Computer and Information Science; Vol. 5, No. 4; 2012
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Rule Extraction on Numeric Datasets Using Hyper-rectangles

Waldo Hasperué[1], Laura Cristina Lanzarini[1] & Armando De Giusti[1]

[1] Instituto de Investigación en Informática LIDI, School of Computer Science, National University of La Plata, La Plata, Argentina

Correspondence: Waldo Hasperué, Instituto de Investigación en Informática LIDI, School of Computer Science, National University of La Plata, CONICET scholarship, Calle 50 y 120, La Plata (1900), Argentina. Tel: 54-221-422-7707. E-mail: whasperue@lidi.info.unlp.edu.ar

Received: March 20, 2012 Accepted: April 28, 2012 Online Published: June 19, 2012

doi:10.5539/cis.v5n4p116 URL: http://dx.doi.org/10.5539/cis.v5n4p116

Abstract

When there is a need to understand the data stored in a database, one of the main requirements is being able to extract knowledge in the form of rules. Classification strategies allow extracting rules almost naturally. In this paper, a new classification strategy is presented that uses hyper-rectangles as data descriptors to achieve a model that allows extracting knowledge in the form of classification rules. The participation of an expert for training the model is discussed. Finally, the results obtained using the databases from the UCI repository are presented and compared with other existing classification models, showing that the algorithm presented requires less computational resources and achieves the same accuracy level and number of extracted rules.

Keywords: rule extraction, classification, numeric datasets, large datasets, hyper-rectangles, supervised learning

1. Introduction

Nowadays, various areas such as medicine, bio-informatics, financial services, marketing, biology, ecology, etc. have a significant volume of information whose analysis and understanding is highly important for those who need to use it. In this context, data mining is an area that allows transforming the information contained in large databases into knowledge that is useful for the user.

The extraction of knowledge is a process that combines machine learning, statistics and pattern recognition techniques, among others, used to assist the decision making process, understand the data and explain certain situations or phenomena. Various data mining techniques have been successfully applied to various areas that handle or have large volumes of data, as tools to model the available information and thus obtain knowledge (Bouguessa & Wang, 2009; Hsu & Chen, 2009; Koul, Caragea, Honavar, Bahirwani, & Caragea, 2008).

Among the tasks that can be carried out with data mining techniques, clustering and classification are of great interest. Clustering involves techniques capable of clustering data in different groups by means of a similarity measurement. It is expected that, using the selected metric or any given pattern, the elements within a group are similar to each other and at the same time different from the data in other groups (Bandyopadhyay & Saha, 2009; Chen, Chuang, & Chen, 2009; Lu, Wang, Lu, & Sun, 2008; Wang, Qi, & Xu, 2008; Aslanidis, Souliou, & Polykrati, 2008). On the other hand, the classification task includes techniques that know the class of each data element and whose purpose is establishing common patterns that allow explaining or summarizing the data belonging to each class (Bakar, Othman, Hamdan, Yusof, & Ismail, 2008; Kamwa, Samantaray, & Joos, 2009; Chen, Chuang, & Chen, 2009).

If-then rules are the most common way of passing knowledge, since they are easy to understand. Additionally, these rules can be modified or even removed, or new rules can appear, by applying adaptive techniques. This is why these rules are used by most existing techniques to produce knowledge (Kamwa et al., 2009; Martens, Baesens, & Van Gestel, 2009; Shi & Lei, 2008; Hasperué, Osella, & Lanzarini, 2008; Konig, Johansson, & Niklasson, 2007).

As regards data types, these can be numeric (continuous or discrete) or categoric (nominal or ordinal). There are various techniques to deal with them, even in a joint manner. In particular, when data are numeric and class limits are not well defined or overlap, the development of techniques that can describe and explain the different classes present in the database to analyze is of interest (Hasperué & Lanzarini, 2010). When the data domain to be treated is numeric, one of the tools used in several papers is the hyper-rectangle (Thornton, 1987; Salzberg,

1991) and more recently in (Garcia, Derrac, Luengo, & Herrera, 2009; Wei, Wang, & Yuan, 2010; Hasperué & Lanzarini, 2010). Given the various names that are found in the literature, in this paper we will use the term hyper-rectangle to refer to a rectangular body in a space that has more than three dimensions, rather than using hypercube or hypercuboid.

Hyper-rectangles are a powerful technique to group and describe the data in a class, since they can be easily handled in the input data domain. In these cases, the techniques must be able to deal with the possible overlapping between different hyper-rectangles. At the same time, hyper-rectangle boundaries can be used as conditional in the if-then rules that result from the adaptation process (Hasperué & Lanzarini, 2010).

Any classification algorithm must somehow make decisions when it finds classes where the upper limit separating both classes is not very clear or even non-linearly separable. Algorithms have various strategies that determine the end result and the internal structure for representing each class; the latter allows specifying a set of rules that describe it. In Cao (2010), the benefits that would be gained with the help of an expert in a data mining problem are described, particularly during the tasks of classification and subsequent rule extraction. Therefore, the idea of implementing algorithms that include human expertise as a parameter is of interest.

In this paper, a new technique is presented that uses hyper-rectangles to build a model based on classification rules that is applicable to numeric databases. The technique proposed starts by forming a hyper-rectangle for each class of the database and then, by using a series of indexes that measure different aspects of the existing overlaps between two hyper-rectangles, decides their reduction or division. Finally, rules describing the corresponding class are extracted from each hyper-rectangle.

The most interesting aspect of this proposal are the indexes used to decide the creation of new hyper-rectangles, which can be enabled or not during the training phase, and can even be replaced by a set of completely different indexes. This allows for a continued research on this topic to find new indexes that improve the ones presented here. In this paper, the indexes presented and their effect on the decision of reducing or dividing the hyper-rectangles is studied. We also discuss how the help of an expert can be useful.

Another positive feature of this technique is that it is not random at all, which means that, for the same input, the method proposed will always find the same answer. This is not the case with other strategies, which must be executed several times on the same data to find an average answer.

When working with large databases, a desirable feature for any classification technique is that it can build the model in a reasonable amount of time. The results achieved here show that the algorithm presented requires a lower amount of computational resources than the classification techniques that use evolutionary and optimization strategies, achieving similar results as regards the accuracy of the model built and the number of rules extracted.

The rest of the paper is organized as follows: in section 2, some previous works that have used hyper-rectangles as classification tools are described. In section 3, some definitions and concepts on hyper-rectangles are presented. In section 4, the indexes used as criteria to divide hyper-rectangles are described. In section 5, the proposed algorithm is discussed. In section 6, the results obtained are detailed. Finally, in section 7, the experiments carried out are discussed and some future works are presented.

2. Related Work

There are several published papers that use the concept of hyper-rectangles to solve classification problems in numeric data spaces (Thornton, 1987; Salzberg, 1991; Garcia et al., 2009; Wei et al., 2010; Holden & Freitas, 2008). All these papers present different solutions to deal with the problem of hyper-rectangle intersection and how to avoid assigning database elements to the wrong class.

2.1 Hyper-Cuboid Formation

In Thornton (1987), an algorithm is proposed that deals with the classification of patterns belonging to two classes divided into positive and negative samples. The algorithm starts with a hyper-rectangle that contains all the patterns. This hyper-rectangle is subdivided into other hyper-rectangles; the number of hyper-rectangles to be created depends on the different values of some selected attribute. The newly created hyper-rectangles that contain both positive and negative samples are in turn divided into smaller hyper-rectangles. This procedure continues until all the hyper-rectangles that are created contain samples belonging to only one class.

