International Journal of Computational Intelligence Systems, Vol. 9, No. 3 (2016) 416-432

SONFIS: Structure Identification and Modeling with a Self-Organizing Neuro-Fuzzy Inference System

Héctor Allende-Cid [1] *, Rodrigo Salas [2], Alejandro Veloz [2,4], Claudio Moraga [3], Héctor Allende [4]

[1] Pontificia Universidad Católica de Valparaíso,
Escuela de Ing. Informática, Av. Brasil 2241,
Valparaíso, Chile
E-mail: hector.allende@pucv.cl

[2] Universidad de Valparaíso
Escuela de Ing. Biomédica, General Cruz 222
Valparaíso, Chile
E-mail: {alejandro.veloz; rodrigo.salas}@uv.cl

[3] Technical University of Dortmund, 44221 Dortmund, Germany.
E-mail: claudio.moraga@udo.edu

[4] Universidad Técnica Federico Santa María, Chile
Departamento de Informática, Avda. España 1680, Valparaíso, Chile.
E-mail: hallende@inf.utfsm.cl.

Received 25 June 2015

Accepted 27 January 2016

Abstract

This paper presents a new adaptive learning algorithm to automatically design a neural fuzzy model. This constructive learning algorithm attempts to identify the structure of the model based on an architectural self-organization mechanism with a data-driven approach. The proposed training algorithm self-organizes the model with intuitive adding, merging and splitting operations. Sub-networks compete to learn specific training patterns and, to accomplish this task, the algorithm can either add new neurons, merge correlated ones or split existing ones with unsatisfactory performance. The proposed algorithm does not use a clustering method to partition the input-space like most of the state of the art algorithms. The proposed approach has been tested on well-known synthetic and real-world benchmark datasets. The experimental results show that our proposal is able to find the most suitable architecture with better results compared with those obtained with other methods from the literature.

Keywords: Neuro-Fuzzy Models, Self-Organization, Nonlinear Structure Identification.

1. Introduction

Determining the appropriate architectural design of machine learning models is one of the most challenging issues in system identification and modeling of several engineering and science applications. Nowadays there are machine learning models that still rely on a rigid and pre-specified topology, where

* corresponding author

their learning algorithms are limited to search only in the parametric space for a suitable architecture. The users, based on their empirical intuitions and experience, are usually the ones that select the appropriate architecture to solve specific learning problems. For example, both the selection of the number of hidden neurons of a Feedforward Artificial Neural Network[7,9] and the activation functions in an Adaptive Network[13] are crucial and difficult decisions for system modeling in many real world problems. If the model size (complexity) is underestimated the model will not be able to effectively solve the problem, while an overly large size tends to overfit the training data and consequently results in poor generalization performance[28].

An important and difficult issue in neural modeling is the structure identification of complex and high dimensional systems. Several solutions that introduce flexibility to the architecture have been proposed to face this difficult problem. The most common approach is based on two learning stages. The first stage consists in the partitioning of the input space, while the second stage is used to estimate the parameters and adjust the neuronal model to the system. Nevertheless, this approach assumes regularity and independence of the neuronal (sub)model for each partition, an assumption that in general is not correct.

The most well known schemes for input space partitioning are the rigid-grid-type, the clustering-based, the Genetic-Algorithm-based, scatter-type and the self-organization partitioning. In the grid-type partitioning[13,11], each block of the grid is associated to a neural sub-network. However, a major drawback of this scheme is the stiffness of the architecture during the learning process and the fact that the number of neurons required for a suitable system representation increases exponentially with the dimension size. The clustering-based partitioning provides a more flexible partition, where the subnetworks of the model are located according to the input data distribution. However, the interpretability of the model is hard for the user[14,8] and the learning process requires two separated stages. The Genetic-Algorithm-based partitioning is based on evolutionary paradigms. These global search heuristics optimize the input space partitioning[24]. This approach requires high computational time to evaluate and find suitable partitions. Hence this scheme is not adequate for on-line operations. The scatter type partitioning divides the input space into patches[21]. It covers a subset of the whole input space that characterizes a region of possible occurence of the input vectors. The scatter-type partitioning can also limit the number of rules to a reasonable amount. This makes it hard to estimate the overall mapping directly from the consequent of each rule output. Finally, a self-organizing partitioning solves the input space partitioning problem by means of a learning algorithm which automates structure and parameters identification simultaneously[1,19,32,38]. This type of partitioning has a strong dependence on the self-organizing operations. Most operations are parametric, thus, a bad choice of these parameters can lead to over-partitioning of the input space. In spite of this, it has the advantage that it does not rely on too many assumptions, like the other methods.

The remainder of the article is structured as follows. In the next section we give a brief state of the art of identification of neuro-fuzzy systems. In the following section the Adaptive Network-based Fuzzy Inference System (*ANFIS*) is described due to its importance as the underlying structure of our proposed *SONFIS* model, which is presented in the following section. Because of this relationship we kept the suffix "FIS" in the name of our model, even though no explicit fuzzy inference is pursued. Section 5 presents experimental results on *SONFIS* and some discussion. Finally, the last section concludes with some remarks and we delineate some future works.

2. The State of the Art of Constructive Methods

Nowadays, constructive methods for flexible modeling and identification have attracted the atention [1,19,32,38]. Several authors have extended the neuro-fuzzy models in order to endow them with some constructive capabilities. Originally, the neuro-fuzzy models were meant to extract fuzzy if-then rules from numerical data, which have shown to be

able to reach very accurate numerical models. The interpretability of extracted fuzzy if-then rules and the numerical accuracy of the final model with these kind of systems are however conflicting goals, that hardly ever may simultaneously be achieved[12].

Juang et al. introduced the self-constructing neural fuzzy inference network[14] (*SONFIN*) with online learning ability. *SONFIN* first performs a structure identification by means of an aligned clustering-based algorithm in the input space. After acquiring the initial structure of the input space, the algorithm requires the identification of the parameters of the consequent rules. Following the structure initialization, the consequent parameters are adjusted by either least mean squares (*LMS*) or recursive least squares (*RLS*). On the other hand, Chuang et al. presented the robust fuzzy regression agglomeration[8] (*RFRA*) that uses a robust clustering approach to determine the neuro-fuzzy structure. Tung et al. introduced the *GenSoFNN*[38] model that employs a new clustering technique known as discrete incremental clustering (*DIC*). Furthermore the fuzzy rules obtained by the network are consistent and compact, since *GenSoFNN* has built-in mechanisms to identify and prune redundant and/or obsolete rules. Leng et al. proposed a structure learning which attempts to achieve an economical network size with a self-organizing approach based on operators that add or prune fuzzy rules to the model structure depending on an error measure[19]. Qiao et al. proposed the self-organizing fuzzy neural network [32] (*SOFNN*), where a self-organizing clustering approach is used based on rival penalized competitive learning (*RPCL*), to establish the structure of the network and to obtain the initial values of its parameters. Then they use a supervised learning method to optimize these parameters. Qiao et al. later presented a self-organizing approach based on Radial basis functions[33]. Khayat et al presented an implementation of SOFNN with Genetic Algorithms and PSO[17]. Jakubek et al. used a residual of the generalized total least squares[11] (*GTLS*) parameter estimation with an iterative decomposition of the partition space. Afterwards in each step of this approach, an axis-oblique partitioning is performed by multiobjective optimization using the Expectation-Maximization algorithm. Liu et al. introduced the self-spawning neuro-fuzzy system[21] (*SSNFS*) consisting in a self-spawning competitively learning algorithm to incrementally search the rule patches. This method is capable of both structural and parametric learning. It constructs the fuzzy system by detecting a suitable number of rule patches with their positions and shapes in the input space. Initially the rule base consists of one single fuzzy rule and during the iterative learning process the rule base expands according to a supervised spawning validity measure. Kasabov et al. proposed *DENFIS*[16] (dynamic evolving neural-fuzzy inference system), for on-line and off-line learning, applied specially to time series prediction. The particularity of *DENFIS* is that it evolves through incremental, hybrid learning and accomodates new input data, including new features, new classes, etc. through local element tuning. Leng et al. proposed an algorithm for the generation of Takagi-Sugeno-type (TS) neuro fuzzy systems[20]. The algorithm consists of two stages: in the first stage, an initial structure is adapted from an empty neuron or fuzzy rule set, based on the geometric growth criteron and the ε-completeness of fuzzy rules; then, in the second stage, the initial structure is refined through a hybrid learning algorithm. We will refer to this method as "Leng's method".

