ADT: A decision tree algorithm based on concepts

Anilu Franco-Arcega[1], Guillermo Sánchez-Díaz[1], José Ruiz-Shulcloper[2] [1]*Center of Technologies Research on Information and Systems, UAEH, Carr. Pachuca-Tulancingo Km. 4.5, C.P. 42084, Pachuca, Hgo., Mexico,* [2]*Center of Applications of Advanced Technologies, Havana, Cuba*
anifranco6@yahoo.com.mx, sanchezg@uaeh.reduaeh.mx, jshulcloper@cenatav.co.cu

***Abstract*— In this paper, a new method named Alternative Decision Trees (ADT) for the generation of decision trees is introduced. This proposed method generated a decision tree based in concepts of minimum covering, obtained of concepts or properties of each class described in the data set. Starting from the limitations of others decision tree algorithms, we worked in the development of new techniques that allow us to improve those limitations. A simplification of these concepts, and a split criterion are introduced. Besides, the performance of ADT method is presented.**

***Keywords*— decision trees, conceptual algorithms, classification.**

I. Introduction

One of the tools for solve the problem of classification is the based in decision trees [1], [2]. These tools have been used successfully in diverse areas such as radar signal classification, character recognition, remote sensing, medical diagnosis, expert systems, speech recognition and others [3].

Perhaps, the most important characteristic of decision trees is their capability to break down a complex decision-making process into a collection of simpler decisions, thus providing a solution which is often easier to interpret.

The methods usually use a training test for generating the decision tree. A decision tree classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance [4]. Each node in the tree specifies a test of some feature of the instance, and each branch descending from that node corresponds to one of the possible values for this feature. An instance is classified by starting at the root node of the tree, testing the feature specified by this node, then moving down the tree branch corresponding to the value of the feature in the given instance. This process is repeated for the subtree rooted at the new node.

Decision trees are made of two major components: a procedure to build the symbolic tree, and an inference procedure for decision making. In some algorithms for building decision trees, the resulting tree can be pruned, which often leads to improved generalization by incorporating additional bias [5].

There are several algorithms that allow us to build decision trees of a data set [6]. These algorithms, use data set straightforward for generating the tree. In addition, some of them can not work with mixed data sets [7].

This paper proposed a new method for generating decision trees using concepts of minimum covering, and is organized as follow. Related works about decision trees are present in Section 2. We analyzed them, for detecting their limitations and overcome them. Main basic concepts related with supervised classification and decision trees are exposed in Section 3 . The proposed method and the new techniques developed are described in Section 4. Experimental results are presented in Section 5. Finally, conclusions are provided in Section 6.

II. Related works

Currently, three kinds of algorithms in order to generated decision trees have been developed: Algorithms for qualitative data, quantitative data, and mixed data.

A. Algorithms for qualitative data

ID3 [4] is the most known algorithm in this kind. However, there is another algorithm very similar to him, the k-dimensional algorithm [8], both of them use the data set to build the tree. The main difference between ID3 and k-d is the split rule used to generate the nodes. ID3 use the information gain, and k-d use confusion induce by a feature. Other method that only works with qualitative data is a fuzzy decision tree algorithm that is only an extension of ID3 [5], and it use memberships functions to represent the values of a feature. All of these algorithms based its construction in the methodology top-down.

B. Algorithms for quantitative data

These kind of algorithms allow us to manipulate data sets describe by quantitative data. For processing this kind of data, the algorithms applied a discretization process that transform the values of the features in intervals or ranges. The methodology to build the tree is the top-down induction. C4.5 [6], and CART [9] are the most representative algorithms of this family. Both of them applied a pruning process at the end of the construction, and their split rule are base in the gain information use by ID3. The processes of these techniques are known as gain proportion, and Gini diversity index, respectively. Other significative difference between C4.5 and CART is that the second only generates binary decision trees.

Others algorithms found in this family are FACT [10] and QUEST [11]. FACT is the predecessor of QUEST, and their main difference is the number of branches created for each node, FACT form as many branches as number of classes have, and QUEST generate binary trees. These algorithms realize its process of split in two steps, at each node, an analysis of variance F-statistic is calculated for each feature. The feature with largest F-statistic is selected, and a linear discriminant analysis is applied to find the split point selection. It means that look the values that each branch will have.

