Extreme Learning for Evolving Hybrid Neural Networks

Fernando Bordignon and Fernando Gomide
School of Electrical and Computer Engineering
University of Campinas
Campinas, SP, Brazil
E-mail: {bordi,gomide}@dca.fee.unicamp.br

Abstract—This paper addresses a structure and introduces an evolving learning approach to train uninorm-based hybrid neural networks using extreme learning concepts. Evolving systems are high level adaptive systems able to simultaneously modify their structures and parameters from a stream of data, online. Learning from data streams is a contemporary and challenging issue due to the increasing rate of the size and temporal availability of data, turning traditional learning methods impracticable. Uninorm-based neurons, rooted in triangular norms and conorms, generalize fuzzy neurons. Uninorms bring flexibility and generality to fuzzy neuron models as they can behave like triangular norms, triangular conorms, or in between by adjusting identity elements. This feature adds a form of plasticity in neural network modeling. An online clustering method is used to granulate the input space, and a scheme based on extreme learning is developed to train the neural network. Computational results show that the learning approach is competitive when compared with alternative evolving modeling methods.

Keywords—hybrid neural networks; unineurons; evolving systems; online learning; extreme learning.

I. INTRODUCTION

Machine learning methods are being reevaluated over the last years as the need of online capabilities is made evident by the massive growth in numbers and accessibility of computing devices [1] [2]. Computers are connected in a unprecedented level making available a massive amount of data, generating interest to understand and forecast key economic, technical, and social variables.

With this context in mind new methods have been conceived during recent years. A new class of machine learning approach with adaptive abilities to simultaneously learn a system structure and its parameters emerged, namely, the evolving system approach. The term evolving means online, gradual systems development and adaptation. Evolving systems are an alternative and innovative way of adapt, learn and represent knowledge about changing environments [1].

Fuzzy set theory provides solid methods and efficient mechanisms to develop evolving systems as reported in early works on fuzzy rule-based modeling [3] and neural fuzzy inference DENFIS models [4]. More recently, more general evolving connectionist systems (ECOS) framework [5], flexible fuzzy inference systems (FLEXFIS) [6], evolving Takagi-Sugeno (eTS) modeling [7] and its improved versions [8] [9] have been developed. Current research focuses on granular computing and modeling extensions such as interval and fuzzy set-based evolving modeling (FBeM) systems [2].

From the neural networks perspective, extensions of classic neuron model using triangular norms and co-norms (t-norms and t-conorms) have been extensively studied in the literature lately e.g. [10], [11], [12], [13] and [14]. In particular, the notion of uninorm has been explored to generalize further both, the basic fuzzy neuron model, and neural architectures through hybridizations. Uninorms are interesting not only because they add flexibility in neuron and neural network design, but also because they naturally provide a way to add neuronal plasticity through a single parameter called the identity element. This is also very useful for evolving systems and neural structures.

In this paper we address a hybrid neural fuzzy network whose structure has two main parts: a fuzzy neural system and a neural network. The multilayer structure of the network has membership functions in the input layer neurons, uninorm-based neurons in the second layer, and classic neurons in the third layer. Uninorms provide flexibility at the cost of an extra parameter (identity element) to learn for each unineuron. Interesting, however, the identity element can be either chosen by the designer if he has prior knowledge or needs a particular structure, or leave it to be learnt using data. Due to its inherent highly nonlinear nature, learning complexity can be reduced using extreme learning [15]. Basically, extreme learning consists in randomly assigning weights for the hidden layer of feed-forward neural networks and analytically determine the output weights. Extreme learning has shown to provide fast training, accurate results and good generalization performance [16].

The input layer of the hybrid neural fuzzy network addressed in this paper uses Gaussian membership functions to granulate the input space. Centers of the membership functions are determined online via recursive clustering methods. The hidden layer unineurons are compositions of uninorms at local synaptic processing, with max t-conorm at global aggregation level. A typical neural network layer with sigmoid activation function forms the output layer.

After this brief introduction, the paper proceeds as follows. Section II gives detailed description of the learning steps including the evolving fuzzy neural network system approach. Next, computational results of the tests conducted...
using well known benchmarks are reported in Section III. Generalization issues encountered in evolving and non-evolving approaches is discussed in Section IV. Section V concludes the paper summarizing its contributions and suggesting issues for future investigation.

II. EVOLVING EXTREME LEARNING UNINETWORK

In this section we present an evolving hybrid fuzzy neural network based on uninorms. Learning involves recursive clustering to granulate the input space and extreme learning algorithm to simultaneously adjust the network weights and parameters. For short, we call the evolving hybrid fuzzy neural network model eXUninet (evolving eXtreme learning Uninetwork) in the rest of the paper.

