International Journal on Artificial Intelligence Tools
Vol. 27, No. 7 (2018) 1860010 (25 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218213018600102

FCM Expert: Software Tool for Scenario Analysis and Pattern Classification Based on Fuzzy Cognitive Maps

Gonzalo Nápoles[*,§], Maikel Leon Espinosa[†], Isel Grau[‡] and Koen Vanhoof[*]
[*]*Faculty of Business Economics, Universiteit Hasselt, Belgium*
[†]*Department of Business Technology, University of Miami Business School, USA*
[‡]*Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Belgium*
[§]*napoles.gonzalo@gmail.com*
[†]*mleon@bus.miami.edu*

Received 21 May 2018
Accepted 11 September 2018
Published 14 November 2018

Fuzzy Cognitive Maps (FCMs) have become a suitable and proven knowledge-based methodology for systems modeling and simulation. This technique is especially attractive when modeling systems characterized by ambiguity, and/or non-trivial causalities among its variables. The rich literature that is found related to FCMs reports very clearly many successful studies solved through the use of FCMs; however, when it comes to software implementations, where domain experts can design FCM-based systems, run simulations or perform more advanced experiments, not much is found or documented. The few existing implementations are not proficient in providing options for experimentation. Therefore, we believe that a gap exists, specifically between the theoretical advances and the development of accurate, transparent and sound FCM-based systems; and we advocate for the creation of more complete and flexible software products. The goal of this paper is to introduce "FCM Expert", a software tool for fuzzy cognitive modeling, where we focus on scenario analysis and pattern classification. The main features of FCM Expert rely on Machine Learning algorithms to compute the parameters that might define a model, optimize its network topology and improve the system convergence without losing information. Also, FCM Expert allows performing WHAT-IF simulations and studying the system behavior through a friendly, intuitive and easy-to-use graphical user interface.

Keywords: Fuzzy cognitive maps; software tool; scenario analysis; pattern classification; machine learning algorithms.

1. Introduction

Fuzzy Cognitive Maps (FCMs) were presented by B. Kosko as a knowledge-based methodology for modeling and simulating dynamic systems.[1] FCMs are in fact, a kind of combination of fuzzy logic, neural networks and cognitive mapping, serving as a way to represent knowledge of systems that are characterized by uncertainty, causality and complex processes. From a structural point of view, an FCM may be represented by fuzzy directed digraphs with feedback, seen as a collection of neural processing units and signed weighted relations. Using this methodology, a system could conveniently be modeled in terms of concepts (e.g., variables, objects or entities) and causal relations between these concepts. Each concept is characterized by its activation degree, which denotes to what extent this variables influences the others. The fuzzy approach allows us to have degrees of causality, represented as links between the concepts.[2] The fuzzy nature of FCMs is confined to the network construction phase where experts define the causal relations using linguistic terms. After that, no explicit fuzzy operations are used.

Since FCMs allow feedback in their connections, we can explore the system dynamics by describing the effect of specific changes over the whole causal network. Therefore, during the inference phase, the FCM calculates the activation value of all concepts at each discrete-time step according to the standard McCulloch-Pitts model.[3] After a number of discrete-time steps, an FCM may arrive to three possible states: a fixed-point, a cyclic state or a totally chaotic behavior. The former scenario implies that a hidden pattern was discovered,[4] while the last ones suggest that the FCM is unable to confidently recognize the target pattern. However, in scenarios devoted to time series forecasting, the convergence to a fixed-point attractor becomes a serious drawback since the FCM-base forecaster is unable to fit the expected values during time.

FCMs have received increasing attention among researchers and both practical and theoretical results have been introduced. Some representative application fields include: decision making,[5] system control,[6] engineering,[7] protein modeling,[8] transport management,[9] intrusion detection,[10] etc. Also, Papakostas *et al.*[11] introduced FCM-based classifiers as *light grey box models*, being used for classification tasks. In order to construct an accurate FCM-based classifier from historical data, the estimation of several parameters is required, and we believe in the power of Machine Learning to do so. This opens up the need for a suitable software platform to execute loads of work. Nevertheless, the scientific literature shows just a few software products capable of drawing FCMs and performing very simple simulation tasks, and these tools cannot be used for solving pattern classification problems anyway due to the absence of experimentation facilities.

With the goal of filling this important gap, in this paper, we present a Java software tool that allows designing, learning and simulating FCM-based systems. FCM Expert extends a previous specific-purpose software tool called FCM Tool, which was developed by León *et al.*[12] to address a decision-making problem concerning public transportation in Belgium (2008–2012). The key advantages of FCM Expert rely on the inclusion of several experimentation facilities and Machine Learning algorithms, which are supported by a friendly visual interface. Overall, the most attractive features of FCM Expert can be concisely summarized as follows:

- Experimentation options to configure the FCM model and perform WHAT-IF simulations for analyzing hypothetical scenarios.

- The possibility to model pattern classification problems by using different architectures.[11] To encourage the compatibility with other Machine Learning software, FCM Expert uses the well-known Attribute Relation File Format (ARFF) to handle historical data.
- The inclusion of supervised and unsupervised Machine Learning algorithms to estimate the weight set, optimize the network topology without losing relevant information and improve the convergence of the FCM-based system being modeled. These algorithms rely on population-based heuristic search methods, capable of computing near-optimal solutions in a reasonable time, thus ignoring analytically properties of the error function such as continuity, convexity or differentiability.
- Several visualization options oriented to model, adjust and exploit the FCM-based system. Some of such options include real-time visualization of the learning progress (error minimization), analysis of fixed-point attractors and graphical simulations of new virtual scenarios to support the decision-making process.

The rest of this paper is organized as follows: In Section 2 we briefly formalize the mathematical theory behind FCM-based systems, whereas in Section 3 we examine existing software tools related to FCM-based modeling. In Section 4, we present the main functions and general architecture of FCM Expert. Sections 5–7 describe the algorithms implemented into FCM Expert, as the most advanced features of FCM Expert. Section 8 explores the FCM Expert facilities through a specific case study while Section 9 provides concluding remarks and future research and implementation goals.

2. Fuzzy Cognitive Maps

In general terms, FCMs can be understood as recurrent neural networks with interpretable features that have been widely used in modeling tasks.[1] They consist of a set of neural processing entities called concepts (neurons) and the causal relations among them. The activation value of such neurons regularly takes values in the $[0, 1]$ interval, so the stronger the activation value of a neuron, the greater its impact on the network. Also, connected weights are relevant in this scheme. The strength of the causal relation between two neurons C_i and C_j is quantified by a numerical weight $w_{ij} \in [-1, 1]$ and denoted via a causal edge from C_i to C_j.

There are three types of causal relationships between neural units in an FCM, being detailed as follows:

- If $w_{ij} > 0$ then there is a *positive causality*, an increase (decrement) on C_i produces an increment (decrement) on C_j with intensity $|w_{ij}|$.
- If $w_{ij} < 0$ then there is a *negative causality*, an increase (decrement) on C_i produces an decrement (increment) on C_j with intensity $|w_{ij}|$.
- If $w_{ij} = 0$ then there is no causal relation.

Equation (1) formalizes Kosko's activation rule, with $A^{(0)}$ as the initial state. A new activation vector is calculated at each step t and after a fixed number of iterations the FCM will be at one of the following states: (i) equilibrium point, (ii) limited cycle or (iii) chaotic behavior.[4] The FCM is said to have converged if it reaches a fixed-point attractor, otherwise the updating process terminates after a maximum number of iterations T is reached.

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

In the above rule, $f(\cdot)$ denotes a monotonically non-decreasing function to clamp the activation value of each concept to the allowed interval. The functions most extensively used based on literature are depicted as follows:

- *The bivalent function*

$$f_1(x) = \begin{cases} 1\,, & x > 0 \\ 0\,, & x \leq 0 \end{cases} \tag{2}$$

- *The saturation function*

$$f_2(x) = \begin{cases} 0\,, & x \leq 0 \\ x\,, & 0 < x < 1 \\ 1\,, & x \geq 1 \end{cases} \tag{3}$$

- *The trivalent function*

$$f_3(x) = \begin{cases} -1\,, & x < 0 \\ 0\,, & x = 0 \\ 1\,, & x > 0 \end{cases} \tag{4}$$

- *The hyperbolic function*

$$f_4(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{5}$$

- *The sigmoid function*

$$f_5(x) = \frac{1}{1 + e^{-\lambda(x-h)}} \tag{6}$$

Equation (1) shows an inference rule widely used in many FCM-based applications, but it is not the only one possible. Stylios and Groumpos[13] proposed a modified inference rule, found at Eq. (7), where neurons also take into account its own past value. This rule is preferred when updating the activation value of independent neurons, i.e., neurons that are not influenced by any other neural processing entities.

$$A_i^{(t+1)} = f\left(\sum_{\substack{j=1 \\ i\neq j}}^{M} w_{ji}A_j^{(t)} + A_i^{(t)}\right) \tag{7}$$

Another rule proposed in Ref. 14 is used to avoid the conflicts emerging in the case of non-active neurons. The re-scaled inference depicted in Eq. (8) allows dealing with scenarios where there is no information about an initial neuron-state and helps preventing the saturation problem.

$$A_i^{(t+1)} = f\left(\sum_{\substack{j=1 \\ i\neq j}}^{M} w_{ji}(2A_j^{(t)} - 1) + (2A_i^{(t)} - 1)\right) \tag{8}$$

If the network is able to converge, then the system will produce the same output towards the end, so the activation degree of neurons will remain unaltered (or subject to infinitesimal changes). Convergence is often desired in pattern classification and scenario analysis, whereas it becomes a serious problem when modeling time series problem.