Angelov et al. proposed an evolving Takagi-Sugeno model[3] (*eTS*). It is based on a learning algorithm that recursively updates the TS model structure and parameters by combining supervised and unsupervised learning. The rule base and the parameters evolve continuously by adding new rules if the potential of the new rules is higher than the potential of the existing ones. The potential is based on the modelling ability that the set of rules have on new data. Based on this principle *SAFIS*[34] was proposed. It is an algorithm based on the functional equivalence between a radial basis function network and a fuzzy inference system (FIS). They introduce the concept of influence of a fuzzy rule and use this to add or remove rules based on the input data received so far. Both of these algorithms are online learning algorithms, so the adaptation of the rules occurs whenever a new data point is received.

A Growing-Pruning algorithm for Fuzzy Neural Networks (FNN) that can add new fuzzy rules and eliminate useless fuzzy rules using a sensitivity analysis (SA) of the output from the model was proposed by Han et al. and was called *GP-FNN*[10]. The SA method has been successfully used by other researchers for neural network structure design[18]. The relevance of the fuzzy rules is determined by analyzing the Fourier decomposition of the variance of the FNN's output. Each contribution to the fuzzy rule is given by assigning a sensitivity ratio for measuring the contribution of a neuron (that could be interpreted as the premises of a fuzzy if-then rule) for generating the output value. The training algorithm for the parameters is implemented using a supervised gradient decent method, which ensures the convergence of the GP-FNN-learning process. An efficient self-organization learning neural network[40] (SOSLINN) was presented, which according to the authors present a fast and accurate model based on the significance evaluation of hidden networks with respect to the network output. There are several more applications of Self-Organizing Fuzzy Neural Networks to real-world problems[2,22,26].

In this work we propose a constructive learning algorithm inspired in the self-organization mechanism to simultaneously identify the structure and the weights of an ANFIS neuro-fuzzy system[13] with a data-driven approach. The self-organization mechanism searches for a suitable structure by means of attraction that merges correlated rules, repulsion that splits rules with unsatisfactory accuracy performance and creation of rules when there are subspaces without rule base representation. The algorithm behaves competitively for the learning of specific patterns, and, along with the iterations, it adds or prunes, constructively neurons to the model until the topology stabilizes. Hence, the model without being guided by an external source, like the user, starts to construct the fuzzy rules from the available data and then makes them compete to model it. This means that if there is expert knowledge, it can be used as a starting point of our algorithm. Nevertheless, if there is no expert knowledge, our algorithm has the ability to find a stable architecture that has a better performance than other (single) neural network systems reported in the literature with respect to approximation problems. This will be measured by means of Mean-Square Error and other performance measures presented in section 5. Some initial results[1] were later improved and made to the original split, merge and grow operators, and a formalization of their corresponding algorithms. Also an exhaustive experimentation was carried out, by validating the model with several real world and synthetic data sets, and meta-heuristics were applied in order to find optimal parameters. We show in section 5, that the model architectures obtained are sparse and parsimonious. It should be emphasized that our proposed model is mainly dealing with the problem of getting a good approximation performance, without aiming to obtain interpretable rules.

Although the concept of growing, splitting and pruning operations is not new in neuro-fuzzy models, we provide an alternative way to apply these kinds of operations. This proposal does not use the same criteria/methods to apply these operations. It only applies the same concept. Despite the concept of these operations not being new, by applying our own growing, splitting and prunning operations we outperform state of the art methods that use similar operations. Another thing to be pointed out is that these operations have intuitive fundamentals, relying on thresholds of number of examples, activation of rules and error perfomance. Another difference between these proposal and other state-of-the-art methods, is that most of them have an initial phase where they use clustering algorithms to create an initial set of rules. Our method does not rely on an initial clustering algorithm (see Section 4) and, if available, it allows the use of an initial set of rules defined by experts. This set of rules can be then improved by means of the proposed growing, splitting and pruning operators. Table 1 presents a qualitative comparison with state-of-the-art self-organizing and constructive models.

3. ANFIS: Adaptive-Network-Based Fuzzy Inference System

The *Adaptive Network-based Fuzzy Inference System* (*ANFIS*) was proposed by Jyh-Shing R. Jang[13]

Table 1. Table summarizing models

Model	Structure Search	Learning Process	Operators Foundation
SONFIN	Aligned clustering-based algorithm in the input space	Least Mean Squares (LMS) or Recursive Least Squares (RLS)	Initial input-space partitioning depends initially on the clustering algorithm and its parameters
GenSoFNN	Discrete Incremental Clustering in the initial phase followed by the use of an adding and prunning mechanism	Backpropagation learning	Initial input-space partitioning depends initially on the clustering algorithm and its parameters
SOFNN	Self-organizing clustering based on Rival Penalized competitive Learning	An improved backpropagation algorithm with adaptive learning rate and momentum	Initial input-space partitioning depends initially on the clustering algorithm and its parameters
GP-FNN	Self-organizing mechanism based on Prunning and Growing operators	Supervised gradient descent method	A set of fuzzy rules can be inserted or reduced during the learning process.
Leng's Method	Geometric Growth Criterion and ε-completeness	Hybrid Learning based on backpropagation and recursive weight learning algorithm	Initial input-space partitioning based on geometric growth criterion and ε-completeness
DENFIS	Evolving Clustering Method	Widrow-Hoff LMS Algorithm	Initial input-space partitioning based on evolving clustering method
SONFIS (Proposal)	Iterative simultaneous parameter and structure identification via Prunning, Spliting and Growing operations.	Hybrid Learning Algorithm: LMS and Backpropagation	No initial input-space partitioning. Partitioning performed iteratively with operators that update the structure which rely on intuitive heuristics

as an outcome of his doctoral thesis[12]. This model is functionally equivalent to a *Takagi-Sugeno-Kang* Inference System. Therefore, it is able to model continuous input/output relationships by means of fuzzy IF-THEN rules without employing precise quantitative analysis. This fuzzy inference model has been successfully used in several applications, e.g., to build function approximators[31,6], fuzzy controllers[5] and fuzzy classifiers[27]. There have also been developed many practical systems, such as prediction and inference[15], signal processing and communication systems[25,35], non-linear control[36,29], just to mention a few (Notice that none of the above mentioned applications had as a goal an interpretable if-then fuzzy rule-based system). The model is a fuzzy inference system implemented in the framework of adaptive neural networks that can construct an input-output mapping based on both human intelligence and data samples. In this section we present the architecture and the classical learning procedure of the original *ANFIS* model.

Let the input vector be $\mathbf{x} = (x_1, ..., x_d)' \in X$, $X \subseteq \mathbb{R}^d$ and d is the dimension of the input space, where v' is the transpose of the vector v, and let the target be $y \in Y$, $Y \subseteq \mathbb{R}$. The rule base consists of K fuzzy *if-then* rules of Takagi and Sugeno's type (see [37]), i.e. for each k rule we have:

Rule k: If x_1 is $A_1^{(k)}$ and ... and x_d is $A_d^{(k)}$, then
$f_k(\mathbf{x}, \theta) = \theta_1 x_1 + ... + \theta_d x_d + \theta_{d+1} = \Theta_k' \mathbf{x}$.

