International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:3, No:8, 2009

An Attribute-Centre Based Decision Tree Classification Algorithm

Gökhan Silahtaroğlu

***Abstract*—**Decision tree algorithms have very important place at classification model of data mining. In literature, algorithms use entropy concept or gini index to form the tree. The shape of the classes and their closeness to each other some of the factors that affect the performance of the algorithm. In this paper we introduce a new decision tree algorithm which employs data (attribute) folding method and variation of the class variables over the branches to be created. A comparative performance analysis has been held between the proposed algorithm and C4.5.

***Keywords*—**Classification, decision tree, split, pruning, entropy, gini.

I. Introduction

CLASSIFICATION has gained a lot of importance in literature and it has a great deal of application areas from medicine to astronomy, from banking to text classification. The aim of the classification is to find the similar data items which belong to the same class. The term class may be considered as the dependent variable in statistics. For example, as the tail length, ear size, number of teeth etc are the variables which may vary from one specie to another, the variables 'cat' and 'dog' will be determined according to the values of the other variables. Classification is a predictive model of data mining that predicts the class of a dataset item using the values of some other variables. [2],[3]

If

$D = \{t_1, t_2, ...,t_n\}$.

is a dataset and each t_i is a record in the dataset with more than one attributes (data fields) and as

$C = \{C_1, C_2, ...,C_m\}$

C Represents m number of the classes. Then,

f: $D \rightarrow C$ and each $t_i \in$ any C_j

Each C_j is a different class and has its own records, i.e.

$C_j = \{ t_i \mid f(t_i) = C_j , 1 \leq i \leq n, \text{ and } t_i \in D\}$.

So far, many algorithms have been introduced with different models. Classification algorithms may be categorized as follows.

G. Silahtaroğlu is with Beykent University, Department of Mathematics and Computing, Istanbul 34900, Turkey (e-mail: gsilah@beykent.edu.tr).

- Distance based algorithms
- Statistical algorithms
- Neural networks
- Genetic algorithms.
- Decision tree algorithms

K-Nearest Neighbor is one of the best known distance based algorithms, in the literature it has different version such as single link, complete link, K-Most Similar Neighbor etc. Distance based algorithms use Euclidian and Hamming distance measures or any similarity measure such as Jaccard, Cosine, Dice etc. to determine the class of each item [6],[17].

Regression and Bayesian algorithms are some of the most frequently used statistical classification algorithms in the literature. Bayesian algorithms predict the class depending on the probability of belonging to that class. On the other hand, regression, after determining the model i.e linear or non-linear, predicts the class with the coefficients produced from the data set [7].

Neural Networks and Genetic algorithms are other methods of clustering [4],[5]. Neural Network algorithms, predict the class of a data item with the help of the *w* weights calculated with lots of scans over the dataset[15].

Genetic algorithms, introduced by John Holland in 1975 [1],[2] are numerical optimization algorithms inspired by the nature evolution process and directed random search techniques; using, cross over, mutation techniques genetic algorithms create the best population to classify the data items [3],[4],[6].

Decision tree algorithms build a tree which also yields a series of *if- else-then* rules to classify the data items. ID3, C4.5, CART and sprint are among the best known decision tree algorithms. Since we propose a decision tree algorithm in this paper, we will discuss the decision tree model and some of the algorithms in the next section [3].

II. Background and Related Work

A. Tree Based Classification

A tree approach is used for classification which is a model of data mining [8],[9]. To reach the classes through the shortest path the most suitable attribute is chosen to be the root and recursively the algorithm make calculations to determine the most suitable attribute in dataset to be the next node. Here 'the most suitable' adjective is to divide the rest of

the database roughly into two or more equal parts. The attribute which is the closest to this is chosen as the most suitable attribute.