3. Examining Software Tools Related to Fuzzy Cognitive Maps

Most of the FCM papers exhibit theoretical contributions or practical applications related to FCMs application, but less is usually found about well-defined software for handling FCM-based systems. Moreover, the existing software implementations fail in providing advanced options to adjust the model parameters. The absence of such features leads to a gap between the recent theoretical advances on FCMs research and the development of accurate and mathematically sound FCM-based systems. In this section, we review the most representative software tools for FCMs we have found.

We begin with FCM Modeler,[15] a desktop implementation aiming to model generic FCMs. It involves a simple graphical user interface offering support for group decision making on a qualitative static model. It intended to be a general modeling tool, but regrettably the project never evolved into that. The authors also advocated for including a basic Machine Learning algorithm for adjusting the weight set.

A similar approach, FCM Designer[16] was also found during our survey. This tool allows adapting the inference rule by selecting the transfer function and the stopping criterion. The key drawback relies on the lack of learning algorithms to compute the parameters that characterize the system. In spite of this limitation, several modeled scenarios were found using this software tool, including the use of FCM for simple tasks related to supervision and control. In a similar line, FCM-Analyst[17] facilitates the simulation and implementation of FCMs with basic drawing supported and options for configuring the transfer function. This feature was interesting as they offer an editor that is able to recognize equation supporting different functions.

FCM Tool[12] is a Java software tool that allows designing complex FCM-based models through an interactive graph visualization. It allows analyzing scenarios and customizing the FCM reasoning process. Likewise, FCM Tool provides a population-based learning algorithm based on Swarm Intelligence to learn the weight set from historical data. Another relevant feature is the inclusion of aggregation operators for combining several FCM-based systems into a single knowledge-based representation. As FCM Tool was designated to address a specific decision-making problem, this implies that its algorithms could not be used for solving more generic pattern classification applications. On the other hand, FCM Tool uses specific files for handling historical data, which are generated by an *Automated Knowledge Engineer* implementation, such features contribute to the lack of generality.

Another promising software tool recently proposed is Mental Modeler,[18] which comprises a web-based interface to support individuals and communities to capture their knowledge in a standardized format for scenario analysis. This software tool was developed to support group decision-making, allowing domain experts to collaboratively represent and test their assumptions about a system. Mental Modeler can be mainly used by non-IT people, usually experts or stakeholders in a given domain who need to design a simple cognitive map and simulate its behavior for some scenarios. The key drawbacks of this FCM implementation relies on the lack of learning methods and its limited experimenting options.

More recently, the Java Fuzzy Cognitive Maps (JFCM), an open source library for fuzzy cognitive mapping modeling was presented.[19] The library is small and simple, but can be used to create a variety of cognitive networks. The JFCM library allows loading networks from XML files, thus increasing its portability. The idea behind the library is to create reusable modules that could be used when needing FCM solutions in a given problem. Interacting with this library requires to reuse and modify the source code to model specific features, which becomes a limitation for non-expert users.

Finally, in our study we found the Intelligent Expert System based on Cognitive Maps (ISEMK)[20] that allows modeling decision support systems based on FCMs and neural networks. ISEMK includes a multi-step gradient learning algorithm and two evolutionary search methods (e.g., Real-Coded Genetic Algorithm) for adjusting the FCM model. Moreover, it includes two learning algorithms for multi-layer neural networks used in time series forecasting, while it supports the visualization of results through an adequate graphical interface. This software however is mostly focused on time series forecasting, which reduces its usability in more generic domains.

Table 1 provides a comparison among the revised software tools for several features such as area of application, experimentation facilities, inclusion of machine learning algorithms and graphical support.

Table 1. Comparison of existing software tools for Fuzzy cognitive maps.

		Main Features			
	Year	Experimentation Facilities	Learning Algorithms	Graphical Support	Application Domain
FCM Modeler	1997	Not at all	Only one	Poor	Ecosystem
FCM Designer	2005	Some, but not enough	Not at all	Adequate	Supervision, Control
Mental Modeler	2013	Some, but not enough	Not at all	Adequate	Social Sciences, Ecological Systems
JFCM	2013	Few, only for developers	Not at all	Not at all	Traffic Analysis
FCM Tool	2011	Many, but oriented to the domain	Only one	Adequate	Transportation Management

When comparing among the surveyed software tools for several features, we can conclude that FCM Designer, Mental Modeler, FCM-Analyst and FCM Tool provide to the experts an appropriate graphical support when analyzing scenarios and experimenting new situations; JFCM is suitable for developing FCM modules that could be reused in more complex solutions, while ISEMK resulted in the best implementation for time series forecasting. The surveyed software tools lack advanced algorithms and experimentation options, making this a strong motivation for introducing FCM Expert. Furthermore, none of these software products allow handling pattern classification problems, which reduce their usability when investigating new FCM-based solutions in this domain.

4. Architecture and Features of FCM Expert

As mentioned, FCM Expert is a software tool for designing FCM-based systems. This software is written in Java language and comprises more than 25 000 source code lines, which are distributed in 120 source files. These files are organized in five global packages (i.e., *Network*, *Algorithms*, *Learning*, *Software* and *Resources*) and several sub-packages. Figure 1 shows the structure of the packages in FCM Expert, including primary packages and some sub-packages.

In a rough picture, FCM Expert involves three groups of functions that are distributed in five menus: *File*, *Edit*, *Build*, *Run* and *Reset.* The first group is oriented to the design of the FCM-based model, where the expert (user) in a given domain can model a complex system (visual options do not require deep expertise in Mathematics or Computer Science). The second group comprises Machine Learning algorithms for adjusting the model parameters and optimizing its performance.

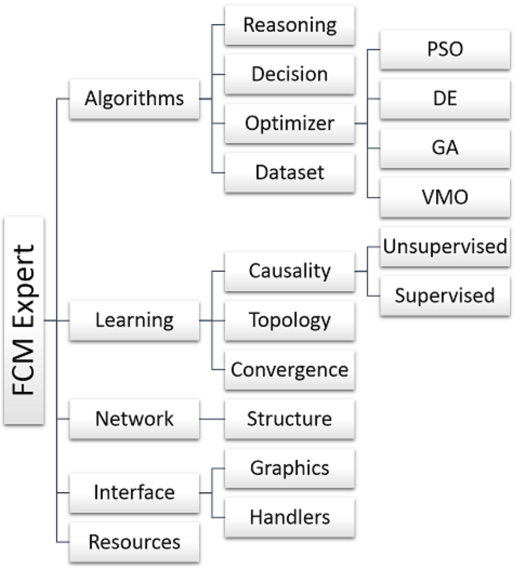


Fig. 1. Packages tree of the FCM Expert software tool.

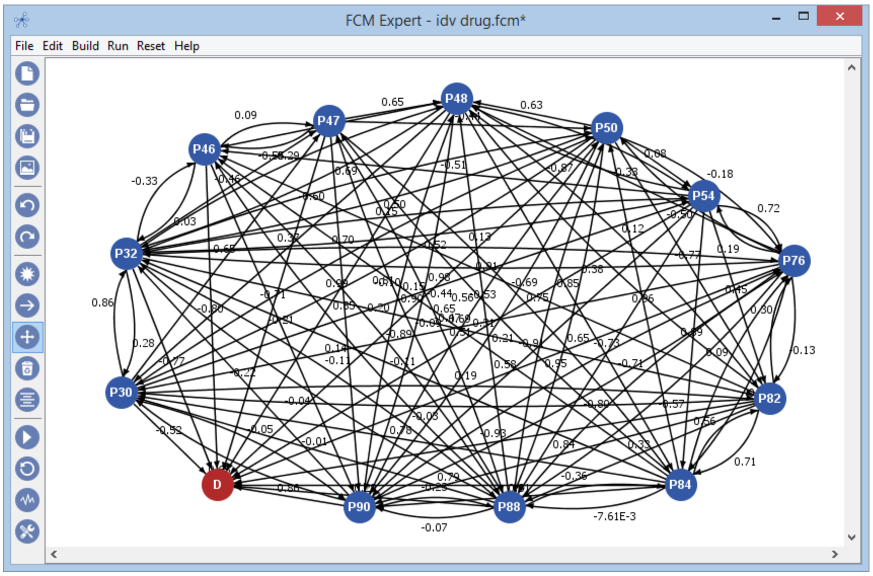
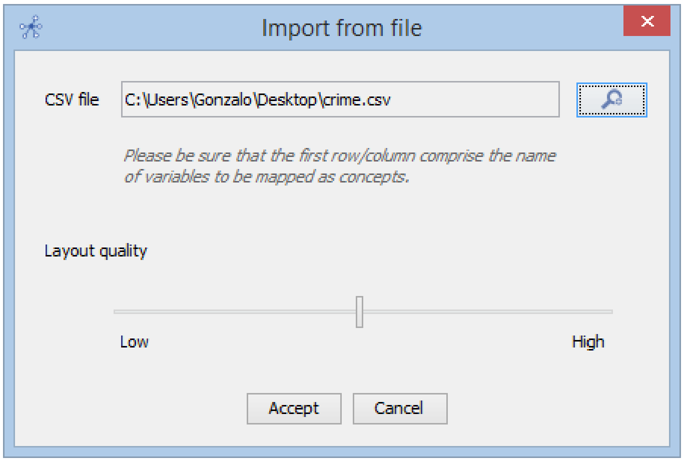


Fig. 2. Snapshot of the proposed FCM expert.

