Short-term Cognitive Networks, Flexible Reasoning and Nonsynaptic Learning

Peer-reviewed author version

NAPOLES RUIZ, Gonzalo; VANHOENSHOVEN, Frank & VANHOOF, Koen (2019) Short-term Cognitive Networks, Flexible Reasoning and Nonsynaptic Learning. In: NEURAL NETWORKS, 115, p. 72-81.

DOI: 10.1016/j.neunet.2019.03.012
Handle: http://hdl.handle.net/1942/29769

Short-term Cognitive Networks, Flexible Reasoning and Nonsynaptic Learning

Gonzalo Nápoles, Frank Vanhoenshoven, Koen Vanhoof

Faculty of Business Economics, Hasselt University, Belgium

Abstract

While the machine learning literature dedicated to fully automated reasoning algorithms is abundant, the number of methods enabling the inference process on the basis of previously defined knowledge structures is scanter. Fuzzy Cognitive Maps (FCMs) are recurrent neural networks that can be exploited towards this goal because of their flexibility to handle external knowledge. However, FCMs suffer from a number of issues that range from the limited prediction horizon to the absence of theoretically sound learning algorithms able to produce accurate predictions. In this paper we propose a neural system named *Short-term Cognitive Networks* that tackle some of these limitations. In our model, used for regression and pattern completion, weights are not constricted and may have a causal nature or not. As a second contribution, we present a nonsynaptic learning algorithm to improve the network performance without modifying the previously defined weight matrix. Besides, we derive a stop condition to prevent the algorithm from iterating without significantly decreasing the global simulation error.

Keywords: Short-term memory, cognitive mapping, nonsynaptic learning, modeling, simulation

1. Introduction

Fuzzy Cognitive Maps (FCMs) [16, 17] continue to grow in popularity mainly because of their potential to deal with expert knowledge. In these recurrent neural networks, neurons have a specific meaning for the problem under investigation, whereas edges denote causal relationships [6]. In a simulation context, the reasoning is devoted to computing the system state attached to an initial vector, which is regularly provided by the expert. This is equivalent to computing the activation value of each decision neuron from non-decision ones.

Likewise, fuzzy cognitive mapping has been used to design more complex machine learning solutions. The development of forecasting models for univariate / multivariate time series [19, 26, 8] and granular cognitive classifiers [22, 23] are examples that illustrate the potential attached to this technique. Even the *Neurocomputing* journal recently dedicated a special issue [9] to relevant theoretical advances in this field. But to what extent some contributions reported in the literature can be considered authentic FCM solutions (as originally defined by Kosko [16, 17]) may be questionable. For example, would it be correct to claim that an FCM model adjusted using a data-driven heuristic learning algorithm properly reflects real-world causalities between any two concepts?

Generally speaking, FCMs are far from being a theoretically robust simulation technique. The rather limited prediction horizon of neurons, the absence of an accurate learning algorithm and the ambiguous semantics of fixed points are examples of shortcomings identified in this model. Carvalho [4] discussed some of these problems, but for some reason the current FCM research remains inside the box. While some of the above-mentioned drawbacks seem perfectly solvable, the fact is that traditional FCMs face theoretical barriers difficult to surmount. For example, Concepción et al. [5] introduced the contraction theorem for FCMs which states that the activation space of a sigmoid neuron may shrink infinitely, without any guarantee of reaching an equilibrium point. Even if the system converges, there is no guarantee that the discovered equilibrium attractor leads to the lowest simulation error.

In spite of the above-mentioned issues, the cognitive mapping principle stands as a powerful approach to perform simulations on the basis of previously defined knowledge structures. In our research, we refer to this valuable characteristic as the *flexible reasoning*. Not many machine learning techniques allow to directly embed knowledge into their reasoning process. It is worth mentioning that in the flexible reasoning paradigm the knowledge structures can be defined by experts or computed by other algorithms. However, designing such a highly flexible, sound simulation model under the umbrella of traditional FCMs may become (unnecessarily) difficult. For example, what if the experts have a clear picture of how the variables correlate, but they are unable to confidently claim the existence of causal relationships between them?

This paper brings to life the following contributions. Firstly, we discuss some major problems affecting FCM-based models which motivated this research. Secondly, we introduce a neural system referred to as *Short-term Cognitive Networks*, which allow reasoning on the basis of previously defined knowledge structures. In our research, these structures refer to the weight matrix that determines the interaction among domain variables.

Email address: gonzalo.napoles@uhasselt.be (Gonzalo Nápoles)

Thirdly, we propose a nonsynaptic learning algorithm to reduce the global simulation error without modifying the weight matrix. This gradient-based algorithm regulates the excitation degree of each neuron in each iteration. As a last contribution, we analytically derive a stop condition to prevent the learning algorithm from iterating without decreasing the global error, thus notably increasing its efficiency.

The rest of this paper is organized as follows. Section 2 provides a concise background on traditional FCMs, whereas Section 3 discusses the key motivations behind our proposal. Section 4 presents our flexible reasoning model baptized as *Short-term Cognitive Networks*, while Section 5 is dedicated to the nonsynaptic learning method. Section 6 introduces the numerical simulations, and Section 7 outlines the concluding remarks and future research avenues.

2. Fuzzy Cognitive Mapping

In a nutshell, an FCM denotes a complex network of causal relationships among abstract concepts. Figure 1 shows an FCM describing the complex interrelations in a simplified food chain. The reasoning model is intuitive and fairly self-explanatory because the modeled system can be graphically visualized. It can be seen that predators thrive when there is a lot of prey, whereas the prey animals themselves are under pressure from the predators but can be boosted by the presence of grass. Given a set of predators, prey and grass, the FCM can evolve in several *iterations*. It is likely that after a sufficient number of iterations, the network will eventually find a stable state (i.e., fixed point) in which the system reaches a balance.

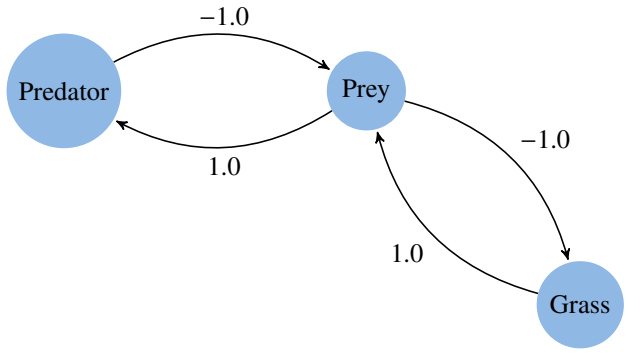

Figure 1: FCM-based system comprised of three neuronal concepts modeling the interrelations in a simplified food chain.

In its mathematical form, an FCM can be defined by a 4-tuple $\langle C, \mathcal{W}, \mathcal{A}, f(\cdot)\rangle$, where $C = \{C_1, \dots C_M\}$ comprises the variables describing the physical system, typically visualized as nodes in the network while $\mathcal{W}$ denotes an $M \times M$ weight matrix where $w_{ij} \in [-1, 1]$ encodes a causal relation of C_i upon C_j. In the graph, each weight is visualized as a labeled edge between the corresponding concepts. The interpretation of causal weights is depicted as follows:

- $w_{ij} > 0$: If activated, C_i will *excite* C_j. More explicitly, 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, A_i will *inhibit* C_j. More explicitly, lower (higher) activation values of C_i in the current iteration will lead to higher (lower) activation values of C_j in the following iteration.

- $w_{ij} = 0$: C_i will not influence C_j. In the graphical model, this relation is usually indicated by the absence of an edge between the two concepts.

These statements are different from common phrases like *an increase (decrease) in one will cause an increase (decrease) in the other*. The systematic misinterpretation of causal weights has been widely discussed by Carvalho [3]. In this paper, we have chosen a definition that is more in line with the inference rule attached to the neural system.