These rules are modeled with an ANFIS model whose architecture and execution are summarized as follows. The architecture of ANFIS consists of five layers of neurons. The first layer is used to associate the input variables to linguistic terms; a T-norm operator is employed in the second layer to obtain the strengths of the rules; and the third layer normalizes these rules strengths. The fourth layer computes the weighted hyperplane, where the consequents of the rules are determined. Finally, the output of the network is calculated as the summation of all the incoming signals pertaining to the previous layer.

The nodes of **layer 1** *(Fuzzification Layer)* compute the degree to which a given input x_i satisfies the linguistic quantifier $A_i^{(k)}$. The output of the node is given by the membership function $\mu_{A_i^{(k)}}(x_i)$. The Gaussian-type membership function[13] is used:

$$\mu_{A_i^{(k)}}(x_i; \eta_i^{(k)}) = \exp\left[-\left(\frac{x_i - v_i^{(k)}}{\sigma_i^{(k)}} \right)^2 \right] \quad (1)$$

where $\eta_i^{(k)} = \{v_i^{(k)}, \sigma_i^{(k)}\}$ are the *premise parameters* that should be estimated for the linguistic label $A_i^{(k)}$, the $v_i^{(k)}$ parameter stands for the location while $\sigma_i^{(k)}$ is the width of the membership function of the linguistic operator $A_i^{(k)}$.

The nodes of **layer 2** *(Generalized "AND Layer")* consist of $T-norm$ operators that perform the generalized AND. Each node of this layer represents the firing strength of some specific rule. In ANFIS the product $T-norm$ is used, because it is differentiable, a property that is necessary for using the backpropagation learning algorithm, which is a gradient descend algorithm.

$$w_k = w_k(\mathbf{x}; \eta^{(k)}) = \mu_{A_1^{(k)}}(x_1, \eta_1^{(k)}) \cdot ... \cdot \mu_{A_d^{(k)}}(x_d, \eta_d^{(k)}) \quad (2)$$

for $k = 1 \ldots K$.

Layer 3 *(Normalization Layer)* computes the *normalizing firing strength* of the weights of the previous layer: $w_k = \overline{w}_k(\mathbf{x}; \eta^{(k)}) = \frac{w_k}{\sum_{j=1}^{K} w_j}$, $k = 1..K$.

The nodes of **layer 4** *(Consequent Layer)* compute the weighted hyperplane that approximates the nonlinear mapping, i.e.,

$$\begin{aligned} \overline{f}_k(\mathbf{x}; \eta^{(k)}, \Theta_k) &= \overline{w}_k(\mathbf{x}; \eta^{(k)}) f_k(\mathbf{x}; \Theta_k) \\ &= \overline{w}_k \Theta_k' \mathbf{x} \end{aligned} \quad (3)$$

where $\overline{w}_k$ is the output of the k-th node of layer 3 and $\Theta_k = (\theta_1^{(k)}, ..., \theta_d^{(k)}, \theta_{d+1}^{(k)})'$ is the consequent parameter.

Finally, **layer 5** *(Network Output)* consists in a single node that computes the overall output as the sum of all the incoming signals:

$$g(\mathbf{x}; \eta, \Theta) = \sum_{k=1}^{K} \overline{w}_k(\mathbf{x}; \eta^{(k)}) f_k(\mathbf{x}; \Theta_k) \quad (4)$$

where $\eta = (\eta_1^{(1)}, ..., \eta_d^{(K)})'$ and $\Theta = (\Theta_1, ..., \Theta_K)'$ correspond to the premise and consequent set of parameters respectively.

To estimate the parameters, the classical ANFIS employs a hybrid learning procedure that uses the Ordinary Least-Square (OLS) estimation procedure to estimate the consequent parameters Θ and consecutively the backpropagation learning algorithm to determine the premise parameters η.

The OLS procedure estimates the consequent parameters for each rule separately by minimizing the following criterion for each node, $k = 1,\ldots,K$, of the layer 4:

$$\min_{\Theta_k} \frac{1}{N}(\mathbf{y} - \mathbf{x}_e\Theta_k)^T \Phi_i (\mathbf{y} - \mathbf{x_e}\Theta_k) \tag{5}$$

where $\mathbf{x}_e = [\mathbf{x}; 1]$ is the regressor matrix extended by a unitary column, N is the data set size and Φ_k is a matrix having the firing strengths (w_i) on its main diagonal

$$\Phi_k = \begin{bmatrix} w_{k,1} & 0 & \ldots & 0 \\ 0 & w_{k,2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & w_{k,N} \end{bmatrix} \tag{6}$$

Therefore, the least-square estimates of the consequent parameters are given by

$$\Theta_k = (\mathbf{x}_e^T \Phi_k \mathbf{x}_e)^{-1} \mathbf{x}_e^T \Phi_k \mathbf{y}, \text{ where } k = 1,\ldots K \tag{7}$$

The premise parameters $\eta_i^{(k)} = \{v_i^{(k)}, \sigma_i^{(k)}\}$ are estimated iteratively by the following updating rules:

$$v_i^{(k)}(t+1) = v_i^{(k)}(t) + 4\alpha(t)\frac{1}{(\sigma_i^{(k)})^2}(x_i - v_i^{(k)})\overline{w}_k z_k \tag{8}$$

$$\sigma_i^{(k)}(t+1) = \sigma_i^{(k)}(t) + 4\alpha(t)\frac{1}{(\sigma_i^{(k)})^3}(x_i - v_i^{(k)})^2\overline{w}_k z_k \tag{9}$$

where we denote $g = g(\mathbf{x}; \eta, \Theta)$, $z_k = (f_k - g)(y - g)$ and $f_k = f_k(\mathbf{x}; \Theta_k)$ for short. Moreover $\alpha(t)$ is the learning rate function, that behaves as the following equation:

$$\alpha(t) = \frac{\gamma}{\sqrt{\sum_\eta \left(\frac{\partial E}{\partial \eta}\right)^2}} \tag{10}$$

where γ, $0 < \gamma < 1$ is the step size, and it controls the speed of convergence.

4. SONFIS: Self-Organizing Neuro-Fuzzy Inference System

In this work we propose an extension to Jang's *ANFIS* model called *Self-Organizing Neuro-Fuzzy Inference System* (SONFIS).The basic structure of the proposal and its functionality is similar to the ANFIS model explained in the previous section. However, during the learning procedure, our proposed model self-organizes its architecture in order to automatically identify the number of premises and consequents needed to model the available complex and highly dimensional data.

The self-organization learning procedure consists of two stages. In the first stage we construct a base model with a predefined number of nodes and we estimate the parameters using the hybrid algorithm explained in the previous section. During the self-organization stage we proceed to apply three types of operators: *Grow Net* (GrowNet), *Split Subnetworks* (SplitNet) and *Vanish Sub-networks* (VanishNet). These operators are applied iteratively until the net self organizes and stabilizes satisfying some user's performance criterion. Before applying any operator, the current base model is frozen, meaning that none of the parameters can be any longer updated. From now on we will refer to each if-then Takagi-Sugeno Fuzzy rule as sub-networks. The sub-networks can be added, split or vanished.

The structure of a sub-network consists of the following elements: a set of nodes of the first layer that are Gaussian-type functions, one node for each dimension; a node of the second layer that computes the product of the incoming values; a normalization node of the third layer; the weighted regression line modeled by the node of the fourth layer; and all the incoming and outgoing links of aforementioned nodes.

The proposal can start either from a predefined number of nodes or from scratch. For example, if there are experts with some kind of knowledge related to the phenomena under analysis, their expertise can be used as base rules for our proposal. On the other hand, we can start from a model with no prior knowledge of the number of nodes necessary to satisfy some performance criteria.

