Regularized Fuzzy Neural Network Based on Or Neuron for Time Series Forecasting

Paulo Vitor de Campos Souza[1,2](✉) and Luiz Carlos Bambirra Torres[3](✉)

[1] Secretariat of Information Governance-CEFET-MG, Avenue Amazonas 5253, Belo Horizonte, Minas Gerais 30.421-169, Brazil
goldenpaul@informatica.esp.ufmg.br
pauloc@prof.unibh.br

[2] Institute of Communication and Design-UNI-BH, Avenue Prof. Mario Werneck 1685, Belo Horizonte, Minas Gerais 30.575-180, Brazil

[3] Post Graduate Program in Electrical Engineering- UFMG, Av. Antônio Carlos 6627, Belo Horizonte, Minas Gerais 31270-901, Brazil
luizlitc@gmail.com

Abstract. This paper presents a training algorithm for regularized fuzzy neural networks which is able to generate consistent and accurate models while adding some level of interpretation to applied problems to act in the prediction of time series. Learning is achieved through extreme learning machines to estimate the parameters and a technique of selection of characteristics using regularization concept and resampling, which is able to perform the definition of the network topology through the selection of subsets of fuzzy neuron more significant to the problem. Numerical experiments are presented for time series problems using benchmark bases on machine learning problems. The results obtained are compared to other techniques of prediction of reference series in the literature. The model made rough estimates of the responses obtained by the models of fuzzy neural networks for time series forecasting with fewer fuzzy rules.

Keywords: Fuzzy neural networks · Regularization · Bolasso
Fuzzy logic neurons · Time series forecasting

1 Introduction

Intelligent models that are composed of artificial neural networks and concepts of fuzzy systems have great utility for classification, regression or prediction of time series. The fuzzy neural networks use the structure of an artificial neural network, where classical artificial neurons are replaced by fuzzy neurons [16]. It has as its relevance factor its transparency, allowing the use of information a priori to define the initial structure of the network and the extraction of relevant information from the resulting topology. Thus, the neural network is seen as a linguistic system with level of interpretation, preserving the learning capacity of RNA. Its fuzzy neurons are composed of triangular norms, which generalize the union and intersection operations of classical clusters to the theory of fuzzy sets.

G. A. Barreto and R. Coelho (Eds.): NAFIPS 2018, CCIS 831, pp. 13–23, 2018.
https://doi.org/10.1007/978-3-319-95312-0_2

Examples of fuzzy neurons include neurons, *and*, *or*, *nullneurons* and *unineuron* where their differences are based on the way in which inputs and weights are aggregated [8]. The use of extreme learning machine theory (ELM) [9] has been used for the training of fuzzy neural networks as in [3, 14, 18, 20]. This motivation is mainly due to the performance and low computational cost presented by algorithms based on ELMs, however, most of these cases were used in pattern classification techniques. Already [12, 17, 21, 22] uses fuzzy neural networks to work with time series prediction.

This paper proposes to use the learning methodology for fuzzy neural networks of the feed forward type with a layer of fuzzification, a hidden layer composed of fuzzy neurons, and a linear output layer. The algorithm is able to generate consistent and accurate models by aggregating interpretation to the resulting structure. The learning of the model is carried out through extreme learning machine concepts, but a regularization term is added in the cost function which, together with a resampling technique, is able to perform the selection of the best neurons in internal layers, generating parsimonious models. Initially, fuzzy sets with equally spaced membership functions are defined for each input variable in the fuzzification layer. Subsequently an initial set of fuzzy candidate neurons is generated. From this initial set of neurons, the bootstrap lasso algorithm [2], is used to define the network topology, selecting a subset of the significant fuzzy neurons. Finally, the least squares algorithm is used to estimate the weights of the network output layer. This technique was used to classify binary patterns using *unineuron* [18] and *andneuron* [20], but due to the characteristics present in the ELM, the model's ability to predict time series was verified. Through all the steps the fuzzy neural network is able to act effectively in the forecast of time series. In this paper, we performed tests of time series prediction in the Box and Jenkins gas furnace [5] and the proposed model was compared to other algorithms of fuzzy neural networks for the same purpose widely used in the literature.

