Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

Evolving Granular Classification Neural Networks

Daniel F. Leite+; Pyramo Costa Jr.*; Fernando Gomide+
+Faculty of Electrical and Computer Engineering, University of Campinas, Brazil
*Graduate Program in Electrical Engineering, Catholic University of Minas Gerais, Brazil
danfl7@dca.fee.unicamp.br, pyramo@pucminas.br, gomide@dca.fee.unicamp.br

***Abstract*—The objective of this study is to introduce the concept of evolving granular neural networks (eGNN) and to develop a framework of information granulation and its role in the online design of neural networks. The suggested eGNN are neural models supported by granule-based learning algorithms whose aim is to tackle classification problems in continuously changing environments. eGNN are constructed from streams of data using fast incremental learning algorithms. eGNN models require a relatively small amount of memory to perform classification tasks. Basically, they try to find information occurring in the incoming data using the concept of granules and T-S neurons as basic processing elements. The main characteristics of eGNN models are continuous learning, self-organization, and adaptation to unknown environments. Association rules and parameters can be easily extracted from its structure at any step during the evolving process. The rule base gives a granular description of the behavior of the system in the input space together with the associated classes. To illustrate the effectiveness of the approach, the paper considers the Iris and Wine benchmark problems.**

I. Introduction

One important aspect of the intelligence of systems lies in their ability to adapt to new situations [1]. To achieve adaptation, systems must be equipped with learning algorithms to continuously improve or, at least, no degrade the system performance in changing environments. The approaches discussed in this paper are classification oriented models that need incremental adaptability. In particular, researchers in the pattern classification area are facing the challenge of classify streams of data using models that adapt or evolve their structures and parameters based on possible new information contained in the data. These are online models often referred to as evolving classifiers.

A classifier is said to be evolving if it has the following characteristics:

- Ability of online learning from data streams. Once data are processed, there is no way to look back to historical data;
- No *a priori* knowledge about the system structure is needed (structural online adaptation);
- No *a priori* knowledge about the statistical properties of data is required (nonparametric classification);
- No *a priori* knowledge about the number of classes and prototypes per class is needed;
- No prototype initialization is required.

Moreover, it is very desired that evolving classifiers possess fast knowledge assimilation, small memory requirements, and nonlinear separation ability.

The design of evolving classifiers is mainly concerned with the construction of learning mechanisms to induce new knowledge without 'catastrophic forgetting' and/or to refine the existing knowledge. The whole problem is viewed as how to accommodate new data in an incremental way while keeping the system in use.

Classification processes often require simultaneous model construction and testing in environments which constantly evolves over time. If a static classifier is used to handle data streams, then the accuracy of the underlying classification process is likely to decrease when, for example, there is a sudden burst of exemplars belonging to a particular class. Off-line pre-trained classifiers may be good in certain contexts, but they need to be redesigned for new circumstances.

Examples of evolving classifiers that have been widely discussed in the literature include the following:
i) Fuzzy ARTMAP (FAM) [2] is one of many adaptive resonance network models. Its incremental learning ability naturally suggests its use in real-time problems solving;
ii) Growing Neural Gas (GNG) [3] tries to incrementally generate prototypes within the data space by constructing a graph that consists of a set of interconnected neurons;
iii) Unsupervised evolving connectionist systems [4] perform classification from fully unlabeled data streams. They develop their connectionist structure to model the incoming data. One of the major examples of classifiers in this category is the Evolving Self-Organizing Map (eSOM) [5]. eSOM uses the principles of self-organizing maps (SOM). It allows the prototype neurons to evolve in the input space, and at the same time, acquires and keeps topological representations;
iv) The evolving clustering method (ECM) is a one-pass algorithm for dynamic clustering of an input stream of data [6] when there is no predefined number of clusters. ECM is a distance-based clustering method where a threshold value determines the maximum radius a cluster can assume to accommodate input exemplars;
v) EFuNN [7] is an evolving fuzzy neural network model which adapts its structure and parameters through incremental, hybrid supervised/unsupervised online learning. EFuNN can accommodate new input data, including new features or new classes, through local element tuning;
vi) Evolving systems called eClass (evolving Classification) [8] were specifically designed to address dynamic classification problems. eClass can operate in online mode using non-iterative and recursive calculations. Several architectures for eClass models were introduced in [9], [10];

vii) The fuzzy min-max neural network (FMM) is a nonlinear classifier that uses fuzzy sets as pattern classes [11]. The fuzzy min-max learning algorithm is based on the expansion-contraction paradigm, and can be used in an one-pass through data scheme. Thus, FMM reveals itself as being a natural candidate for real-time applications. Actually, the FMM model is a particular case of eGNN models in the sense that granules/prototypes can individually assume intermediate values between the min and max operations. eGNN generalizes min-max using nullnorms and T-S neurons. Moreover, there are considerable differences and improvements in the learning algorithm that make eGNN models more effective and competitive. The intuitive idea of contraction-expansion learning paradigm remains.

In common, the learning mechanisms mentioned above generate prototypes when the incoming data are sufficiently dissimilar to the existing prototypes. Otherwise, an adjustment of some of the existing prototypes is conducted. The decision to assign new data to either an existing prototype or to a new prototype is based on control parameters like the size of granules/hyperboxes/clusters, error threshold or dissimilarity threshold.

After this introduction, the paper proceeds introducing T-S neurons in Section II. Next, Sections III and IV detail the eGNN modeling approach and its associated learning procedure. Section V presents results on how eGNN behaves when solving benchmark classification problems and compares its performance against alternative approaches. The paper concludes with Section VI summarizing the main contribution and suggesting issues for further investigation.

