2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

Generating Interpretable Mamdani-type fuzzy rules using a Neuro-Fuzzy System based on Radial Basis Functions

Diego G. Rodrigues and Gabriel Moura and Carlos M. C. Jacinto and
Paulo Jose de Freitas Filho and Mauro Roisenberg

***Abstract*— This paper presents a novel neuro-fuzzy inference system, called RBFuzzy, capable of knowledge extraction and generation of highly interpretable Mamdani-type fuzzy rules. RBFuzzy is a four layer neuro-fuzzy inference system that takes advantage of the functional behavior of Radial Basis Function (RBF) neurons and their relationship with fuzzy inference systems. Inputs are combined in the RBF neurons to compound the antecedents of fuzzy rules. The fuzzy rules consequents are determined by the third layer neurons where each neuron represents a Mamdani-type fuzzy output variable in the form of a linguistic term. The last layer weights each fuzzy rule and generates the crisp output. An extension of the ant-colony optimization (ACO) algorithm is used to adjust the weights of each rule in order to generate an accurate and interpretable fuzzy rule set. For benchmarking purposes some experiments with classic datasets were carried out to compare our proposal with the EFuNN neuro-fuzzy model. The RBFuzzy was also applied in a real world oil well-log database to model and forecast the Rate of Penetration (ROP) of a drill bit for a given offshore well drilling section. The obtained results show that our model can reach the same level of accuracy with fewer rules when compared to the EFuNN, which facilitates understanding the operation of the system by a human expert.**

I. Introduction

Classical machine learning techniques such as regression analysis and artificial neural networks (ANNs) suffer from the lack of explanation of the knowledge learned from the problem data, i.e., they don't explain the relationship between input and output variables in a natural language description. They also do not allow explicit knowledge elicited from an expert to be easily incorporated into the created model. Neuro-Fuzzy Systems (NFS) are used to overcome this kind of restrictions. NFSs combine the linguistic terms and human-like language description of fuzzy systems with the learning capability of neural networks. NFSs use a linguistic model to mathematically describe human reasoning. This linguistic model is composed of a set of IF-THEN fuzzy rules and fuzzy sets [33]. Having an interpretable model, a domain expert can read the model and fine-tune it, thus overcoming some limitations inherent to many machine learning techniques. The domain expert can also introduce new knowledge to further improve the model.

However, this approach introduces a problem known as the Accuracy-Interpretability tradeoff [34]. Interpretability refers to the the representation of the real system in an understandable way, while accuracy refers to the precision of the response given by the model. These concepts are contradictory in Fuzzy Systems and after a certain threshold they become almost mutually exclusive (e.g., a model with only one rule will be highly interpretable but will have no accuracy; a model with thousands of rules will be highly accurate but will be incomprehensible to a human reader). So as the complexity of the model increases, the quality of interpretability decreases and the quality of accuracy increases (See Figure 1). The problem then arises as to what is the best balance between accuracy and interpretability.

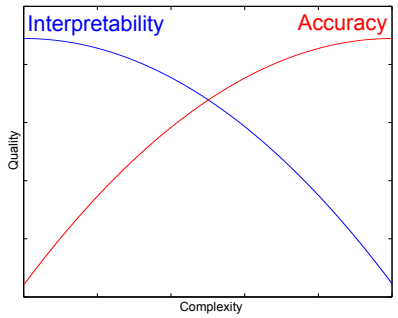

Fig. 1. Accuracy-Interpretability tradeoff

According to Wang [32] and Gabryel [12] NFS can be classified depending on the connections between the antecedents and the consequences in fuzzy rules as: Takagi-Sugeno neuro-fuzzy systems which are characterized by a functional dependence between antecedents and consequents [31]; and Mamdani-type neuro-fuzzy systems where antecedents and consequents in the rules are connected by a t-norm [26] [27].

Although Takagi-Sugeno type neuro-fuzzy systems are more accurate and automatic learning can be more easily implemented, the generated rules suffer from the lack of a model easily understandable by a human. Mamdani-type neuro-fuzzy systems are less accurate, but the generated rules are much more understandable [17].

In this paper we present a new neuro-fuzzy system called RBFuzzy that implements Mamdani-type fuzzy rules and seeks to provide the best cost-benefit ratio between accuracy and interpretability. It can be applied in applications where low quality of available data hinders the application of traditional machine learning techniques as it allows that additional explicit expert knowledge to be easily incorporated into the model. RBFuzzy is a four layer neural network that uses Radial Basis Function (RBF) neurons as a fuzzy rule

Diego G. Rodrigues and Paulo Jose de Freitas Filho and Mauro Roisenberg and Gabriel Moura are with the Department of Informatics and Statistics, Federal University of Santa Catarina, Brazil (email: {dgr, freitas, mauro}@inf.ufsc.br, gabriel.moura@posgrad.ufsc.br). Carlos M. C. Jacinto is with Petrobras Research Center, Rio de Janeiro, RJ, Brazil (email: cmcj@petrobras.com.br)

This work was supported by Petrobras and the Federal University of Santa Catarina (UFSC) as part of a research project.