The remainder of the paper is organized as follows. Section 2 presents the theoretical concepts related to fuzzy neural networks and neural logic neurons. Section 3 describes the methodology used to train fuzzy neural networks. In Sect. 4 results of numerical experiments are presented. Finally, Sect. 5 presents the conclusions.

2 Fuzzy Neural Networks

2.1 Artificial Neural Networks and Fuzzy Systems

In [23] the authors defines *an* artificial neural network composed of an input layer, one or more hidden layers and an output layer. The network can be completely connected where each neuron is connected to all the neurons of the next layer, partially connected where each neuron is, or locally connected where there is a partial connection oriented to each type of functionality. To perform the training of a neural network, a set of data is required that contains patterns for training and desired outputs. In this way, the problem of neural network training is summarized in an optimization problem in which we want to find the best set of weights that minimizes the mean square error calculated between the network outputs and the desired outputs. The fuzzy systems are based on fuzzy logic, developed by [24]. His work was motivated because of the wide variety of vague and uncertain information in making human decisions. Some problems can be not solved with classical Boolean logic. In some situations only two values are insufficient to solve a problem.

2.2 Neural Logic Neurons

Among the several studies performed to simulate the behavior of the human neuron, we highlight those who sought to add fuzzy nature to the artificial neuron model, adding the ability to treat inaccurate information. This neuron is called fuzzy neuron [8]. This paper deals with a class of neurons called fuzzy logic neurons [16]. These neurons perform a mapping in the space formed by the Cartesian product between the input space and the space of the weights in the unit interval, i.e. $\mathbf{X} \times \mathbf{W} \rightarrow [0, 1]$ [8]. Examples of such neurons are neurons *and* and *or* [16] and unineuron [16, 14].

The logical *or* neuron uses a *t-norm* in the weighting operation and an *s-norm* in the final aggregation [8]. Given an input vector $\boldsymbol{x} = [x_1, x_2, \ldots x_n]$ and a vector of weights of neuron $\mathbf{w} = [w_1, w_2, \ldots w_n]$ for $a_i \in [0, 1]$ and $w_i \in [0, 1]$ for i of 1, …, n. The output of logical neuron *or* is described as [16]:

$$z = OR(x, w) = S_{i=1}^{n}(x_i t w_i) \tag{1}$$

where t is *t-norms* and S is s-norms. Figure 1 presents the structure of an OR-type neuron

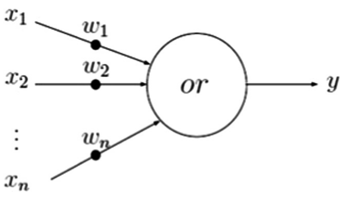

Fig. 1. *Orneuron* architecture

2.3 Fuzzy Neural Networks

The fuzzy neural networks have several characteristics from which we can distinguish the models through their basic properties, such as the way the network is connected, the type of fuzzy neurons used, the type of learning (training) and the way the inputs are handled in the first layers of the models. In this models, each layer is responsible for a specific function or task. Usually the first layer is responsible for handling the inputs and the last to bring the network response. Between these two layers there are other intermediate, which can be hidden or not. Depending on the model and what it is proposed, each layer has a specific function. Evaluating the type of training for fuzzy neural networks we can highlight that these algorithms are a set of well-defined rules for solving learning problems. These training methodologies seek to simulate human learning by learning or updating their new concepts, mainly by updating network factors, such as synaptic weights.

As the use of the extreme learning machine can be verified as a faster and more efficient alternative to adjust the parameters of a fuzzy neural network, it was proposed ways of modifying methodologies that act on these models, either the way of updating the parameters or the way of granulize the input in the model. In [4] a new methodologies to train fuzzy neural networks based also on extreme learning machine concepts, creating a model which they called XUninet. To train their fuzzy network, [17] used the extreme learning machine where weights of the hidden layer of a neural network are randomly chosen. To find the weights of the output layer we use the technique of recursive weighted least squares. For their algorithm, they defined the weights in the hidden layer and the identity elements of the uninorm [19] between zero and one. These values are updated recursively in training. Finally, [20] uses the ELM and to train the parameters of its network after the regularization method select the most representative neurons to the problem. In this context pattern classification techniques are used.