II. Brief Introduction to T-S Neurons

T-S neurons are neural implementations of T-S aggregation functions. In this paper we emphasize nullnorms [12], a special class of T-S operators. Nullnorms include T-norms and T-conorms (S-norms) as boundary cases. Nullnorms extend T-norms and S-norms in the sense they allow a flexible choice of a neutral element, called the absorbing element, of aggregation operations on fuzzy sets [13]. Nullnorms bind T-norms and S-norms constructs. As a result, we can switch between pure *and* and *or* properties of the logic operators occurring in this construct. T-S neurons inherit triangular norms logic connective processing as a conjunction, which make them flexible and logically appealing.

A. Nullnorms

Triangular norms T are commutative, associative and increasing binary operators on the unit square whose boundary conditions are $T(a,0) = 0$ and $T(a,1) = a$, $a \in [0,1]$. Similarly, T-conorms S are commutative, associative and increasing binary operators on the unit square whose boundary conditions are $S(a,0) = a$ and $S(a,1) = 1$. The neutral elements of T and S norms are $e = 1$ and $e = 0$, respectively.

Consider a continuous triangular norm T and a continuous triangular conorm S. A binary operator F is called a T-S aggregation function if it is increasing and commutative, and satisfies the boundary conditions $F(a,0) = T(F(1,0),a)$ and $F(a,1) = S(F(1,0),a)\ \forall\ a \in [0,1]$.

Nullnorms may be viewed as a way to generalize triangular norms. Nullnorms relax the assumption about the neutral element because they allow to choose $e \in [0,1]$. When the element e is equal to 0, a nullnorm turns into a T-norm, whereas when $e = 1$, the nullnorm becomes a S-norm.

Formally, a nullnorm is a binary relation $V : [0,1]\times[0,1] \rightarrow [0,1]$ satisfying the following properties:

- Commutativity: $V(a,b) = V(b,a)$,
- Monotonicity: $V(a,b) \leq V(a,c)$, if $b \leq c$,
- Associativity: $V(a,V(b,c)) = V(V(a,b),c)$,
- Absorbing element $e \in [0,1]$: $V(a,e) = e$,

and that satisfies $V(a,0) = a\ \forall\ a \in [0,e]$ and $V(a,1) = a\ \forall\ a \in [e,1]$, where $a,b,c \in [0,1]$. Nullnorms are a special class of T-S aggregation functions. In this work, we confine ourselves to the following family of constructs of nullnorms that seems to be highly interpretative [14]:

$$V(a,b) = \begin{cases} e\ \ S\ \ \left(\frac{a}{e},\frac{b}{e}\right) & \text{if } a,b \in [0,e], \\ (e+(1-e))\ \ T\ \ \left(\frac{a-e}{1-e},\frac{b-e}{1-e}\right) & \text{if } a,b \in [e,1], \\ e & \text{otherwise.} \end{cases}$$

In virtue of the above restriction, we can observe in Fig. 1 that the choice of the value of e defines the size of the region Ω and deliver some flexibility to nullnorms realization. Note also that the description of a nullnorm has the T-norm "above" the S-norm (T-norm for larger values and S-norm for small values). There is no duality requirement between any of the assumed triangular norms. The discrimination of and-dominated and or-dominated nullnorms refers to the value of the absorbing element $e \in [0,0.5[$ and $e \in\ ]0.5,1]$, respectively.

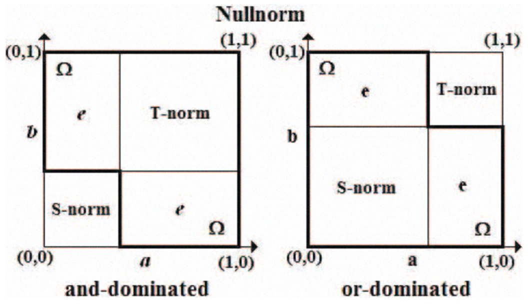


Fig. 1. Alternative realizations of nullnorms

A motivation for using nullnorms is deduced from the fact that T-norms describe situations when both conditions a and b are absolutely necessary, so if one of these conditions is not satisfied, we completely reject the corresponding alternative. There are many such situations, but there are also many other situations in which, although we want the first condition to be satisfied and the second condition to be satisfied, etc., but if one of these conditions is not satisfied, we may still consider the corresponding alternative.

B. T-S Neurons

T-S neurons are neural implementations of nullnorms. Formally, let $X = \{x_1, x_2, ..., x_n\}$ be an input vector in the unit hypercube, $X \in [0,1]^n$, and $W = \{w_1, w_2, ..., w_n\}$ its corresponding connections (weights), $W \in [0,1]^n$. The nullnorm aggregation of the weighted inputs produces the output $y \in [0,1]$, namely

$$y = V(x_1 w_1, x_2 w_2, ..., x_n w_n).$$

This relation emphasizes the parametric flexibility residing with the connections that are necessary to support learning. We adopt the notation $y = V(x, w)$ for short. Fig. 2 shows a schematic view of the neural processing. The absorbing element e can be adjusted through learning. What is remarkable is the diversity of nonlinear characteristics of the mapping produced by such neurons can assume depending on how W and e are chosen.

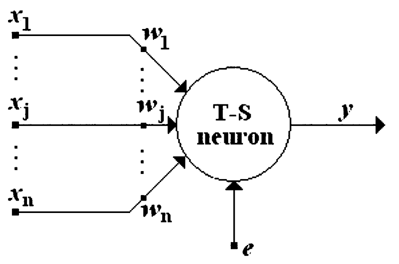


Fig. 2. Schematic view of a T-S neuron: the synaptic processing uses algebraic product and the neuron employs nullnorm aggregation

Given this structure, the processing for each input can be done independently and because of the adjustable value of the absorbing element, we encounter a different way of aggregation processing that permits intermediate assumption between the operations *and* (T-noms) and *or* (S-norms) [14]. A possible drawback of this structure is that although the neuron exhibits a high level of flexibility, its interpretation could be more complicated given the semantics of nullnorms. Likewise, the learning approach might cause more difficulties.