The function $\mathcal{A} : C \times \mathbb{N} \rightarrow A_i^{(t)}$ comprises the activation value of the C_i concept in the tth iteration step. The initial activation value $A_i^{(0)}$ can be either determined by the expert at the beginning of the simulation process, or computed from historical data. During the reasoning process, such values are updated using the inference rule depicted in Equation 1. In reference to the previous paragraph, it can be seen that $A_i^{(t+1)}$ is defined by the activation values of its connected concepts in the previous iteration, not by any increase or decrease. On the other hand, this reasoning rule treats the concepts as standard McCulloch-Pitts neurons [20] in a causal graph, which is why FCMs are considered (recurrent) neural systems.

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

A pivotal aspect of any neural reasoning model is the transfer function $f : \mathbb{R} \rightarrow I$ used to maintain the incoming evidence within the activation space I, often $I = [0, 1]$ or $I = [-1, 1]$. Equation 2 displays the sigmoid function, a common choice in fuzzy cognitive modeling because of its ability to represent both qualitative and quantitative scenarios,

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

where $\lambda > 0$ controls the function slope. Recent studies [24, 25] have shown that this parameter may have a significant impact on the FCM's dynamic behavior.

For a sufficient number of iterations T, the recurrent system can be attracted to one of the following states:

- **Fixed-point** ($\exists t_\alpha \in \{1, 2, \dots, (T-1)\} : A^{(t+1)} = A^{(t)}, \forall t \geq t_\alpha$): the map produces the same output after the cycle t_α, so $A^{(t_\alpha)} = A^{(t_\alpha+1)} = A^{(t_\alpha+2)} = \cdots = A^{(T)}$.

- **Limit cycle** ($\exists t_\alpha, P \in \{1, 2, \dots, (T-1)\} : A^{(t+P)} = A^{(t)}, \forall t \geq t_\alpha$): the map produces the same output periodically after the cycle t_α, so $A^{(t_\alpha)} = A^{(t_\alpha+P)} = A^{(t_\alpha+2P)} = \cdots = A^{(t_\alpha+jP)}$ where $t_\alpha + jP \leq T$, such that $j \in \{1, 2, \dots, (T-1)\}$.

- **Chaos**: the map continues to produce different state vectors for successive cycles.

Typically, the meaning of an iteration is ignored when modeling simulation or pattern classification scenarios. In these cases, it is desired that the system converges to a fixed point in order to draw conclusions. When forecasting time series, however, iterations are typically equated with the time span between two discrete observations in a dataset. Generically speaking, fixed-point attractors are less desirable when forecasting long-term time series since such equilibrium points suppress the model's ability to forecast fluctuations.

3. Criticism and Motivation

For many years we have presented FCMs as an almost infallible technique for modeling and simulating complex systems. However, as already mentioned, FCM-based systems exhibit serious drawbacks that may affect the simulation outcomes. In this section, we discuss three of such issues: (1) the causation fallacy behind existing learning algorithms, (2) the restrictions imposed by the FCM model, and (3) the implications of unique attractors in simulation scenarios.

According to Kosko [17], each arrow in an FCM denotes a causal relationship that is described by a signed and bounded weight. The causality assumption is a key aspect here, otherwise we would be in presence of traditional associative neural networks. But some of the most accurate learning algorithms reported in the literature [34] [33] [29] rely on heuristic search methods, whose results depend on the initial conditions and random sequences. This implies that we should not claim any kind of causation unless we establish domain-dependent constraints beforehand to guide the search. As an alternative, we could employ Hebbian-like algorithms [30] which attempt to compute a model with minimal deviation from a predefined initial matrix. Although those methods could be useful in control problems, their poor generalization capability [6] makes them unsuitable for more challenging prediction scenarios.

Another issue with the fuzzy cognitive model is that both causal weights and neurons' activation values are bounded. It can be verified that such constraints reduce the FCM prediction horizon, which can be defined as the interval of activation values that a neuron can produce.

Proof. Let C_i be a sigmoid neuron, then its activation value in the tth iteration can be expressed as:

$$A_i^{(t)}(k) = \frac{1}{1 + e^{-\lambda\left(\sum_{j=1}^{M} w_{ji} A_j^{(t-1)}(k)\right)}}, \forall k$$

where k denotes the index of the activation vector used to start the recurrent reasoning process.

Since $-1 \leq w_{ji} \leq 1$ and $0 \leq A_j^{(t-1)}(k) \leq 1$, then

$$\min(C_i) \leq \sum_{j=1}^{M} w_{ji} A_j^{(t-1)}(k) \leq \max(C_i), \forall k$$

where

$$\min(C_i) = \sum_{j=1}^{M} \frac{w_{ji}(1 - \text{sig}(w_{ji}))}{2}$$

$$\max(C_i) = \sum_{j=1}^{M} \frac{w_{ji}(1 + \text{sig}(w_{ji}))}{2}$$

denote the minimal and maximal bounds, respectively, for the raw activation values of neurons in the current iteration, before being modified with the sigmoid transfer function. Such values are obtained by assuming that both neurons' activation values and and weights have extreme values.

Moreover, since the sigmoid function $f(x)$ is monotonically non-decreasing, we can confidently state that:

$$\frac{1}{1 + e^{-\lambda(\min(C_i))}} \leq A_i^{(t)}(k) \leq \frac{1}{1 + e^{-\lambda(\max(C_i))}}, \forall k.$$

Observe that the neuron will never reach values outside this activation interval, which reduces its prediction horizon. For example, let us assume that the event C_i is caused by C_1 and C_2 such that $w_{1i} = w_{2i} = -1$ and $\lambda = 1$, then the minimal value that C_i can produce is 0.1192, regardless of the input to C_1 and C_2, or the number of iterations. ■

It is fair to mention that we can tackle this problem from different angles. For example, if we increase the lambda value to 5 in the previous example, then the activation bounds will be expanded. However, the sigmoid function will resemble the binary activator, and again we are reducing the prediction horizon. Perhaps the easiest approach to increase the prediction ability of these models is to assume that $w_{ji} \in \mathbb{R}$ which could comprise a causal meaning or not.

An important choice that has to be made refers to the interpretation of a node in the FCM. Intuitively, one would be inclined to make a one-to-one mapping between a variable and a node. This is not the only option though. Previous research [12] has produced successful solutions where the nodes in the FCM are actually information granules, bundling more information than just a bare observation in a time series. Other approaches [27] apply machine learning to make an appropriate selection of concepts based on data. Whatever approach has been selected, simplification strategies [13] can be applied to enhance the understandability of complex maps.

The last issue refers to the FCM convergence to a unique fixed point, which becomes a cause for concern in prediction problems. The existence and uniqueness of the fixed-point attractor on FCM-based models is a complex problem that has been rigorously studied in the literature [15, 11]. While these results have been found relevant in control problems, their usability in other scenarios is less evident. Being more explicit, if the FCM converges to a unique fixed-point attractor, then the model will produce the same state vector regardless of the initial activation vector. Nápoles et al. [21] concluded that reaching a suitable tradeoff between convergence and accuracy is not always possible, while Concepción et al. [5] analytically proved that the feasible state space of a sigmoid FCM will always shrink after each iteration with no guarantee of convergence to a fixed-point attractor.

The aforementioned issues lead to the hypothesis that we can obtain lower simulation errors with an FCM-like neural model by 1) relaxing the stringent causality assumption where $w_{ij} \in [-1, 1]$ and 2) limiting the number of iteration step when performing the reasoning process.

4. Short-term Cognitive Networks

In this section, we present a neural system baptized as *Short-term Cognitive Networks* (STCNs) that attempts to address the main drawbacks discussed above. More explicitly, we intend to develop a flexible neural model devoted to *scenario simulation* where the domain experts have the responsibility to design the network architecture, while the learning algorithm is dedicated to reducing the global simulation error without modifying the expert knowledge. In order to make a clear distinction between these contributions, the present section focuses on the network architecture and reasoning, whereas the next section elaborates on the supervised learning algorithm.

4.1. Architecture and Reasoning

In a simulation context, an STCN can be defined as a neural network where each problem variable is denoted by a neural processing entity, whereas weights define the relation between variables. Hidden neurons are not allowed as the expert could not establish a comprehensible weight matrix due to the lack of semantics attached to these entities.

