Evolving Systems (2021) 12:303–317
https://doi.org/10.1007/s12530-019-09278-5

ORIGINAL PAPER

Self-organized direction aware for regularized fuzzy neural networks

Paulo Vitor de Campos Souza[1] · Cristiano Fraga Guimaraes Nunes[2] · Augusto Junio Guimares[3] · Thiago Silva Rezende[3] · Vanessa Souza Araujo[3] · Vincius Jonathan Silva Arajuo[3]

Received: 6 July 2018 / Accepted: 4 March 2019 / Published online: 15 March 2019

Abstract

The fuzzy neural networks are efficient hybrid structures to perform tasks of regression, patterns classification and time series prediction. To define its architecture, some models use techniques that fuzzification of data that can divide the sample space in grid format through membership functions. The models that use such techniques achieve results with a high degree of accuracy in their activities, but their structures can vary greatly when the number of features of the problem is high, making of fuzzy neurons an exponential relationship between the number of inputs and the membership functions numbers used in the model of the input space. A multi-neuron structure can make the training and update of parameters damaging to the model's computational performance, making it impossible to work with problems of high dimensions or even with a high number of samples. To solve the problem of the creation of structures of hybrid models based on neural networks and fuzzy systems this paper proposes the use of a novel fully data-driven algorithm. This algorithm uses an extra cosine similarity-based directional component to work together with a traditional distance metric and nonparametric Empirical Data Analytics to data partitioning and forming data clouds in the first layer of the model. Another problem that exists in fuzzy neural network models is that some of their parameters are defined at random, so they challenging to interpret and can introduce casual situations that may impair model responses. In this paper we also propose the definition of bias and weights of the neurons of the first layer using the concepts of the wavelet transform, allowing the parameters of the neurons also to be directly related to the input data submitted to the model. In the second layer, the unineurons aggregate the neurons generated in the first layer and a regularization function is activated to determine the most significant unineurons. The weights used in the third layer, represented by an artificial neural network with an activation function of type ReLU, are generated using the concepts of the extreme learning machine. To verify the new training approach for fuzzy neural networks, tests with real and synthetic databases were performed for pattern classification, which led to the conclusion that the cloud-based approach and neuron weights generation based on the data frequency of training proves that the accuracy of the model is adequate to perform binary classification problems.

Keywords Fuzzy neural networks · SODA · Wavelets · ReLU · Pattern classification

* Paulo Vitor de Campos Souza
pauloc@prof.una.br; paulovitor@cefetmg.br

Cristiano Fraga Guimaraes Nunes
cfgnunes@gmail.com

Augusto Junio Guimares
augustojunioguimaraes@gmail.com

Thiago Silva Rezende
silvarezendethiago@gmail.com

Vanessa Souza Araujo
v.souzaaraujo@yahoo.com.br

Vincius Jonathan Silva Arajuo
vinicius.j.s.a22@hotmail.com

[1] Faculty Una of Betim CEFET-MG, Belo Horizonte, Brazil

[2] Postgraduate Program in Mathematical and Computational Modeling CEFET-MG, Belo Horizonte, Brazil

[3] Faculty Una of Betim, Betim, Brazil

1 Introduction

The fuzzy neural networks (FNN) are hybrid models based on the incorporation of fuzzy systems, which are capable of generating interpretability to the results, with the generalist capacity of artificial neural networks, which have several training techniques to solve problems that normally humans act. These structures have been applied in several contexts in the area of artificial intelligence, such as binary patterns classification (de Campos Souza et al. 2018; de Campos Souza and de Oliveira 2018; Lughofer et al. 2018; Lughofer 2012), regression (Juang et al. 2010), time series forecasting (Han et al. 2018; Bordignon and Gomide 2014; Rosa et al. 2013; de Campos Souza and Torres 2018), rainfall (Sharifian et al. 2018), financial market (Rosa et al. 2014), software effort estimation (Souza et al. 2018), failures prediction in some engineering contexts (Song et al. 2018; Tang et al. 2017; de Jesús Rubio 2018, 2017) and so on.

The architecture of fuzzy neural networks have layers that can perform various tasks. Generally, the first layer is responsible for partitioning the input data according to the chosen fuzzy technique. Fuzzy neurons are constructed according to training data and may generate fuzzy rules for the construction of expert systems (Buckley and Hayashi 1994). In the second layer, the updating of the parameters involved may involve techniques such as backpropagation, gradient descent (Amari 1993) and extreme learning machine (Huang et al. 2006), which consists in determining parameters of the hidden layers of the networks at random and calculate the final weights using least squares concepts. The second layer may contain artificial neurons or neural logic neurons. These neurons enable the transformation of model elements into if/else fuzzy rules. The neurons *and*, *or*Pedrycz and Gomide (2007), *unineurons*, Pedrycz (2006) and *nullneuron*, Hell et al. (2008) are highlighted as neurons with this capacity. Evolutionary and genetic approaches are also used. Finally, these models use a neural network of aggregation with artificial neurons to carry out their responses. In general, neurons use activation functions commonly known to obtain the final network output.

When verifying that fuzzy neural networks suffer from problems related to the number of neurons, regularization techniques were incorporated into the models, allowing the less significant neurons to be discarded from the model. In particular techniques such as the regression ridge (Tikhonov et al. 2013), LARS (Hansen 1982) and the bootstrap lasso (Bach 2008) are employed to define the architecture of fuzzy neural networks. This paper presents a new training model in fuzzy neural networks where the first layer of the model has its fuzzy neurons with the synaptic weights and bias defined by wavelet transform functions (Daubechies 1990). The Gaussian membership functions of the fuzzy neurons in the first layer are defined by an algorithm fully data-driven called SODA (Self-Organized Direction Aware) (Gu et al. 2018). This algorithm applies the concept of a directional component based on extra cosine similarity to work in conjunction with a traditional distance metric. In summary, SODA uses nonparametric Empirical Data Analysis (EDA) (Angelov et al. 2017) operators to automatically identify the critical modes of the data pattern from the empirically observed training samples and uses them as focal points to form data clouds. The second layer of the model is composed of unineurons that perform the aggregation of the fuzzy neurons of the first layer. In order to eliminate unnecessary neurons to the model, the algorithm bolasso Bach (2008) to eliminate neurons using the lasso method according to a decision consensus and some bootstraps. Finally, the artificial neural network is present in the third layer of the model, but different from the model of de Campos Souza et al. (2018), where linear activation functions are used, the concepts of rectified linear units (ReLU) are used (Maas et al. 2013).

This type of approach using logical neurons that aggregate neurons formed by cloud techniques allows a more significant number of input data to be worked by the model in a time less than exponential approaches proposed by models that have fuzzification processes based on the model Anfis (Jang 1993).

To verify the capacity of the new model, binary pattern classification tests will be performed in order to evaluate aspects of model accuracy. The paper is organized as follows: Sect. 2 presents the main concepts that guide the research, such as the definitions of fuzzy neural networks, wavelets, regularization and activation functions. Section 3 will present the steps and concepts related to the methodology proposed to generate the first layer weights of the FNN based on the wavelet transform, and the concepts of SODA to construct the first layer neurons, in addition to the artificial neurons based on the activation functions of type ReLU to perform the of binary patterns classification in the output of the model. Section 4 will present the methodology used in the tests, including the bases, and the algorithms used to perform the binary pattern classification. Finally, in Sect. 5 the conclusions of the work will be presented.

2 Literature review

2.1 Fuzzy neural network

Over the last few decades, fuzzy systems and their hybrid derivations have been shown to be able to simulate the typical human reasoning ability in a computationally efficient way. An important area of current research is the development of such systems with a high level of flexibility and autonomy to evolve their structures and knowledge based on changes in the environment, being able to handle modeling, control, prediction and classification of patterns in a situation not stationary, susceptible to constant changes. Fuzzy neural networks are characterized by neural networks composed of fuzzy neurons (Pedrycz and Gomide 2007). The motivation for the development of these networks lies in its easy interpretability, being possible to extract knowledge from its topology. These networks are formed by a synergistic collaboration between fuzzy set theory and neural networks allowing a wide range of learning abilities, thus providing models that integrate the uncertain information handling provided by the fuzzy systems and the learning ability granted by the neural networks (Pedrycz 1991). Thus a Fuzzy neural network can be defined as a fuzzy system that is trained by an algorithm provided by a neural network. Given this analogy, the union of the neural network with the fuzzy logic comes with the intention of softening the deficiency of each of these systems, making us have a more efficient, robust and easy to understand a system.

2.2 Fuzzy neural networks models

FNNs are composed of logical neurons, which are functional units that add relevant aspects of processing with learning capacity. They can be seen as multivariate nonlinear transformations between unit hypercubes (Pedrycz 1991). Studies propose the generalization of logical neurons *and* and *or* that are constructed through extensions of *t-norms* and *s-norms*. One of the most important features of these neurons, called *unineurons*, Pedrycz (2006) and *nullneurons*, Hell et al. (2008), are their ability to vary smoothly from a neuron *or* to *and* and vice versa, depending on the need for the problem to be solved. This causes the final structure of the network to be determined by the training process, making this structure more general than fuzzy neural networks formed only by classical logical neurons.