The main problem of this algorithm is that the boundaries of the hyper-rectangles that are created either contain an only value or the entire range of possible values for any given attribute. This makes it impossible to extract rules from large databases, given the possibility of obtaining a huge number of hyper-rectangles.

2.2 NGE Theory

In Salzberg (1991), an exemplar-based learning algorithm is presented. This type of learning gradually forms the knowledge structure as the data are presented to the algorithm. Thus, the algorithm starts with points in the hyperspace and then, by measuring distances, the new exemplars are grouped with the nearest elements. Points are thus transformed into small hyper-rectangles, which gradually grow as new elements are analyzed.

This method adds the concept of element weight, indicating the prediction reliability of the element; this weight is used to calculate distances and thus decide to which hyper-rectangle the new element belongs. These weights are continuously modified during the execution of the algorithm.

In the case of overlaps, the algorithm proposed in Salzberg (1991) allows creating hyper-rectangles within others as exception hyper-rectangles.

2.3 EHS-CHC

In Garcia et al. (2009), an efficient solution to the problem presented in the paper mentioned in section 2.2 and Nested Generalized Exemplar (NGE) theory is presented. In this problem, the classification of a new exemplar consists in finding the nearest hyper-rectangle based on a given distance measurement. To do so, the authors propose using an evolutionary algorithm to determine the set of optimal hyper-rectangles to carry out the desired classification. To carry out this task, a representation of a subset of possible hyper-rectangles is used as population individual, and a classification rate that is measured using the number of hyper-rectangles used and their volume is used as fitness function. After the evolutionary process is finished, the hyper-rectangles of the best individual found are extracted from the phenotype, and each of these is used to extract a rule. This also allows hyper-rectangle overlapping and, in the case of new data that need to be classified and simultaneously fall in an overlap, the authors propose that these data are classified by the hyper-rectangle class with a lower volume.

2.4 RHC

In Wei et al. (2010), Rough Sets theory is extended to work with what the authors have called Rough Hypercuboid. The Rough Sets theory defines sets with lower and upper approximations, which allows working with uncertainty and unknown data. The authors gradually and dynamically create the hyper-rectangles making sure they use the lowest possible number of attributes for this task. In this particular paper, the authors solve the cancer classification problem, so the end result is a set of hyper-rectangles that are used to successfully classify cancer samples.

2.5 PSO/ACO2

The technique presented in Holden and Freitas (2008) uses hyper-rectangles in an indirect manner to find the data model. This technique uses the PSO heuristics to find the minimum set of rules that ensures a maximum accuracy in the data model. Each particle of the PSO swarm represents a numeric interval for each of the attributes of the database, so it can be compared with the lower and upper limits of a hyper-rectangle. Thus, the results obtained with this technique can be compared with those obtained with the technique proposed here.

3. Hyper-Rectangles

A hyper-rectangle is a body in the D-dimensional Euclidean space, equivalent to a rectangle in the two-dimensional space or a cuboid in the three-dimensional space. Even though a hyper-rectangle can rotate on each of its axis, for this paper, only those whose faces are parallel to the axes are of interest. In these particular cases, a hyper-rectangle H in a D-dimensional space can be defined by the minimum and maximum values of each of the dimensions (axis in the space). Thus, H is defined by two sets of values, the set of minimum values $H_n = \{h_{ni}, \forall i=1..D\}$, and the set of maximum values $H_x = \{h_{xi}, \forall i=1..D\}$, where h_{ni} is the lower limit in dimension i and h_{xi} is the upper limit in dimension i. It is clear that, for any hyper-rectangle in any space, $h_{ni} < h_{xi}$, $\forall i=1..D$. As it can be observed, these notations are only valid for hyper-rectangles whose faces are parallel to the axes. Any rotation in at least one dimension causes these notations not to be valid. Additionally, a hyper-rectangle that is parallel to the axes can be directly used to establish classification rules, which would be impossible to do with a rotated hyper-rectangle.

Definition: An element e that belongs to the D-dimensional Euclidean space is inside hyper-rectangle H if $h_{ni} <= e_i <= h_{xi}$, $\forall i=1..D$, and it will be notated with the operator $e \in H$.

Definition: The number of elements that fall inside a hyper-rectangle H is notated as $\Xi(H)$.

Two hyper-rectangles may present a partial or total overlap in the space.

Definition: Two hyper-rectangles H and P overlap in the D-dimensional space if any of these four conditions is

met in each and every one of the dimensions:

i. $h_{ni} <= p_{ni} <= h_{xi}$

ii. $h_{ni} <= p_{xi} <= h_{xi}$

iii. $p_{ni} <= h_{ni} <= p_{xi}$

iv. $p_{ni} <= h_{xi} <= p_{xi}$

In particular, when conditions i and ii are met in all space dimensions, P is entirely included in H, and if conditions iii and iv are simultaneously met in all dimensions, H is entirely included in P.

Hyper-rectangles can be used to solve classification problems and for the subsequent extraction of rules to represent the knowledge acquired. In a database B whose elements have D attributes in the numeric domain, each element in the database is a point in the D-dimensional space. Thus, a hyper-rectangle H can be defined over the elements in B in such a way that each element in B is inside H. As it can be seen, there are infinite hyper-rectangles that can include all of the elements in B. For the classification task, the hyper-rectangle with the lowest possible hypervolume will be used.

Definition: the lowest-hypervolume hyper-rectangle that contains all of the elements in B is the hyper-rectangle such that $h_{mi} = \min(B_i)$ and $h_{xi} = \max(B_i)$, $\forall i=1..c$, where $\min(B_i)$ and $\max(B_i)$ are the minimum and maximum values, respectively, taken by attribute i in B.

If B simultaneously has an attribute for each element that defines to which class such element belongs, a subset $C \subset B$ can be created, where each element in C has the same value in the class attribute. Therefore, a hyper-rectangle H can be built for C and the same label as that of the elements in C can be assigned to H, so $Class(H)$=C. With this criterion, if B has elements that belong to several classes, it is possible to define as many hyper-rectangles as classes exist so that each hyper-rectangle represents its corresponding class.

Definition: Within B, any class C with q elements can be modeled using k hyper-rectangles. If these q data in C are each contained in an only hyper-rectangle H_i, then the following condition holds true:

$$\Phi(\mathrm{C}) = \sum_{i=1}^{k} \Xi(H_i) = q \tag{1}$$

If in a database all hyper-rectangles that represent each class do not overlap with each other, the hyper-rectangle adjustment task is unnecessary and each class is represented by its corresponding hyper-rectangle. This does not happen in real problems, where it is in fact very frequent that hyper-rectangles overlap with each other. This is not desirable for the classification task, since an element e can be included in more than one hyper-rectangle and be therefore classified as belonging to more than one class.