In what follows, several learning parameters will be discussed, where their adjustments are based on "user-defined" values. With this we mean, values based on the experience of the designer or adjusted with heuristic methods (e.g. evolutionary algorithms). It should be remarked that if the designer adjusts the parameters, these are based on intuitive principles like percetange of data or number of examples. Both alternatives will be illustrated with examples in section 5.

The *Grow Net* operator consists in adding sub-networks to increase the granularity of the partition of the feature space. For each input $\mathbf{x}$ of the training data we compute the firing strength w_k for all the K sub-networks with equation (2). If the maximum of these strengths is less or equal than a user defined threshold δ to the power of d, where d is the dimension of the input space, i.e.,

$$\max_{k=1..K} w_k \leqslant \delta^d \tag{11}$$

then we say that the sample $(\mathbf{x}, y)$ is not well-modeled by the current model. We add it to a "bad samples" set, together with the information of the best matching criteria that currently best models the sample, i.e.,

$$w_\kappa = \arg\max_{k=1..K} w_k(\mathbf{x}, \eta^{(k)}) \tag{12}$$

After having revised all the training data, we group the data into the set $\mathscr{V}_\kappa$ according to their best matching criteria w_κ. For each group that has more than N_{grow} samples, where N_{grow} is user-defined, we construct and add a new sub-network for each dimension $\mu_{A_i^{(K+1)}}(x_i; \eta_i^{(K+1)}), i = 1..d$, with the premise parameters $\eta_i^{(K+1)} = \nu_i^{(K+1)}, \sigma_i^{(K+1)}$ initialized with the mean and standard deviation of the samples belonging to this group, i.e.,

$$\nu_i^{(K+1)} = \frac{1}{N_\kappa} \sum_{i=1}^{N_\kappa} x_i^{(\kappa)} \tag{13}$$

$$\sigma_i^{(K+1)} = \sqrt{\frac{1}{N_\kappa} \sum_{i=1}^{N_\kappa} \left(x_i^{(\kappa)} - \nu_i^{(K+1)} \right)^2} \tag{14}$$

The consequent parameters of the sub-network are randomly initialized.

The *Split Sub-network* operator consists in splitting a sub-network with bad performance into two new ones. To evaluate the sub-network performance, the training set is partitioned in K sets where the sample $(\mathbf{x}, y)$ is assigned to the set $\mathscr{V}_k$ if its best matching criteria (12) is w_k. For each set we compute the mean square error,

$$E_k = \frac{1}{N_k} \sum_{(\mathbf{x},y)\in\mathscr{V}_k} (y - g(\mathbf{x}; \eta, \Theta))^2 \tag{15}$$

where N_k is the number of samples belonging to the set $\mathscr{V}_k$ and $g(\mathbf{x}; \eta, \Theta)$ corresponds to the artificial neural network output given by equation (4). If the performance of sub-network k, E_k, is higher than a user defined threshold ε and N_k is higher than the minimum required samples N_{split}, where N_{split} is user-defined, then the sub-network is divided into two new ones. If the premise parameters of the k-th sub-network are ν and σ, then the premise parameters of the new sub-networks are:

$$\begin{array}{ll} \nu_i^{(K+1)} = \nu_i - \frac{\sigma_i}{2}; & \sigma_i^{(K+1)} = \frac{\sigma_i}{2}; \\ \nu_i^{(K+2)} = \nu_i + \frac{\sigma_i}{2}; & \sigma_i^{(K+2)} = \frac{\sigma_i}{2}; \end{array} \tag{16}$$

The hyperplane parameters are initialized randomly. After the inclusion of the new sub-networks, the k-th sub-network is eliminated.

The *Vanish Sub-network* operator consists in eliminating nodes that model less than N_{vanish} sample data, where N_{vanish} is user-defined. To accomplish this, we introduce an age_k variable that starts from zero and is increased by one if a sub-network models no data, i.e. the set $\mathscr{V}_k$ is empty. If the age variable of the k-th sub-network reaches the threshold λ, then the k-th sub-network is eliminated, where λ is user-defined. If the set $\mathscr{V}_k$ is no longer empty, then the age_k variable is set back to 0. After the application of the operators, all the unfrozen parameters (sub-networks) are updated according to

equations (5), (8) and (9). Afterwards, the whole net architecture is frozen. The operators and the training steps are applied iteratively until the architecture self-stabilizes and no longer changes. Finally, all the parameters (frozen and unfrozen) of the network are updated in the last iteration.

The *SONFIS* algorithm works as follows: At first the model must be initialized. The initialization consists in starting from a predefined number of nodes (if there is no prior knowledge the default is 1) and setting the user-defined thresholds. This step will provide an initial *SONFIS* architecture. After that, the algorithm will proceed with the iterations to establish a final parsimonious architecture.

In each iteration the current architecture must be frozen, so that the current nodes remain stable. The first operator to be evaluated is the GrowNet operator. This operator decides if the current number of nodes are having a good performance with all the training data. If not, this operator creates new subnetworks. The next step evaluates the SplitNet operator only if GrowNet has not added a new node. This operator splits a specific sub-network into 2 if it has a poor performance based on a defined threshold and if it does not model a minimum of data points.

The following step evaluates the VanishNet operator. This operator deletes a sub-network that has a poor performance or is modeling not a sufficient amount of data points. This step is crucial because it may eliminate nodes that could have been created by the other 2 operators and that are not making a relevant contribution to the performance of the model. After that, the next step adjusts the membership functions and hyperplane parameters of the sub-networks that the operators have created. The iterations stop when the operators fail to create new sub-networks. Finally the algorithm unfreezes the parameters of all the nodes of the *SONFIS* architecture and proceeds to adjust the premise and consequent parameters of all the nodes.

The order of complexity of the proposal can be defined by means of the following parameters of the model:

- Number of examples (N_e).
- Maximum number of subnetworks (N_s).
- Training epochs (E).
- Number of examples times the input dimension ($N_e \times dimension$).
- Vapnik-Chervonenkis dimension (VC).

Considering the aforementioned, the complexity of the model will be $O((N_e \times N_s \times E + N_e \times dimension)) \times VC$ Since the proposal iterates until the SONFIS architecture stabilizes, the complexity is also affected by how many times the model iterates. The number of iterations is strictly related by the complexity of the data that is being modeled. Regarding the convergence of the proposal, the user must be careful with the choice of several parameters. Each of the operations performed by the model (create, split and vanish subnetworks) has 2 user-defined parameters each. As stated previously, these parameters are intuitive, but still a bad choice of them will lead to several iterations, and will not garantee convergence. It has been established that it is necessary to specify the parameters according to the rule $N_{grow} \geqslant N_{split} \geqslant N_{vanish}$. The N_{grow} should be larger than the N_{split}, in order that the operations allow the network to create subnetworks in parts of the subspace where the model has not a good approximation. And N_{vanish} should be lesser that N_{split}, since it eliminates subnetworks that are modelling a small number of examples, so it enforces the creation of a parsimonius network. In other words a bad choice of these parameters will affect the model's stability.

5. Experimentation

In this section we present the performance of our proposed *Self-Organizing Neuro-Fuzzy Inference System* (*SONFIS*) model compared to the models: *Self-Constructing Neural Fuzzy Inference Network*[14] (*SONFIN*), a *Generic Self-Organizing Fuzzy Neural Network*[38] (*GenSoFNN*), *Self-Organizing Fuzzy Neural Network*[32] (*SOFNN*), the Leng's algorithm[20], *Dynamic Evolving Neural-Fuzzy Inference System* (*DENFIS*), *Genetic Dynamic Fuzzy Neural Network*[30] (*GDFNN*), *Novel hybrid algorithm for creating self-organizing fuzzy neural networks*[17] (*SOFNNGAPSO*), *Efficient Self-Organization Learning Neural Network*[40] (*SOSLINN*) and *Robust Fuzzy Regression Agglomeration*[8] (*RFRA*). The simulation studies were con-

ducted on both synthetic and real datasets. The latter were obtained from benchmark sites. In what follows, we first present all the synthetic and real data sets and the performance measures that we will compute; after that, we present numerical results and, at the end, we give some discussion about them.