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

base with gaussian membership functions so that the RBF neurons represent the fuzzy input membership functions. The regularization of RBF neurons is accomplished by a clustering algorithm. The other parameters of the system are determined by an improved Ant Colony Optimization Algorithm (ACO) designed for continuous domain variables called $ACO_{\mathbb{R}}V$, proposed in [10]. Just like other Evolutionary Methods, the $ACO_{\mathbb{R}}V$ uses a fitness function to measure the quality of the represented solution. So in order to find the most interpretable rules without losing accuracy, a fitness function is designed having two objectives: minimize the error and minimize the complexity. This way the learning algorithm discards high error and high complexity solutions where the complexity is essentially related to the amount of rules needed to model the input-output mapping present in the training data.

This paper is organized as follows. In Section II interpretability and accuracy are defined. Section III describes two classic neuro-fuzzy systems. Section IV describes the proposed method, namely, RBFuzzy, including the architecture and learning algorithm. Section V presents the experiments, using simulated and real data, and the analysis of the results. Finally, in Section VI conclusions and suggestions for further research are presented.

II. Interpretability and Accuracy

Accuracy in a prediction model can be defined as the capability of the model to precisely represent a real world system [7]. This capability can be measured in the percentage of correctly classified patterns by a classification model or the mean square error(MSE) in a regression model. Usually the main focus when making a prediction model is to get the highest possible accuracy.

Interpretability is considered as the main advantage of fuzzy systems over other prediction models [28], such as neural networks. Interpretability is the ability of the model to represent a real world system in an understandable way [7]. The more comprehensive the model is to a human reader the more interpretable it is. Interpretability is not as easily measured as accuracy and often there is some controversy when choosing the optimal way to measure it ([15], [35], [2], [13]). The measure used in this paper to evaluate interpretability is based on the number of rules of the model. Regarding the number of rules the optimal model would be the model with the least number of rules in which the accuracy is in a satisfactory level.

Fuzzy systems when constructed from expert knowledge present a well understandable model with satisfactory accuracy. But when the fuzzy system is extracted from data, most methods focus on improving the accuracy [16]. In order to provide an accurate and at the same time understandable model this paper proposes an approach using a Neuro-Fuzzy System using an optimization algorithm with a fitness function designed to satisfy these two constraints, which is described in the next sections.

III. Background

In literature several neuro-fuzzy systems have been studied and developed (see [21], [18]). They offer the linguistic knowledge description of fuzzy systems combined with the learning capability of neural networks. In this section two classic NFSs are described.

A. ANFIS

The Adaptive Neural Fuzzy Inference System (ANFIS) was introduced by Jang in [18]. The ANFIS consists of a Generalized Neural Network (GNN), which is a feed-forward network with a gradient-descent-based learning procedure, that is used to build a Takagi-Sugeno Fuzzy Inference System.

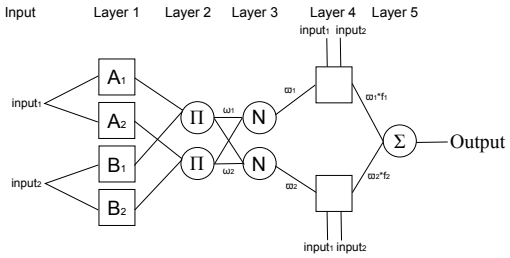

Fig. 2. ANFIS Architecture

Five layers are used in the ANFIS, as depicted in Figure 2. Each one of these layers has several nodes described by a function. There are two types of nodes: adaptive nodes and fixed nodes, the adaptive nodes can be added and modified, while the fixed nodes only have the connection weights changed.

The first layer has adaptive nodes, describing fuzzy membership functions for each input. The second layer has product nodes, the output of the product nodes is the product of all incoming signals. The third layer is a normalized layer, its output is the sum of all incoming signals weighted by ϖ_i which is then normalized. The fourth layer represents the consequent part of the rules, where each node is an adaptive node, its output is the result of a function that takes as parameter the incoming signal from the third layer and the inputs. The fifth layer computes the sum of all incoming signals weighted by $\varpi_i * f_i$.

Despite being the first neuro-fuzzy system found in literature, recent research attribute good results to the ANFIS ([29], [8]). The main disadvantage of the ANFIS is the fact that it uses the Takagi-Sugeno FIS ([18]). Takagi-Sugeno FIS have lower interpretability when compared to Mamdani FIS ([20], [19]).

B. EFuNN

Evolving Connectionist Systems (ECoS) is a family of artificial neural networks introduced by Kasabov in [21]. Evolving in the context of ECoS is not related to evolution in the context of genetic algorithms, but refers to the fact that these systems change over time, the structure of a ECoS network is dynamic. Figure 3 shows the basic architecture of an ECoS.

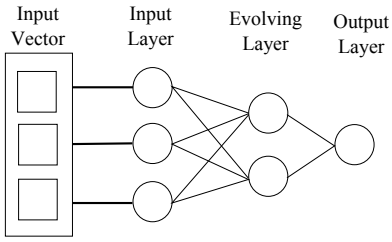

Fig. 3. ECoS Basic Architecture

Evolving Fuzzy Neural Networks (EFuNN) is a special application of ECoS and was introduced in [21]. It is a five-layer feed-forward neural network. The first neuron layer is the input layer. The second layer is the condition layer. Each neuron of the condition layer represents a fuzzy membership function (MF) for an input. The third layer of neurons is the evolving layer, where the fuzzy rules are added and stored. The fourth layer is the action layer, the neurons in this layer represent fuzzy membership functions attached to the output neurons. The fifth layer is the output layer, where a crisp output value is calculated from the fuzzy output value from the fourth layer by using a defuzzification method.