In the classical FCM formalism, the w_{ij} weight defines the type of causality between C_i and C_j, and to what extent the C_i event causes the C_j event. But would it be possible design a meaningful FCM-like reasoning model if we remove the causality assumption? We can definitely do so, although weights' interpretation will definitely change. In the STCN approach, $w_{ji} \in \mathbb{R}$ denotes the rate of change on the conditional mean of C_i with respect to C_j, assuming that the other neurons impacting C_i are fixed. Observe that this is similar to interpreting the coefficients in a multiple regression equation.

Another important component of any neural network refers to the mechanism used to propagate the initial information. Equation 3 and 4 display the neural reasoning rule attached to the proposed cognitive network,

$$A_i^{(t+1)}(k) = f_i^{(t+1)}\left(\sum_{j=1}^{M} w_{ji}\Psi_j^{(t)}(k)\right) \tag{3}$$

where

$$\Psi_j^{(t)}(k) = \begin{cases} f_j^{(t)}\left(\sum_{l=1}^{M} w_{lj}A_l^{(0)}(k)\right) & t > 0 \\ A_j^{(0)}(k) & t = 0 \end{cases} \tag{4}$$

represents *short-term evidence* attached to the ith neural entity in the current iteration t such that $t = \{1, 2, \ldots, T\}$, using the kth example as the excitation vector. Equation 5 formalizes the generalized sigmoid function used to squash the neuron's activation within the [0, 1] interval,

$$f_i^{(t)}(x) = \frac{1}{(1 + q_i^{(t)}e^{-\lambda_i^{(t)}(x-h_i^{(t)})})^{1/v_i^{(t)}}} \tag{5}$$

where $q_i > 0$, $\lambda_i > 0$, $h_i \in \mathbb{R}$ and $v_i > 0$ are parameters that can be used to compensate the lack of flexibility imposed by the fact that weights cannot be modified, as they were fixed by the expert during the modeling stage.

The STCNs' reasoning process comprises two mechanisms: the forgetting step and the information propagation. During the first step (see Figure 2) the network discharges the evidence accumulated until the current iteration; only the sigmoid parameters associated with each neuron are preserved. The second step (see Figure 3) is concerned with computing the short-term evidence $\Psi_j^{(t)}(k)$ by multiplying the initial activation vector with the fixed weight matrix. As a final step, the evidence collected from neurons in the iteration t is propagated to the following iteration. In computational terms, each block has a complexity of $O(M^2)$ for each activation vector.

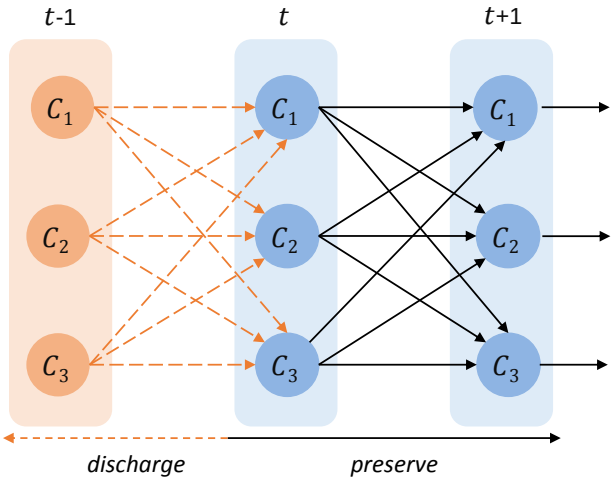

Figure 2: Forgetting mechanism in the STCN model.

In the following sub-section, we discuss how the proposed iterative neural architecture helps increase the model's accuracy when performing prediction tasks.

4.2. Dynamical Properties

When analyzing the way in which STCNs operate, the issues related with time series forecasting ring a bell. However, it is convenient to remark that STCN iterations and time steps are not necessarily equivalent. This implies that our neural system can be used in presence of static data, so the subsequent iterations can be seen as a *regularizer*.

As illustrated in Figures 2 and 3, in each STCN iteration the model can be effectively unfolded into a three-layer feed-forward neural network. This process can be done by mapping each iteration t with a layer in the multilayer network where $w_{ji}^{(t)} = w_{ji}^{(t+1)}, \forall t$. Due to the successive discharging of the old evidence it might be claimed that the system can be described by the last neural block. However, we should take into account

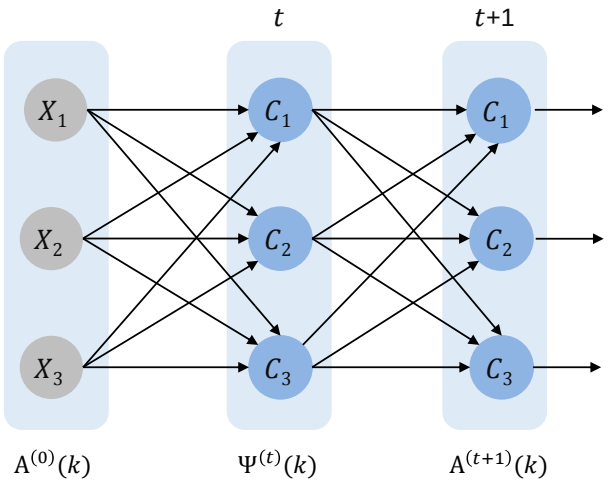

Figure 3: Inference propagation in the STCN model.

that the forgetting operator only discharges the first abstract layer of the current neural block.

Therefore, the STCN model does involve an implicit long-term recurrence such that the knowledge preserved from the previous iteration is stored within the neurons. This happens because the data used to adjust the sigmoid parameters in the $t+1$ iteration (during the learning phase) is obtained after transforming the original data with the previously computed parameters. This is similar to the regularization based on data transformation [10] used in deep learning models.

In the STCN algorithm, the updating rule finishes once the network reaches a fixed number of iterations. If the iterations are not explicitly defined, and due to we have no knowledge on the real simulation error when exploiting the model for a unseen pattern, the maximum number of iterations will be determined during the supervised training phase.

5. Nonsynaptic Learning

Nonsynaptic learning was first introduced and applied by Nápoles et al. [24] to improve the convergence of FCM-based models equipped with sigmoid transfer functions. In the STCN context, the nonsynaptic learning seems a suitable approach to reduce the simulation error without the need of altering the knowledge stored into the synaptic connections. This can be implemented by computing the sigmoid function parameters $q_i > 0$, $\lambda_i > 0$, $h_i \in \mathbb{R}$ and $v_i > 0$ associated with the ith transfer function in each iteration.

The nonsynaptic learning procedure relies on the gradient descent method [28, 2]. The first step towards formalizing the learning rule is concerned with the definition of a differentiable error function for *each* iteration, as we assume that the network will forget the old information due to the short-term memory. This implies that we will perform a separate learning process for iteration in an incremental fashion where only the previous sigmoid parameters are exploited, thus reducing the algorithm's complexity. Equation 6 shows how to compute the error of the ith neuron in the t iteration,

$$E_i^{(t)} = \sum_{k=1}^{K} \left(\frac{1}{\left(1 + q_i^{(t)} e^{-\lambda_i^{(t)}\left(\hat{A}_{ik}^{(t)} - h_i^{(t)}\right)}\right)^{\frac{1}{v_i^{(t)}}}} - Y(k) \right)^2 \tag{6}$$

where $\hat{A}_{ik}^{(t)} = \sum_{j=1}^{M} w_{ji} \Psi_j^{(t-1)}(k)$ is the raw value for the ith neuron before being transformed, and K is the number of training instances used as simulation examples. We can define a training example as a $|C|$-dimensional vector comprising the activation values of all variables, where any variable in the model can be predicted from the remaining ones.

Before applying the nonsynaptic learning method, we first transform the data into a sigmoid space since our model lacks a formal output layer with linear units, as typically occurs in multilayer perceptron networks.