5.1. Datasets

For the synthetic experiments we have created 2 synthetic data sets based on the following nonlinear functions:

$$f = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \tag{17}$$

and

$$y = \frac{\sin x_1 \sin x_2}{x_1 x_2}, \qquad -5 \leqslant x_1, x_2 \leqslant 5 \tag{18}$$

The training samples of the first synthetic experiment[32], consists of 216 uniformly sampled three-input data from input ranges $[1,6] \times [1,6] \times [1,6]$ and their corresponding target data. For the testing samples, 125 uniform samples from $[1.5,5.5] \times [1.5,5.5] \times [1.5,5.5]$ were used.

In the second synthetic experiment[8], 196 uniformly sampled two-input data from input ranges [-5,5] are used and their corresponding target data for training the model. Then, 392 testing data examples were generated, similar to the training data, to evaluate the generalization ability of the model.

Moreover Chuang et al. apply to equation (18) a gross error model[8] generated with a linear combination of a Gaussian Noise $G \sim N(0, \sigma_a)$ and an outlier generating distribution $H \sim N(0, \sigma_i)$ $F = G(1-\varepsilon) + \varepsilon H$ with $0 < \varepsilon < 1$, $\sigma_a = 0.01$ and $\sigma_i = 0.05$.

In the third synthetic experiment[20], we tested our approach with the well-known Mackey Glass Time-series, with the same experimental configuration. The training data are 1000 input-target data generated between $t = 124$ and 1123, and another 1000 input-target data between $t = 1124$ and 2123. The prediction model with the series by the following equation:

$$\widehat{x}(t+6) = f(x(t), x(t-6), x(t-12), x(t-18)) \tag{19}$$

For the real data experiments, we compare the performance of the techniques with the following datasets: The Laser Dataset that was contributed by Udo Huebner, Phys.-Techn. Bundesanstalt, Braunschweig, Germany. These data were registred from a Far-Infrared-Laser in a chaotic state[38]; The Wastewater dataset[32], Boston dataset and Building dataset were obtained from UCI Machine Learning repository[4]. The first real dataset[38] was obtained from the Forecasting and Time Series Analysis Santa Fe Institute Competition. The original dataset consists of 1100 observations of the fluctuations of infrared spectra in Laser. The data was separated in training and test data. The first 1000 samples were used as training data and the last 100 samples were used as testing data. The laser data time series has relatively simple oscillations, but has global events that are hard to predict (rapid decay of oscillations). The sample data consists of 5 inputs and 1 output given by the following autoregressive time series structure:

$$x(t) = F(x(t-1), x(t-2), x(t-3), x(t-4), x(t-5)) \tag{20}$$

where $x(t)$ is the output to be predicted, $\{x(t-1), x(t-2), x(t-3), x(t-4), x(t-5)\}$ is the set of 5 past observations and F is the nonlinear function that relates the next observations with the past ones. In the second real experiment a sludge wastewater treatment process is modeled. The key point to simulate and control this kind of treatment process is to establish the forecast model of the output-water quality. Input-output-water quality data from a sewage treatment plant in 2003 is taken as our original data[32]. The original data has 521 examples, but the authors claim to have deleted abnormal data. The resulting data has 330 examples. The first 300 samples are set as training data and the rest as test data. The experiment was conducted reducing the original size to 330. We used a median filter approach to obtain the 330 examples.

To compare *SONFIS* with other models we use the following performance measures: Mean Square Error (MSE), Root Mean Square Error (RMSE), Normalized Mean Square Error (NMSE), Average Percentage Error (APE), Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), Co-

	Model	MSE	MAE	R^2	CE	IA
Synthetic 1	ANFIS-4	5,025 ± 7,977	1,084 ± 0,620	0,876 ± 0,070	0,531 ± 0,759	0,907 ± 0,005
	FANN-4	0,588 ± 0,296	0,430 ± 0,151	0,952 ± 0,021	0,946 ± 0,026	0,995 ± 0,006
	SONFIS-4	0,218 ± 0,206	0,282 ± 0,146	0,982 ± 0,016	0,982 ± 0,016	0,997 ± 0,001
	DENFIS-45	0,1279 ± 0,0497	0,2342 ± 0,0473	0,976 ± 0,0113	0,9769 ± 0,0113	0,990 ± ≈ 0
Synthetic 2	ANFIS-9	0,0255 ± 0,0054	0,122 ± 0,014	0,675 ± 0,067	0,675 ± 0,067	0,793 ± 0,047
	FANN-9	0,0074 ± 0,0019	0,063 ± 0,009	0,904 ± 0,018	0,903 ± 0,019	0,972 ± 0,004
	SONFIS-9	0,0018 ± 0,0001	0,033 ± 0,001	0,976 ± 0,002	0,976 ± 0,002	0,993 ± 0,001
	DENFIS-21	0,0024 ± 0,0009	0,0346 ± 0,0052	0,970 ± 0,011	0,963 ± 0,0067	0,989 ± ≈ 0
Boston	ANFIS-5	592,11 ± 0,05	22,5 ± 0,00	≈ 0 ± ≈ 0	-6,250 ± 0,256	0,336 ± 0,001
	FANN-5	22,64 ± 4,14	3,1 ± 0,41	0,731 ± 0,051	0,722 ± 0,057	0,876 ± 0,044
	SONFIS-5	16,94 ± 1,23	3,0 ± 0,12	0,794 ± 0,016	0,794 ± 0,016	0,940 ± 0,015
	DENFIS-44	22,4254 ± 9,541	2,930 ± 0,3158	0,7469 ± 0,1061	0,7469 ± 0,1061	0,9505 ± ≈ 0
Building	ANFIS-5	0,0030 ± ≈ 0	0,0447 ± 0,0001	0,855 ± 0,000	0,855 ± 0,000	0,922 ± 0,054
	FANN-5	0,0028 ± 0,0002	0,0414 ± 0,0012	0,866 ± 0,012	0,866 ± 0,011	0,966 ± 0,002
	SONFIS-5	0,0030 ± ≈ 0	0,0447 ± 0,0001	0,855 ± 0,000	0,855 ± 0,000	0,960 ± 0,000
	DENFIS-82	0,0061 ± ≈ 0	0,0575 ± 0,0001	0,4363 ± 0,0071	0,4363 ± 0,0071	0,8592 ± ≈ 0
Wastewater	ANFIS-4	9142,9 ± 1,9	87,57 ± 0,0066	≈ 0 ± ≈ 0	-5,738 ± 0,327	0,324 ± 0,000
	FANN-4	328,2 ± 103,2	9,7107 ± 1,6220	0,806 ± 0,049	0,785 ± 0,059	0,927 ± 0,011
	SONFIS-4	47,1 ± 1,1	5,0075 ± 0,0411	0,967 ± 0,001	0,967 ± 0,001	0,991 ± 0,000
	DENFIS-108	116,01 ± 63,40	7,4178 ± 2,2666	0,7612 ± 0,1305	0,7612 ± 0,1305	0,9432 ± ≈ 0
Forest Fire	ANFIS-4	9142,9 ± 1,9	87,57 ± 0,007	≈ 0 ± ≈ 0	-5,738 ± 0,327	0,324 ± 0,000
	FANN-4	4145,6 ± 23,44	18,889 ± 0,9525	0 ± 0	-0,0595 ± 0,002	0,7441 ± 0,002
	SONFIS-4	4080 ± 34,4513	19,98 ± 0,188	0,0057 ± 0,0028	-0,150 ± 0,071	0,731 ± 0,0028
	DENFIS-102	10580 ± 2021	36,63 ± 2,472	≈ 0 ± ≈ 0	-9,072 ± 3,591	-1,623 ± 3,213
Concrete Strength	ANFIS-14	1561,7 ± ≈ 0	35,818 ± 0	≈ 0 ± ≈ 0	-4,634 ± 0,017	-0,417 ± 0,103
	FANN-14	36,910 ± 0,955	4,468 ± 0,0377	0,8668 ± 0,0026	0,8668 ± 0,0026	0,9632 ± 0,0093
	SONFIS-14	51,005 ± 15,24	4,711 ± 0,527	0,809 ± 0,058	0,809 ± 0,058	0,9351 ± 0,0189
	DENFIS-74	49,155 ± 6,0766	4,5367 ± 0,2182	0,8222 ± 0,0220	0,8222 ± 0,0220	0,9593 ± 0,0153