III. Evolving Granular Neural Network - eGNN

Granular Computing (GrC) has emerged as a new information processing paradigm in the last ten years [15], [16], [17]. The foundation of GrC is the fact that precision is an expensive and often unnecessary goal in modeling complex systems. In particular, human reasoning does not appear to operate at a numeric level of precision, but at a coarser, more abstract level of detail. GrC is a framework for expressing these abstractions in computational processes.

The concept of granular neural networks (GNN) was first established in [18] emphasizing artificial neural networks capable of processing granular data. These granules of data are inputs and outputs of GNN. For example, environmental data are inputs and human actions are outputs of biological neural networks. Basically, the development of GNN involves two main phases, refer to Fig. 3:

- Granulation of numeric data. At this level a collection of information granules is formed.
- The construction of the neural network. Now any learning that takes place with the neural network is based on the information granules rather than original data. As a consequence, the neural network is not exposed to the original data of a far higher granularity and far more numerous as the information granules.

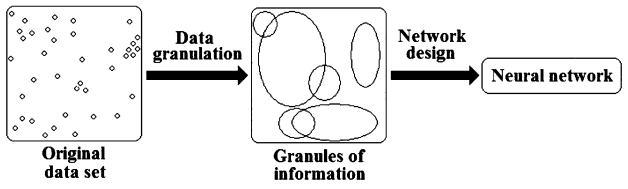


Fig. 3. Two-phase design of granular neural networks

Since the conception of GNN, studies suggesting variations and extension of the original content have been developed [19], [20]. In this paper we introduce evolving versions of GNN. The aim is to develop granule-based learning algorithms to tackle classification problems in continuously changing, dynamic environments. The evolving granular neural networks suggested here are constructed by processing streams of data using fast incremental learning algorithms. eGNN modeling requires a relatively small memory amount to perform classification tasks. Basically, eGNN try to find information granules occurring in the incoming data using known granules and T-S neurons in the processing steps. The main characteristics of eGNN modeling include continuous learning, self-organization and adaptation to unknown environments. Association rules can be easily extracted from its structure and parameters at any time during the evolving process. The evolved rule base offers a granular description of the current behavior of the system considering the input space and the associated classes.

Generally, eGNN models develop and work with a small amount of granules. The number of granules needs not to be pre-defined. No granules exist prior to learning; they are created by the evolving process. Predominantly, the evolving granular neural models use constructive (bottom-up) modeling mechanisms and decomposition-based (top-down) mechanisms. Association rules in the form of IF-THEN statements are easily extracted from the eGNN models at any step.

Fundamentally, eGNN models use class exemplars in the form of fuzzy hyperrectangles, the granules, to perform classification tasks. During evolution, granules are created, updated or shrunk along one or more dimension of the input space. More specifically, the newest input is first matched against the existing granules. Three possibilities may happen: i) the new input fits more than one existing granule. Thus, the one presenting highest membership degree given by a nullnorm-based aggregation operation is selected to accommodate the input. The selected granule must, however, be associated with the same class label of that of the incoming data. Otherwise, the granule with the second highest membership degree is selected to accommodate the input, and so on. The accommodation basically consists in adjusting the parameters of the membership functions of the granules and/or the parameters of the T-S neurons to better fit similar inputs at future steps; ii) the input falls into only one existing granule. Then, this granule and its corresponding T-S neuron are updated to accommodate the input; and iii) none of the existing granules accommodates the input. Thus, a new granule is created perfectly matching the input. Shrunk is a refinement mechanism of the collection of granules. It is applied when the granule that matches an input is assigned to a different class label, the following decomposition mechanism proceeds: i) splitting the granule into two finer ones; ii) creating a new granule perfectly matching the input; and iii) assigning this new granule to the correct class label. Details and characteristics of eGNN models are addressed in next sections.

IV. The eGNN Learning Approach

eGNN learn from a stream of data $(x, C)^{[h]}$, $h = 1, 2, ...$, where the class label $C^{[h]}$ is assumed to be known, given the corresponding input vector $x^{[h]}$. The model accommodates new knowledge contained in the incoming data either creating a new information granule or adapting the parameters of existing granules and that of their associated T-S neurons. Each information granule γ^i of a finite collection of information granules $\gamma = \{\gamma^1, \gamma^2, ..., \gamma^c\}$, defined in the input space $X \subseteq \Re^n$, is associated with a class k of the finite collection of classes $C = \{Class_1, Class_2, ..., Class_m\}$ in an output space $Y \subseteq \mathbb{N}^m$. The granule γ^i may assume different geometric forms and sizes, but here we emphasize multidimensional fuzzy intervals or, alternatively, fuzzy hyperrectangles or fuzzy hyperboxes.

The structure of the proposed eGNN is illustrated in Fig. 4. The network has 5-layers. The input layer basically distributes n-dimensional data vectors $x_j^{[h]}$, $j = 1, ..., n$, into the network. The evolving layer consists of granules of information γ^i, $i = 1, ..., c$, extracted from the data stream. Granules γ^i are defined by membership functions A_j^i, $j = 1, ..., n$, with different dimensions and universes, each one representing an attribute of the input vector. The aggregation layer encompasses the T-S neurons TSn^i, $i = 1, ..., c$. They process standardized (normalized) inputs $\tilde{x}_j^{i[h]}$, $j = 1, ..., n$, $i = 1, ..., c$, provenient from the granulation process, and aggregate data to generate a single output o^i, $i = 1, ..., c$. The output may be viewed as a measure of compatibility between a data pair and the existing granules. The decision layer compares the aggregation values of the granules, and the granule with the highest activation (winner) produces a binary output vector $\bar{C}_k^{[h]}$, $k = 1, ..., m$, with 1 in the entry corresponding to the class label assigned to the winner granule, and 0 in the remaining ones. The output layer compares the m-dimensional binary output of the network $\bar{C}_k^{[h]}$ with the desired output vector $C_k^{[h]}$, $k = 1, ..., m$. This operation may result in the network estimation error ϵ_k, $k = 1, ..., m$. In eGNN models, the number of granules in the evolving layer needs not to be pre-defined. In fact, no granules and T-S neurons exist prior to learning, they are created and adapted during the evolving process as new information is found in the data stream. The connection weights w_j^i, $j = 1, ..., n$, $i = 1, ..., c$, assign different degrees of relevance to different input attributes, while weights δ^i, $i = 1, ..., c$, depends on the amount of data within each granule γ^i. Generally speaking, δ^i is a mechanism to indicate denser and sparser data regions in the input space.

