International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 5 (2018) pp. 2985-2991

Regularized Fuzzy Neural Networks for Pattern Classification Problems

Paulo Vitor C. Souza

System Analyst, Secretariat of Information Governance- CEFET-MG,
Teacher, University Center Of Belo Horizonte- UNI-BH,
Belo Horizonte-MG,Brazil.

Abstract

This paper presents a novel learning algorithm for fuzzy logic neuron based networks able to generate accurate and transparent models. The learning algorithm is based on ideas from Extreme Learning Machine, to achieve a low time complexity, and regularization theory, resulting in sparse and accurate models. A compact set of incomplete fuzzy rules can be extracted from the resulting network topology. Experiments considering pattern classification are detailed. Results suggest the proposed approach as a promising alternative for pattern recognition with a good accuracy and some level of interpretability.

Keywords: Extreme Learning Machine, Fuzzy Neural Networks, Regularization Theory, Pattern Recognition.

INTRODUCTION

Fuzzy neural networks are networks based on fuzzy logic neurons [1]. These models are a synergy between fuzzy sets theory, as a mechanism for knowledge representation and information compactation, and neural networks. The main feature of these models is transparency since a set of fuzzy rules can be extracted from the network structure after training [2]. Furthermore, these models have a neural network topology which enables the utilization of a large variety of existing machine learning algorithms for structure identification and parameter estimation. Fuzzy neural networks have already been used to solve several distinct problems including pattern classification [2], time series prediction [3]–[5] and dynamic system modelling [6]–[9].

Examples of fuzzy neurons are *and* and *or* neurons [10]. These logic based neurons are nonlinear mappings of the form $[0, 1]^N \rightarrow [0, 1]$, where N is the number of inputs. The processing of these neurons is performed in two steps. Firstly, the input signals are individually combined with connection weights. Next, an overall aggregation operation is performed over the results obtained in the first step *and* and *or* neurons use t-norms and s-norms (t-conorms) to performed output processing.

Several learning algorithms for fuzzy neural networks have already been proposed in the literature. Usually, learning is performed in two steps. Firstly, the network topology is defined. This step involves defining fuzzy sets for each input variable, selecting a suitable number of neurons and defining network connections. The most commonly used methods for structure definition are clustering [2], [3], [5], [6], [8], [9] and evolutionary optimization [11]–[13]. Once the network structure is defined, free parameters are estimated. A number of distinct methods have already been used in this step including reinforcement learning [2], [3], [6], gradient based methods [12], [14], genetic algorithms [8] and least squares [5], [9].

Regarding network structure optimization, the two most commonly used methods may present significant deficiencies. Clustering has a low computational cost when compared with evolutionary optimization. Nevertheless, interpretable fuzzy rules usually can't be extracted from the resulting network, since fuzzy sets generated by clustering are typically difficult to be interpreted [15]. Evolutionary optimization based methods may be able to generate interpretable fuzzy rules. However they have a high computational complexity.

This paper proposes a novel learning algorithm for fuzzy neural networks able generate compact and accurate models. The learning is performed using ideas from Extreme Learning Machine [16], to speed-up parameter tunning, and regularization theory [17]. First, fuzzy sets are generated for each input variable. Next, a large number of candidate fuzzy neurons are created with randomly assigned weights. In this step, only a random fraction of the input variables are used in each candidate neuron. Following, the bootstrap Lasso algorithm [18] is used to define the network topology by selecting a subset of the candidate neurons. Finally, the remaining network parameters are estimated through least squares. The resulting network is a sparse model and can be expressed as a compact set of incomplete fuzzy rules, that is, rules with antecedents defined using only a fraction of the available input variables [15].

The paper is organized as follows. Next section reviews the necessary basic concepts about fuzzy logic neurons and fuzzy neural networks. Section III details the proposed new learning algorithm able to generate compact and transparent networks. Section IV presents the experimental results for pattern classification problems and comparison with alternative classifiers. Finally, the conclusions and further developments are summarized in section V.