Finally, the third group includes procedures for exploiting the FCM-based system, as a tool for supporting decision-making processes. Figure 2 shows the main window of FCM Expert displaying a real case study concerning the resistance mechanism of HIV mutations to existing inhibitors.[21]

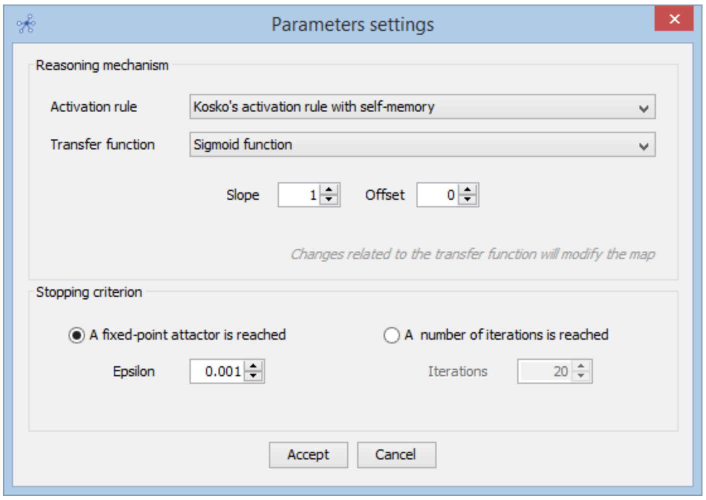


CSV file: C:\Users\Gonzalo\Desktop\crime.csv

Please be sure that the first row/column comprise the name of variables to be mapped as concepts.

Layout quality: Low — High

Fig. 3. Dialog to build an FCM-based network from a CSV file.



Reasoning mechanism

Activation rule: Kosko's activation rule with self-memory

Transfer function: Sigmoid function

Slope: 1 Offset: 0

Changes related to the transfer function will modify the map

Stopping criterion

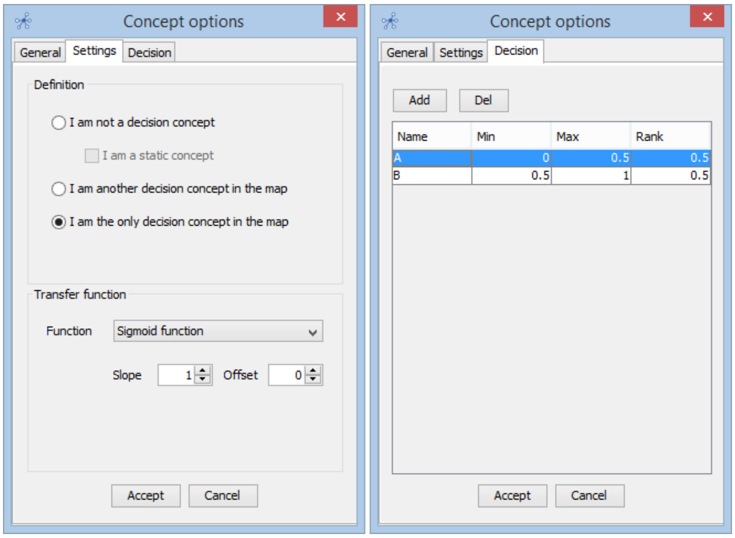
[✓ A fixed-point attactor is reached] A number of iterations is reached

Epsilon: 0.001 Iterations: 20

Fig. 4. Configuration of the FCM reasoning process (settings).

Also, FCM Expert allows designing an FCM-based system from scratch. This can be done by manually drawing the network structure or importing the weight matrix from a CSV file (see Fig. 3). This last option includes a heuristic layout procedure for efficiently drawing the network topology, which minimizes both the distance between concepts and the cuts between graph edges and concepts.

Other options however require more expertise as they were conceived for supporting the FCM research community. For example, Fig. 4 illustrates how to configure the parameters related to the FCM reasoning: *the inference rule*, *the transfer function* and *the stopping criterion*.



Definition

( ) I am not a decision concept

[ ] I am a static concept

( ) I am another decision concept in the map

(●) I am the only decision concept in the map

Transfer function

Function: Sigmoid function

Slope: 1 Offset: 0

Name	Min	Max	Rank
A	0	0.5	0.5
B	0.5	1	0.5

Fig. 5. Configuration of parameters for the selected concept.

The proposed software implements the three inference rules (see Eqs. (1), (7) and (8)) and the five transfer functions earlier mentioned (see Eqs. (2)–(6)). This suggests that the concepts can take values in $[-1, 1]$ or $[0, 1]$, providing flexibility during the modeling phase.

Unlike other tools, our software allows handling different architectures for both scenario analysis and pattern classification. In the first case, the FCM does not comprise a decision concept, whereas in the second case we implement two different FCM architectures for pattern classification defined in Ref. 22 that differ in the number of decision concepts.

The *single-output architecture* uses a single decision concept such that classes are defined as closed partitions of the decision space, while in the *class-per-output architecture*, each class is defined by an output neuron. Figure 5 shows how to specify the role of each neuron in an FCM-based network. Each neuron may use its own transfer function when updating its activation values. In this example, we show how to configure the decision table in a single-output architecture for a two-class (binary) classification problem.

FCM Expert allows performing WHAT-IF simulations by directly modifying the activation values of each concept and next running the inference process. This generates a plot and a table (see Fig. 6) with the activation value of each concept at each iteration-step for the specified stimulus.

Additionally, FCM Expert allows performing simulations in a visual mode where the size of each concept is determined according to its activation value. Other interesting feature is the aggregation of multiple FCM-based systems into a single knowledge-based structure. In the next sections, we describe the learning algorithms

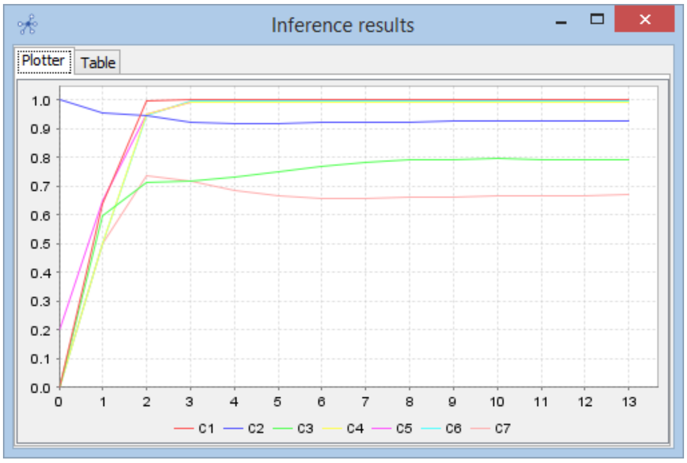
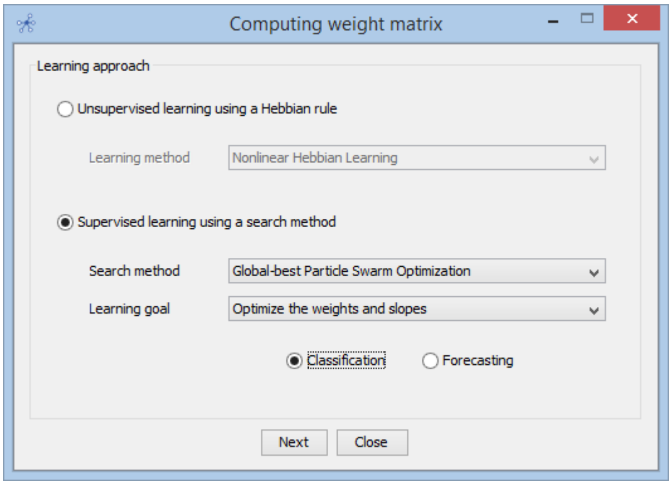


Fig. 6. Dialog summarizing the FCM inference process.

implemented, which comprise one of the main advantages over other software tools found in the surveyed literature.

5. Computing the FCM Parameters

Learning methods for computing the weight set are pivotal when designing an FCM-based system. The most prominent algorithms for FCM learning may be gathered into two large groups[23]: unsupervised and supervised models. Figure 7 displays how to select the learning approach in FCM Expert, as a first step towards learning the weight set.



Learning approach

( ) Unsupervised learning using a Hebbian rule

Learning method: Nonlinear Hebbian Learning

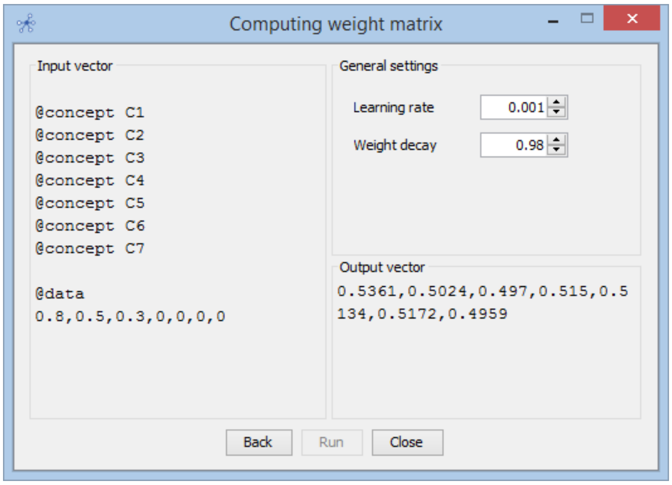
(●) Supervised learning using a search method

Search method: Global-best Particle Swarm Optimization

Learning goal: Optimize the weights and slopes

(●) Classification ( ) Forecasting

Fig. 7. Settings for the weight estimation algorithm.