Mamdani fuzzy rules can be extracted from a EFuNN using a specific algorithm as described in [23].

Good results showing fast performance and low error have been attributed to the EFuNN (see [9], [1]). However, disadvantages attributed to the EFuNN is the fact that some variables, such as number and type of membership functions for each output variable, are automatically calculated and can't be changed and optimal learning parameters are difficult to be configured (see [30]).

IV. Proposed Method

The proposed fuzzy inference system, called RBFuzzy, is a neuro-fuzzy system that implements Mamdani-like fuzzy rules. Similar to the algorithm proposed by Li ([24]), which gives a fuzzy interpretation to RBF neural networks, the RBFuzzy also uses neurons activated by radial basis functions and a clustering algorithm, but different from that proposal the RBFuzzy uses neurons that represent linguistic terms for the output variable.

RBFuzzy can be trained using a hybrid learning algorithm which occurs in two steps. In the first stage, a clustering algorithm determines the membership functions for the input variables based in the data distribution on the input space. In the second step an optimization algorithm, in this case the $ACO_{\mathbb{R}}V$, determines the weights of the network. Through the use of the $ACO_{\mathbb{R}}V$ it is possible to incorporate more than one objective in the learning algorithm thus optimizing at the same time accuracy and interpretability.

The next subsections are organized as follows. Subsection IV-A describes the architecture of the RBFuzzy. Subsection IV-B describes how the learning algorithm of the RBFuzzy works. Subsection IV-C explains how to convert the trained network to the form of linguistic fuzzy rules.

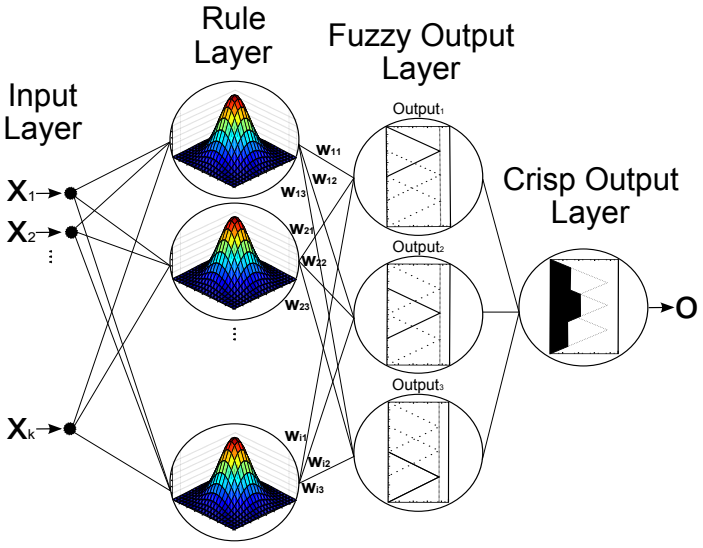

Fig. 4. RBFuzzy Architecture

A. Architecture

The architecture is depicted in Figure 4. It consists of a 4-Layer Neural Network. This Neural Network is basically a representation of a Fuzzy Inference System.

The first layer is the Input Layer, where the data is presented to the network and which represents input variables.

The second layer is called the Rule Layer. Each neuron of the Rule Layer represents a set of fuzzy membership functions and is generated by the result of a clustering algorithm on the training data set. By using a clustering algorithm, each neuron will represent a cluster in the input space. The data distribution of each cluster is used to define the parameters of the activation function for each input variable. The activation function of the Rule Layer is a Radial Basis Function (RBF). For each input variable a symmetric Gaussian Function is calculated and then the activation of a given neuron i is calculated as being the minimum of all input gaussians as shown in equation 1:

$$f(x,c,s) = exp\left(-\frac{(x-c)^2}{2*s^2}\right)$$
$$g_i = min(f(x_1,c_1,s_1),\ldots,f(x_k,c_k,s_k)) \quad (1)$$

where x_k represents an input variable, c_k and s_k represent the center and spread value of the membership function respectively. For each cluster, c_k is defined as being equal to the mean of the cluster's data and s_k is the standard deviation of the cluster's data.

The number of neurons in the Rule Layer is a user chosen parameter to control the number of clusters in the input space. A higher number of neurons will result in a higher number of rules, since the number of neurons will determine the number of membership functions for each input. In each neuron of the Rule layer the input variables are combined to form the antecedent part of the fuzzy rules as depicted in Figure 6. For example, suppose that a given neuron of the Rule layer has two input variables with parameters (c_1, s_1) for Input_1 and

(c_2, s_2) for Input$_2$ as can be seen in the red cluster of Figure 6. In this case the antecedent of the fuzzy rules represented by this neuron would be: *"IF Input$_1$ IS IN gaussmf(c1, s1) AND Input$_2$ IS IN gaussmf(c2, s2) THEN..."*

When an input vector is submitted to the network, the output activation of the neurons from the Rule Layer describe the degree of membership of the input vector to each rule represented by the neurons. The connections w_{ij} from the Rule Layer to the Fuzzy Output Layer represent the weight of the rules for each fuzzy output, which are adjusted by the ACO$_{\mathbb{R}}$V. This step is covered in detail in section IV-B.

The third layer is the Fuzzy Output Layer. The number of neurons in this layer is a user chosen parameter and is equal to the number of the desired output fuzzy sets, e.g., by defining 3 as the number of neurons in this layer, 3 output fuzzy sets are generated and are linguistically interpreted as: LOW, MEDIUM and HIGH. The membership functions employed are triangular functions.

