ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 3, September 2013

A New Approache for Design and Implementation of Inducion Algorithm for Training Oblique Decision Tree

Sonal P Patil ,Sonal V Badhe

Abstract-In this paper we present new algorithm for oblique decision tree induction. We propose new classifier that performs better than the other decision tree approaches in terms of accuracy, size, and time. Proposed algorithm uses geometric structure in the data for assessing the hyper planes. At each node of the decision tree, we suggest the clustering hyper planes for both the classes and using this representation their angle bisectors is selected as split rule at that node. The algorithm we present here is applicable for2-class and multiclass problems. Through empirical investigation we demonstrate that this idea leads to small decision trees and better performance. We also present some analysis to show that the angle bisectors of clustering hyper planes that we use as the split rules at each node are solutions of an interesting optimization problem and hence argue that classifier obtained with new approach is as good and novel classification method.

***Keywords*- oblique decision tree, CART, OC1, SVM, GDT.**

I. INTRODUCTION

A classification model is built form available data using the known values of the variables and the known class. The available data is called training set and class value is called class label. Constructing the classification model is called supervised learning. Then, given some previously unseen data about an object or phenomenon whose class label is not known, we use the classification model to determine its class. There are many reasons why we may wish to set up a classification procedure or develop a classification model.

- Such a procedure may be much faster than humans (postal code reading).
- Procedure needs to be unbiased (credit applications where humans may have bias)
- Accuracy (medical diagnosis).
- Supervisor may be a verdict of experts used for developing the classification model.

Some issues that must be considering in developing a classifier are listed below:

Accuracy: represented as proportion of correct classifications. However, some errors may be more serious than others and it may be necessary to control different error rates.

Speed: the speed of classification is important in some applications (real time control systems). Sometimes there is tradeoffs between accuracy and speed.

Comprehensibility of classification model especially when humans are involved in decision making (medical diagnosis).

Time to learn: the classification model from the training data, e.g. in a battlefield where correct and fast classification is necessary based on available data. Decision tree can be explained a series of nested if-then-else statements. Each non-leaf node has a predicate associated, testing an attribute of data. Terminal node denotes class, or category. To classify a data ,we have to traverse down the tree by starting from root node ,testing predicates(test attribute) and taking branches labelled with corresponding value[2]. Decision tree classifiers have been popular in pattern recognition, concept learning, and other AI branches. They enable a divide-and-conquer strategy to be applied to classification problems, and they enable context sensitive feature-subset selection to tackle high-dimensionality problems. Decision tree cab be broadly classified into two types i.e. axis parallel and oblique decision tree. Axis parallel decision trees that take into account only a single attribute at a time make splits parallel to the axis in the feature space of the dataset. On the other hand, oblique decision trees split the feature space by considering combinations of the attribute values, be them linear or otherwise[5] .Oblique decision trees have the potential to outperform regular decision trees because with a smaller Number of splits an oblique hyper plane can achieve better separation of the instances of data that belong to different classes. in a decision tree, each hyper plane at a nonleaf node should split the data in such a way that it aids further classification; the hyper plane itself need not be a good classifier at that stage. In view of this, many classical top-down decision tree learning algorithms are based on rating hyper planes using the so-called impurity measures. The main idea is given as follows: Given the set of training patterns at a node and a hyper plane, we know the set of patterns that go into the left and right children of this node. If each of these two sets of patterns have predominance of one class over others, then, presumably, the hyper plane can be considered to have contributed positively to further classification. At any stage in the learning process, the level of purity of a node is some measure of how skewed is the distribution of different classes in the set of patterns landing at that node. If the class distribution is nearly uniform, then the node is highly impure; if the number of patterns of one class is much larger than that of all others, then the purity of the node is high. The impurity measures used in the algorithms give higher rating to a hyper plane, which results in higher purity of child nodes. The Gini index, entropy, and towing rule are some of the frequently used impurity measures. Information gain is based on Claude Shannon's work on information