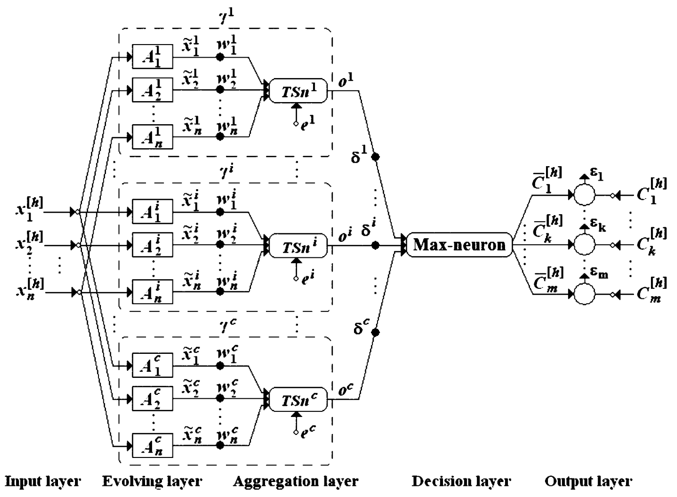


Fig. 4. Evolving granular neural network structure

Let $\rho \in X \subseteq \Re^n$ be the maximum size that an information granule can assume in the input space. Suitable choices of ρ are very important as it is directly related to the model transparency and accuracy. Any granule larger than the maximum size may result in losing some desirable regions. The value of ρ dictates the granularity (coarser, finer) of the granulation process and a possible control over the shape of the granules. Generally, the larger ρ, the fewer granules are created and, despite being easier to interpret, the less is the ability to capture nonlinear boundaries between classes. On the other hand, smaller ρ values may lead to data overfitting and loss of interpretability. In the limit, the input may become a single numeric quantity. Then, the attempt is to find an acceptable compromise between the extreme situations. An example of the granular classification we are interested in this paper is depicted in Fig. 5. The figure shows a collection of granules γ^i, $i = 1, ..., c$, constructed in light of the data available for the purpose of identifying the five classes (1 to 5) that appear in the input space.

A. Creating and Updating Granules

No granules exist until eGNN learning starts. They are created during the evolution process. The membership functions A_j^i, $j = 1, ..., n$, associated with a granule γ^i are defined in each of the corresponding input variable domain. They represent class attributes. We assume trapezoidal or triangular membership function for each A_j^i of the granule γ^i. Thus, they are defined by four parameters: the $j-th$ lower bound, l_j^i; the $j-th$ upper bound, L_j^i; and intermediate values of the function, λ_j^i and Λ_j^i. For trapezoidal functions $\lambda_j^i < \Lambda_j^i$, and

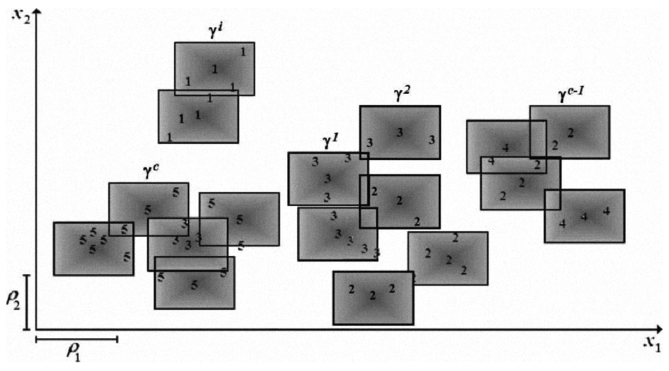
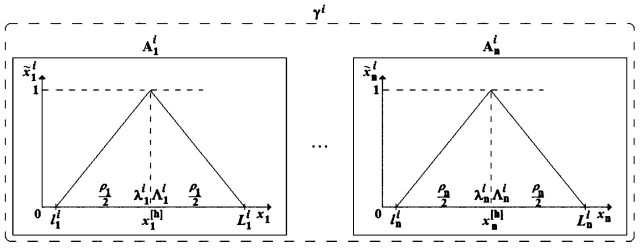


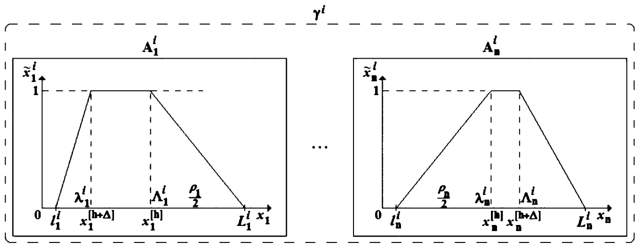
Fig. 5. Information granulation and five classes in input space

for triangular functions $\lambda_j^i = \Lambda_j^i$. Granules γ^i are associated with corresponding class labels C_k, $k = 1, ..., m$.

