Evolving granular neural network for semi-supervised data stream classification

Conference Paper · August 2010

DOI: 10.1109/IJCNN.2010.5596303 · Source: IEEE Xplore

Daniel Leite
Universidade Federal de Lavras (UFLA)
38 PUBLICATIONS **303** CITATIONS

SEE PROFILE

Pyramo Costa Jr.
Pontifícia Universidade Católica de Minas Gerais
45 PUBLICATIONS **339** CITATIONS

SEE PROFILE

WCCI 2010 IEEE World Congress on Computational Intelligence
July, 18-23, 2010 - CCIB, Barcelona, Spain

IJCNN

Evolving Granular Neural Network for Semi-supervised Data Stream Classification

Daniel Leite, *Student Member, IEEE*, Pyramo Costa Jr. and Fernando Gomide, *Senior Member, IEEE*

***Abstract*—In this paper we introduce an adaptive fuzzy neural network framework for classification of data stream using a partially supervised learning algorithm. The framework consists of an evolving granular neural network capable of processing nonstationary data streams using a one-pass incremental algorithm. The granular neural network evolves fuzzy hyperboxes and uses nullnorm based neurons to classify data. The learning algorithm performs structural and parametric adaptation whenever environment changes are reflected in input data. It needs no prior statistical knowledge about data and classes. Computational experiments show that the fuzzy granular neural network is robust against different types of concept drift, and is able to handle unlabeled examples efficiently.**

I. Introduction

The continuous increase in availability of large amounts of data has motivated considerable research to develop new online algorithms to process data streams [1]-[8]. Mining fast-moving, quickly changing data streams brings unique problems and requires considerable effort.

One of the most important issues in online data mining concerns the intrinsic nonstationarity of the data streams. For instance, in industry, machines suffer from stresses, aging, and faults; in economic systems, performance indicators and stock indices vary; in communication systems, parameters and conditions of transmission media are also subject to continuous changes.

In nonstationary environments, the characteristics and underlying distribution of data are time varying. In machine learning terminology this means concept drift [9]-[11]. For example, in classification, concepts refer to classes or class boundaries, and drifts are the boundary changes over time. Drifts can be gradual or abrupt (concept shift), contracting or expanding, deterministic or random, or cyclical (due to seasonality) [12]. A major requirement of online learning is to promptly detect and cope with drifts mirrored in the data.

A challenge to develop models online from large data streams concerns the cost to store historical states. Ideally, the current system model should retain all previous relevant knowledge, and rely on the newest inputs to perform classification. New data may carry new information about new classes and require structural adaptation of classifiers. Noisy data and data with missing features or labels are also common and must be handled properly. Standard data mining techniques usually assume a form of stationarity while learning algorithms often need multiple passes over datasets. Therefore, they do not meet the requirements needed for online learning.

This paper suggests a granular neural network (eGNN) approach to develop classification models online. The eGNN approach is an evolving variant of neural fuzzy systems suited to deal with dynamic environments. It originated from our recent studies [6] on adaptive processing of nonstationary data streams. Here we extend the original supervised approach to the more general and efficient partially and unsupervised approaches.

Generally speaking, eGNN summarizes the system and associated classes using fuzzy hyperboxes to granulate data in the feature space and fuzzy neurons to aggregate input data features. The learning algorithm incrementally adapts the eGNN structure and its parameters whenever the system changes. The algorithm can handle labeled and unlabeled inputs concomitantly.

After this introduction the paper proceeds as follows. Section II describes the semi-supervised learning task we address in this paper. Section III gives a brief review of the fuzzy neuron which is part of the basic processing elements of eGNN. Sections IV details the eGNN structure and processing characteristics, and Section V focuses on the learning procedure. Section VI presents the experimental results and shows how eGNN behaves when dealing with different types of drifts and proportions of labeled examples. Section VII concludes the paper, summarizes its contributions, and suggests issues for further investigation.

II. Semi-supervised Learning

In terms of learning, classification and clustering can be done using supervised or unsupervised algorithms. In supervised learning, class labels are known in advance and decision boundaries between classes are sought. Unsupervised learning processes unlabeled training examples and attempts to find natural groupings in data. In both cases the final result is a partition of data into classes.

Semi-supervised approaches [13]-[15] use both, labeled and unlabeled data for training. Mixture of labeled and unlabeled patterns is easily found in practice. Often, the acquisition of labeled data requires human experts to manually classify training examples. This can be infeasible especially

Daniel Leite and Fernando Gomide are with the Department of Computer Engineering and Automation, Faculty of Electrical and Computer Engineering, University of Campinas, Sao Paulo, Brazil (phone: +55 19 35213823; email: danfl7@dca.fee.unicamp.br, gomide@dca.fee.unicamp.br).

Pyramo Costa Jr. is with the Graduate Program in Electrical Engineering, Catholic University of Minas Gerais, Brazil (phone: +55 31 33194305; email: pyramo@pucminas.br).

when dealing with large datasets in online environments. There are situations in which examples are labeled and on surface they may involve fully supervised learning methods and call for the standard mechanisms of classifier design. However, the labeling process could have been unreliable and therefore our confidence in the labels already assigned is relatively low. In this case we recur to the semi-supervised learning and accept only a small fraction of examples that we deem to be labeled accurately. The spectrum of possibilities of semi-supervised learning is illustrated in Fig. 1.