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 3, September 2013

theory, which calculates the value of messages. Gain ratio biases the decision tree against considering attributes with a large number of distinct values. So it solves the drawback of information gain [7]. Distance measure, like Gain ratio, normalizes the gini index [3]. Here, in our tree-induction algorithm, we employ a hypreplane and angel biseor as split rule technique .Empirically, this oblique decision tree induction algorithm is found to be effective optimization problems .We shows that using our algorithm, we learn smaller decision trees with better classification accuracy than is possible with other standard methods. The rest of the paper is organized as follows. In Section II, we present the complete details of the proposed algorithm. Through empirical studies presented in Section III, we show that the proposed algorithm learns compact and accurate decision trees. In Section IV, we present a new pruning technique, along with some simulation results to show its effectiveness. Section V presents discussion and conclusions.

II. INDUCTION ALGORITHM FOR OBLIQUE DECISION TREE

The performance of any top-down decision tree algorithm depends on the measure used to rate different hyperplanes at each node and the split criteria. The problem of having a suitable algorithm to find the hyperplane that optimizes the chosen rating function is important. For all impurity measures, the optimization is difficult because finding the gradient of the impurity function with respect to the parameters of the hyperplane is not possible, by these considerations; we propose a new criterion function to assess the suitability of a hyperplane at a node that can capture the geometric structure of the class regions. For our criterion function, the optimization problem can also be solved more easily. Proposed method can be applied for two class and multiclass problem. We first explain our method by a two-class problem. For the Given the set of training patterns at a node, we first find two hyperplanes(one for each class). Each hyperplane is such that it is closest to all patterns of one class and is farthest from all patterns of the other class. We call These as clustering hyperplanes (for the two classes). These clustering hyperplanes capture the dominant linear tendencies in the examples of each class that are useful for discriminating between the classes. Hence, a hyperplane that passes in between both of them could be good for splitting the feature data space. Thus, we take the hyperplane that bisects the angle between the clustering hyperplanes as the split rule at this node. Since, in general, there would be two angle bisectors, we choose the bisector that is better, based on an impurity measure, i.e., the information gain. If the two clustering hyperplanes happen to be parallel to each other, then we take a hyperplane midway between the two as the split rule.

A. Proposed Method for Two class Classification

Let $S = \{(\mathbf{x}i, yi) : \mathbf{x}i \in R^d; yi \in \{-1,1\}, i = 1. . .n\}$, be the training data set. Let $C+$ be the set of points for which $yi = 1$. In addition, let $C-$ be the set of points for which $yi = -1$. For an oblique decision tree learning algorithm, the main computational task is given as follows: Given a set of data points at a node, find the best hyperplane to split the data. Let St be the set of points at node t. Let $nt+$ and $nt-$ denote the number of patterns of the two classes at that node. Let $A \in \Re^{n_+^t * d}$ be the matrix containing$t$ points of class $C+$ at Node t as rows. Similarly, let $B \in \Re^{n_-^t * d}$ be the matrix whose rows contain points of class $C-$ at node t. Let $h_1(w_1, b_1): w_1^T x + b_1 = 0$ and $h_2(w_2, b_2): w_2^T x + b_2 = 0$ be the two be the two clustering hyperplanes. Hyperplane $h1$ is to be closest to all points of class $C+$ and farthest from points of class $C-$. Similarly, hyperplane $h2$ is to be closest to all points of class $C-$ and farthest from points of class $C+$. To find the clustering hyperplanes, we use the idea as in GEPSVM [3]. The nearness of a set of points to a hyperplane is represented by the average of squared distances.

The average of squared distances of points of class $C+$ from a hyperplane $w^Tx+b=0$ is

$D_+(w,b)=(1/n_+^t \ \|w\|^2)\sum_{x_i \in C_+} |w^T x_i + b|^2$, where $\|.\|$ denotes the standard Euclidean norm.