Once the sigmoid transformation step is done, we iteratively improve an initial solution until a maximum number of epochs P is reached. The parameter vector $X = [\lambda_i^{(t)}, h_i^{(t)}, q_i^{(t)}, v_i^{(t)}]$ is updated by using the gradient-based rule:

$$X^{(l+1)} = X^{(l)} - Z^{(l+1)} \tag{7}$$

$$Z^{(l+1)} = \beta Z^{(l)} + \eta \nabla E^{(t)} \tag{8}$$

where $\beta \geq 0$ is the momentum, $\eta > 0$ is the learning rate, while $\nabla E^{(t)}(C_i) = \left[\frac{\partial_E}{\partial_{\lambda_i^{(t)}}}, \frac{\partial_E}{\partial_{h_i^{(t)}}}, \frac{\partial_E}{\partial_{q_i^{(t)}}}, \frac{\partial_E}{\partial_{v_i^{(t)}}}\right]$, such that

$$\frac{\partial_E}{\partial_{\lambda_i^{(t)}}} = \sum_{k=1}^{K} \frac{\gamma(\hat{A}_{ik}^{(t)} - h_i^{(t)})(q_i^{(t)} \vartheta^{\left(-1-\frac{1}{v_i^{(t)}}\right)})(\vartheta^{-\frac{1}{v_i^{(t)}}} - Y(k))}{v_i^{(t)}},$$

$$\frac{\partial_E}{\partial_{h_i^{(t)}}} = \sum_{k=1}^{K} -\frac{\gamma(q_i^{(t)} \vartheta^{\left(-1-\frac{1}{v_i^{(t)}}\right)})(\vartheta^{-\frac{1}{v_i^{(t)}}} - Y(k))\lambda_i^{(t)}}{v_i^{(t)}},$$

$$\frac{\partial_E}{\partial_{q_i^{(t)}}} = \sum_{k=1}^{K} -\frac{\gamma(\vartheta^{\left(-1-\frac{1}{v_i^{(t)}}\right)})(\vartheta^{-\frac{1}{v_i^{(t)}}} - Y(k))}{v_i^{(t)}},$$

$$\frac{\partial_E}{\partial_{v_i^{(t)}}} = \sum_{k=1}^{K} \frac{(2\vartheta^{-\frac{1}{v_i^{(t)}}} \ln(\vartheta))(\vartheta^{-\frac{1}{v_i^{(t)}}} - Y(k))}{v_i^{(t)} v_i^{(t)}},$$

where γ and ϑ are defined as follows:

$$\gamma = 2e^{-(\hat{A}_{ik}^{(t)} - h_i^{(t)})\lambda_i^{(t)}}$$

$$\vartheta = 1 + q_i^{(t)} e^{-\left(\hat{A}_{ik}^{(t)} - h_i^{(t)}\right)\lambda_i^{(t)}}.$$

Equation 9 shows the inverse sigmoid function used to reverse the forecasted values back to the original domain. This function is not defined when $-\frac{1-y^{-v_i}}{q_i} \leq 0$, but this scenario is not possible because $q_i > 0$, $v_i > 0$ and $0 < y < 1$. Due to this function may produce values that grow towards infinity, we confine the real values of each variable X_i to $[L_i, U_i]$, where L_i and U_i denote the minimum and maximum values observed in the dataset for the ith variable, respectively.

$$f_i^{-1}(y) = \frac{-\ln\left(-\frac{1-y^{-v_i}}{q_i}\right) + h_i\lambda_i}{\lambda_i} \tag{9}$$

This learning method is applied sequentially to each STCN iteration since the parameters estimated in the current iteration will be adopted to compute the following short-term evidence. Each learning process will stop either when a maximal number of epochs is reached or when the variations on the parameters from an iteration to another are small enough. It can be proved that, if the latter situation comes to light, then the STCN model will enter into a *stationary* state in which the network will continue to produce similar approximation errors.

Proof. Let us assume that an implicit STCN reached a good enough local optimum in the $t+1$ iteration step, then the global simulation error for the network is:

$$E^{(t+1)} = \sum_{i=1}^{M}\sum_{k=1}^{K}\left(f_i^{(t+1)}\left(\sum_{j=1}^{M} w_{ji}\Psi_j^{(t)}(k)\right) - Y_i(k)\right)^2$$

where $Y_i(k)$ represents the real (expected) value associated with the C_i neuron for the kth simulation example, M denotes the number of neural processing entities, while the short-term evidence is $\Psi_j^{(t)}(k) = f_j^{(t)}\left(\sum_{l=1}^{M} w_{lj}A_l^{(0)}(k)\right)$.

If $\Delta E^{(t+1)} = \left(E^{(t+1)} - E^{(t+2)}\right)^2 < \xi_1$, $t+2 \leq T$, for a small enough $\xi_1 > 0$, then two scenarios are possible:

- $F_1 = \{f_i^{(t+1)}\}_{i=1}^{M}$ and $F_2 = \{f_i^{(t+2)}\}_{i=1}^{M}$ represent two different local optima. Consequently,

$$\exists C_i \in C \mid \int_{-M}^{+M}\left(f_i^{(t+1)}(x) - f_i^{(t+2)}(x)\right)^2 dx > \xi_2.$$

- $F_1 = \{f_i^{(t+1)}\}_{i=1}^{M}$ and $F_2 = \{f_i^{(t+2)}\}_{i=1}^{M}$ represent the same local optimum. Consequently,

$$\nexists C_i \in C \mid \int_{-M}^{+M}\left(f_i^{(t+1)}(x) - f_i^{(t+2)}(x)\right)^2 dx > \xi_2.$$

If the first scenario comes to light, then we are in presence of a multimodal space since the families of sigmoid functions F_1 and F_2 lead to equally accurate solutions, but there exist at least a pair of functions $f_i^{(t+1)}(x) \in F_1$ and $f_i^{(t+2)}(x) \in F_2$ that differ in their shapes. This implies that the network could still have a chance to reach a better solution, thus we should continue iterating. In the second case, the shapes of all pairs of functions $f_i^{(t+1)}(x) \in F_1$ and $f_i^{(t+2)}(x) \in F_2$ are significantly similar. For a small enough ξ_2 and given the fact that STCNs employ a short-term reasoning process on which weights do not change from an iteration to the following, we can claim that $\Psi_j^{(t+1)}(k) \approx \Psi_j^{(t+2)}(k), \forall\, j,k$ because

$$f_j^{(t+1)}\left(\sum_{l=1}^{M} w_{lj}A_l^{(0)}(k)\right) \approx f_j^{(t+2)}\left(\sum_{l=1}^{M} w_{lj}A_l^{(0)}(k)\right).$$

Moreover, since the proposed learning algorithm is deterministic, we can expect the same behavior for the sigmoid functions in $\{f_i^{(t+3)}\}_{i=1}^{M}, \ldots, \{f_i^{(T-1)}\}_{i=1}^{M}$, therefore

$$\int_{-M}^{+M}\left(f_i^{(p)}(x) - f_i^{(p+1)}(x)\right)^2 dx \approx 0$$

such that $p = \{t+1, t+2, \ldots, T-1\}$. Given the fact that the sigmoid activator is a continuous, non-noisy function we can claim that $\Delta E^{(p)} \approx 0$. This suggests that the STCN reached a stationary state on which further learning iterations will not significantly reduce the simulation error. ■

6. Numerical Simulations

In this section, we evaluate the performance of STCNs in simulation scenarios where the goal is to forecast the value of any variable from the remaining ones.

6.1. Proof of Concept

In our first experiment, we illustrate how to build an STCN modeling the well-known Iris problem. As our neural system is not devoted to solving any classification problem, we remove the decision class during the preprocessing stage. This implies that our dataset will be comprised of 4 numerical attributes, so the prediction problem consist on forecasting the value of any attribute from the remaining ones.