A. Uninorms and the Unineuron

A generalization of t-norms and t-conorms called uninorm was introduced in [17]. Formally, a uninorm is a mapping $u : [0,1] \times [0,1] \rightarrow [0,1]$ that satisfies the following properties:

1) Commutativity: $a \ t b = b \ t a$
2) Associativity: $a \ u (b \ u c) = (a \ u b) \ u c$
3) Monotonicity: if $b \leq c$, then $a \ u b \leq a \ u c$
4) Identity element: $a \ u e = a$, $\forall a \in [0,1]$

While t-norms respect the first three conditions with the identity element $e = 1$, t-conorms satisfies the first three as well but with identity element $e = 0$. Uninorms extends triangular norms by allowing the identity element to be in the unity interval.

Fuzzy neurons were introduced as basic units of fuzzy neural networks in [18]. The two basic neuron models, called or and and neurons, are constructed using s-norms and t-norms as follows:

$$o_r(a, w) = S_{i=1}^n (a_i \ t w_i) \quad (1)$$
$$a_n(a, w) = T_{i=1}^n (a_i \ s w_i) \quad (2)$$

Here $T$ is a t-norm, $S$ is a s-norm, $a_i \in [0,1]$ are the inputs, and $w_i \in [0,1]$ the synaptic weights. Neural networks based on these neurons were successfully used in various applications such as thermal modeling of power transformers [19] and time series prediction [20]. Recent work suggested uninorm-based neurons (unineurons) [10] to explore uninorms in modeling distinct neuron operators, a desired characteristic to add flexibility in fuzzy neural networks. Further improvements of the unineuron model was reported in [12]. The uninorm used in the eXUninet is of the type $O_{r parcel},$ which consists of uninorms at synaptic processing level and a s-norm at global aggregation level, as shown in (3).

$$O_{r parcel} = S_{i=1}^n (a_i \ u w_i) \quad (3)$$

The uninorm realization used in this paper adopts the following construct [17]

$$u(a, b) = \begin{cases} a \ s b, & \text{if } a, b \in [e, 1] \\ a \ t b, & \text{otherwise} \end{cases} \quad (4)$$

with $t$ a t-norm and $s$ a s-norm.

B. Topology of the Network

The structure of eXUninet was first studied in [11]. The network has two major parts, fuzzy inference systems and aggregation neural network, assembled into three layers. The fuzzy neural system is composed by the first two layers, the input and hidden layer, respectively. The input layer consists of neurons whose activation functions are membership functions of fuzzy sets that granulate the input space to form a fuzzy partition. Here we use Gaussian membership functions centered at $c_l$ with radius $\sigma$. The membership degree of input $x_i$ is computed using (5). For each dimension $x_i$ of a $n$-dimensional input vector $x$ there are $L^i$ fuzzy sets $A^i_l$, $l = 1, \ldots, L^i$. $L^i$ corresponds to the number of fuzzy rules of the system at step $t$.

$$a_{li} = e^{-\frac{(x_i - c_{li})^2}{2\sigma^2}} \quad (5)$$

where $l = 1, \ldots, L^i$, $i = 1, \ldots, n$ and $c_{li}$ is the $i$th coordinate of the $l$th cluster center. The radius $\sigma$ is defined a priori and kept constant during the evolving process.

The second layer contains $O_{r parcel}$ unineurons to aggregate the outputs of the input layer weighted by connections $w_{li}$:

$$z_l = S_{i=1}^n (a_{li} \ u w_{li}) \quad (6)$$

Here $z_l$, $l = 1, \ldots, L^l$ is the output of the $l$th unineuron, $S$ implemented using the $\text{max} s$-norm; $w_{li}$ are the synaptic weights. The third layer is a traditional neural network layer with sigmoidal activation functions $f(\cdot)$ as follows:

$$\hat{y}_j = f \left( \sum_{l=1}^{L^l} r_{lj} z_l \right) \quad (7)$$

where $j = 1, \ldots, m$, $m$ is the dimension of the output space, and $r_{lj}$ is the output weight connecting the $j$th output neuron with the $l$th hidden layer neuron. Figure 1 shows the unineuron structure.

C. Clustering Procedures

Most batch fuzzy neural networks relies on a clustering method to granulate the input space. We present two alternatives to perform clustering online: eClustering+ [9] and ePL [21].
The first approach estimates the density recursively at each data point by using (8).

\[ D_t(o_t) = \left( (t - 1)(so_t + 1) + b_t - 2 \sum_{j=1}^{n+m} o_{tj}h_{tj} \right)^{-1} \tag{8} \]

with \( o_t = [x^T, y^T]^T; D_1(o_1) = 1; b_t = b_{t-1} + \sum_{j=1}^{n+m} o_{(t-1)j}; b_1 = 0; h_{tj} = h_{(t-1)j} + o_{(t-1)j}; h_{1j} = 0; j = 1, \ldots, n + m; so_t = \sum_{j=1}^{n+m} o_{tj}^2 \).

