SOM+PSO
A Novel Method to Obtain Classification Rules

Laura Lanzarini, Augusto Villa Monte and Franco Ronchetti
Instituto de Investigación en Informática LIDI (III-LIDI)
Facultad de Informática, Universidad Nacional de La Plata (UNLP)
Calle 50 y 120, S/N (CP 1900). La Plata, Buenos Aires, Argentina
{laural,avillamonte,fronchetti}@lidi.info.unlp.edu.ar

ABSTRACT

Currently, most processes have a volume of historical information that makes its manual processing difficult. Data mining, one of the most significant stages in the Knowledge Discovery in Databases (KDD) process, has a set of techniques capable of modeling and summarizing these historical data, making it easier to understand them and helping the decision making process in future situations. This article presents a new data mining adaptive technique called SOM+PSO that can build, from the available information, a reduced set of simple classification rules from which the most significant relations between the features recorded can be derived. These rules operate both on numeric and nominal attributes, and they are built by combining a variation of a population metaheuristic and a competitive neural network. The method proposed was compared with the PART method and measured over 19 databases (mostly from the UCI repository), and satisfactory results were obtained.

Keywords—Classification Rules, Data Mining, Adaptive Strategies, Particle Swarm Optimization, Self-Organizing Maps.

1. INTRODUCTION

Data mining is a research field that in recent years has gained attention from various sectors. Government employees, business people and academics alike, for very different reasons, have contributed to the development of various techniques that can summarize the information that is available. This is one of the most important stages in the Knowledge Discovery in Databases (KDD) process, and it is characterized for producing useful and novel information without any prior hypotheses. It encompasses a set of techniques capable of modeling available information and, even though there are different types of models, decision makers usually choose those that are self-explanatory. For this reason, rules, i.e., statements of the *IF condition1 THEN condition2* type, are preferred when characterizing that huge volume of historical data that were automatically saved.

There are different types of rules. An association rule is an expression whose conditions are conjunctions of propositions of the *attribute=value* type and whose only restriction is that the attributes included in the antecedent of the rule must not be part of its consequent. When the set of association rules has the same attribute in the consequent, it is said that this is a set of classification rules, while if they must be interpreted in the same order as they were obtained, they are considered as a decision list.

Unfortunately, most of the existing methods available to obtain rules include examples of database view with a set of rules that is so large and complex that, despite having the *IF-THEN* structure, it becomes almost unreadable. For this reason, a new method to obtain classification rules is proposed in this article, with two essential features: the cardinality of the set of rules obtained is low, and the antecedent of the rules that are generated is reduced. To this end, the method proposed combines an optimization technique responsible for directing the search towards the appropriate set of rules, and a neural network that allows assessing the significance of each of the attributes when defining the antecedent for the rule.

This paper is organized as follows: Section 2 lists some related articles, Sections 3 and 4 briefly describe the neural network and metaheuristic used, respectively, Section 5 details the method proposed, Section 6 presents the results obtained, and Section 7 presents a summary of the conclusions along with possible future work lines.

2. RELATED WORK

There are several methods for building rules. When it comes to obtaining association rules, the Apriori method [1] or some of its variants [2] can be used. The goal is to identify the most frequent sets of attributes and then combining them to obtain the rules. There are variations to this method, usually oriented to reducing calculation time [3].

If working with classification rules, the literature includes different tree-based methods for building them, such as C4.5 [4], or trimmed tree-based methods, such as PART [5]. In either case, it is essential that the set of rules obtained covers the examples with a preset error level. Tree-based rule building methods are partitive and based on various attribute metrics in order to assess their coverage ability.

This article presents a different approach, one that is based on the optimization achieved with particle swarms (PSO, Particle Swarm Optimization) to

determine the rules. Even though there are rule-generation methods that use PSO [6–13], when operating on nominal attributes the body of available examples should be large enough to cover all search space areas, which is not always feasible. The result is a poor initialization of the population, which in turn causes a premature convergence. As a way to solve this problem, and at the same time reducing generation time, the initial state is obtained from a SOM (Self-Organizing Map) competitive neural network.

The literature describes methods that optimize SOM with PSO and significantly reduce the calculation time for the training phase [14], or methods that use PSO to determine the optimal number of competitive neurons in the network, such as [15]. Unlike these papers, our proposal is using PSO to obtain the set of rules, and SOM to avoid the premature convergence of the population. Even though in this case the SOM network being used is static, it could be replaced by a dynamic competitive network such as the one defined in [16].

3. Self-Organizing Maps (SOM)

The SOM (Self-Organizing Maps) neural network was defined by Kohonen in 1982 [17]. Its main application is grouping all available information, and it is characterized by its ability to preserve input data topology.