The first step in modeling this problem is to represent each attribute with a sigmoid neuron. As an alternative to the lack of expert knowledge for this problem, we can model the relation between neurons C_j and C_i as a linear regression equation $A_i = w_{ji}A_j + r_{ji}$. Equations 10 and 11 display how to compute such coefficients from the training dataset,

$$w_{ji} = \frac{K\sum_k x_i(k)x_j(k) - \sum_k x_i(k)\sum_k x_j(k)}{K(\sum_k x_j(k)^2) - (\sum_k x_j(k))^2} \tag{10}$$

$$r_{ji} = \frac{\sum_k x_j(k)^2 \sum_i x_i(k) - \sum_k x_i(k)x_j(k)\sum_k x_j(k)}{K(\sum_k x_j(k)^2) - (\sum_k x_j(k))^2} \tag{11}$$

where K denotes the number of instances in the training set, whereas $x_i(k)$ represents the value of the i-th variable as coded in the k-th training example. It should be stated that the r_{ji} coefficients are used to initialize the offset parameter of the ith neuron as $h_i^{(0)} = \min\{1, \sum_{j=1}^{M} r_{ji}\}$, which simulates the bias in a McCulloch-Pitts neuron [20]. This initialization strategy may help reach better results with less effort when training neural system. The remaining sigmoid function parameters are internalized using a random approach

Figure 4 portrays the Pearson correlation between variables describing the Iris problem. This symmetric matrix provides a clear picture of the strength of each synaptic connection on the resulting nonlinear neural system. It should be remarked that such correlation values are not the weights defining the relation among concepts, but they serve as an indicator on the strength on weights computed by Equation 10.

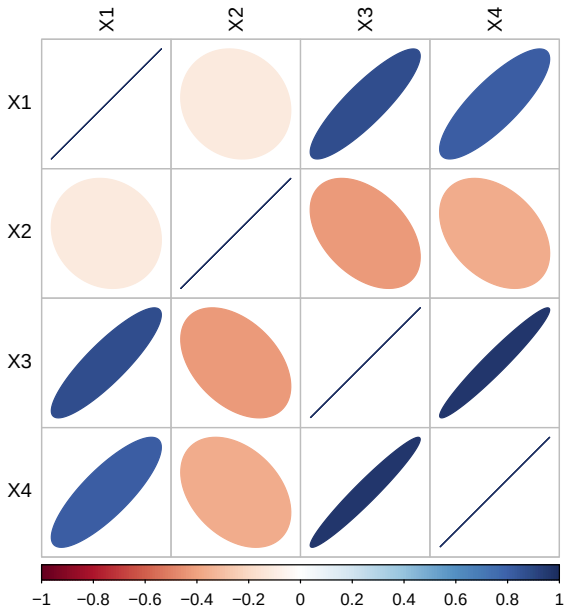

Figure 4: Pearson correlation between Iris variables.

Figure 5 shows the STCN for the Iris dataset after computing the coefficient matrix. As a second step, we fine-tune the model by using our nonsynaptic approach such that the learning rate is set to 0.001, the momentum is set to 0.85, while the number of epochs for each training process is set to 500. These values have been arbitrarily selected as they reported good average results during preliminary simulations.

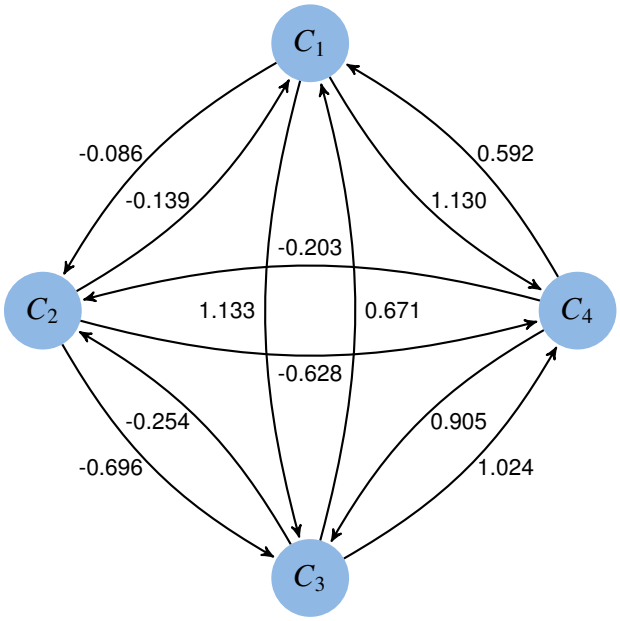

Figure 5: STCN model associated with the *Iris* dataset where neurons in the cognitive network represent the length (C_1) and width (C_2) of the sepal leaf and the length (C_3) and width (C_4) of the petal leaf. In this model, w_{ji} denotes the rate of change on the conditional mean of C_i with respect to C_j, assuming that the other neurons impacting C_i are fixed.

In this example, the initial parameter values are $h_1 = 0.762$, $h_2 = 1.0$, $h_3 = 0.808$, $h_4 = 0.686$, $\lambda_i = 2.5$, $q_i = 1.0$ and $v_i = 1.0$. Moreover, the sigmoid parameters used to transform the space are arbitrarily set to $\lambda = 2.5$, $h = 0.5$, $v = q = 1.0$. This configuration reports an average error of 0.124 across all neurons, but after applying the nonsynaptic learning algorithm the global error decreased to 0.037.

Figure 6 and 7 display, as an example, the sigmoid function adjustment for the the neuron denoting the $X1$ variable (i.e., sepal length). The orange line represents the sigmoid function in the previous iteration step, while the blue one indicates the adjusted function. In this simulation, the initial solution was set as follows: $\lambda_1 = 5$, $h_1 = 0.5$, $q_1 = 1$ and $v_1 = 1$. After a few iterations the learning algorithm was able to decrease the simulation error from 0.3789 to 0.0007.

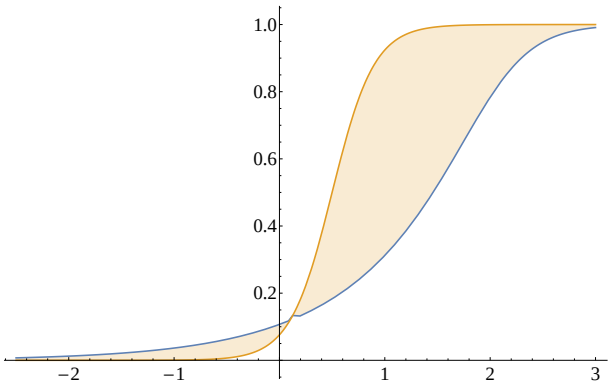

Figure 6: Adjustment of the sigmoid transfer function associated with the $C1$ neuron at the beginning of the STCN reasoning process.

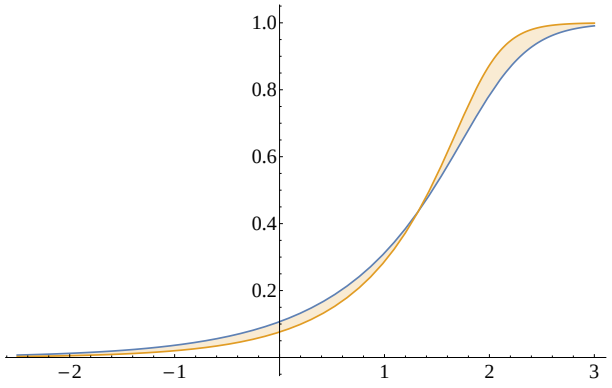

Figure 7: Adjustment of the sigmoid transfer function associated with the $C1$ neuron towards the end of the STCN reasoning process.

Figure 8 portrays the training error after performing several STCN iterations. This simulation illustrates how the error progressively decreases from an STCN iteration to the following, until a local optimum is discovered. Towards the end, the STCN becomes stationary since there is no reason to update the shape of the sigmoid function attached to each neural processing entity. Therefore, when this situation is detected (as explained in Section 5) the learning process stops.

