CLUIN – A New Method for Extracting Rules for Large Databases

Waldo Hasperué[1,2] and Leonardo Corbalan[1]

[1]Instituto de Investigación en Informática LIDI – School of Computer Science – National University of La Plata, Argentina.
[2] CONICET scholarship
{whasperue, corbalan}@lidi.info.unlp.edu.ar

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, the CLUHR classification strategy is extended to work with databases that have nominal attributes. 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, large datasets, supervised learning.*

1 Introduction

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 [1] [2] [3].

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 [4] [5] [6] [7] [8]. 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 [9] [10] [3].

IF-THEN rules are the most common way of passing knowledge, since they are easy to understand. Additionally, adaptive techniques allow adding new rules and removing or modifying existing ones. This is why rules are used by most existing techniques to produce knowledge [10] [11] [12] [13] [14] [15].

In [16], a powerful extraction strategy that extracts knowledge as IF-THEN rules is presented, called CLUHR. This strategy uses hyper-rectangles as tool to describe the characteristics of the various data classes. The greatest disadvantage of CLUHR is

that it can only be used in databases whose attributes are numeric, since hyper-rectangles can only be handled in numeric domains. This disadvantage greatly reduces the scope of the strategy, since there are not many databases with only numeric attributes.

In this paper, and extension of the CLUHR technique, called CLUIN, is presented; CLUIN uses hyper-rectangles and overlap indexes for handling numeric attributes, adding the ability of dealing with nominal or categoric dataset attributes.

The same as CLUHR [16], this technique is deterministic because it no longer depends on absolute randomness, and also, response time in model building is lower than with other classification techniques, achieving similar results regarding the accuracy of the model built and the number of rules extracted.

The rest of the paper is organized as follows: in Section 2, the strategy proposed and the CLUIN algorithm are presented. In Section 3, the results obtained are detailed. Finally, the conclusions drawn from this work are discussed in Section 4.

2 CLUIN

The strategy proposed in this paper, called CLUIN, uses the same algorithm as CLUHR to work with numeric attributes by adding a new algorithm that is responsible for handling the nominal attributes in the database.

Each of the nominal attributes i in the database has its own domain represented by a set S_i. Thus, the data model of a given class is formed by a hyper-rectangle that describes the numeric attributes of the class and a set S, called set of nominal values, that is formed by sets S_i.

Definition: Be C a class of data and $value(x,i)$ the nominal value of attribute i for element $x \in C$. Set S_i for C is formed as follows: $S_i = \{ v \mid \exists x \in C, value(x,i) = v \}$

Definition: Be C a class of data and S_i, i=1..N, the corresponding sets of the N nominal attributes of class C. The set of nominal values S is formed by all S_i: $S = \{ S_i, i=1..N \}$

Definition: Be C a class with N nominal attributes and M numeric attributes, the data in class C are represented by the set of nominal values S formed from the N nominal attributes and hyper-rectangle H formed from the M numeric attributes.

In CLUHR, two hyper-rectangles whose classes are different can present an overlap in space, the same as two sets of nominal values from different classes can intersect.

Definition: Be S and T two sets of nominal values in classes C and D, respectively. There is an intersection between both sets if the intersection of all pairs of sets S_i and T_i is different from the empty set.

From the definition above, it can be seen that with just one attribute i for which $S_i \cap T_i = \varnothing$, classes C and D will not intersect. Attribute i is precisely the one that allows splitting data from both classes.

Definition: Two classes C and D overlap if their respective hyper-rectangles overlap and their respective sets of nominal values also overlap.

Therefore, the overlapping classes can be split just by dividing the sets of nominal values or the hyper-rectangles.

The intersections detected in the sets of nominal values must be removed; to do so, similarly to CLUHR, certain indexes are calculated. In CLUHR, divisibility indexes Z_i are calculated, which are applicable to each numeric attribute. These indexes Z_i are used to calculate the divisibility index Ω_i, which in turn determines the attribute to be used for the adjustment. CLUIN uses the same index Ω_i for numeric values and also uses the divisibility indexes Y_j that are calculated for each nominal attribute and are used for calculating the divisibility index Ψ_j. The maximum value between Ψ_j and Ω_i determines the attribute to be used for the *splitting* operation.