FUZZY NEURAL NETWORKS

A. Fuzzy Logic Neurons

Fuzzy logic neurons are functional units able to perform multivariate nonlinear operations in unit hypercube $[0, 1]^N \rightarrow [0, 1]$ [1], where N is the number of inputs. The term "logic" is associated with the logic disjunction or and conjunction and operations performed by these neurons using, in this work, t-norms and s-norms.

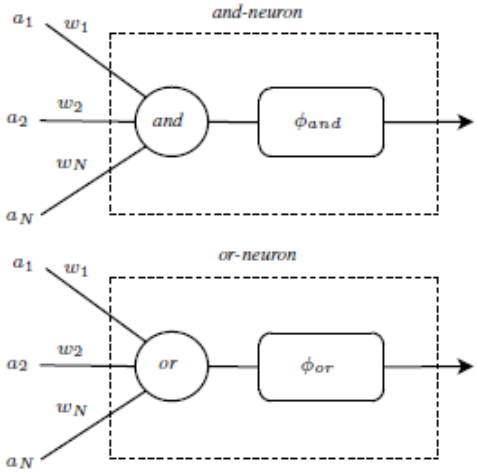

Figure 1. Fuzzy Logic Neurons

The *or* and *and* neurons aggregates input signals $a = [a_1, a_2, \ldots, a_N]$ by firstly combining them individually with the weights $w = [w_1, w_2, \ldots, w_N]$, where $a_j \in [0, 1]$ and $w_j \in [0, 1]$ for $j = 1, \cdots, N$, and subsequently globally aggregating these results. They were initially defined as follows [1]:

$$h = OR(a, w) = S^n{}_{i=1}(a_i t w_i) \quad (1)$$

$$h = AND(a, w) = T^n{}_{i=1}(a_i s w_i) \quad (2)$$

where T and t are t-norms and S and s are s-norms.

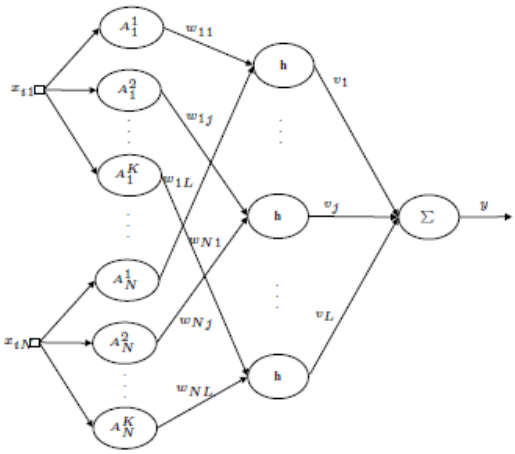

Figure 2. Feedforward fuzzy neural network

The or-neuron is interpreted as a logic expression that performs an and-type local aggregation of the inputs and the weights using a t-norm, followed by an or-type global aggregation of the results using an s-norm. The and-neuron is interpreted similarly, with an or-type local aggregation and of the associated input. Figure 1 illustrates these neurons, as defined in [1].

The activation functions φ_{and} and φ_{or} can, in general, be nonlinear mappings. In this paper $\varphi_{and}(\xi) = \varphi_{or}(\xi) = \xi$, i.e., they are defined as the identity function.

According to [11], the local aggregations performed by these neurons can be interpreted as weighting operations of the inputs, since the role of the weights is to differentiate among distinct levels of impact that individual inputs might have on the global aggregation result. In the case of the *or*-neuron, lower values of w_j reduces the impact of the corresponding input, while higher values do not affect the original value of the corresponding input. In limit, if all weights are set to 1, the neuron output is a plain *or* combination of the inputs. For the *and*-neuron, the interpretation of the weight's values is inverse, i.e., higher values of w_j reduces the impact of the corresponding input. For this neuron, in the limit, if all weights are set to 0, the output is a plain *and* combination of the inputs.