Several actions can be taken to solve these problems and avoid the existence of overlaps, such as reducing one or more hyper-rectangles until the overlap is minimal (if false positive data are accepted) or null. Another possibility is dividing hyper-rectangles into smaller ones. Regardless of the actions taken, there are numerous combinations of different actions to achieve a data model that represents an acceptable classification, which becomes more complex as the number of work space dimensions and the number of classes involved grow.

i. Figure 1a shows an example where two hyper-rectangles with an intersection in the space do not present data from any of the two hyper-rectangles. Even though this case does not pose any significant difficulties in the model with training data, it is not a desirable model, since if in the future new elements are to be classified, those falling in the intersection will belong to both classes. The solution to this case is dividing one of the two hyper-rectangles.

ii. Figure 1b shows two hyper-rectangles that intersect, but the data in the intersection come from only one of the hyper-rectangles. The most practical solution would be to divide the hyper-rectangle that has no data in the intersection.

iii. Figure 1c shows the intersection between two hyper-rectangles that includes data from both hyper-rectangles. This example can be easily solved by dividing both hyper-rectangles. Figure 1d shows a similar case, where it is impossible to eliminate the intersection without creating hyper-rectangles that contain only one element. For this type of problems, one of the two classes will have to keep the data from the other, which will later on result in false positives.

iv. Figure 1e shows a case that is similar to the one shown in 1d, but the solution in this case requires more divisions and adjustments, and the number of possible solutions is also higher. Figure 1f shows maybe the most complex case of all, since it requires a greater number of divisions and has the greatest number of possible solutions.

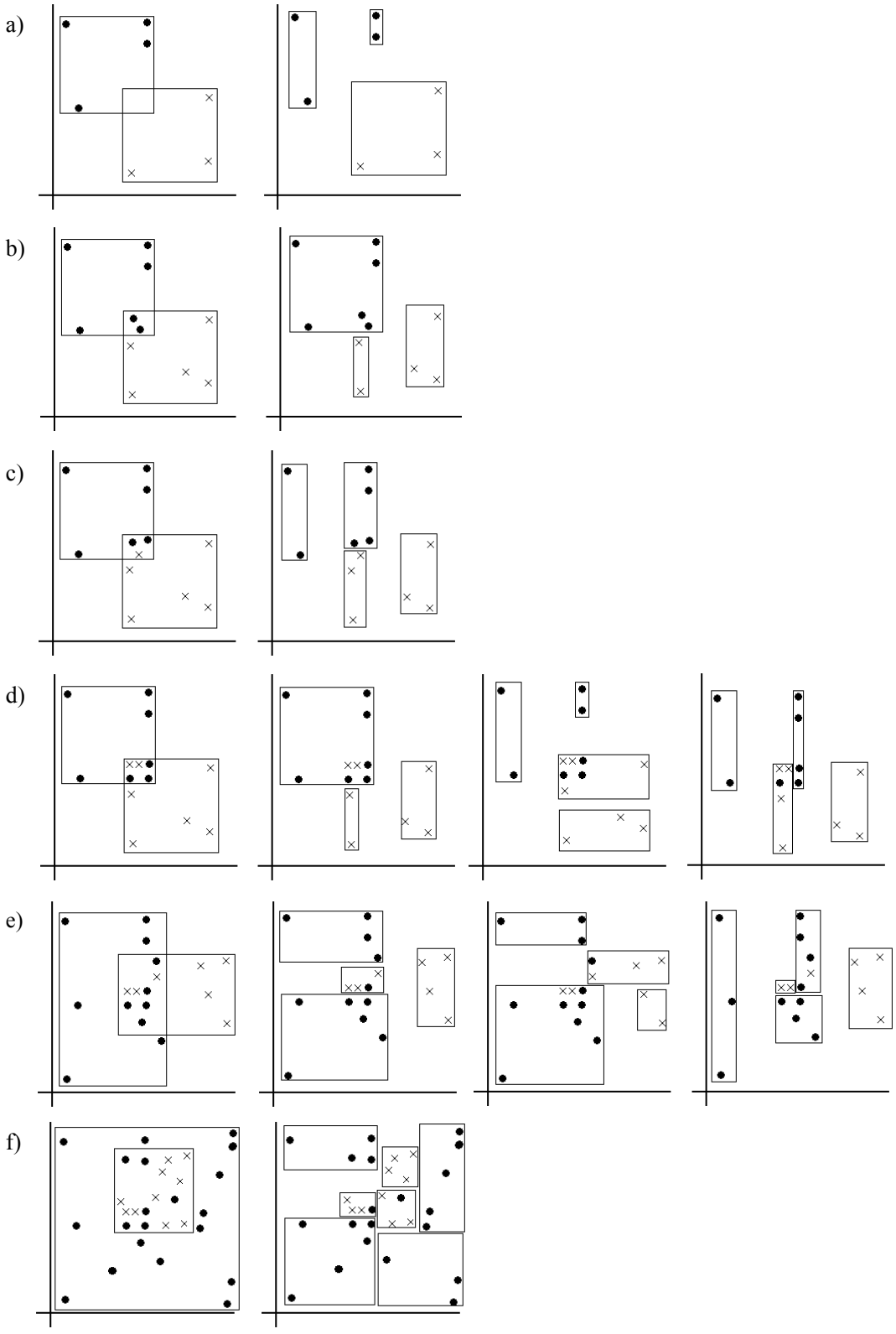


Figure 1. Remove overlap

Description: To the left, different problems of intersected areas that have to be divided are shown; to the right, possible solutions to remove or minimize the overlap are shown.

As it can be seen, in every one of these cases decisions must be made in favor of dividing one hyper-rectangle or the other, and when the intersection is between three or more hyper-rectangles that belong to different classes,

the problem is much more cumbersome. It is in these cases where an algorithm that is able of making all these decisions is useful, so that the minimum number of hyper-rectangles per class is used, and the rules extracted from them are few and simple for their understanding and subsequent use.

3.1 Definitions and Concepts

In this section, some definitions are presented before explaining the indexes.

Definition: If two hyper-rectangles H and P overlap, this overlap is in turn a new hyper-rectangle S that is defined by these boundaries:

$$s_{ni} = \max(h_{ni}, p_{ni}) \tag{2}$$

$$s_{xi} = \min(h_{xi}, p_{xi}) \tag{3}$$

Be H and P two hyper-rectangles that each represent a different class, and be S the overlap between both hyper-rectangles, we will define function Ψ that determines the number of elements from H that are included in S and the number of elements from P that are included in S.

$$\Psi(S, H) = \#\{e \mid e \in H \text{ and } e \in S\} \tag{4}$$

$$\Psi(S, P) = \#\{e \mid e \in P \text{ and } e \in S\} \tag{5}$$

Projecting the intersected area S in any dimension i, the area interval is the interval formed by the values s_{ni} and s_{xi}.