Input vector

```
@concept C1
@concept C2
@concept C3
@concept C4
@concept C5
@concept C6
@concept C7

@data
0.8,0.5,0.3,0,0,0,0
```

General settings

Learning rate 0.001

Weight decay 0.98

Output vector

```
0.5361,0.5024,0.497,0.515,0.5
134,0.5172,0.4959
```

Fig. 8. Unsupervised learning of the weight matrix.

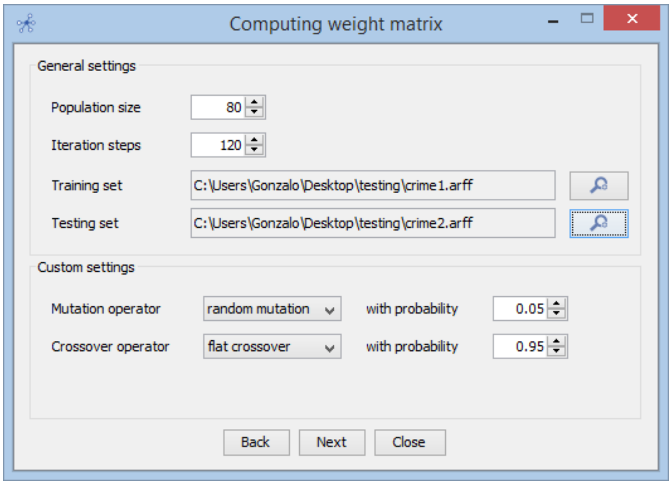
According to Ref. 23, the unsupervised learning algorithms are convenient to fine-tune the weight set with a small deviation from the initial configuration. However, these algorithms lack generalization capabilities, hence they are not advised when solving pattern classification problems; instead we may use heuristic (supervised) learning algorithms.

5.1. *Scenario analysis*

Unsupervised learning of FCM is based on the Hebbian law to iteratively adjust the causal weights by using a single representative pattern as training data. Hebbian-based learning was initially applied on the training of artificial neural networks, however, recently this approach has been successfully used for training FCM-based systems. The key feature of this learning rule is that the change of a synaptic is computed by taking into account the presynaptic and postsynaptic signals flow towards each neural processing unit.

The proposed software tool includes the following Hebbian-type algorithms: the *Differential Hebbian Learning*,[24] the *Balanced Differential Algorithm*,[25] the *Nonlinear Hebbian Learning*[26] and the *Improved Nonlinear Hebbian Learning*.[27] The expert must specify two parameters related to the weight decay and the learning rate. Figure 8 portrays graphical interface to capture the training example (i.e., an activation vector) and visualize the response vector obtained after adjusting the weights.

It should be mentioned that the adjusted weights partially preserve their physical meaning, which is often desired when performing WHAT-IF simulations. Of course, the requirement of experts' knowledge is a serious drawback. The flexibility on data requirements of these algorithms is the key aspect behind their poor generalization capability.


Computing weight matrix

General settings

Population size 80

Iteration steps 120

Training set C:\Users\Gonzalo\Desktop\testing\crime1.arff

Testing set C:\Users\Gonzalo\Desktop\testing\crime2.arff

Custom settings

Mutation operator random mutation with probability 0.05

Crossover operator flat crossover with probability 0.95

Fig. 9. Settings for the Real-Coded Genetic Algorithm.

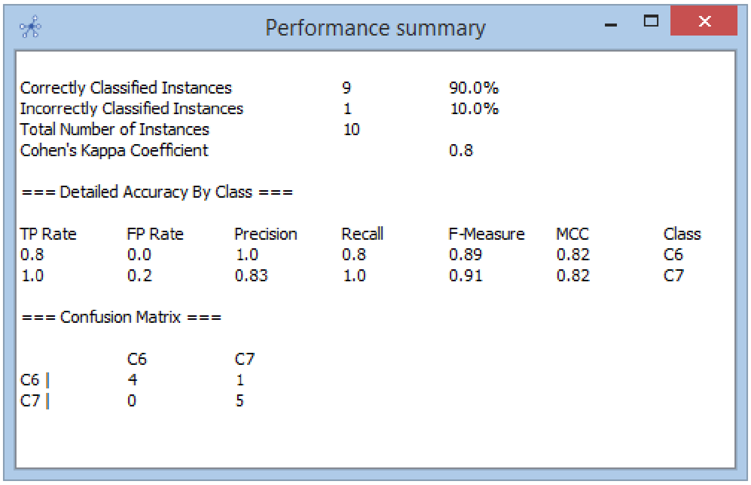
5.2. *Pattern classification*

In the context of FCM-based classifiers, the learning goal is to compute, in a supervised fashion, a weight matrix minimizing the dissimilarity between the expected outputs and the predicted ones. Unlike Hebbian-based procedures, supervised learning approaches use a set of training instances instead of using a single example. FCM Expert also allows optimizing the parameters attached to the sigmoid transfer function. The algorithm computes a custom transfer function for each concept hoping to increase the overall prediction rates. The software allows including expert knowledge related to causal relations during learning, therefore the resulting FCM model is no longer a black-box! The optimization process may be performed by using a variety of heuristic search methods.

More explicitly, we implemented several population-based optimizers including *Particle Swarm Optimization*,[28] *Differential Evolution*,[29] *Real-Coded Genetic Algorithm*[30] and *Variable Mesh Optimization*.[31] The advantage of using heuristic methods relies on their ability for estimating near-optimal solutions, therefore ignoring analytic (often unknown) properties of the error function to be optimized.

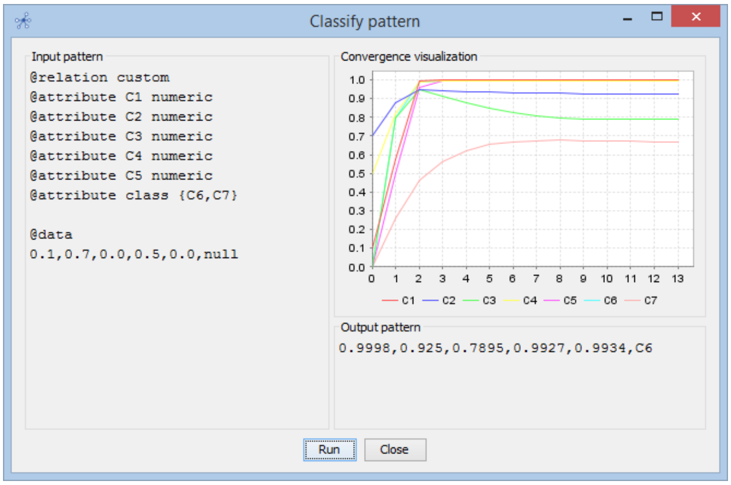
Figure 9 shows, as an illustrative example, how to configure the parameters related to the Real-Coded Genetic Algorithm. The expert may provide to the learning algorithm a separated testing set in order to evaluate the generalization capability of the learned model, which is a pivotal aspect in pattern classification scenarios. F-measure, accuracy, Kappa coefficient and the confusion matrix are some of the statistics used to assess the quality of the FCM-based classifier (see Fig. 10). Once the supervised learning process is completed, we can use the learned FCM model to determine the most likely decision class for unseen instances (see Fig. 11).

Recently, Nápoles *et al.*[32] surveyed the key advances and challenges on FCM-based classifiers. Although the road towards computing truly causal, interpretable



```
Correctly Classified Instances        9          90.0%
Incorrectly Classified Instances      1          10.0%
Total Number of Instances             10
Cohen's Kappa Coefficient                        0.8

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall   F-Measure   MCC    Class
0.8       0.0       1.0         0.8      0.89        0.82   C6
1.0       0.2       0.83        1.0      0.91        0.82   C7

=== Confusion Matrix ===

          C6        C7
C6 |      4         1
C7 |      0         5
```

Fig. 10. Statistics used to asses the quality of an FCM-based classifier.



```
@relation custom
@attribute C1 numeric
@attribute C2 numeric
@attribute C3 numeric
@attribute C4 numeric
@attribute C5 numeric
@attribute class {C6,C7}

@data
0.1,0.7,0.0,0.5,0.0,null
```

```
0.9998,0.925,0.7895,0.9927,0.9934,C6
```

Fig. 11. Classifying new instances with an FCM-based classifier.

FCM classifiers is still narrow, we believe that FCM Expert comprises useful options for supporting new researches in this field.

6. Optimizing the Network Topology

Sometimes, we need to handle FCMs with very complex network structure as a result of modeling a physical system comprised of a large number of variables. In these situations, some concepts/variables could be superfluous or even contradictory due to the uncertainty and subjectivity attached to human reasoning and modeling, thus negatively affecting both the system performance and interpretability.