In order to unify the interpretation of the weights for the *and* and *or* neurons, (i.e., a low value of w reduces the impact of the associated input) the *and*-neuron used in this paper is defined as:

$$h = AND(a, w) = T^n{}_{i=1}(a_i s(1 - w)_i) \quad (3)$$

B. Fuzzy Neural Networks

The fuzzy logic neurons described in the previous section can be used to construct fuzzy neural networks and solve pattern recognition problems. Figure 2 illustrates the feed forward topology of the fuzzy neural networks considered in this paper.

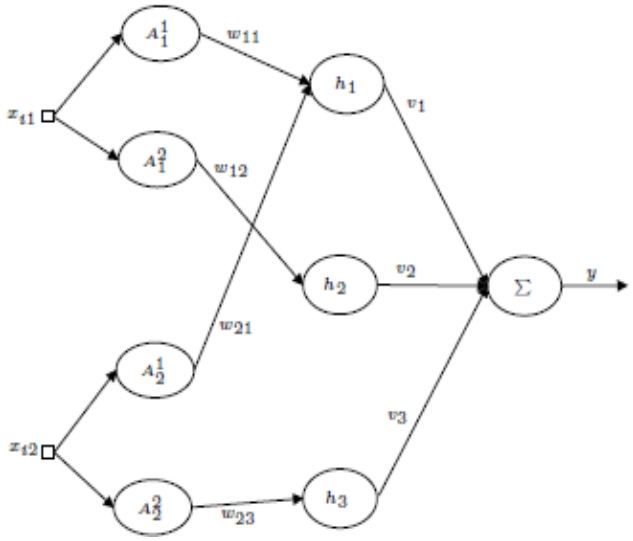

Figure 3. An example of a fuzzy neural network

The first layer is composed by neurons whose activation functions are membership functions of fuzzy sets defined for the input variables. For each input variable x_{ij}, K fuzzy sets are defined A_k, $k = 1 \ldots K$ 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., $a_{jk} = \mu_{A^k}$. for $j = 1 \ldots, N$ and $k = 1, \ldots, K$, where N is the number of inputs and K is the number of fuzzy sets for each input. The second layer is composed by L fuzzy logic neurons. Each neuron performs a weighted aggregation of some of the first layer outputs. This aggregation is performed using the weights w_{il} (for $i = 1 \ldots N$ and $l = 1 \ldots L$). For each input variable j, only one first layer output a_{jk} is defined as input of the l-th neuron. Furthermore, in favor of generating sparse topologies, each second layer neuron is associated with only $n_l < n$ input variables, that is, the weight matrix w is sparse.

Finally, the output layer uses a classic linear perceptron neuron to compute the network output:

$$y = \sum_{j=0}^{L} h_l v_l \quad (4)$$

where h_l for $l = 1, \ldots, L$ are the outputs of second layer neurons, v_l are the output layer weights and $h_0 = 1$.

As discussed, incomplete fuzzy rules can be extracted from the network topology. Figure 3 illustrates an example of a fuzzy network composed by and-neurons. This network has 2 input variables, 2 fuzzy sets for each variable and 3 neurons, i.e., $M = 2$, $K = 2$ and $L = 3$. The following if-then rules can be extracted from the network structure:

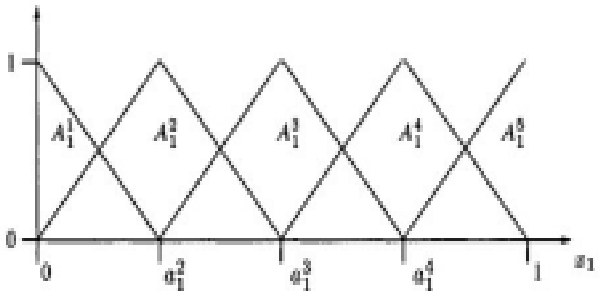

Figure 4. Input domain partition using 5 triangular membership functions

Rule1: If x_{i1} is $A^1{}_1$ with certainty $w_{11} \cdots$

and x_{i2} is $A^1{}_2$ with certainty $w_{21} \cdots$

Then y_1 is v_1

Rule2: If x_{i1} is $A^2{}_1$ with certainty $w_{12} \cdots$ (5)

Then y_2 is v_2

Rule3: If x_{i2} is $A^2{}_2$ with certainty $w_{23} \cdots$

Then y_3 is v_3

where each free parameter v_l for $l = 1, \cdots, 3$ can be interpreted as a singleton.