Table 2: Cross-validation experiments.

efficient of Determination (R^2), Coefficient of Efficiency (CE) and Index of Agreement (IA).

5.2. Results

The first part of this section, consists in presenting a complete study of 5 real benchmark datasets and 2 synthetic ones (Wastewater and both of the synthetic experiments that were presented above, the rest were obtained from the UCI Machine Learning repository). We compare 4 algorithms: Our proposed *SONFIS* algorithm, the classic *ANFIS*, the expanded high-order *DENFIS* algorithm and a Feedforward Neural Network (*FANN*). The results can be seen in Table 2. The architectural parameters of both *ANFIS* and *FANN*, were set according to the resulting Membership Functions created by our proposal (setting the corresponding number of neurons in the first layer and the number of hidden neurons respectively). The experimental configuration consisted in 4 rounds of a 5-fold cross-validation experiment. The cross-validation indices were generated randomly for each round.

The results of the experiments shown in Table 2 we can observe that *SONFIS* outperforms in all measures the classic *ANFIS* algorithm. If we compare *SONFIS* with *FANN* we can observe that in the Synthetic 1, Synthetic 2, Boston and Wastewater data sets, *SONFIS* outperformed the *FANN* algorithm in every measure. In only the Building and Concrete Slump Strength data sets, the *FANN* algorithm outperformed *SONFIS*. Nevertheless, the results are comparable between these two algorithms. It should be pointed out that the parameters chosen for the *SONFIS* algorithm were the same used in the experimentation of synthetic data set 2. We did not perform an exhaustive search for the values of this parameters. If we notice the number of sub-networks between *SONFIS* and *DENFIS*, clearly *SONFIS* algorithm generates less of them, which gives us a more parsimonious model. It should be noted that in most cases *SONFIS* outperforms *DENFIS*, and it does it with a smaller number of subnetworks. In some cases, the opossite occurs but the *DENFIS* model in all cases creates a larger final network. Despite the fact, that in some cases *DENFIS* outperforms *SONFIS*, the results obtained by the proposal are comparable, and it accomplishes it with a much smaller number of subnetworks.

The second part of the numerical results consists in conducting the experiments following the methodology that were reported in the original articles of the methods. The results obtained by *SONFIS* were compared to *GDFNN*, *SOFNN* and *SOFNNGAPSO* in the first synthetic experiment and are shown in Table 3. *SONFIS C-V* is the same proposed algorithm, applying a 5-fold cross-validation experimental approach. Details will be given in the next subsection.

Table 3. Performance of *SONFIS*, *GDFNN*, *SOFNN* and *SOFNNGAPSO* in the synthetic experiment 1.

Algorithm	APEtrain	APEtest	Training Epochs
SONFIS	0.750	0.650	140
GDFNN	2.110	1.54	160
SOFNN	0.560	2.320	200
SOFNNGAPSO	1.210	1.240	90
SONFIS C-V	0.937	0.939	100

The results from the second synthetic experiment are presented in Table 4. We present the comparison between *RFRA*, *SONFIN* and *SONFIS*.

Table 4. Performance of *RFRA*, *SONFIN* and *SONFIS* in the synthetic experiment 2.

Model	$RMSE_{test}$
RFRA	0.0353
SONFIN	0.0575
SONFIS	**0.0221**
SONFIS C-V	0.0298

In Table 5 we present the values of the parameters used in the synthetic experiments. In these cases, we used an empirical approximation to obtain the resulting parameters.

Table 5. Parameters used in the *SONFIS* Operators.

Data	ε	N_{split}	δ	N_{grow}	λ	N_{vanish}
Synt1	0.5	40	0.8	30	2	30
Synt2	0.7	30	0.8	20	2	20

Table 6. RMSE of *SOFNN*, Leng's method [20], *DENFIS*, *SOSLINN* and *SONFIS* for the Mackey Glass Timeseries.

Model	*neurons*	$RMSE_{train}$	$RMSE_{test}$
SOFNN	**4**	0.0123	0.0118
Leng's method	10	0.0084	0.0088
DENFIS	4	0.0048	0.0134
SOSLINN	6	0.0065	0.0061
SONFIS	8	**0.0024**	**0.0051**

Now we present the results obtained with the first real dataset (Laser dataset). In Ref. 38 they compare their proposed model *GenSoFNN* with Feedforward Neural Networks *FANN*, with different topological definitions. The results can be seen in Table 7. We compare the results of our proposal with the results obtained in Ref. 38. *SONFIS GA* is the same proposed algorithm, applying a Genetic Algorithm to find the optimal parameters. Further details will be given in the next subsection.

Table 7. Performance of *FANN*, *GenSoFNN* and *SONFIS* for the Laser Time Series.

Model	$NMSE_{test}$
Feedforward(200-100-1)	0.770
Feedforward(50-20-1)	1.000
Feedforward(50-350-50-1)	0.380
GenSoFNN	0.244
SONFIS	0.004
SONFIS GA	**0.003**

As a performance measure in the Wastewater experiment, [32] presented the APE and RMSE as performance measures.

The results of *SOFNN*, *ANFIS* and *SONFIS* are presented in tables 8 and 9. The data used with *SONFIS* was with and without outliers.

Table 8. APE of *SOFNN*, *ANFIS* and *SONFIS* for the Wastewater dataset.

Model	APE_{train}	APE_{test}
SOFNN	3.77%	3.98%
ANFIS	3.99%	4.38%
SONFIS	**2.64**%	**3.14**%

Table 9. RMSE of *SOFNN*, *ANFIS* and *SONFIS* for the Wastewater dataset.

Model	$RMSE_{train}$	$RMSE_{test}$
SOFNN	3.717	3.871
ANFIS	3.909	4.135
SONFIS	**2.873**	**3.042**

The results obtained in the first synthetic experiment (Table 3) show that the *SOFNN* algorithm outperforms *SONFIS* in the training data, where *SOFNN* obtains an APE of 0.56 and *SONFIS* 0.75; but if we observe the test results we can clearly see that *SONFIS* obtains a much lower APE than *SOFNN*, which seems to be affected by overfitting. *SONFIS* outperforms *GDFNN* and *SOFNNGAPSO* in both Train and Test results. The results of *SONFIS* are the mean value of 20 experimental runs. The minimum obtained by *SONFIS* in the training data was 0.63 and the minimum in the testing data was 0.52. Quiao and Wang[32] report, 200 epochs of training for *SOFNN*. The number of training epochs for *SONFIS* was 140, i.e. 60 less than *SOFNN*. Also with *SONFIS* a 5-fold cross-validation was performed (*SONFIS C-V*). As can be seen in the table *SONFIS C-V* also outperforms *SOFNN* in the testing phase. The training epochs used are remarkable, because this result is obtained with only 100 epochs, exactly half of the epochs required by *SOFNN*.