These intelligent models have an architecture based on multilayer networks, where each one of them has different functions for the activities carried out. The layers of a fuzzy neural network can act as fuzzification, transforming numerical data into representations of fuzzy sets, other layers can perform with the defuzzification making the inverse process (convert fuzzy sets into numerical values). Some layers have with fuzzy rules, where they are usually called of fuzzy inference systems and layers representing neural aggregation networks. Each model has layers and different training techniques to solve problems. As examples of three-layers architectures, the proposals of Souza 2018, de Campos Souza and Torres (2018), Guimarães et al. (2018), de Campos Souza et al. (2018) and Guimaraes et al. (2018). Already the models that have four and five layers in its structure, we can highlight the models of Lin et al. (2018) and Kasabov (2001) respectively. In most models, the first layer is the one that partitions the input data, transforming them into fuzzy logical neurons. Algorithms common to these approaches are fuzzy c-means (Bezdek et al. 1984), clouds (Koutrika et al. 2009) and functions based on ANFIS techniques (Jang 1993). The fuzzy neural networks may also present training characteristics based on recurring functions (Yen et al. 2018; Ballini and Gomide 2002), evolving concepts (Silva et al. 2014; Rosa et al. 2013, 2014) and contours-correlated functions (Ebadzadeh and Salimi-Badr 2018).

In this paper, the highlight will be the extreme learning machine (Huang et al. 2006) in conjunction with fuzzy data processing techniques in the first layers. These approaches have already been used in models such as (Souza 2018; de Campos Souza and Torres 2018; Lemos et al. 2012; Rong et al. 2009), differing from the model proposed in this paper by the type of algorithm used for the fuzzification process.

The main difference will be between the change of the ANFIS model (Jang 1993) that uses equally spaced membership functions to a cloud approach. The nature of the input data of the model will have greater significance for the construction of the neurons than an exponential relationship of grid division proposed by models that are based on the techniques of the division of the sample space. This will allow the fuzzification technique used in the fuzzy neural network to create the number of neurons in the first layer much lower when compared to the approaches that use the ANFIS. In techniques that the main fuzzification parameters are based on structures of pertinence functions, many neurons may represent empty or inexpressive spaces for the problem. The SODA technique works with the representativeness of the data, allowing only representative neurons to be created according to the density of the data in the sample space. The fuzzification approach in the input sets defines the number of neurons that will make up the network. Therefore the cloud fuzzification technique leaves the fuzzy neural network more optimized, without losing its ability to solve problems.

Another difference in the approach proposed in this work is how the parameters of the neurons of the first layer (weight and bias) are defined according to the wavelet transform (Daubechies 1990), thus allowing a relationship between the input data of the model and its initial parameters. For this, the concept of the discrete wavelet transform is used through the application of filter banks. This technique can process data at different scales or resolutions and, regardless of whether the function of interest is an image, a curve or a surface, wavelets offer an excellent technique in representing the detail levels present in the data, thus allowing the values recovered are derived from the representation of the input data. In this case, the values obtained by the techniques to be assigned to the weights and bias of the neurons of the first layer will have a representation on the data that will operate, different from the traditional approach that determines these values in a random way and without a meaning of relation with the of the problem.

The unineuron proposed by Lemos et al. (2010) is used to facilitate the actuation of the model, being able to act in different moments like type AND and type OR. This approach allows greater flexibility of the rules of the fuzzy inference system. Unlike the FNN algorithms explained in this topic, the model proposed in this paper intends to use a data cloud technique to create the first layer neurons. Also, in the neuron of the neural aggregation network, we want to insert an activation function that does not activate all fuzzy rules of the problem at the same time. This means only a few features are taken into account in the problem, making the neural network sparse, efficient and easy to process.

2.3 Evolving hybrid models

Intelligent evolving systems are based on online machine learning methods for intelligent hybrid models. These systems are characterized by their ability to extract knowledge from data and adapt their structure and parameters to better adapt to changes in the environment (Kasabov and Filev 2006). They are formed by an evolutionary set of locally valid subsystems that represent different situations or points of operation. The concepts of this learning methodology make it possible to develop unsupervised clustering algorithms capable of adapting to changes in the environment as the current knowledge is not sufficient to describe such changes (Angelov et al. 2008).

The term "evolving" should not be confused with "evolutionary." Genetic algorithms (Goldberg and Holland 1988) and genetic programming, are based on the evolutionary process that occurs in populations of individuals and use operators based on the concepts of selection, crossing, and mutation of chromosomes as adaptive mechanisms. Also evolving fuzzy systems are based on the process of evolution of individuals throughout their life; specifically the process of human learning, based on the generation and adaptation of knowledge from experiences (Angelov and Zhou 2008).

The evolving models and evolutionary algorithms, which alter parameters as they update new training inputs (Angelov et al. 2010), can be exemplified by the hybrid models proposed by Angelov et al. (2008), Zhang et al. (2006), Aliev et al. (2009), Liao and Tsao (2004), Kasabov (2001), Wang and Li (2003), Yu and Zhang (2005), Hell et al. (2014), Kasabov and Song (1999), Fei and Lu (2018), Maciel et al. (2012), Yu et al. (2018), Pratama et al. (2017), Rong et al. (2009), Lughofer (2011), Angelov and Filev (2004), Subramanian and Suresh (2012), Rong et al. (2006), Rong et al. (2011), Kasabov and Song (2002), de Campos Souza et al. (2019), Angelov and Kasabov (2005), Angelov et al. (2004), Baruah and Angelov (2012), Angelov and Kasabov (2006), Perova and Bodyanskiy (2017).

2.4 Self-organized direction aware data partitioning algorithm- SODA

The process by which fuzzy models treat data can determine how hybrid models can have the interpretability of their results closer to their real world. Models that are fully data-driven are the targets of recent research and have achieved satisfactory results in a cloud data cluster. This clustering concept focused on data is called Empirical Data Analytics (EDA) (Angelov et al. 2017). This concept brings together the data without statistical or traditional probability approaches, based entirely on the empirical observation of the input data of the model, without the need for any previous assumptions and parameters (Gu et al. 2018).

SODA is a data partitioning algorithm capable of identifying peaks/modes of data distribution and uses them as focal points to associate other points to data clouds that resemble Voronoi tessellation. Data clouds can be understood as a particular type of clusters, but with a much different variety. They are non-parametric, but their shape is not predefined and predetermined by the type of distance metric used. Data clouds directly represent the properties of the local set of observed data samples (Gu et al. 2018). The approach employs a magnitude component based on a traditional distance metric and a directional/angular component based on the cosine similarity.

The main EDA operators are described in Angelov et al. (2017), which are also suitable for streaming data processing. The EDA operators include the Cumulative Proximity, Local Density, and Global Density. The local density D_n is defined as the inverse of the normalized cumulative proximity and directly indicates the main pattern of observed data Angelov et al. (2017), where D for the training input x_i= (1, 2,...,N); $N_u > 1$ is defined as follow Gu et al. (2018):

$$D_n(x_i) = \frac{\sum_{j=1}^{n} \pi_n(x_j)}{2n\pi_n(x_j)} \tag{1}$$

Global density is defined for unique data samples together with their corresponding numbers of repeats in the dataset/stream, and of a particular unique data sample, u_i (i=1, 2, ...n_u; $n_u \geq 1$) is expressed as the product of its local density and its number of repeats considered as a weighting factor Angelov et al. (2017) as follows:

$$D_n^G(u_i) = f_i D_n(u_i) \tag{2}$$

As the main EDA operators (cumulative proximity, local density (D) and global density (D^G)) can be updated recursively, the SODA algorithm can be suitable for online processing of streaming data, causing the updating of density groups of data in an evolving process. The algorithm is performed used in this paper utilizing the following steps (Gu et al. 2018):

Stage 1- Preparation: we calculate the average values between every pair of input data, $x_1, x_2, \ldots, x_n$ for both, the square angular components, d_A and square Euclidean components, d_M.

Stage 2- DA Plane Projection: The DA projection operation works with the unique data sample that has the most significant global density, namely $u^*{}_1$. It is initially set to be the first reference, $\mu_1 \leftarrow u_1$, which is also the origin point of the first DA plane, denoted by P1 ($L_c \leftarrow 1$, L_c is the number of existing DA planes in the data space).

Stage 3: Identifying the Focal Points: for each DA plane, expressed as P_e, find the adjacent DA planes.

Stage 4: Forming Data Clouds: After all the DA planes reaching for the modes/peaks of the data density are identified, we consider their origin points, denoted by μ_o, as the focal points and use them to form data clouds according to as a Voronoi tessellation (Okabe et al. 2009). It is worth to stress that the theory of data clouds is quite similar to the idea of clusters, but differs in the following characters:

(i) data clouds are nonparametric;
(ii) data clouds do not have a specific shape;
(iii) data clouds represent the real data distribution. Figure 1 shows an example of the SODA definition and the center of cloud grouping defined by the algorithm. The data submitted to the SODA model are normalized.

