A Note on Deterministic Learning of Hybrid Fuzzy Cognitive Maps and Network Reduction Approaches

Gonzalo Nápoles[a1,2], Agnieszka Jastrzębska[3], Carlos Mosquera[4], Koen Vanhoof[1], and Władysław Homenda[3]

[1]Faculty of Business Economics, Universiteit Hasselt, Belgium
[2]Department of Cognitive Science & Artificial Intelligence, Tilburg University, The Netherlands
[3]Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
[4]Faculty of Sciences and Bioengineering Sciences, Vrije Universiteit Brussel, Belgium

We have proposed a hybrid Fuzzy Cognitive Map-based model that incorporates static expert knowledge and information automatically inferred from the data. The paper introduces a fast and deterministic algorithm to construct such a model and a post-optimization procedure to fine-tune the resulting architecture. Simulations showed the superiority of our proposal in problems devoted to modeling complex systems.

Contribution overview

Fuzzy Cognitive Maps (FCMs) belong to the family of cognitive semantic models. They are represented as directed weighted graphs in which vertices are concepts (knowledge granules) and arcs correspond to relations between them. FCMs are used to visualize, model, and simulate the behavior of systems.

In our new paper [1] the focus is on an FCM-based model for simulating dynamic systems. In such a system, we often have several input variables that influence the values of the output variables. In this paper, a new hybrid approach for designing FCMs simulating such a system is presented. The proposed approach combines the ability to integrate expert knowledge into the map architecture with elements of automatic model learning from historical data. In practice, in the proposed model we can incorporate information about the weights between input variables given by experts, and the remaining weights are learned automatically. The use of the term "hybrid" refers in our case to the possibility of integrating expert knowledge about input variables into a model whose other parts are learned from the data. The specific research goals and our novel contributions can be summarized as follows:

- We developed a new method for constructing cognitive maps, in which we incorporate both static expert knowledge and knowledge extracted from the data in the learning procedure.
- We introduced a new very fast, deterministic way of learning model weights.
- We introduced a post-optimization approach for eliminating irrelevant weights and correcting the model to preserve the accuracy.

The new method

Figure 1 shows the proposed architecture with three inputs (x_1, x_2, x_3) and three outputs (y_1, y_2, y_3).

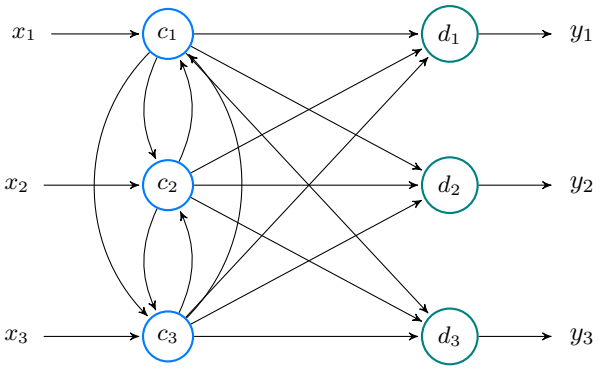

Figure 1: FCM-based model with three input neurons (c_1, c_2, c_3) and three output neurons (d_1, d_2, d_3).

The reasoning process is given in below,

$$a_{ki}^{(t+1)} = f_i\left(\sum_{j=1}^{P} w_{ji} a_{kj}^{(t)}\right), i \neq j \tag{1}$$

where P is the number of nodes in the network, k is the input activation vector index, w_{ji} is the weight connecting c_j with c_i, and $f_i(\cdot)$ is the transfer function. We adopted a simplified variant of the sigmoid function, which is depicted below,

$$f_i(x) = l_i + \frac{u_i - l_i}{(1 + e^{-\lambda_i(x-h_i)})} \tag{2}$$

such that $\lambda_i > 0$ and $h_i \in \mathbb{R}$ are parameters that define the shape of the sigmoid function.

The initial activation values $a_1^{(0)}$, $a_2^{(0)}$ and $a_3^{(0)}$ for input neurons correspond to the problem variables x_1, x_2 and x_3, respectively. Let us use N to note the number of input neurons $C = \{c_1, \ldots, c_N\}$ and M to note the number of output neurons $D = \{d_1, \ldots, d_M\}$. The weight matrix W is composed of two sub-matrices W^I and W^O. The first one contains the connections among the input neurons. W^I should ideally be defined by domain experts and it will not be modified during the learning phase. Sub-matrix W^O contains the weights between the input and the output nodes. This matrix will be learned automatically.

The inverse learning method

In the paper, we developed a learning method to compute W^O, the sub-matrix that contains the weights between the input and the output nodes.

Assume that $[X, Y]$ is a training dataset where $X = [x_{ij}], i = 1, \ldots, K, j = 1, \ldots, N$. K is the number of input instances, each instance is described with N input variables, $x_{ij} \in [0, 1]$. $Y = [y_{ij}], i = 1, \ldots, K, j = 1, \ldots, M$ is a matrix containing the true values of the M output variables for each one of the K instances.