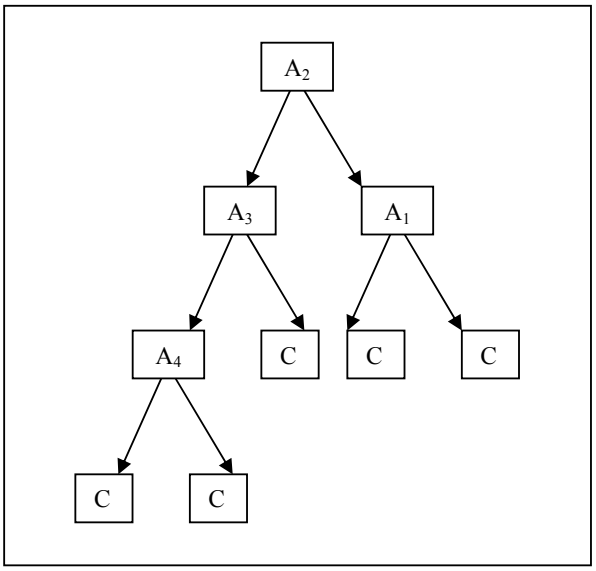

Fig. 1 A sample decision tree

After choosing the root node, the algorithm adds arcs to the root for each predicate. Adding node to new branches is carried on as it is done for the first (root) node. During the processes if the pre-assigned criterion is reached, the arcing stops and the end of the final branch is labeled as one of the classes. Here, an algorithm differs from one another with the splitting criteria it exercises. In the literature, *entropy* and *gini* index are mostly employed by different algorithms as spitting criteria.

B. Tree Based Classification Algorithms

In literature, there are different approaches to build a tree. One is to use entropy concept. ID3 and C4.5 algorithms use *entropy* to find the node representatives [10]. If we represent the probabilities as $<p_1, p_2,\ldots,p_n>$ then the sum equals to 1 [10],[11].

$$\sum_{i=1}^{n} p_i = 1 \tag{1}$$

In this case, entropy is

$$H(p_1, p_2, ..., p_n) = \sum (p_i \log(1/p_i)) \tag{2}$$

ID3 algorithm determines the nodes with a gain value

$$Gain(D,S) = H(D) - \sum_{i=1}^{n} P(D_i)H(D_i) \tag{3}$$

The attribute which yields the highest gain is chosen as the root or the next node.

C4.5 divides this gain value by splitting information which is calculated as:

$$Split(D,S) = H\left(\frac{|D_1|}{|D|},...,\frac{|D_s|}{|D|}\right) \tag{4}$$

Another approach is to use *gini* index as it is introduced in SLIQ algorithm [11],[13],[14].

$$gini(K) = 1 - \sum p_j^2 \tag{5}$$

Applying the equation below, the most suitable attribute to split the database into two equal (or more) parts is roughly uncovered [8].

$$gini_{split}(K) = \frac{n_1}{n_2} gini(K_1) + \frac{n_2}{n_2} gini(K_2) \tag{6}$$

CART algorithm use another entropy based algorithm to create binary trees. This algorithm is different from the others as it produces *if-then rules* with a Boolean answer only such as true/false, yes/no. It uses the following formula to determine the class of each data set [3].

$$\Psi(s/t) = 2P_L P_R \sum_{J=1}^{M} |P(C_J|t_L) - P(C_J|t_R)| \tag{7}$$

III. Our Contribution

We have adapted an algorithm to choose the right attribute to split the dataset into equal parts. The algorithm has been introduced for the databases which has categorical values. Obviously, categorical values need special treatment since it is not possible to calculate the average, standard deviation, sum of the values etc. for the entries.

We firstly determine the center of each attribute (CA). CA is calculated as in Eq. 9.

Here N represents the total number of cases or the number of rows in the dataset. This value is divided by NV, that is the number of variables which reside in the current attribute. For example if the attribute is considered as a vector

Vector = {A,B,A,B,A,A,A,A,A,B,A,C,C,C}

Here N= 14 because there are 14 different values in the vector. NV equals 3 because there are 3 different variables in the vector , they are A, B and C.

Equal-Split Parameter is the sum of the absolute difference between CA and NV_i.

NV_i represents the number of each variable of the attribute. For the vector above we have NV_i values for i = 3. NV_A, NV_B, NV_C . Thus,

$NV_A = 8$; $NV_B = 3$ and $NV_C = 3$.

Without regarding the class attribute EP values give the rate to fold the attribute into the number of variable types. If the attribute has two different variables, it will be folded in two, if it has three, four or five different variables, the attribute will be folded in three, four or five accordingly. When the attribute is folded with number of the variables, the length of the folded attribute is supposed to be minimum in size in order to generate an efficient decision tree which reaches the leaves through the shortest path available. So, minimum EP value

will give us the right attribute which reduces the attribute to the minimum length.

$$EP = \sum_{i=1}^{n} \left| CA - NV_i \right| \tag{8}$$

$$CA = \frac{N}{NV} \tag{9}$$

EP value may give us an idea to choose the right attribute for splitting the dataset, however EP value is something to fold the attribute only not the dataset itself. When the class attributed is taken into account, EP value would not be enough whereas it is for clustering. Thus, the distribution of the class attribute over the branches is an important parameter to determine, if the attribute really folds the whole dataset into a minimum length. So, over the branches we calculate the variation of the class attribute.