Density is calculated recursively for each cluster center by (9):

\[ D_t(o^{\ast}) = \frac{t - 1 + (t - 2) \left( \frac{1}{D_{t-1}(o^{\ast})} - 1 \right)}{1 + d_{\text{dist}}} \tag{9} \]

where \( d_{\text{dist}} = \sum_{j=1}^{n+m} (o_{tj} - o_{(t-1)j}) \) and \( D_1(o^{\ast}) = 1 \). Cluster centers candidates are selected if the following condition is true: Condition A: \( D_t(o_t) > \max D_t(o^{\ast}) \) or \( D_t(o_t) < \min D_t(o^{\ast}) \) where \( t^{\ast} \) are the indexes of cluster centers. To avoid overlapping and redundancy, the cluster is created only if it does not satisfy: Condition B: \( \exists l \in [1, L^t] : a_{lj} > \nu \), \( 1 \leq i \leq n \). Figure 2 summarizes the steps of the eClustering+ algorithm.

2) ePL: The second alternative is to use the ePL (evolving Participatory Learning) clustering algorithm of [21]. Participatory learning assumes that learning and knowledge about a system depend on what the learning mechanism knows about the system itself. It uses a scheme with a compatibility measure \( \rho \) and an arousal index \( \gamma \). If a data sample with high compatibility arrives, then learning proceeds. Else, if data sample with low \( \rho \) are input, then the system treats it as new knowledge and a new cluster center is created by ePL.

The compatibility of data input \( x^t \) with each cluster is found as follows:

\[ \rho_i^t = 1 - \frac{||x^t - v_i||}{n} \tag{10} \]

The arousal index \( \gamma \) is updated by:

\[ \gamma_i^{t+1} = \gamma_i^t + \beta(1 - \rho_i^{t+1} - \gamma_i^t) \tag{11} \]

The value of \( \beta \in [0, 1] \) controls the rate of change of arousal: the closer \( \beta \) is to one, the faster the system is to sense compatibility variations. If the value of the arousal index exceeds a threshold \( \tau \in [0, 1] \), then a new cluster is created centered at \( x^t \). Otherwise, the most compatible cluster center, i.e. the one with the highest \( \rho \), is updated using:

\[ v_i^{t+1} = v_i^t + \alpha \rho_i^{t+1}(x^t - v_i^t) \tag{12} \]

Compatibility between centers are also computed to check for redundancy:

\[ \rho_{ij}^t = 1 - \sum_{j=1}^{L^t} ||v_i - v_j|| \tag{13} \]

That is, if \( \rho_{ij}^t > \lambda \), \( \lambda \in [0, 1] \), then the cluster is excluded for being redundant. Figure 3 summarizes the ePL steps.

D. Extreme Learning

The extreme learning method was introduced as a way to train single hidden layer feed-forward neural networks (SLFNNs). Originally developed in [15], the method chooses the hidden layer weights randomly and estimates the weights of the output layer using the least squares algorithm. Results
III. COMPUTATIONAL RESULTS

Simulations were performed using a 2.27GHz dual-core personal computer and MATLAB. The forgetting factor was fixed at 0.9. This choice is justified later in section IV. The root mean squared error (RMSE), computed at step $t$, is:

$$RMSE = \sqrt{\frac{1}{t} \sum_{j=1}^{t} (y^j - \hat{y}^j)^2}$$

The two clustering approaches will be identified by eXUninet+ when eClustering+ is used, and by eXUninet-ePL when ePL is adopted. Data were normalized in the range $[0.1, 0.9]$, and the errors were computed for normalized data. All the results reported were obtained running each method 20 times. The best and average performances are displayed.

A. Mackey-Glass Time Series

The Mackey-Glass time series is a well known benchmark whose data is generated using:

$$\frac{dx}{dt} = \frac{Ax^{t-\tau} - Bx^t}{1 + (x^{t-\tau})^C} - Bx^t, \ A, B, C > 0$$

Semi-periodic or chaotic behaviour depends on the parameters values chosen. Several studies adopt: $A = 0.2$, $B = 0.1$, $C = 10$ and $\tau = 17$, with a time step of 0.1 for integration [1, 4]. 3200 data samples were generated and used for training and testing simultaneously. The goal is to predict the value of $x^t$ 85 steps ahead, that is:

$$x^{t+85} = p(x^t, x^{t-6}, x^{t-12}, x^{t-18})$$

Results comparing the best and average RMSE values with alternative approaches reported in the literature are shown in Table I.