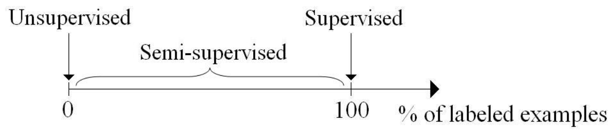


Fig. 1. Spectrum of semi-supervised learning

Consider an input variable x and a response variable y related to each other through $y = f(x)$. We seek an approximation to f that allow us to predict the value of y given x. In classification problems y is a class label, a value in a set $\{C_1, ..., C_m\} \in \mathbb{N}^m$ [9]. The relation f specifies the classes boundaries. In the general semi-supervised case, C_k may or may not be known when input x arrives.

Classification of data streams involves pairs $(x, C)^{[h]}$ of time-sequenced data indexed by h. When the characteristics of the classes change with time we say that there are *concept drifts*. Concept drift requires that online adaptive classifiers identify time varying relations $f^{[h]}$ to perform classification.

III. T-S Neuron Model

T-S neurons are neural implementations of aggregation operators. In this paper, we focus in nullnorms [16], a special class of aggregation operators that includes T-norms and S-norms as boundary cases. Nullnorms generalize T-norms and S-norms by allowing a flexible choice of the neutral element of the aggregation operation. In general, 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 an aggregation operation.

A. Nullnorms

T-norms (T) and T-conorms (S) are commutative, associative and monotone binary operators on the unit square whose boundary conditions are $T(a, 0) = 0$ and $T(a, 1) = a$; $S(a, 0) = a$ and $S(a, 1) = 1$, $a \in [0, 1]$. The neutral elements of T and S norms are $e = 1$ and $e = 0$, respectively [17].

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 an S-norm.

Formally, a nullnorm V is a binary relation $V : [0, 1] \times [0, 1] \rightarrow [0, 1]$ that satisfies the properties of commutativity, $V(a, b) = V(b, a)$; monotonicity, $V(a, b) \leq V(a, c)$, if $b \leq c$; associativity, $V(a, V(b, c)) = V(V(a, b), c)$; with neutral element $e \in [0, 1]$ such that $V(a, e) = e$, and that satisfies $V(a, 0) = a \; \forall \; a \in [0, e]$ and $V(a, 1) = a \; \forall \; a \in [e, 1]$, where $a, b, c \in [0, 1]$. In this paper, we limit ourselves to the family of nullnorms defined by the following constructor [18]:

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

A motivation to adopt nullnorms comes from the fact that T-norms describe situations when both conditions a and b are absolutely necessary. If one of these conditions is not satisfied, we completely reject the corresponding alternative. With nullnorms, if one of the conditions is not satisfied, we may still consider the contribution of the other condition in the aggregation operation. Partially specified conditions are handled similarly.

B. T-S Neurons

T-S neurons are artificial neuron models that use nullnorm as aggregation operation. Formally, let $x = (x_1, ..., x_n)$, $x \in [0, 1]^n$, be an input vector and $w = (w_1, ..., w_n)$, $w \in [0, 1]^n$ be its corresponding vector of weights. Employing the algebraic product to perform synaptic processing and the nullnorm operator to aggregate its result, the neuron output $o \in [0, 1]$ is

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

This expression will be denoted by $o = V(x, w) = TSn(x, w)$ for short. Basically, T-S neurons structures provide a diversity of nonlinear mappings between model inputs and outputs that depends of the choice of w, e, and triangular norms T and S. Particularly, the value of the neutral element allows intermediate assumption between the T and S norms specified.

IV. Evolving Granular Neural Networks

A. Introduction

The concept of granular neural networks (GNN) was first introduced in [19] and eGNN in [6]. Both approaches emphasize artificial neural networks capable of processing data originally numeric or granular. The eGNN approach focus on online incremental learning from data streams.

Learning in GNN and eGNN follows a common principle that involves two stages as summarized in Fig. 2. First, information granules - intervals or more generally fuzzy sets -

are constructed on a basis of original numeric representation. Then, learning - adaptation and refinement - in the neural network is based on the information granules rather than on original data. Therefore, the neural network does not need to be exposed to all data, which are far more numerous than the information granules; when lacking new information, examples are discarded.

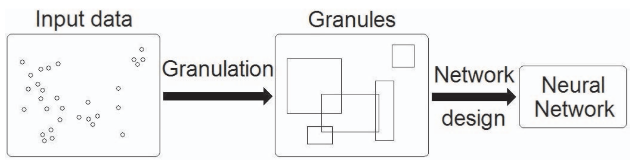


Fig. 2. Two-stage design of granular neural networks

Fundamentally, eGNN processes data streams using a fast incremental one-pass-through-the-data learning algorithm. The eGNN may start learning from scratch and with no prior knowledge of statistical properties of data and classes. The approach consists in forming decision boundaries between classes by granulating the feature space with fuzzy hyperboxes. A summary of eGNN main characteristics is as follows. The eGNN:

- adjusts its structure and parameters to learn new concepts, and forgets what is no longer relevant;
- deals with labeled and unlabeled examples simultaneously;
- detects drifts and can cope with uncertainty in the data;
- exhibits nonlinear separation ability;
- develops life-long learning using constructive bottom-up and destructive top-down mechanisms;

Before proceeding with the details of the learning algorithm, in the next section we address the structure of eGNN, and describe its processing flow and working principle.