NETWORK TRAINING ALGORITHM

The learning algorithm detailed in [8] for fuzzy neural networks uses clustering for topology definition and genetic algorithms for parameter tuning. This learning procedure is able to generate accurate models. However, the resulting network is usually not interpretable, since the fuzzy sets for each input variable are defined using clustering [15]. Furthermore, the learning algorithm has a high time complexity, given the nature of the parameter tuning algorithm.

In [9] the genetic algorithm is replaced by a novel parameter tuning procedure based on ideas from ELM. Extreme Learning

Machine is a learning algorithm originally developed for single hidden layer feed forward neural networks (SLFNs). The algorithm assigns random values for the first layer weights and analytically estimates the output layer weights. When compared with traditional methods for SLFNs learning, this algorithm has good generalization performance and extremely low time complexity, since only the output layer parameters are tuned [19]. It has been proved that SLFNs trained using this approach have a universal approximation property, for a Fig. 4. Input domain partition using 5 triangular membership functions given choice of the hidden nodes [16]. Inspired by this SLFN learning scheme, a new fast learning algorithm for fuzzy neural networks is described in [9]. This approach assigns random values for the neuron parameters and estimates the parameters of the output layer neuron using least squares. This learning algorithm has a low computational cost, but still generates networks which are not easily interpretable, since the fuzzy sets are still defined using clustering.

This paper proposes an improvement in the learning algorithm described in [9] in order to improve the interpretability of the resulting networks. A variable selection algorithm based on regularization theory is used to define the network topology. This algorithm is able to generate a sparse topology which can be interpreted as a compact set of incomplete fuzzy rules. The proposed learning algorithm initially defines first layer neurons by dividing each input variable domain interval into K fuzzy sets, where K is usually a small number, as shown in Figure 4 for $K = 5$. All input variables are partitioned using the same number of fuzzy sets.

Next, L_c candidate neurons are randomly generated, where $L_c < L$. For each candidate neuron $l = 1, \ldots, L_c$, first, a random fraction of the input variables is selected. A random value n_l is sampled from a discrete uniform distribution defined over the interval $[1, N]$.

The value n_l represents the number of input variables associated with the l-th neuron. Next, n_l input variables are randomly selected. For each input variable chosen, a random fuzzy set (first layer neuron) is selected as input of the l-th neuron and the corresponding weight w_{jl} is sampled from a uniform distribution over the interval [0, 1]. One must note that, after this step, for each neuron l, only n_l weights w_{jl} will have nonzero values, i.e., only the weights associated with the selected input variables. Once the candidate neurons are created, the final network topology is defined by selecting an optimal subset of these neurons. This procedure can be seen as a variable selection problem in which one has to find the

optimum subset of variables (in this case, neurons) for a given cost function. The learning algorithm assumes that the output of a network composed by all L_c candidate neurons can be written as:

$$(x_i) = \sum_{l=0}^{Lc} v_l h_l(x_i) = h(x_i)\, v \qquad (6)$$

where $\mathbf{v} = [v_0, v_1, v_2, \ldots, v_L]^T$ is the output layer weight vector and $\mathbf{h}(x_i) = [h_0, h_1(x_i), h_2(x_i), \ldots, h_{Lc}(x_i)]$ is the augmented output (row) vector of the second layer, for h_0=1.

In this context, $\boldsymbol{h(x_i)}$ is non-linear mapping from the input space to a L_c + 1-dimensional fuzzy feature space, performed using the candidate neurons.