<table>
<thead>
<tr>
<th>Model</th>
<th>Ref.</th>
<th>Final # of Rules</th>
<th>RMSE</th>
<th>AVG. RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTS</td>
<td>[1]</td>
<td>24</td>
<td>0.0779</td>
<td>0.0779</td>
</tr>
<tr>
<td>DENFIS</td>
<td>[4]</td>
<td>25</td>
<td>0.0730</td>
<td>0.0730</td>
</tr>
<tr>
<td>FBeM</td>
<td>[2]</td>
<td>26</td>
<td>0.0968</td>
<td>0.0968</td>
</tr>
<tr>
<td>eXUninet</td>
<td>-</td>
<td>27</td>
<td>0.0801</td>
<td>0.0622</td>
</tr>
<tr>
<td>eXUninet-ePL</td>
<td>-</td>
<td>27</td>
<td>0.0550</td>
<td>0.0701</td>
</tr>
</tbody>
</table>

For this case the eXUninet AVG. RMSE was better than the others with 99% confidence. Training took an average of 28.7s for the entire dataset, an average of 9ms per sample. Figure 5 depicts the results of eXUninet+ for the first 500 data samples.

The methods addressed in this section utilize parameters to tune the final number of rules achieved. Although in some cases one can get lower errors or better clusters structures by optimizing these parameters, the goal was to demonstrate that the eXUninet does provide competitive results with
considerable lower training time and effort. Extensive tests considering parameters variations still need to be done to statistically evaluate its performance. As eXUninet+ does not make use of parameters for clustering, when possible, parameters of the other methods were adjusted to match approximately the same number of rules. This is also the case for the next example.

### B. Box-Jenkins Gas Furnace

Box-Jenkins gas furnace [22] also is a well known benchmark identification problem. The furnace input is the oxygen-methane gas flow rate \( x \) and the output is the concentration of carbon dioxide emitted \( y \). Positive correlation between the variables indicates the use of a predictor of the following form:

\[
y^t = p(y^{t-1}, x^{t-4})
\]  

(20)

Table II shows the results. In this case, eXUninet+ on average performed better than the eTS with 99% confidence, but worse than the remaining methods. Training took \(680\,\text{ms} \), \(2.3\,\text{ms} \) per sample. Figure 6 depicts the results of eXUninet+.

### IV. DISCUSSION

An issue that arises in evolving systems modeling is the trade-off between local tracking and global learning. Ideally the model should learn in a global manner with a small error so that knowledge from all current regions of the training data-set are embedded in the structure and parameters. The other extreme is to work only at the region near the current training example, still forecasting but more like locally tracking the output, similar to a filter.

The RLS algorithm with forgetting factor accounts for how long a past training sample contributes to the current network weight estimation. Combining the forgetting factor idea with the grow or shrink of the number of rules as suggested by the clustering algorithm, it is possible to address the local tracking or global learning. Evolving systems need an adaptation capability to deal with changing environments and by lowering the forgetting factor value the model becomes more responsive to detect and handle concept drifts.

Therefore to asses the generalization capabilities of the model analyzing only the RMSE during online learning is not enough due to local tracking, but if the target application allows or requires, it is possible to free resources by lowering the number of rules using forgetting factor \( \lambda \) values lower than the ones typically chosen in \([0.994, 0.998]\) range to indicate the need of a certain number of rules to learn globally. As we note in Table III, the model with lower Online RMSE loses generalization, but uses less resources because it stores a much lower number of rules. Similarly as in [4], the Test RMSE was computed using a set of 500 data samples distinct from those used for training. Parameters of the clustering algorithm were modified to generate different final number of rules.

Figure 7 illustrates the local track, global learning effect using the Mackey-Glass data and the eXUninet-ePL trained up to \(t = 250\) and setting \(\lambda = 0.9\).

### V. CONCLUSION

This paper has suggested a new evolving learning approach for hybrid fuzzy neural networks based on uninorms.
Computational results show that the fuzzy neural network is competitive with alternative evolving methods. The learning approach is simple, fast, accurate, and suitable for online applications.

A new view of the compromise between local-tracking and global-learning was given. This is relevant when the use of low cost hardware is an important requirement.

Future work shall explore examples with significant concept drift to further test the online learning capability of the approach. Detailed statistical analysis is also required to verify if the evolving learning approach is statistically superior. One can also analyse the use of various types of uninorms realizations at the global and local levels of neurons in the fuzzy layer of the eXUninet, as well as extensions for nullnorms-based neural fuzzy networks.

ACKNOWLEDGMENT

The authors are indebted to CNPq, the Brazilian National Research Council, for fellowship 151212/2010-4, and grants 471138/2010-0 and 304596/2009-4. They also thank CAPES, Brazilian Ministry of Education, for its support. The comments of the anonymous referees are also gratefully acknowledged.

REFERENCES