Aiming at overcoming this issue, Nápoles *et al.*[21] proposed a reduction algorithm for optimizing the network topology on FCM-based systems, without losing relevant information. The learning goal of this algorithm is to find the minimal subset of concepts capable of preserving, in some extent, the original system performance. Since we are mainly focused on FCM-based systems used in pattern classification scenarios, the performance could be measured as the number of patterns positively recognized by the system under investigation. Finding this minimal subset involves a difficult combinatorial problem defined by a search space comprised of 2^{M-1} solutions, with M being the number of concepts in the network.

The proposed implementation offers the possibility to select and configure the search method and the threshold for accuracy. This algorithm will not produce models with prediction rates below that threshold; therefore the algorithm may get trapped into local optima. The more strict the threshold, the more likely the algorithm to get trapped into a local optimum.

Two distinctive features of this learning algorithm should be highlighted. First, we use continuous search methods (e.g., Particle Swarm Optimization, Real-Coded Genetic Algorithm) to solve the constricted, combinatorial optimization problem. This can be achieved by discretizing the continuous space into non-homogeneous partitions, each denoting a specific state, where the size of each partition is heuristically determined. Second, the learning method recalculate the values of sigmoid parameters to compensate the alterations on the FCM topology. It is worth mentioning that this second feature is only applicable for FCM models using sigmoid neurons.

7. Improving the System Convergence

Most supervised learning methods reported in the literature do not accurately consider the FCM convergence into their learning scheme.[33] As a result, we obtain FCM-based model for scenario analysis and pattern classification with acceptable performance, but unable to converge to a fixed-point attractor. Ensuring the convergence to a fixed-point is often mandatory in decision-making scenarios where obtaining a non-oscillatory solution is expected, otherwise making a confident decision is not possible. In other domains (e.g., time series forecasting) the network convergence is less desirable.

To overcome this issue, Nápoles *et al.*[34] introduced a novel learning method to improve the convergence of sigmoid FCM-based classifiers once the causal relations defining the system semantics have been defined. The algorithm computes the sigmoid parameters attached to each neural entity leading to improved convergence features. This involves a continuous optimization problem with $2M$ variables, with M being the number of neurons.

FCM Expert implements this learning algorithm for both scenario analysis and pattern classification scenarios. Due to the fact that this algorithm is supervised, the expert must specify a set of examples where each attribute matches with a

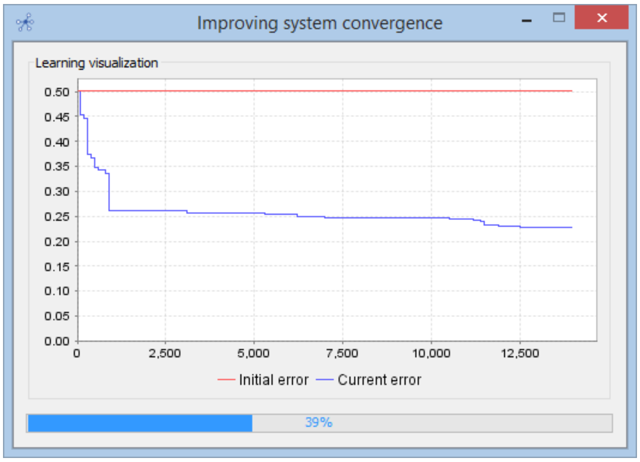


Fig. 12. Real-time visualization of error curve.

specific concept. Moreover, the expert must specify the accuracy level defining the expected deviation to the original responses. For simulation and experimentation tasks, real-time visualization (i.e., the error curve) is essential. Figure 12 shows this functionality, where the current parameters are used as seed in the learning procedure, allowing the inclusion of expert knowledge to guide the search process.

In a recent study, Nápoles *et al.*[35] analytically proved that, sometimes, under the weights constriction, the algorithm will be unable to improve the system convergence without harming the accuracy. Moved by this result, in Ref. 36 the authors proposed a supervised learning method for computing the weight set and the sigmoid parameters leading to convergence features from the beginning. This approach is specially useful when no constraint over the weight set exists, otherwise the method discussed in Ref. 34 should be adopted.

8. Exploring the FCM Expert Facilities Through a Case Study

In this section we illustrate some functionalities of the FCM Expert (mainly those related to learning algorithms) by using several FCM-based classifiers taken from Nápoles *et al.*[21,37] Such systems describe the resistance mechanism of the HIV-1 *protease* protein to existing inhibitors and the causal relation among sequence positions. The authors modeled the protein as an FCM where each sequence position was taken as a neuron, whereas a decision concept for the resistance target was also defined. The HIV-1 protease chain is defined by 99 amino acids, however, with the goal of reducing the model only sequence sites associated with drug resistance were adopted. In this topology, input neurons are fully connected, hence causal relations among concepts are established.

Figure 13 displays the resulting FCM-based classifier for the inhibitor Indinavir (IDV) where causal connections among neurons have been randomly generated due

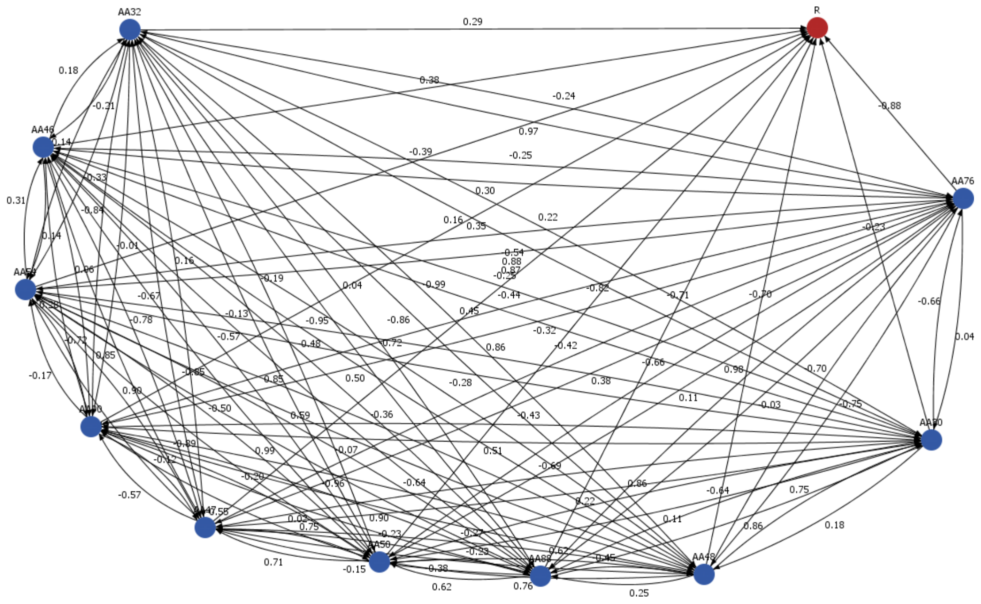
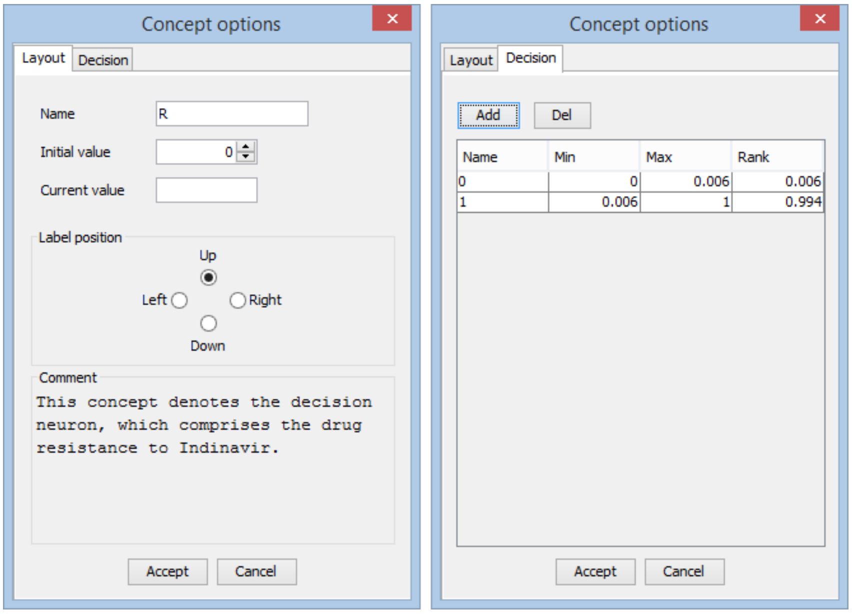


Fig. 13. FCM-based classifier for predicting the resistance of HIV-1 mutations.

to the lack of biological knowledge. In this modeling, the blue concepts represent the input neurons (i.e., sequence positions) and the red neuron denotes the resistance concept to the target inhibitor. This decision concept will be used to determine the decision class associated to each activation vector.

It is important to highlight that each FCM has associated a high-quality filtered datasets consisting on reported mutations and their resistance value.[38,39] For the sake of simplicity, only the FCM-based classifier for IDV will be used for illustrating the functionality of FCM Expert. Figure 14 displays the configuration of the decision neuron (once the modeling phase is done) for two classes, where the label "0" denotes the *susceptible class*, whereas "1" indicates the *resistant one*. This suggests that all input sequences leading to a numerical response between 0.007 and 1 will be classified as *resistant*, otherwise the input pattern (i.e., mutation) will be classified as *susceptible*.