3 Fuzzy Neural Networks for Time Series Forecasting

3.1 Fuzzy Neural Networks Architecture

The fuzzy neural networks discussed below use the two types of neural logic neurons described in the previous subsection. The logical neurons that make up the network are described in (1). The structure of the network is illustrated in Fig. 2 in which the z-neurons are orneurons.

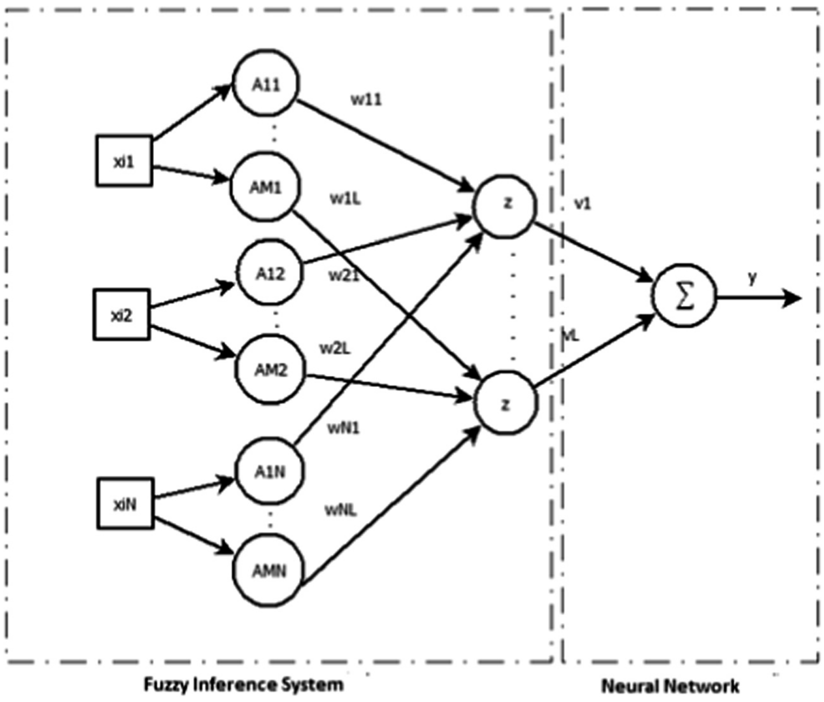

Fig. 2. Fuzzy neural network.

The architecture of the model used in this article presents its structure in [20], however changes are necessary so that the network can act as a model capable of predicting a time series. The first layer is defined as a fuzzification layer and is composed of neurons whose activation functions are membership functions of the respective fuzzy sets used in the partition of the input variables. For each input variable x_{ij} are defined M fuzzy sets $\mathrm{A}_{\mathrm{j}}^{\mathrm{m}}$, for m varying from $1\ldots M$. The outputs of the first layer are the degrees of membership associated with the input values, that is, $\mathrm{a}_{\mathrm{jm}} = \mu_{\mathrm{A}_{\mathrm{j}}^{\mathrm{m}}}$ for $j = 1, .., N$ at $= 1, \ldots, M$, where N is the number of inputs and M is the number of fuzzy sets for each input variable. The second layer is composed of L fuzzy neurons of the *orneuron* type. Each neuron performs a weighted aggregation of some outputs of the first layer. The fuzzy logic neurons perform the aggregation using the w_{il} weights (for $i = 1, \ldots, N$, and $l = 1, \ldots, L$). The strategy of creation of the fuzzy neurons uses the grid partition defined by the ANFIS model [10]. Finally, the output layer is composed of a single linear neuron. In [15, 20] used the output of the neuron adapted for pattern recognition, transforming their final responses into −1 or 1. For time series problems, we consider the following neuron:

$$y = \left(\sum_{j=0}^{L} z_j v_j\right) \tag{2}$$

where $z_0 = 1$, v_0 is bias, and z_j and v_j, $j = 1, \ldots, l$ are the output of each fuzzy neuron of the second layer and their corresponding weight, respectively. Fuzzy rules can be extracted from the network topology. To see how the fuzzy rules are generated, see [15, 20].