B. eGNN Structure and Processing

eGNN learns from a stream of data $x^{[h]}$, $h = 1, 2, ...$ Training examples may or may not be accompanied by a class label $C^{[h]}$. Each information granule γ^i of a finite collection of granules $\gamma = \{\gamma^1, ..., \gamma^c\}$ in the feature space $X \subseteq \mathbb{R}^n$ is associated with a class C_k of the finite collection of classes $C = \{C_1, ..., C_m\}$ in the output space $Y \subseteq \mathbb{N}$. eGNN links the feature and output spaces using granules extracted from the data stream and a layer of T-S neurons.

The neural network has a 5-layer structure as shown in Fig. 3. The input layer basically fans feature vectors $x^{[h]} = (x_1, ..., x_j, ..., x_n)^{[h]}$, $h = 1, ...$, into the network; the granular layer consists of the set of information granules γ^i $\forall i$ formed within a scope of the feature space. Granules are allowed to partially overlap; the aggregation layer encompasses nullneurons TSn^i $\forall i$. They perform aggregation of membership values to generate values o^i $\forall i$ representing class compatibility between examples and granules; the decision layer compares the compatibility values o^i, and the class $\bar{C}_k$ associated with the granule γ^i with the highest compatibility value is output; the output layer are class label indicators. All layers, except the input layer, evolve as $x^{[h]}$, $h = 1, ...$, is input.

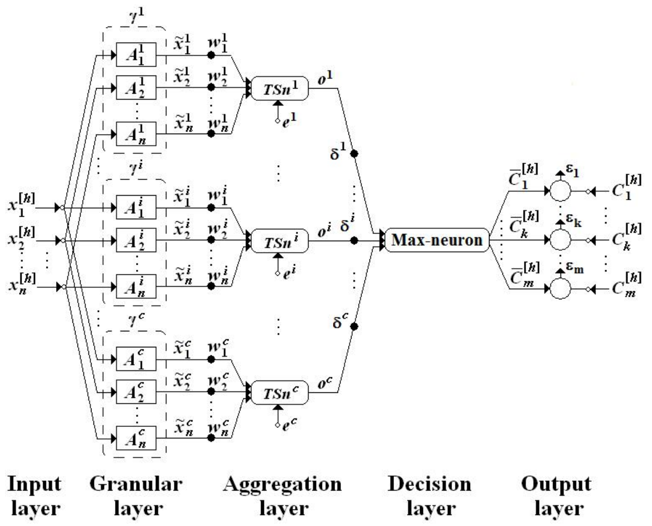


Fig. 3. Structure of the evolving granular neural network for classification

Structural and parametric adaptation of the eGNN classifier can be pursued in different ways depending on the application situation. For instance, the number of classes can be automatically controlled when it is known beforehand. The number of granules in the model structure can also be bounded if memory and processing time are constraints. In unknown environments both the number of granules and classes can be automatically controlled by the learning algorithm. We shall discuss about control modes opportunely in Section V-*C*.

V. Learning Algorithm

This section details the procedures embodying the eGNN learning algorithm.

A. Updating the Model Granularity

Guessing initial model parameters plays a key role to obtain problem solutions efficiently. This issue is somewhat alleviated if we provide a learning algorithm with procedures to continually update the parameters. The maximum size ρ that information granules can assume in the feature space defines model ability to capture nonlinear boundaries between classes. A procedure to adapt ρ online is as follows. Let $\{\gamma^1, ..., \gamma^\Theta\}$ be the granules created after a certain number of steps H_G. If the amount of granules Θ grows at a rate faster than a threshold η, then ρ is increased as follows

$$\rho(new) = \left(1 + \frac{\Theta}{H_G}\right)\rho(old).$$

Otherwise, if Θ grows at a rate smaller than η, then ρ is decreased as follows

$$\rho(new) = \left(1 - \frac{(\eta - \Theta)}{H_G}\right) \rho(old).$$

The value of $\rho(new)$ is kept constant during the next steps.

B. Creating and Updating Granules

No granules exist until eGNN starts learning. They are created during the evolution process. The membership functions $A_j^i\ \forall j$ associated with a granule γ^i are defined in each of the corresponding feature domain. For simplicity, we assume trapezoidal or triangular membership functions. They are defined by the quadruple $\{l_j^i,\ \lambda_j^i,\ \Lambda_j^i,\ L_j^i\}$, where l_j^i and L_j^i are endpoints, and λ_j^i and Λ_j^i are intermediate values of the function. For trapezoidal functions $\lambda_j^i < \Lambda_j^i$, and for triangular functions $\lambda_j^i = \Lambda_j^i$. Granules γ^i are associated with class labels $\bar{C}_k\ \forall k$.

The procedure to create granules runs whenever $x_j^{[h]} \notin [l_j^i, L_j^i]\ \forall j, i$; or when $x_j^{[h]} \in [l_j^i, L_j^i]\ \forall j$ and some i, but the class label of the input data $C^{[h]}$ differs from the one previously assigned to γ^i. The new granule γ^{c+1} is built using triangular membership functions $A_j^{c+1}\ \forall j$, with parameters $l_j^{c+1} = x_j^{[h]} - \frac{\rho_j}{2}$; $\lambda_j^{c+1} = \Lambda_j^{c+1} = x_j^{[h]}$; and $L_j^{c+1} = x_j^{[h]} + \frac{\rho_j}{2}$. Latter, if new input data $x^{[h+\Delta]}$, with Δ a positive integer, is within the current bounds of some granule γ^i and its respective class $C^{[h+\Delta]}$ is the same as the one assigned to γ^i, then the parameters λ_j^i and $\Lambda_j^i\ \forall j$ are updated to accommodate $x^{[h+\Delta]}$. Basically, adaptation consists of setting either $\lambda_j^i = x_j^{[h+\Delta]}$, if $x_j^{[h+\Delta]} \in [l_j^i, \lambda_j^i]$; or $\Lambda_j^i = x_j^{[h+\Delta]}$, if $x_j^{[h+\Delta]} \in [\Lambda_j^i, L_j^i]$.