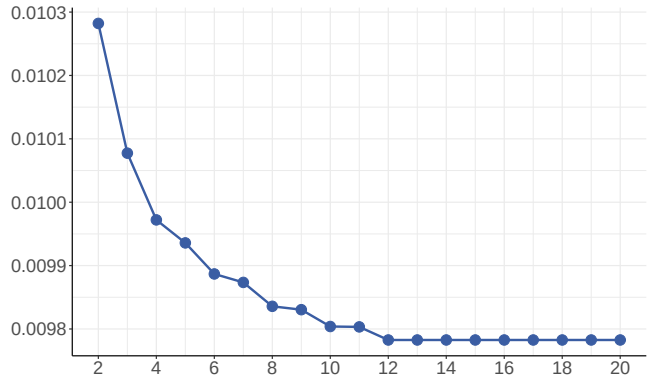

Figure 8: Global simulation error in each STCN iteration.

Next, we compare the performance of our algorithm (in terms of approximation error) against state-of-the-art algorithms used for regression and pattern completion.

6.2. State-of-the-Art Algorithms

In the regression context, we adopted several state-of-the-art models for comparison purposes, which are implemented in the R programming language. The selected algorithms include: linear regression (LREG) from the `nnet` package, support vector regression (SVM) from the `e1071` package, k-nearest neighbors (kNN) from the `class` package, random forest (RF) from the `randomForest` package and multilayer perceptron (MLP) from the `monmlp` package.

Associative memories [18] are powerful tools for handling pattern completion tasks that involve several dependent variables simultaneously. As a representative of these networks we have selected the Hopfield model [14] since its architecture is reasonably similar to the one discussed in this paper, i.e. each neuron has a well-defined meaning for the prediction problem. Likewise, we compare our method against an FCM using sigmoid neurons where weights are confined to the $[-1, 1]$ interval. The kNN algorithm was also selected as it can be easily adapted to perform pattern completion tasks.

6.3. Parameter Settings

For each algorithm, hyperparameter tuning using grid search has been performed. By doing so, each dataset is split into three data pieces: the validation set, the test set and the training set. The best parameter configuration is selected by using the validation set, which is kept aside when training the model. The results reported in this paper are obtained from running the model on a test set, different from the validation set but also unknown during the learning phase, using the optimal parameters according to the validation set. Table 1 displays the parameter values considered during the grid search.

It should be stated that we couple the above-explained hyperparameter tuning scheme with a 5-fold cross validation process. With the dataset randomly split into 5 folds of equal size, each combination of 4 folds is used as the training set. Half of the remaining fold is randomly selected to create a validation set, the other half is used for testing. Finally, we report the average across all folds. For the sake of fairness, the same folds have been used for all method and configurations. This validation scheme is described in Algorithm 1.

Table 1: Parameter values used in the cross search.

Method	Parameter	Values
SVM	kernel	*linear*, *polynomial*, *radial*, *sigmoid*
	γ	$1/(attributes)$
kNN	k	1,3,5
RF	number of trees	300, 500, 800, 1000
MLP	learning rate	0.01, 0.1, 0.3, 0.5, 0.8
	momentum	0, 0.001, 0.01, 0.05, 0.1
FCM	sigmoid slope (λ)	1.0, 3.0, 5.0
STCN	learning rate (α)	0.001, 0.003, 0.005, 0.008, 0.01
	momentum (β)	0.8, 0.82, 0.85, 0.88, 0.9

Algorithm 1 Parameter tuning procedure

```
1: Randomly split dataset into 5 folds
2: for each parameter configuration cf: do
3:     for each fold pt: do
4:         Train model on remaining folds (not pt)
5:         Split fold f in validation set and test set
6:         Calculate validation error
7:         Calculate test error
8:     end for
9:     Calculate the average validation error
10:    Calculate the average test error
11: end for
12: Determine which configuration bs reported the lowest average validation error
13: Return average test error with configuration bs
```

Aiming at estimating the FCM weight matrix, we adopt an evolutionary learning approach based on an elitist Real-Coded Genetic Algorithm [31]. The required parameters are fixed as follows: the mutation probability is set to 0.1, the crossover probability is set to 0.9, the number of generations is set to 100, whereas the number of chromosomes is equal to the number of weight parameters to be estimated.

6.4. Dataset Characterization

Aiming at evaluating the accuracy of the proposed neural system, we have selected 35 datasets from the study reported by Nápoles et al. [23]. Table 2 outlines the main features of such datasets where the number of attributes ranges from 3 to 22, the number of decision classes from 2 to 8, whereas the number of instances goes from 106 to 625. Similarly to the proof of concept, we remove the decision class since our goal is to predict the value of numerical attributes.

Observe that our experiments are devoted to rather small datasets for the following reasons: (i) to keep the simulation time low, and (ii) to better resemble the real-world simulation scenarios on which domain experts are able to provide a limited number of training examples.

Table 2: Datasets used during simulations.

ID	Dataset	Instances	Attributes	Noisy
1	acute-inflammation	120	6	no
2	acute-nephritis	120	6	no
3	appendicitis	106	7	no
4	balance-noise	625	4	yes
5	balance-scale	625	4	no
6	blood	748	4	no
7	echocardiogram	131	11	no
8	ecoli	336	7	no
9	glass	214	9	no
10	glass-10an-nn	214	9	yes
11	glass-20an-nn	214	9	yes
12	glass-5an-nn	214	9	yes
13	haberman	306	3	no
14	hayes-roth	160	4	no
15	heart-5an-nn	270	13	yes
16	heart-statlog	270	13	no
17	iris	150	4	no
18	iris-10an-nn	150	4	yes
19	iris-20an-nn	150	4	yes
20	iris-5an-nn	150	4	yes
21	liver-disorders	345	6	no
22	monk-2	432	6	no
23	new-thyroid	215	5	no
24	parkinsons	195	22	no
25	pima	768	8	no
26	pima-10an-nn	768	8	yes
27	pima-20an-nn	768	8	yes
28	pima-5an-nn	768	8	yes
29	planning	182	12	no
30	saheart	462	9	no
31	tae	151	5	no
32	vertebral2	310	6	no
33	vertebral3	310	6	no
34	wine	178	13	no
35	wine-5an-nn	178	13	yes

* From the study by Nápoles et al. [23], we have discarded the problems nominal attributes or those involving class imbalance since they are described by the same attributes, thus resulting in 35 problems.

Table 3: MSE achieved by each regression algorithm. The best-performing algorithm for each method is highlighted in boldface.

ID	STCN	MLP	LREG	RF	SVM	kNN
1	0.0661	**0.0348**	0.0402	0.0786	0.0487	0.0579
2	0.0751	0.0501	**0.0457**	0.0851	0.0914	0.0944
3	0.0155	0.0051	**0.0013**	0.0083	0.0036	0.0117
4	**0.1255**	0.1258	0.1349	0.1476	0.1278	0.2867
5	**0.1257**	0.1261	0.1338	0.1431	0.1288	0.3037
6	0.0103	0.0100	**0.0081**	0.0097	0.0100	0.0160
7	**0.0468**	0.0565	0.1195	0.0565	0.0583	0.0866
8	**0.0308**	0.0381	0.0472	0.0354	0.0355	0.0663
9	0.0121	0.0095	**0.0068**	0.0113	0.0064	0.0177
10	**0.0196**	0.0288	0.1620	0.0248	0.0268	0.0421
11	**0.0375**	0.0494	0.0792	0.0454	0.0514	0.0825
12	**0.0179**	0.0250	0.0889	0.0235	0.0249	0.0353
13	0.0506	**0.0500**	0.0528	0.0566	0.0509	0.0898
14	**0.1260**	0.1270	0.1446	0.1330	0.1311	0.2269
15	**0.0892**	0.1039	0.2043	0.1092	0.1144	0.1728
16	**0.0809**	0.1015	0.1754	0.1017	0.1071	0.1538
17	0.0128	**0.0083**	0.0088	0.0090	0.0084	0.0136
18	**0.0215**	0.0339	0.0436	0.0305	0.0340	0.0482
19	**0.0270**	0.0435	0.0540	0.0442	0.0427	0.0746
20	**0.0150**	0.0246	0.0311	0.0232	0.0247	0.0303
21	**0.0119**	0.0139	0.0173	0.0142	0.0141	0.0264
22	**0.1898**	0.1904	0.2541	0.2116	0.2735	0.5284
23	**0.0083**	0.0135	0.0265	0.0116	0.0129	0.0193
24	0.0185	0.0070	0.0158	0.0069	**0.0068**	0.0125
25	**0.0152**	0.0187	0.0228	0.0177	0.0182	0.0316
26	**0.0266**	0.0358	0.0380	0.0347	0.0369	0.0628
27	**0.0369**	0.0453	0.0511	0.0451	0.0475	0.0838
28	**0.0196**	0.0290	0.0302	0.0273	0.0292	0.0496
29	0.0127	0.0044	**0.0022**	0.0091	0.0023	0.0189
30	**0.0412**	0.0461	0.0517	0.0462	0.0505	0.0807
31	**0.0619**	0.0883	0.3430	0.0709	0.1037	0.1007
32	0.0093	**0.0053**	0.0066	0.0085	0.0066	0.0120
33	0.0105	0.0067	**0.0062**	0.0095	0.0074	0.0132
34	**0.0166**	0.0191	0.0545	0.0171	0.0175	0.0297
35	**0.0211**	0.0281	0.0505	0.0248	0.0249	0.0417
AVR	**0.0430**	0.0458	0.0729	0.0495	0.0508	0.0864