The ELM [9] is a learning algorithm developed for hidden layer feedforward neural networks (SLNFs) where random values are assigned to the weights of the first layer and the weights of the output layer are estimated analytically. In [15, 20] defined a training model for the fuzzy neural network where the parameters of the neurons are randomly assigned and the output parameters are calculated through least squares. The difference between them is the approach in creating the fuzzy rules of the first layer performed directly the amount of input data [20] and using the grid to divide the input space [15], in addition to [20] the neuron used to be the andneuron and in [20] there is also the use of unineuron [15].

This paper will use the same partitioning technique proposed in [15] where it will use equally spaced membership functions for each input variable to define the fuzzification layer neurons and the use of a smoothing technique to define the topology of the hidden layer. The model is able to generate parsimonious models, selecting more relevant neurons within the context of the problem. From the resulting model it is possible to extract a set of fuzzy rules.

The learning algorithm initially defines the neurons of the first layer through the partition of each interval of each input variable into M fuzzy sets with equally spaced Gaussian membership functions with its center at 0.5. Then, a strategy of partitioning the input space by a grid [10] is used to define an initial set of candidate neurons. The initial number of neurons in the hidden layer is defined as M^N, that is, for each possible combination of the membership functions of each input, a neuron is generated and its inputs are defined. The weights associated to the neuron inputs are randomly defined in the interval [0, 1], similarly to the ELMs. This approach to defining the network topology facilitates the interpretability of the extracted rules. For example, if three fuzzy sets are used per input variable ($M = 3$), each fuzzy set can be interpreted as "Small", "Medium" and "Large". Figure 3 presents the Gaussian relevance functions proposed for problem solving.

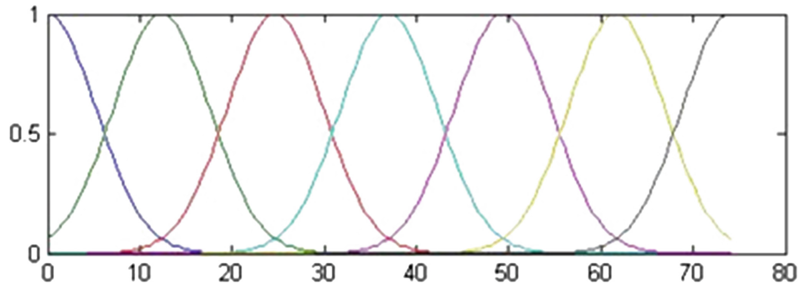

Fig. 3. Gaussian membership functions.

The final architecture of the network is defined by a feature extraction technique based on $L1$ regularization and resampling, called Bolasso [2]. The learning algorithm assumes that the output hidden layer composed of the candidate neurons can be written as:

$$f(x_i) = \sum_{l=0}^{L} v_l z_l(x_i) = z(x_i)v \tag{3}$$

where $\boldsymbol{v} = [v_0, v_1, v_2, \ldots, v_L]$ is the weight vector of the output layer and $\mathbf{z}(x_i) = [z_0, z_1(x_i), z_2(x_i)]$ 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. Since the weights connecting the first two layers are randomly assigned, the only parameters to be estimated are the weights of the output layer. Thus, the problem of network parameter estimation can be seen as a simple linear regression problem, allowing the use of regression techniques [7] for estimating parameters and selecting candidate neurons. The regression algorithm used by the model proposed by [15] for high-dimensional data estimating the regression coefficients and the subset of candidate regressors to be included in the final model is the LARS [6]. When we evaluate a set of K distinct samples (x_i, y_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{iN}] \,\varepsilon\, \mathbb{R}$ and $y_i \,\varepsilon\, \mathbb{R}$ for $i = 1, \ldots, K$, the cost function of this regression algorithm can be defined as:

$$\sum_{i=1}^{K} \|z(x_i)v - y_i\|_2 + \lambda\|v\|_1 \tag{4}$$

where λ is a regularization parameter of L_1 norm, commonly estimated via cross-validation [15].