The output of each neuron j in the Fuzzy Output Layer is the degree of membership of the input to the output fuzzy variables weighted by w_{ij}. As a result we have the following activation function for the Fuzzy Output Layer, defined by equation 2:

$$z_j = max(w_{1j} * g_1, w_{2j} * g_2, \ldots, w_{ij} * g_i), \qquad (2)$$

Each neuron in the Rule Layer is connected to all neurons of the Fuzzy Output Layer. Thus, the total number of rules generated by RBFuzzy model is $i * j$, where i is the number of neurons in the Rule Layer and j is the number of neurons in the Output Layer. Observe that for each neuron of the Rule Layer, j rules are created, each one with the same antecedent part and with a different consequent part. Each one of these rules will have a different weight determined by the learning algorithm. Note that rules with low weights are cut from the final set of rules and tests show that this doesn't significantly affect the accuracy of the model. In the following example we show the rules generated when the neuron of the Rule Layer represented by the red region of Figure 6 is connected to the neurons of the Fuzzy Output Layer:

Rule 1 = **IF** Input$_1$ IS IN gaussmf(c$_1$, s$_1$) AND **IF** Input$_2$ IS IN gaussmf(c$_2$, s$_2$) **THEN** OUTPUT IS HIGH WITH w = (w$_{11}$)

Rule 2 = **IF** Input$_1$ IS IN gaussmf(c$_1$, s$_1$) AND **IF** Input$_2$ IS IN gaussmf(c$_2$, s$_2$) **THEN** OUTPUT IS MEDIUM WITH w = (w$_{12}$)

Rule 3 = **IF** Input$_1$ IS IN gaussmf(c$_1$, s$_1$) AND **IF** Input$_2$ IS IN gaussmf(c$_2$, s$_2$) **THEN** OUTPUT IS LOW WITH w = (w$_{13}$)

The fourth and final layer is the Crisp Output Layer, which computes the numeric value of the network output. In this paper, this value is computed by applying the centroid defuzzification method to the outputs of the Fuzzy Output Layer, but other defuzzification methods can be applied.

B. Learning Algorithm

Figure 5 depicts the basic process of the learning algorithm that RBFuzzy implements. In the first step the number of RBF neurons should be chosen by the user. The algorithm could also choose the optimal number of RBF neurons using a specialized clustering algorithm, the ECM for example [22]. In the second step the number of output membership functions is chosen, typically between 3 and 5 output membership functions give good results. Then the data is divided using a cluster algorithm. In our tests k-means and fuzzy c-means yielded good results. Then the data are statistically analyzed to define the parameters of the Radial Basis Functions, in this step the centers and spread values of the input membership functions are defined for each cluster separately, Figure 6 shows the result of this stage. In the final step the ACO$_{\mathbb{R}}$V is used to minimize the error and minimize the total weight of the rules by using an appropriate fitness function, which is shown in equation 3. The weights of the rules are minimized because rules whose weights are so small that they have little influence on the output are discarded in the cut off step, thus making the model more interpretable. This whole process is explained in more depth below.

1: $N\_RULES \leftarrow$ Choose the number of rules
2: $N\_OUT \leftarrow$ Choose the number of output membership functions
3: $CLUSTERS \leftarrow$ Use a cluster algorithm to cluster the training data into N_RULES clusters
4: **for all** cluster$_i$ in CLUSTERS **do**
5: Define the RBF center parameter of the rule$_i$ equals to cluster$_i$ center.
6: Define the RBF spread parameter of the rule$_i$ equals to cluster$_i$ standard deviation
7: **end for**
8: Use the ACO$_{\mathbb{R}}$V to find the optimal weights for the network
9: Cut off the rules below the user defined output weight threshold

Fig. 5. RBFuzzy Learning Algorithm Overview

The learning algorithm can be divided into two phases, the first being responsible for defining the input space covered by each rule and the second phase being responsible for finding the optimal weights of the network and cutting unnecessary rules.

In the first phase the training data is divided in i regions in the input space, being i equal to the number of RBF neurons in the Rule Layer. In order to do this a clustering algorithm is used to partition the training data into i clusters. The activation function for the neurons of the Rule Layer is defined by equation 1. Figure 6 shows the result of this process, where each cluster is represented by a different

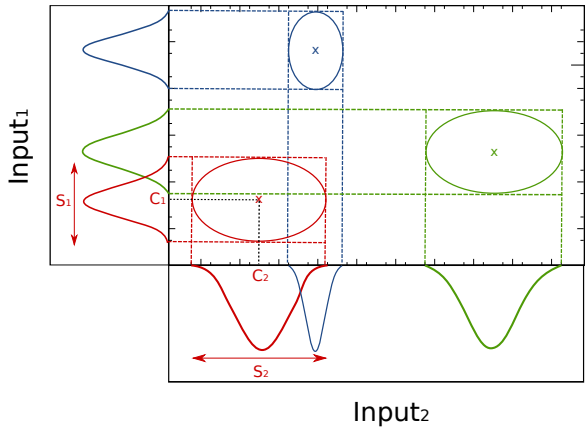

Fig. 6. Input Membership Functions

color.