C. Monitoring the Distance Matrix

A mechanism to measure how close the granules are is to monitor a matrix $Dis(\gamma)$ whose entries are the distances between any two granules. Null entries mean that the granules are the same, with identical structure and bounds. Higher distance values mean different granules, possibly with no overlap. The distance matrix helps to monitor the evolution process to merge granules and classes. This mechanism maintains a compact and updated model structure.

Defining

$$Dis(\gamma^w, \gamma^z) = \left\| l_j^w + \frac{(L_j^w - l_j^w)}{2},\ l_j^z + \frac{(L_j^z - l_j^z)}{2} \right\|,$$

where $Dis(\gamma^w, \gamma^z)$ is the distance between the granules γ^w and γ^z, and $||.||$ any distance measure, and collecting all $Dis(\gamma^w, \gamma^z)\ \forall w, z$ into a matrix we get:

$$Dis(\gamma) = \begin{bmatrix} Dis(\gamma 1, \gamma 1) & Dis(\gamma 1, \gamma 2) & .. & Dis(\gamma 1, \gamma^c) \\ Dis(\gamma 2, \gamma 1) & Dis(\gamma 2, \gamma 2) & .. & Dis(\gamma 2, \gamma^c) \\ \vdots & \vdots & .. & \vdots \\ Dis(\gamma^c, \gamma 1) & Dis(\gamma^c, \gamma 2) & .. & Dis(\gamma^c, \gamma^c) \end{bmatrix}$$

$Dis(\gamma)$ is a $c \times c$ symmetric matrix with zeros in the main diagonal. If an input datum causes the creation of a new granule, the matrix $Dis(\gamma)$ is updated accordingly.

Whenever the entry

$$Dis(\gamma^w, \gamma^z) \leq \hat{\Psi},$$

with $\hat{\Psi}$ being a predefined threshold value, granules γ^w and γ^z are declared similar and they can be either reassigned to the same label or merged. A way to monitor the number of classes is to assign granules with the smallest entries of $Dis(\gamma)$ to the same output class every time that the number of classes increases. Analogously, to keep the number of granules in the network structure constant a way is to merge the nearest neighbor granules γ^w and γ^z into a single granule γ^{c+1} with parameters $\lambda_j^{c+1} = \Lambda_j^{c+1} = min(l_j^z, l_j^w) + max(|L_j^w - l_j^z|, |L_j^z - l_j^w|)/2$; $l_j^{c+1} = \lambda_j^{c+1} - \rho/2$; and $L_j^{c+1} = \Lambda_j^{c+1} + \rho/2$.

D. Labeling Unlabeled Data

The procedure to online label unlabeled inputs we suggest in this paper falls within the category of *pre-labeling-based* approaches [13]. The procedure consists of labeling an example at a time whenever class information is lacking. When class label becomes available, the learning algorithm processes the example as usual.

Let the midpoint of a granule γ^i be

$$mp(\gamma^i) = \left(l_1^i + \frac{(L_1^i - l_1^i)}{2},\ ...,\ l_n^i + \frac{(L_n^i - l_n^i)}{2} \right).$$

The labeling procedure basically assigns the class label $\bar{C}_k$ associated with γ^i to some $(x, C)^{[h]}$ whenever $mp(\gamma^i)$ is the closest to $x^{[h]}$ according with

$$mp(\gamma^i) = \arg\min_i \left(\left\| x^{[h]} - mp(\gamma^1) \right\|, ..., \left\| x^{[h]} - mp(\gamma^c) \right\| \right)$$

E. Adapting the Weights of the Aggregation Layer

The weights $w_j^i\ \forall j, i$ of the aggregation layer aim at capturing the contribution of feature j of γ^i to differentiate classes. Initially, learning starts setting all w_j^i to 1. During evolution steps, some w_j^i may decrease their values depending on the input data.

Data pairs may cause revision of γ^i if this granule is the most compatible with $x^{[h]}$, but $C^{[h]}$ differs from the current class assigned to it. The following procedure is used to compress γ^i and reduce its compatibility with $x^{[h]}$. Two situations are of interest: i) if $A_j^i\ \forall i$ and some j is such that the membership degree $\tilde{x}_j^{i[h]} \in\ ]0,1[$, 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 γ^i; and ii) if $A_j^i\ \forall i$ is such that $\tilde{x}_j^{i[h]} = 1$ for some j, then A_j^i parameters are kept the same and the weight associated with the $j-th$ feature of γ^i is adapted as follows:

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

where $\beta \in [0,1]$ is a decay constant. The updating procedure is justified because A_j^i does not helps to satisfactorily differentiate classes. Note that this procedure resembles feature selection techniques such as [20] and [21] because different degrees of importance are allowed for each feature and the procedure looks for the relevant subsets of features.

F. Pruning

Keeping granules during long periods may turn the eGNN learning algorithm inefficient in tracking fast changes. One way to reduce this problem is to consider pruning mechanisms that preserve classification efficiency. Common pruning strategies include:

- replace-the-oldest granule;
- replace-the-weakest granule (delete the granule that induce classification errors); and
- delete the most inactive granule.