Let $\tilde{w} = [w^T \ b]^T \in \Re^{d+1}$ and $\tilde{x} = [x^T \ 1]^T \in \Re^{d+1}$. Then $w^Tx_i+b=\tilde{w}^T\tilde{x}$. The average of the squared distances of points of class $C-$ from h will be $D-(\mathbf{w}, b)$ and the average of the squared distances of points of class $C+$ from h will be $D+(\mathbf{w}, b)$. To find each clustering hyperplane, we need to find h such that one of $D+$ or $D-$ is maximized while minimizing the other. The two clustering hyperplanes, which are specified by $\tilde{\mathbf{w}}_1 = [\mathbf{w}^T_1 \ b_1]^T$ and $\tilde{\mathbf{w}}_2 = [\mathbf{w}^T_2 \ b2]^T$, can be formalized as the solution of optimization problems. Once we find clustering hyperplanes, the hyperplane we associate with the current node will be one of the angle bisectors of these two hyperplanes. Let $\mathbf{w}^T_3 \mathbf{x} + b_3 = 0$ and $\mathbf{w}^T_4 \mathbf{x} + b_4 = 0$ be the angle bisectors of $\mathbf{w}^T_1 \mathbf{x} + b_1 = 0$ and $\mathbf{w}^T_2 \mathbf{x} + b_2 = 0$. Choose the angle bisector that has a lower value of the information gain. Let $\tilde{\mathbf{w}}_t$ be a hyperplane that is used for dividing the set of patterns S_t in two parts S^{tl} and S^{tr} .Let $n^{tl}_+$and n^{tl}_- denote the number of patterns of the two classes in set S^{tl} , and let n^{tr}_+ and n^{tr}_- denote the number of patterns of the two classes in set S^{tr} .We choose $\mathbf{w}3$ or $\mathbf{w}4$ to be the split rule for St based on which of the two gives lesser value of the Information gain the complete algorithm can be described as : At any given node, give the set of patterns St, we find the two clustering hyperplanes (by solving the generalized eigenvalue value problem) and choose one of the two angle bisectors, based on the information gain, as the hyperplane to be associated with this node. We then use this hyperplane to split St into two sets, i.e., those that go into the left and right child nodes of this node. We recursively do the same at the two child nodes. The recursion stops when the set of patterns at a node are such that the fraction of patterns belonging to the minority class of this set are below a user-specified threshold or the depth of the tree reaches a prespecified maximum limit.

B. Proposed Method For Multiclass Classification

The algorithm presented in the previous section can be easily generalized to handle the case when we have more than two classes. Let $S = \{(\mathbf{x}i, yi) : \mathbf{x}i \in \_d; yi \in \{1, . . . , K\} \ i = 1. . . n\}$ be the training data set, where K is the number of classes.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 3, September 2013

At a node *t* of the tree, we divide the set of points *St* at that node in two subsets, i.e.,*St*+ and *St*− . *St*+ contains points of the majority class in *St*, whereas *St*− contains the rest of the points. We learn the tree as in the binary case discussed earlier. The only difference here is that we use the fraction of the points of the majority class to decide whether a given node is a leaf node or not. A complete description of the decision tree method for multiclass classification is given in Algorithm 1. Algorithm 1 recursively calls the procedure GrowTreeMulticlass(*St*), which will learn a split rule for node *t* and return a subtree at that node.

Steps in implementation of novel approach:-

1. At any given node, given the set of patterns *St*,
2. We find the two clustering hyperplanes (by solving the generalized eigenvalue value problem) and
3. Choose one of the two angle bisectors, based on the Information gain, as the hyperplane to be associated with this node.
4. We then use this hyperplane to split *St* into two sets, i.e., those that go into the left and right child nodes of this node.
5. We then recursively do the same at the two child nodes.
6. The recursion stops when the set of patterns at a node are such that the fraction of patterns belonging to the minority class of this set are below a user-specified threshold or the depth of the tree reaches a prespecified maximum limit.