Once the decision neuron is configured, we adjust the causal relations by using a supervised learning approach and the set of training sets associated to the target inhibitor. With the aim of illustrating this functionality we select the Global-best PSO[40] using 40 particles as swarm size and 80 generations as stopping criterion. Figure 15 displays the windows on which the expert must configure the required parameters (i.e., the population size, the generation number, the knowledge base path, and other PSO parameters such as the acceleration coefficients). This wizard can be reached by clicking the option *Learning Algorithms* into the menu *Run*, and next clicking the menu item *Compute Causal Relations*. In this section we will be



Concept options

Layout | Decision

Name: R

Initial value: 0

Current value:

Label position: [✓ Up] Left Right Down

Comment: This concept denotes the decision neuron, which comprises the drug resistance to Indinavir.

Accept Cancel

Concept options

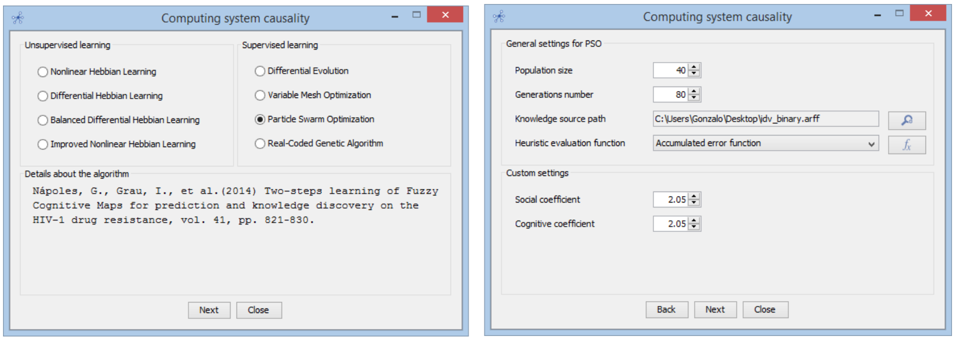
Layout | Decision

Add Del

Name	Min	Max	Rank
0	0	0.006	0.006
1	0.006	1	0.994

Accept Cancel

Fig. 14. Configuration of the decision space for the decision neurons. These windows can be reached by clicking the decision neuron (which is often denoted by a red concept).



Computing system causality

Unsupervised learning: Nonlinear Hebbian Learning; Differential Hebbian Learning; Balanced Differential Hebbian Learning; Improved Nonlinear Hebbian Learning

Supervised learning: Differential Evolution; Variable Mesh Optimization; [✓ Particle Swarm Optimization]; Real-Coded Genetic Algorithm

Details about the algorithm:

```
Nápoles, G., Grau, I., et al.(2014) Two-steps learning of Fuzzy
Cognitive Maps for prediction and knowledge discovery on the
HIV-1 drug resistance, vol. 41, pp. 821-830.
```

Next Close

Computing system causality

General settings for PSO

Population size	40
Generations number	80
Knowledge source path	C:\Users\Gonzalo\Desktop\idv_binary.arff
Heuristic evaluation function	Accumulated error function

Custom settings

Social coefficient	2.05
Cognitive coefficient	2.05

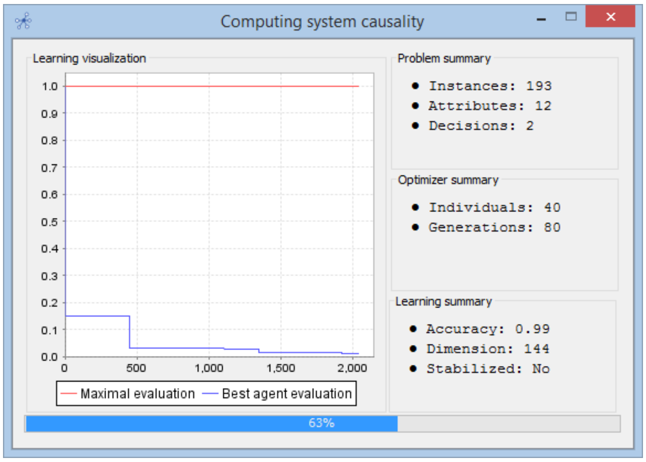
Back Next Close

(a) (b)

Fig. 15. Computing system causality. (a) Unsupervised and supervised learning algorithms. (b) Parameters configuration for the selected optimizer (Global-best PSO).

focused on supervised learning algorithms since they allow illustrating the strength of the proposed Java software tool in pattern recognition scenarios.

Figure 16 illustrates the learning progress for the IDV drug where experts can inspect relevant statistics such as the prediction rates and the convergence features. Moreover, we included a panel for plotting the best evaluation computed at each step, hence the experts can check whether the learning algorithm falls into non-progress configurations or not. Both the error curve and the statistics are updated



Problem summary

- Instances: 193
- Attributes: 12
- Decisions: 2

Optimizer summary

- Individuals: 40
- Generations: 80

Learning summary

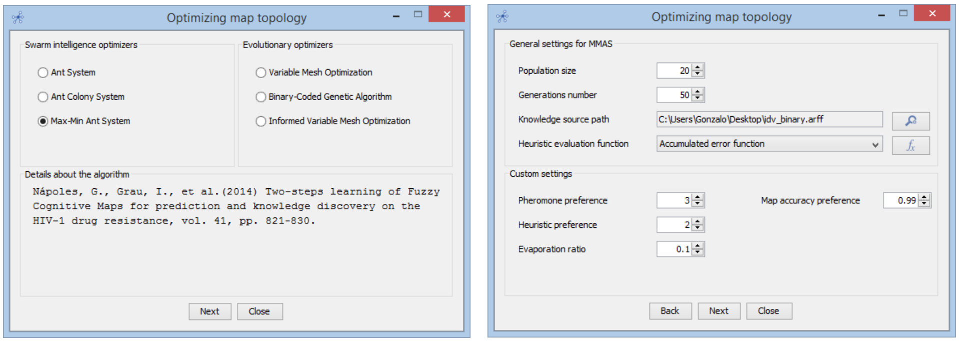
- Accuracy: 0.99
- Dimension: 144
- Stabilized: No

Fig. 16. Window summarizing relevant statistics related to the algorithm progress, when computing the causal weight matrix.

step-by-step, which allow visualizing the learning progress in real-time. For example, the reader may observe that the best configuration found so far leads to high prediction rates since 98% of instances were correctly classified. Nevertheless, this weight matrix does not ensure the classifier stability and thus there is no confidence about the predicted decision classes.

Once the causal weight matrix has been estimated, the expert could optimize the network topology by removing non-relevant concepts (if possible). With the purpose of illustrating this scenario we select the MAX-MIN Ant System[41] by using the following parameters: 20 artificial ants, 80 iterations as stopping criterion, the evaporation constant ρ is set to 0.1, the pheromone relevance $\alpha = 3$, while the heuristic preference $\beta = 2$. This wizard can be reached by clicking the option *Learning Algorithms* into the menu *Run*, and next clicking the menu item *Optimizing Network Topology*. Similarly to the algorithms for adjusting the causal weights, we need a set of training patterns since the goal of this algorithm is to reduce the map structure without losing information.

Figure 18 displays relevant statistics (e.g., norm of the best solution found so far, the reduction rate, the error induced when removing superfluous nodes) step-by-step in real-time. Moreover, we included a panel for plotting the best evaluation computed at each step, therefore the experts can check when the model is prone to converge to a global solution. In this representative example, the algorithm found a candidate sub-graph comprising only seven concepts, leading to a reduction rate of 30%. Observe that $\triangle Error = -0.01$, therefore when removing the weak neurons and their connections, the classifier is capable of recognizing more patterns (1% of the testing patterns). This suggests that the removed casual connections comprise conflicting knowledge, or perhaps they lead to unstable configurations where the FCM-based classifier produce chaotic or cyclic decision classes.



Swarm intelligence optimizers

- Ant System
- Ant Colony System
- [✓ Max-Min Ant System]

Evolutionary optimizers

- Variable Mesh Optimization
- Binary-Coded Genetic Algorithm
- Informed Variable Mesh Optimization

Details about the algorithm

```
Nápoles, G., Grau, I., et al.(2014) Two-steps learning of Fuzzy
Cognitive Maps for prediction and knowledge discovery on the
HIV-1 drug resistance, vol. 41, pp. 821-830.
```

General settings for MMAS

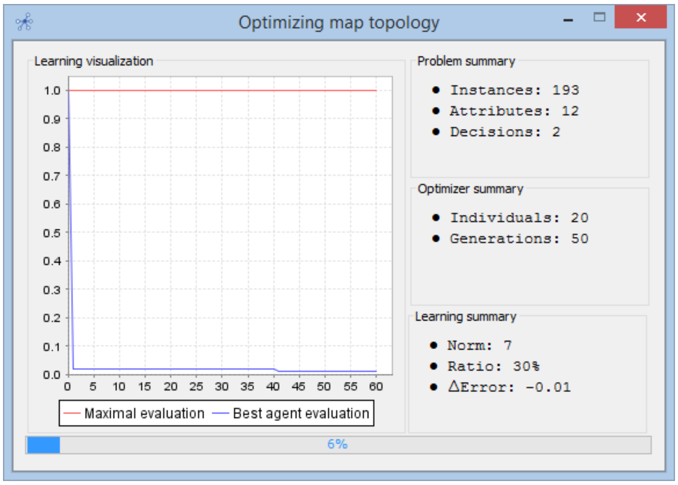
Population size	20
Generations number	50
Knowledge source path	C:\Users\Gonzalo\Desktop\idv_binary.arff
Heuristic evaluation function	Accumulated error function

Custom settings

Pheromone preference	3	Map accuracy preference	0.99
Heuristic preference	2		
Evaporation ratio	0.1		

(a) (b)

Fig. 17. Optimizing network topology. (a) Swarm Intelligence and Evolutionary optimizers. (b) Parameters configuration for the selected optimizer (MAX-MIN Ant System).