The procedure to create granules is run whenever an incoming data $x_j^{[h]}$ is not in $[l_j^i, L_j^i]$, $j = 1, ..., n$, $i = 1, ..., c$, where c is the number of currently existing granules. It is also run when, although $x_j^{[h]} \in [l_j^i, L_j^i] \forall j$, the class label of the incoming data $C^{[h]}$ differs from that assigned to γ^i. The new granule created, denoted by γ^{c+1}, is constructed using triangular membership functions A_j^{c+1}, $j = 1, ..., n$, with center $x^{[h]} \in \Re^n$. Therefore, the parameters of the membership fucntions are $l_j^{c+1} = x_j^{[h]} - \frac{\rho_j}{2}$; $\lambda_j^{c+1} = \Lambda_j^{c+1} = x_j^{[h]}$; and $L_j^{c+1} = x_j^{[h]} + \frac{\rho_j}{2}$; $\forall j$. Fig. 6(a) illustrates the procedure considering a generic granule γ^i.



(a) Creation of γ^i to accommodate the current input $x^{[h]}$



(b) Adaptation of γ^i to accommodate the current input $x^{[h+\Delta]}$

Fig. 6. Process of granules creation and adaptation

If a new input data vector $x^{[h+\Delta]}$, where Δ is a positive integer, is within the current bounds of γ^i for some i, and its respective class $C^{[h]}$ is the same as the one assigned to γ^i, then the parameters λ_j^i and Λ_j^i, $j = 1, ..., n$, are updated to accommodate $x_j^{[h+\Delta]}$, $j = 1, ..., n$. Basically, adaptation consists of setting either $\lambda_j^i = x_j^{[h+\Delta]}$, if $x_j^{[h+\Delta]}$ is in $[l_j^i, \lambda_j^i]$; or $\Lambda_j^i = x_j^{[h+\Delta]}$, if $x_j^{[h+\Delta]}$ is in $[\Lambda_j^i, L_j^i]$. Fig. 6(b) illustrates the adaptation approach.

It should be noted that there are certain situations where more than one granule could be selected to be updated to accommodate new data $(x, C)^{[h]}$. In these situations, the granule to be updated should be the one most compatible with the current data. A way to compute the compatibility between a data and candidate granules is to use T-S neurons.

B. Adjusting Evolving Layer Weights and Compressing Granules

The weights between the evolving and aggregation layer, namely w_j^i, $j = 1, ..., n$, $i = 1, ..., c$, aim at capturing the relevance with which each attribute j of granule i differentiate data classes. Initially, learning starts setting all w_j^i to 1. During evolution, some w_j^i may reduce their values depending on the data stream.

Similarly to other evolving models, eGNN suffer from the problem of sensitivity to the order in which data arrives. New data pairs $(x, C)^{[h]}$ may cause revision of γ^i if this granule is the one most compatible with $x^{[h]}$, but $C^{[h]}$ is different from the class assigned to γ^i. The following procedure is used to compress granule γ^i to reduce its compatibility with $x^{[h]}$. Two situations are of interest: i) if A_j^i, $i = 1, ..., c$, for any j results in a $\tilde{x}_j^{i[h]} \in \,]0, 1[$, then set $l_j^i = x_j^{[h]}$ if $x_j^{[h]} \leq \lambda_j^i$, or set $L_j^i = x_j^{[h]}$ if $x_j^{[h]} \geq \Lambda_j^i$. This procedure compress the membership function A_j^i of the granule γ^i; and ii) if A_j^i, $i = 1, ..., c$, is such that $\tilde{x}_j^{i[h]} = 1$ for any j, then A_j^i parameters are kept the same and the weight associated with the $j-th$ attribute of the granule i, w_j^i, is reduced as follows

$$w_j^i(\text{new}) = \beta w_j^i(\text{old}) \quad ,$$

where β is a decay constant with values in the open interval $]0, 1[$. The update procedure is justified because A_j^i does not satisfactorily helps to differentiate classes. After steps i) and ii), a granule γ^{c+1} is created adding normal triangular membership functions A_j^{c+1} centered at the input $x_j^{[h]}$, $j = 1, ..., n$. γ^{c+1} is then assigned to the correct class $C^{[h]}$.

Note that, because different degrees of importance to each attribute of the input vector are allowed, the procedure resembles techniques such as principal component analysis and wrappers in the sense that it searches relevant subsets from the current set of attributes. However, differently from principal component analysis and wrappers, the procedure does not eliminate less relevant attributes of the model as a whole, but it reduces the degree of importance of less relevant attributes as perceived by the local processing units.

C. Adjusting the T-S Neurons' Absorbing Element

The eGNN addressed in this paper uses and-dominated T-S neurons to process the attributes of the granules. The $i-th$ T-

S neuron, TSn^i, processes the n attributes of the normalized input $\tilde{x}_j^{i[h]}$, $j = 1, ..., n$, associated to the granule γ^i through $V(\tilde{x}_j^{i[h]}, w_j^i)$ $\forall j$. The result is a single output value o^i which may be viewed as the compatibility between $x^{[h]}$ and γ^i. A procedure to initialize and adjust the absorbing element of T-S neurons is as follows. First, at the beginning of the learning process, choose a T-norm and a S-norm for the T-S neurons. Create a granule γ^{c+1} and set the absorbing element of its corresponding T-S neuron as $e^{c+1} = 0$. This induces a T-norm. During evolution, some of the e^i, $i = 1, ..., c$, may increase their values depending on the data arrival. For instance, if only one or a small number of attributes of the input vector $x_j^{[h]}$, $j = 1, ..., n$, does not activate functions A_j^i, $j = 1, ..., n$, for any i, we still consider the granule γ^i as candidate to accommodate $(x, C)^{[h]}$ increasing its respective T-S neuron absorbing element e^i as follows

$$e^i(\text{new}) = e^i(\text{old}) + \chi(1 - e^i(\text{old})) \quad ,$$

where χ is a growth constant with values in $]0, 1[$.