Be $E_i(H)$ and $E_i(P)$ the sets formed by the values in dimension i of those data that belong to H and P that fall within intersection S, that is:

$$E_i(H) = \{e \mid e \in H \text{ and } s_{ni} \le e_i \le s_{xi}\}, \forall i=1..D \tag{6}$$

$$E_i(P) = \{e \mid e \in P \text{ and } s_{ni} \le e_i \le s_{xi}\}, \forall i=1..D \tag{7}$$

$j_{ni}(H)$ and $j_{xi}(H)$ are the minimum and maximum values of the elements in $E_i(H)$. The same for $E(P)$. These two intervals can in turn intersect with each other; if this happens, it is called data interval. t_{ni} and t_{xi} are defined as the boundaries of the intersection formed by the data from $E_i(H)$ and $E_i(P)$:

$$t_{ni} = \max(j_{ni}(H), j_{ni}(P)) \tag{8}$$

$$t_{xi} = \min(j_{xi}(H), j_{xi}(P)) \tag{9}$$

Be $F_i(H)$ and $F_i(P)$ the subsets in $E_i(H)$ and $E_i(P)$, respectively, formed by the data that are within the data interval for dimension i.

$$F_i(H) = \{e \mid e \in E_i(H) \text{ and } t_{ni} \le e_i \le t_{xi}\}, \forall i=1..D \tag{10}$$

$$F_i(P) = \{e \mid e \in E_i(P) \text{ and } t_{ni} \le e_i \le t_{xi}\}, \forall i=1..D \tag{11}$$

$v_{ni}(H)$ and $v_{xi}(H)$ are defined as the limits of the interval formed by $F_i(H)$ that fall within the intersection of data established by t_{ni} and t_{xi} (Figure 2):

$$v_{ni}(H) = \min\{e_i \mid e_i \in F_i(H)\}, \forall i=1..D \tag{12}$$

$$v_{xi}(H) = \max\{e_i \mid e_i \in F_i(H)\}, \forall i=1..D \tag{13}$$

Similarly, the values $v_{ni}(P)$ and $v_{xi}(P)$ can be defined.

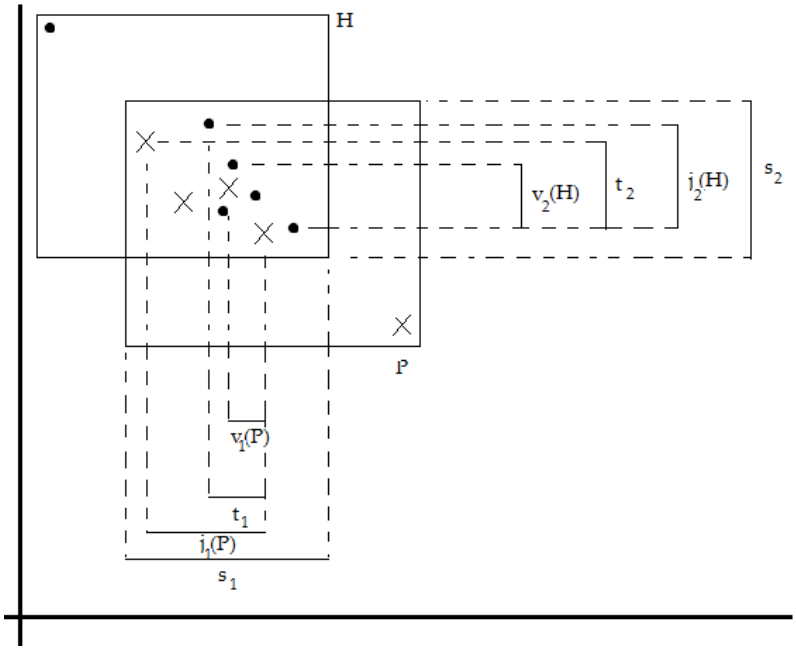


Figure 2. A case of overlap

Description: A case of overlap is shown, with the area and data intervals established and the s, j, t and v markers.

4. Indexes

The technique described in this paper to solve the task of removing overlaps consists in calculating a series of indexes for each existing intersection and in each of the dimensions of the space, where each index measures different aspects of the intersection. Then, these indexes are combined to obtain an only final value, which we will call divisibility index Ω, that will determine the dimension that has to be modified at the intersection to reduce or divide the hyper-rectangles.

The indexes proposed in this paper are the following:

$z_1(H)$ and $z_1(P)$. Proportion of the width of the intersection compared to the hyper-rectangle.

$$z_1(H) = 1 - (s_{xi} - s_{ni}) / (h_{xi} - h_{ni}) \quad (14)$$

$$z_1(P) = 1 - (s_{xi} - s_{ni}) / (p_{xi} - p_{ni}) \quad (15)$$

This index tends to 1 as the intersection decreases, and it tends to 0 as the intersection between the hyper-rectangles increases. Thus, those dimensions where the intersection proportion is small are favored.

$z_2(H)$ and $z_2(P)$. Proportion of the width of the data intersection between classes compared to the data width of the hyper-rectangle.

$$z_2(H) = 1 - (t_{xi} - t_{ni}) / (j_{xi}(H) - j_{ni}(H)) \quad (16)$$

$$z_2(P) = 1 - (t_{xi} - t_{ni}) / (j_{xi}(P) - j_{ni}(P)) \quad (17)$$

This index tends to 1 as data intersection decreases, and it tends to 0 as the mixture of data from both classes increases within the intersection. Therefore, it favors those cases where data intersection or *mixture* is minimal. It should be noted that even if there is intersection as regards volume, there could be no intersection of the data themselves, even if data from both hyper-rectangles are present in the intersection (see Figure 3). In these cases, the maximum value t_{xi} will be lower than the minimum value t_{ni} and it is established that, since there is no data intersection, both $z_2(H)$ and $z_2(P)$ take a value of 1.

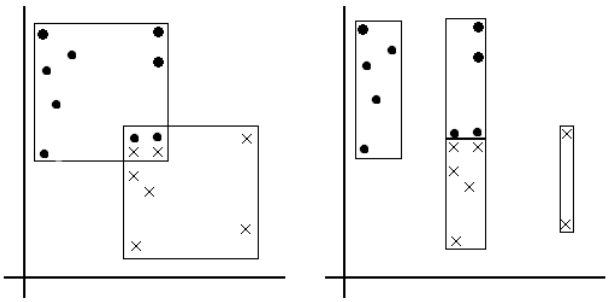


Figure 3. Overlap easily solved

Description: This figure shows an example of an area over-lap with data overlap, where divisibility is easily solved without allowing false positives.

$z_3(H)$ and $z_3(P)$. Proportion of the width of intersected data from a hyper-rectangle compared to the width of the intersected data.

$$z_3(H) = 1 - (v_{xi}(H) - v_{ni}(H)) / (t_{xi} - t_{ni}) \quad (18)$$

$$z_3(P) = 1 - (v_{xi}(P) - v_{ni}(P)) / (t_{xi} - t_{ni}) \quad (19)$$

This index tends to 1 as the data interval of a hyper-rectangle within the data intersection interval decreases, and it tends to 0 as the mixture of data from both classes increases within the intersection. Therefore, it favors those cases where data intersection is minimal. The same as the previous index, if there is no data intersection within the area intersection, it can be established that both $z_3(H)$ and $z_3(P)$ are 1.

$z_4(H)$ and $z_4(P)$. Proportion of the width of the data from a hyper-rectangle within the area intersection compared to the total width of the area intersection.

$$z_4(\mathrm{H}) = 1 - (j_{xi}(H) - j_{ni}(H)) / (s_{xi} - s_{ni}) \quad (20)$$

$$z_4(\mathrm{P}) = 1 - (j_{xi}(P) - j_{ni}(P)) / (s_{xi} - s_{ni}) \quad (21)$$

This index tends to 1 as the data interval of a hyper-rectangle within the area intersection interval decreases, and it tends to 0 as the data from both the hyper-rectangle are more *spread out* within the intersection. Therefore, it favors those cases where the data from a hyper-rectangle are grouped in a small interval within the data intersection.