2.5 Wavelets

Wavelet is a function capable of decomposing and representing another function described in the time domain so that we can investigate this other function in different frequency and time orders. In Fourier analysis, can only identify information about the frequency domain, but we can not know when these repetitions that we study happen. Meantime, in wavelet analysis, we can also extract information from the function in the time domain. The detailing of the frequency domain analysis decreases as time resolution increases, and it is impossible to increase the detail in one domain without decreasing it in the other. Using wavelet analysis, you can choose the best combination of details for an established goal. Adapting this concept to the fuzzy neural networks, the use of wavelet functions can allow the values destined for the bias and the weights of the neurons to be determined according to their nature and no longer in a random way (Daubechies 1990). In this paper, the discrete wavelet will be adopted. This type of methodology is much used in data compression.

In order to calculate the discrete wavelet transforms, it is through the filter bank application where the filter determined by the coefficients $h = \{h_n\}_{n\in\mathbb{Z}}$ corresponds to a high pass filter and the filter $g = \{g_n\}_{n\in\mathbb{Z}}$ to a low pass filter. Each of these coefficients in the discrete wavelet transform is tabulated. Emphasis is given to the use of the operator $(\downarrow 2)$ is the sub-sampling operator. This operator applied to a discrete function (a sequence) reduces its number of elements in half, recovering only the components in even positions, allowing the procedure to be faster and more precise (Daubechies 1990). The filters h and g are linear operators, which can be employed to the input $\mathbf{x}$ as a convolution:

$$c(n) = \sum_k g(k)x(n-k) = g * x \tag{3}$$

$$d(n) = \sum_k h(k)x(n-k) = h * x \tag{4}$$

The decomposition with the filter decays the signal into only two frequency bands. The chaining of a series of filter banks can be accomplished using sub-sampling operation to provide the division of the sampling frequency by 2 to each new filter bank threaded (Daubechies 1990). Figure 2 shows a schematic of the two filters.

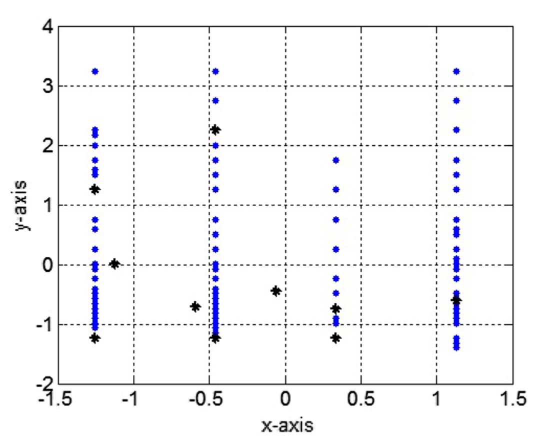

Fig. 1 SODA algorithm

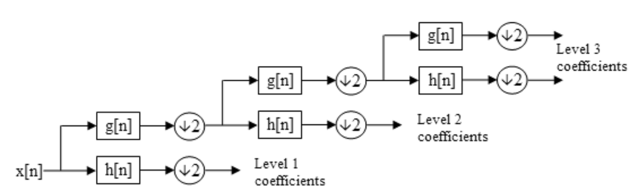

Fig. 2 Filter decomposition signal of input. Avaliable: https://zh.wikipedia.org/wiki/File:Wavelets-Filter_Bank.png

2.6 Rectified linear activation—ReLU

The activation functions allow the introduction of a nonlinear component in the intelligent models, especially those neurons that use logical representations of the human artificial neuron. This characteristic allows intelligent models, such as fuzzy neural networks, to learn more than linear relationships between dependent and independent variables (Karlik and Olgac 2011). Therefore, understanding the functioning of the activation function and in which contexts it can best be applied are preponderant foundations for the success of the model in performing activities that simulate human behavior. They are essential to provide a representative capability for fuzzy neural networks by introducing a nonlinearity component. On the other hand, with this power, some difficulties arise, mainly due to the diversified nature of activation functions, that can vary the effectiveness of their actions according to specific characteristics of the database to which the model is being submitted. In general, by introducing non-linear activation, the cost surface of the neuron is no longer convex, making optimization more complicated. In problems that use parameterization by descent gradients, non-linearity makes it more identifiable which elements need adjustment (Karlik and Olgac 2011). In models of fuzzy neural networks, the main functions of activation are those that use the hyperbolic tangent, Gaussian and linear. Other functions can be highlighted for convolutional and big data problems such as the ReLU (2011), Elu (2015) and Leaky Relu (2013) functions.

A model that has been used to solve various problems is Rectified Linear Activation (ReLU). It a the nonlinear activation function more usually applied to compose neural networks to solve image detection problems. His proposes that if the input is no important to the model, the ReLU function will apply its value to zero and the feature will not be activated. This proposes that at the same moment, only several features are activated, creating the sparse neuron, efficient and straightforward for computing. In these circumstances, the inputs and combinations of a more representative characteristic can act dynamically and efficiently to improve the accuracy of the model (Karlik and Olgac 2011).

Artificial neural networks with the ReLU function are secure to optimize since the ReLU is hugely similar to the identity function. The only difference is that ReLU produces zero in half of its domain. As a consequence, the derivatives stay large while the unit is active (Goodfellow et al. 2016).

3 SODA wavelets regularized fuzzy neural network and ReLU activation function

3.1 Network architecture

The fuzzy neural network described in this chapter follows most of the structure defined in de Campos Souza et al. (2018). However, modifications were made in the first layer (fuzzification) and the third layer (the neural network of aggregation). Unineuron is used to construct fuzzy neural networks in the second layer to solve pattern recognition problems and bring interpretability to the model.

The first layer is composed of neurons whose activation functions are membership functions of fuzzy sets defined for the input variables. For each input variable x_{ij}, L_c clouds are defined A_{lcj}, $l_c = 1 \ldots L_c$ whose membership functions are the activation functions of the corresponding neurons. Thus, the outputs of the first layer are the membership degrees associated with the input values, i.e., $a_{jlc} = \mu^A_{lc}$ for $j = 1 \ldots N$ and $l_c = 1 \ldots L_c$, where N is the number of inputs and L_c is the number of fuzzy sets for each input results by SODA.

The second layer is composed by L_c fuzzy unineuron. Each neuron performs a weighted aggregation of all of the first layer outputs. This aggregation is performed using the weights w_{ilc} (for $i = 1 \ldots N$ and $l_c = 1 \ldots L_c$). For each input variable j, only one first layer output a_{jlc} is defined as input of the l_c-th neuron. So that $\mathbf{w}$ is sparse, each neuron of the second layer is associated with an input variable. Finally, the output layer is composed of one neuron whose activation functions (f) are ReLU Maas et al. (2013). The output of the model is:

$$\mathbf{y} = sign \sum_{j=0}^{l_c} f(z_l v_l) \tag{5}$$

where $z_0 = 1$, v_0 is the bias, and z_j and v_j, $j = 1, \ldots, l_c$ are the output of each fuzzy neuron of the second layer and their corresponding weight, f is the activation function and sign is an operator that transforms the output of the neuron to 1 if it is greater than zero and -1 if it is less than zero, respectively.

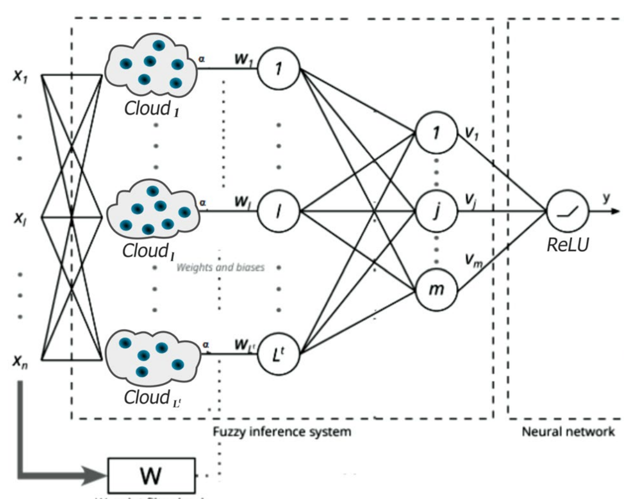

Fig. 3 FNN architecture

Figure 3 presents an example of FNN architecture proposed in this paper.

3.2 A proposition to update first layer weights and bias using wavelets

For the first layer of the FNN, training will be performed with each output of the filters of each level of the wavelet transform, thus allowing to update the weights, which by the original definition in de Campos Souza et al. (2018) should be randomly assigned, assigning them the corresponding values of the output of the wavelet filters. Thus, the training of the fuzzy neural network can happen in a parallel way. In addition to allowing better representation of the problems for the weights and bias of the fuzzy neuron.