To adjust e^i, $i = 1, ..., c$, during evolution, low membership values are compensated by high values using S-norm-based processing only for low membership values. If the actual input does not fit γ^i by a small number of attributes, then it could be made compatible to a certain degree with this granule through increasing e^i. This procedure generally avoids the creation of very similar granules and helps to keep a small number of granules in the model structure. Fig. 7 illustrates the idea of the procedure, which can be seen as a form of adaptive adjustment of the granules size. Note that, depending on the T- and S-norm chosen e.g. maximum S-norm and Lukasiewicz T-norm; bounded sum and nilpotent minimum, then the granules with their respective T-S neurons can be viewed in the input space as assuming different geometric forms.

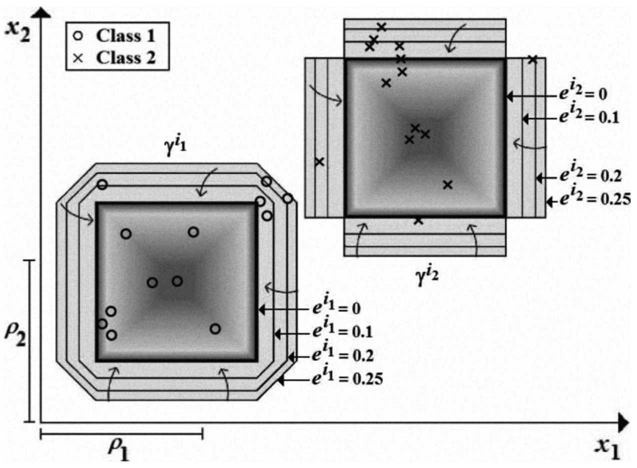


Fig. 7. Higher values of the absorbing element of a T-S neuron increases the corresponding granule size

D. Adjusting Decision Layer Weights and Deleting Granules

The connections between the T-S neurons and the max-neuron - see Fig. 4 - are weighted by δ^i, $i = 1, ..., c$, which encode the amount of data assigned to the corresponding granules γ^i. Generally speaking, the weights δ^i are a way to identify denser and sparser data regions in the input space. In general, the higher the value of δ^i, the bigger the probability to activate the $i-th$ granule in subsequent steps.

Learning starts setting all δ^i to 1. During evolution, the values of some δ^i may reduce or increase depending on *a priori* established criteria. For instance, a simple criteria is the following: if the $i-th$ granule does not activate after a certain number of steps, then reduce δ^i as

$$\delta^i(\text{new}) = \zeta\delta^i(\text{old}) \quad ,$$

where ζ is in $]0, 1[$. Otherwise, if γ^i activates often, then increase δ^i using

$$\delta^i(\text{new}) = \delta^i(\text{old}) + \zeta(1 - \delta^i(\text{old})) \quad .$$

A natural consequence of this procedure is that in continuously changing environments if granules are inactive for a number of steps, then it may be suppressed. This means that the actual process has changed and deleting granules means $\delta^i \to 0^+$, which is justified to maintain a reasonable amount of information and updated knowledge available. Clearly, if the application requires memorization of rare events, then the deletion procedure becomes prohibitive. Notice that the learning procedure detailed in this section enhances the eGNN ability to track the actual process evolution.

E. The Role of the Max-neuron

The single max-neuron of the eGNN model of Fig. 4 computes the highest value of its c-dimensional input vector. The entries of the input vector are the degrees of compatibility between γ^i, $i = 1, .., c$, and $x^{[h]}$. Using the degrees of compatibility, the max-neuron determines the class assigned to the granule that presents the highest compatibility to the current input. The max-neuron implements a winner-takes-all scheme and outputs a m-dimensional vector with value 1 in the winner entry and 0 in the remaining entries.

Let $\bar{C}_k^{[h]}$, $k = 1, ..., m$, be the binary m-dimensional vector computed by the network for the input $x^{[h]}$; and let ϵ_k be the error between $\bar{C}_k^{[h]}$ and the expected/desired class $C^{[h]}$. The error ϵ_k for all k is

$$\epsilon = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_k \\ \vdots \\ \epsilon_m \end{bmatrix} = \begin{bmatrix} C_1^{[h]} - \bar{C}_1^{[h]} \\ \vdots \\ C_k^{[h]} - \bar{C}_k^{[h]} \\ \vdots \\ C_m^{[h]} - \bar{C}_m^{[h]} \end{bmatrix} \quad .$$

When $\epsilon = \vec{0}$, the learning procedure refines the network parameters. When $\epsilon \neq \vec{0}$, the mechanisms of granules compression and weights updating proceed.

F. Choosing Model Granularity

Optimal granulation is a difficult and challenging problem whose solution is still open in the current literature. Estimation of granules sizes is done using heuristic and knowledge-based approaches. For instance, after an initial guess and preprocessing a small amount of incoming data, the spatial distribution and statistical properties of data may suggest values for the granules size ρ. The size of the granules can also be estimated using *a priori* knowledge about the problem in hands. The main idea to be kept in mind is that granules larger than ρ may cause loss of information about some desirable regions, specially when handling nonlinear mappings. If ρ assumes very small values, then the number of granules created can be very high, what may not be of practical interest.

Another mechanism to handle granularity is to learn values for ρ itself. For instance, consider a procedure that takes into account the amount of granules being created during a certain number of evolution steps. If the number of granules grows fast, then ρ could be increased during the next steps. Otherwise, if the number of granules grows slowly, then ρ could be decreased. Combinations of preprocessing and learning could be considered at the expense of additional computational effort.