If the attribute is numeric, hyper-rectangles are divided as described for CLUHR. If the attribute is nominal, the division is done by modifying the sets of nominal values as detailed further on.

2.1 Indexes

In CLUHR, overlap indexes are calculated to measure how much two hyper-rectangles overlap in each space dimension. This index is calculated for all space dimensions and for each of the two hyper-rectangles. Thus, if two hyper-rectangles H and J overlap, the indexes $Z_i(H)$ and $Z_i(J)$ are calculated for i=1..M.

Similarly, CLUIN calculates the intersection indexes for each nominal attribute and for each of the intersected sets of nominal values S and T. Thus, indexes $Y_j(S)$ and $Y_j(T)$ are calculated for j=1..N, and are then used to calculate the divisibility index Ψ_j.

In this paper, we present the use of two indexes Y that have been successfully used in various classification algorithms. The first of these indexes is the Information Gain Ratio used in many decision trees [17] [18] [19] and the second one is Kolmogorov-Smirnoff distance, used in more recent works [20] [21].

Information Gain Ratio: This index determines the nominal value v in an attribute j that has the lowest entropy among all values of i. This index returns a value between 0 and 1, where 0 indicates the greatest possible entropy and the lowest entropy is indicated by the value tending to 1. As this index is calculated using the values from both sets S and T, a single calculation is done, and it is assigned to indexes $Y_j(S)$ and $Y_j(T)$:

$$Y_j(\mathrm{S}) = Y_j(T) = \mathrm{IGR}(S \cup T, j)$$

$$\mathrm{IGR}(ST, j) = \mathrm{IG}(ST, \mathrm{j}) \,/\, \mathrm{IV}(ST, j)$$

$$\mathrm{IG}(ST,i)=\mathrm{H}(ST)- \sum_{v \in valores(i)} \frac{\#\{x \in ST \mid valor(x,i)=v\}}{\#ST} . H(\{x \in ST | valor(x,i) = v\})$$

$$\mathrm{IV}(ST, i) = - \sum_{v \in valores(i)} \frac{\#\{x \in ST \mid valor(x,i)=v\}}{\#ST} . log_2(\frac{\#\{x \in ST \mid valor(x,i)=v\}}{\#ST})$$

Where H(E) is the entropy of set E.

Once the attribute with the highest value of Y is selected, the nominal value v is determined; this value is used to carry out the *split* operation that has the lowest entropy.

Kolmogorov-Smirnoff distance: This distance is a statistical test that finds the maximum distance between two probability functions. When using this index, the division of the two sets of nominal values is carried out by means of the nominal value that generates the greatest division of data from both classes. This index tends

to 1 when the use of a value generates a better division of the classes, and it tends to 0 in the opposite case.

$$Y_i(S) = \max_{v \in valores(i)}(\#\{x \in S \mid valor(x,i) = v\} - \#\{x \in S \mid valor(x,i) \neq v\})$$
$$Y_i(T) = \max_{v \in valores(i)}(\#\{x \in T \mid valor(x,i) = v\} - \#\{x \in T \mid valor(x,i) \neq v\})$$

Once the attribute j with the highest value of Y is selected, the nominal value v is determined; this value is used to carry out the split operation that generated the value of the index (that with the maximum distance).

These two indexes are used for calculating the divisibility index Ψ_i.

$$\Psi_i(H, S) = \frac{\sum_{i=1}^{n} Y_i(S)}{n}$$

Thus, the maximum between Ψ_j and Ω_i determines the attribute to be used to split the classes. If the attribute is numeric, the split operation is carried out by modifying the respective hyper-rectangles from each class as in CLUHR, while if the attribute is nominal, then the split process is carried out by modifying the sets of nominal values.