In the second phase the ACO$_{\mathbb{R}}$V algorithm is used to find the weights between the Rule Layer and the Fuzzy Output Layer. The idea is to simultaneously minimize the network error and the network weights. The fitness function optimized by the ACO$_{\mathbb{R}}$V is defined by equation 3

$$fit = \alpha \sqrt{\frac{1}{N}\sum_{i=1}^{N}(o_i - t_i)^2} + \lambda \sum_{i=1}^{m}\sum_{j=1}^{n}(w_{ij}) \qquad (3)$$

where α and λ are respectively the weights for accuracy and interpretability, N is the total number of training data points, o_i is the output value for the data point i, t_i is the correct value for the data point i, and w_{ij} is the weight between the Rule Layer neuron i to the Fuzzy Output Layer neuron j. The definition of the complexity should be modeled based on the interpretability measure chosen. In this paper the measure used to evaluate interpretability is the number of rules, so our complexity function is the weighted sum of all rule weights. By adopting this definition the ACO$_{\mathbb{R}}$V algorithm will favor models with lower weights. Since rules with low weights have almost no influence on the output, all rules with low weights will be discarded from the model in the Rule Cutting step. This ensures that a model with lower weights will result in a fuzzy inference system with higher interpretability.

Thus the fitness function is responsible for minimizing the error and maximizing the interpretability of the rules. By changing the weights α and λ the precision and interpretability can be adjusted to better suit different problems. If interpretability is an issue λ could be raised and more interpretable rules will be generated.

After the ACO$_{\mathbb{R}}$V algorithm has found the optimal weights the Rule Cutting step takes place. The Rule Cutting is the final step of the learning algorithm, where every rule with the output weight below a user defined threshold is discarded from the model. Since the fitness function used favors models with lower weights, this step is expected to cut off a good number of rules without significant degradation in the accuracy of the neuro-fuzzy model.

C. Fuzzy Interpretation

After the training phase, the network can be interpreted as a Mamdani Fuzzy Inference System (FIS), where the RBFs of the Rule Layer represent the antecedent membership functions and the Fuzzy Output Layer neurons represent the consequent membership functions for each rule. The crisp output can be obtained by the defuzzification of the consequent. The RBFuzzy extracted rules are defined in the form:

$$\begin{cases} \textbf{IF } x_1 \text{ IS IN gaussmf}(c_1, s_1) \text{ AND} \\ \textbf{IF } x_2 \text{ IS IN gaussmf}(c_2, s_2) \text{ AND} \\ \vdots \\ \textbf{IF } x_k \text{ IS IN gaussmf}(c_k, s_k) \\ \textbf{THEN OUTPUT IS Output}_j \text{ with w = } (w_{ij}) \end{cases}$$

being k the number of input variables, x_k an input variable, c_k and s_k being the RBF center and spread for the respective input variable, $gaussmf$ the gaussian membership function, j the number of fuzzy output sets or number of neurons in the Fuzzy Output Layer and w_{ij} being the connection weight from the neuron i of the Rule Layer to the neuron j of the Fuzzy Output Layer, found by the ACO$_{\mathbb{R}}$V algorithm.

V. Experiments and Results

In this section the results are presented. The experiments used simulated data from the Mackey-Glass Time-Series [25] and the Engine dataset [3], where RBFuzzy was tested against EFuNN. An experiment using real world data from an oil well-log database was also performed.

A. Mackey-Glass Time-Series Prediction

In order to evaluate the performance of our method, we have performed experiments using the Mackey-Glass (MG) chaotic time-series function [25]. This time-series is defined by the differential equation 4:

$$\frac{\partial x}{\partial t} = \beta \frac{x_\tau}{1 + x_\tau^n} - \gamma x(t), \quad \gamma, \beta, n > 0, \qquad (4)$$

where β, γ, τ, n are real numbers, and x_τ represents the value of the variable x at time $(t-\tau)$. The experiments were carried out using the MG data set with 1000 data points. The exact same data sets were used for both models.

Prediction quality has been evaluated using the non-dimensional error index (NDEI). NDEI is defined as the RMSE divided by the standard deviation of the target series.

Experiment results are summarized in Table I and Figures 7 and 8 show the graphical results for the EFuNN and RBFuzzy respectively. Analyzing the graphical results we can conclude that both models offered good approximations. The RBFuzzy had difficulty to predict the extremes of the series. We believe that this is due to the values found by the clustering algorithm for the centers and spreads of the gaussian membership functions. Some adjustments in the clustering algorithm could mitigate this problem.

EFuNN achieved a NDEI of 0.4481, but failed to provide a good number of rules, generating 53 fuzzy rules while the

NFS	# of Rules	NDEI
RBFuzzy	7	0.3684
EFuNN	53	0.4481

TABLE I
MACKEY-GLASS DATASET RESULTS

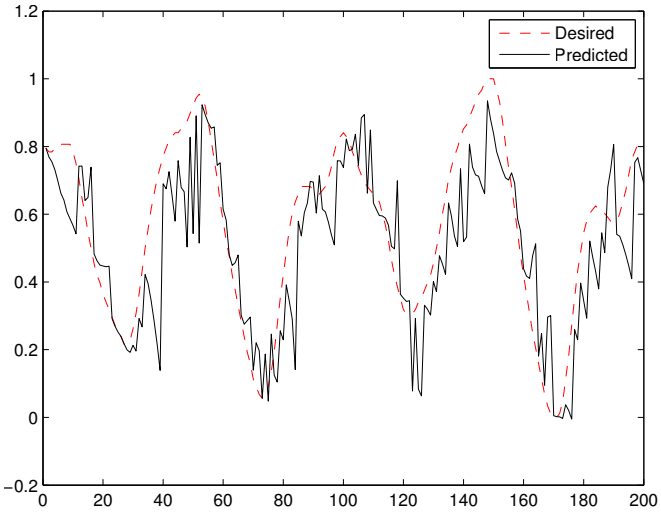

Fig. 7. EFuNN Mackey-Glass Dataset Prediction

RBFuzzy generated only 7 fuzzy rules and a NDEI of 0.3684. RBFuzzy achieved better interpretability due to the lower number of rules when compared to the EFuNN. The fitness function that was used in this experiment had a weight of 0.8 to improve accuracy and 0.2 to the interpretability term.

