Classification Rules obtained from Dynamic Self-organizing Maps

Waldo Hasperué[1] and Laura Lanzarini[2]
{whasperue, laural}@lidi.info.unlp.edu.ar

III-LIDI (Institute of Research in Computer Sciences LIDI)
Faculty of Computer Sciences. National University of La Plata.
La Plata, Argentina, 1900

Abstract

Knowledge discovery from existing information is a non-trivial process that consists in identifying valid, new, potentially useful, and understandable patterns from available data. Data Mining is the area of Computer Sciences which refers to the application of different methods so as to obtain patterns and models.
One of the mostly used solutions is based on non-supervised adaptive strategies allowing the classification of available data. Towards this direction, dynamic competitive neural networks have proved to be capable of providing good results. However, their application in the area of Data Mining is constrained due to their "black box" type functioning, in which it is hard to justify the acquired knowledge.
This paper proposes a new strategy for obtaining classification rules from a dynamic competitive neural network trained with the AVGSOM method. Such method has been selected for its capacity of preserving input data topology, essential characteristic necessary to obtain the proper initial hypercubes. The strategy developed in this paper combines non-supervised learning of AVGSOM and the information available of the problem in order to reduce the dimension of rule antecedent.
The proposed method has been applied to three sets of data obtained from UCI repository with really satisfactory results. In particular, the results obtained in the Iris data base classification have been compared with other existing methods showing the supremacy of the new proposed method.
Finally, some of the conclusions as well as some future lines of work are presented.
Key Words: Clustering Methods, Self-organizing Maps, Neural Networks, Rule Extraction, Data Mining.

Resumen

La obtención de conocimiento a partir de la información existente es un proceso no trivial que consiste en identificar patrones válidos, novedosos, potencialmente útiles y comprensibles a partir de los datos disponibles. La Minería de Datos es el área de la Informática referida a la aplicación de diferentes métodos para la obtención de patrones y modelos.
Una de las soluciones más utilizadas se basa en estrategias adaptativas no supervisadas que permitan clasificar la información disponible. En esta dirección, las redes neuronales competitivas dinámicas han demostrado ser capaces de brindar buenos resultados. Sin embargo, su aplicación en el área de la Minería de Datos se encuentra limitada por su funcionamiento tipo "caja negra" donde resulta complejo justificar el conocimiento adquirido.
Este artículo propone una nueva estrategia para obtener reglas de clasificación a partir de una red neuronal competitiva dinámica entrenada con el método AVGSOM. Dicho método ha sido seleccionado en base a su capacidad para preservar la topología de los datos de entrada, característica fundamental para obtener los hipercubos iniciales adecuados. La estrategia desarrollada en este trabajo combina la capacidad del aprendizaje no supervisado del AVGSOM con información disponible del problema para reducir la dimensión del antecedente de las reglas.
El método propuesto ha sido aplicado a tres conjuntos de datos obtenidos del repositorio UCI con resultados muy satisfactorios. En particular, los resultados obtenidos en la clasificación de la base de datos Iris han sido comparados con otros métodos existentes mostrando la superioridad del nuevo método propuesto.
Finalmente se presentan algunas conclusiones así como algunas líneas de trabajo futuras.

Palabras Claves: Métodos de Agrupamiento, Mapas Auto-organizativos, Redes Neuronales, Extracción de Reglas, Minería de Datos.

[1] Becario de estudio CIC – III-LIDI – Facultad de Informática - UNLP
[2] Profesor Titular DE – III-LIDI – Facultad de Informática - UNLP

1. Introduction

At present, Computer Sciences advances allow us to cope with large volume of data obtained as results of observation and / or different process sampling. There exist technical differences which allow obtaining knowledge from the stored data. These processes, known as “Data Mining”, allow extracting useful and understandable knowledge - previously unknown -, from large quantities of data stored in different formats.

So as to make data mining processes efficient, their total or partial automation should be accomplished in order to reuse patterns found to generate new knowledge and allow "perfectible" decision making, taking into account the knowledge obtained from the data stored and previous decisions and their effects.

One of the most accepted approaches in data mining is that which pursues the extraction of knowledge in a non-directed and non-supervised fashion. In these cases, cluster detection techniques are a really useful tool both for detecting interesting patterns within a set and for determining relations among them.