In this paper we opt by the approach of deleting inactive granules because it helps to detect drifts faster in the framework of semi-supervised learning. The weights of the decision layer (refer to Fig. 3) helps pruning as follows.

Decision layer weights $\delta^i\ \forall i$ encode the amount of data assigned to γ^i. Generally speaking, the weights δ^i are a way to identify dense and sparse regions in the feature space. The higher the value of δ^i, the bigger the chance to activate γ^i in subsequent steps. Learning starts setting all δ^i to 1. During evolution δ^i is reduced whenever γ^i is not activated during a certain number of steps as follows:

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

where $\zeta \in [0,1]$. If γ^i is activated, then δ^i is increased as follows

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

When $\delta^i \leq \vartheta$, with ϑ a threshold, γ^i is pruned to maintain a reasonable amount of information and update knowledge available. The values of ζ and ϑ depend on the stability and plasticity levels required by the application. If the application requires memorization of rare events, or if cyclical drifts are expected, then it may be the case to set $\vartheta = 0$ and let $\delta^i \rightarrow 0^+$.

G. Updating the Neutral Elements of T-S Neurons

eGNN uses and-dominated T-S neurons [17] to aggregate classification features. The $i-th$ neuron TSn^i processes $\tilde{x}_j^{i[h]}\ \forall j$ associated with γ^i through $V(\tilde{x}_j^{i[h]}, w_j^i)\ \forall j$. The result is a single output value o^i which gives the compatibility degree between $x^{[h]}$ and γ^i.

A procedure to initialize and adjust the neutral elements $e^i\ \forall i$ of the T-S neurons is as follows. First, at the beginning of the learning process, choose a T-norm and a S-norm. Whenever a granule γ^{c+1} is created, set the neutral element of the corresponding T-S neuron e^{c+1} to 0. This induces a T-norm. During evolution, some e^i may increase their values depending on the input data. For instance, if some (not all) features of $x_j^{[h]}\ \forall j$ activate membership functions $A_j^i\ \forall j$ of some γ^i, we may keep γ^i as a candidate to accommodate $(x,C)^{[h]}$ increasing e^i as follows:

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

where $\chi \in [0,1]$ is a growth constant. This procedure generally avoids creation of similar granules and helps to keep a small number of granules in network structure.

H. Making the Classification Decision

The single max *winner-takes-all* neuron of the eGNN model determines the highest value appearing in $o^i\ \forall i$. At its output the max-neuron yields the class label $\bar{C}_k$ assigned to the granule γ^ν with the highest compatibility degree for the current input $x^{[h]}$. The error between $\bar{C}_k$ and the expected/desired class $C^{[h]}$ is computed using:

$$\epsilon = |\bar{C}_k - C^{[h]}|.$$

When ϵ = 0, the learning algorithm refines the model parameters. When $\epsilon \neq 0$, the procedures for granule compression and weight updating are run. Note that if $C^{[h]} = \emptyset$, then the pre-labeling approach of Section V-*D* assigns a label to the corresponding input.

I. Detecting Outliers

Data not complying with the general classification model are called outliers [22]. Certain application environments must discard outliers because they may carry unacceptable noise, be wrong, be exceptions. There are, however, circumstances in which the emphasis is somehow the opposite; the interest is in rare events. Examples include fault detection, and financial fraud and diagnosis problems. The question on how to distinguish between rare events from outliers is still an open issue.

Outliers are detected in eGNN through a parameter $\check{\Psi}$ whose purpose is the opposite of the parameter $\hat{\Psi}$ of the merging procedure of Section V-*C*. An input data vector is considered an outlier whenever it induces the creation of a new granule γ^{c+1} such that

$$Dis(\gamma^{c+1}, \gamma^{i}) \geq \check{\Psi},\ \forall i.$$

This means that the new granule is far away from the remaining ones. Clearly, the meaning of far depends of the application domain. An alternative to monitor outliers is to use an arousal mechanism such as the one suggested in [23].

J. The eGNN Learning Algorithm

The learning algorithm detailed in the previous sections can be summarized as follow:

```
BEGIN
Initialize $\rho$, $H_G$, $\eta$, $\hat{\Psi}$, $\check{\Psi}$, $\beta$, $\chi$, $\zeta$, $\vartheta$, $c = 0$;
Select a type of T-norm and S-norm;
Do forever
  Read $(x, C)^{[h]}$, $h = 1, ...;$
  If ($h$ = 1)    //First iteration
    Create $\gamma^{c+1}$ (Sec. V-B) and $TSn^{c+1}$; Assign it to class $C^{[h]}$; $c = c + 1$;
  Else
    If ($C^{[h]} = \emptyset$)
      Execute the labeling procedure (Sec. V-D);
    Feed $x^{[h]}$ into the network;
    Compute the number of granules $G$ with compatibility to $x^{[h]}$ higher than 0;
    If ($G = 0$)
      Adjust $e^i$, $i = 1, ..., c$ (Sec. V-G);
      Create $\gamma^{c+1}$ and $TSn^{c+1}$; Assign it to class $C^{[h]}$; $c = c + 1$;
    Else
      For $g = 1, ..., G$
        Compute the $g - th$ winner granule ($g - th$ place) for $x^{[h]}$, namely $\gamma^{\nu}$;
        Extract the resulting output error $\epsilon_k$, $k = 1, ..., m$ (Sec. V-H);
        If ($\epsilon_k$ $\forall k$ = 0)
          Adjust $A_j^{\nu}$, $j = 1, ..., n$, of $\gamma^{\nu}$ (Sec. V-B);
          Break;
        Else
          Compress $A_j^{\nu}$, $j = 1, ..., n$; Adjust $w_j^{\nu}$, $j = 1, ..., n$ (Sec. V-E);
          If ($g = G$)    //Last winner
            Adjust $e^i$, $i = 1, ..., c$;
            Create $\gamma^{c+1}$ and $TSn^{c+1}$; Assign it to $C^{[h]}$; $c = c + 1$;
    If ($h = H_G$)
      Update the model granularity (Sec. V-A);
  Compute $Dis(\gamma)$; Merge granules and classes (Sec. V-C);
  Detect and delete outliers (Sec. V-I);
  Adjust $\delta^i$, $i = 1, ..., c$; Prune inactive granules (Sec. V-F);
END
```