It is a partitive clustering technique, since it associates each example to an average vector or centroid. It can also relate centroids in order to identify similarities among them. This is an exclusive feature of this type of networks; is not available in most centroid-based clustering techniques. For this reason, it is commonly used as visualization tool and to reduce the number of dimensions of the input space. It can be represented as a two-layer structure: the input layer, whose function is only to allow information to enter the network, and the competitive layer, which is responsible for the clustering task. The neurons that form this second layer are connected and have the ability of identifying the number of "hops" or connections that separate them from each of the remaining neurons in this level. Each competitive neuron is associated to a weight vector or centroid represented by the values of the arches that reach this neuron from the input layer. Thus, the SOM network handles two information structures: one in relation to the centroids linked to the competitive neurons, and another one responsible for determining proximity among neurons. This, unlike a "winner-take-all" style method such as the K-means method [18], provides additional information about the clusters, since those neurons that are close by within the architecture will represent similar clusters in the input data space.

Figure 1 shows the structure of a SOM network where the input layer is formed by an n-dimensional vector and the competitive layer has 4x5=20 neurons. Each neuron in this second layer has 4 direct neighbors (immediate connections). This connection

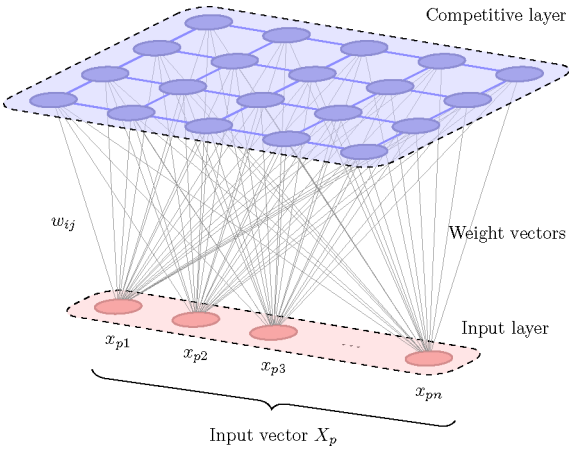

Fig. 1. Classic structure of a SOM network

pattern can change depending on the problem to solve.

Network weights, whose values are represented in the figure by matrix W, are initially random, but they adapt with the successive presentations of input vectors. w_{ij} is used to denote the weight of the arch going from the j^{th} input neuron to the i^{th} competitive neuron.

Since this is a competitive structure, each input vector is considered to be represented by (or associated with) the competitive neuron that has the most similar weight vector based on a given similarity measurement. The final value of W is obtained by means of an iterative process that is repeated until the weight vectors do not present any significant changes or, in other words, until each input vector is represented by the same competitive neuron than in the previous iteration.

During the training process of the SOM network, in each iteration, for each input vector $X_p = (x_{p1}, x_{p2}, ..., x_{pn})$, the representative neuron, that is, the most similar neuron so far, is identified. This neuron is called "winning neuron", since it is the one that "wins" the competition to represent the vector for being the closest one using a distance measurement. That is to say, that being $W_i = (w_{i1}, w_{i2}, ..., w_{in})$ the weight vector of the i^{th} competitive neuron, SOM identifies the winning neuron as the one that meets equation (1)

$$\|W_{win} - X\| = \min(\|W_i - X\|) \quad i = 1..M \quad (1)$$

where the winning neuron is the neuron that wins p vector representation, $\|.\|$ is a distance measurement (usually Euclidean distance), and M is the total number of competitive neurons. Then, the SOM updates only the weight vector for that neuron and its neighborhood following equation (2)

$$w_{ij} = w_{ij} + \alpha(x_{pj} - w_{ij}) \quad j = 1..n \quad (2)$$

where n is the input space dimension, i is the competitive neuron whose vector is being updated and α is a value between 0 and 1 that represents a

$W \leftarrow$ random initial values
$Neighborhood \leftarrow$ set the size if the initial neighborhood
$NoIteReduction \leftarrow$ set the number of iterations that must occur to reduce the neighborhood
while (termination criterion is not reached) **do**
  for (each input vector) **do**
    Input the vector to the network and calculate the winning neuron
    Update the winning neuron and its neighborhood
  end for
  Reduce $Neighborhood$ if applicable based on $NoIteReduction$
end while

Fig. 2. Basic training pseudocode for the SOM network

learning factor. The weight vectors of the remaining competitive neurons remain unchanged. Equation (2) has variations that can be consulted in [19].
The concept of neighborhood is used to allow the network to adapt correctly. This implies that neighboring competitive neurons represent similar input patterns. For this reason, the training process (obtaining W values) is started with a wide neighborhood that is then reduced as iterations occur. Figure 2 describes the pseudocode corresponding to the basic process for the adaptation of the SOM network.