C. Algorithms for mixed data

Currently, a model that generate decision trees of a mixed data set [12] was proposed. This algorithm introduced the use of support vector machines to the process of construct the decision tree. This tool is used to transform the quantitative data in synthetic boolean features, changing the initial space representation of the features. Besides, it uses the ID3 algorithm to process the qualitative features, in this way the total set of features never is manipulated for only one method.

LMDT is another algorithm that works with mixed data [13], it builds a decision tree in the well known top-down manner. The LMDT algorithm trains a linear machine, which then serves as a multivariate test for the decision trees, this indicated the split rule that it uses to generate the internal nodes. In order to construct that linear machine test, each instance must be represented as a vector consisting of a constant threshold value of ones, and numerically encoded features that describe the objects of data set

D. Limitations found

Main limitations found in the analyzed algorithms are the following:

- All of them used the data set for generated the decision tree. This characteristic could had some problems, because the size of data set can be too large. On the other hand, if in the data set any object is modified, deleted or adding, then generally, the decision tree should be rebuild for cover the new objects in the matrix.
- The algorithms that applied a pruning process do not know with exactitude the stop condition.
- Some of these algorithms, transform the values of the features, applying a discretization process, or changing the initial space for manipulated the data set, for building the tree.

Taking some of the above limitations mentioned, we proposed a new method, which is capable to process mixed data sets, using concepts obtained by a conceptual algorithm, instead of handled data set straightforward for generating the decision tree.

Particulary, we uses a modified RGC algorithm [14], which allows generated concepts of classes of data set. The concepts obtained by RGC have characterizing and excluyent properties. Characterizing property allows to cover all object of all classes of the data set. And excluyent property, has the characteristic of do not confuse any objects descriptions belonging to different classes of data set.

III. Some Basic Concepts

Let U be an universe of objects, and DS = $\{O_1, O_2, \cdots, O_m\}$ a finite set of objects in U, each one described by a set of features $R = \{x_1, x_2, \cdots, x_n\}$. Each feature x_i has associated a set of admissible values, denoted by M_i, $i = 1, \cdots, n$. Features can be of any nature (qualitative: Boolean, multi-valued, etc. or quantitative: integer, real). A description for each object O_i, as a finite sequence of values of features $(x_1(O_i), x_2(O_i), \cdots, x_n(O_i))$ is defined. Besides, for every feature, a comparison criterion $C_r = M_r \times M_r \rightarrow L_r$, where L_r is a total ordered set, is defined too. It is known that U is distributed in a finite number of subsets $K_1, \cdots, K_c$ named classes. Each class is defined by a number of objects.

TABLE I

Learning Matrix with m-objects, n-features, and c-classes

Class	Object	Features		
K_1	O_1	$x_1(O_1)$	$\cdots$	$x_n(O_1)$
	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	O_p	$x_1(O_p)$	$\cdots$	$x_n(O_p)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
K_c	O_q	$x_1(O_q)$	$\cdots$	$x_n(O_q)$
	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	O_m	$x_1(O_m)$	$\cdots$	$x_n(O_m)$

This information about objects and classes is represented in a Learning Matrix (LM) [15] as in Table 1.

Given a LM, where each object is described by a feature set $x_1, x_2, \cdots, x_n$, and a class set $C = C_1, \ldots, C_r$, a decision tree is associated to LM if fulfill the following properties [3]:

- Each internal node is labeled with a feature, x_i.
- Each branch is labeled with a predicate, applied to the selected feature.
- Each leaf or terminal node is labeled with a class, C_j

We known as a split rule to any question that split the training set in at least two subsets, no empty, that allow to any algorithm build a decision tree. Each question is evaluated by some information metric.

Most of the algorithms to build decision trees base its construction in a strategy call Top-down Induction [16], that follows the next procedure: it builds a leaf if all the cases in LM belong to one class, it builds a leaf too, if it can not choose another feature for expand the tree. Finally, it builds an internal node using a criterion for split in subsets to LM.

