A Framework for a Novel Scalable FCM Learning Method

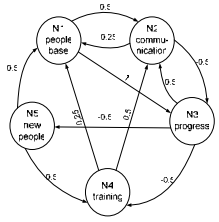
Wojciech Stach, Lukasz Kurgan, and Witold Pedrycz
Department of Electrical and Computer Engineering,
University of Alberta, Edmonton, Canada
{wstach, lkurgan, pedrycz}@ece.ualberta.ca

Abstract

Fuzzy Cognitive Maps (FCMs) are a handy method for dynamic systems modeling. A recently introduced RCGA learning method allows for establishing high-quality FCMs from historical data. The main drawback of this method is its scalability. In this paper, a framework for a novel scalable learning method for FCMs is introduced.

1. Background

Fuzzy Cognitive Maps, proposed by Kosko [1], are a qualitative tool for modeling of dynamic systems, which combine elements of neural networks and fuzzy logic. FCMs represent a given system as a collection of concepts connected by mutual relationships. Strength of each relationship is quantified and expressed by a real number from -1 to 1. Most conveniently, the model can be represented as a graph, which consists of nodes (concepts) connected by directed arcs (relationships). Alternatively, FCM is defined by a matrix, called *connection matrix*, which stores all the relationships values. Figure 1 shows a simple example of FCM (graph and connection matrix) that models software development project [2].



	N1	N2	N3	N4	N5
N1	0	0.5	1	0	0
N2	0.25	0	-0.5	0	0
N3	0	0.5	0	-0.5	-0.5
N4	0.25	0.5	0	0	0
N5	0.5	0	0	-0.5	0

Figure 1. Example of FCM

There are two main groups of approaches to develop FCMs: using an expert knowledge from the domain of application (deductive modeling), and using learning algorithms to establish FCMs from historical data (inductive modeling) [3].

Once developed, FCM can be used to qualitative analysis of the modeled systems, and answering "what-if" questions. This is achieved by simulating the model, i.e. by calculating its state over successive iterations. State of FCM is defined as a set of activation levels of all the concepts. Activation level of a certain concept expresses a degree to which the concept is present in the system at a particular iteration. Each state (except for the first one, which has to be defined prior to the simulation) is computed based on the system state at the preceding iteration.

In this paper, we use a learning method for FCMs that is based on real-coded genetic algorithms (RCGA). Figure 2 shows a high-level diagram of this method [4].

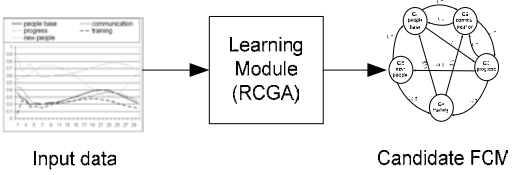


Figure 2. RCGA Learning Method

The RCGA learning method uses *input data* to develop an FCM (*candidate FCM*), which is capable to mimic the data. For a given system with N concepts, the learning task boils down to establish N^2 parameters that define FCM (they correspond to all possible relationships between the concepts). The core element of this method is a real-coded genetic algorithm [5], which is a floating-point extension to generic genetic algorithms and concerns chromosome representation. The RCGA method was comprehensively tested, and the experiments proved its effectiveness and high quality of generated models.

2. Motivation

The bottleneck of the RCGA method for FCMs is its scalability, as the number of parameters that have to be established grows quadratically with the map size defined as the number of concepts. In addition, genetic optimization is time consuming when employed to problems with large number of variables. At the same time, in some areas such as systems biology, the underlying networks that could be modeled with FCMs are large (several dozens of nodes). These issues call for

development of learning approaches for FCMs that would be fast enough to be applied to larger systems.

3. Parallel RCGA

In our recent work [6], we have proposed an approach to speed-up the RCGA method, which was based on parallelization of genetic algorithms. The reported results showed that learning of FCMs on eight processors was four times faster than the sequential learning, i.e. on a single processor.

In this paper, a framework for a novel scalable learning method based on a divide-and-conquer strategy is introduced. A high-level diagram of the proposed method, *Parallel RCGA*, is shown in Figure 3.