Competitive neural networks with non-supervised training are one of the mostly used tools to solve clustering problems not requiring - for their learning - knowledge on isolated solutions of the problem. Within this category, Self-organizing Maps (SOM) [13] have proved to be capable of learning the input data organization allowing obtaining a structure which respects its topology.

However, SOM and other similar networks present two important limitations. In first place, the dimension and structure of the network should be defined a priori, before starting the training, thus conditioning the results and efficacy of the response obtained. In second place, the network capacity is defined by the number of nodes that it has as well as by the learning parameters.

Dynamic self-organizing maps search for the solution of these problems. Among the different existing methods proposed for defining the architecture, we can see that the incorporation of elements is varied, finding neural networks that add them in an isolated manner and others that add complete layers.

The application of a Neural Network into the solution of Data Mining problems allows exploiting to the maximum its capacity in order to generalize the available data with an excellent noise tolerance, though its weakness at the time of explaining and justifying the acquired knowledge [2] [4] should not be disregarded. One of the most important disadvantages of the NN application in these topics is rooted in its “black-box” functioning, in which it is hard to justify the acquired knowledge. Looking for an answer for this problem, methods for extracting rules from feedforward networks [15] [17] [18], decision trees [14] [16], based on genetic algorithms [11], and even competitive neural networks like SOM [3] [8] [12] have been defined.

This paper proposes a new strategy for obtaining *classification rules* from a dynamic competitive neural network trained with the AVGSOM method. Such method has been selected for its capacity of preserving input data topology, essential characteristic necessary to obtain the proper initial hypercubes. The strategy developed in this paper combines the capacity of AVGSOM’s non-supervised learning and the information available of the problem in order to reduce the dimension of rules antecedent.

The next sections are organized as follows. In the first place, Section 2 presents a brief description of the general characteristics of dynamic self-organizing maps, which will allow us to understand AVGSOM’s method training algorithm detailed in Section 3. The analysis with the results obtained with this method can be viewed in [9]. Section 4 describes the rule extraction process. Finally, some of the results are analyzed and some future lines of work are presented.

2. Dynamic Self-Organizing Maps

In general, most of the existing strategies for defining Dynamic self-organizing maps present the following characteristics [7]:

- The network structure is a graph made up by interconnected competitive neurons. The connection is regular and plays an important role at the time of visualizing the reduction of input space.
- Each network neuron corresponds to a prototype vector, which aims at representing a set of similar input data set. The similarity measure to be used depends on the problem.
- Training is carried out through a competitive process in which neurons aim at representing input data. For each datum, its resemblance with the prototype vector of each neuron is evaluated, considering the winner as the most alike. Adaptation is mainly applied to the wining neuron and, to a lesser extent, to its closest surrounding. This is what allows us to gradually correct the structure so as to preserve the topology.
- In each step of the adaptation, local error data is stored in the winner neuron. This aims at avoiding that a same element of the network stores the representation of most of the input patterns. The error calculation depends on the application.
- The stored error information is used to determine where new units should be inserted in the network. When an addition is carried out, the error information is locally redistributed thus avoiding new additions in a same place.

We can see in the previous list that the first three characteristics correspond to the SOM defined by Kohonen [13], while the last two are related to the need of identifying the place in which new elements are to be added in the architecture.

In addition, the need of modifying - during the training - the quantity of network elements leads us to represent separately, and for each neuron, the corresponding prototype vector and its closest neighbors. This does not happen in SOM, since the quantity of neurons in the structure and the way they are interconnected are defined a priori and, thus, it only remains to identify the corresponding prototype vector values.

There exist several solutions for determining the architecture of a dynamic competitive neural network [1] [5] [6], whose main differences are rooted in the way neighbor neurons are connected and the strategy used for inserting new elements. In this paper, we have used AVGSOM [9], whose performance is detailed in the next section.

3. AVGSOM

This method makes use of a rectangular grid in which each neuron has a maximum of four neighbors. Training begins with a minimum structure of four neurons, in which each has two neighbors, making up a matrix of 2x2 and the corresponding prototype vectors are initialized at random. At each iteration step, the prototype vectors are adapted, and the error is stored in the winning neuron, as usual.