VI. Experimental Results

In this section we report experimental results which show the effectiveness of the eGNN. Three experiments, each of which with a different purpose, have been performed. The first considers data distributed according to two partially overlapping Gaussians rotating around a central point. Each Gaussian represents a class. The task is to find a discriminant boundary between classes relying on the most recent examples only. The purpose of this experiment is to demonstrate the eGNN ability in capturing gradual drifts appearing in the data stream. The second experiment aims at evaluating eGNN capability to discover new classes. The purpose is to show that eGNN self adapts structurally to handle the concept shift, the new class in the data stream. The last experiment shows the eGNN ability to classify partially labeled drifting data. For reference, [9] and [12] have conducted similar experiments to evaluate performance of incremental algorithms. However, direct comparisons between the experimental approaches are not feasible due to slightly different assumptions.

The following parameter values were adopted during all experiments: $\rho^{[0]} = 0.25$, $H_G = 100$, $\eta = 2$, $\check{\Psi} = 5$, $\chi = \hat{\Psi} = 0.1$, $\beta = \zeta = 0.9$, $\vartheta = 0.2$. These are default values that have particularly worked well for different types of problems. The learning algorithm was run several times for evaluation. The results obtained were essentially the same in different runs.

A. Experiment 1: Rotation of Twin Gaussians

In this experiment, two partially overlapping Gaussians centered at $(4, 4)$ and $(6, 6)$ with dispersion fixed at 0.8 rotate by 90^o anti-clockwise around the point $(5, 5)$ as shown in Fig. 4. We look for the decision boundary between classes using the newest, randomly chosen input data.

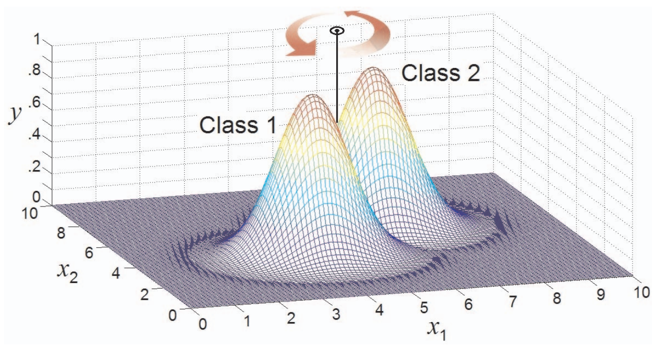
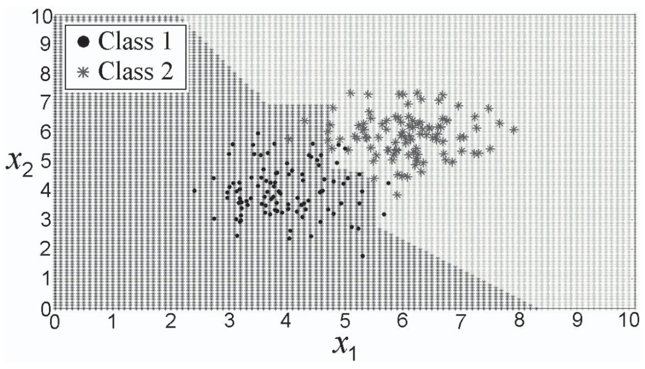


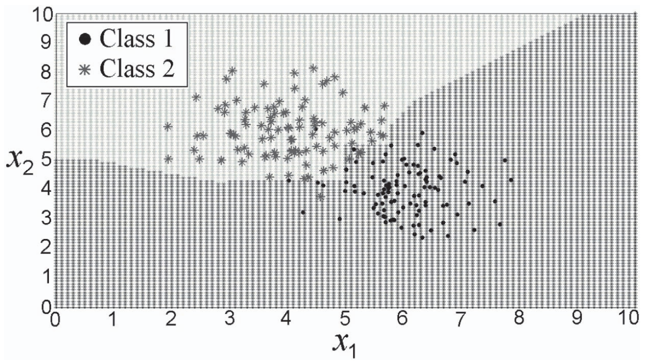
Fig. 4. The rotating Gaussians problem

Figure 5 shows the decision boundary and the last 200 examples at different evolution instants. The drift starts at $h = 200$ and ends at $h = 400$. At $h = 200$ the eGNN has 5 granules, two associated to Class 1 and three with Class 2. It attained a correct/wrong rate of 189/11, about

94.5% classification rate. After rotation, that is, after $h = 400$, eGNN employs 5 granules in its structure, three for Class 1 and two for Class 2, and achieves a 195/5, about 97.5% recognition performance.