4. Obtaining Classification Rules with PSO

Particle Swarm Optimization or PSO is a populational metaheuristic proposed by Kennedy and Eberhart [20] where each individual in the population, called particle, represents a possible solution to the problem and adapts by following three factors: its knowledge of the environment (its fitness value), its historical knowledge or previous experiences (its memory), and the historical knowledge or previous experiences of the individuals in its neighborhood (its social knowledge).
PSO was originally defined to work on continuous spaces, so a few considerations should be taken into account when working on discrete spaces. For this reason, Kennedy and Eberhart defined in [21] a new binary version of the PSO method. On of the key problems of this last method is its difficulty to change from 0 to 1 and from 1 to 0 once it has stabilized. This has resulted in different versions of binary PSO that seek to improve its exploratory capacity. In particular, the variation defined by Lanzarini *et al.* [22] will be used in this article.
Using PSO to generate classification rules that can operate on nominal and numerical attributes requires a combination of the methods mentioned above, since the attributes that will be part of the antecedent (discrete) have to be selected and the value or range of values they can take (continuous) has to be determined.
Since this is a populational technique, the required information has to be analyzed for each individual in the population. A decision has to be made between representing a single rule or the entire set for each individual, and the representation scheme has to be selected for each rule. Given the objectives proposed for this work, the Iterative Rule Learning (IRL) [23] approach was followed, where each individual represents a single rule and the solution to the problem is built from the best individuals obtained after a sequence of runs. Using this approach implies that the populational technique will be applied iteratively until achieving the desired coverage and obtaining a single rule in each iteration: the best individual in the population. Additionally, a fixed-length representation was chosen, where only the antecedent of the rule will be coded and, given the approach adopted, an iterative process will be carried out to associate all individuals in the population to a preset class, which does not require consequent codification.
To move in an n-dimensional space, each particle p_i in the population is formed by:

- $pBin_i = (pBin_{i1}, pBin_{i2}, \ldots, pBin_{in})$ stores the current position of the particle.
- $v_{1i} = (v1_{i1}, v1_{i2}, \ldots, v1_{in})$ and $v2_i = (v2_{i1}, v2_{i2}, \ldots, v2_{in})$ are combined to determine the direction in which the particle will move.
- $pBestBin_i = (pBestBin_{i1}, \ldots, pBestBin_{in})$ stores the best solution found for the particle so far.
- $fitness_i$ is the fitness value for the individual.
- $fitness\_pBest_i$ is the fitness value for the best local solution found ($pBestBin_i$ vector).
- $pReal_i = (pReal_{i1}, pReal_{i2}, ..., pReal_{in})$ is used only for numerical attributes and it contains the current boundaries of the intervals.
- $v3_i = (v3_{i1}, v3_{i2}, ..., v3_{in})$ indicates the change direction of $pReal_i$.
- $pBestReal_i = (pBestReal_{i1}, ..., pBestReal_{in})$ stores the best solution found for the particle within interval boundaries.

Every time the i^{th} particle moves, its current position and the intervals corresponding to the numerical attributes are changed as follows:

Binary part

$$v1_{ij}(t+1) = w_{bin}.v1_{ij}(t) + \varphi 1.rand_1.(2.pBestBin_{ij} - 1) + \varphi 2.rand_2.(2.localBestBin_{ij} - 1) \quad (3)$$

where w_{bin} represents the inertia factor, $rand_1$ and $rand_2$ are random values with uniform distribution in [0,1], and $\varphi 1$ and $\varphi 2$ are constant values that indicate the significance assigned to the respective solutions previously found. $pBin_{ij}$ and $localBestBin_{ij}$ correspond to the j^{th} digit in the binary vectors $pBestBin_i$ and $localBestBin_i$, respectively. With the method proposed, each particle will take into

account the position of its closest neighbor whose fitness value is higher than its own; therefore, the value of $localBestBin_i$ corresponds to the $pBin_k$ vector of the particle that is closest to $pBin_i$ provided that $fitness_k$ is higher than $fitness_i$ using the Euclidean distance.
It should be noted that, unlike the Binary PSO method described in [21], the movement of vector $v1_i$ in the directions corresponding to the best solution found by the particle and the best local value do not depend on the current position of the particle, as indicated in [22]. Then, each element of the velocity vector $v1_i$ is controlled by applying (4)

$$v1_{ij}(t) = \begin{cases} \delta 1_j & if\ v1_{ij}(t) > \delta 1_j \\ -\delta 1_j & if\ v1_{ij}(t) \leq \delta 1_j \\ v1_{ij}(t) & otherwise \end{cases} \quad (4)$$

where

$$\delta 1_j = \frac{limit1_{upper_j} - limit1_{lower_j}}{2} \quad (5)$$

That is, velocity vector $v1_i$ is calculated with (3) and controlled with (4). Its value is used to update the value of velocity vector $v2_i$, as shown in (6).