The fact that a neuron overpasses, during the training, the stored error threshold established a priori, points out the need of adapting the structure in order to avoid the excessive accumulation of patterns around the winner. Any information on the structure should respect the initially proposed rectangular topology, since this eases the visualization of input data.

For such reason, if the winner has four neighbors, the difference between the winner and its closest and direct neighbors should be reduced, thus allowing other elements of the network to win the

contest during the successive adaptive steps.

If the winner has less than four direct neighbors, a new element will be inserted in the structure, whose initial weight vector will be close to that of their neighbors so as to not distort the network topology during the learning.

The method used for determining the weight vector of a new neuron in AVGSOM is the average of the prototype vectors of those that will be neighbors of the new neuron. In this way, not only the elements of the network relate to each other within a graph but also the weight vector of the new neuron lies "in the middle" of its neighbor weight vectors in the input space, preserving the data topology almost totally.

Next we will present the training algorithm used by AVGSOM.

3.1. Algorithm

Begin with a neural network made up by four neurons in which each has two neighbors.

Initialize, with random values, the prototype vectors corresponding to each Network neuron.

Repeat

For each input pattern

- Enter the pattern to the neural network
- Identify the winner neuron and adapt its weight vector and those of its neighbors, as SOM usually does.
- Store in the winner neuron the magnitude of the corresponding error. Such value corresponds to the similarity measure evaluated in the previous point.

End For

For each network neuron

- If the neuron error surpasses the *GT* threshold then
  - If the neuron has four direct neighbors then
    - Distribute the error stored by the winner neuron among its direct neighbors. With this, in the next iterations, the neighbors will have more opportunities for saturating themselves, and in this way “push” the error towards the limits of the network, thus achieving its expansion.

  Otherwise
    - Select at random one of the free spaces to generate the new neighbor neuron.
    - The prototype vector corresponding to this new neuron is computed as the average of the prototype vectors of those which will be its neighbors.
    - Assign zero as stored error for this new neuron.
    - Assign the zero value to the error stored by this neuron.
- If this neuron never wins then
  - Increase by 1 its failure counter.
- If the failure counter of this neuron reaches a pre-established threshold then it should be eliminated.

End For

Until no new neurons are created or the growing rate is minimum.

GT threshold is computed as $GT = -D * ln(SF)$, *SF* being a value between 0 and 1 corresponding to the dispersion factor, indicated as parameter [1].

A complete description of the method can be found in [9].

4. Classification Rule Extraction from AVGSOM

From a dynamic, competitive neural network trained with the AVSGOM method, we will obtain the initial hypercube's centroids. If L is the set of clusters represented by the weight vectors of the AVGSOM's trained dynamic neural network competitive neurons, the interval corresponding to feature *z* for cluster *k* is indicated as follows:

$$H_{kz} = (min_{kz}, max_{kz}) \quad \forall c_k \in L, \ \forall z \in 1..m$$

where min_{kz} and max_{kz} contain the minimum and maximum values found in the feature *z* of the training patterns corresponding to cluster *k*, respectively.

If we wish to obtain classification rules, it will be necessary to count with additional information that allows us to associate each hypercube to a certain class. In this way, belonging to a given hypercube will mean belonging to a given class. This is represented by rules such as:

IF (*condition*) THEN *belong to class_j*

where the association between a hypercube (cluster) and a class to which the elements represented by it belong is externally (supervised) established. It is important to notice that otherwise, rules will just only be categorizations of the groups, incapable of making reference to a specific class.

The method here proposed allows obtaining an ordered sequence of rules, in which each of them identifies elements of a class. The order of rules within a sequence is determined by its precision, being the first that with the most precise classification.

In order to establish this order it is necessary to identify the clusters with lesser overlapping. d_{kjz} will be used to refer to the overlapping between clusters c_k and c_j belonging to L for the feature *z* and it will be computed as follows:

$$minmax = \min(\max(H_{kz}), \max(H_{jz}))$$
$$maxmin = \max(\min(H_{kz}), \min(H_{jz}))$$
$$d_{kjz} = minmax - maxmin \quad , \ \forall z \in 1..m; \ \forall j,k \in 1..K \text{ con } j \neq k$$

where min and max represent the common minimum and maximum functions.