If the attribute to be used for the division is nominal, then it is possible that each of the calculated indexes Y may have yielded a different value for the division. This strategy does not suggest the use of any specific value over the others; any of them can be used. The value chosen to be used for the *split* process will depend on the problem to be solved, and it is determined by the user.

To carry out a division by modifying a nominal attribute with a value v, one of the two sets S_j is arbitrarily selected and two new sets are formed for class C. One of them will be formed by all data such that for attribute j it is equal to v, and the other one will be formed by its complement.

$$S_j1 = \{ x \mid \text{value}(x, j) = v \}$$
$$S_j2 = \{ x \mid \text{value}(x, j) \neq v \}$$

2.2 Regrouping sets of nominal values and hyper-rectangles.

If the split operation is carried out using a numeric attribute, the hyper-rectangles from both classes are modified and new hyper-rectangles are generated. Each new hyper-rectangle has a new set of corresponding data from the class with which the minimum representative hyper-rectangle is adjusted (see [16] for more details). At the same time, with each data subset, the corresponding set of nominal values is calculated.

Similarly, if the division is carried out using a nominal attribute, the new sets of nominal values obtained will each have a subset of data. With these same data, the corresponding minimum representative hyper-rectangles are formed.

After the adjustments are made, there is a new search for overlaps and indexes Ω_i and Ψ_j are calculated once again.

The CLUIN algorithm is as follows:

```
Initialize hyper-rectangles and sets of nominal values
   for each class of data
Detect overlaps and calculate indexes Ω_i
while there are overlaps
   Divide by hyper-rectangles or by sets of nominal values
   Adjust the new hyper-rectangles and sets of nominal
     values
   Calculate indexes Ω_i
end while
Extract resulting rules
```

2.3 Rule extraction

When the algorithm finishes, the result obtained are the hyper-rectangles and the sets of nominal values that form the entire data model. Each data class is mapped to one or more pairs of hyper-rectangles and sets of nominal values. Each pair of hyper-rectangle/set will be used to extract a classification rule. This rule will be formed by the limits of the hyper-rectangle itself for numeric attributes and the nominal values contained in the set of nominal values.

The clauses of the numeric attributes will be formulated as: (value_attribute_i $\geq$ Hni) AND (value_attribute_i $\leq$ hx1).

While the clauses of the nominal attributes can have either of two formulations: (value_attribute_i = nominal_value) or (value_attribute_i $\in$ subset_of _nominal_values)

For high-dimension problem spaces, this procedure produces very complex and inadequate rules if the purpose is extracting knowledge from the base that the user can understand and explain knowledge through these rules.

The conditions of these complex rules can be refined by simple inspection; the problem, however, is when there is a large number or dimensions of rules. The reader can find methods to automatically simplify rules in the bibliography ([22] [23]).

3 Results

CLUIN, the technique proposed in this paper, has been tested on 13 databases of the UCI repository [24]. Both performance and the computational effort required to build the model were compared with the results presented by [25] and [26]. The comparative analysis of the results obtained by CLUHR in [16] and CLUIN is not necessary, because the latter inherits the same classification strategy used by CLUHR for handling databases with numeric attributes, and therefore, the same results are obtained.

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 CLUIN and the results presented by [25] and [26] are shown in Table 1. Table 2 shows the average number of rules 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 CLUIN and PSO/ACO2 are statistically significant. In tables 1 and 2, the signs "+" or "-" are used to indicate if CLUIN 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 [25] does not include statistical data and it was therefore impossible to compare it.

Based on tables 1 and 2, it can be concluded that, even though the efficacy of the method proposed is similar to those presented for the remaining techniques, the number of hyper-rectangle/set pairs generated is lower. This reduces the number of rules and facilitates understanding the knowledge extracted by the user.

The techniques proposed in [25] and [26], 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 [25], 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 hyper-rectangle is built using both numeric and nominal attributes – when a dimension *i* of the hyper-rectangle corresponds to a nominal attribute, this dimension is built with a subset of values of attribute *i*, converting the hyper-rectangle into a structure that is not numerically complete. 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 [25], 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. Optimistically, it can be assumed that this technique will achieve an optimal result in a shorter 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 2,000 assessments of individuals to achieve the optimal result, which in turn means that the database has to be examined 2,000 times.