6.5. Discussion: Multiple Regression

Table 3 displays the Mean Squared Error (MSE) achieved by each algorithm across selected datasets, after performing a 5-fold cross-validation process. It should be mentioned that, in the case of the state-of-the-art regression methods, we report the average MSE across several independent models (i.e., one per each dependent variable) since these algorithms allow to predict a single variable at each time. Therefore, each variable in the dataset will be used one time as the dependent variable, and N-1 times as an independent one.

The numerical results indicate that our model is the best-performing technique followed by MLP. It is remarkable the superiority of our model with respect to these well-established algorithms, even when they build an independent regression model per each dependent variable to be predicted. Or perhaps, being capable of approximating the value of multiple variables using a single model eventually becomes a key piece towards producing smaller simulation errors.

Aiming at exploring whether the differences in algorithms' performance are statistically significant or not, we rely on the Friedman two-way analysis of variances by ranks [7]. This test advocates for the rejection of the null hypothesis (p-value = 1.0547E-14 < 0.05) for a confidence interval of 95%, hence we can conclude that there are significant differences between at least two models across datasets.

As a second step, we perform a pairwise significance analysis using STCNs as the control method. With goal in mind, we resorted to the Wilcoxon signed rank test [32] and post-hoc procedures to adjust the p-values instead of using mean-ranks approaches, as suggested by Benavoli et al. [1]. Table 4 reports the unadjusted p-value computed by the Wilcoxon test and the corrected p-values associated with each pairwise comparison. In this paper, we assume that a null hypothesis H_0 can be rejected for a certain confidence level if at least two post-hoc procedures supports the rejection.

Table 4: Pairwise analysis using the STCN algorithm as the control method (multiple regression setting).

Algorithm	p-value	Bonferroni	Holm	Holland
kNN	1.352E-6	6.763E-6	6.763E-6	6.763E-6
RF	0.000118	5.927E-4	4.742E-4	4.741E-4
LREG	0.000582	0.002912	0.001747	0.001746
SVM	0.005928	0.029643	0.011857	0.011822
MLP	0.045688	0.228444	0.045688	0.045688

The statistical analysis reveals that the proposed algorithm is superior in performance to LREG, kNN, RF, MLP and SVM since both Holm and Holland suggest to reject the null hypotheses for a 95% confidence interval. Conclusions derived from Bonferroni's test are consistent with Holm and Holland for all methods except for MLP. These results evidence the capability of our algorithm to produce high-quality predictions with less effort, i.e. without the need of building a separate model for each dependent variable. In spite of that, it should be highlighted that the main contribution attached to our proposal is the flexible reasoning scheme that allows including expert knowledge into the network structure.

6.6. Discussion: Associative Memories

For this experiment, we modify the datasets listed in Table 3 so that each record has probability 0.2 to be corrupted. However, flexible reasoning goes beyond human-in-the-loop, since it refers to the use of any knowledge possible to enhance the model's performance. For example, when completing a corrupted pattern, we could initialize the neurons representing missing positions with estimations of the expected values. In the following experiment, those neurons are initialized with the average value of the corresponding attribute.

Table 5 shows the performance of each associative memory after performing a 5-fold cross-validation. The results show that our proposal is capable of outperforming the other algorithms, while Hopfield reports the higher prediction errors. This could be a result of using a Hebbian learning procedure over datasets with rather small patterns.

The Friedman test confirms that there are significant differences in the algorithms' performance (p-value=1.6892E-18 $<$ 0.05) for a confidence interval of 95%. Table 6 depicts the p-value obtained with the Wilcoxon signed rank test and the adjusted p-values. As in Table 4, the methods used to correct the o-values are Bonferroni, Holm and Holland. All post-hoc procedures advice rejecting the conservative hypothesis for a significance level of 0.05, thus corroborating the superiority of the STCN algorithm in this experiment.

Overall, the results have shown that STCNs are a convenient simulation and prediction model in terms of simulation error. Remark that STCNs are perfectly suited to deal with static data since abstract layers resulting from the iterations act as a regularizer. However, the forecasting of times series with STCNs emerges as the most logical step towards exploring the potential behind the proposed neural system.

Table 5: MSE achieved by each associative memory. The best-performing algorithm for each method is highlighted in boldface.

ID	STCN	FCM	kNN	Hopfield
1	0.1574	0.2006	**0.0588**	0.2589
2	0.1888	0.2050	**0.1144**	0.2567
3	**0.0176**	0.0429	0.0255	0.0619
4	**0.1123**	0.1256	0.1885	0.2433
5	**0.1162**	0.1218	0.1628	0.2375
6	0.0204	0.0763	**0.0127**	0.0615
7	**0.0473**	0.0908	0.0819	0.1195
8	**0.0362**	0.0618	0.0399	0.1312
9	**0.0180**	0.0527	0.0290	0.0696
10	**0.0282**	0.0520	0.0399	0.0659
11	**0.0390**	0.0631	0.0588	0.0728
12	**0.0256**	0.0597	0.0321	0.0719
13	**0.0451**	0.0567	0.0699	0.1306
14	**0.1023**	0.1283	0.1335	0.2079
15	**0.0999**	0.1171	0.1389	0.2420
16	**0.0964**	0.1215	0.1134	0.1976
17	0.0210	0.0581	**0.0146**	0.1192
18	0.0322	0.0491	**0.0286**	0.1583
19	**0.0343**	0.0605	0.0430	0.1413
20	**0.0243**	0.0784	0.0379	0.1535
21	**0.0130**	0.0410	0.0173	0.0934
22	**0.1832**	0.2061	0.3907	0.2251
23	**0.0099**	0.0903	0.0193	0.1075
24	0.0162	0.0702	**0.0149**	0.1220
25	**0.0220**	0.0564	0.0263	0.0626
26	**0.0372**	0.0604	0.0403	0.0747
27	**0.0444**	0.0654	0.0707	0.0768
28	**0.0315**	0.0516	0.0346	0.0778
29	**0.0224**	0.0324	0.0274	0.1904
30	**0.0430**	0.0760	0.0559	0.0699
31	**0.0776**	0.0933	0.0918	0.1911
32	**0.0106**	0.0453	0.0152	0.0906
33	**0.0121**	0.0295	0.0137	0.0832
34	**0.0218**	0.0518	0.0283	0.1470
35	**0.0277**	0.0528	0.0308	0.1505
AVR	0.0524	0.0813	0.0658	0.1361

Table 6: Pairwise analysis using the STCN algorithm as the control method (associative memory setting).