Note that the value of d_{kjz} is proportional to the overlapping between clusters. Its value will be less than zero when there does not exist overlapping between clusters c_k and c_j for the feature *z*, while a positive value will indicate the opposite.

For instance, if cluster c_k verifies for any feature *z*,

$$\#\{d_{kjz} < 0 / \ c_j \in L, \ j \in [1..K], j \neq k\} = (K-1)$$

this cluster does not represent overlapping with any of the remainings for this feature and, thus, it will be sufficient to incorporate rule

$$\text{« If (Feature\_z} \in [min(H_{kz}), max(H_{kz})]) \text{ then it\_is\_of\_cluster\_k »}$$

in order to separate it from the rest.

In those cases in which a feature with these characteristics cannot be obtained, the rule antecedent will be expressed as a set of conditions, in which each will be as:

$$(\text{Feature\_x} \in [min(H_{kx}), max(H_{kx})])$$

Each rule is made up from the intervals established for each feature of each cluster. Thus, it is necessary to select the cluster in order to write the rule.

The selection should prioritize clusters with less overlapping in order reduce the error in the classification. Such selection is carried out as follows:

$priority_{kz} = \#\{d_{kjz}<0 \ / c_j \in L,\ j\in[1..K],\ j\neq k\ \} \quad \forall k\in[1..K],\ \forall z\in[1..m]$

$maximum = max(priority_{kz})\ ,\ c_k\in L,\ \forall k\in[1..K],\ \forall z\in[1..m]$

$T = \{c_t \ / \ priority_{tz} = maximum,\ c_t\in L,\ \exists z\in[1..m]\}$

$Intersec_{kz} = sum(d_{kjz})\ con\ d_{kjz}>0,\ c_k\in T,\ j\in[1..K],\ j\neq k,\ \forall z\in[1..m]$

$Obtain\ t\ such\ that\ (Intersec_{tz} = min(Intersec_{kz}))\ con\ c_k,\ c_t \in T, z\in[1..m]$
$and\ (priority_{tz} = maximum)$

where:

- $priority_{kz}$ measures the *k*-th cluster overlapping with respect to the rest and its value will be given by the quantity of clusters which do not overlap with it for feature *z*.
- *maximum* is a whole value representing a higher level for the quantity of clusters from which a given cluster can be separated.
- *T* is the set of clusters which have at least a feature allowing them to separate themselves from the maximum number of possible clusters.
- $Intersec_{kz}$ is a positive value proportional to the overlapping the *k*-th cluster presents with the rest. The objective lies in establishing a second selection criterion.

The rule with at least a feature allowing its separation from the maximum number of clusters, indicated by *maximum*, will be selected. If there exists more than one fulfilling this condition, that with the least overlapping value with respect to the rest will be selected. This last is represented by *Intersec*.

The generated rules are incorporated in the *Rules* array. Once this process is finished, and in order to make a classification, these rules should be run in order, beginning by the array index 1 until the corresponding class is found. This mechanism implies that the used training patterns should properly characterize the input space; otherwise, it will be necessary to incorporate representatives so as to establish a "*reject*" class.

Figure 1 presents the pseudo-code corresponding to the algorithm here described.

```
P = {w_j / w_j∈RN input with AVGSOM and w_j maps patterns of a single cluster}
L = {clusters represented by vectors w_j∈P}
Obtain H_kz = (min_kz,max_kz)   ∀c_k∈L, ∀z∈1..m
i=0
Repeat
   minmax = min(max(H_kz), max(H_jz))
   maxmin = max(min(H_kz), min(H_jz))
   d_kjz = minmax – maxmin    , ∀z∈1..m; ∀j,k∈1..K con j≠k
   priority_kz = #{d_kjz<0 /c_j ∈ L, j∈[1..K], j≠k }   ∀k∈[1..K], ∀z∈[1..m]
   maximum = max(priority_kz) , c_k∈L, ∀k∈[1..K], ∀z∈[1..m]
   T = {c_t / priority_tz = maximum, c_t∈L, ∃z∈[1..m]}
   Intersec_kz = sum(d_kjz) con d_kjz>0, c_k∈T, j∈[1..K], j≠k, ∀z∈[1..m]
   Obtain t such that (Intersec_tz = min(Intersec_kz)) con c_k, c_t ∈ T,z∈[1..m]
                      y (priority_tz = maximum)
   Rules[++i] = Generate_rule(t, priority_t, Intersec_t, d_t, H)
   L = L - {c_t}
Until ( #L=1 )
```