G. The eGNN Modeling Procedure

The eGNN modeling procedure is summarized below:

```
BEGIN
Initialize $\rho$, $\beta$, $\chi$, $\zeta$, $c = 0$;
Select a type of T-norm and S-norm;
Do forever
  Read $(x, C)^{[h]}$, $h = 1, \ldots$;
  If (h = 1)    //First iteration
    Create $\gamma^{c+1}$ and $TSn^{c+1}$; Assign it to class $C^{[h]}$; $c = c + 1$;
  Else
    Feed $x^{[h]}$ into the network;
    Compute the number of granules $G$ with compatibility to $x^{[h]}$ higher than 0;
    If (G = 0)
      Adjust $e^i$, $i = 1, \ldots, c$;
      Create $\gamma^{c+1}$ and $TSn^{c+1}$; Assign it to class $C^{[h]}$; $c = c + 1$;
    Else
      For $g = 1, \ldots, G$
        Compute the $g - th$ winner granule ($g - th$ place) for $x^{[h]}$, namely $\gamma^\nu$;
        Extract the resulting output error $\epsilon^k$, $k = 1, \ldots, m$;
        If ($\epsilon^k$ $\forall k$ = 0)
          Adjust $A_j^\nu$, $j = 1, \ldots, n$, of $\gamma^\nu$;
          Break;
        Else
          Compress $A_j^\nu$, $j = 1, \ldots, n$; Adjust $w_j^\nu$, $j = 1, \ldots, n$, of $\gamma^\nu$;
          If ($g = G$)    //Last winner
            Adjust $e^i$, $i = 1, \ldots, c$;
            Create $\gamma^{c+1}$ and $TSn^{c+1}$; Assign it to $C^{[h]}$; $c = c + 1$;
          End
        End
      End
    End
  End
  //After a number of iterations
  Adjust $\delta^i$, $\delta = 1, \ldots, c$; Delete inactive granules;
END
```

V. Computational Experiments

A. Example A: Iris Classification

Basic description: The Iris benchmark dataset, obtained from the UCI Machine Learning Repository, is a widely used dataset in the pattern recognition context. Data contains 3 classes of 50 instances each, where each class refers to a type of Iris flower. One class is linearly separable from the other two. The latter are not linearly separable from each other.

Dataset characteristics:

- Number of instances (observations): 150;
- Number of variables: 4 quantitative input variables namely Sepal Length, Sepal Width, Petal Length and Petal Width, all in cm; and 1 output variable representing 3 classes of Iris plant: Setosa, Versicolour and Virginica;
- Training/Test data: 60/100 %.

For classification performance comparison we consider the following models: multi-layer Perceptron (MLP), Elman neural network, fuzzy C-Means (FCM), EFuNN [21], FMM [11], eClass [8], GFMM [22], and eGNN.

Results: To evaluate the effect of different parameterizations, the eGNN approach considers three sets of parameters: eGNN-1 parameters are $\rho_1 = [1.8 \;\; 1.2 \;\; 2.95 \;\; 1.2]$, $\beta_1 = \zeta_1 = 0.95$, $\chi_1 = 0.05$; eGNN-2 uses $\rho_2 = 0.6\rho_1$, $\beta_2 = \zeta_2 = 0.95$, $\chi_2 = 0.05$; and eGNN-3 adopts $\rho_3 = 0.4\rho_1$, $\beta_3 = \zeta_3 = 0.95$, $\chi_3 = 0.05$. Each experiment was run five times with the same parameters. In each run the data were shuffled to induce different orders in the data stream. Data is presented sequentially to each eGNN model only once to emulate a stream. eGNN start learning with an empty rule base and no pre-training. This is to simulate the true evolving process. Table I shows the performance of the eGNN models and that of the remaining classification models.

TABLE I
Iris classification performance

Model	No. granules/neurons/clusters	Training epochs	% Accuracy (Best)	% Accuracy (Avg.)
MLP	23	2600	97.3	96.6
Elman	23	4500	98.0	97.3
FCM	25	25	96.0	93.3
EFuNN	17	1	95.3	unavailable
FMM	48	1	97.3	unavailable
eClass	4	1	96.0	unavailable
GFMM	29	1	98.7	96.0
GFMM	43	1	100.0	96.0
eGNN-1	5	1	98.0	96.3
eGNN-2	12	1	99.3	97.7
eGNN-3	21	1	100.0	99.5

The results show that eGNN is the most accurate approach, similarly as GFMM, but eGNN uses a considerably lower number of granules (21 against 43). Interestingly, with only 5 granules the eGNN reaches a 98% accuracy, a performance slightly better than eClass using a similar compact structure. We noticed that the learning algorithm, the structural assumptions, the neural flexibility and the granular view of eGNN models are the major ingredients to achieve higher performance.

B. Example B: Wine classification

Basic description: The wine dataset, also obtained from the UCI Machine Learning Repository, has 3 classes representing the chemical analysis of wines derived from three Italian regions. From the classification point of view, this is a well behaved example in the sense that classes are clearly separated.

Dataset characteristics:

- Number of instances (observations): 178;
- Number of variables: 13 quantitative input variables namely Alcohol, Malic Acid, Ash, Alkalinity of Ash, Magnesium, Phenols, Flavanoids, Nonflavanoid Phenols, Proanthocyanins, Color Intensity, Hue, OD280/OD315 and Proline; 1 output representing 3 different cultivars;
- Training/Test data: 60/100 %.

For classification performance comparison we consider the following models: MLP, Elman, FCM, eClass [8], eClassB [9], eClassM [9], FAM [2], FMM [11], GFMM [22], and eGNN.

Results: As in the previous experiment, to evaluate the influence of the parameters, eGNN models employ two sets of parameter: eGNN-1 uses $\rho_1 = [2.3\quad 3.1\quad 1.1\quad 11.6\quad 55\quad 1.7\quad 2.8\quad 0.3\quad 1.9\quad 7\quad 0.7\quad 1.6\quad 840]$, $\beta_1 = \zeta_1 = 0.96$, $\chi_1 = 0.04$; and eGNN-2 adopts $\rho_2 = 0.6\rho_1$, $\beta_2 = \zeta_2 = 0.96$, $\chi_2 = 0.04$. Each experiment was run five times with the same parameters. Each run presents data sequentially in different orders. Table II shows the performance of the eGNN models and of the remaining classification models.