$$v2_{ij}(t+1) = v2_{ij}(t) + v1_{ij}(t+1) \quad (6)$$

Vector $v2_i$ is also controlled as vector $v1_i$ by changing $limit1_{upper_j}$ and $limit1_{lower_j}$ by $limit2_{upper_j}$ and $limit2_{lower_j}$, respectively. This will yield $\delta 2_j$, which will be used as in (4) to limit the values of $v2_i$. Then, sigmoid function (7) is applied and the new position of the particle is calculated with (8).

$$sig(x) = \frac{1}{1+e^{-x}} \quad (7)$$

$$pBin_{ij}(t+1) = \begin{cases} 1 & rand_{ij} < sig(v2_{ij}(t+1)) \\ 0 & if\ no \end{cases} \quad (8)$$

Continuous part

$$pReal_{ij}(t+1) = pReal_{ij}(t) + v3_{ij}(t+1) \quad (9)$$

$$\begin{aligned} v3_{ij}(t+1) = {} & w_{real}.v3_{ij}(t) \\ & +\varphi 3.rand_3.(pBestReal_{ij} - pReal_{ij}) \\ & +\varphi 4.rand_4.(localBestReal_{ij} - pReal_{ij}) \end{aligned} \quad (10)$$

where, once again, w_{real} represents the inertia factor, $rand_3$ and $rand_4$ are random values with uniform distribution in [0,1], and $\varphi 3$ and $\varphi 4$ are constant values that indicate the significance assigned to the respective solutions previously found. In this case, $localBestReal_i$ corresponds to the $pReal_k$ vector of the particle that is closest to $pReal_i$, where $fitness_k$ is greater than $fitness_i$ using Euclidean distance. This is the same particle from which vector $pBin_i$ was taken to adjust $v1_i$ in (3). The values assigned to w_{real} [24], $\varphi 1$, $\varphi 2$, $\varphi 3$ and $\varphi 4$ are important to ensure algorithm convergence. More detailed information on how to select these values can be found in [20] and [25]. Additionally, it should be mentioned that the incorporation of sigmoid function (7) radically changes the way in which the velocity vector is used to update the position of the particle. In continuous PSO, the velocity vector takes on higher values first to facilitate the exploration of the solution space, and then reduces them to allow the particle to stabilize. In binary PSO, the opposite procedure is applied. Extreme values, when mapped by the sigmoid function, produce similar probability values, close to 0 or 1, reducing the chance of change in particle values. On the other hand, velocity vector values that are close to zero increase the probability of change. Also, if the speed of a particle is the null vector, each of the binary digits that determine its position has a 0.5 probability of changing to 1. This is the most random situation that can occur.
In this paper, the values of $limit1$ and $limit2$ are the same for all dimensions, [0,1] and [0,6], respectively. Therefore, the values of velocity vectors $v1$ and $v2$ were limited to ranges [-0.5, 0.5] and [-3,3], respectively. This means that probabilities within the interval [0.0474, 0.9526] can be obtained. The values for $\varphi 1$, $\varphi 2$, $\varphi 3$ and $\varphi 4$ were established at 0.25, 0.25, 0.5 and 0.25, respectively. The values of w_{bin} and w_{real} were established between 1.25 and 0.25 in linearly and proportionally to the number of iterations executed, in ascending order for w_{bin} and descending order for w_{real}.
As regards the fitness of each individual, it depends on two aspects: rule relevance and rule size. The former is calculated based on rule support and confidence, while the latter is the proportion of attributes used in the antecedent versus the total number of attributes. This is indicated in equation (11).

$$\begin{aligned} Fitness = {} & PenalizationFactor * support \\ & *confidence - SizeFactor * size \end{aligned} \quad (11)$$

$Support$ and $confidence$ are the metrics that correspond to the rule representing the particle. $Support$ is the proportion of examples that comply with the rule, i.e., the number of examples for which both the antecedent and the consequent occur, divided by the total number of examples. $Confidence$ is the quotient of the number of examples that fulfill the rule and the number of those that only fulfill the antecedent.
The $PenalizationFactor$ is a value between 0.1 and 1 whose purpose is balancing the relation between $support$ and $confidence$. Those rules with a high $confidence$ but also representative $support$ are granted a high $fitness$ value. This is handles by means of a pair of limits that establish minimum and maximum values that are required to that effect. Penalization is maximal below the lower limit, it is linearly proportional in between limits, and it is

function PENALIZE ($value$, $L1$, $L2$, min, max)
  if $value < L1$ **then**
    $response = min$
  else if $value < L2$ **then**
    $Pos = (value - L1)/(L2 - L1)$
    $response = min + (max - min) * Pos$
  else $response = 1$
  end if
  return $response$
end function

s1 = max(2, 5% of the examples in the class)
s2 = max(4, 7.5% of the examples in the class)
PenaSup = Penalize(support, s1, s2, 0.1, 0.75)
c1 = lower confidence limit (e.g.: 0.4)
c2 = upper confidence limit (e.g.: 0.7)
PenaConf=Penalize(confidence, c1, c2, 0.1, 0.7)
PenalizationFactor=min(PenaSup, PenaConf)