IV. ADT: Proposed method

Fig. 1 shows the general scheme of the method ADT. The procedure that ADT follows is: taking as input the concepts resulting of RGC, a procedure for obtaining the concepts of minimum covering is applied. Then, that set of concepts is the parameter that a recursive function received for building the final decision tree. Finally, a control matrix (CM) can be classify by the decision tree obtained for checking the performance of ADT.

A. RGC a conceptual algorithm

We choose the conceptual algorithm RGC [14], to generate the concepts that allow us to build the decision tree, because he shows more capabilities than others algorithms. RGC allows the manage of mixed data. Its main property is the generation of concepts that are characterizing and excluyent to the class of LM that cover. The main goal of this algorithm is that given a data set return a conceptual structuring of these objects.

The algorithm calculates a structure named star for each class for each support set, each star are composed by a set

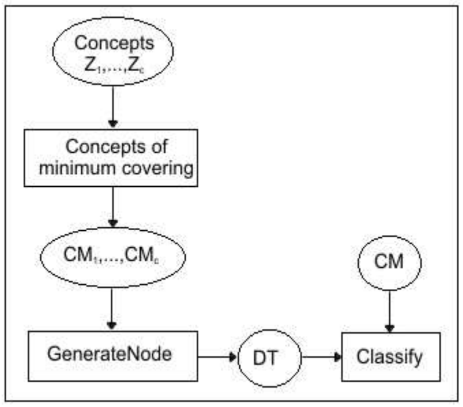

Fig. 1. General scheme of ADT.

of l-complexes that cover each object in the class. Then a generalization of those l-complexes are applied for constructing the final l-complexes $\alpha_1,\ldots,\alpha_q$.

The concept that characterizes the cluster K_i is:

$$Z_i = (\alpha_1 \vee \ldots \vee \alpha_q) \wedge \neg(\alpha_{O_1} \vee \ldots \vee \alpha_{O_t})$$

where α_{O_j}, $j = 1,\ldots,t$ are the elemental l-complexes associated to the objects $O_1,\ldots,O_t \in LM\backslash K_i$ that satisfied the concept $\alpha_1 \vee \ldots \vee \alpha_q$.

B. Concepts of minimum covering

The proposed method is based in the concepts generated by RGC. Because RGC applies a generalization operator, named GEN, the l-complexes lose their excluyent property, that is why RGC build the l-complexed associated to the objects that are cover by the concept but that they belong to a different class. This l-complexes represents a disadvantage for ADT, so that the method proposed only builds decision trees for a LM that not generated l-complexes associated.

Starting from there, we note that having a set of l-complexes as a concept, ADT gives a tree with a high level of depth. For this, a new manner to simplifying the concept obtained by RGC in a concept of minimum covering is proposed.

Definition 1. A concept of minimum covering CM_i is formed of simplifying the set of excluyent and generalized l-complexes that form a concept Z_i in the following way:

```
Given a set of concepts Z,
1. For each $Z_i$, $i = 1,\ldots,c$ do
      Create $CM_i = \emptyset$
      For each $X_j \in \alpha_{i_1}$, $j = 1,\ldots,s$ where $\alpha_{i_1} \in Z_i$ do
         If $\forall \alpha_{i_m}$, $m = 2,\ldots,q$, $R^1_{i_j} = R^m_{i_j}$ then
            $CM_i = CM_i \wedge \left[X_j = R^1_{i_j}\right]$
      For each $\alpha_{i_m}$, $m = 1,\ldots,q$ do
         $\forall\,(X_j \notin CM_i)$
            $CJ_m = \bigwedge_{j=1}^{s} \left[X_{i_j} = R^m_{i_j}\right]$
      $CM_i = CM_i \wedge \{CJ_1 \vee \ldots \vee CJ_m\}$,
```

where, $R^m_{i_j} \subseteq M_j$ is the reference set of values of the feature j in the l-complex m of concept i.

C. Split criterion

Most of the algorithms based their split criterion in the gain information, several criterions are only modifications of that metric. Analyzing it, we noted that this metric look for the homogeneity that exist in each class of LM. The main goal of this metric is to find the confusion that a feature induce to classify the objects in LM.