TABLE II
WINE CLASSIFICATION PERFORMANCE

Model	No. granules/neurons/clusters	Training epochs	% Accuracy (Best)	% Accuracy (Avg.)
MLP	33	376	99.4	97.9
Elman	33	345	98.9	97.0
FCM	30	73	94.4	91.9
eClass	7	1	95.9	unavailable
eClassB	9	1	94.4	unavailable
eClassM	28	1	97.2	unavailable
FAM	4	1	96.6	95.9
FMM	51	1	97.7	unavailable
GFMM	9	1	99.4	94.3
eGNN-1	8	1	99.4	98.0
eGNN-2	17	1	100.0	99.1

eGNN reaches the best performance if it uses 17 granules. An 8-granule structure achieves the best average accuracy. These results illustrate the potential of eGNN-based models to solve classification problems that demand incremental adaptability.

VI. Conclusion

The concept of evolving granular neural models has been introduced in this paper. A novel evolving granular neural network, eGNN, was suggested as a mechanism to develop evolving models of systems. The eGNN approach evolves models from information granules and from T-S neurons associated with granules. This provides eGNN models the ability to develop highly flexible structures and strong mechanisms to track the evolution. Experiments with Iris and Wine benchmark classification problems have shown that eGNN is competitive when compared against alternative nonlinear classification techniques. Further work shall consider refinement mechanisms and the development of evolving granular neural networks to handle unlabeled data and a mixture of labeled and unlabeled data.

Acknowledgment

The first author acknowledges CAPES, Brazilian Ministry of Education, for fellowship. The second author thanks the Energy Company of Minas Gerais - CEMIG, Brazil, for grant - P&D178. The last author is grateful to CNPq, the Brazilian National Research Council, for grant 304857/2006-8.

References

[1] A. Bouchachia, B. Gabrys, Z. Sahel, "Overview of some incremental learning algorithms", *IEEE Int. Fuzzy Systems Conf.*, Jul. 2007, pp: 1-6.
[2] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, D. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps", *IEEE Transactions on Neural Networks*, Vol. 3-5, Sep. 1992, pp: 698-713.
[3] B. Fritzke, "A Growing Neural Gas Network Learns Topologies", *Advances in Neural Information Processing Systems*, 1995, pp: 625-632.
[4] N. Kasabov, *Evolving Connectionist Systems: The Knowledge Engineering Approach*, Springer, 2^{nd} edition, 2007, 451p.
[5] Da Deng, N. Kasabov, "ESOM: an algorithm to evolve self-organizing maps from online data streams", *Proceedings of the IEEE International Joint Conference on Neural Networks*, Jul. 2000, pp: 3-8.
[6] N. Kasabov, Q. Song, "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its Application", *IEEE Transactions on Fuzzy Systems*, Vol. 10-2, Apr. 2002, pp: 144-154.
[7] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning", *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, Vol. 31-6, 2001, pp: 902-918.
[8] P. Angelov, Z. Xiaowei, F. Klawonn, "Evolving Fuzzy Rule-based Classifiers", *IEEE Symposium on Computational Intelligence in Image and Signal Processing*, Apr. 2007, pp: 220-225.
[9] P. Angelov, Z. Xiaowei, D. Filev, E. Lughofer, "Architectures for evolving fuzzy rule-based classifiers", *IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2007, pp: 2050-2055.
[10] P. Angelov, X. Zhou, "Evolving Fuzzy Rule-Based Classifiers from Data Streams", *IEEE Tran. on Fuzzy Systems*, Vol.16-6, Dec.2008, pp:1462-75.
[11] P. Simpson, "Fuzzy min-max neural networks. Part I: Classification", *IEEE Trans. on Neural Networks*, Vol. 3-5, Sept. 1992, pp: 776-786.
[12] T. Calvo, J. De Baets, J. Fodor, "The functional equations of Frank and Alsina for uninorms and nullnorms", *Fuzzy Sets and Systems*, vol. 120, 2001, pp: 385-394.
[13] M. Hell, F. Gomide, P. Costa Jr., "Neurons and Neural Fuzzy Networks Based on Nullnorms", *Brazilian Sym. on Neural Net.*,Oct.2008,pp:123-28.
[14] W. Pedrycz, "Logic-Based Fuzzy Neurocomputing With Unineurons", *IEEE Transactions on Fuzzy Systems*, vol. 14-6, Dec. 2006, pp: 860-873.
[15] W. Pedrycz, F. Gomide, *Fuzzy Systems Engineering: Toward Human-Centric Computing*, Wiley-IEEE Press, 1^{st} edition, 2007, 526p.
[16] A. Bargiela, W. Pedrycz, *Granular Computing: An Introduction*, Kluwer Academic, 1^{st} edition, 2002, 452p.
[17] W. Pedrycz, A. Skowron, V. Kreinovich, *Handbook of Granular Computing*, Wiley-Interscience, 2008, 1148p.
[18] W. Pedrycz, W. Vukovich, "Granular Neural Networks", *Neurocomputing*, Vol. 36, 2001, pp: 205-224.
[19] S. Dick, A. Tappenden, C. Badke, O. Olarewaju, "A Novel Granular Neural Network Architecture", *Annual Meeting of the North American Fuzzy Information Processing Society* - NAFIPS, Jun. 2007, pp: 42-47.
[20] Y. Zhang, B. Jin, Y. Tang, "Granular Neural Networks With Evolutionary Interval Learning", *IEEE Transactions on Fuzzy Systems*, Vol. 16-2, Apr. 2008, pp: 309-319.
[21] N. Kasabov, B. Woodford, "Rule insertion and rule extraction from evolving fuzzy neural networks: algorithms and applications for building adaptive, intelligent expert systems", *IEEE International Fuzzy Systems Conference Proceedings*, Vol. 3-1, 1999, pp: 1406-1411.
[22] B. Gabrys, A. Bargiela, "General fuzzy min-max neural network for clustering and classification", *IEEE Transactions on Neural Networks*, Vol. 11-3, 2000, pp: 769-783.