Fig. 3. Pseudocode corresponding to rule penalization calculation

minimum above the upper limit. This is applicable both to $support$ and $confidence$, and the lowest of both calculated penalizations is used. Figure 3 contains pseudocode that represents how to obtain this factor.

In the second term of equation (11), $size$ is the quotient between the number of comparisons involved in the antecedent of the rule (one per attribute) and the maximum number of comparisons that could be present. $SizeFactor$ is simply a constant that quantifies the significance assigned to the length of the antecedent.

In brief, the purpose of the second term in equation (11) is reducing the fitness value of the individual as the number of attributes in the antecedent increases.

5. SOM+PSO. PROPOSED METHOD FOR OBTAINING RULES

Rules are obtained through an iterative process that analyzes non-covered examples in each class, starting with the largest classes (those with the highest number of representatives). Each time a rule is obtained, the examples that are correctly covered by the rule are removed from the input data set. The process continues until all examples are covered or until the number of non-covered examples in each class is below the corresponding established minimum support or until a maximum number of tries has been done to obtain a rule, whichever happens first. It should be noted that, since the examples are removed from the input data set as they are covered by the rules, the rules operate as a classification list. That is, in order to classify a new example, the rules must be applied in the order in which they were obtained, and the example will be classified with the class that corresponds to the consequent of the first rule whose antecedent is verified for the example at hand.

Before starting the iterative process for obtaining the rules, the method starts with the non-supervised training of a SOM network using the entire set of examples that are to be covered. The network is trained in a non-supervised fashion following the algorithm described in Section 3; the purpose of this training is identifying the most promising areas of the search space. Since the SOM network operates only with numerical data, nominal attributes are represented as binaries. Therefore, an attribute such as blood pressure, whose possible nominal values are "LOW", "MEDIUM" and "HIGH" must be transformed into a set of three binary attributes "Pressure = LOW", "Pressure = MEDIUM" and "Pressure = HIGH" and, for each record in the database, only one of them will have a value of 1 and the other two will have a value of 0. Also, before starting the training process, each dimension that corresponds to a numerical attribute is linearly escalated in [0,1]. The similarity measurement used is the Euclidean distance. Once training is finished, each centroid will contain approximately the average of the examples it represents. To obtain each of the rules, the class to which the consequent belongs is first determined. Seeking high-support rules, the method proposed will start by analyzing those classes with higher numbers of non-covered examples. The minimum support that any given rule has to meet is proportional to the number of non-covered examples in the class upon rule generation. That is, the minimum required support for each class decreases as iterations are run, as the examples in the corresponding class are gradually covered. Thus, it is to be expected that the first rules will have a greater support than the final ones.

After selecting the class, the consequent for the rule is determined. To obtain the antecedent, a swarm population will be optimized using the algorithm described in Section 4. This swarm will be initialized with the information of all centroids capable of representing a minimum number of examples from the selected class and its immediate neighbors. Centroid information is used to determine vector $v2$. In the case of nominal attributes, centroid information is linearly escalated to interval $[limit2_{lower_j}, limit2_{upper_j}]$, but in the case of numerical attributes, the value to be escalated is (1-1.5*$deviation_j$), $deviation_j$ being the j^{th} dimension of the deviation of the examples represented by the centroid. In both cases, the goal is to operate with a value between 0 and 1 that measures the participation level of the attribute (if numerical) or the attribute value (if nominal) when building the antecedent for the rule. In the case of nominal attributes, it is clear that the average indicates the proportion of elements represented by the centroid that match the same value, but in the case of numerical attributes, this proportion is not present in the centroid but in the deviation of the examples (always considering a specific dimension). If this deviation in any given dimension is zero, it means that all of the examples match the value in the centroid; on the other hand, if the deviation is too wide, the centroid would not be representative of the group and it would therefore not be convenient to include it in the antecedent for the rule. Using (1-1.5*$deviation_j$) if the deviation is high, the velocity value $v2$, argument of the sigmoid function, will be

Train SOM network using all training examples
Calculate the minimum support for each class
while (termination criterion is not reached) **do**
Choose the class with the highest number of non-covered examples
Build a reduced population of individuals from centroids
Evolve the population using PSO as seen in Section 4
Obtain R, the best rule for the population
if (R meets support and confidence requirements) **then**
Add the rule to the set of rules
Consider the examples classified by this rule as correctly covered
Recalculate the minimum support for this class
end if
end while

Fig. 4. Pseudocode of the method SOM+PSO

lower and the probability of using the attribute will be reduced. In all cases, velocity $v1$ is initialized randomly within $[limit1_{lower_j}, limit1_{upper_j}]$. Figure 4 corresponds to the pseudocode of the method proposed.