User-specified threshold will help to generates trees with smaller depth with lesser number of leaves, compared with other decision tree approaches. So the method performs better than the other decision tree approaches in terms of accuracy, size of the tree, and time.

Algorithm : Algorithm Multiclass
Input: $S = \{(\mathbf{x}i, yi)\}^n_{i=1}$, Max-Depth, $\epsilon 1$
Output: Pointer to the root of a decision tree
begin
Root = GrowTreeMulticlass(*S*);
return *Root*;
end
GrowTreeMulticlass (S^t)
Input: Set of patterns at node *t* (S^t)
Output: Pointer to a subtree
begin
Divide set *St* in two parts, i.e., S^t_+ and S^t_- ;
S^t_+ contains points of the majority class, and S^t_- contains points of the remaining classes;
Find matrix *A* corresponding to the points of *St*+;
Find matrix *B* corresponding to the points of *St*−;
Find $\tilde{}\mathbf{w}1$ and $\tilde{}\mathbf{w}2$, which are the solutions of optimization problems (1) and (2);
Find angle bisectors $\tilde{}\mathbf{w}3$ and $\tilde{}\mathbf{w}4$ using (4);
Choose the angle bisectors having lesser information gain value. Call it $\tilde{}\mathbf{w}*$;
Let $\tilde{}\mathbf{w}t$ denotes the split rule at node *t*. Assign $\tilde{}\mathbf{w}^t \leftarrow \tilde{}\mathbf{w}*$;
Let $S^{tl} = \{\mathbf{x}i \in S^t | \tilde{}\mathbf{w}^T_t \tilde{}\mathbf{x} < 0\}$ and $S^{tr} = \{\mathbf{x}i \in S^t | \tilde{}\mathbf{w}^T_t \tilde{}\mathbf{x} \geq 0\}$;
Define $\eta 1(S^t) = (\max(n^t_1, \ldots, n^t_K))/(n^t)$;
if (*Tree-Depth* = *Max-Depth*) **then**
Get a node t_l, and make t_l a leaf node;
Assign the class label associated to the majority class to t_l;
Make t_l the left child of *t*;
else if ($\eta_1(S^{tl}) > 1 - \_1$) **then**
Get a node t_l, and make t_l a leaf node;
Assign the class label associated to the majority class in set S^{tl} to t_l;
Make t_l the left child of *t*;
else
t_l = GrowTreeMulticlass(S^{tl});
Make t_l the left child of *t*;
end
if (*Tree-Depth* = *Max-Depth*) **then**
Get a node t_r, and make t_r a leaf node;
Assign the class label associated to the majority class to t_r;
Make t_l the right child of *t*;
else if ($\eta 1(S^{tr}) > 1 - \_1$) **then**
Get a node t_r and make t_r a leaf node;
Assign the class label associated to the majority class in the set S^{tr} to t_r;
Make t^l the right child of *t*;
else
t_r = GrowTreeMulticlass(S^{tr});
Make t_r the right child of *t*;
end
return *t*
end

III. ANALYSIS OF OBLIQUE DECISION TREE CLASSIFIRES.

The core idea of the CART-LC algorithm is how it finds the value of δ that maximizes the goodness of split but the limitations of algorithm are, CART-LC is fully deterministic [6]. There is no built in mechanism for escaping local minima, although such minima may be very common for some domains. It produces only a single tree for given set of data. There is no upper bound on the time spent at any node in the decision tree. It halts when no perturbation changes the impurity more than €,but because impurity may increase and decrease, the algorithm can spend arbitrarily long time at a node. OC1 uses multiple iterations which improves the performance. The technique of perturbing the entire hyper plane in the direction of randomly chosen vector is good means for escaping from local minima. The oc1 algorithm produces remarkably small, accurate trees as compared to CART-LC. The algorithm differ from CART-LC as it can modify several coefficients at once where as CART-LC modifies one coefficient of the hyper plane at a time [10]. Breiman et al. report no upper bound on the time it takes for a hyper plane to reach a optimal position, where as OC1 accepts limited no of perturbations. SVM-ODT that exploits the benefits of multiple splits over single attributes and combines them with Support Vector Machine (SVM) techniques to take