The algorithm below presents the necessary information about the steps performed to carry out the training and present in Algorithm 1.

In the first layer of this architecture, the initial vector has l_c values. After the application of the wavelet transform, the resulting vector still with l_c elements but part of this vector is responsible for the high frequencies (detail), and the other part is responsible for the low frequencies (approximation).

Initially, the wavelet transform is applied to the input data resulting in a vector ψ_1. This vector is then passed to a detail removal function that matches the size of the obtained vector to the size of the output of the current layer so that training can be done resulting in a vector ϕ_1. In other words, if the first hidden layer of the FNN has seven neurons, only the first seven values of the vector ψ_1 will be used for the attribution of the weights, the others will be discarded.

Consider that for the FNN example, the initial vector has nine elements. After applying the Wavelet transform, the resulting vector continues with nine elements but part of this vector is responsible for the high frequencies (detail), and the other part is responsible for the low frequencies (approximation). When operating *RemoveDetails*(ψ_1) only the first seven elements of the vector are used. In this way, we have two vectors: a vector of 9 items (input of the first layer) and another vector of 7 features (output of the first layer). From this vector of 7 elements, the values responsible for the approximation are assigned to the bias and the detail value to the weights of the neurons in the first layer.

The high filter values will be assigned to the neuron weights, and the low filter values will be allocated to the bias. This procedure ensures that the same amount of weights and bias that would be randomly assigned are provided based on the wavelet transform, allowing these two parameters to be based on the characteristics of the database submitted to the model.

Figure 4 shows that with the input data of the fuzzy neural network model the low and high pass filter functions generate approximation and detail vectors with the input data. In this case, each of these vectors will be assigned to the bias (low) and the weights (high) of the neurons of the first layer. This assignment was made arbitrarily because in preliminary tests it did not matter if it was otherwise.

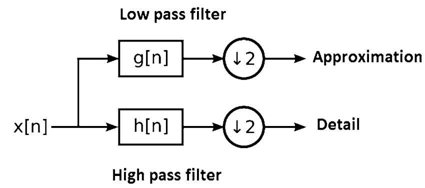

Fig. 4 Value assignment wavelet for the weights of the neuron and the bias

Algorithm 1: FNN training with filter bank *Wavelets* for weight w and bias b

$\psi_1 \leftarrow Wavelet(input);$
$\phi_1 \leftarrow RemoveDetails(\psi_1);$
$Train(in = input, out = \phi_1)$

3.3 Training fuzzy neural network

The membership functions in the first layer of the FNN are adopted in this paper as Gaussian, constructed through the centers (β) obtained by the method of granularization of the input space (SODA) and by the randomly defined sigma (σ). Another difference in the first layer is the definition of the fuzzy neuron weights using the wavelet transform. The number of neurons L_c in the first layer is defined according to the input data, and by the number of partitions (ρ) defined parametrically. This approach partitions the input space, following the definition logic of creating data nodes. The centers of these created clouds make up the Gaussian activation functions of the fuzzy neurons. These changes will allow the adaptation of the data according to the basis submitted to the model, allowing a more independent and data-centered approach. The second layer performs the aggregation of the L_c neurons from the first layer through the unineurons proposed by Lemos et al. (2010). These neurons use the concept of uninorm Yager and Rybalov (1996), which extends *t-norm* and *s-norm*, allowing the values of the identity element (o) to change between 0 and 1. Therefore, the identity element allows the change of the calculation in the fuzzy neuron in a simple way by alternating the aggregation of elements between an *s-norm* (if $o = 0$) and an *t-norm* (if $o = 1$). Thus the value of the identity element allows the uninorm Yager and Rybalov (1996) to have the freedom to transform the unineurons into andneurons or in orneurons, within the resolution of the problem. In this paper, the uninorm (**U**) is expressed as follows:

$$U(x,y)=\begin{cases} oT(\frac{x}{o},\frac{y}{o}), & if\ y\in[0,o] \\ o+(1-o)S(\frac{x-o}{1-o},\frac{y-o}{1-o}), & if\ y\in[o,1] \\ max\ (x,y)\ or\ min\ (x,y), & otherside \end{cases} \tag{6}$$

where T are *t-norms*, S is a *s-norms* and o is the identity element. In this paper, we considered the *t-norm* operator the product and as *s-norm* operator the probabilistic sum.

The unineuron proposed in Lemos et al. (2010) performs the following operations to compute its output:

1 each pair (a_i, w_i) is transformed into a single value $b_i = \mathbf{h}\,(a_i, w_i)$;
2 calculate the unified aggregation of the transformed values with uninorm $\mathbf{U}\,(b_1, b_2 \dots b_n)$, where n is the number of inputs.

The function p (relevancy transformation) is responsible for transforming the inputs and corresponding weights into individual transformed values. This function fulfills the requirement of monotonicity in value which means if the input value increases the transformed value must also increase. Finally, the function p can bring consistency of effect of w_i. A formulation for the p function can be described as Lemos et al. (2010):

$$p(w,a) = wa + wo \tag{7}$$

using the weighted aggregation reported above the unineuron can be written as Lemos et al. (2010):

$$\mathbf{z} = UNI(w;x;a) = U^n_{i=1}p(w_i, a_i) \tag{8}$$

The fuzzy rules can be extracted from the network topology and are presented in Eq. 9.

$$\begin{aligned}
&Rule_1 : If\ x_{i1}\ is\ A^1_1\ with\ certainty\ w_{11} \dots \\
&\quad and/or\ x_{i2}\ is\ A^2_1\ with\ certainty\ w_{21} \dots \\
&\qquad Then\ y_1\ is\ v_1 \\
&Rule_2 : If\ x_{i1}\ is\ A^1_2\ with\ certainty\ w_{12} \dots \\
&\quad and/or\ x_{i2}\ is\ A^2_2\ with\ certainty\ w_{22} \dots \\
&\qquad Then\ y_2\ is\ v_2 \\
&Rule_3 : If\ x_{il}\ is\ A^1_3\ with\ certainty\ w_{13} \dots \\
&\qquad Then\ y_3\ is\ v_3 \\
&Rule_4 : If\ x_{i2}\ is\ A^2_3\ with\ certainty\ w_{23} \dots \\
&\qquad Then\ y_4\ is\ v_4
\end{aligned} \tag{9}$$

After the construction of the L_c unineuron, the bolasso algorithm (Alg. 2) Bach (2008) is executed to select LARS using the most significant neurons (called L_ρ). The final network architecture is defined through a feature extraction technique based on *l1* regularization and resampling. The learning algorithm assumes that the output hidden layer composed of the candidate neurons can be written as de Campos Souza et al. (2018):

$$f(x_i) = \sum_{i=o}^{L_\rho} v_i z_i(x_i) = z(x_i)v \tag{10}$$

where $\mathbf{v} = [v_0, v_1, v_2, \dots, v_{L\rho}]$ is the weight vector of the output layer and $\mathbf{z}\,(x_i) = [z_0, z_1(x_i), z_2(x_i) \dots zL\rho(x_i)]$ the output vector of the second layer, for $z_0 = 1$. In this context, $\mathbf{z}\,(x_i)$ is considered as the non-linear mapping of the input space for a space of fuzzy characteristics of dimension L_ρ (de Campos Souza et al. 2018).

Subsequently, following the determination of the network topology, the predictions of the evaluation of the vector of weights' output layer are performed. In this paper, this vector is considered by the Moore-Penrose pseudo Inverse de Campos Souza et al. (2018):

$$\mathbf{v} = \mathbf{Z}^+\mathbf{y} \tag{11}$$

where Z^+ is pseudo-inverse of Moore-Penrose of $\mathbf{z}$ which is the minimum norm of the least squares solution for the weights of the output layer and $\mathbf{y}$ is the vector of expected output in supervised training.

3.4 Model consistent Lasso estimation through the bootstrap—Bolasso

A universal algorithm used for estimating the parameters of a regression model and selecting relevant characteristics is the Least Angle Regression (LARS) Efron et al. (2004). LARS is a regression algorithm for high-dimensional data that is capable of estimating not only regression coefficients but also a subset of candidate regressors to be included in the final model. LARS is used in the de Jesús Rubio et al. (2018) and de Jesus Rubio et al. (2018) models to perform operator hand movements learning in a manipulator. A modification of the LARS allows the creation of the lasso using the ordinary least squares, a restriction of the sum of the regression coefficients (Efron et al. 2004). Consider a set of n distinct samples (x_i, y_i), where $x_i = [x_{i1}, x_{i2}, ..., x_{iN}]\,\epsilon\,\mathbb{R}^N$ and $y_i\,\epsilon\,\mathbb{R}$ for $i = 1, ..., N$, the cost function of Lasso algorithm can be defined as:

$$\sum_{i=1}^{N} \|z(x_i)\mathbf{v} - y_i\|_2 + \lambda\|\mathbf{v}\|_1 \tag{12}$$

where λ is a regularization parameter, commonly estimated by cross-validation.