6. Results obtained

In this section, the performance obtained with the method proposed is compared against that of the PART method defined by Frank and Witten in [5] for generating classification rules for a known set of 17 databases of the UCI repository [26] and the Drug databases.

Thirty separate runs were performed for each method, and a SOM network with 30 neurons organized in 6 rows and 5 columns with 4 neighbors per neuron was used. The PART method was run with a confidence factor of 0.3 for trimming the tree and the default values for all remaining parameters. Tables I, II and III summarize the results obtained with both methods. In each case, not only the coverage accuracy of the set of rules was considered (Table I), but also the clarity of the model obtained, which is reflected in the average number of rules obtained (Table II) and the average number of terms used to form the antecedent (Table III). In each case, a two-tailed mean difference test with a significance level of 0.05 was carried out, where the null hypothesis establishes that the means are equal. Since each includes has 30 separate runs for each of the methods, the central limit theorem was used to assume a normal distribution for each sample. Based on the results obtained, when the difference is significant, in accordance to the level indicated, the best option was highlighted in bold in the table.

In average, considering the 8 cases in which the accuracy of PART is higher than that of the method proposed, the difference in accuracy in favor of PART is 4%, but the size of the set of rules is four

TABLE I
ACCURACY OF THE RULE SET OBTAINED WHEN APPLYING THE SOM+PSO AND PART METHODS

Database	SOM+PSO	PART
Balance scale	$0,713 \pm 0,01$	$\mathbf{0,815 \pm 0,02}$
Breast cancer	$\mathbf{0,708 \pm 0,02}$	$0,657 \pm 0,02$
Breast w	$0,945 \pm 0,00$	$\mathbf{0,955 \pm 0,00}$
credit_a	$\mathbf{0,859 \pm 0,01}$	$0,738 \pm 0,02$
Credit-g	$0,703 \pm 0,01$	$0,701 \pm 0,01$
Diabetes	$\mathbf{0,738 \pm 0,01}$	$0,731 \pm 0,01$
Heart-c	$0,721 \pm 0,02$	$\mathbf{0,764 \pm 0,02}$
Heart-statlog	$0,727 \pm 0,02$	$\mathbf{0,767 \pm 0,01}$
Iris	$0,924 \pm 0,02$	$\mathbf{0,939 \pm 0,02}$
Mushroom	$0,939 \pm 0,00$	$\mathbf{0,997 \pm 0,00}$
Promoters	$0,65 \pm 0,03$	$0,67 \pm 0,05$
Soybean	$\mathbf{0,856 \pm 0,01}$	$0,577 \pm 0,07$
Kr_vs_kp	$0,933 \pm 0,00$	$\mathbf{0,99 \pm 0,00}$
Zoo	$\mathbf{0,923 \pm 0,02}$	$0,187 \pm 0,05$
Slice	$\mathbf{0,812 \pm 0,01}$	$0,722 \pm 0,01$
Wine	$0,867 \pm 0,03$	$\mathbf{0,888 \pm 0,01}$
Drug5	$0,85 \pm 0,01$	$0,87 \pm 0,06$
DrugY	$\mathbf{0,844 \pm 0,02}$	$0,668 \pm 0,09$
Adult	$0,802 \pm 0,00$	$0,801 \pm 0,00$