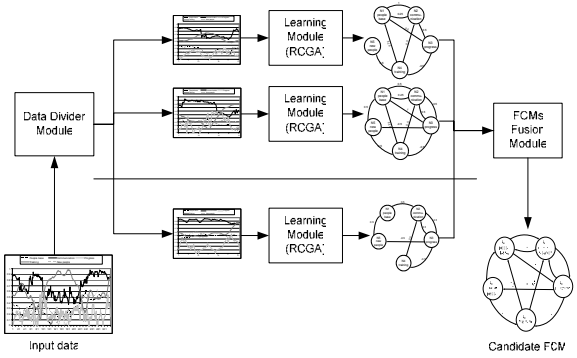


Figure 3. Parallel RCGA Learning Method

The proposed approach involves: (1) dividing the input data into subsets, (2) performing independent, parallel, learning on each subset, and (3) merging the sub-models. The motivation behind using this method is that the number of calculations in fitness function evaluation, which is the most time-consuming part of RCGA optimization, depends linearly on the number of input data length. Therefore, we may expect nearly linear speed-up in the execution time when running the experiments on a multiprocessor machine (each sub-problem on a different processor). The two critical modules in this approach are *Data Divider* and *FCMs Fusion*.

Data Divider takes the input data and split it into subsets that are then used to learn the sub-models. Taking advantage of FCMs characteristics, various strategies may be employed. In the first strategy, the data are divided into *K* non-overlapping, subsequent subsets (where K is the number of available processors). In the second one, the split is done as above except that the subsets overlap. The reason for that is to improve the quality of each sub-model, as more data points are used in learning. In the third strategy, the subsets are obtained by a random selection of data points (along with their predecessors). This is possible since in the RCGA method any two consecutive data points can be added to the learning process. In the fourth strategy, the random selection is carried out (as in the third one); however it is allowed that some data points can be selected to more than one subset. The motivation is the same as in case of the second strategy.

FCMs Fusion merges the sub-models. The subject of combining multiple FCMs into a single model is described in literature with respect to deductive modeling [7]. The simplest method is to obtain the final model by calculating averages of corresponding relationship values over all the sub-models. Another described method is to add credibility coefficient to each sub-model. In our learning approach, they can be calculated using in-sample error value for each sub-model – the higher the error is, the lower is the credibility of the sub-model, and vice versa. Then, the final model is established by calculating weighted averages of all the relationships (using these coefficients as weights) over all the sub-models.

4. Conclusions

In this work, a framework for a novel scalable method for learning FCMs is introduced. The new approach uses divide-and-conquer strategy to break the learning process into sub-processes that can be executed in parallel.

We expect that this approach will reduce the learning time almost eight times on eight processors comparing to the sequential learning. A comprehensive set of experiments needs to be carried out in order to both verify the above hypothesis and elaborate on the quality of the models developed by using this method.

5. References

[1] B. Kosko, "Fuzzy Cognitive Maps", *International Journal of Man-Machine Studies*, vol. 24, pp. 65–75, 1986

[2] W. Stach, and L. Kurgan, "Modeling Software Development Projects Using Fuzzy Cognitive Maps", *4th ASERC Workshop on Quantitative and Soft Software Engineering (QSSE'04)*, pp.55-60, 2004

[3] W. Stach, L. A. Kurgan, and W. Pedrycz, "A Survey of Fuzzy Cognitive Map Learning Methods", In: P. Grzegorzewski, M. Krawczak, and S. Zadrozny, (Eds.), Issues in Soft Computing: Theory and Applications, *Exit*, pp. 71–84, 2005

[4] W. Stach, L. Kurgan, W. Pedrycz, and M. Reformat, "Genetic Learning of Fuzzy Cognitive Maps," *Fuzzy Sets and Systems*, vol. 153, no. 3, pp. 371–401, 2005

[5] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling Real-coded Genetic Algorithms: Operators and Tools for Behavioural Analysis", *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265-319, 1998

[6] W. Stach, L. Kurgan, and W. Pedrycz, "Parallel Learning of Large Fuzzy Cognitive Maps", *submitted*, 2007

[7] B. Kosko, "Hidden Patterns in Combined and Adaptive Knowledge Networks", *International Journal of Approximate Reasoning*, vol. 2, pp. 377-393, 1988