So, a new metric that can find the separability that exist among all the classes in LM is proposed. With this criterion each class can be found in only one path of each internal node of the decision tree. Besides, the proposed split criterion is based in the concepts of minimum covering, not in the LM.

The algorithm to obtain the result of this rule is describe below:

```
Input: $CCM = \{CM_1,\ldots,CM_c\}$
Output: VS
1. Create $CJT = \emptyset$.
2. For each $CM_i$ do
      For each $CJ_p \in CM_i$, $p = 1,\ldots,card(CM_i)$ do
         $CJT = CJT \cup CJ_p$
3. For each $CJ_t \in CJT$, $t = 1,\ldots,card(CJT)$ do
      $VS_t = ST(CJ_t)$,
```

where, CCM is the set of concepts of minimum covering (CM). CJT contains all the possible conjunctions (CJ) that we have to check to generate the internal node. c is the number of concepts present in the node. CJ is a subset of features that form a conjunction and it can be described by only one feature or by the total set of features. Finally, VS contains the separability values for each CJ_t.

The value VS, for each conjunction, is calculated applying the next procedure:

$$ST(CJ_t) = \begin{cases} CL(CJ_t) & if\ \exists\ R_i^{CJ_t} \cap R_j^{CJ_t} = \emptyset,\ \ i \neq j \\ S(CJ_t) & another\ case \end{cases} \tag{1}$$

In (1) $R_i^{CJ_t} \subseteq M_{CJ_t}$ is the reference set of values of CJ_t that belong to the concept i.

CL(CJ_t) is defined as follow:

$$CL(CJ_t) = \frac{1}{c}\sum_{i=1}^{c} V_i \tag{2}$$

where:

$$V_i = \begin{cases} 1 & if\ R_i^{CJ_t} \cap R_q^{CJ_t} = \emptyset,\ \ q \neq i \\ 0 & another\ case \end{cases} \tag{3}$$

Equation (2) is applied to $CJ_t \in CJT$ if it found at least one class that is separated of the rest with CJ_t. Equation (3) added one, each time that the reference set for CJ_t in the concept i was different of the rest of the reference sets, $q = 1,\ldots,c$.

If CJ_t does not separate at least one class, then the second part of (1) is applied, where:

$$S(CJ_t) = \frac{1}{c}\sum_{i=1}^{c}\left(\frac{card(R_i^{CJ_t} \cap R_q^{CJ_t})}{card(R_i^{CJ_t} \cup R_q^{CJ_t})}\right),\ \ q \neq i \tag{4}$$

q go through all the concepts in CCM. The result will be a value between 0 and 1, where the maximum value is the most separability among the classes, so we must use that CJ for generate the node.

In (4), the expression

$$card(R_i^{CJ_t} \cap R_q^{CJ_t}) \tag{5}$$

represents the cardinality of the intersection between the reference sets of CJ_t in the concepts i and q, as well as

$$card(R_i^{CJ_t} \cup R_q^{CJ_t}) \tag{6}$$

represents the cardinality of the union between those sets.

When ADT works with qualitative features in CJ_t, it used (5) and (6) as in (4) and the proportion of data that not distinguish to one class with the others is obtained, it determines how many values are present in both classes.

When features are quantitative, the reference sets will be described by numbers and/or intervals, so the division in (4) is replace by

$$\sum_{v=1}^{ne} \left(\frac{InD_v(CJ_t)}{InT_v(CJ_t)} \right) \tag{7}$$

In (7) ne represents the number of elements presents in $R_i^{CJ_t}$, multiply with the number of elements in $R_q^{CJ_t}$. Besides, the equation

$$\begin{aligned} InD_v(CJ_t) = abs(min(E_{i_a}^{CJ_t}) - min(E_{q_b}^{CJ_t})) + \\ abs(max(E_{i_a}^{CJ_t}) - max(E_{q_b}^{CJ_t})) \end{aligned} \tag{8}$$