The LARS algorithm is used to perform the model selection, since, for a given value of λ only a fraction (or none) of the regressors have corresponding weights other than zero. For the problem considered in this work, the regressors z_{ls} are the outputs of the significant neurons. Thus, the LARS algorithm can be used to select an optimal subset of the significant neurons (L_s) that minimize (4) for a given value of λ. The approach used to increase the stability of the model selection algorithm is the use of resampling. This procedure developed by [2] is defined as a bootstrap-enhanced least absolute shrinkage operator where LARS algorithm runs on several bootstrap replications of the training data set. For each repetition, a distinct subset of the regressors is selected. The regressors to be included in the final model are defined according to the frequency with which each of them is selected through different tests. A consensus threshold is defined, say $\gamma = 60\%$, and a regressor is included, if selected in at least 60% of the assays. Finally, after the definition of the network topology, the calculations of the estimation of the vector of weights of the output layer are performed. In this paper, this vector is estimated by the Moore-Penrose pseudo Inverse:

$$v = Z^{+}y \tag{5}$$

where Z^+ is pseudo-inverse of Moore-Penrose of Z which is the minimum norm of the solution of the least squares for the weights of the exit. The learning process can be synthesized as demonstrated in Algorithm 1. It has three parameters:

- the number of fuzzy sets that will partition the input space, M;
- the number of bootstrap replications, b;
- the consensus threshold, γ.

Algorithm 1- Learning Algorithm for Fuzzy Neural Networks Proposed in time series forecasting

```
Define $M$ equally spaced fuzzy sets for each input variable.
Define candidate neurons $(L)$
For all $K$ entries do
Calculate the mapping $\mathbf{z}(x_i)$
end for
Select $Ls$ significant neurons using bootstrap lasso.
Estimate the weights of the output layer (5)
```

4 Tests and Experiments

The learning model of normalized fuzzy neural networks was evaluated through numerical experiments of time series prediction. A time series, $\mathbf{x}(t)$, can be defined as a function of an independent time t variable, tied to a process in which a mathematical description is considered to be unknown. Its most relevant feature is that its future behavior can not be predicted exactly, as can be predicted from a deterministic function, known at t. However, the behavior of a time series can sometimes be anticipated through stochastic procedures. The database used is the time series of Box-Jenkings (Gas-Furnace). The gas furnace of Box and Jenkins [5] consists of a furnace where u_k is the feed rate of methane gas (cubic feet per minute) and the output y_k is the concentration of carbon dioxide (% CO_2) in a gas mixture. A set of 296 samples (pairs of input and output data) is available for identification. The normalized data set represents the concentration of CO_2, and k, from the values y^{k-1} and u^{k-4}. See more in [12]. The studies in [5] state that a suitable model to act on this data set is in the form of:

$$y'^{k} = f\left(y^{k-1}, u^{k-4}\right) \tag{6}$$

Figure 4 shows the input data of the experiment and the output data of the base used in the experiments.

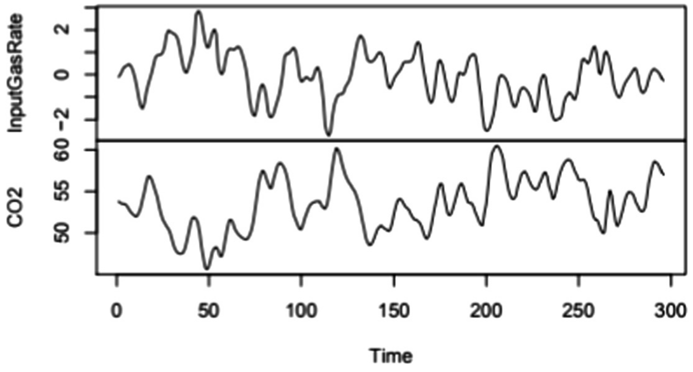

Fig. 4. Sample of the gas furnace input and output data.