Please note that Equation (6) can be seen as a simple linear regression model, since the weights connecting the first two layers are randomly assigned and the only parameters left to be estimated are the weights of the output layer. Thus, the problem of selecting the best candidate neurons subset can be seen as an ordinary linear regression model selection problem [20]. One commonly used approach for model selection is the Least Angle Regression (LARS) algorithm [21]. The LARS is a regression algorithm for high-dimensional data which is able to estimate not only the regression coefficients but also a subset of candidate regressors to be included in the final model. Given a set of M distinct observations (x_i, y_i), where $\mathbf{x_i} = [x_{i1}, x_{i2}, \ldots, x_{iN}]^T \in R^N$ and $y_i \in R$ for $i = 1, \ldots, M$, the cost function of this regression algorithm is defined as:

$$\sum_{i=1}^{M} \|h(x_i)v - y_i\|_2 + \lambda \|v\|_1 \qquad (7)$$

where λ is a regularization parameter usually estimated using cross-validation [22].

The first term of (7) corresponds to the residual sum of squares (RSS). This term decreases as the training error decreases. The second is a ℓ_1 regularization term. This term is used for two reasons. First, it improves the network generalization, avoiding over fitting [17]. Second, it can be used to generate sparse models [22].

In order to understand why LARS can be used as a variable selection algorithm, Equation (7) is rewritten as:

$$\min_{v} RSS(v) \quad \text{s.t} \quad \|v\|_1 \leq \beta \qquad (8)$$

where β is an upper-bound on the $\ell 1$-norm of the weights. A small value of β corresponds to a large value of λ, and vice-versa. This equation is known as "least absolute shrinkage and selection operator" [20], or lasso.

Figure 5 illustrates the contours of the RSS objective function, as well as the $\ell 1$ constraint surface. It is well known from the theory of constrained optimization that the optimum solution corresponds to the point where the lowest level of the objective function intersects the constraint surface. Thus, one should note that, when β grows until it meets the objective function, the points in the corners of the $\ell 1$ surface (i.e., the points on the coordinate axes) are more likely to intersect the RSS ellipse than one of the sides [22].

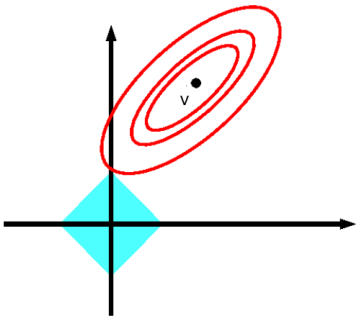

Figure 5. ℓ_1 regularization

The LARS algorithm can be used to perform model selection since, for a given value of λ only a fraction (or none) of the regressors have non-zero corresponding weights.

If $\lambda = 0$, the regression problem becomes unconstrained, and all weights are non-zero. As λ increases from 0 to a certain λ_{max} value, the number of non-zero weights decreases from N to 0. For the problem considered in this paper, the regressors h_l are the outputs of the candidate neurons. Thus, the LARS algorithm can be used to select an optimum subset of the candidate neurons which minimizes (8) for a given value of λ, selected using cross-validation.

One most note that this is not a novel approach and has al- ready been used for SLFN pruning [23], [24] and fuzzy model identification [25]. However, one disadvantage of using the LARS algorithm for model selection is that it can give quite different results for small perturbations in the dataset [22]. If this algorithm is executed for two distinct noisy datasets drawn from the same problem very distinct network topologies may be selected, compromising model interpretability.

One existing approach to increase the stability of this model selection algorithm is to use bootstrap resampling. The LARS algorithm is executed on several bootstrap replicates of the training dataset. For each replicate considered, a distinct subset of the regressors are selected. The regressors to be included in the final model are defined according to how often each one of them is selected across different trials. A consensus threshold is defined, say γ = 80%, and a regressor is included if it is selected in at least 80% of the trials. This algorithm is known as bootstrap lasso [18]. In this paper the bootstrap lasso algorithm is used for topology definition.

Finally, once the network structure is defined, the learning algorithm has only to estimate the output layer vector $\mathbf{v} = [v_0, v_1, v_2 \ldots v_L]^T$ which best fits the desired outputs. In this paper these parameters are computed using the Moore- Penrose pseudo-inverse:

$$v = (H^T H)^{-1} H^T Y = H^+ Y \qquad (9)$$

where H is defined as:

$$H = \begin{bmatrix} h_0 & h_1(x_1) & h_2(x_1) & \dots & h_L(x_1) \\ h_0 & h_1(x_2) & h_2(x_2) & \dots & h_L(x_2) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ h_0 & h_1(x_N) & h_2(x_N) & \dots & h_L(x_N) \end{bmatrix} \quad (10)$$

The learning procedure is summarized in Algorithm 1. The algorithm has four parameters:

• the number of fuzzy sets for input space partition, K;

• the number of candidate neurons, L_c;

• the number of bootstrap replicates, b;

• the consensus threshold, γ.