Figure 10 shows the rules generated by the RBFuzzy using Matlab's Fuzzy Logic Toolbox. For comparison reasons, Figure 9 shows the rules generated by the RBFuzzy before the Rule Cutting is applied, a set with 36 rules is reduced to 7 rules after the Rule Cutting. The Rule Cutting step used a threshold of 0.1 and after the Rule Cutting step the error percentage showed a difference of only 1.2%. This shows that our fitness function improves the interpretability of the resulting fuzzy inference system without compromising accuracy.

B. Engine Dataset Prediction

The next experiments were performed using the Engine dataset [3] with 1199 data points, having the Fuel rate as the input to predict the Nitrous oxide emissions.

The RBFuzzy generated 6 fuzzy rules and a NDEI of 0.4225. The fitness function that was used in this experiment had a weight of 0.8 to improve accuracy and 0.2 to improve interpretability.

The graphical result, shown in Figure 11, shows that the RBFuzzy gives a good approximation of the desired function using 6 fuzzy rules.

C. Application in offshore oil drilling

In order to reduce costs in offshore oil drilling operations the time required to successfully drill a well has to be

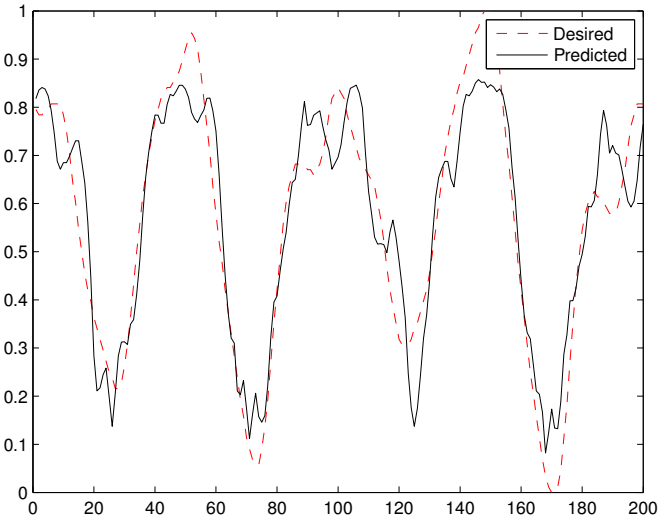

Fig. 8. RBFUZZY Mackey-Glass Dataset Prediction

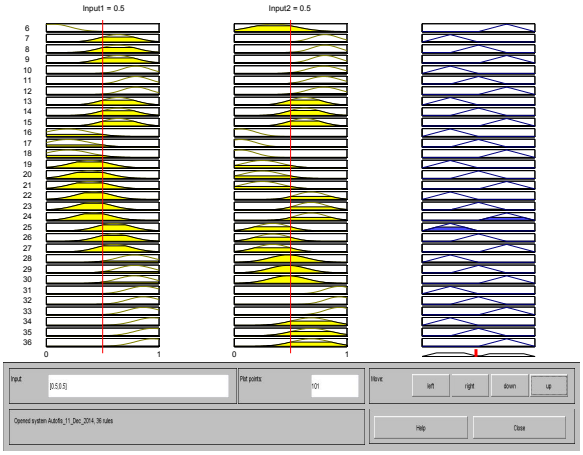

Fig. 9. RBFuzzy Mackey-Glass Rule Set Before the Rule Cutting

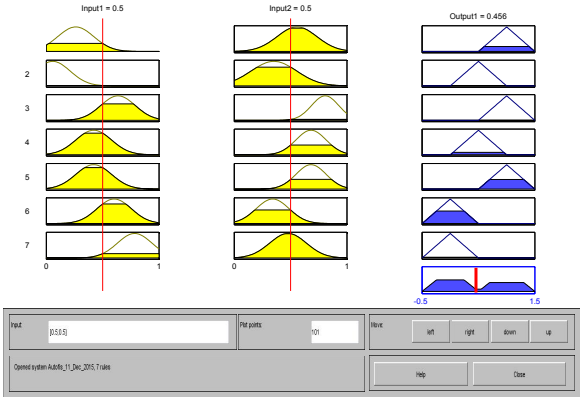

Fig. 10. RBFuzzy Mackey-Glass Rule Set After the Rule Cutting

estimated fairly precisely, since most of the costs associated are tied to the rental of equipment required for the operation [14]. However, each operation has unique properties that make this task highly difficult. Many properties vary, such as rock type, rock porosity, gas presence, pressure, drill bit wear rate among others. All these properties affect the Rate