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 3, September 2013

advantage of the accuracy of combined splitting on correlated numeric attributes. SVM system provides highly accurate classification in diverse and changing environments. The application of the particular ensemble algorithm is an excellent fit for online-learning applications where one seeks to improve performance of self-healing dependable computing systems based on reconfiguration by gradually and adaptively learning what constitutes good system configurations. SVM is however; more appealing theoretically and in practice, its strength is its power to address non-linear classification task The major strengths of SVM is the relatively easy training. No local optimal. It scales relatively well to high dimensional data .The trade-off between classifier complexity and error can be controlled explicitly. The weakness includes the need for a good kernel function The results of OC1 compared to SVM-ODT are with lower accuracy for overlapping datasets [5]. Classifier obtained with proposed algorithm is as good as that with SVM, whereas it is faster than SVM. The performance of proposed algorithm is comparable to that of SVM in terms of accuracy. Proposed algorithm performs significantly better than SVM on 10 and 100-dimensional synthetic data sets and the Balance Scale data set .The algorithm is effective in terms of capturing the geometric structure of the classification problem. For the first two hyperplanes learned by proposed algorithm approach and OC1 for 4 × 4 checkerboard data, It shows that proposed algorithm approach learns the correct geometric structure of the Classification boundary, whereas the OC1, which uses the Gini index as impurity measure, does not capture that[7]. Also, In terms of the time taken to learn the classifier, proposed algorithm is faster than SVM on majority of the cases[7]. Time wise GDT algorithm is much faster than OC1 and CART. The C4.5 algorithm basically used to implements Univariate DT's. AS Mutivariate DTs are advantageous. than univatriate DTs, the C4.5 approach is stated that it can be implemented for Mutivariate DT's by using Linear Machine approach, using the Absolute Error Correction and also the Thermal perceptron rules[9].

IV. EXPERIMENTAL RESULTS

In this section, we present empirical results to show the effectiveness of our decision tree learning algorithm. We test the performance of our algorithm on several synthetic and real data sets. We compare our results with among the best state-of-art oblique decision tree algorithms, OC1 [6] and CART-LC [1]. We also compare our results with among the best generic classifiers today, SVM classifier

Data Set Description: We generated four synthetic data sets in different dimensions, which are described here.

1) 2 × 2 *checkerboard data set*: 2000 points are sampled uniformly from $[-1\ 1] \times [-1\ 1]$. A point is labeled +1 if it is in $([-1\ 0] \times [0\ 1]) \cup ([0\ 1] \times [0\ -1])$; otherwise, it is labeled −1. Out of 2000 sampled points, 979 points are labeled +1, and 1021 points are labeled −1. Now, all the points are rotated by an angle of $\pi/6$ with respect to the first axis in counterclockwise direction to form the final training set. 2) 4 × 4 *checkerboard data set*: 2000 points are sampled uniformly from [0 4] × [0 4]. This whole square is divided into 16 unit squares having unit length in both dimensions. These squares are given indexes ranging from 1 to 4 on both axes.

Dataset	Method	Accuracy	Time(sec)	leaves	depth
2X2 Checker Board	New approach	99±0.20	0.31	4	2
	OC1	98±0.27	2.73±0.27	17.38±2.6	9.02±1.39
	CART	96±2.62	2.43±3.33	26.15±3.1	9.97±0.96
4X4 Checker Board	New approach	94±0.52	0.16	16	4
	OC1	93±0.46	4.14±0.44	82.83±6.3	15/14±1.6
	CART	88±2.75	4.24±2.57	91.79±9.0	14.61±1.1