Algorithm 1- Learning algorithm for Fuzzy Neural

Networks Define K equally spaced fuzzy sets for each input variable Create L_c candidate neurons
for all N inputs **do**
Compute the mapping $h(x_i)$
end for
Select L candidate neurons using bootstrap lasso
Estimate the output layer weights (9)

EXPERIMENTS

The fuzzy neural network learning algorithm described in the previous section is evaluated using pattern classification problems. For all experiments in this section, only networks composed by *and*-neurons are considered. All *and*-neurons use the product t-norm and the probabilistic sum s-norm and only Gaussian fuzzy sets are used. The network used in the experiments was named *Regularized ANDNET*.

First, a simple toy binary classification problem is considered to illustrate the transparency of the resulting network. Next, the accuracy of the *Regularized ANDNET* is evaluated for benchmark binary classification problems.

A. Artificial Classification Dataset

Initially, the transparency of the *Regularized ANDNET* was evaluated using a toy problem. An artificial dataset with 20 samples was generated from a mixture of 2 Gaussian distributions, as shown in Figure 6. The algorithm parameters were set to: $K = 2$, $L_c = 10$, $b = 16$ and $\gamma = 100\%$. Two thirds of the samples were used for training and one third for performance evaluation.

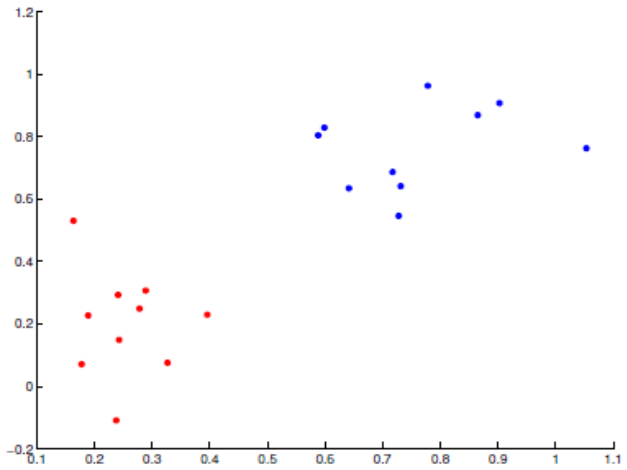

Figure 6. Artificial Classification dataset

The resulting network is able to classify all test set samples correctly and has only 2 neurons, which can be interpreted as the following fuzzy rules:

Rule$_1$: If x_{i1} is Small with certainty 0.80 · · ·

Then y_1 is -0.34 (11)

Rule$_2$: If x_{i1} is Big with certainty 0.31 · · ·

and x_{i2} is Big with certainty 0.92 · · ·

Then y_2 is 0.38

B. Benchmark Classification Datasets

In this section the proposed learning algorithm is evaluated using benchmark binary classification datasets taken from the UCI Machine Learning repository [26].

In order to verify the performance of the Regularized ANDNET, datasets with different sizes and input dimensions were used. The following binary datasets were considered: the Statlog Australian credit (acr), the Statlog German credit (gcr), the Statlog heart disease (hea), the Johns Hopkins university ionosphere (ion), the Pima Indians diabetes (pid) and the Wisconsin breast cancer (wbc). The main characteristics of these datasets are summarized in Table I.