$z_5(H)$ and $z_5(P)$. Proportion of the shift of the intersected data interval from a hyper-rectangle compared to the minimum of the data interval from the other hyper-rectangle.

$$z_5(H) = 1 - (v_{ni}(H) - j_{ni}(P)) / (j_{xi}(P) - j_{ni}(P)) \quad (22)$$

$$z_5(P) = 1 - (v_{ni}(P) - j_{ni}(H)) / (j_{xi}(H) - j_{ni}(H)) \quad (23)$$

This index tends to 1 as the lower limit of the intersected data interval is closer to a hyper-rectangle compared to the minimum of the data interval within the area intersection of another hyper-rectangle, and it tends to 0 as it is farther away. This index favors those cases where the intersected data are closer to their own hyper-rectangle. If there is no data intersection within the area intersection, then it can be established that both $z_5(H)$ and $z_5(P)$ are 1.

$z_6(H)$ and $z_6(P)$. Proportion of the shift of the intersected data interval from a hyper-rectangle compared to the maximum of the data interval from the other hyper-rectangle.

$$z_6(H) = (v_{xi}(H) - j_{xi}(P)) / (j_{xi}(P) - j_{ni}(P)) \quad (24)$$

$$z_6(P) = (v_{xi}(P) - j_{xi}(H)) / (j_{xi}(H) - j_{ni}(H)) \quad (25)$$

This index, similar to the previous one, measures the shift of the data interval within the data intersection compared to the maximum of the data interval within the area intersection of the other hyper-rectangle. This index is the opposite of the previous one, because it tends to 1 as the shift of the interval increases, and it tends to 0 as it is closer to the maximum. Thus, those cases where a hyper-rectangle has data that are closer to the limits of the interval of the other hyper-rectangle are punished.

The final index Ω is calculated as the average of all the z indexes, weighing the amount of data from both classes

that are involved in the intersection. In cases of similar divisions where the indexes have similar values, this weighing favors acting on those overlaps that have a larger amount of data involved. It is calculated as the proportion of data involved versus the total amount of data from the classes (not from the hyper-rectangles).

Therefore, the divisibility index Ω is defined as follows:

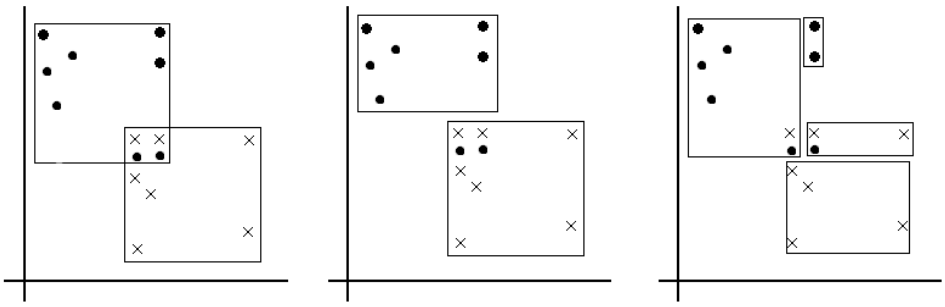
$$\Omega_i(H,P,S) = \frac{\sum_{i=1}^{m}(Z_i(H)+Z_i(P))}{m} \frac{\Psi(S,H)+\Psi(S,P)}{\Phi(Class(H))+\Phi(Class(P))} \tag{26}$$

5. Algorithm

In this section, the operation of the technique proposed, called CLUHR (Classification Using Hyper-Rectangles) is detailed. The purpose of CLUHR is minimizing functions Ψ(*S*, *H*) and Ψ(*S*, *P*). In particular, it is desirable that conditions Ψ(*S*, *H*) = 0 and Ψ(*S*, *P*) = 0 are achieved as a result, which would occur by removing *S*, since there would be no overlap, and thus achieve a data model formed by class representations that ensure an error-free classification.

It should be noted that even if there is an overlap between both hyper-rectangles, it may happen that Ψ(*S*, *H*) = 0 and Ψ(*S*, *P*) = 0 (see Figure 1a). Those cases where [Ψ(*S*, *H*) > 0 and Ψ(*S*, *P*) = 0] or [Ψ(*S*, *H*) = 0 and Ψ(*S*, *P*) > 0] are simple to deal with, since dividing the hyper-rectangle that presents Ψ = 0 is enough to remove the overlap, and this can be done by dividing only one of the two hyper-rectangles (see Figure 1b).

The most critical case to be solved is when Ψ(*S*, *H*) > 0 and Ψ(*S*, *P*) > 0. There are two possible ways for removing a hyper-rectangle overlap. One is dividing one or both hyper-rectangles in lower volume hyper-rectangles (see Figure 1f). The other one is reducing the volume of one or both hyper-rectangles (see Figure 4). Either way, the resulting hyper-rectangles must always be parallel to the axis, since this ensures that the hyper-rectangle can then be transformed into a classification rule. Therefore, this division or reduction operation is always done by modifying the values of only one dimension, as shown in Figure 4. The question that must be taken into account when dividing or reducing the hyper-rectangle is which dimension to modify. The answer to this question is critical, and depending on how the task is carried out, the result can be good or poor classifications. The ideal result would be achieving a modification that represents a minimal change and at the same time generates the lowest possible number of hyper-rectangles.



a) b) c)

Figure 4. Remove overlap

Description: This figure shows a case of overlap (a) and two possible solutions by reducing the volume (b) or by dividing into smaller hyper-rectangles (c).

The algorithm that determines the number of hyper-rectangles defined for each class and how they are defined is the following.

It starts by forming the closest hyper-rectangle for each class, and it then searches for overlaps between the hyper-rectangles formed. If there is no overlap, there is nothing else to be done. If there are overlaps where Ψ(*S*, *H*) = 0, then it goes on and removes such overlaps. If all Ψ(*S*, *H*) > 0 for all overlaps, then the Ω_i indexes are calculated for the *D* dimensions. The highest Ω_i determines the hyper-rectangle intersection that is to be modified, and on which dimension this is to be done.

There are two possible ways of removing an intersection: By dividing one or both involved hyper-rectangles into smaller volume ones or by reducing the volume of one or both hyper-rectangles. The technique proposed in this

paper does not suggest any method in particular, since in most of the cases the decision will favor one class over the other and it will therefore depend on the result sought.

If there is only one area overlap, then the decision is simple and one of the hyper-rectangles is divided (Figure 1a), but which one? The largest volume hyper-rectangle could be divided, as shown in Figure 1b or, if both hyper-rectangles are similar in volume, they could both be divided (Figure 1c). There are more complex cases, with one hyper-rectangle entirely included within the other, and which will require dividing both hyper-rectangles (Figure 1f).

When the data overlap is impossible to remove, the limits of the hyper-rectangles have to be adjusted; this can be done by adjusting one or both hyper-rectangles (Figure 4b). Again, the decision as to which one to adjust, and how much, will depend on the problem at hand.

The algorithm has an initialization stage during which the first hyper-rectangles of each class are calculated. Then, the existing overlaps are determined and all indexes are calculated for each of the overlaps found. The intersection and dimension with the highest Ω index are found, and hyper-rectangles are divided in that intersection. Then, the affected hyper-rectangles are recalculated, the new intersections are determined, and the indexes are recalculated. This iterative process continues until all overlaps have been removed or until a stop condition is reached, which can be a maximum number of hyper-rectangles or when the overlaps have very few elements.

To avoid the algorithm having to deal with overlaps of only four or five elements, which in a database that has hundreds of thousands of data would not be meaningful, an additional condition can be established indicating that overlaps are to be analyzed only if the number of data involved is greater than a certain amount. This amount of data is a parameter of the algorithm that is called α.