TABLE II
NUMBER OF RULES OBTAINED WHEN APPLYING THE SOM+PSO AND PART METHODS

Database	SOM+PSO	PART
Balance scale	$\mathbf{7,106 \pm 0,51}$	$38,71 \pm 1,18$
Breast cancer	$\mathbf{5,033 \pm 0,19}$	$19,243 \pm 1,49$
Breast w	$\mathbf{2,853 \pm 0,18}$	$10,11 \pm 0,54$
credit_a	$\mathbf{3,446 \pm 0,21}$	$32,95 \pm 1,88$
Credit-g	$\mathbf{2,323 \pm 0,2}$	$62,543 \pm 1,65$
Diabetes	$\mathbf{2,55 \pm 0,2}$	$7,723 \pm 0,64$
Heart-c	$\mathbf{3,19 \pm 0,17}$	$19,24 \pm 0,75$
Heart-statlog	$\mathbf{4,276 \pm 0,35}$	$17,616 \pm 0,68$
Iris	$\mathbf{3,623 \pm 0,15}$	$4,113 \pm 0,32$
Mushroom	$\mathbf{3,503 \pm 0,14}$	$11,196 \pm 0,24$
Promoters	$\mathbf{7,013 \pm 0,26}$	$7,243 \pm 0,41$
Soybean	$\mathbf{24,856 \pm 0,5}$	$31,92 \pm 0,65$
Kr_vs_kp	$\mathbf{3 \pm 0}$	$22,196 \pm 0,63$
Zoo	$\mathbf{6,993 \pm 0,1}$	$7,67 \pm 0,07$
Slice	$\mathbf{10,046 \pm 0,33}$	$103,303 \pm 1,58$
Wine	$\mathbf{4,803 \pm 0,38}$	$5,44 \pm 0,25$
Drug5	$\mathbf{7,313 \pm 0,31}$	$15,42 \pm 0,45$
DrugY	$\mathbf{6,45 \pm 0,23}$	$11,79 \pm 0,47$
Adult	$\mathbf{2,153 \pm 0,14}$	$975,423 \pm 10,59$

times larger and generally more attributes are used in the antecedent. As it can be observed, in those cases mentioned above, the coverage obtained by SOM+PSO is very good if the reduced number of rules used is taken into account. For instance, for the "Breast w" database, the accuracy of SOM+PSO is lower than that of PART by approximately 1%, but the former uses one third of the number of rules of the latter and with less queries in each antecedent. Something similar is observed in all other 7 cases where PART yields a higher accuracy than SOM+PSO. This is due to the emphasis placed

TABLE III
ANTECEDENT AVERAGE LENGTH FOR EACH RULE OBTAINED WHEN APPLYING THE SOM+PSO AND PART METHODS

Database	SOM+PSO	PART
Balance scale	$\mathbf{1,93 \pm 0,1}$	$3,098 \pm 0,06$
Breast cancer	$\mathbf{1,164 \pm 0,03}$	$2,039 \pm 0,07$
Breast w	$\mathbf{1,88 \pm 0,15}$	$2,115 \pm 0,1$
credit_a	$\mathbf{1,278 \pm 0,11}$	$2,46 \pm 0,08$
Credit-g	$\mathbf{1,055 \pm 0,05}$	$2,581 \pm 0,07$
Diabetes	$\mathbf{1,176 \pm 0,08}$	$1,98 \pm 0,1$
Heart-c	$\mathbf{1,565 \pm 0,08}$	$2,564 \pm 0,12$
Heart-statlog	$\mathbf{1,616 \pm 0,13}$	$2,875 \pm 0,1$
Iris	$1,132 \pm 0,05$	$\mathbf{1,02 \pm 0,03}$
Mushroom	$1,34 \pm 0,1$	$\mathbf{1,259 \pm 0,02}$
Promoters	$1,097 \pm 0,02$	$\mathbf{1,03 \pm 0,04}$
Soybean	$5,962 \pm 0,26$	$\mathbf{2,731 \pm 0,05}$
Kr_vs_kp	$\mathbf{2,362 \pm 0,06}$	$3,122 \pm 0,08$
Zoo	$1,527 \pm 0,1$	$\mathbf{1,478 \pm 0,01}$
Slice	$\mathbf{1,485 \pm 0,05}$	$2,663 \pm 0,04$
Wine	$3,052 \pm 0,45$	$\mathbf{1,53 \pm 0,07}$
Drug5	$\mathbf{1,741 \pm 0,07}$	$2,066 \pm 0,03$
DrugY	$\mathbf{1,657 \pm 0,13}$	$1,868 \pm 0,03$
Adult	$\mathbf{2,229 \pm 0,22}$	$4,679 \pm 0,05$

on the simplification of the model. For the 11 remaining cases, there are no significant differences in relation to the accuracy achieved or SOM+PSO is better. Regardless of accuracy, the average length of the set of rules is always smaller in the method proposes, with cases such as the "Adult", "Credit-g" and "Slice" databases in which the difference is too evident.