Figure 1. Cluster selection algorithm for building up rules

Once the cluster for which the rule is to be built up has been chosen, it is necessary to analyze its interval features considering first the most significant, i.e., those allowing its separation from most of the remaining clusters. If there exist several features with the same importance, that presenting the lesser overlapping with the rest of the clusters should be selected, i.e., that having the lesser value in *Intersec*.

The selected feature interval will be used to build up the rule antecedent condition. It is important to notice that between the clusters to be separated, overlapping may occur. The process Adjust_Limit aims at minimizing it, adjusting the corresponding interval.

Then, features are selected until the indicated cluster is separated from the rest. Figure 2 sums up the way to obtain such rule. Notice that, when the rule is incorporated, the Class(t) process allows obtaining the class represented by cluster t.

```
function Generate_Rule(t, priority_t, Intersec_t, d_t, var H)
   Better = {s / priority_ts = max(priority_tz) con ∀z∈1..m}
   Obtain r such that (Intersec_tr = min(Intersec_tz))with z∈Better
   Remaining_Clusters= L – {c_t}
   Adjust_Limits(t, r, Remaining_Clusters, d_t, H)
   Condition = (feature_r ∈ [min(H_tr), max(H_tr)]
   Remaining_Clusters = Remaining_Clusters – {c_j / d_tjr<0, c_j ∈ L, j≠t }
   Remaining_Feat= {1..m} – {r}
   While (#Remaining_Clusters> 0)
      Priority_tz = #{d_tjz<0, c_j ∈ Remaining_Clusters, z ∈ Remaining_Feat }
      Intersec_tz= sum(d_tjz) con d_tjz>0, c_j∈ Remaining_Clusters,
                                  z∈ Remaining _Feat
      Better= {s / priority_ts = max(priority_tz ) con ∀z∈ Remaining _Feat}
      Obtain r such that(Intersec_tr = min(Intersec_tz))with z∈Better
      Adjust_Limits (t,r, Remaining_Clusters, d_t, H)
      Condition = Condition AND (feature_r ∈ [min(H_tr), max(H_tr)]
         Remaining_Clusters = Remaining_Clusters – { c_j/d_tjr<0, c_j∈
                              Remaining_Clusters, j≠t }
      Remaining _Feat = Remaining _Feat – {r}
   end While
   return(« if » + condition+ «then is_of_class » + Class(t))
end process
```

Figure 2. Generation of the classification rule for cluster t

The adjustment of the interval corresponding to the feature used to build up the condition may be carried out in different ways. A simple solution would be taking as the interval the mean point separating the centroids of each cluster. However, in the code indicated in Figure 3, the mean and variance of each of the clusters have been used in order to minimize the quantity of patterns wrongly classified.

Function *dist*, used in Figure 3, represents a distance measure between the cluster centroids receiving it as parameter.

```
process Adjust_Limits(t, r, Remaining_Clusters, d_t, var H)
    Overlapped = {c_j / d_tjr>0, c_j ∈ Remaining_Clusters }
    For each c_j ∈ Overlapped
        s = [dist(μ(c_j), μ(c_t)) - σ(c_j)- σ(c_t)]/2
        If μ(c_t)< μ(c_j)then
            H_tr = (min(H_tr), μ(c_t)+σ(c_t)+s)
            H_jr = (μ(c_j)-σ(c_j)-s, max(H_jr))
        otherwise
            H_tr = (μ(c_t)+σ(c_t)+s, max(H_tr))
            H_jr = (min(H_jr), μ(c_j)-σ(c_j)-s)
        end if
end process
```

Figure 3. Adjustment of r feature interval of cluster t

5. Results Obtained

The method for building up rules presented in this paper was used for classifying three sets of data of the UCI repository [10]:

- *Iris Plants Database*
  It is a data base which contains information on different types of the iris plant. In it, there appear three different classes represented by 50 patterns each. For each data, four features are presented, with the width and height of the petal and sepal measured in centimeters. It presents the peculiarity that only one of the classes is linearly separable.
- *Wine Recognition Database*
  These data are the result of a viticultural analysis in a same region of Italy, applied to three different types of wines, which allows identifying their origin. It is composed by 178 patterns of 13 continuous features grouped in three classes.
- *Glass Identification Database*
  This data base contains information on seven types of different glasses. It has 214 input patterns and each pattern is represented by nine features: refraction index, and the percentage of occurrence of eight metals (sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron).