Firstly, we generate a class distribution matrix as in Eq. 10.

$$A_{nxm} = \begin{bmatrix} a_{ij} & a_{ij+1} & a_{ij+2} & \dots & a_{ij+m} \\ a_{i+1j} & a_{i+1j+1} & a_{i+1j+2} & \dots & a_{i+1j+m} \\ a_{i+2j} & a_{i+2j+1} & a_{i+2j+2} & \dots & a_{i+2j+m} \\ \dots & \dots & \dots & \dots & \dots \\ a_{i+nj} & a_{i+nj+1} & a_{i+nj+2} & \dots & a_{i+nj+m} \end{bmatrix} \tag{10}$$

Here i = 1,2,…,n represents the number of variable type, and j = 1,2,…,m is the number of the classes in the dataset.

Variation of this matrix will certainly give the distribution of the classes regarding the variable of each attribute.

$$V_n = \mathrm{var}(A_{nxm}) \tag{11}$$

So, we use average of $var(V_m)$ as the single value to show how well balanced the classes are distributed throughout the attribute.

Finally,

$$BC = \frac{1 + EP}{\dfrac{\sum_{i=1}^{m} \mathrm{var}(V_m)}{m}} \tag{12}$$

the equation will give an idea about the balance of the classes (BC). Adding 1 to EP value is necessary in case of EP = 0.

Considering the minimum length, the attribute which has the minimum BC value will relatively fold the dataset better in comparison with the other attributes.

Here, we give the pseudo code of the proposed algorithm.

Input:
D: Data // Get Data
P: Purity of each class // a value betweeb 0 -1. ie . 0.80, 0.95 etc
Output
Decision Tree

For each data atribute **do**
  NV = number of variables
  CA = N / NV // *Center is calculated*
  For i = 1 to NV
    NV_i = number of each variable
    EP = CA - NV_i
      End
  For i = 1 to NV
    EP = abs(CA - NV_i) + EP // *EP value is calculated for each attribute*

  // *distortion of class distribution is calculated*.

$$BC = \frac{1 + EP}{\dfrac{\sum_{i=1}^{m} \mathrm{var}(V_m)}{m}}$$

  End

Loop
Choose the lowest BC value as the node and arch it for NV
If the preasinged purity has not been reached
    go to step 1 for each branch
Else
    Prune the tree according to the given initial criteria.
STOP

Fig. 2 General structure of the algorithm

Example:

Let Table I be the dataset of which decision tree will be generated for.

For the dataset
N = 15.
For the variable A1; NV = 2, NV_{TRUE}= 7, NV_{FALSE} = 8 thus,
so CA = 7.5 and EP = 1.
For A2; NV = 3, NV_{BIG}= 10, NV_{Medium} = 3 thus, NV_{SMALL}=2 ,
thus, CA = 5 and EP = 10.
For A3; NV = 3, NV_{RED}= 6, NV_{GREEN} = 5,
then, NV_{BLUE}=4 , thus, CA = 5 and EP = 2.

Therefore, BC values for each attribute is

BC_{A1} =1.7071, BC_{A2} =8.9241, BC_{A3} = 1.6457.

In this case A3 which has the minimum BC value will be the node to split.

TABLE I
SAMPLE DATA SET (1)

CLASS	A1	A2	A3
Type2	TRUE	BIG	RED
Type1	TRUE	BIG	RED
Type3	TRUE	Medium	GREEN
Type2	TRUE	BIG	BLUE
Type1	TRUE	BIG	RED
Type3	TRUE	SMALL	GREEN
Type2	TRUE	SMALL	BLUE
Type2	FALSE	BIG	RED
Type3	FALSE	BIG	GREEN
Type1	FALSE	Medium	BLUE
Type2	FALSE	Medium	RED
Type3	FALSE	BIG	GREEN
Type2	FALSE	BIG	BLUE
Type2	FALSE	BIG	RED
Type3	FALSE	BIG	GREEN

C4.5 algorithm chooses the same attribute (A3) because it has the highest GAIN value (0.5890).

For C.4.5 gain values are as follows.

A1 = 0.0298, A2 = 0.0525, A3 = 0.5890.