represents the difference between the intervals $E_{i_a}^{CJ_t}$ and $E_{q_b}^{CJ_t}$. Then, $E_{i_a}^{CJ_t}$ is the element a of $R_i^{CJ_t}$, where $a = 1, \ldots, ti$ and ti is the total number of elements in that reference set. $E_{q_b}^{CJ_t}$ is the element b of $R_q^{CJ_t}$, $b = 1, \ldots, tq$ and tq is the total number of elements in that set, and with

$$InT_v(CJ_t) = \max(E_{i_a}^{CJ_t}, E_{q_b}^{CT_t}) - \min(E_{i_a}^{CJ_t}, E_{q_b}^{CT_t}) \tag{9}$$

the total interval that exist between $E_{i_a}^{CJ_t}$ and $E_{q_b}^{CT_t}$ is obtained.

So, this split rule, for each feature subset (CJ), measure the separability among the classes in LM.

D. The ADT Algorithm

The methodology that ADT follows for building the decision tree is the top-down induction. This method selects the best conjunction that separate the classes of the data set applying the proposed split rule. The number of branches will depend in how many features has the conjunction. If only has one feature, the number of branches will be according to the number of its admissibles values. If has two or more features, it will depend in how many descriptions of that CJ we can form in each concept.

The decision tree is built by a recursive function that creates all the nodes in the tree (root, internal and terminal nodes). It starts to build the root and for each branch it generates a new node, when it has only one concept to process it builds a leaf. The subset of concepts that each branch include will be formed by the concepts that fulfill the condition in that path. The algorithm ADT is described in the following way:

```
Input: A concept set ${Z_1, \ldots, Z_c}$.
Output: Decision tree (DT) based in concepts.
1. Obtain the set of concepts of minimum covering
    $CCM = \{CM_1, \ldots, CM_c\}$.
2. Generate DT
    Applying GenerateNode(CCM)
3. Classify the object of a Control Matrix (CoM) with DT.
```

The recursive function GenerateNode take as parameter the set of concepts of minimum covering, the procedure of this function is presented below:

```
GenerateNode(CCM)
    If $card(CCM) = 1$ then
        Create leaf with that value.
    else
        VS = SplitRule(CCM).
        $CJ_E = \max(VS)$.
        Create internal node, with value $CJ_E$.
        Create branches for $CJ_E$.
        For each branch $r$ do
            $CCM'$ = CreateSubsetCM(CCM,r)
            GenerateNode($CCM'$)
```

CJ_E is the elected conjunction since it has the highest value of separability found by the split rule.

V. EXPERIMENTAL RESULTS

This section shows the robustness behavior of ADT. The comparison with the algorithm ID3 when a change occurs in the original data set (Zoo data set) is presented too.

An available data set Zoo [17] is processed. This data set contains descriptive information about 101 animals described in terms of 15 boolean features and 1 numerical feature. 91 objects composed the LM and remaining the CoM, we choose this objects in aleatory way. The concepts of minimum covering to can generate the final decision tree are obtained. One example of these concepts are shown below:

For class 7:

$$\begin{aligned} CM_7 = {} & [X1 = \{0\}] \wedge [X2 = \{0\}] \wedge [X3 = \{1, 0\}] \wedge \\ & [X4 = \{0\}] \wedge [X5 = \{0\}] \wedge [X6 = \{0, 1\}] \wedge \\ & [X8 = \{0\}] \wedge [X9 = \{0\}] \wedge [X10 = \{1, 0\}] \wedge \\ & [X11 = \{0, 1\}] \wedge [X12 = \{0\}] \wedge [X14 = \{0, 1\}] \wedge \\ & [X15 = \{0\}] \wedge [X16 = \{0, 1\}] \wedge [\{([X7 = \{1\}] \wedge \\ & [X13 = \{[0, 8]\}]) \vee ([X7 = \{0, 1\}] \wedge \\ & [X13 = \{[0, 5], 8\}])\}] \end{aligned}$$

The final decision tree generated by ADT is shown in Fig. 2.