When the algorithm ends, and if there are no data overlaps (or only those remain that were minimal and were not resolved for not going over the number established by parameter α), there may still be area overlaps. At this point, a decision must be made whether to allow those overlaps or not. If they are allowed, it could happen that future cases presented to the model could be classified in two different classes, and the problem of deciding the class to which these future elements belong would have to be solved at that time. The method proposed in Garcia et al. (2009) also has this type of problems, and the authors decided to assign the elements to the class represented by the smallest volume hyper-rectangle. If no area overlaps are allowed because accurate classification rules are sought for, the hyper-rectangles must then be reduced. When $\Psi(S, H) = \Psi(S, P) = 0$, either of the hyper-rectangles can be easily divided. If only $\Psi(S, H) = 0$, then H is divided and P is left untouched. And if $\Psi(S, H) > 0$ and $\Psi(S, H) > 0$, then one of the two hyper-rectangles is divided and the data in one of the classes are left as false positives. The hyper-rectangle with a lower $\Psi(S, H)$ could be divided, but this will always depend on the problem at hand and whether for that problem, the presence of false positives in one class is more important than in the other. For instance, in the cancer detection database, it is preferable that the model prioritizes the successful detection of malignancies.

The pseudo-code of the algorithm for the proposed method is the following:

Create the initial hyper-rectangle for each class

Calculate overlaps among all hyper-rectangles

***while** there are overlaps to deal with*

begin

calculate and look for the overlap and dimension i of the highest Ω_i index

divide or adjust the appropriate hyper-rectangle in dimension i

Recalculate overlaps among all hyper-rectangles

end

Remove all remaining overlaps

5.1 Rule Extraction

Once the algorithm ends, the result obtained is a data model that consists in a set of hyper-rectangles. Each class has one or more hyper-rectangles that represent it. Each hyper-rectangle will be used to extract a classification rule. This rule will be formed by the limits of the hyper-rectangle itself. Thus, the rule extracted from a hyper-rectangle H will be the following:

IF (attribute_1_value $\geq h_{n1}$ AND attribute_1_value $\leq h_{x1}$) AND (attribute_2_value $\geq h_{n2}$ AND attribute_2_value $\leq$

h_{x2}) AND … AND (attribute_d_value ≥ h_{nd} AND attribute_d_value ≤ h_{xd}) THEN value belongs to *Class*(*H*)

This procedure obviously produces very complex rules, in particular if the space dimension of the problem is high. If the purpose is generating knowledge and extracting a set of rules that explain some aspects of the data to the user, then these complex and long rules will not be very helpful. It can be easily seen that the inclusion of many of the conditions would not be essential, as in the case shown in Figure 4b. Only the minimum value of the Y axis of the circles hyper-rectangle and the maximum value of the Y axis of the crosses hyper-rectangle would be necessary to pass knowledge to the expert for him/her to understand the nature of these data.

This type of rule condition refinement can be done by simple inspection; the problem is when the number of dimensions or rules is high. There are methods in the literature that can automatically simplify rules (Darrah, Taylor, & Skias, 2004; Ma, Guo, Liu, Ma, & Chen, 2009).

5.2 Classification

Once the training stage is finished, the data model is ready for two different tasks. On the one hand, with the rules extracted, it can provide the user information that can be of interest, for example, explaining if there are large differences between two classes. It can also classify future samples. To do this, when a new sample is presented, the model looks for the hyper-rectangle in which the sample falls and thus determines the data class to which it belongs. If the sample does not fall in any hyper-rectangle, then the model looks for the one that is closest to the sample to determine the class to which the sample belongs. The search for the hyper-rectangle *H* that is closest to a sample *m* is carried out by applying Equation 27. This equation returns the lowest Euclidean distance between the sample and the side or vertex of *H* that is closest to *m*. It should be noted that if *m* falls within *H*, then the equation returns 0 as distance.

$$Dist(m,H) = \sqrt{\sum_{i=1}^{D} dif_i} \tag{27}$$

where

$$dif_i = \begin{cases} (m_i - h_{ni})^2 & if\ m_i < h_{ni} \\ (m_i - h_{xi})^2 & if\ m_i > h_{xi} \\ 0 & otherwise \end{cases} \tag{28}$$

5.3. Problem-Domain Expert Intervention

The algorithm in CLUHR, to build the data model, has to make numerous decisions regarding which overlap to solve. Even though these decisions are made by calculating the Ω indexes, there are different ways to calculate these indexes, and the elements used for calculating them can also be weighed differently. On the other hand, once the overlap that is to be removed has been selected, a decision has to be made regarding the need - or not - to divide the hyper-rectangles, reduce their volume, which hyper-rectangle to divide, or how to carry out this division. In this sense, CLUHR does not make any decision in particular, since the correct option will to a great extent depend on the problem.

In any case, all these decisions can be made before building the model, configuring the algorithm to build and run the model in a fully automated manner to achieve the end result. Even though CLUHR is entirely flexible in the sense that an expert, before running the process, can determine which indexes to use and how to carry out the divisions, customizing the automated process, in real problems it is never possible to build an automatic process that is able to improve the "on-line" decisions that would make a human being, even more so when this human component is an expert on the domain of the problem.

Therefore, the availability of a tool that allows the expert to monitor the building of the model is of interest, so that the expert can participate in each and every one of the overlaps that have to be removed, deciding the order in which overlaps are eliminated and the way in which each of them will be removed by modifying the hyper-rectangles based on the needs of the problem. Thus, the expert can oversee the building of the model and contribute his or her knowledge, since the following advantages are available:

- Possibility of including or excluding Z indexes, and how to weigh each of them.
- Possibility of seeing the results of calculating the Ω indexes and deciding to remove an overlap even if its Ω index is not the highest, since the characteristics of the classes involved or the overlap itself can be more significant for removal in certain cases.

- Possibility of viewing partial results of applying any particular division, so that the expert can decide which division to carry out at any given time in the algorithm. The analysis of an entire tree of n levels with possible alternatives can even be performed.
- Possibility of viewing which overlaps exist within the same work space, which classes are involved, and how many data are involved.
- The expert also has numeric help available in the form of the number of data involved, the area of the space that is affected by the overlap, statistical information on involved data, confusion matrix, etc.

6. Results

The technique proposed in this paper has been tested with 13 databases of the UCI repository (Frank & Asuncion, 2010). All data in these databases are in the numeric domain. The results obtained by CLUHR are compared with the results presented by Garcia et al. (2009) and Holden and Freitas (2008). The computational effort required by these techniques to build the model is also compared.

The test known as 10-fold cross validation was performed over each database; the test was run 10 separate times over each dataset and the final accuracy was determined as the average of these 10 runs. The comparative results of classification accuracy for CLUHR and the results presented by Garcia et al. (2009) and Holden and Freitas (2008) are shown in Table 1. Table 2 shows the average number of hyper-rectangles that were created during the process.

The two-sided t-student test with a confidence level of 95% was carried out to determine if the differences between CLUHR and PSO/ACO2 are statistically significant. In Tables 1 and 2, the signs "+" or "-" are used to indicate if CLUHR was better or worse, respectively, with a statistically significant difference, and the sign "=" is used to indicate that there was no difference. It should be noted that the work presented in Garcia et al. (2009) does not include statistical data and it was therefore impossible to compare it.