TABLE I
SPECIFICATION OF BINARY CLASSIFICATION DATASETS

Dataset	# features (n)	#train	# test
acr	14	460	230
gcr	20	666	334
hea	13	180	90
ion	33	234	117
pid	8	512	256
wbc	9	455	228

All observations with missing values were removed and the outputs were normalized to be in $[-1, 1]$. The inputs were normalized to zero mean and unit variance. Two thirds of the data samples were randomly selected for training and the remaining for performance evaluation. For all experiments, the algorithm parameters were set to: $K = 3$, $b = 32$, $\gamma = 70\%$ and the number of candidate neurons, Lc, was set to half of the training set size of each dataset.

The performance of the proposed approach was compared with two alternative classifiers. A fuzzy neural network com- posed by and-neurons and trained using the learning algorithm described in [9], the ANDNET, and a state-of-the art pruning algorithm for ELMs, the OP-ELM [23]. The number of candidate neurons for the OPL-ELM was also set to half of the training set size and sigmoidal neurons were considered. The number of neurons for the ANDNET was estimated using cross-validation. Since some parameters of these models are selected randomly and may affect the final training results, each approach was run 30 times and the mean values of the performance indexes used for comparison.

Table II details the test set accuracy for each dataset. Table III details the average number of neurons after training. The results presented in these tables suggest that the Regularized ANDNET performs better than ANDNET and has a similar performance as OP-ELM in most of the experiments. Furthermore, for all experiments, the resulting network has a very simple topology (small number of neurons), if compared with the other models.

Table II: Performance Comparison for Binary Classification Datasets

Dataset	*ANDNET*	OP-ELM	*Regularized ANDNET*
acr	75.26(3.54)	84.17(1.97)	85.52(1.72)
gcr	68.26(2.39)	72.37(2.43)	72.43(2.00)
hea	74.78(4.80)	81.00(3.49)	79.22(4.91)
ion	69.06(3.12)	87.26(4.27)	89.23(2.77)
pid	67.73(3.80)	73.24(2.51)	76.02(2.04)
wbc	96.32(0.93)	96.45(1.31)	96.10(1.41)

Table III: Number of Neurons for Binary Classification Datasets

Dataset	*ANDNET*	OP-ELM	*Regularized ANDNET*
acr	81.30(12.28)	74.00(12.95)	10.40(3.63)
gcr	42.80(26.74)	82.50(22.12)	18.40(8.10)
hea	30.50(23.41)	24.00(12.52)	10.40(4.35)
ion	16.60(5.15)	51.50(8.96)	15.10(4.25)
pid	43.70(22.39)	73.00(26.37)	6.10(3.28)
wbc	58.00(20.79)	90.00(15.78)	9.80(2.44)

CONCLUSION

This paper has introduced a new learning algorithm for fuzzy neural networks based on ideas from Extreme Learning Machine and regularization theory. Random parameters are defined for the second layer neurons and the bootstrap lasso algorithm is used to generate sparse models. The experiments performed and the results suggest the network as a promising alternative to build accurate and transparent models. For all experiments considered, the net- work trained by the proposed method was able to achieve a similar accuracy when compared with a state-of-the-art pruning algorithm for ELMs while generating simpler models. Furthermore, the resulting models have some level of interpretability since incomplete fuzzy rules can be extracted from the network topology.

Future work shall address methods to improve the network interpretability and evaluation on multi-class classification problems.

ACKNOWLEDGMENT

The author acknowledges the support of UNI-BH for financial support.

REFERENCES

[1] W. Pedrycz, "Fuzzy Neural Networks and Neurocomputations," Fuzzy Sets and Systems, vol. 56, no. 1, pp. 1-28, MAY 25 1993.

[2] W. Caminhas, H. Tavares, F. Gomide, and W. Pedrycz, "Fuzzy sets based neural networks: Structure, learning and applications," Journal of Advanced Computational Intelligence, vol. 3, no. 3, pp. 151-157, 1999.

[3] R. Ballini and F. Gomide, "Learning in recurrent, hybrid neurofuzzy networks," in IEEE International Conference on Fuzzy Systems, 2002, pp. 785-791.

[4] M. Hell, F. Gomide, R. Ballini, and P. Costa, "Uninetworks in time series forecasting," in NAFIPS 2009. Annual Meeting of the North American Fuzzy Information Processing Society, 2009., june 2009, pp. 1 -6.