The first term of (12) corresponds to the sum of the squares of the residues (RSS). This term decreases as the training error decreases. The second term is an L_1 regularization term. Generally, this term is added, since it improves the generalization of the model, avoiding the super adjustment and can generate sparse models (Efron et al. 2004).

The LARS algorithm can be used to perform the model selection since for a given value of λ only a fraction (or none) of the regressors have corresponding nonzero weights. If $\lambda = 0$, the problem becomes unrestricted regression, and all weights are nonzero. As λ_{max} increases from 0 to a given value λ_{max}, the number of nonzero weights decreases to zero. For the problem considered in this paper, the $z_{L\rho}$ regressors are the outputs of the significant neurons. Thus, the LARS algorithm can be used to select an optimal subset of the significant neurons that minimize (12) for a given value of λ.

Bolasso can be seen as a regime of consensus combinations where the most significant subset of variables on which all regressors agree when the aspect is the selection of variables is maintained (Bach 2008). Bolasso uses the decision threshold system (γ) that represents the choice of model regularization, that is, when the value of γ is indicated, it defines the percentage involved in choosing the best regressors. For example, if $\gamma = 0.5$ means that if the neuron is at least 50% of resampling as a relevant neuron, it will be chosen for the final model.

Bolasso procedure is summarized in Algorithm 2.

Algorithm 2: Bolasso- bootstrap-enhanced least absolute shrinkage operator

(b1) Let n be the number of samples in $\mathbf{z}$:
(b2) Show n examples of ($\mathbf{z}$, $\mathbf{y}$), uniformly and with substitution, called here (z_{samp}, y_{samp}).
(b3) Determine which weights are nonzero given a λ value.
(b4) Repeat steps b1: b3 for a specified number of bootstraps bt.
(b5) Take the intersection of the non-zero weights indexes of all bootstrap replications.
(b6) Select the resulting variables.
(b7) Revise the results using the variables selected via non-regularized least squares regression.
(b8) Repeat the procedure for each value of bt bootstraps and λ (actually done more efficiently by collecting interim results).
(b9) Determine "optimal" values for λ and bt.
(b10) Use the consensus threshold (γ) to determine the most significant neurons in the model.

3.5 Use of activation functions of type rectified linear activation (ReLU) in the neural network aggregation

In sequence to classify higher efficient functions to act as activation functions the paper (Karlik and Olgac 2011) determined the rectified linear activation (ReLU). This function is defined by:

$$f_{ReLU}(z_{L\rho}) = max\ (0, z_{L\rho}). \qquad (13)$$

In Eq. (5) the function f is replaced by the function f_{ReLU}.

The learning method can be synthesized as demonstrated in Algorithm 3. It has three parameters:

1 the number of grid size, ρ;
2 the number of bootstrap replications, bt;
3 the consensus threshold, γ.

Algorithm 3: SODA-FNN training

(1) Define grid size, ρ.
(2) Define bootstrap replications, bt.
(3) Define consensus threshold, γ
(4) Calculate L_c cluster in the first layer using SODA and ρ.
(5) Construct L_c fuzzy neurons with Gaussian membership functions constructed with center values (β) derived from SODA and sigma (σ) defined at random.
(6) Define the weights and bias of the fuzzy neurons using the transform wavelets.
(7) Construct L_c unineurons with random weights and bias on the second layer of the network by welding the L_c fuzzy neurons of the first layer.
(8) **For** all K inputs do
(8.1) Calculate the mapping $\mathbf{z}$ (x_i)
end for
(9) Select significant L_s neurons using the bootstrap lasso according to the settings of bt and γ.
(10) Estimate the weights of the output layer (Eq. 11)
(11) Calculate the output of the model using an artificial neuron with activation function of type ReLU (Eq. 5 and Eq. 13).

4 Test of binary patterns classification

4.1 Assumptions and initial test configurations

In this section, the assumptions of the classification tests for the model proposed in this paper are presented. To perform the tests, real and synthetic bases were chosen, seeking to verify if the accuracy of the proposed model surpasses the traditional FNN techniques of pattern classification. The following tables present information about the tests, presenting factors such as the percentage of samples destined for the training and testing of fuzzy neural networks. All the tests with the involved algorithms were done randomly, avoiding tendencies that could interfere in the evaluations of the results. The model proposed in this paper, called SODA-FNN, was compared to fuzzy neural network classifiers using fuzzy c-means (FCM-FNN) (Lemos et al. 2012) and genfis1 (GN-FNN) (de Campos Souza et al. 2018) in the fuzzification process.

In the last two models, the weights and bias were used in the first and second layers randomly, already in the approach proposed in this paper, the weights and bias in the first layer are defined by the wavelets. The number of primary neurons

Table 1 Synthetic dataset used in the experiments

Dataset	Init.	Feature	Train	Test
Half Kernel	HKN	2	350	150
Spiral	SPR	2	350	150
Cluster	CLU	2	350	150
Corner	COR	2	350	150

of each model is defined according to the number of centers (FCM-FNN), membership functions (GN-FNN) and grid size (SODA-FNN). For uniformity of the tests, the values involved in the first layers of the models, which end up defining the number of L_c neurons, were arbitrated in the range of [3–5], where the best results were defined using cross-validation. In the three models, the unineuron is adopted as logical neuron of the second layer. The activation functions of the neurons used in artificial neural networks were ReLU (SODA-FNN), sigmoid (FCM-FNN) and a linear function (GN-FNN). A total of 30 experiments were performed with the three models submitted to all test bases.

In all tests and all models, the samples were shuffled in each test to demonstrate the actual capacity of the models. Percentage values for the classification tests are presented in the results tables, accompanied by the standard deviation found in the 30 replicates. The outputs of the model were normalized to 0 and 1 to aid the correct calculations. The factors evaluated in this paper are as follows:

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{14}$$

$$AUC = \frac{1}{2}(sensitivity + specificity) \tag{15}$$

where the sensitivity and specificity are calculated using the following equations:

$$sensitivity = \frac{TP}{TP + FN} \tag{16}$$

$$specificity = \frac{TN}{TN + TP} \tag{17}$$

where, TP = true positive, TN = true negative, FN = false negative and FP = false positive. All weights of the output layer were obtained using ELM methods in all models.

4.2 Database used in the tests

The following tables identify the settings applied in the tests. In Table 1, the information of the synthetic bases used in the binary pattern classification tests. In Table 2 the real bases extracted from Bache and Lichman (2013) for classification problems.

Table 2 Real dataset used in the experiments

Dataset	Init.	Feature	Train	Test
Haberman	HAB	3	214	92
Transfusion	TRA	4	523	225
Mammographic	MAM	5	581	249
Liver Disorder	LIV	6	242	103
Diabetes	DIA	8	538	230
Heart	HEA	13	189	81
German Credit	GER	14	390	168
Australian Credit	AUS	24	700	300
Ionosphere	ION	32	245	106

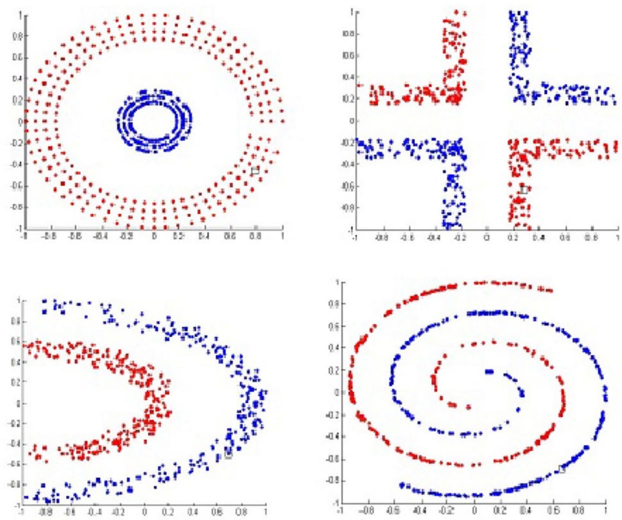

Fig. 5 Synthetic dataset

Table 3 Acurracy of the model in the tests performed

Dataset	SODA-FNN	FCM-FNN	GN-FNN
HKN	99.87 (0.11)	99.64 (0.07)	**99.93 (0.17)**
SPR	97.60 (0.74)	**98.15 (0.87)**	87.12 (2.43)
CLU	**99.88 (0.04)**	98.71 (0.14)	99.12 (0.44)
COR	88.94 (4.87)	94.65 (0.82)	**97.65 (2.11)**

Figure 5 shows the characteristics of the synthetic bases used in the tests.

4.3 Binary pattern classification tests

Tables 3 and 4 present the accuracy and AUC results respectively of the tests with the synthetic bases.