IV. A COMPARATIVE ANALYSIS

C4.5 algorithm is one of the most frequently used algorithms in literature, in this part of the paper we will discuss classification performance of C4.5 and the proposed algorithm. C4.5 algorithm classifies the data more accurately then ID3 does , because it employs a gain ratio by:

$$Split(D,S) = H\left(\frac{|D_1|}{|D|},...,\frac{|D_s|}{|D|}\right) \quad (13)$$

This put an advantage on C4.5 if the class borders are very close to each other. In this case C4.5 gives a better performance. Likewise our algorithm is superior to C4.5 when the distance is not much between the classes.

Consider the Table II for a decision tree. If we apply C4.5 algorithm the root node will be A1 attribute with a gain ratio of 0.1245 whereas A2 attribute will have a lower gain ratio which is 0.0608.

When the proposed algorithm is applied, since A1 attribute has a BC value of 1, A2 attribute will have BC value, 2.5396 the same result as the C4.5 algorithm has produced will be held.

If Fig. 3 and Fig. 4 are examined it is clear that both algorithm have chosen the right attribute as the root node, because A1 attribute produces a two-branch and better balanced tree in comparison with attribute A2. Attribute A2 not only arcs three branches but also the branches it gives are not as balanced as A2 attribute's branches are.

TABLE II
SAMPLE DATA SET (2)

ID	Class	A1	A2
1	Type1	Yes	Hot
2	Type2	Yes	Hot
3	Type2	Yes	Hot
4	Type2	No	Warm
5	Type1	No	Warm
6	Type1	No	Warm
7	Type1	No	Cold
8	Type2	Yes	Cold
9	Type2	Yes	Cold
10	Type2	No	Cold

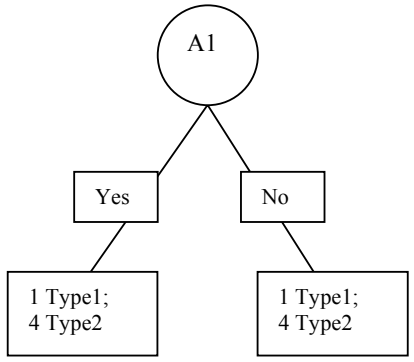

Fig. 3 When A1 is chosen as root node

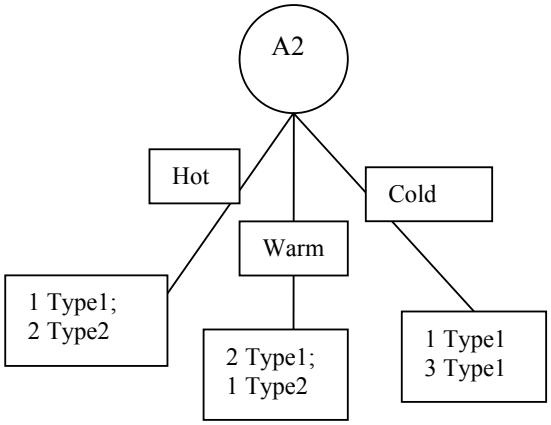

Fig. 4 When A2 is chosen as root node

Nonetheless, when a new record is injected to the dataset as it is depicted in Table III, the tree will be like in fig. 6 when A1 attribute is chosen and in Fig. 5 depicts the tree when A2 is chosen as the root node.

Now, let us check the new results given by both algorithms. C4.5 produces gain values as follows:

Gain(A1) = 0.1665 and Gain(A2= 0.0540).
Then according to C4.5 A1 is still the root node.
For the proposed algorithm, BC(A1) = 3.3094 and BC(A2) = 2.2810.

Thus, A2 is the root node. In the figures it is clear that with the new record, A1 attribute creates a less balanced tree than A2 does.

With the injection of the new record, our algorithm changes the root node however; C4.5 cannot adopt itself to the new situation of the dataset.

TABLE III
SAMPLE DATA SET (3)
AFTER THE 11TH RECORD ADDED

ID	Class	A1	A2
1	Type1	Yes	Hot
2	Type2	Yes	Hot
3	Type2	Yes	Hot
4	Type2	No	Warm
5	Type1	No	Warm
6	Type1	No	Warm
7	Type1	No	Cold
8	Type2	Yes	Cold
9	Type2	Yes	Cold
10	Type2	No	Cold
11	*Type2*	*Maybe*	*Hot*