The experiment was set up to use 200 samples for the neural network training phase and 96 samples for the validation phase of the model for this time series. All samples were normalized with mean zero and unit variance. In all experiments, the test assumptions defined in [15] were considered, as well as Gaussian activation functions. The performance of the proposed model was evaluated using the Root Mean Square Error (RMSE). The RMSE was calculated in the same way as in [13].

$$RMSE = \left(\frac{1}{N}\sum\nolimits_{k=1}^{n}\left(y^{k} - y'^{k}\right)\right)^{\frac{1}{2}} \tag{7}$$

In the tests carried out using Matlab, we tried to verify the ability of the learning model proposed in this paper to improve the structure of the network through the method of definition of the proposed structure, in addition to verifying that the method has the capacity to work in solving problems of time series. Table 1 shows the rules used (best value in the test), the best value of RMSE obtained in the test and the average value of RMSE for the model used in this work, besides presenting a comparison with the results of fuzzy neural networks commonly employed in time series problems. For a version where it is desired to evaluate the best network, RMSE can be considered as the value to be considered for analysis, but the mean value of RMSE can help in the definition of a more stable model. To avoid that the choice of the parameters of the models used interfere in the accuracy of the final training of the models, each algorithm was executed 30 times and the RMSE mean values were the indices used for the comparison.

The algorithm proposed in this paper was compared with other efficient methodologies to solve time series problems that are widely used in the literature. R-ORNEURON is considered the network formed by logical neurons composed by neurons or. The other fuzzy neural network models used were the DENFIS, proposed by [11], the FbeM, proposed by [12], the XUninet, developed by [4], eRFH, proposed by [17], the model of [13] based on uninorms, called in this paper of UN-RNN and also proposed by [14] FL-RNN which deals with a rapid learning approach, the model eTS, developed by [1]. Table 1 summarizes the results obtained.

Table 1. Performance evaluation of the algorithms.

Models	Rules	RMSE	Standard deviation	RMSE average
DENFIS	12	0.021	0.005	0.021
FEeM	3	0.052	0.012	0.052
XUninet	13	0.038	0.432	0.048
eRFH	9	0.027	0.004	0.031
eTS	7	0.066	0.065	0.066
NU-RNN	6	0.052	0.008	0.052
FL-RNN	6	0.047	0.014	0.046
OrNeuron	**2**	**0.019**	**0.007**	**0.020**

The results of Table 1 allow an analysis that the proposed model uses a smaller number of rules to solve the problem and presents better RMSE to the models that are traditionally used in the literature to solve time series problems. Figure 5 shows the result obtained by the *OrNeuron* model in the final validation of the results.

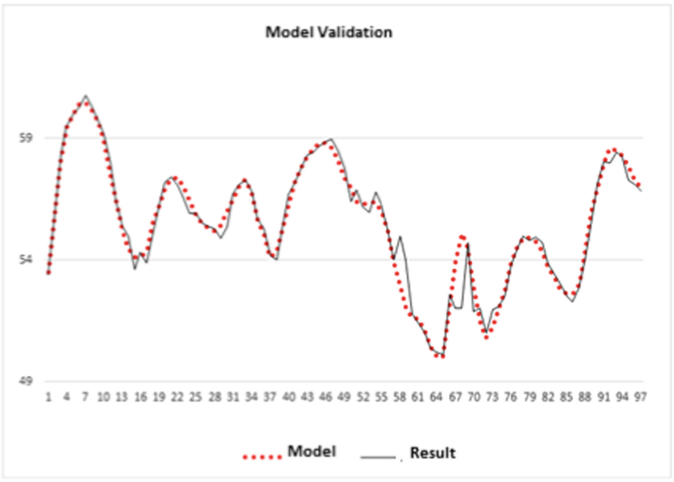

Fig. 5. Validation of the model.

5 Conclusion

This paper presents a new way to use regularized fuzzy neural networks based on the concepts of extreme learning machine to act in the forecast of time series. The method presented highlighted numerical results compared to the models of fuzzy neural networks to act in time series prediction, in addition to using a smaller number of fuzzy rules to solve the problem. The experiments performed and the results suggest that the network is able to act as a model capable of solving time series, presenting consistent results and close to the results obtained by models commonly used for this purpose in the literature. Future actions can be taken so that the model is submitted to other types of time series models and to regression problems and their results with statistical tests.