After carrying out the tests with synthetic bases, it was confirmed that the proposed model presented smaller accuracy results in the spiral base, which has greater complexity

Table 4 AUC of the model in the tests performed

Dataset	SODA-FNN	FCM-FNN	GN-FNN
HKN	**0.9981 (0.03)**	0.9654 (0.14)	0.9965 (0.01)
SPR	0.9831 (0.04)	**0.998 (0.01)**	0.9762 (0.54)
CLU	0.9907 (0.01)	0.9920 (0.74)	**0.9931 (0.19)**
COR	0.9165 (0.54)	**0.9650 (0.65)**	0.8856 (0.53)

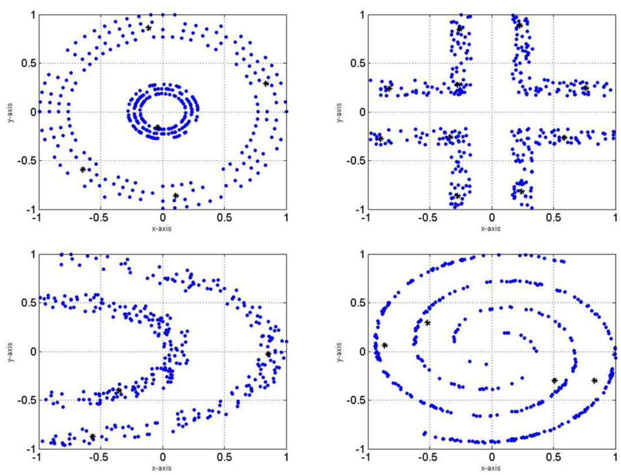

Fig. 6 Synthetic dataset—SODA result

in its composition. The other bases had an equivalent precision within the standard deviation found in the experiments. We highlight the precision of the proposed model and the one that uses a sigmoid activation function, which had a high success rate in all experiments. Figure 6 presents the result of SODA and Fig. 7 the model decision space and Fig. 8 decision 3d plot.

The decision space present in Fig. 7 demonstrates that the technique can act as an excellent pattern classification. Decision spaces are suitable for separating the main samples intended for testing.

In the next test of pattern classification using real databases will be compared in each one of the models the accuracy of model (Table 5), AUC (Table 6), execution time (Table 7) and the number of fuzzy rules (Table 8) used to obtain the results. Tests performed on a desktop machine with Intel Core i5-3470 processor 3.20GHz and 4.00GB Memory.

In the execution of real tests, it was verified that the model proposed in this paper obtained superior results of accuracy in six of the nine datasets proposed in the test. In the datasets that the model did not take the best test results, it obtained results close to the models evaluated in the test.

From the lower results of the model, the values of the test with the heart dataset are highlighted. The model proposed in this paper obtained a significant difference for the

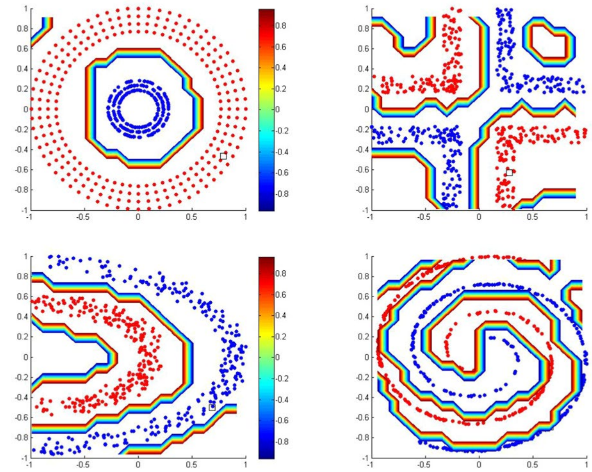

Fig. 7 Synthetic Dataset- FNN decision

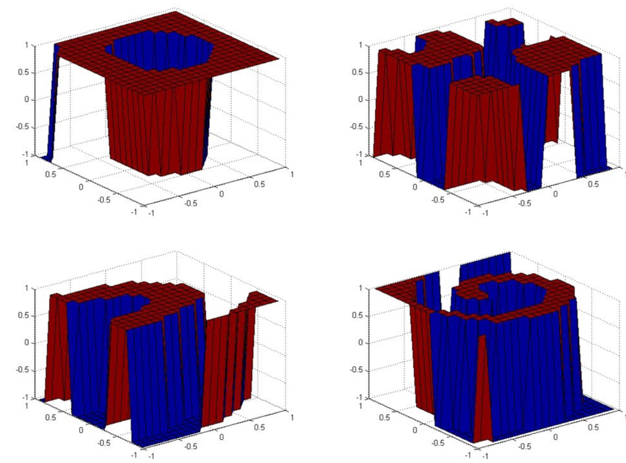

Fig. 8 Synthetic Dataset- FNN decision -3d

Table 5 Accuracy of the model in the tests performed

Dataset	SODA-FNN	FCM-FNN	GN-FNN
HAB	**70.83 (4.16)**	67.12 (3.32)	61.59 (2.23)
TRA	75.49 (2.36)	**78.37 (2.19)**	75.85 (2.29)
MAM	**83.91 (1.75)**	82.10 (2.11)	81.27 (2.24)
LIV	**67.86 (5.57)**	65.51 (4.12)	66.08 (2.16)
DIA	**74.78 (3.30)**	67.40 (3.80)	74.67 (2.48)
HEA	74.86 (6.06)	74.78 (11.76)	**79.22 (1.54)**
GER	**70.51 (2.91)**	69.77 (1.54)	70.34 (2.98)
AUS	71.30 (3.06)	67.53 (0.78)	**75.59 (4.51)**
ION	**78.59 (11.65)**	69.44 (1.14)	75.66 (2.51)

other models in the analysis. Another factor that can also be considered as non-positive was the high standard deviation for the result of the ionosphere base. Although the model

Table 6 AUC of the model in the tests performed

Dataset	SODA-FNN	FCM-FNN	GN-FNN
HAB	**0.5650 (0.05)**	**0.5650 (0.03)**	0.5642 (0.466)
TRA	0.5109 (0.76)	0.6300 (0.03)	**0.6393 (0.23)**
MAM	**0.8399 (0.02)**	0.8210 (0.02)	0.8349 (2.24)
LIV	0.6465 (0.05)	**0.6591 (0.04)**	0.6560 (0.04)
DIA	0.7000 (0.26)	0.6523 (0.03)	**0.7057 (0.03)**
HEA	0.7444 (0.04)	0.7797 (0.05)	**0.7927 (0.41)**
GER	0.5764 (0.13)	0.8718 (0.01)	**0.8218 (0.01)**
AUS	0.5807 (0.03)	0.6506 (0.06)	**0.7533 (0.04)**
ION	0.7468 (0.06)	0.6506 (0.06)	**0.7533 (0.04)**

Table 7 Algorithm execution time for pattern classification (in seconds)

Dataset	SODA-FNN	FCM-FNN	GN-FNN
HAB	**28.55 (2.33)**	52.14 (6.08)	74.17 (12.14)
TRA	**24.51 (3.42)**	66.12 (1.54)	123.44 (21.43)
MAM	**32.16 (4.57)**	102.44 (15.49)	144.23 (12.15)
LIV	**17.41 (0.12)**	58.43 (2.11)	83.45 (5.24)
DIA	**13.51 (0.52)**	17.58 (7.01)	19.44 (2.23)
HEA	**8.45 (0.17)**	14.55 (0.53)	17.14 (1.10)
GER	**14.01 (0,44)**	27.16 (2.54)	44.15 (6.21)
AUS	**16.17 (1.43)**	38.53 (12.42)	104.23 (32.53)
ION	**16.12 (1.14)**	33.51 (8.51)	54.17 (13.14)

Table 8 Number of fuzzy rules used by the model

Dataset	SODA-FNN	FCM-FNN	GN-FNN
HAB	**8.65 (1.75)**	12.77 (2.65)	17.61 (1.76)
TRA	**9.77 (2.11)**	22.06 (0.88)	17.60 (2.24)
MAM	**17.88 (5.12)**	33.64 (12.96)	24.57 (8.65)
LIV	**9.76 (8.56)**	33.74 (14.90)	12.17 (7.43)
DIA	**6.87 (0.81)**	14.89 (2.43)	7.43 (1.08)
HEA	14.16 (2.43)	25.76 (3.87)	**10.12 (1.43)**
GER	**18.17 (3.49)**	22.43 (2.98)	19.32 (4.65)
AUS	**14.98 (3.44)**	48.42 (12.55)	16.01 (2.20)
ION	**14.56 (2.33)**	32.44 (4.58)	19.67 (4.54)

obtained the best average results, it showed much instability in solving the problem. On the other hand, in the tests of mammography, transfusion and German credit, the model was very stable in the results. This factor may have happened because the nature of the data is greatly varying in the problems that it presented high standard deviation and remained more stable (or the issues have this nature) during the patterns classification tests.

The results of Table 7 of the tests prove that the model presents a shorter execution time of binary pattern classification when compared to the other FNN in the test. This enables the proposal acts with less time due to the techniques used to carry out the correct identification of the patterns.