As it can be seen in Tables 1 and 2, it can be concluded that even though the efficacy of the method proposed is similar to the efficacy of the other techniques, the number of hyper-rectangles generated is lower and therefore, the user can work with less rules.

Table 1. Accuracy comparison

Dataset	EHS-CHC	PSO/ACO2	CLUHR	
Ecoli	0.7948		0.7891 (0.0160)	
Glass	0.6287	0.7095 (0.0750)	0.6215 (0.0360)	-
Haberman	0.7122		0.7356 (0.0064)	
Image		0.9667 (0.0117)	0.8538 (0.0135)	-
Ionosphere		0.8806 (0.0491)	0.8777 (0.0169)	=
Iris	0.9267	0.9467 (0.0526)	0.9300 (0.0079)	=
Liver	0.6167		0.5918 (0.0211)	
Pima	0.7384		0.5595 (0.0191)	
Sonar	0.7650	0.7505 (0.0911)	0.6666 (0.0283)	-
Vehicle		0.7305 (0.0445)	0.6819 (0.0171)	-
Vowel		0.8616 (0.0347)	0.7120 (0.0132)	-
Wine	0.9490		0.9530 (0.0113)	
Wisconsin	0.9599	0.9487 (0.0253)	0.9251 (0.0102)	-

Description: Accuracy of the method proposed versus the results presented in Garcia et al. (2009) and Holden and Freitas (2008). Standard deviation is indicated between brackets; statistical differences are shown in the last column.

Table 2. Number of hyper-rectangles comparison

Dataset	EHS-CHC	PSO/ACO2	CLUHR	
Ecoli	11.1		12.62 (1.44)	
Glass	12.2	20.4 (1.35)	15.17 (1.30)	+
Haberman	4.4		4.29 (0.33)	
Image		21.9 (0.99)	10.93 (0.47)	+
Ionosphere		3.6 (0.97)	3.98 (0.37)	=
Iris	3.4	3.0 (0.0)	3.21 (0.12)	-
Liver	9.8		17.79 (2.21)	
Pima	11		10.45 (0.91)	
Sonar	10.3	4.4 (1.58)	4.14 (0.20)	+
Vehicle		37.8 (1.2)	32.35 (2.03)	=
Vowel		29.0 (0.82)	31.74 (0.78)	-
Wine	3.6		3.18 (0.11)	
Wisconsin	3.8	10.2 (1.87)	9.63 (1.39)	+

Description: Number of hyper-rectangles created for the data model by the strategies studied. Standard deviation is indicated between brackets; statistical differences are shown in the last column.

The techniques proposed in Garcia et al. (2009) and Holden and Freitas (2008), being based on evolutionary and optimization strategies, are able to find an individual from their respective populations that represents an optimal solution to the problem. Even if this is the case, it can be seen that CLUHR achieves similar results to those obtained by these techniques. The main disadvantage of the latter is the computational effort that they require to build the model.

In Garcia et al. (2009), an evolutionary algorithm is used to find a set of hyper-rectangles that represent a model of the data. From an initial set *HS* of hyper-rectangles, each individual of the evolutionary population represents a subset of *HS*, which are evolved until the best individual is found, after several generations. This initial set *HS* is built before the evolutionary algorithm is run, and it is not modified throughout the execution of the evolutionary algorithm. The first disadvantage is therefore that the end result depends on how the initial set *HS* is built.

On the other hand, the fitness of an individual is assessed by searching, for each element x of the database used, the hyper-rectangle that is closest to x. This means that, for each assessment of the fitness of each individual, the entire database must be examined. In Garcia et al. (2009), it is mentioned that for the tests carried out, one run of the evolutionary algorithm consists in 10 000 assessments, which means that the entire database is examined 10 000 times. From a more optimistic viewpoint, this technique may achieve an optimal result in a shorter amount of time. Either way, for a population of 100 individuals, the optimal result will hardly be achieved in less than 20 generations, which means that the lower limit would be 2000 assessments of individuals to achieve the optimal result, which in turn means that the database has to be examined 2000 times.

In Holden and Freitas (2008), a conventional PSO algorithm is used with some modifications to find rule clauses. Each particle vector has two values for each attribute – one for the lower limit and one for the upper limit. To assess a particle, its vector is turned into a rule that is then assessed.

The rule extraction algorithm starts with an empty set of rules RS and then, for each class, finds the corresponding classification rules. All the data from class C are stored in a set TS.

The particle with the best solution found by PSO, after being simplified, is added to the RS set, and all data that meet the rule are removed from the TS set. Thus, the PSO/ACO2 algorithm finishes one loop and continues with the iterative process of applying PSO/ACO2 to all remaining data and completing the rule returned with its continuous attribute clauses. This iterative process continues until the data that still remain in TS are lower than a threshold.

The first time that the PSO algorithm tries to find a rule, it works with all the data in a class. Assessing the fitness of a particle implies exploring the entire database and measuring the accuracy of the rule represented by the vector of

the particle. The result of this operation is a rule, and all the data that meet this rule are not analyzed when looking for a second rule.

Since there is no way of knowing the subset of data that is assessed over and over again as the rules are generated, the lower limit will be considering that for each class, an only rule is extracted. It is therefore established that for each class, the PSO algorithm is run once with the data from that class.

In Holden and Freitas (2008), it is explained that the runs of PSO are carried out with a swarm of 100 particles, and that the algorithm is run a maximum of 100 iterations. In the best of cases (which is almost impossible) PSO runs an only iteration to obtain the optimal result, meaning that it would have had to assess the fitness value of 100 particles and that each of these assessments explored the data in one class. If for each class, an only PSO is run with only one iteration (hypothetical case that is almost impossible), then the database had to be explored in its entirety at least 100 times.

This is the minimum number of times that the database is explored in its entirety. More realistically, assuming that two rules are extracted for each class, with the second rule being built only with 50% of the data from each class, and each PSO running 20 iterations, then the database would have to be explored 2000 to find the first rule and 1000 times to find the second one (since it would be working with half the data), for a total of 3000 times that the database is explored.

With CLUHR, the data is explored in its entirety only the first time to build the hyper-rectangles, a second time to determine how many data fall within each intersection, and a third time to calculate the indexes and determine how to remove the intersection selected. In summary, for each intersection that is to be removed, the database is explored three times. The total number of times that the database is explored will be $3*Q$, where Q is the number of intersections that is removed during the execution of the algorithm.

When removing an intersection, only the hyper-rectangles involved are modified, which means that only the data represented by those hyper-rectangles are explored. These data, in each intersection q, represent a proportion f_q of the database, so the database itself is explored $3*Q*f_q$ times.

Table 3 shows, for each database used for the tests, the number of times that the database was explored.

Table 3. Number of times the entire database is explored

Dataset	EHS-CHC	PSO/ACO2	CLUHR
Ecoli	2000	3000	4.65 (0.15)
Glass	2000	3000	5.37 (0.18)
Haberman	2000	3000	2.54 (0.06)
Image	2000	3000	3.74 (0.10)
Ionosphere	2000	3000	5.17 (0.18)
Iris	2000	3000	2.08 (0.05)
Liver	2000	3000	5.01 (0.06)
Pima	2000	3000	5.27 (0.12)
Sonar	2000	3000	16.27 (0.72)
Vehicle	2000	3000	7.38 (0.33)
Vowel	2000	3000	8.13 (0.27)
Wine	2000	3000	4.08 (0.09)
Wisconsin	2000	3000	3.59 (0.11)

Description: Number of times the entire database is explored with algorithms EHS-CHC, PSO/ACO2 and CLUHR. For CLUHR, the average and the standard deviation for 10 runs are shown. For the other two techniques, the number of times estimated in this section is used.