In [26], a hybrid algorithm between PSO and ACO is used to find rule clauses. ACO is used to find the clauses of the nominal attributes, while PSO is used to find the clauses of the numeric attributes. In ACO, each particle has a pheromone matrix for each nominal attribute, and in PSO, each particle vector has two values for each attribute – one for the lower limit and one for the upper limit. To test a particle in the pheromone matrix, the nominal clauses are chosen probabilistically and the vector in PSO is converted into clauses that are then tested.

The first time that the PSO algorithm tries to find a rule, it works with all the data in a class. Both in ACO and PSO, 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 which subset of data is assessed over and over again as the rules are generated, the lower limit is set by considering that for each class, a single rule is extracted. It is therefore established that for each class, the ACO algorithm is run once and the PSO algorithm is run once with the data from that class.

In [26], it is explained that the runs of ACO and PSO are carried out with a swarm of 100 particles, and that the algorithms are run a maximum of 100 iterations. In ACO, the 100 iterations are always run to obtain the pheromone matrixes that yield the best possible result; in each iteration rule quality has to be measured, which requires going through the entire database. Therefore, one run of ACO goes through the database 100 x 100 = 10,000 times.

As regards PSO, in the best of cases PSO runs a single iteration to obtain the optimal result. To do so, it has to assess the fitness of 100 particles by going through all of the data in a class for each of these assessments. If for each class, a single PSO is run with only one iteration (hypothetical case that is almost impossible), then the database has to be explored in its entirety at least 100 times.

ACO's 10,000 times added to PSO's 100 times are the minimum number of times that the database is explored in its entirety. More realistically, assuming that two rules are extracted for each class in PSO, 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 2,000 to find the first rule and 1,000 times to find the second one (since it would be working with half the data), for a total of 3,000 times that the database is explored in full. This plus the 10,000 times required for ACO adds up to a total of 13,000 times.

With CLUIN, the data is explored in its entirety only the first time to build the hyper-rectangles and the nominal data sets, 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 an intersection is removed, only the hyper-rectangles and the sets of nominal data involved are modified, and therefore, only the data represented by such pairs of hyper-rectangle/set are explored. At each intersection q, these data represent a fraction f_q of the database, meaning that the database is explored $3*Q*fq$ times.

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

Table 1. Accuracy of the method proposed versus the results presented in [25] and [26]. Standard deviation is indicated between brackets; statistical differences are shown in the last column.

Dataset	EHS-CHC	PSO/ACO2	CLUIN	
Contraceptive	0.4983		0.4852 (0.0265)	
Credit	0.8464	0.8560 (0.0284)	0.8497 (0.0541)	=
Zoo	0.9300	0.9718 (0.0625)	0.9621 (0.0357)	=
Balance scale		0.8272 (0.0477)	0.8236 (0.0219)	=

Australian credit		0.8531 (0.0414)	0.8479 (0.0387)	=
German credit		0.6790 (0.0582)	0.6802 (0.0468)	=
Statlog heart		0.8111 (0.0616)	0.8257 (0.0521)	=
Mushroom		0.9990 (0.0110)	0.9742 (0.0205)	-
Promoter		0.8100 (0.1212)	0.8351 (0.0981)	=
Soybean		0.8701 (0.0653)	0.8594 (0.0782)	=
Tic-Tac-Toe		1.000 (0.0000)	0.9863 (0.0029)	-
Chess (Kr vs. Kp)		0.9947 (0.0510)	0.9746 (0.0268)	=
Splice		0.9348 (0.0124)	0.9420 (0.0254)	=

Table 2. Number of hyper-rectangle/set pairs created for the data model by the strategies studied. Standard deviation is indicated between brackets; statistical differences are shown in the last column.