Problem summary

- Instances: 193
- Attributes: 12
- Decisions: 2

Optimizer summary

- Individuals: 20
- Generations: 50

Learning summary

- Norm: 7
- Ratio: 30%
- ΔError: -0.01

Fig. 18. Window summarizing relevant statistics related to the algorithm progress, when optimizing the network topology.

FCM Expert provides several facilities for handling convergence. For example, in order to explore the convergence, the expert could run a Convergence Plotter where the FCM responses for all training patterns are depicted. Instead of being focused on the last response, the Convergence Plotter displays the convergence behavior of the FCM-based system by inspecting the behavior of the decision neuron at each discrete-time step. It provides an overall picture of the stability features of the modeled system and therefore a measure of its reliability. Figure 19 portrays this interface for the FCM-based classifier designated to predict the HIV resistance to the IDV inhibitor.

The wizard for handling the convergence can be reached by clicking the option *Learning Algorithms* into the menu *Run*, and next clicking the item *Improve*

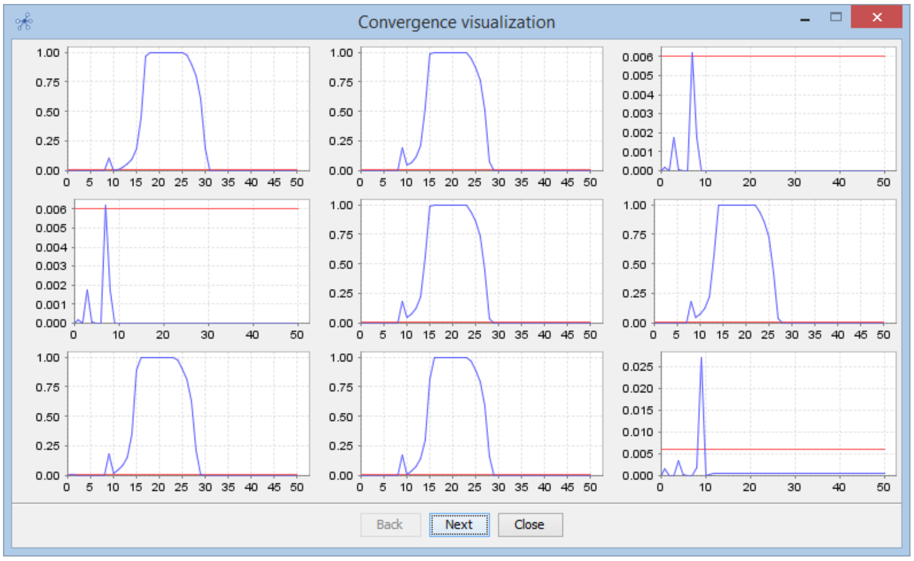
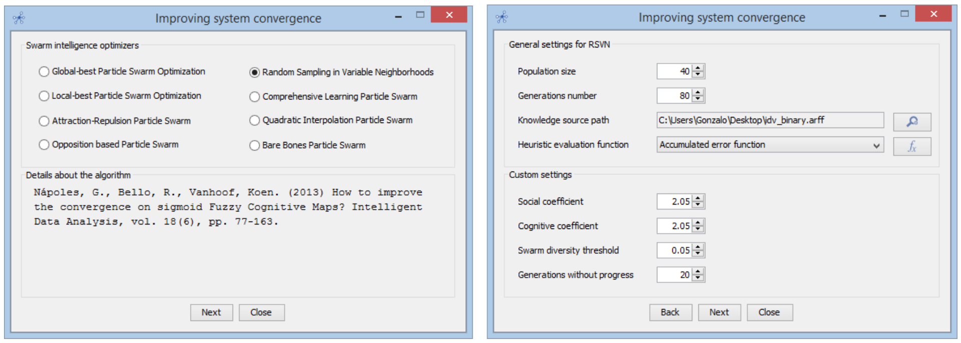


Fig. 19. Convergence plotter for IDV training patterns.



Improving system convergence

Swarm intelligence optimizers

○ Global-best Particle Swarm Optimization
○ Local-best Particle Swarm Optimization
○ Attraction-Repulsion Particle Swarm
○ Opposition based Particle Swarm
● Random Sampling in Variable Neighborhoods
○ Comprehensive Learning Particle Swarm
○ Quadratic Interpolation Particle Swarm
○ Bare Bones Particle Swarm

Details about the algorithm

```
Nápoles, G., Bello, R., Vanhoof, Koen. (2013) How to improve
the convergence on sigmoid Fuzzy Cognitive Maps? Intelligent
Data Analysis, vol. 18(6), pp. 77-163.
```

Next Close

Improving system convergence

General settings for RSVN

Population size	40
Generations number	80
Knowledge source path	C:\Users\Gonzalo\Desktop\idv_binary.arff
Heuristic evaluation function	Accumulated error function

Custom settings

Social coefficient	2.05
Cognitive coefficient	2.05
Swarm diversity threshold	0.05
Generations without progress	20

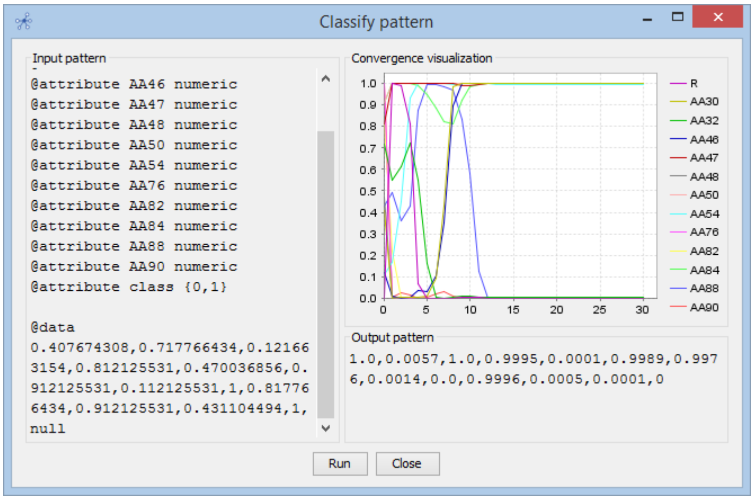
Back Next Close

(a) (b)

Fig. 20. Improving system convergence. (a) PSO-based optimizers to deal with non-progress configurations. (b) Parameters configuration for the selected optimizer (LSVN-PSO).

System Convergence. Regarding the optimizer we select the Global-best PSO[40] with a swarm size of 40 particles (each particle codifies a family of sigmoid functions, where the ith function will be used for transforming the activation value of the ith neuron) and 80 generations as stopping criterion. In this simulation we adopt the default values for the remaining parameters such as the diversity threshold or the number of generation without progress since they proved to be quite effective across selected benchmark.

After the learning phase we can confidentially classify new patterns since the algorithm improves the classifier convergence without affecting its prediction



```
@attribute AA46 numeric
@attribute AA47 numeric
@attribute AA48 numeric
@attribute AA50 numeric
@attribute AA54 numeric
@attribute AA76 numeric
@attribute AA82 numeric
@attribute AA84 numeric
@attribute AA88 numeric
@attribute AA90 numeric
@attribute class {0,1}

@data
0.407674308,0.717766434,0.12166
3154,0.812125531,0.470036856,0.
912125531,0.112125531,1,0.81776
6434,0.912125531,0.431104494,1,
null
```

```
1.0,0.0057,1.0,0.9995,0.0001,0.9989,0.997
6,0.0014,0.0,0.9996,0.0005,0.0001,0
```

Fig. 21. Windows for classifying new patterns.

capability. Figure 21 displays the interface to accomplish this goal where the expert must provide the input pattern to be classified. In this option the patterns are encoded using the well-known Attribute Relation File Format.[39] Observe that the default value for the decision neuron is "null" since it will be predicted from the activation space of the decision neuron. After executing the inference rule, the software provides the activation value of input-type neurons and the predicted decision class. The reader may notice that this option may also be used in modeling and control environments, although the option *Run Inference Process* is more convenient and intuitive when analyzing new scenarios.

Before concluding the paper, it should be highlighted that we preferred to be focused on the software options rather than the mathematical formulation of the implemented algorithms. For further information about these learning procedures and the numerical simulations, the reader could consult the original papers.

9. Concluding Remarks

Fuzzy Cognitive Maps have been extensively reported in literature for modeling real-life problems by recreating virtual scenarios. A weak point concerning this field is the lack of well-established software tools providing options for simulation and/or experimentation. In this paper, we presented FCM Expert to the scientific community as a flexible software that allows modeling, learning and simulating FCM-based systems. FCM Expert is an object-oriented implementation organized in packages, with a compact and intuitive interface. Our software proposal includes algorithms for computing the parameters defining the FCM model, optimizing the FCM topology and improving the convergence without losing information. These options are not available in other implementations as we checked the existing literature.