Algorithm	p-value	Bonferroni	Holm	Holland
Hopfield	2.477E-7	7.431E-7	7.431E-7	7.431E-7
FCM	2.477E-7	7.431E-7	7.431E-7	7.431E-7
kNN	0.000456	0.001369	4.563E-4	4.563E-4

7. Concluding Remarks

In this paper, we have presented a neural system —referred to as *Short-term Cognitive Networks* — for regression and pattern completion. This model allows performing simulations on the basis of previously defined knowledge structures, where weights may have a causal meaning or not. Aiming at preserving the initial knowledge, we developed a nonsynaptic learning algorithm that relies on the gradient descent to reduce the error without altering the weights. Likewise, we have derived a stopping criterion to prevent the learning method from iterating without decreasing the simulation error.

The statistical analysis reported that the STCN model is superior (in terms of simulation error) to most regression models adopted for comparison. Moreover, our proposal allows making predictions without the need of building a separate model for each decision variable to be forecasted. In the context of associative memories, the results support the superiority of our model over both FCMs and Hopfield.

While our proposal is accurate in terms of simulation errors, it brings to life a sensitive problem: the short-term reasoning mechanism is not able to capture the dynamics governing the system under analysis. Therefore, another point that should be studied is how to expand the STCN memory without affecting the prediction rates, which would allow capturing the dynamic patterns attached to the physical system. But if an explicit long-term memory brings to life the limitations inherent to classic FCMs, then it would be a high price to be paid on behalf of patterns that experts rarely exploit in practice.

Although this research focused on providing an accurate alternative for traditional FCMs used in simulation scenarios, the STCN model can certainly be expanded to other domains. Time series forecasting or pattern classification based on the flexible reasoning paradigm are open problems that deserve attention. Another interesting challenge that may be explored in futures studies is how to efficiently fine-tune the network parameters in presence of time-varying pieces of data. Likewise, how to handle the uncertainty introduced by experts during the modeling stage is deemed pivotal towards designing a robust, still flexible, neural reasoning model.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable and constructive feedback.

References

[1] Benavoli, A., Corani, G., Mangili, F., 2016. Should we really use post-hoc tests based on mean-ranks? Journal of Machine Learning Research 17, 1–10.

[2] Bengio, Y., 2012. Practical Recommendations for Gradient-Based Training of Deep Architectures. Springer. pp. 437–478.

[3] Carvalho, J.P., 2010. On the semantics and the use of fuzzy cognitive maps in social sciences, in: International Conference on Fuzzy Systems, pp. 1–6.

[4] Carvalho, J.P., 2013. On the semantics and the use of fuzzy cognitive maps and dynamic cognitive maps in social sciences. Fuzzy Sets and Systems 214, 6–19.

[5] Concepción, L., Nápoles, G., Vanhoof, K., Bello, R., 2019. Unveiling the dynamic behavior of fuzzy cognitive maps. IEEE Transactions on Fuzzy Systems (submitted) .

[6] Felix, G., Nápoles, G., Falcon, R., Froelich, W., Vanhoof, K., Bello, R., 2017. A review on methods and software for fuzzy cognitive maps. Artificial Intelligence Review doi:10.1007/s10462-017-9575-1.

[7] Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the american statistical association 32, 675–701.

[8] Froelich, W., Pedrycz, W., 2017. Fuzzy cognitive maps in the modeling of granular time series. Knowledge-Based Systems 115, 110–122.

[9] Froelich, W., Salmeron, J.L., 2017. Advances in fuzzy cognitive maps theory. Neurocomputing , 1—-2.

[10] Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.

[11] Harmati, I.Á., Hatwágner, M.F., Kóczy, L.T., 2018. On the existence and uniqueness of fixed points of fuzzy cognitive maps, in: Medina, J., Ojeda-Aciego, M., Verdegay, J.L., Pelta, D.A., Cabrera, I.P., Bouchon-Meunier, B., Yager, R.R. (Eds.), Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations, Springer. pp. 490–500.

[12] Homenda, W., Jastrzebska, A., 2017. Clustering techniques for Fuzzy Cognitive Map design for time series modeling. Neurocomputing 232, 3–15.

[13] Homenda, W., Jastrzebska, A., Pedrycz, W., 2014. Time series modeling with fuzzy cognitive maps: Simplification strategies, in: Saeed, K., Snášel, V. (Eds.), Computer Information Systems and Industrial Management, Springer. pp. 409–420.

[14] Hopfield, J.J., 1984. Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences 81, 3088–3092.

[15] Knight, C.J., Lloyd, D.J., Penn, A.S., 2014. Linear and sigmoidal fuzzy cognitive maps: an analysis of fixed points. Applied Soft Computing 15, 193–202.

[16] Kosko, B., 1986. Fuzzy cognitive maps. International Journal Man-Machine Studies 24, 65–75.

[17] Kosko, B., 1988. Hidden patterns in combined and adaptive knowledge networks. International Journal of Approximate Reasoning 2, 377–393.

[18] Krikelis, A., 1996. Associative Processing and Processors. 1st ed., IEEE Computer Society Press, Los Alamitos, CA, USA.

[19] Lu, W., Yang, J., Liu, X., Pedrycz, W., 2014. The modeling and prediction of time series based on synergy of high-order fuzzy cognitive map and fuzzy c-means clustering. Knowledge-Based Systems 70, 242–255.

[20] McCulloch, W.S., Pitts, W., 1988. A logical calculus of the ideas immanent in nervous activity, in: Anderson, J.A., Rosenfeld, E. (Eds.), Neurocomputing: Foundations of Research. MIT Press, pp. 15–27.

[21] Nápoles, G., Concepción, L., Falcon, R., Bello, R., Vanhoof, K., 2017. On the accuracy-convergence tradeoff in sigmoid fuzzy cognitive maps. IEEE Transactions on Fuzzy Systems .

[22] Nápoles, G., Grau, I., Papageorgiou, E., Bello, R., Vanhoof, K., 2016a. Rough cognitive networks. Knowledge-Based Systems 91, 46–61.

[23] Nápoles, G., Mosquera, C., Falcon, R., Grau, I., Bello, R., Vanhoof, K., 2018. Fuzzy-rough cognitive networks. Neural Networks 97, 19–27.

[24] Nápoles, G., Papageorgiou, E., Bello, R., Vanhoof, K., 2016b. On the convergence of sigmoid Fuzzy Cognitive Maps. Information Sciences 349, 154–171.

[25] Nápoles, G., Papageorgiou, E., Bello, R., Vanhoof, K., 2017. Learning and convergence of fuzzy cognitive maps used in pattern recognition. Neural Processing Letters 45, 431–444.

[26] Pedrycz, W., Jastrzebska, A., Homenda, W., 2016. Design of fuzzy cognitive maps for modeling time series. IEEE Transactions on Fuzzy Systems 24, 120–130.

[27] Poczeta, K., Kubuś, Ł., Yastrebov, A., 2019. Structure Optimization and Learning of Fuzzy Cognitive Map with the Use of Evolutionary Algorithm and Graph Theory Metrics. Springer. pp. 131–147.

[28] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1, MIT Press, Cambridge, MA, USA. chapter Learning Internal Representations by Error Propagation, pp. 318–362.

[29] Salmeron, J., Mansouri, T., Moghadam, M., Mardani, A., 2019. Learning fuzzy cognitive maps with modified asexual reproduction optimisation algorithm. Knowledge-Based Systems 163, 723–735.

[30] Salmeron, J., Palos-Sanchez, P., 2019. Uncertainty propagation in fuzzy grey cognitive maps with hebbian-like learning algorithms. IEEE Transactions on Cybernetics 49, 211–220.

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

[32] Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometrics 1, 80—-83.

[33] Yang, Z., Liu, J., 2019. Learning of fuzzy cognitive maps using a niching-based multi-modal multi-agent genetic algorithm. Applied Soft Computing Journal 74, 356–367.

[34] Zou, X., Liu, J., 2018. A mutual information-based two-phase memetic algorithm for large-scale fuzzy cognitive map learning. IEEE Transactions on Fuzzy Systems 26, 2120–2134.