Dataset	EHS-CHC	PSO/ACO2	CLUIN	
Contraceptive	12.7		11.8 (3.27)	
Credit	6.1	22.5 (3.1)	7.9 (3.80)	+
Zoo	5.6	7.1 (0.32)	6.3 (1.32)	=
Balance scale		26.6 (1.07)	29.4 (2.92)	-
Australian credit		22.7 (2.0)	25.14 (4.84)	=
German credit		54.3 (1.89)	48.67 (1.51)	+
Statlog heart		9.7 (1.34)	10.8 (1.63)	=
Mushroom		8.7 (0.48)	6.5 (0.87)	+
Promoter		5.1 (0.32)	6.3 (0.49)	-
Soybean		24.2 (1.03)	22.1 (2.07)	+
Tic-Tac-Toe		9.0 (0.0)	8.06 (0.62)	+
Chess (Kr vs. Kp)		18.7 (2.0)	25.8 (3.98)	-
Splice		88.0 (2.91)	95.7 (4.79)	-

Table 3. Number of times the entire database is explored with algorithms EHS-CHC, PSO/ACO2 and CLUIN. For CLUIN, the average and the standard deviation for 10 runs are shown. For the other two techniques, the number of times estimated in section 4, Results, is used.

Dataset	EHS-CHC	PSO/ACO2	CLUIN
Contraceptive	2000		25 (5.4)
Credit	2000	13000	47 (4.2)
Zoo	2000	13000	15 (3.2)
Balance scale		13000	32 (1.8)
Australian credit		13000	54 (4.8)
German credit		13000	26 (5.4)
Statlog heart		13000	41 (3.9)
Mushroom		13000	35 (4.8)
Promoter		13000	28 (2.8)
Soybean		13000	52 (8.2)
Tic-Tac-Toe		13000	14 (3.7)
Chess (Kr vs. Kp)		13000	15 (5.3)
Splice		13000	61 (7.1)

5 Conclusions

A new knowledge extraction strategy has been presented, called CLUIN, that extends the strategy proposed by CLUHR [16] to work with nominal attributes. Each hyper-rectangle generated with the numeric attributes corresponds to a set of nominal values. This hyper-rectangle/set pairs are reduced to other, smaller ones as indicated by the result of calculating a battery of indexes to remove or minimize existing overlaps.

The use of indexes as criterion to select two overlapping hyper-rectangles or sets that intersect in order to be modified and thus remove such overlap turns the proposed strategy 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.

CLUIN improves the power of CLUHR by adding the possibility of handling nominal attributes. CLUIN follows the same line as CLUHR; namely, it uses the same definition of overlapping sets of nominal values and how to detect these overlaps, and calculates indexes to determine the degree of intersection between sets of different classes used together with the intersection indexes used in CLUHR to calculate the overlap index. The latter indicates the attribute that should be used to minimize the overlap between the data from both classes.

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

1. Bouguessa, M., Wang, S. Mining Projected Clusters in High-Dimensional Spaces. IEEE Transactions on Knowledge and Data Engineering, 21 (4), 507-522. (2009)
2. Hsu, C.-M., Chen, M.-S. On the Design and Applicability of Distance Functions in High-Dimensional Data Space. IEEE Transactions on Knowledge and Data Engineering, 21 (4), 523-536. (2009).
3. Koul, N., Caragea, C., Honavar, V., Bahirwani, V., Caragea, D. Learning Classifiers from Large Databases Using Statistical Queries. Web Intelligence and Intelligent Agent Technology IEEE/WIC/ACM International Conference on, 923-926. (2008).
4. Bandyopadhyay, S., Saha, S. A Point Symmetry-Based Clustering Technique for Automatic Evolution of Clusters. IEEE Transactions on Knowledge and Data Engineering, 20 (11), 1441-1457. (2009).
5. Chen, H.-L., Chuang, K.-T., Chen, M.-S. On Data Labeling for Clustering Categorical Data. IEEE Transactions on Knowledge and Data Engineering, 20 (11), 1458-1472. (2009).
6. Lu, J.-L., Wang, L.-Z., Lu, J.-J., Sun, Q.-Y. Research and application on KNN method based on cluster before classification. Machine Learning and Cybernetics International Conference on, 307-313. (2008).