[5] F. Bordignon and F. Gomide, "Uninorm based evolving neural networks and approximation capabilities," Neurocomputing, vol. 127, pp. 13-20, 2014.

[6] M. Hell, P. Costa, and F. Gomide, "Participatory learning in power transformers thermal modeling," IEEE Transactions on Power Delivery, vol. 23, no. 4, pp. 2058-2067, Oct. 2008.

[7] A. E. Gobi and W. Pedrycz, "Logic minimization as an efficient means of fuzzy structure discovery," IEEE Transactions on Fuzzy Systems, vol. 16, no. 3, pp. 553-566, JUN 2008.

[8] A. Lemos, W. Caminhas, and F. Gomide, "New uninorm-based neuron model and fuzzy neural networks," in 2010 Annual Meeting of the North

American Fuzzy Information Processing Society (NAFIPS), uly 2010, pp. 1 -6.

[9] A. P. Lemos, W. Caminhas, and F. Gomide, "A fast learning algorithm for uninorm-based fuzzy neural networks," in Fuzzy Information Pro- cessing Society (NAFIPS), 2012 Annual Meeting of the North American. IEEE, 2012, pp. 1-6.

[10] F. Gomide and W. Pedrycz, Fuzzy Systems Engineering: Toward Human- Centric Computing. NJ, USA: Wiley Interscience, 2007.

[11] W. Pedrycz, "Heterogeneous fuzzy logic networks: Fundamentals and development studies," IEEE Transactions on Neural Networks, vol. 15, no. 6, pp. 1466-1481, NOV 2004.

[12] --, "Logic-based fuzzy neurocomputing with unineurons," IEEE Transactions on Fuzzy Systems, vol. 14, no. 6, pp. 860-873, DEC 2006.

[13] X. Liang and W. Pedrycz, "Logic-based fuzzy networks: A study in system modeling with triangular norms and uninorms," Fuzzy sets and systems, vol. 160, no. 24, pp. 3475-3502, 2009.

[14] W. Pedrycz and R. A. Aliev, "Logic-oriented neural networks for fuzzy Neurocomputing," Neurocomputing, vol. 73, no. 1-3, pp. 10 - 23, 2009.

[15] S. Guillaume, "Designing fuzzy inference systems from data: an interpretability-oriented review," Fuzzy Systems, IEEE Transactions on, vol. 9, no. 3, pp. 426-443, 2001.

[16] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," Neural Networks, IEEE Transactions on, vol. 17, no. 4, pp. 879 - 892, july 2006.

[17] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," Neural computation, vol. 7, no. 2, pp. 219-269, 1995.

[18] F. R. Bach, "Bolasso: model consistent lasso estimation through the bootstrap," in Proceedings of the 25th international conference on Machine learning. ACM, 2008, pp. 33-40.

[19] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in 2004 IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, july 2004, pp. 985 - 990 vol.2.

[20] T. Hastie, R. Tibshirani, and J. J. H. Friedman, The elements of statistical learning. Springer New York, 2001, vol. 1.

[21] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," The Annals of statistics, vol. 32, no. 2, pp. 407-499, 2004.

[22] K. P. Murphy, Machine learning: a probabilistic perspective. The MIT Press, 2012.

[23] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "Op-elm: optimally pruned extreme learning machine," Neural Net- works, IEEE Transactions on, vol. 21, no. 1, pp. 158-162, 2010.

[24] J. M. Martinez-Martiez, P. Escandell-Montero, E. Soria-Olivas, J. D. Martin-Guerrero, R. Magdalena-Benedito, and J. Gomez-Sanchis, "Regularized extreme learning machine for regression problems," Neurocomputing, vol. 74, no. 17, pp. 3716-3721, 2011.

[25] F. Montesino-Pouzols and A. Lendasse, "Evolving fuzzy optimally pruned extreme learning machine for regression problems," Evolving Systems, vol. 1, no. 1, pp. 43-58, August 2010.

[26] Asuncion, Arthur, and David Newman. "UCI machine learning repository." (2007).