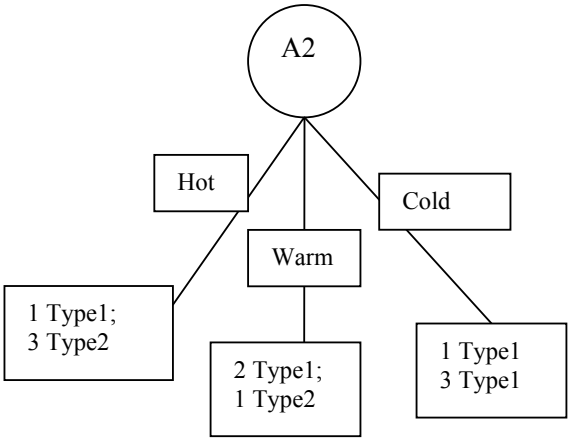

Fig. 5 When A2 is chosen as root node (after the record is injected)

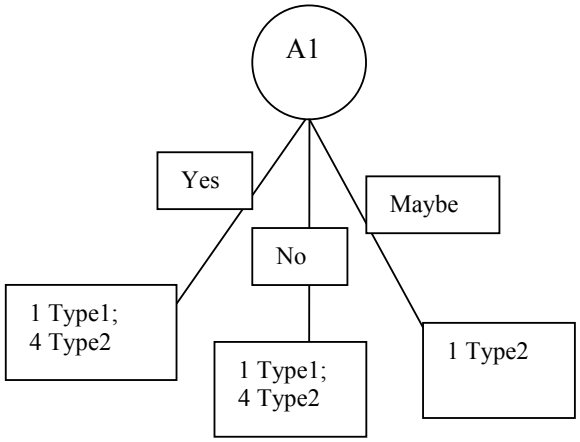

Fig. 6 When A1 is chosen as root node (after the record is injected)

REFERENCES

[1] Coley, A D., (1999). An Introduction to Genetic Algorithms for Scientists and Engineers. World Scientific, Singapore, 188p.
[2] Pham, D.T., and Karaboga, D., 2000. Intelligent Optimization Techniques. Springer, London, Great Britain, 261p.
[3] Han J., & Kamber, Micheline. (2001). Data Mining Concepts and Techniques, Morgan Kaufman Publishers Academic Press.
[4] Authori Fausett L.(1994). Fundamentals of Neural Networks, Prentice-Hall, New Jersey.
[5] Maulik U. & Sanghamitra B.(2000). Genetic Algorithm-based clustering technique, Journal of the Pattern Recognition, Pergamon, issue: 33.
[6] Bill F. (Ed.) (1992). Information retrieval : data structures & algorithms. Prentice Hall.
[7] Breiman, L., J. H. Fried man, R. A. Ol shen, and C. J. Stone. (1984). Classification and regression trees. Mon terey, Calif., U.S.A.Wadsworth, Inc.
[8] Shafer J.C., Agrawal R., Mehta M.: "SPRINT: A Scalable Parallel Classifier for Data Mining", Proc. of the 22th International Conference on Very Large Databases, Mumbai (Bombay), India, Sept. 1996.
[9] Mitchell T.(1997). Machine Learning, McGraw-Hill International.
[10] Quinlan,J.Ross. (1987). Simplifying decision trees, International Journal of Man-Machine Studies,issue: 27(3), (pp. 221 – 234).
[11] Breiman L., & Friedman J. H., & Olshen R. A., & Stone C. J. (1984). Classification and Regression Trees, Wadsworth, Belmont.
[12] Mehta M., & Agrawal R., & Rissanen J. (1996). SLIQ: A Fast Scalable Classifier for Data Mining, Proceedings of 5th International Extending Database Technology Conference.France. (pp. 18-32). Springer-Verlag, London.
[13] Agrawal R. & Shafer J.C. (1996). Parallel Mining of Association Rules, Proceedings. of IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6. (962- 969). IEEE Educational Activities Department. USA.
[14] Hettich, S. , & Bay, S. D. (1999). The UCI KDD Archive, Department of Information and Computer Science, University of California, Irvine, CA. Retrieved September 1, 2008, from http://kdd.ics.uci.edu.
[15] Pham D.T., & Chan A.B.(1998). Control Chart Pattern Recognition using a New Type of Self Organizing Neural Network. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering. Vol 212, No 1, (pp. 115-127). Professional Engineering Publishing.
[16] Keogh, E. & Pazzani, M. (2001). Derivative Dynamic Time Warping. In First SIAM International Conference on Data Mining (SDM'2001), Chicago, USA.
[17] Alcock R.J., & Manolopoulos Y.(1999). Time-Series Similarity Queries Employing a Feature-Based Approach. 7th Hellenic Conference on Informatics. Ioannina,Greece.