7. Wang, Z., Qi, Q., Xu, L. Cluster Analysis Based on Spatial Feature Selecting in Spatial Data Mining. Computer Science and Software Engineering International Conference on, 386-389. (2008).
8. Aslanidis, T., Souliou, D., Polykrati, K. CUZ, An Improved Clustering Algorithm. Computer and Information Technology Workshops IEEE 8th International Conference on, 43-48. (2008).
9. Bakar, A.A., Othman, Z.A., Hamdan, A.R., Yusof, R., Ismail, R. An Agent Based Rough Classifier for Data Mining. Intelligent Systems Design and Applications Eighth International Conference on, 145-151. (2008).
10. Kamwa, I., Samantaray, S. R., Joos, G. Development of Rule-Based Classifiers for Rapid Stability Assessment of Wide-Area Post-Disturbance Records. IEEE Transactions on Power Systems, 24 (1), 258-270. (2009).
11. Martens, D., Baesens, B., Van Gestel, T. Decompositional Rule Extraction from Support Vector Machines by Active Learning. IEEE Transactions on Knowledge and Data Engineering, 21 (2). 178-191. (2009).
12. Shi, X.-J., Lei, H. A Genetic Algorithm-Based Approach for Classification Rule Discovery. Information Management, Innovation Management and Industrial Engineering International Conference on, 175-178. (2008).
13. Hasperué, W., Osella Massa, G., Lanzarini, L. Obtaining a Fuzzy Classification Rule System from a non-supervised Clustering. Information Technology Interfaces (ITI) 30th International Conference of, 341-346. (2008).
14. Konig, R., Johansson, U, Niklasson, L. Genetic programming - a tool for flexible rule extraction. Evolutionary Computation IEEE Congress on, 1304-1310. (2007).
15. Hasperué, W., Lanzarini, L.C. A new clustering strategy for continuous datasets using hypercubes. 36th Conferencia Latinoamericana de Informática. (2010).
16. Hasperué, W., Lanzarini, L., De Guisti, A. Rule Extraction on Numeric Datasets Using Hyper-rectangles. Computer and Information Science. Vol. 5, No 4, pp. 116 131. (2012).
17. Quinlan J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, Inc. ISBN 1-55860-238-0. (1993).
18. Quinlan J. R. Induction of Decision Trees Machine Learning. 1: Vol. 1. - págs. 81-106. (1986).
19. Piao M., Li M. y Ryu K. Ho Using Significant Classification Rules to Analyze Korean Customers' Power Consumption Behavior: Incremental Tree Induction using Cascading-and-Sharing Method. págs. 1649-1653. (2010).
20. Mballo, C., Diday E. Kolmogorov-Smirnov for Decision Trees on Interval and Histogram Variables. (2004).
21. Chao S., Wong Fai. An incremental decision tree learning methodology regarding attributes in medical data mining. International Conference on Machine Learning and Cybernetics. págs. 1694-1699. (2009)
22. Darrah, M., Taylor, B., Skias, S. 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. (2004).
23. Ma, J., Guo, D., Liu, M., Ma, Y., Chen, S. Rules Extraction from ANN Based on Clustering. Computational Intelligence and Natural Computing International Conference on, 19-21. (2009).
24. A. Asuncion and D. Newman, UCI machine learn-ing repository, Available: http://archive.ics.uci.edu/ml/ (2007).
25. Garcia, S., Derrac, J., Luengo, J., Herrera, F. A First Approach to Nearest Hyperrectangle Selection by Evolutionary Algorithms. Intelligent Systems Design and Applications Ninth International Conference on, 517-522. (2009).
26. Holden, N., Freitas, A.A. A hybrid PSO/ACO algorithm for discovering classification rules in data mining. Journal of Artificial Evolution and Applications, 1-11. (2008).