The results from the second synthetic experiment are presented in Table 4. To measure the performance of the models the Root Mean Squared Error (RMSE) was used. The performance results obtained by *SONFIS* are clearly better that the ones obtained with *RFRA* and *SONFIN*. The RMSE reported is the mean value of 20 experimental runs. The best test performance obtained by *SONFIS* was 0.0160. The mean value of the experimental runs with the training data was 0.010. Also the mean value of the number of neurons in the first layer obtained was 6. It is also important to clarify that the mean value of the training epochs was 140. As in the synthetic experiment 1, a 5-fold cross-validation experiment with *SONFIS* was performed, reporting a test RMSE of 0.0416. The resulting training epochs were 200. In this case even *SONFIS C-V* outperforms *RFRA* and *SONFIN* in terms of the RMSE test error.

The results from the Mackey Glass Time series are presented in Table 6. It shows the results of the algorithms *SOFNN*, Leng's Method[20], *DENFIS*, *SOSLINN* and *SONFIS*. The RMSE results obtained by *SONFIS* is the mean of 20 experimental runs. The best train and test performance obtained by *SONFIS* were 0.0016 and 0.0043, respectively. The authors of Leng's Method[20] do not report the number of experimental runs. As can be seen, *SONFIS* outperforms the four other algorithms in the training and testing RMSE. The authors of Leng's Method[20] test their proposed algorithm with several other algorithms. Please refer to this paper for further details about the other algorithms. Our proposed algorithm *SONFIS* was also compared with the *DENFIS*[16] algorithm and *SOSLINN*[40].

In Table 7 we present the normalized mean squared error (NMSE) of *SONFIS*, *GenSoFNN*[38] and 3 different feedforward neural networks models with distinct topological configurations for the Laser dataset. *SONFIS* outperforms *GenSoFNN* and the Feedforward Neural Networks in 2 orders of magnitude. 20 experimental runs were conducted reporting the mean test NMSE. The mean train NMSE obtained was 0.0002. The authors of *GenSoFNN* do not report the number of experimental runs, so we have to assume that the results reported are the best ones. Our minimum NMSE for the test data was 0.001. The parameters of *SONFIS* were obtained empirically. In *SONFIS GA* we used a real-valued Genetic Algorithm to find the optimal parameters of the algorithm. We obtained a better performance of the model with these. The configuration of the Genetic Algorithm (GA) experiment was the following: 3 different configuration for the mutation and cross-over probabilities, 10 populations and 4 individuals per population in each experiment. As it was expected for an evolutionary approach, a large amount of time is necessary to perform these experiments. The best result for the $NMSE_{test}$ was 0.0003. The ε parameter obtained was 0.453428 and a δ of 0.743895. As can be seen in tables 8 and 9 the *SONFIS* model has a better performance compared to *SOFNN* and *ANFIS* in the Wastewater dataset in terms of APE and RMSE. 20 experimental runs were conducted and the mean value was reported. The mean value of the resulting neurons of the first layer was 4. The results of *SONFIS* with the filtered data are significantly better than the ones obtained with *SOFNN* and *ANFIS*. The starting solution for the first and second real experiment were the values of the parameters obtained empirically, described in more detail below.

The parameters were chosen empirically for the real-world benchmark dataset: $\varepsilon = 0.3$, $N_{split} = 30$, $\delta = 0.7$, $N_{grow} = 20$, $\lambda = 2$ y $N_{vanish} = 20$.

5.3. Discussion with state of the art methods

In this paper we have introduced a flexible architecture algorithm inspired on *ANFIS*, called *SONFIS*. Our proposed model self organizes its architecture in order to automatically identify the number of premises and consequents needed to model the available complex data. The adaptation is performed introducing Takagi and Sugeno's fuzzy if-then rules, based on the evaluation of self-organization operators with a data-driven approach, that can improve the fitting performance in the training phase, leading to a better performance in terms of generalization ability.

As a starting point, expert knowledge can be incorporated to initialize the model structure, and our proposed approach is able to improve it iteratively, establishing a proper architecture based on the samples available. This is a key point while comparing our proposal with other state of the art methods. Most of them generate an initial set of rules based on a clustering method, partitioning the input-space. Then with this initial set of rules, they refine the search for suitable rules, applying other methods (Mostly prunning, spliting and growing operations). Our method does not need an initial set of rules derived from the data set, because it constructs it iteratively, with the aforementioned operations. Therefore, state-of-the-art methods cannot use previously acquired or expert knowledge, because they do not take into account the knowledge when they first derive the rules from the data. Also it should be remarked that the operations used in this proposal have intuitive fundamentals, relying on thresholds or parameters that can be easily interpreted by not so experienced users. They represent number of exam-

ples, percentages of data that "belong" to a membership function and user-defined maximum allowed error performance. So the meaning of our proposal parameters are more interpretable by the users. Another advantage of our proposal is that it does not create a huge number of rules, if we compare it for example with the *DENFIS* algorithm. Also the number of rules can be easily controlled by tuning the algorithm parameters properly. This is an advantage, because it creates a more parsimonious model, thus using less computational resources. Despite all the advantages of our proposed model, we have to clarify that our model, is sensible in the sense that if the user-defined parameters are not well-defined it can get stuck in several iterations, creating and pruning rules indefinitely. This can be partially solved by having a threshold parameter that stops the algorithm when the number of iterations reach that threshold parameter. Also, if the data is very noisy (or even with many outliers), the parameter of the number of examples necessary to create new subnetworks, will be insufficient to detect such scenarios thus, the model will not be able to model the problem properly.

Our algorithm shows better results in both synthetic and real data sets in most cases, when compared with other State of the Art algorithms. The experimentation consisted in 2 parts. The first part consisted in performing an exhaustive comparative analysis in order to contrast our algorithm with the classic *ANFIS*, *DENFIS* and *FANN* models. The second one was conducted to compare our proposal with other well-known self-organizing methods, where the results obtained were remarkable. In this part we experimented with two real world benchmark data known as Laser and Wastewater. The comparative study with the classic self-organizing algorithms shows that our algorithm outperforms the alternative models with statistical significance, obtaining good results in both synthetic and real data.

6. Conclusion

As a conclusion to this work we can say that an a priori fixed number of fuzzy rules is not necessary. This is a useful improvement in applications where an inexperienced user needs to work with Neural-based models without knowing the optimal net architecture. Also, another particularity of this method, in comparison with other neuro-fuzzy constructive methods, is that our proposed model has not only the ability of self-organizing based on a data-driven approach, but also to start from an a priori knowledge base set by an expert or by other methods, leading to an algorithm that is more capable of facing different kinds of problems. According to the "No free lunch" theorem, it is known that for every specific problem we have to find some adequate parameters, because it is not possible to find a *super*-model that is able to model every data set in an optimal way. In future works we will deal with problems related with big data and data streams.

Acknowledgement

This work was supported by the following research grants: Fondecyt 1110854, Fondecyt Initiation into Research 11150248, CCTVal, DGIP-UTFSM, DIUV 64-2011 and UVA1402 of the Ministry of Education - Chile. The work of C. Moraga was partially supported by the Foundation for the Advancement of Soft Computing, Mieres, Spain.