Table 1 shows the results obtained for each set of data. In each case, both the quantity of input patterns and the quantity of patterns properly classified from the rules obtained have been indicated. Two tests were carried out; the first one used the complete set of input data for training AVGSOM and the same set of data for testing the rules. In the second test, two thirds of the set of input data were selected at random in order to carry out the training, using the remaining third for testing the rules obtained.

As it can be seen, the first two sets of data provide really satisfactory results, while the set of glass identification data (*Glass Identification Database*) found incorrect classifications for some classes. In particular, classes 1 and 3 present a strong overlap in all of the features values, thus impeding its proper classification.

There exists previous work which provides strategies to obtain classification rules on the Iris base [15][18]. In both cases, a basic method and its corresponding improvement are posed. The results obtained are based on the exactness degree of data pattern classification with the rules obtained.

Table 2 was build up with the results of the application of the method proposed in this paper, and the best solutions of each of the mentioned papers [15][18].

Finally, Table 3 allows comparing the set of rules obtained in each case.
As it can be seen, the effectiveness of the method here proposed is superior to the existing.

Table 1. Classification of three sets of data of UCI repository using the proposed method.

Data bases and their classes	Training with the complete set		Training with 2/3 of the set			
			Training		Test	
	bc / N	%	bc / N	%	bc / N	%
Iris Plants Database						
Iris versicolor	48/50	96%	27 / 28	96,43%	21/22	95,45%
Iris virginica	50/50	100%	38/38	100%	12/12	100%
Iris setosa	50/50	100%	34/34	100%	16/16	100%
Total	**148/150**	**98,67%**	**99/100**	**99%**	**49/50**	**98%**
Wine recognition data						
A	53/59	90%	38/38	100%	21/21	100%
B	68/71	96%	46/51	90;20%	16/20	80%
C	48/48	100%	27/29	93,10%	16/19	84,21%
Total	**169/178**	**94,94%**	**111/118**	**94,06%**	**53/60**	**83,33%**
Glass Identification Database						
1	63/70	90%	33/46	71,74%	21/24	87,5%
2	75/76	98,68%	49/52	94,23%	20/24	83,33%
3	7/17	41,18%	5/11	45,45%	2/6	33,33%
5	12/13	92,31%	9/10	90%	3/3	100%
6	7/9	77,78%	4/4	100%	5/5	100%
7	29/29	100%	19/19	100%	10/10	100%
Total	**193/214**	**90,19%**	**119/142**	**83,8%**	**61/72**	**84,7%**

Table2. Comparison of the method proposed in this paper with the already existing ones (NeuroRule [15] y Full-RE [18])

Method	Reliability	
AVGSOM rule extraction	148/150	98,67%
NeuroRule	73/75	97,33%
Full-RE	146/150	97,33%

Table 3. Rules extracted from the Iris Plants Database set by the method presented in this paper and the NeuroRule [15] and Full-RE [18] methods.

Method	Rules Extracted
AVGSOM rule extraction	If Petal-length $\leq$ 1.9 then *Iris setosa* Otherwise If Petal-width $\leq$ 1.64 then *Iris versicolor* Otherwise *Iris virginica*
NeuroRule	If Petal-length $\leq$ 1.9, then *Iris setosa* If Petal-length < 4.9 and Petal-width < 1.6, then *Iris versicolor* Default Rule: *Iris virginica*
Full-RE	If Petal-length $\leq$ 2.1 then *Iris setosa* If Petal- length $\leq$ 5.1 and Petal- width $\leq$ 1.7 then *Iris-versicolor* If Petal-length $\geq$ 4.8 then *Iris-virginica*

6. Conclusions and Future Lines of Work

We have presented a new method for extracting classification rules from a dynamic competitive neural network trained with AVGSOM. This training strategy presents the peculiarity of properly representing the input data topology, which allows establishing the space of each cluster constituting a starting point adequate for determining classification rules.

