28] W. Stach, L. Kurgan, W. Pedrycz, M. Reformat, Genetic learning of fuzzy cognitive maps, Fuzzy Sets and Systems 153 (3) (2005) 371–401.

[29] W. Stach, L. Kurgan, W. Pedrycz, M. Reformat, Learning fuzzy cognitive maps with required precision using genetic algorithm approach, Electronics Letters 40 (24) (2004) 1519–1520.

[30] W. Homenda, A. Jastrzebska, W. Pedrycz, Modeling time series with fuzzy cognitive maps, in: Proceedings of the 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2014, pp. 2055–2062.

[31] J. L. Salmeron, S. A. Rahimi, A. M. Navali, A. Sadeghpour, Medical diagnosis of rheumatoid arthritis using data driven pso–fcm with scarce datasets, Neurocomputing 232 (2017) 104–112.

[32] E. Yesil, C. Öztürk, M. F. Dodurka, A. Sakalli, Fuzzy cognitive maps learning using artificial bee colony optimization, in: FUZZ-IEEE'13, 2013, pp. 1–8.

[33] B. A. Angélico, M. Mendonça, L. V. R. de Arruda, T. Abrão, Heuristic search applied to fuzzy cognitive maps learning, in: T. Abrão (Ed.), Search Algorithms for Engineering Optimization, InTech, 2013, pp. 221–240.

[34] J. L. Salmeron, P. R. Palos-Sanchez, Uncertainty propagation in fuzzy grey cognitive maps with hebbian-like learning algorithms, IEEE Transactions on Cybernetics forthcoming.

[35] J. L. Salmeron, E. I. Papageorgiou, A fuzzy grey cognitive maps-based decision support system for radiotherapy treatment planning, Knowledge-Based Systems 30 (1) (2012) 151–160.

[36] E. 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) (2012) 150–163.

[37] S. Bueno, J. L. Salmeron, Benchmarking main activation functions in fuzzy cognitive maps, Expert Systems with Applications 36 (3, Part 1) (2009) 5221–5229.

[38] M. Morel, C. Achard, R. Kulpa, S. Dubuisson, Time-series averaging using constrained dynamic time warping with tolerance, Pattern Recognition 74 (2018) 77–89.

[39] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE transactions on acoustics, speech, and signal processing 26 (1) (1978) 43–49.