(a)



(b)

Fig. 5. Decision boundary and last 200 data at (a) $h = 200$ and (b) $h = 400$

B. Experiment 2: New Class

In this section we assume that a new Gaussian class centered at $(7, 3)$ and with dispersion 0.8 appears at $h = 200$ as shown in Fig. 6. The eGNN structural adaptation must learn the previously unknown class as soon as the new class information appears in the input data stream.

Figure 7 shows classes boundaries and the last 200 measurements at $h = 400$. The eGNN evolved a total of 6 granules, three associated with each of the two first classes, during the first 200 steps, achieving a classification rate of 193/7 (96.5%). Data about the third class starts to arrive at $h = 200$ and at $h = 400$ eGNN exhibits 8 granules, three assigned to Class 1, two to Class 2, and three to Class 3, with a 190/10 (95.0%) classification rate. The membership functions that assemble the fuzzy hyperboxes of eGNN are shown in Fig. 8. Here granules $\{\gamma^1, \gamma^4, \gamma^8\}$ are associated with class 1, and $\{\gamma^2, \gamma^6\}$ and $\{\gamma^3, \gamma^5, \gamma^7\}$ with classes 2 and 3, respectively.

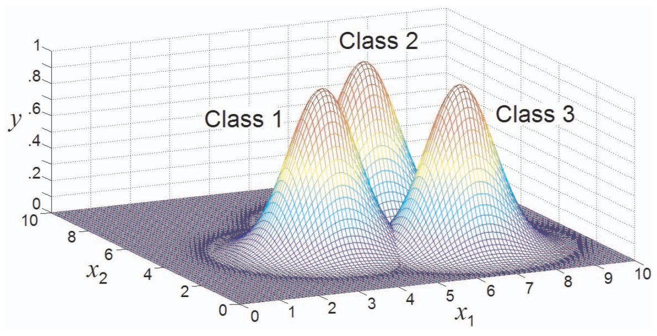


Fig. 6. A third class appears after $h = 200$

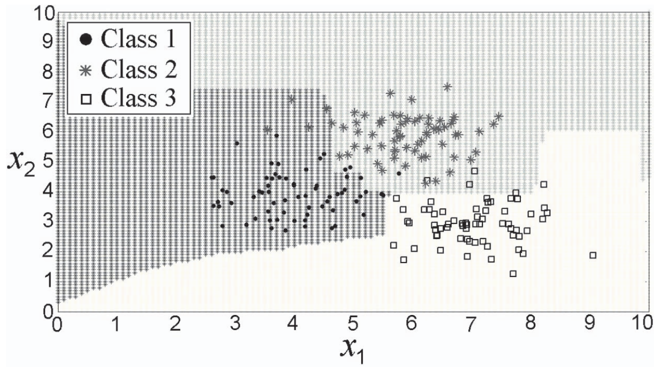


Fig. 7. Decision boundaries and last 200 examples at $h = 400$

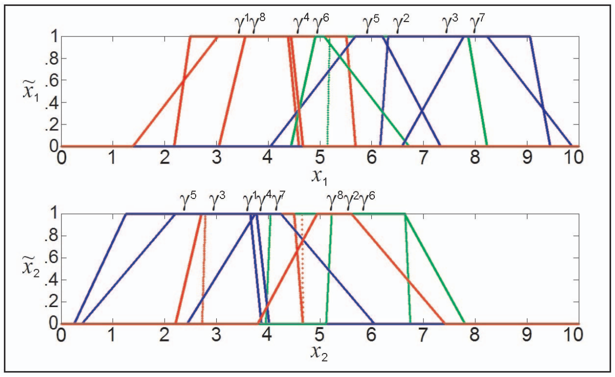


Fig. 8. eGNN membership functions at $h = 400$

C. Experiment 3: Combining Labeled and Unlabeled Data

The performance of the eGNN is evaluated now by varying the proportion of unlabeled data from 0% to 100%. We consider the two rotating Gaussian classes example and the three class example of the previous sections. Figure 9 illustrates the performance of the learning algorithm averaged over 5 runs.

Referring to Fig. 9 we notice that when eGNN considers all information contained in the data stream, including those of the unlabeled data, it achieves a classification rate considerably better than when discarding the unlabeled instances.

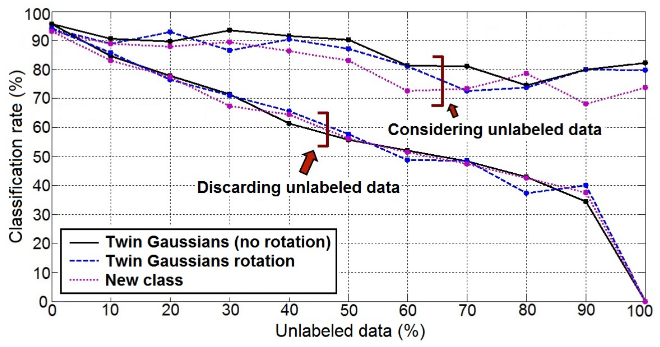


Fig. 9. eGNN performance using different proportions of labeled data

Unlabeled data guides eGNN classification successfully, especially in the cases where they represent larger proportions of the whole set of data. This evaluation is valid for all the experiments conducted, that is, for the problem of the twin Gaussians without drifts (stationary data stream); for the problem of the Gaussians rotation with gradual concept drift; and for the problem of the appearance of a new class (concept shift). The neural network recognition performance using the hybrid eGNN learning algorithm, and considering 50% of unlabeled data, degraded slightly in relation to when all examples were labeled, a degradation of about 6%. By contrast, if the model is not able to handle mixture of labeled and unlabeled data and use only labeled instances for training, a 35% reduction of the overall classification rate can be observed.