Table I-comparison results between our approach and other classifiers

If a point falls in a unit square such that the sum of its two indices is even, then we assign label +1 to that point; otherwise, we assign label −1 to it.Out of 2000 sampled points, 997 points are labeled +1, and 1003 points are labeled −1. Now, all the points are rotated by an angle of $\pi/6$ with respect to the first axis in counterclockwise direction. Proposed algorithm has only one user-defined parameter, which is ϵ. For all our experiments, we have chosen ϵ using tenfold cross validation. SVM has two user-defined parameters, i.e., penalty parameter C and the width parameter σ for Gaussian kernel. The best values for these parameters are found using fivefold cross validation, and the results reported are with these parameters. Both OC1 and CART use 90% of the total number of points for training and 10% points for pruning. OC1 needs two more user-defined parameters. These parameters are the number of restarts R and the number of random jumps J. For our experiments, we have set $R = 20$ and $J = 5$, which are the default values suggested in the package. For the cases where we use GEPSVM with the Gaussian kernel, we found the best width parameter σ using fivefold cross validation *Simulation Results:* We now discuss the performance of propsed algorithm in comparison with other approaches on different data sets. The results provided are based on ten repetitions of tenfold cross validation. We show the average values and standard deviation (computed over the ten repetitions). Table I shows the comparison results of proposed algorithm with other decision tree approaches. In the table, we show the average and standard deviation5 for the accuracy, size, and depth of tree and the time taken for each of the algorithms on each of the problems. We can intuitively take the confidence interval of the estimated accuracy of an algorithm to be one standard deviation on either side of the average. Then, we can say that, on a problem, one algorithm has significantly better accuracy than another if the

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 3, September 2013

confidence interval for the accuracy of the first is completely to the right of that of the second. From Table I, we see that the average accuracy of proposed algorithm is better than all the other decision tree algorithms, proposed algorithm is significantly better than all the other decision tree approaches. Thus, overall, in terms of accuracy, the performance of the proposed algorithm is quiet good. In majority of the cases, proposed algorithm generates trees with smaller depth with lesser number of leaves, compared with other decision tree approaches. This supports the idea that our algorithm better exploits the geometric structure of the data set while generating decision trees. Time wise proposed algorithm algorithm is much faster than OC1 and CART, as can be seen from the results in the table. In most cases, the time taken by proposed algorithm is less by at least a factor of ten. We feel that this is because the problem of obtaining the best split rule at each node is solved using an efficient linear algebra algorithm in case of proposed algorithm, whereas these other approaches have to resort to search techniques because optimizing impurity based measures is tough. At every node of the tree, we are solving a generalized eigen value problem that takes time on the order of $(d + 1)3$, where d is the dimension of the feature space. On the other hand, SVM solves a quadratic program whose time complexity is $O(nk)$, where k is between 2 and 3 and n is the number of points. Thus, in general, when the number of points is large compared to the dimension of the feature space, proposed algorithm learns the classifier faster than SVM. By Experimental results we conclude that our approach learns the correct geometric structure of the classification boundary, whereas the OC1, which uses the Gini index as impurity measure, does not capture that. Although proposed algorithm gets the correct decision boundary for the 4×4 chessboard data set its cross validation accuracy is lesser than that of SVM. This may be because the data here are dense, and hence, numerical round-off errors can affect the classification of points near the boundary. On the other hand, if we allow some margin between the data points and the decision boundary (by ensuring that all the sampled points are at least 0.05 distance away from the decision boundary), then we observed that SVM and proposed algorithm both achieve 99.8% cross-validation accuracy. In the proposed algorithm described in Section II, ϵ is a parameter. If more than $(1 - \epsilon)$ fraction of the points fall into the majority class, then we declare that node as a leaf node and assign the class label of the majority class to that node. As we increase ϵ, chances of any node to become a leaf node will increase. This leads to smaller sized decision trees, and the learning time also decreases. However, the accuracy will suffer. We see that the cross-validation accuracy does not change too much with ϵ [5]. However, with increasing ϵ, the average number of leaves decrease. Thus, even though the tree size decreases with ϵ, the cross-validation accuracy remains in a small interval. This happens because, for most of the points, the decision is governed by nodes closer to the root node. Few remaining examples, which are tough to classify, lead the decision tree to grow further. However, as the value of ϵ increases, only nodes containing these tough-to-classify points become leaf nodes. We can say that ϵ in the range of 0.1–0.3 would be appropriate for all data sets [5].