1. Allende-Cid H., Veloz A., Salas R., Chabert S. and Allende H.: Self-organizing neuro-fuzzy inference system. Progress in Pattern Recognition, Image Analysis and Applications, 13th Iberoamerican Congress on Pattern Recognition, CIARP 2008, pp. 429-436.
2. Alvarez-Molina E.R., Martinez L.G., Castanon-Puga M. and Rodriguez-Diaz, A. A Neuro-Fuzzy System as a complex system of emergent behavior in organizations. Complex Systems (WCCS), 2014 Second World Conference on, 2014, pp. 463 - 468.
3. Angelov P.P. and Filev D.P.: An approach to online identification of Takagi-Sugeno fuzzy models. IEEE Transactions Systems, Man and Cybernetics, Part B, Vol.34 (1), 2004, pp. 484-498.
4. Asuncion A. and Newman D.J.: UCI machine learning repository, 2007.
5. Boukezzoula R., Foulloy L. and Galichet S.: Inverse controller design for fuzzy interval systems. IEEE Transactions on Fuzzy Systems, Vol. 14(1), 2006, pp. 569-582.
6. Chen W. and Saif M.: A novel fuzzy system with dynamic rule base. IEEE Transactions on Fuzzy Systems, Vol. 13(5), 2005, pp. 569-582.

7. Cheng B. and Titterington D.M.: Neural networks - a review from a statistical perspective. Statistical Science, Vol. 9(1), 1994, pp. 2-30.
8. Chuang C.C., Su S.F. and Chen S.S.: Robust TSK fuzzy modeling for function approximation with outliers. IEEE Transactions on Fuzzy Systems, Vol. 9(6), 2001, pp. 810-821.
9. Cotrell M., Girard B., Girard Y., Mangeas M. and Muller C.: Neural modeling for time-series - a statistical stepwise method for weight elimination. IEEE Transactions on Neural Networks, Vol. 6(6), 1995, pp. 1355-1364.
10. Han H. and Qiao J.: A self-organizing fuzzy neural network based on a growing-and-pruning algorithm. IEEE Transactions on Fuzzy Systems, Vol. 18(6), 2010, pp. 1129 -1143.
11. Jakubek S. and Hametner C.: Identification of neurofuzzy models using GTLS parameter estimation. IEEE Transactions on System, Man and Cybernetics, Part B, Vol. 39(5), 2009, pp. 1121-1133.
12. Jang J.S.R.: Neuro-Fuzzy Modeling: Architecture, Analyses and Applications. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, U.S.A., 1992.
13. Jang J.S.R.: ANFIS: Adaptive-network-based fuzzy inference system. IEEE Transaction on Systems, Man and Cybernetics, Vol. 23(3), 1993, pp. 665-685.
14. Juang C.F. and Lin C.T.: An on-line self-constructing neural fuzzy inference network and its applications. IEEE Transactions on Fuzzy Systems, Vol. 6(1), 1998, pp. 12-32.
15. Kandel A.: Fuzzy Expert Systems. CRC Press, 1992.
16. Kasabov N. and Song Q.: DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Transactions on Fuzzy Systems, Vol. 10(2), 2002, pp. 144-154.
17. Khayat O., Ebadzadeh M.M., Shahdoosti H.R., Rajaei R. and Khajehnasiri I.: A novel hybrid algorithm for creating self-organizing fuzzy neural networks. Neurocomputing, Vol 73(1), 2009, pp. 517524.
18. Lauret P., Fock E. and Mara T.A.: A node pruning algorithm based on a Fourier amplitude sensitivity test method. IEEE Transactions on Neural Networks, Vol. 17(2), 2006, pp. 273-293.
19. Leng G., Prasad G. and McGinnity T.M.: An online algorithm for creating self-organizing fuzzy neural networks. Neural Networks, Vol. 17(10), 2004, pp. 1477-1493.
20. Leng G., Zeng X.-J. and Keane J.A.: A hybrid learning algorithm with a similarity-based prunning strategy for self-adaptive neuro-fuzzy systems. Applied Soft Computing, Vol. 9(1), 2009, pp. 1354-1366.
21. Liu Z.Q., Guan T. and Zhang Y.J.: Self-spawning neuro-fuzzy system for rule extraction. Soft Computing, Vol. 13(11), 2009, pp. 1013-1025.
22. Mozaffari A., Fathi A. and Behzadipour S.: An evolvable self-organizing neuro-fuzzy multilayered classifier with group method data handling and grammar-based bio-inspired supervisors for fault diagnosis of hydraulic systems. International Journal of Intelligent Computing and Cybernetics. Vol. 7(1), 2014, pp. 38-78.
23. Nash J. and Sutclife J.: River flow forecasting through conceptual models part I - a discussion of principles. Journal of Hydrology, Vol. 10(3), 1970, pp. 282-290.
24. Nelles O., Fischer M. and Muller B.: Fuzzy rule extraction by a genetic algorithm and constrained nonlinear optimization of membership functions. In Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, Vol. 1, 1996, pp. 213-219.
25. Nie J. and Linkens D.A.: Fuzzy Neural Control: Principles, Algorithms and Applications. Prentice-Hall, 1994.
26. Ocampo-Duque W., Juraske R., Kumar V., Nadal M., Domingo J.L. and Schuhmacher M.: A concurrent neuro-fuzzy inference system for screening the ecological risk in rivers. Environmental Science and Pollution Research, Vol. 19(4), 2013, pp. 983-999.
27. Pal S.K. and Mitra S.: Multi-layer perceptron, fuzzy sets and classification. IEEE Transactions on Neural Networks, Vol. 3(5), 1992, pp. 683-697.
28. Parekh R., Yang J. and Honavar V.: Constructive neural-network learning algorithms for pattern classification. IEEE Transactions on Neural Networks, Vol. 11(2), 2000, pp. 436-451.
29. Pedrycz W.: Fuzzy Control and Fuzzy Systems. Wiley Series, 1989.
30. Pratama M., Er M.J., San L., Richard J.O., Zhai L.-Y., Torabi A. and Arifin I.: Genetic Dynamic Fuzzy Neural Network (GDFNN) for Nonlinear System Identification. Lecture Notes in Computer Science, Vol. 6676, 2011, pp. 525-534.
31. Pomares H., Rojas I., Ortega J., Gonzalez J. and Prieto A.: A systematic approach to a self-generating fuzzy rule-table for function approximation. IEEE Transactions System Man and Cybernetics, Part B, Vol. 30(3), 2000, pp. 431-447.
32. Qiao J. and Wang H.: A self-organizing fuzzy neural network and its applications to function approximation and forecast modeling. Neurocomputing, Vol. 71, 2008, pp. 564-569.
33. Qiao J.-F. and Han H.-G.: Identification and modelling of nonlinear dynamical systems using a novel self-organizing RBF-based approach. Automatica, Vol 48(8), 2012, pp. 17291734.
34. Rong H.-J., Sundararajan N., Huang G.-B. and Saratchandran P.: Sequantial adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. Fuzzy Sets and Systems, Vol. 157(1), 2006, pp. 1260-1275.

35. Sanver M. and Karahoca A.: Fraud detection using an adaptive neuro-fuzzy inference system in mobile telecommunication networks. Journal of Multiple-Valued Logic and Soft Computing, Vol. 15(2-3), 2009, pp. 155-179.
36. Sugeno M.: Industrial Applications of Fuzzy Control. Elsevier, 1985.
37. Takagi T. and Sugeno M.: Derivation of fuzzy control rules from human operator's control actions. In in Proc. IFAC Symposium Fuzzy Information, Knowledge Representation and Decision Analysis, 1983, pp. 55-60.
38. Tung W.L. and Quek C.: GenSoFNN: A generic self-organizing fuzzy neural network. IEEE Transactions on Neural Networks, Vol. 13(5), 2002, pp. 1075-1085.
39. Willmott C.J.: On the validation of models. Phys. Geogr., Vol. 2(1), 1981, pp. 184-194.
40. Yang G., Yang H. and Dai L.: Time-series prediction modelling based on an efficient self-organization learning neural network. IFAC-PapersOnLine, Vol. 48(8), 2015, pp. 248253.