ADT generated a tree with depth of 5, and only contains 6 nodes, so we can considered that with only a few characteristics of the object we can have a good classification. CCM in each branch represents the subset of concepts that we will use in the next node. According to that subset we can verify the

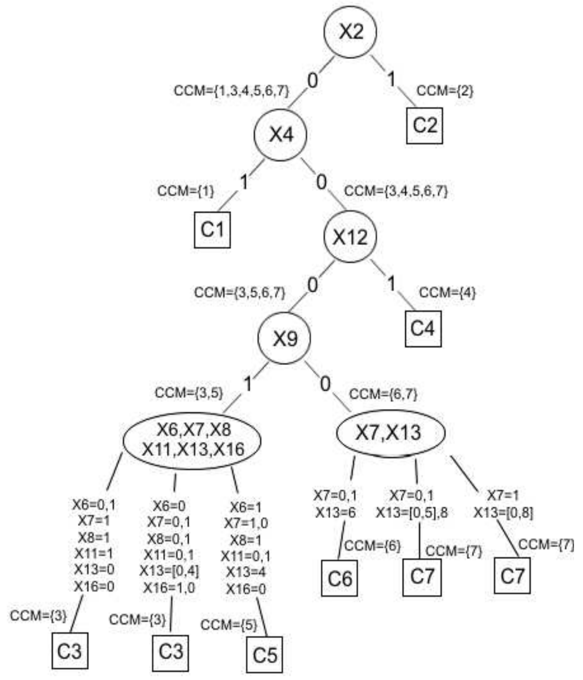

Fig. 2. Decision tree for Zoo generated by ADT.

property of separability that the split criterion proposed give us.

The objects that form the CoM, were classified by the tree obtained for ADT. Table II shows the results of this process.

TABLE II
RESULTS OF CM - ZOO

Object in CM	Class ADT	Real Class
1	1	1
2	1	1
3	1	1
4	1	1
5	2	2
6	2	2
7	3	3
8	4	4
9	6	6
10	7	7
Good Classification:	100%	

ID3 was applied to same data set and its decision tree is shown in Fig. 3. We can observe that the depth of the tree is 4, but the total number of nodes is 9. Besides, the split criterion that ID3 used do not generate a separability among classes, for example, class 7 is present in 5 paths, while in the tree built by ADT is present in only one path. ID3 shows the same percent of classification that ADT classifying the CoM.

In the table III, a new description of a object of LM is shown, this have not taken in count to generate the tree. The object belongs to class 3 in Zoo. When this object arrive to LM, we have to verify if the constructed tree cover the object, if not, the decision tree has to be re-build.

With the tree built by ID3, the new object is not cover. For solving this problem ID3 has to re-build the tree with the new

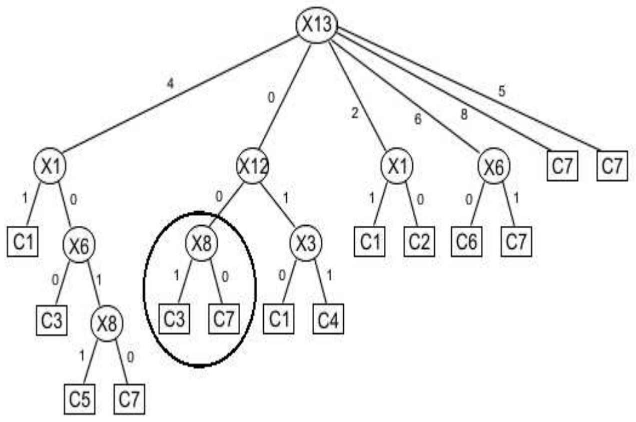

Fig. 3. Decision tree for Zoo obtained by ID3.

TABLE III
NEW OBJECT OF ZOO

	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$	$X8$
O_{m+1}	0	0	1	0	0	0	1	0
	$X9$	$X10$	$X11$	$X12$	$X13$	$X14$	$X15$	$X16$
	1	1	0	0	0	1	0	0

test of object. Fig. 4 shows the new tree built by ID3. The node in the circle shown that it has been changed in order to cover all the objects in LM. In the case of ADT, the tree cover the new object and the tree has not change.