V. CONCLUSION

Many different algorithms for induction of classifier models if trained with a big and diverse enough dataset perform with very high accuracy. But each has its own different strengths and weaknesses. Some perform better over discrete data, some with continuous, other classifiers have different tolerance for noise, and they have different speed of execution. Each algorithm focused on improving known machine learning techniques by introducing new ways of combining the strengths of different approaches to achieve higher performance. CART-LC and OC1 are the basic classifiers in which oc1 can perform better than CART-LC.the new standard classifiers are SVM and GDT. SVM provides highly accurate classification in diverse and changing environments. GDT is faster than SVM and performs significantly better than SVM in terms of accuracy.

REFERENCES

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and Regression Trees." Belmont, Ca: Wadsworth and Brooks, 1984, Ser. Statistics/Probability Series.

[2] J. Quinlan, "Induction Of Decision Trees," Mach. Learn., Vol. 1, No. 1, Pp. 81–106, 1986.

[3] K. P. Bennett and J. A. Blue, "A Support Vector Machine Approach To Decision Trees," In Proc. Ieee World Congr. Comput. Intell., Anchorage, Ak, May 1998, Vol. 3, Pp. 2396–2401.

[4] S. K. Murthy, S. Kasif, And S. Salzberg, "A System For Induction Of Oblique Decision Trees," J. Artif. Intell. Res., Vol. 2, No. 1, Pp. 1–32, 1994.

[5] Naresh Manwani And P. S. Sastry, "Geometric Decision Tree" , Ieee Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics, Vol. 42, No. 1, February 2012

[6] Shesha Shah And P. S. Sastry," New Algorithms For Learning And Pruning Oblique Decision Trees" Ieee Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews, Vol. 29, No. 4, November 1999

[7] Erick Cantú-Paz, Chandrika Kamath, "Inducing Oblique Decision Trees With Evolutionary Algorithms". IEEE Transaction on Evolutionary Computation, Vol. 7, No. 1, February 2003.

[8] Murthy, Kasif, Salzberg. "A System for Induction of Oblique Decision Trees." Journal Of Artificial Intelligence Research 2 (1994) 1-32

[9] Vlado Menkovski, Ioannis T. Christou, and Sofoklis Efremidis, "Oblique Decision Trees Using Embedded Support Vector Machines In Classifier Ensembles", IEEE Cybernetic Intelligent Systems (2008) 1-6.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 3, September 2013

[10] Shreerama Murthy,Simon Kasif,Stivon Salzberg,Richard Beigel,“ Oc1:Randomized Induction Of Oblique Decision Tree”

[11] Guy Michel, Jean Luc Lambert, Bruno Cremilleux & Michel Henry-Amar, “A New Way To Build Oblique Decision Trees Using Linear Programming”

[12] Thales sehn Korting, “C4.5 Algorithm and Mutivibrate Decision Teees”.

AUTHOR’S PROFILE

Son al Patil received the B.E. degree in Computer science and engineering from NMU India, and the M.Tech. Degree in software systems from Bhopal University, India.. She is currently working as Assistant Professor in GHRIEM, India. Her research interests are data mining, machine learning and pattern recognition, computational Neuroscience.

Son al Badhe received the B.E. degree in Computer science and engineering from University of pune India, and she is currently working toward the ME degree in the Department Computer science and Engineering, GHRIEM, Jalgaon, India. Her research interests are data mining, machine learning and pattern recognition.