Another relevant factor about the model is that in addition to presenting a smaller number of fuzzy rules (Table 8), it also presented a much shorter execution time (Table 7) than techniques that use grouping or the model of equally spaced membership functions. If the differences in time and complexity of the fuzzy neural network were much smaller with a limited number of samples, this difference should appear more evident when the model solves problems with many features and also a high number of samples. Therefore, because it has achieved the majority of the best results in the tests of classification of patterns with real databases using less time and a smaller number of neurons/rules, the viability of the model in the resolution of these problems is verified.

An architecture with a smaller number of neurons facilitates the reading of the most relevant fuzzy rules. As the SODA technique works with the data according to their complex nature, problems become more representative directly affecting the constructed fuzzy rules, allowing them to be more representative of the nature of the problem.

It should be noted that FNN now becomes a model to work with a high number of samples or problems with many features, such factor was very complicated when using the ANFIS process in the fuzzification of the model. Extracting knowledge from large volumes of data is a current and fundamental problem for many corporations.

5 Conclusion

The fuzzy neural network proposed in this paper obtained better results than other models that use the extreme learning machine and fuzzy logic neurons. The use of the wavelet transforms allowed the model to use the training data to define the values of the weights and bias in the first layer, thus allowing the parameters of the model to be more coherent with the data submitted to the model. The use of unineuron facilitates the transition of the use of the AND and OR neurons, allowing the interpretation of the fuzzy rules to be closer to the real one. Finally, the use of the SODA technique maintained the interpretability capacity of the FNN model and significantly reduced the execution time when compared to the other FNN models that use logical neurons and fuzzy grouping techniques. Finally, the use of the ReLU activation function helped to improve the responses obtained by the FNN model when compared to the models that use linear and sigmoidal activation functions in real datasets.

The patterns classification tests with less number of fuzzy rules and the use of faster activation functions allow the model proposed in this paper to be identified as a model that maintained the accuracy of pattern classification and at the same time significantly decreased the response time to carry out the activities. This approach accredits the model to work with large-scale databases (big data).

The tests performed verified that the definition of weights and bias using wavelets, the use of a cloud data group and the use of the ReLU activation function is satisfactory for the classification of binary patterns executed by fuzzy neural networks. Basing the parameters to represent the characteristics of the base, we find essential variations in the results of precision found. This approach brings more representativeness to the results of the FNN that can elaborate more fuzzy rules with the input data.

For future work can be checked the impact on the model output and the processing time of your actions using other types of membership functions. Because data clouds theory allows the use of any existing membership function, there may be improvements in the classification of patterns by changing the type of function used.

Other approaches can be performed to optimize parameters related to Grid size, the number of bootstrap repetitions and consensus threshold. Despite finding suitable results, cross-validation spends a high computational time to perform the combinations defined in the tests and determine the models. With advanced optimization techniques, genetic algorithms and other existing intelligent approaches, the best model parameters can be found more dynamically and efficiently. Also in extensions of this work can be applied problems of linear regression, prediction of time series to verify if the model maintains its capacity of universal approximation. Other training approaches can also be evaluated to identify the impacts ELM can generate on parameter setting. Finally, the application of this intelligent model is stimulated for problems with larger dimensions than those that were initially submitted to the test. The SODA technique lowers the complexity of the network structure, so for problems of high dimensionality and Big Data, the model may be suitable to deal with such kind of problems. Testing real problems with large volumes of data is a strongly encouraged approach to examining the model.

Acknowledgements The thanks of this work are destined to CEFET-MG, University Center UNA and University Center UNIBH.

References

Aliev RA, Guirimov B, Fazlollahi B, Aliev RR (2009) Evolutionary algorithm-based learning of fuzzy neural networks. Part 2: recurrent fuzzy neural networks. Fuzzy Sets Syst 160(17):2553–2566

Amari S-I (1993) Backpropagation and stochastic gradient descent method. Neurocomputing 5(4–5):185–196

Angelov PP, Filev DP (2004) An approach to online identification of takagi-sugeno fuzzy models. IEEE Trans Syst Man Cybern Part B (Cybern) 34(1):484–498

Angelov PP, Gu X, Príncipe JC (2017) A generalized methodology for data analysis. IEEE Trans Cybern

Angelov PP, Zhou X (2008) Evolving fuzzy-rule-based classifiers from data streams. IEEE Trans Fuzzy Syst 16(6):1462–1475

Angelov P, Filev D, Kasabov N (2008) Guest editorial: evolving fuzzy systems: preface to the special section. IEEE Trans Fuzzy Syst 16(6):1390–1392

Angelov P, Lughofer E, Zhou X (2008) Evolving fuzzy classifiers using different model architectures. Fuzzy Sets Syst 159(23):3160–3182

Angelov P, Gu X, Kangin D (2017) Empirical data analytics. Int J Intell Syst 32(12):1261–1284

Angelov P, Filev DP, Kasabov N (2010) Evolving intelligent systems: methodology and applications, vol 12. Wiley, Amsterdam

Angelov P, Kasabov N (2005) Evolving computational intelligence systems. In: Proceedings of the 1st international workshop on genetic fuzzy systems, pp 76–82

Angelov P, Kasabov N (2006) Evolving intelligent systems, eis. IEEE SMC eNewsLetter 15:1–13

Angelov P, Victor J, Dourado A, Filev D (2004) On-line evolution of takagi-sugeno fuzzy models. In: IFAC Proceedings Vol 37(16):67–72

Bach FR (2008) Bolasso: model consistent lasso estimation through the bootstrap. In: Proceedings of the 25th international conference on machine learning, ACM, pp 33–40

Bache K, Lichman M (2013) Uci machine learning repository

Ballini R, Gomide F (2002) Learning in recurrent, hybrid neurofuzzy networks. In: Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE international conference on, vol 1, IEEE, pp 785–790

Baruah RD, Angelov P (2012) Evolving local means method for clustering of streaming data. In: 2012 IEEE international conference on fuzzy systems, IEEE, pp 1–8

Bezdek JC, Ehrlich R, Full W (1984) FCM: the fuzzy c-means clustering algorithm. Comput Geosci 10(2–3):191–203

Bordignon F, Gomide F (2014) Uninorm based evolving neural networks and approximation capabilities. Neurocomputing 127:13–20

Buckley JJ, Hayashi Y (1994) Fuzzy neural networks: a survey. Fuzzy Sets Syst 66(1):1–13

Clevert D.-A, Unterthiner T, Hochreiter S (2015) Fast and accurate deep network learning by exponential linear units (elus). arXiv:1511.07289

Daubechies I (1990) The wavelet transform, time-frequency localization and signal analysis. IEEE Trans Inf Theory 36(5):961–1005

de Campos Souza PV, de Oliveira PFA (2018) Regularized fuzzy neural networks based on nullneurons for problems of classification of patterns. In: 2018 IEEE symposium on computer applications industrial electronics (ISCAIE), pp 25–30

de Campos Souza PV, Guimaraes AJ, Araújo VS, Rezende TS, Araújo VJS (2018) Fuzzy neural networks based on fuzzy logic neurons regularized by resampling techniques and regularization theory for regression problems. Intell Artif 21(62):114–133

de Campos Souza PV, Silva GRL, Torres LCB (2018) Uninorm based regularized fuzzy neural networks. In: 2018 IEEE conference on evolving and adaptive intelligent systems (EAIS), pp 1–8

de Campos Souza PV, Torres LCB (2018) Regularized fuzzy neural network based on or neuron for time series forecasting. In: Barreto GA, Coelho R, (eds), Fuzzy information processing, Springer, Cham, pp 13–23

de Campos Souza PV, Torres LCB, Guimaraes AJ, Araujo VS, Araujo VJS, Rezende TS (2019) Data density-based clustering for regularized fuzzy neural networks based on nullneurons and robust