The application of this new method in three sets of data of the UCI repository has been satisfactory. Its comparison with other existing methods- which also generate rules - shows that it has a higher reliability.

However, one of the main drawbacks presented by this method is found in the resolution of problems with a high number of classes. This is due to the behavior of process *Adjust_Limits*, which contracts each classification area - which has the form of a hypercube looking for the minimization of overlappings. For such reason, if classes are not compact enough, it will not be possible to represent them through a feature interval array. So as to solve this, we are currently working on the definition of classes from several hypercubes, which will allow solving more complex problems.

References

[1] Alahakoon, D., Halgamuge, S.K. and Srinivasan, B. Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery. *IEEE Transactions On Neural Networks*. 11 (3): 601-614. 2000.

[2] Andrews, R., Diederich, J. and Tickle, A. B. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Bases System*. 8(6): 373-389. 1995.

[3] Azcarraga, A. P., Hsieh, M.-H., Pan, S. L. and Setiono, R. Extracting Salient Dimensions for Automatic SOM Labeling. *IEEE Transactions on Systems, Man, and Cybernetics—Part C*. 35(4): 595-600. 2005.

[4] Fu, L. M. Knowledge-based connectionism for revising domain theories**.** *IEEE Transactions on Systems, Man and Cybernetics*. 23(1): 173-182. 1993.

[5] Fritzke, B. Growing cell structures: A self organizing network for supervised and un-supervised learning. *Neural networks*. 7, pp. 1441-1460. 1994.

[6] Fritzke, B. A growing neural Gas Network Learns Topologies In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems.* 7: 625-632. MIT Press, Cambridge MA, 1995.

[7] Fritzke, B. Growing Self-organizing Networks – Why?. In: M. Verleysen (ed.), *ESANN'96: European Symposium on Artificial Neural Networks*, D-Facto Publishers, Brussels, pp. 61-72. 1996.

[8] Hadzic, F. and Dillon, T. S. CSOM: Self-Organizing Map for Continuous Data. *3rd IEEE International Conference on Industrial Informatics*. 2005.

[9] Hasperué W. and Lanzarini L. Dynamic Self-Organizing Maps. A new strategy to upgrade topology preservation. *XXXI Congreso Latinoamericano de Informática CLEI 2005*. Cali. Colombia. Octubre 2005. pp.1081-1087.

[10] Hettich, S., Blake, C.L. and Merz, C.J. UCI Repository of machine learning databases. *University of California, Irvine, Dept. of Information and Computer Sciences*. URL http://www.ics.uci.edu/~mlearn/MLRepository.html. 1998.

[11] Hruschka, E.R. and Ebecken, N.F.F. Using a clustering genetic algorithm for rule extraction from artificial neural networks. *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*. 11-13, pp. 199 – 206. 2000.

[12] Hsu, C-C. and Wang, S.-H. An Integrated Framework for Visualized and Exploratory Pattern Discovery in Mixed Data. *IEEE Transactions on Knowledge and Data Engineering*, 18(2): 161-173. 2006.

[13] Kohonen, T. *Self-Organizing Maps*. 2nd Edition. Springer. ISSN 0720-678X. 1997.

[14] Sato, M. and Tsukimoto, H. Rule extraction from neural networks via decision tree induction**.** *Proceedings. IJCNN '01. International Joint Conference on Neural Networks*. Vol. 3, July 15-19. pp. 1870 – 1875. 2001.

[15] Setiono, R. and Liu, H.; Symbolic representation of neural network. *Computer*. 29(3): 71-77. 1996.

[16] Setiono R. and Leow W. Generating Rules from trained Network using Fast Pruning. *International Joint Conference on Neural Networks*. Vol. 6, July 10-16, pp. 4095 – 4098. 1999.

[17] Setiono, R. Extracting M-of-N Rules from Trained Neural Networks. *IEEE Transactions on Neural Networks*. 11(2): 512-519. 2000.

[18] Taha, I.A. and Ghosh J. Symbolic Interpretation of Artificial Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*. 11(3): 448-463. 1999.