7. Conclusions and Future Work

A new classification technique that generates hyper-rectangles from input data has been presented. These hyper-rectangles are reduced to smaller ones based on the results of the calculation of a battery of indexes, until overlapping areas are eliminated or minimized.

The use of indexes as criterion to select two overlapping hyper-rectangles to be modified and thus remove such overlap turns the proposed technique into a robust and efficient tool, in the sense that certain indexes can be calculated or not depending on the interests of the end user, including the possibility of adding new indexes resulting from new research activities or future experiences with problems that are solved using this method.

The pair of hyper-rectangles to divide can be selected by an expert in the domain area while the model is being built. If the expert supervises the learning process, the next pair of hyper-rectangles that are to be divided and the dimension on which the division will be done would be visible at all times. The expert could have a system that shows the results of performing the same division using different dimensions, or the results that would be obtained by dividing different overlaps at a given point in time. This system could even show the expert three or four more divisions in time, showing an entire tree of possible results, which would be helpful to make decisions as to how to create the model.

The results obtained were compared with an evolutionary technique and an optimization technique, and it was observed that a better accuracy and a slightly smaller number of extracted rules were achieved. The greatest advantage of CLUHR, when compared with the strategies mentioned, is that it requires a significantly lower computational effort to achieve similar results. This is very important when working with large databases.

References

Aslanidis, T., Souliou, D., & Polykrati, K. (2008). *CUZ, An Improved Clustering Algorithm.* Computer and Information Technology Workshops IEEE 8th International Conferenc, 43-48. http://dx.doi.org/10.1109/CIT.2008.Workshops.118

Bakar, A. A., Othman, Z. A., Hamdan, A. R., Yusof, R., & Ismail, R. (2008). *An Agent Based Rough Classifier for Data Mining.* Intelligent Systems Design and Applications Eighth International Conference, 145-151. http://dx.doi.org/10.1109/ISDA.2008.29

Bandyopadhyay, S., & Saha, S. (2009). A Point Symmetry-Based Clustering Technique for Automatic Evolution of Clusters. *IEEE Transactions on Knowledge and Data Engineering, 20*(11), 1441-1457. http://dx.doi.org/10.1109/TKDE.2008.79

Bouguessa, M., & Wang, S. (2009). Mining Projected Clusters in High-Dimensional Spaces. *IEEE Transactions on Knowledge and Data Engineering, 21*(4), 507-522. http://dx.doi.org/10.1109/TKDE.2008.162

Cao, L. (2010). Domain-Driven Data Mining: Challenges and Prospects. *IEEE Transactions on Knowledge and Data Engineering, 22*(6), 755-769. http://dx.doi.org/10.1109/TKDE.2010.32

Chen, H. L., Chuang, K. T., & Chen, M. S. (2009). On Data Labeling for Clustering Categorical Data. *IEEE Transactions on Knowledge and Data Engineering, 20*(11), 1458-1472. http://dx.doi.org/10.1109/TKDE.2008.81

Darrah, M., Taylor, B., & Skias, S. (2004). *Rule Extraction from Dynamic Cell Structure Neural Networks Used in a Safety Critical Application.* Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, 629-634.

Frank, A., & Asuncion, A. (2010). *UCI Machine Learning Repository.* Irvine, CA: University of California, School of Information and Computer Science. Retrieved from http://archive.ics.uci.edu/ml

Garcia, S., Derrac, J., Luengo, J., & Herrera, F. (2009). *A First Approach to Nearest Hyperrectangle Selection by Evolutionary Algorithms.* Intelligent Systems Design and Applications Ninth International Conference, 517-522.

Hasperué, W., Osella, M. G., & Lanzarini, L. (2008). *Obtaining a Fuzzy Classification Rule System from a non-supervised Clustering.* Information Technology Interfaces (ITI) 30th International Conference, 341-346.

Hasperué, W., & Lanzarini, L. C. (2010). *A new clustering strategy for continuous datasets using hypercubes.* 36th Conferencia Latinoamericana de Informática.

Holden, N., & Freitas, A. A. (2008). A hybrid PSO/ACO algorithm for discovering classification rules in data mining. *Journal of Artificial Evolution and Applications, 2008*, 1-11. http://dx.doi.org/10.1155/2008/316145

Hsu, C. M., & Chen, M. S. (2009). On the Design and Applicability of Distance Functions in High-Dimensional Data Space. *IEEE Transactions on Knowledge and Data Engineering, 21*(4), 523-536. http://dx.doi.org/10.1109/TKDE.2008.178

Kamwa, I., Samantaray, S. R., & Joos, G. (2009). Development of Rule-Based Classifiers for Rapid Stability Assessment of Wide-Area Post-Disturbance Records. *IEEE Transactions on Power Systems, 24*(1), 258-270. http://dx.doi.org/10.1109/TPWRS.2008.2009430

Konig, R., Johansson, U, & Niklasson, L. (2007). *Genetic programming - a tool for flexible rule extraction.* Evolutionary Computation IEEE Congress, 1304-1310.

Koul, N., Caragea, C., Honavar, V., Bahirwani, V., & Caragea, D. (2008). *Learning Classifiers from Large Databases Using Statistical Queries.* Web Intelligence and Intelligent Agent Technology IEEE/WIC/ACM International Conference, 923-926.

Lu, J. L., Wang, L. Z., Lu, J. J., & Sun, Q. Y. (2008). *Research and application on KNN method based on cluster before classification.* Machine Learning and Cybernetics International Conference, 307-313.

Ma, J., Guo, D., Liu, M., Ma, Y., & Chen, S. (2009). *Rules Extraction from ANN Based on Clustering.* Computational Intelligence and Natural Computing International Conference on, 19-21.

Martens, D., Baesens, B., & Van Gestel, T. (2009). Decompositional Rule Extraction from Support Vector Machines by Active Learning. *IEEE Transactions on Knowledge and Data Engineering, 21*(2), 178-191. http://dx.doi.org/10.1109/TKDE.2008.131

Salzberg, S. (1991). A Nearest Hyperrectangle Learning Method. *Machine Learning, 6*, 251-276. http://dx.doi.org/10.1007/BF00114779

Shi, X. J., & Lei, H. (2008). *A Genetic Algorithm-Based Approach for Classification Rule Discovery.* Information Management, Innovation Management and Industrial Engineering International Conference, 175-178. http://dx.doi.org/10.1109/ICIII.2008.289

Thornton, C. (1987). *Hypercuboid-Formation Behaviour of Two Learning Algorithms.* Artificial Intelligence International Joint Conference, 301-303.

Wang, Z., Qi, Q., & Xu, L. (2008). *Cluster Analysis Based on Spatial Feature Selecting in Spatial Data Mining.* Computer Science and Software Engineering International Conference, 386-389. http://dx.doi.org/10.1109/CSSE.2008.1334

Wei, J. M., Wang, S. Q., & Yuan, X. J. (2010). Ensemble Rough Hypercuboid Approach for Classifying Cancers. *IEEE Transactions on Knowledge and Data Engineering, 22*, 381-391. http://dx.doi.org/10.1109/TKDE.2009.114