7. CONCLUSIONS

A novel method for obtaining classification rules has been presented. This method is based on PSO and can operate with numerical and nominal attributes.
A SOM neural network was used adequately initialize the population of rules. The centroids obtained when grouping available data in an unsupervised manner allow identifying the relevance of each attribute for the swarm of examples. In any case, this metric is not enough to select the attributes that will form the rule, and it is at this point where PSO takes control to carry out the final selection.
A representation for the rules was proposed, combining a binary representation that allows selecting the attributes that are used in the rule with a continuous representation used only to determine the boundaries of the numerical attributes that are part of the antecedent. A variation of binary PSO was designed whose population is adequately initialized with the information from the centroids in the previously trained SOM network.
The results obtained when applying the method proposed on a set of test databases show that the SOM+PSO method obtains a simpler model. In average, it uses approximately 40% of the number of rules generated by PART, with antecedents formed by just a few conditions and an acceptable accuracy given the simplicity of the model obtained.
Although not included in this article, the measurements performed using the method proposed but skipping the optimization stage introduced by PSO resulted in a set of rules with similar accuracy but a slightly higher cardinality than PART. This shows that the few iterations carried out by PSO with the information from the centroids is essential to determine an appropriate set of rules.
Currently, even though based on the tests carried out there is no evidence of any dependence between the results obtained and the initial size of the SOM network, the measurements will be repeated using a dynamic SOM network.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[2] T. Scheffer, "Finding association rules that trade support optimally against confidence," in *Principles of Data Mining and Knowledge Discovery*, ser. Lecture Notes in Computer Science, L. Raedt and A. Siebes, Eds. Springer Berlin Heidelberg, 2001, vol. 2168, pp. 424–435.

[3] Y. Ye and C.-C. Chiang, "A parallel apriori algorithm for frequent itemsets mining," in *Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications*, ser. SERA '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 87–94.

[4] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[5] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 144–151.

[6] Z. Wang, X. Sun, and D. Zhang, "A pso-based classification rule mining algorithm," in *Proceedings of the 3rd International Conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, ser. ICIC '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 377–384.

[7] T. Sousa, A. Silva, and A. Neves, "Particle swarm based data mining algorithms for classification tasks," *Parallel Comput.*, vol. 30, no. 5-6, pp. 767–783, May 2004.

[8] N. Khan, M. Iqbal, and A. Baig, "Data mining by discrete pso using natural encoding," in *Future Information Technology (FutureTech), 2010 5th International Conference on*, 2010, pp. 1–6.

[9] N. Khan, A. Baig, and M. Iqbal, "A new discrete pso for data classification," in *Infor-*

mation Science and Applications (ICISA), 2010 International Conference on, 2010, pp. 1–6.

[10] M. Chen and S. Ludwig, "Discrete particle swarm optimization with local search strategy for rule classification," in *Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on*, 2012, pp. 162–167.

[11] Y. Jiang, L. Wang, and L. Chen, "A hybrid dynamical evolutionary algorithm for classification rule discovery," in *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, vol. 3, 2008, pp. 76–79.

[12] H. Wang and Y. Zhang, "Improvement of discrete particle swarm classification system," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, vol. 2, 2011, pp. 1027–1031.

[13] L. Yan and J. Zeng, "Using particle swarm optimization and genetic programming to evolve classification rules," in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 1, 2006, pp. 3415–3419.

[14] A. Özçift, M. Kaya, A. Gülten, and M. Karabulut, "Swarm optimized organizing map (swom): A swarm intelligence based optimization of self-organizing map," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10 640 – 10 648, 2009.

[15] C. Hung and L. Huang, "Extracting rules from optimal clusters of self-organizing maps," in *Computer Modeling and Simulation, 2010. ICCMS '10. Second International Conference on*, vol. 1, 2010, pp. 382–386.

[16] H. W. and L. L., "Dynamic self-organizing maps," in *XXXI Conf. Latinoamericana de Informatica, CELI 2005*, 2005.

[17] T. Kohonen, "Neurocomputing: foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Self-organized formation of topologically correct feature maps, pp. 509–521.

[18] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[19] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.

[20] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[21] ——, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5. Washington, DC, USA: IEEE Computer Society, 1997, pp. 4104–4108.

[22] L. Lanzarini, J. Lopez, J. A. Maulini, and A. Giusti, "A new binary pso with velocity control," in *Advances in Swarm Intelligence*, ser. Lecture Notes in Computer Science, Y. Tan, Y. Shi, Y. Chai, and G. Wang, Eds. Springer Berlin Heidelberg, 2011, vol. 6728, pp. 111–119.

[23] G. Venturini, "Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts," in *Machine Learning: ECML-93*, ser. Lecture Notes in Computer Science, P. Brazdil, Ed. Springer Berlin Heidelberg, 1993, vol. 667, pp. 280–296.

[24] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, ser. Lecture Notes in Computer Science, V. Porto, N. Saravanan, D. Waagen, and A. Eiben, Eds. Springer Berlin Heidelberg, 1998, vol. 1447, pp. 591–600.

[25] J. Kennedy and R. C. Eberhart, *Swarm intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[26] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

Received: October 2014. Accepted: February 2015.