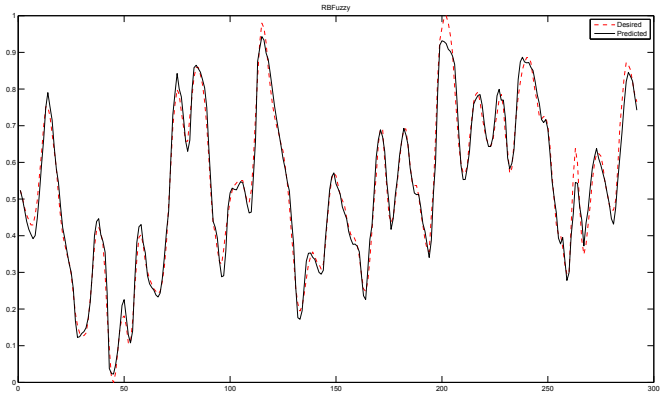

Fig. 11. RBFUZZY Engine Dataset Prediction

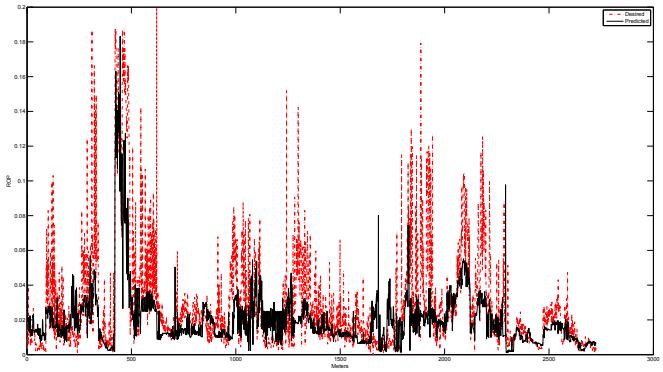

Fig. 12. RBFUZZY Training Set for ROP prediction

Of Penetration (ROP), that is the speed at which a drill bit breaks the rock in order to deepen the petroleum borehole [6]. There are other parameters that also affect the ROP and that can be controlled by a drilling operator [5]: weight on bit (WOB), revolutions per minute (RPM), bit type, bit diameter and drilling fluid pressure.

Most of the work in the planning phase of a drilling operation is restricted to adjusting the bit type, RPM and WOB in order to achieve an acceptable ROP. To optimize this work many systems using artificial neural networks (ANN) were proposed in the past [5] and even choose automatically some parameters such as RPM and WOB [11].

These models were tested against the available data but they did not achieve good results, mostly because the available data is of poor quality. The available data is from the pre-salt layer, where target depths are greater than 5000 meters, crossing salt layers that can reach 2000 meters in thickness [4]. Because of those extreme conditions, the poor quality is an inherent and unavoidable characteristic of the data.

The RBFuzzy is used to overcome this problem by providing the linguistic interpretation of the forecast model to the oil drilling specialist. The specialist reads the model and corrects it where mistakes were inserted by bad parts of the data. If there are input variables missing from the data the specialist can insert them using his knowledge. After the model is corrected by the specialist it is saved in a database where it is accessible for the rest of the company.

The fitness function that was used in this experiment had a weight of 0.7 to improve accuracy and 0.3 to improve interpretability. Figure 12 shows the graphical result for the generated model. Using 35 rules the RBFuzzy achieved a RMSE of 0.0319.

VI. Conclusion

In this paper we have proposed the RBFuzzy, a new NFS used to extract Mandami fuzzy rules from data in order to perform time series forecasting. Using radial basis functions as activation for the neurons in the middle layer and an optimization algorithm in the training of the network, we showed that it is possible to extract knowledge from data in the form of fuzzy rules with a high level of accuracy and interpretability.

The proposed neuro-fuzzy inference system uses an optimization algorithm to improve the extraction of knowledge from data in the form of fuzzy if-then rules. This method has a lot of advantages, its multi-objective nature makes it possible to insert new constraints to knowledge extraction process. This can be done by modifying the fitness function and adding new objectives.

Experimental results show that our proposed model can reach a good level of accuracy with better interpretability when compared to EFuNN.

References

[1] Almomani, A., Wan, T.C., Altaher, A., Manasrah, A., Almomani, E., Anbar, M., Alomari, E., Ramadass, S.: Evolving fuzzy neural network for phishing emails detection. Journal of Computer Science 8(7), 1099 (2012)

[2] Alonso, J.M., Magdalena, L., González-Rodríguez, G.: Looking for a good fuzzy system interpretability index: An experimental approach. International Journal of Approximate Reasoning 51(1), 115–134 (2009)

[3] Beale, M., Hagan, M.T., Demuth, H.B.: Neural network toolbox. Neural Network Toolbox, The Math Works pp. 5–25 (1992)

[4] Beltrao, R.L., Sombra, C., Lage, A., Fagundes Netto, J., Henriques, C.: Ss: Pre-salt santos basin-challenges and new technologies for the development of the pre-salt cluster, santos basin, brazil. In: Offshore Technology Conference (2009)

[5] Bilgesu, H., Tetrick, L., Altmis, U., Mohaghegh, S., Ameri, S.: A New Approach for the Prediction of Rate of Penetration (ROP) Values. In: SPE Eastern Regional Meeting. Society of Petroleum Engineers, Lexington, Kentucky (Oct 1997)

[6] Bourgoyne Jr., A., Young Jr., F.: A Multiple Regression Approach to Optimal Drilling and Abnormal Pressure Detection. SPE Journal 14(4), 371–384 (1974)