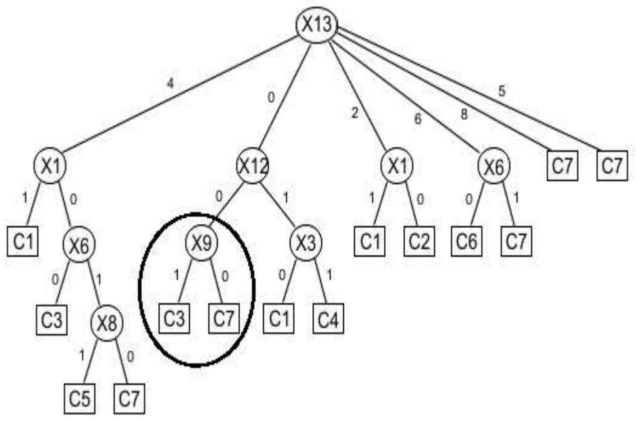

Fig. 4. Decision tree for Zoo re-building by ID3.

VI. CONCLUSIONS

We proposed a new method to generate decision trees which is based in concepts or properties that characterized the classes in a data set. This concepts give us a robust tree that allows changes in the learning matrix, so not always will be necessary to re-build the tree when this occur.

Due to the characteristic of using properties instead of the LM, the proposed method is feasible to apply in large data sets.

Another characteristic of the proposed method is a new split criterion, which allows to choose, not only a feature to form the node, but a subset of features that describe some similar characteristics of the objects presents in the LM. Besides, this criterion look for the separability among classes, so in this sense is different to others split rules used in classical algorithms to generate decision trees.

REFERENCES

[1] J. R. Quinlan, *Induction of decision trees*, Machine Learning, Vol. 1, 1986.
[2] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.
[3] R. Safavian and D. Landgrebe, *A survey of decision tree classifier methidology*, IEEE Transactions on Systems, Man and Cybernetics, 21, 1991.
[4] T. Michell, *Machine Learning*, McGraw Hill, Inc. U.S.A., 1997.
[5] Cesary Z. Janikow, *Fuzzy decision trees: Issues and methods*, IEEE Transactions on systems, man and cybernetics, Vol. 28, No. 1, 1998.
[6] R. Konavi, J. R. Quinlan, *Decision tree discovery*, 1999.
[7] A. R. Webb *Statistical pattern recognition*, John Wiley & Sons, 2002
[8] G. Martínez, G. Núñez, A. Guzman, A. García. *Inducción de árboles de decisión, algoritmo k-d*, Simposium Internacional de Computacin, IPN, México, 1998.
[9] R. Olshen, L. Breiman, J. Friedman, *Classification and Regression Trees*, Wadsworth International Group, 1984.
[10] W.-Y. Loh and N. Vanichsetakul, *Tree-structured classification via generalized discriminant analysis*, Journal of the American Statistical Association 83, 1988.
[11] W.-Y. Loh and Y.-S. Shih, *Split selection methods for classification trees*, Statistica Sinica, Vol. 7, 1997.
[12] C. Shou-Chih, P. Hsing-Kuo and L. Yuh-Jye, *Model trees for classification of hybrid data types*, Intelligent Data Engineering and Automated Learning - IDEAL: 6th International Conference, Brisbane, Australia, 2005.
[13] P. E. Utgoff and C. E. Brodley, *Linear Machine Decision Trees*, COINS Technical Report 91-10, 1991.
[14] A. P. Porrata, J. R. Shulcloper and J. F. Martínez-Trinidad, *RGC: a new conceptual clustering algorithm for mixed incomplete data sets*, Journal of Mathematical and Computer Modeling, Volume 36, Issue 11-13, 1375-1385, 2002.
[15] J. Ruiz-Shulcloper and M. A. Abidi, *Logical Combinatorial Pattern Recognition: A Review*, Recent Research Developments in Pattern Recognition, S. G. Pandalai (Ed), Recent Research Developments in Pattern Recognition, Transworld Research Networks, Kerala, India, 2002.
[16] F. Berzal Galiano, *ART. Un método alternativo para la construcción de árboles de decisión*, Tesis, Granada, 2002.
[17] UCI Repository, ftp://ftp.ics.uci.edu/pub/machine-learning-databases.