VII. Conclusion

An evolving granular neural fuzzy network structure and learning algorithm to classify streams of nonstationary data has been developed. The eGNN performs classification online, and handles concept drift efficiently. The effectiveness of the learning algorithm has been verified through experiments concerning different types of drifts and proportions of labeled and unlabeled data. The results confirm that the algorithm is robust and copes well with data nonstationarities. Further work shall consider evolving granular neural network with different types of granular input and output data.

Acknowledgment

The first author acknowledges CAPES, Brazilian Ministry of Education, for his 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 304596/2009-4.

References

[1] Angelov, P.; Filev, D.; “An approach to on-line identification of evolving Takagi-Sugeno models”, IEEE Transactions on SMC - Part B, vol. 34-1, 2004, pp: 484-498.

[2] Angelov, P.; Zhou, X.; “Evolving Fuzzy Systems from Data Streams in Real-Time”, Int. Symp. on Evolving Fuzzy Systems, 2006, pp: 29-35.

[3] Bouchachia, A.; Gabrys, B.; Sahel, Z.; “Overview of some incremental learning algorithms”, IEEE International Fuzzy Systems Conference, Jul. 2007, pp: 1-6.

[4] Gabrys, B.; Bargiela, A.; “General fuzzy min-max neural network for clustering and classification”, IEEE Transactions on Neural Networks, vol. 11-3, 2000, pp: 769-783.

[5] Leite, D. F.; Costa Jr., P.; Gomide, F.; “Interval-based evolving modeling”, IEEE Workshop on Evolving and Self-Developing Intelligent Systems, 2009, pp: 1-8.

[6] Leite, D. F.; Costa Jr., P.; Gomide, F.; “Evolving Granular Classification Neural Networks”, International Joint Conference on Neural Networks, 2009, pp: 1736-1743.

[7] Muhlbaier, M. D.; Topalis, A.; Polikar, R. “Learn^{++}.NC: Combining Ensemble of Classifiers With Dynamically Weighted Consult-and-Vote for Efficient Incremental Learning of New Classes”, IEEE Transactions on Neural Networks, vol. 20-1, 2009, pp: 152-168.

[8] Ozawa, S.; Pang, S.; Kasabov, N.; “Incremental Learning of Chunk Data for Online Pattern Classification Systems”, IEEE Transactions on Neural Networks, vol. 19-6, 2008, pp: 1061-1074.

[9] Garnett, R.; Roberts, S. J.; “Learning from Data Streams with Concept Drift”, Technical Report PARG-08-01, Dept. of Engineering Science, University of Oxford, Jun. 2008, 64p.

[10] Widmer, G.; Kubat, M.; “Learning in the Presence of Concept Drift and Hidden Contexts”, Machine Learning - Springer Netherlands, vol. 23-1, 1996, pp: 69-101.

[11] Bouchachia, A.; “Incremental Induction of Fuzzy Classification Rules”, IEEE Workshop on Evolving and Self-Developing Intelligent Systems, 2009, pp: 32-39.

[12] Elwell, R.; Polikar, R. “Incremental Learning in Nonstationary Environments with Controlled Forgetting”, International Joint Conference on Neural Networks, 2009, pp: 771 - 778.

[13] Gabrys, B.; Petrakieva, L.; “Combining labelled and unlabelled data in the design of pattern classification systems”, International Journal of Approximate Reasoning, vol. 35, 2004, pp: 251-273.

[14] Gabrys, B.; “Learning hybrid neuro-fuzzy classifier models from data: to combine or not to combine?”, Fuzzy Sets and Systems, vol. 147, 2004, pp: 39-56.

[15] Pedrycz, W.; “Knowledge-based clustering: from data to information granules”, Wiley, 1^{st} edition, 2005, 336p.

[16] Calvo, T.; De Baets, J.; Fodor, J.; “The functional equations of Frank and Alsina for uninorms and nullnorms”, *Fuzzy Sets and Systems*, vol. 120, 2001, pp: 385-394.

[17] Pedrycz, W.; Gomide, F.; “Fuzzy systems engineering: Toward human-centric computing”, Wiley, Hoboken, NJ, USA, 2007, 526p.

[18] Klement, E.; Mesiar, R.; Pap, E.; “Triangular Norms”, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000, 412p.

[19] Pedrycz, W.; Vukovich, W.; “Granular Neural Networks”, *Neurocomputing*, vol. 36, 2001, pp: 205-224.

[20] Pang, S.; Ozawa, S.; Kasabov, N.; “Incremental Linear Discriminant Analysis for Classification of Data Streams”, IEEE Transactions on SMC - Part B, vol. 35-5, 2005, pp: 905-914.

[21] Zhao, H.; Yuen, P. C.; Kwok, J. T.; “Incremental Linear Discriminant Analysis for Classification of Data Streams”, IEEE Transactions on SMC - Part B, vol. 36-4, 2006, pp: 873-886.

[22] Barnett, V.; Lewis, T.; “Outliers in Statistical Data”, Wiley Series in Probability and Mathematical Statistics, 3^{rd} edition, 1994, 604p.

[23] Silva, L.; Gomide, F.; Yager, R.; “Participatory Learning in Fuzzy Clustering”, IEEE Int. Conf. on Fuzzy Systems, 2005, pp: 857-861.