activation function. Soft Comput. https://doi.org/10.1007/s00500-019-03792-z
de Jesús Rubio J (2017) Usnfis: uniform stable neuro fuzzy inference system. Neurocomputing 262:57–66
de Jesús Rubio J (2018) Error convergence analysis of the sufin and csufin. Appl Soft Comput
de Jesus Rubio J, Garcia E, Aquino G, Aguilar-Ibañez C, Pacheco J, Meda-Campaña JA (2018) Mínimos cuadrados recursivos para un manipulador que aprende por demostración. Revista Iberoamericana de Automática e Informática industrial
de Jesús Rubio J, Garcia E, Aquino G, Aguilar-Ibañez C, Pacheco J, Zacarias A (2018) Learning of operator hand movements via least angle regression to be teached in a manipulator. Evol Syst pp 1–16
Ebadzadeh MM, Salimi-Badr A (2018) Ic-fnn: A novel fuzzy neural network with interpretable, intuitive, and correlated-contours fuzzy rules for function approximation. IEEE Trans Fuzzy Syst 26(3):1288–1302
Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. Ann Stat 32(2):407–499
Fei J, Lu C (2018) Adaptive sliding mode control of dynamic systems using double loop recurrent neural network structure. IEEE Trans Neural Netw Learn Syst 29(4):1275–1286
Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99
Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press http://www.deeplearningbook.org
Gu X, Angelov P, Kangin D, Principe J (2018) Self-organised direction aware data partitioning algorithm. Inf Sci 423:80–95
Guimaraes AJ, Araújo VJ, de Oliveira Batista L, Souza PVC, Araújo V, Rezende TS (2018) Using fuzzy neural networks to improve prediction of expert systems for detection of breast cancer. In: XV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), no 1, pp 799–810
Guimarães AJ, Araujo VJS, de Campos Souza PV, Araujo VS, Rezende TS (2018) Using fuzzy neural networks to the prediction of improvement in expert systems for treatment of immunotherapy. In: Ibero-American conference on artificial intelligence, Springer, pp 229–240
Hansen LP (1982) Large sample properties of generalized method of moments estimators. Econometrica pp 1029–1054
Han M, Zhong K, Qiu T, Han B (2018) Interval type-2 fuzzy neural networks for chaotic time series prediction: a concise overview. IEEE Trans Cybern pp 1–12
Hell M, Costa P, Gomide F (2008) Hybrid neurofuzzy computing with nullneurons. In: Neural networks, IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, IEEE 2008:3653–3659
Hell M, Costa P, Gomide F (2014) Participatory learning in the neurofuzzy short-term load forecasting. In: Computational intelligence for engineering solutions (CIES), (2014) IEEE symposium on, IEEE, pp 176–182
Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
Jang J-S (1993) Anfis: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–685
Juang C-F, Huang R-B, Cheng W-Y (2010) An interval type-2 fuzzy-neural network with support-vector regression for noisy regression problems. IEEE Trans Fuzzy Syst 18(4):686–699
Karlik B, Olgac AV (2011) Performance analysis of various activation functions in generalized mlp architectures of neural networks. Int J Artif Intell Expert Syst 1(4):111–122
Kasabov N (2001) Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. IEEE Trans Syst Man Cybern Part B (Cybern) 31(6):902–918
Kasabov NK, Song Q (1999) Dynamic evolving fuzzy neural networks with" m-out-of-n" activation nodes for on-line adaptive systems. Department of Information Science, University of Otago
Kasabov NK (2001) On-line learning, reasoning, rule extraction and aggregation in locally optimized evolving fuzzy neural networks. Neurocomputing 41(1–4):25–45
Kasabov NK, Song Q (2002) Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Trans Fuzzy Syst 10(2):144–154
Kasabov N, Filev D (2006) Evolving intelligent systems: methods, learning, and applications. In: Evolving fuzzy systems, (2006) international symposium on, IEEE, pp 8–18
Koutrika G, Zadeh ZM, Garcia-Molina H (2009) Data clouds: summarizing keyword search results over structured data. In: Proceedings of the 12th international conference on extending database technology: advances in database technology. ACM, pp 391–402
Lemos AP, Caminhas W, Gomide F (2012) A fast learning algorithm for uninorm-based fuzzy neural networks. In: Fuzzy information processing society (NAFIPS) (2012) Annual Meeting of the North American, IEEE pp 1–6
Lemos A, Caminhas W, Gomide F (2010) New uninorm-based neuron model and fuzzy neural networks. In: Fuzzy information processing society (NAFIPS) (2010) annual meeting of the North American, IEEE 1–6
Liao G-C, Tsao T-P (2004) Application of fuzzy neural networks and artificial intelligence for load forecasting. Electr Power Syst Res 70(3):237–244
Lin Y.-C, Wu Z.-Y, Lee S.-J, Ouyang C.-S (2018) Neuro-fuzzy network for pm2. 5 prediction. In: International conference on smart vehicular technology, transportation, communication and applications, Springer, New York, pp 269–276
Lughofer E (2011) Evolving fuzzy systems-methodologies, advanced concepts and applications, vol 53. Springer, New York
Lughofer E (2012) Single-pass active learning with conflict and ignorance. Evol Syst 3(4):251–271
Lughofer E, Pratama M, Skrjanc I (2018) Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation. IEEE Trans Fuzzy Syst 26(4):1854–1865
Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of ICML, vol 30(1), p 3
Maciel L, Lemos A, Gomide F, Ballini R (2012) Evolving fuzzy systems for pricing fixed income options. Evol Syst 3(1):5–18
Okabe A, Boots B, Sugihara K, Chiu S N (2009) Spatial tessellations: concepts and applications of Voronoi diagrams, vol 501. Wiley, Amsterdam
Pedrycz W (1991) Neurocomputations in relational systems. IEEE Trans Pattern Anal Mach Intell 3:289–297
Pedrycz W (2006) Logic-based fuzzy neurocomputing with unineurons. IEEE Trans Fuzzy Syst 14(6):860–873
Pedrycz W, Gomide F (2007) Fuzzy systems engineering: toward human-centric computing. Wiley, Amsterdam
Perova I, Bodyanskiy Y (2017) Fast medical diagnostics using autoassociative neuro-fuzzy memory. Int J Comput 16(1):34–40
Pratama M, Zhang G, Er MJ, Anavatti S (2017) An incremental type-2 meta-cognitive extreme learning machine. IEEE Trans Cybern 47(2):339–353
Rong H-J, Huang G-B, Sundararajan N, Saratchandran P (2009) Online sequential fuzzy extreme learning machine for function approximation and classification problems. IEEE Trans Syst Man Cybern Part B (Cybern) 39(4):1067–1072
Rong H-J, Sundararajan N, Huang G-B, Saratchandran P (2006) Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction. Fuzzy Sets Syst 157(9):1260–1275

Rong H-J, Sundararajan N, Huang G-B, Zhao G-S (2011) Extended sequential adaptive fuzzy inference system for classification problems. Evol Syst 2(2):71–82

Rosa R, Gomide F, Ballini R (2013) Evolving hybrid neural fuzzy network for system modeling and time series forecasting. In: Machine learning and applications (ICMLA), 2013 12th international conference on, vol 2, IEEE, pp 78–383

Rosa R, Maciel L, Gomide F, Ballini R (2014) Evolving hybrid neural fuzzy network for realized volatility forecasting with jumps. In: Computational intelligence for financial engineering and economics (CIFEr), 2104 IEEE Conference on, IEEE, pp 481–488

Sharifian A, Ghadi MJ, Ghavidel S, Li L, Zhang J (2018) A new method based on type-2 fuzzy neural network for accurate wind power forecasting under uncertain data. Renew Energy 120:220–230

Silva AM, Caminhas W, Lemos A, Gomide F (2014) A fast learning algorithm for evolving neo-fuzzy neuron. Appl Soft Comput 14:194–209

Song L-K, Wen J, Fei C-W, Bai G-C (2018) Distributed collaborative probabilistic design of multi-failure structure with fluid–structure interaction using fuzzy neural network of regression. Mech Syst Signal Process 104:72–86

Souza PVDC, Guimaraes AJ, Araujo VS, Rezende TS, Araujo VJS (2018) Regularized fuzzy neural networks to aid effort forecasting in the construction and software development. Int J Artif Intell Appl 9(6):13–26

Souza PVC (2018) Regularized fuzzy neural networks for pattern classification problems. Int J Appl Eng Res 13(5):2985–2991

Subramanian K, Suresh S (2012) A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system. Appl Soft Comput 12(11):3603–3614

Tang J, Liu F, Zou Y, Zhang W, Wang Y (2017) An improved fuzzy neural network for traffic speed prediction considering periodic characteristic. IEEE Trans Intell Transp Syst 18(9):2340–2350

Tikhonov AN, Goncharsky A, Stepanov V, Yagola AG (2013) Numerical methods for the solution of ill-posed problems. Springer, New York, vol 328

Wang W-Y, Li Y-H (2003) Evolutionary learning of bmf fuzzy-neural networks using a reduced-form genetic algorithm. IEEE Trans Syst Man Cybern Part B (Cybern) 33(6):966–976

Yager RR, Rybalov A (1996) Uninorm aggregation operators. Fuzzy Sets Syst 80(1):111–120

Yen VT, Nan WY, Van Cuong P (2018) Recurrent fuzzy wavelet neural networks based on robust adaptive sliding mode control for industrial robot manipulators. Neural Comput Appl pp 1–14

Yu X, Fu Y, Li P, Zhang Y (2018) Fault-tolerant aircraft control based on self-constructing fuzzy neural networks and multivariable smc under actuator faults. IEEE Trans Fuzzy Syst 26(4):2324–2335

Yu L, Zhang Y-Q (2005) Evolutionary fuzzy neural networks for hybrid financial prediction. IEEE Trans Syst Man Cybern Part C (Appl Rev) 35(2):244–249

Zhang Q-Z, Gan W-S, Zhou Y-L (2006) Adaptive recurrent fuzzy neural networks for active noise control. J Sound Vib 296(4–5):935–948