The first step toward computing the W^O matrix is to capture the system semantics with the use of input and the weight matrix W^I. Thus, let $\Psi^{(T)}(X)$ denote an $N \times K$ matrix after performing T iterations of the FCM inference process on the input matrix X, that is $\Psi^{(T)}(X) = [a_{ij}^{(T)}], i = 1, \ldots, K, j = 1, \ldots, N$.

The second learning step computes W^O by using the proposed pseudoinverse learning rule,

$$W^O = \left(\Psi^{(T)}(X)\right)^{\ddagger} F^{-}\left(Y\right) \tag{3}$$

where $(\cdot)^{\ddagger}$ represents the Moore-Penrose (MP) inverse of a given matrix, and

$$F^{-}\left(Y\right) = \begin{bmatrix} f_1^{-}(y_{11}) & \ldots & f_i^{-}(y_{1i}) & \ldots & f_M^{-}(y_{1M}) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ f_1^{-}(y_{k1}) & \ddots & f_i^{-}(y_{ki}) & \ddots & f_M^{-}(y_{kM}) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ f_1^{-}(y_{K1}) & \ldots & f_i^{-}(y_{Ki}) & \ldots & f_M^{-}(y_{KM}) \end{bmatrix}$$

is a $K \times M$ matrix containing the inverse of the transfer functions attached to output neurons:

$$f_i^{-1}(y) = \frac{-\ln(-1-y) + h_i \lambda_i}{\lambda_i}. \tag{4}$$

We use an orthogonal projection to obtain the MP inverse. If a matrix H has linearly independent columns ($H^\top H$ is nonsingular), then $H^{\ddagger} = (H^\top H)^{-} H^\top$. In contrast, if H has linearly independent rows ($HH^\top$ is nonsingular), then $H^{\ddagger} = H^\top (H^\top H)^{-}$. The former is a left inverse because $H^{\ddagger} H = I$, while the latter is a right inverse because $HH^{\ddagger} = I$.

To overcome the issue of weights that lie outside of the desired $[-1, 1]$ interval, in the paper, we proposed a post-training weight normalization method to ensure that $w_{ji} \in [-1, 1]$. What is more, we proposed a method for superfluous weights elimination. After this step, the procedure provides the means for calibrating the weights to be retained.

Numerical simulations

In the paper, we have conducted several simulations using 35 datasets. We compared the efficiency of the new method with well-known population-based optimizers: Global-best Particle Swarm Optimization, Real-Coded Genetic Algorithm, and Differential Evolution. Figure 2 shows the box-plots associated with Mean Squared Error (MSE) for each optimization model after being tested on datasets used for comparison. The results confirmed that the MP inverse learning method is outperforms the remaining optimization approaches when it comes to the prediction error.

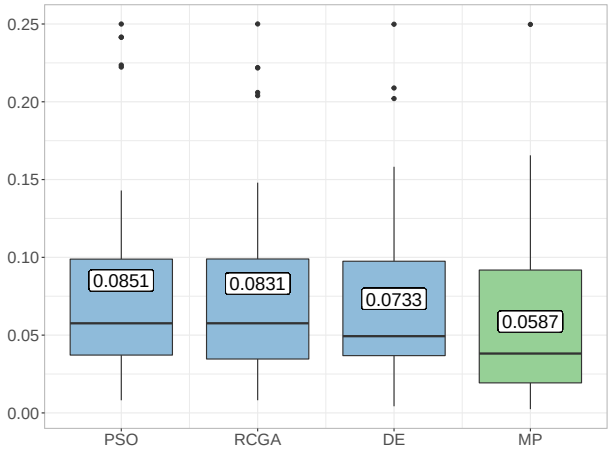

Figure 2: MSE reported by each optimization model across the 35 datasets used for simulation.

The training time of the proposed MP learning rule is significantly smaller when compared with the ones attached to state-of-the-art population-based optimizers. The average processing time of tested algorithms was as follows: Particle Swarm Optimization 4.951s, Real-Coded Genetic Algorithm 5.509s, Differential Evolution 12.964s. The new method took on average 0.003s.

Concluding remarks

The new method based on the Moore-Penrose pseudoinverse is fast and deterministic. Both those qualities are rarely seen together in the domain of FCM learning. The numerical simulations have shown that our system is able to significantly outperform state-of-the-art population-based algorithms in terms of both simulation error and training time.

*

Notes

a. Email: G.R.Napoles@tilburguniversity.edu

*

References

[1] Nápoles, G., Jastrzębska, A., Mosquera, C., Vanhoof, K., Homenda, W., Deterministic learning of hybrid Fuzzy Cognitive Maps and network reduction approaches, *Neural Networks,* **Volume 124** (2020) Pages 258–268