[7] Casillas, J., Cordon, O., Herrera, F., Magdalena, L.: Interpretability improvements to find the bal-

ance interpretability-accuracy in fuzzy modeling: an overview. Springer (2003)
[8] Chang, F.J., Wang, K.W.: A systematical water allocation scheme for drought mitigation. Journal of Hydrology 507(0), 124 – 133 (2013)
[9] Chang, P.C., Fan, C.Y., Lin, J.J.: Monthly electricity demand forecasting based on a weighted evolving fuzzy neural network approach. International Journal of Electrical Power & Energy Systems 33(1), 17–27 (2011)
[10] Conti, C.R., Roisenberg, M., Neto, G.S.: Aco$_{\mathbb{R}}$v-an algorithm that incorporates the visibility heuristic to the aco in continuous domain. In: Evolutionary Computation (CEC), 2012 IEEE Congress on. pp. 1–8. IEEE (2012)
[11] Fonseca, T.C., Mendes, J.R.P., Serapião, A.B.S., Guilherme, I.R.: A Genetic Neuro-Model Reference Adaptive Controller for Petroleum Wells Drilling Operations. In: International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce. p. 3 (2006)
[12] Gabryel, M., Rutkowski, L.: Evolutionary learning of mamdani-type neuro-fuzzy systems. In: Artificial Intelligence and Soft Computing–ICAISC 2006, pp. 354–359. Springer (2006)
[13] Gacto, M.J., Alcalá, R., Herrera, F.: Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. Information Sciences 181(20), 4340–4360 (2011)
[14] Gandelman, R.A.: Predição da ROP e otimização em tempo real de parâmetros operacionais na perfuração de poços de petróleo offshore. Ph.D. thesis, UFRJ (2012)
[15] Guillaume, S.: Designing fuzzy inference systems from data: an interpretability-oriented review. Fuzzy Systems, IEEE Transactions on 9(3), 426–443 (2001)
[16] Guillaume, S., Magdalena, L.: Expert guided integration of induced knowledge into a fuzzy knowledge base. Soft computing 10(9), 773–784 (2006)
[17] Hsu, C.F., Lin, C.M., Yeh, R.G.: Supervisory adaptive dynamic rbf-based neural-fuzzy control system design for unknown nonlinear systems. Applied Soft Computing 13(4), 1620–1626 (2013)
[18] Jang, J.S.R.: Anfis: adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man and Cybernetics 23(3), 665–685 (1993)
[19] Jassbi, J., Alavi, S., Serra, P.J., Ribeiro, R.A.: Transformation of a mamdani fis to first order sugeno fis. In: Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International. pp. 1–6. IEEE (2007)
[20] Jassbi, J., Serra, P., Ribeiro, R., Donati, A.: A comparison of mandani and sugeno inference systems for a space fault detection application. In: Automation Congress, 2006. WAC'06. World. pp. 1–8. IEEE (2006)
[21] Kasabov, N.K.: The ECOS Framework and the ECO Learning Method for Evolving Connectionist Systems. Journal of Advanced Computational Intelligence and Intelligent Informatics 2, 195–202 (1998)
[22] Kasabov, N.K., Song, Q.: Denfis: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Transactions on Fuzzy Systems 10(2), 144–154 (2002)
[23] Kasabov, N., Woodford, B.: Rule insertion and rule extraction from evolving fuzzy neural networks: algorithms and applications for building adaptive, intelligent expert systems. In: Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE'99. 1999 IEEE International. vol. 3, pp. 1406–1411. IEEE (1999)
[24] Li, W., Hori, Y.: An algorithm for extracting fuzzy rules based on rbf neural network. Industrial Electronics, IEEE Transactions on 53(4), 1269–1276 (2006)
[25] Mackey, M.C., Glass, L., et al.: Oscillation and chaos in physiological control systems. Science 197(4300), 287–289 (1977)
[26] Mamdani, E.H.: Application of fuzzy algorithms for control of simple dynamic plant. Electrical Engineers, Proceedings of the Institution of 121(12), 1585–1588 (1974)
[27] Mamdani, E.H.: Application of fuzzy logic to approximate reasoning using linguistic synthesis. Computers, IEEE Transactions on 100(12), 1182–1191 (1977)
[28] Mikut, R., Jäkel, J., Gröll, L.: Interpretability issues in data-based learning of fuzzy systems. Fuzzy Sets and Systems 150(2), 179–197 (2005)
[29] Ocak, H., Ertunc, H.M.: Prediction of fetal state from the cardiotocogram recordings using adaptive neuro-fuzzy inference systems. Neural Computing and Applications pp. 1–7 (2012)
[30] Petrovic-Lazarevic, S., Zhang, J.Y.: Neuro-fuzzy models and tobacco control. In: Proceedings of the European Computing Conference. pp. 25–31. Springer (2009)
[31] Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. Systems, Man and Cybernetics, IEEE Transactions on (1), 116–132 (1985)
[32] Wang, J.S., Lee, C.G.: Self-adaptive neuro-fuzzy inference systems for classification applications. Fuzzy Systems, IEEE Transactions on 10(6), 790–802 (2002)
[33] Zadeh, L.A.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)
[34] Zadeh, L.A.: Outline of a new approach to the analysis of complex systems and decision processes. Systems, Man and Cybernetics, IEEE Transactions on (1), 28–44 (1973)
[35] Zhou, S.M., Gan, J.Q.: Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling. Fuzzy Sets and Systems 159(23), 3091–3131 (2008)