Furthermore, we have illustrated the existence of options for configuring the decision space, for analyzing new scenarios based on the modeled concepts and successive state vectors, for evaluating the convergence of the FCM-based system, among other options. These facilities have a graphical support that allows completely designing FCM-based systems with minimal effort to subject matter experts that are not necessarily experts in Computer Science. On the other hand, FCM Expert is a platform-independent software where the user/expert can mine the knowledge related to the system under analysis. FCM Expert allows modeling a wide range of simulation and pattern classification problems with high flexibility. We are currently developing other learning algorithms with stronger mathematical principles. Likewise, we are testing a new module with methods for time series forecasting.

Acknowledgments

The authors would like to thank Prof. Rafael Bello, Prof. Maria M. García, Prof. Elpiniki Papageorgiou, Ciro Rodríguez, Ricardo Pérez, Frank Vanhoenshoven and Yailen Hernández for their kind support to this project.

References

1. B. Kosko, Fuzzy cognitive maps, *International Journal of Man-Machine Studies* **24**(1) (1986) 65–75.
2. E. I. Papageorgiou and J. L. Salmeron, Methods and algorithms for fuzzy cognitive map-based modeling, in *Fuzzy Cognitive Maps for Applied Sciences and Engineering* (Springer, 2014), pp. 1–28.
3. W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *The Bulletin of Mathematical Biophysics* **5**(4) (1943) 115–133.
4. B. Kosko, Hidden patterns in combined and adaptive knowledge networks, *International Journal of Approximate Reasoning* **2**(4) (1988) 377–393.
5. G. Nápoles, I. Grau, K. Vanhoof and R. Bello, Hybrid model based on rough sets theory and fuzzy cognitive maps for decision-making, in *Rough Sets and Intelligent Systems Paradigms* (Springer, 2014), pp. 169–178.
6. J. L. Salmeron and E. Papageorgiou, Fuzzy grey cognitive maps and nonlinear hebbian learning in process control, *Applied Intelligence* **41**(1) (2014) 223–234.
7. T. L. Kottas, Y. S. Boutalis and M. A. Christodoulou, Fuzzy cognitive network: A general framework, *Intelligent Decision Technologies* **1**(4) (2007) 183–196.
8. G. Nápoles, I. Grau, M. León and R. Grau, Modelling, aggregation and simulation of a dynamic biological system through fuzzy cognitive maps, in *Advances in Computational Intelligence* (Springer, 2012), pp. 188–199.
9. M. León, G. Nápoles, R. Bello, L. Mkrtchyan, B. Depaire and K. Vanhoof, Tackling travel behaviour: An approach based on fuzzy cognitive maps, *International Journal of Computational Intelligence Systems* **6**(6) (2013) 1012–1039.
10. G. Nápoles, I. Grau, R. Falcon, R. Bello and K. Vanhoof, A granular intrusion detection system using rough cognitive networks, in *Recent Advances in Computational Intelligence in Defense and Security* (Springer, 2016), pp. 169–191.
11. G. A. Papakostas, Y. S. Boutalis, D. E. Koulouriotis and B. G. Mertzios, Fuzzy cognitive maps for pattern recognition applications, *International Journal of Pattern Recognition and Artificial Intelligence* **22**(8) (2008) 1461–1486.

12. M. León, G. Nápoles, C. Rodriguez, M. M. García, R. Bello and K. Vanhoof, A fuzzy cognitive maps modeling, learning and simulation framework for studying complex system, in *New Challenges on Bioinspired Applications* (Springer, 2011), pp. 243–256.
13. C. D. Stylios and P. P. Groumpos, Modeling complex systems using fuzzy cognitive maps, *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans* **34**(1) (2004) 155–162.
14. E. I. Papageorgiou, A new methodology for decisions in medical informatics using fuzzy cognitive maps based on fuzzy rule-extraction techniques, *Applied Soft Computing* **11**(1) (2011) 500–513.
15. S. Mohr, Software design for a fuzzy cognitive map modeling tool (Tensselaer Polytechnic Institute, 1997).
16. J. Aguilar and J. Contreras, The FCM designer tool, in *Fuzzy Cognitive Maps* (Springer, 2010), pp. 71–87.
17. M. Margaritis, C. Stylios and P. Groumpos, Fuzzy cognitive map software, in *10th Int. Conf. on Software, Telecommunications and Computer Networks* (2002).
18. S. A. Gray, S. Gray, L. J. Cox and S. Henly-Shepard, Mental modeler: A fuzzy-logic cognitive mapping modeling tool for adaptive environmental management, in *2013 46th Hawaii Int. Conf. on System Sciences* (*HICSS*) (IEEE, 2013), pp. 965–973.
19. D. De Franciscis, JFCM: A java library for fuzzy cognitive maps, in *Fuzzy Cognitive Maps for Applied Sciences and Engineering* (Springer, 2014), pp. 199–220.
20. K. Poczeta, A. Yastrebov and E. Papageorgiou, Learning fuzzy cognitive maps using structure optimization genetic algorithm, *Annals of Computer Science and Information Systems* **5** (2015) 547–554.
21. G. Nápoles, I. Grau, R. Bello and R. Grau, Two-steps learning of fuzzy cognitive maps for prediction and knowledge discovery on the HIV-1 drug resistance, *Expert Systems with Applications* **41**(3) (2014) 821–830.
22. G. Papakostas and D. Koulouriotis, Classifying patterns using fuzzy cognitive maps, in *Fuzzy Cognitive Maps* (Springer, 2010), pp. 291–306.
23. G. Felix, G. Nápoles, R. Falcon, K. Vanhoof, W. Froelich and R. Bello, Hidden patterns in combined and adaptive knowledge networks, *Artificial Intelligence Review* (2017).
24. J. A. Dickerson and B. Kosko, Virtual worlds as fuzzy cognitive maps, *Presence: Teleoperators & Virtual Environments* **3**(2) (1994) 173–189.
25. A. V. Huerga, A balanced differential learning algorithm in fuzzy cognitive maps, in *Proc. of the 16th Int. Workshop on Qualitative Reasoning* (2002).
26. E. Papageorgiou, C. Stylios and P. Groumpos, Fuzzy cognitive map learning based on nonlinear Hebbian rule, in *AI 2003: Advances in Artificial Intelligence* (Springer, 2003), pp. 256–268.
27. S.-J. Li and R.-M. Shen, Fuzzy cognitive map learning based on improved nonlinear Hebbian rule, in *Proc. of 2004 Int. Conf. on Machine Learning and Cybernetics* (IEEE, 2004), pp. 2301–2306.
28. R. Poli, J. Kennedy and T. Blackwell, Particle swarm optimization, *Swarm Intelligence* **1**(1) (2007) 33–57.
29. R. Storn and K. Price, Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization* **11**(4) (1997) 341–359.
30. F. Herrera, M. Lozano and J. L. Verdegay, Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis, *Artificial Intelligence Review* **12**(4) (1998) 265–319.

31. A. Puris, R. Bello, D. Molina and F. Herrera, Variable mesh optimization for continuous optimization problems, *Soft Computing* **16**(3) (2012) 511–525.
32. G. Nápoles, M. León, I. Grau, K. Vanhoof and R. Bello, *Fuzzy Cognitive Maps Based Models for Pattern Classification: Advances and Challenges* (Springer, 2017).
33. G. Nápoles, R. Bello and K. Vanhoof, Learning stability features on sigmoid fuzzy cognitive maps through a swarm intelligence approach, in *Progress in Pattern Recognition, Image Analysis, Computer Vision and Applications* (Springer, 2013), pp. 270–277.
34. G. Nápoles, E. Papageorgiou, R. Bello and K. Vanhoof, On the convergence on sigmoid fuzzy cognitive maps, *Information Sciences* **349–350** (2016) 154–171.
35. G. Nápoles, L. Concepción, R. Falcon, R. Bello and K. Vanhoof, On the accuracy-convergence trade-off in sigmoid fuzzy cognitive maps, *IEEE Transactions on Fuzzy Systems* (submitted) (2017).
36. G. Nápoles, E. Papageorgiou, R. Bello and K. Vanhoof, Learning and convergence of fuzzy cognitive maps used in pattern recognition, *Neural Processing Letters* **45** (2017) 431–444.
37. G. Nápoles, I. Grau, M. León and R. Grau, Modelling, aggregation and simulation of a dynamic biological system through fuzzy cognitive maps, in *Advances in Computational Intelligence* (Springer, 2012), pp. 188–199.
38. I. Grau, G. Nápoles and M. M. García, Predicting HIV-1 protease and reverse transcriptase drug resistance using fuzzy cognitive maps, in *Progress in Pattern Recognition, Image Analysis, Computer Vision and Applications* (Springer, 2013), pp. 190–197.
39. M. Hall, I. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 3rd edn. (Morgan Kaufmann Publishers, 2011).
40. Y. Shi and R. Eberhart, A modified particle swarm optimizer, in *The 1998 IEEE Int. Conf. on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence* (IEEE, 1998), pp. 69–73.
41. T. Stützle and H. H. Hoos, MAX–MIN ant system, *Future Generation Computer Systems* **16**(8) (2000) 889–914.