References

1. Angelov, P.P., Filev, D.P.: An approach to online identification of Takagi-Sugeno fuzzy models. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **34**(1), 484–498 (2004)
2. Bach, F.R.: Bolasso: model consistent lasso estimation through the bootstrap. In: Proceedings of the 25th International Conference on Machine learning, pp. 33–40. ACM, July 2008
3. Bordignon, F.L.: Aprendizado extremo para redes neurais fuzzy baseadas em uninormas (2013)
4. Bordignon, F., Gomide, F.: Uninorm based evolving neural networks and approximation capabilities. Neurocomputing **127**, 13–20 (2014)
5. Box, G.E., Box, G.M.J., Gregory, C.R.: Time series analysis: forecasting and control, No. 04, QA280, B6 1994 (1994)
6. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Ann. Stat. **32**(2), 407–499 (2004)

7. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. vol. 2, no. 1. Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7
8. Hell, M.B.: Abordagem neurofuzzy para modelagem de sistemas dinamicos não lineares. Doctoral dissertation, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas (UNICAMP) (2008)
9. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1), 489–501 (2006)
10. Jang, J.S.: ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans. Syst. Man Cybern. **23**(3), 665–685 (1993)
11. Kasabov, N.K., Song, Q.: DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Trans. Fuzzy Syst. **10**(2), 144–154 (2002)
12. Leite, D., Gomide, F., Ballini, R., Costa, P.: Fuzzy granular evolving modeling for time series prediction. In: 2011 IEEE International Conference on Fuzzy Systems (FUZZ), pp. 2794–2801. IEEE, June 2011
13. Lemos, A., Caminhas, W., Gomide, F.: New uninorm-based neuron model and fuzzy neural networks. In: 2010 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS), pp. 1–6. IEEE, July 2010
14. Lemos, A.P., Caminhas, W., Gomide, F.: A fast learning algorithm for uninorm-based fuzzy neural networks. In: 2012 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS), pp. 1–6. IEEE, August 2012
15. Murphy, K.P.: Machine learning: a probabilistic perspective. MIT Press, Cambridge (2012)
16. Pedrycz, W.: Fuzzy neural networks and neurocomputations. Fuzzy Sets Syst. **56**(1), 1–28 (1993)
17. Rosa, R., Gomide, F., Ballini, R.: Evolving hybrid neural fuzzy network for system modeling and time series forecasting. In: 2013 12th International Conference on Machine Learning and Applications (ICMLA), vol. 2, pp. 378–383. IEEE, December 2013
18. Souza, P.V.C., Lemos, A.P.: Redes Neurais Nebulosas para problemas de Classificação. In: Simpósio Brasileiro de Automação Inteligente, Natal-RN. XII SBAI-Simpósio Brasileiro de Automação Inteligente (2015)
19. Yager, R.R., Rybalov, A.: Uninorm aggregation operators. Fuzzy Sets Syst. **80**(1), 111–120 (1996)
20. Souza, P.V.C.: Regularized fuzzy neural networks for pattern classification problems. Int. J. Appl. Eng. Res. **13**(5), 2985–2991 (2018)
21. Dash, P.K., Ramakrishna, G., Liew, A.C., Rahman, S.: Fuzzy neural networks for time-series forecasting of electric load. IEE Proc.-Gener. Transm. Distrib. **142**(5), 535–544 (1995)
22. Maguire, L.P., Roche, B., McGinnity, T.M., McDaid, L.J.: Predicting a chaotic time series using a fuzzy neural network. Inf. Sci. **112**(1–4), 125–136 (1998)
23. Braga, A.D.P., Carvalho, A.P.L.F., Ludermir, T.B.: Redes neurais artificiais: teoria e aplicações, pp. 5–55. Livros Técnicos e Científicos (2000)
24. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**(3), 338–353 (1965)