

Oblique Decision Tree Learning Approaches - A Critical Review

Setu Chaturvedi, Ph.D
H.O.D.(CSE. Dept)
TIT,
Bhopal

Sonal Patil
Second Year, Mtech
TIT,
Bhopal

ABSTRACT

Decision tree classification techniques are currently gaining increasing impact especially in the light of the ongoing growth of data mining services. A central challenge for the decision tree classification is the identification of split rule and correct attributes. In this context, the article aims at presenting the current state of research on different techniques for classification using oblique decision tree. A variation to the traditional approach is the called oblique decision tree or multivariate decision tree, which allows multivariate tests in its non-terminal nodes. Univariate trees can only perform axis-parallel splits, whereas Oblique decision trees can model the decision boundaries that are oblique to attribute axis. The majority of these decision tree induction algorithms performs a top-down growing tree strategy and rely on an impurity-based measure for splitting nodes criteria. In this context, the article aims at presenting the current state of research on different techniques for Oblique Decision Tree classification. For this, the paper analyzes various traditional Multivariate and Oblique Decision Tree algorithms CART, OC1 as well as standard SVM, GDT implementation.

General Terms

.Data mining ,algorithm for tree building.

Keywords

oblique decision tree,CART,OC1,SVM,GDT

1. INTRODUCTION

1.1 Classification Model

A classification model is built from 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
- Developing the classification model.

1.2 Classifier Issues

Some issues that must be considered 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 a tradeoff 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.

1.3 Decision tree

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].

1.4 Univariate and Multivariate Decision tree

Multivariate decision trees differ from univariate decision trees in the way they test the attributes. Univariate decision trees test single attribute at internal node. Multivariate decision tree several attribute participate in single node split test. The limitation to one attribute reduces the ability of expressing concepts, due to its disability in three forms. Splits could only be orthogonal to axes, subtrees may be replicated and fragmentation.

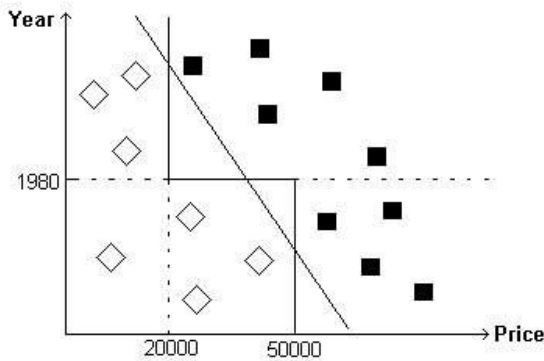


Figure 1. Comparison of univariate and multivariate splits on the plane

This example shows the well-known problem that a univariate test using feature x_i can only split a space with a boundary that is orthogonal to the x_i axis. This results in larger trees and poor generalization. The multivariate decision tree-constructing algorithm selects not the best attribute but the best linear

combination of the attributes:
$$\sum_{i=1}^f w_i x_i > w_0 \cdot w_i$$

are the weights associated with each feature x_i and w_0 is the threshold to be determined from the data. .

Multivariate decision trees differ from univariate trees as the symbolic features are converted into numeric features. And all splits are binary, final weighted sum is numeric.

1.5 Oblique Decision tree

Tree induction algorithms like Id3 and C4.5 create decision trees that take into account only a single attribute at a time. For each node of the decision tree an attribute is selected from the feature space of the dataset which brings maximum information gain by splitting the data on its distinct values. The information gain is calculated as the difference between the entropy of the initial dataset and the sum of the entropies of each of the subsets after the split.

Id3 selects at each node the split on the attribute which gives the biggest gain Such trees 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[1] .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.

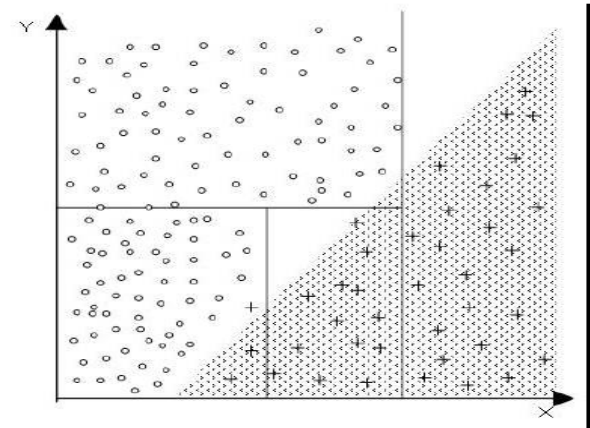


Figure 2. Feature space of two attributes X and Y, several axis parallel splits and representation of the region of both classes.

2. OBLIQUE DECISION TREE CLASSIFICATION ALGORITHMS

2.1 CART-LC

The first oblique decision tree algorithm to be proposed was CART with linear combinations .Breiman, Friedman, Olsen, and Stone (1984) introduced CART with linear combinations(CART-LC) as an option in their popular decision tree algorithm CART. At each node of the tree, CART-LC iteratively finds locally optimal values for each of the a_i coefficients. Hyper planes are generated and tested until the marginal benefits become smaller than a constant [10].

Recognizing that the oblique splits are harder to interpret than a simpler univariate split, Breiman et al. used a backward deletion procedure to simplify the structure of the split by weeding out variables that contribute little to the effectiveness of the split.

2.2 OC1

Murthy, Kasif, and Salzberg (1994) introduced OC1, which uses an ad-hoc combination of hill climbing and randomization. CART-LC, uses hill climber that finds locally optimal values for one coefficient at a time.OC1 offers several variants to select the order in which the coefficients are optimized. The randomization of component takes two form. OC1 implements multiple random restarts; the hyper plane is perturbed in a random direction when hill climbing reaches a local minimum.

Murthy et al. present OC1 as an efficient algorithm that overcomes difficulties and limitations of CART-LC. The deterministic nature of CART-LC may cause it to get trapped in local minima, and by using randomization may improve the quality of the decision tree [6]. OC1 produces multiple trees using the same data, and the time used at each node in the tree is limited.

```

Induce Split node T of the Decision Tree.

Normalize values of all D attributes.

L=0

While (true)

    L=L+1

    Search for  $\delta$  that maximizes the goodness of split

    And Sum variance of two resulting nodes should be
    minimized.

    If  $| \text{goodness}(s_L) - \text{goodness}(s_{L-1}) | < \epsilon$  ,

    Exit while loop.

Eliminate irrelevant attributes using backward elimination.

Convert  $s_L$  to split on the un-normalized attributes.

Return the better of  $s_L$  and the best axis-parallel split as the
split for T.
    
```

Figure 3. The procedure used by CART with linear combinations (CART-LC) at each node of a decision tree

2.3 SVM

The iterative simplification procedure deletes one variable at a time until no further variables can be deleted, and the coefficient optimization algorithm is executed on the remaining variables.

SVM are based on statistical learning theory. It can be used for learning to predict future data. SVM implements mapping .SVM builds an optimal hyper-plane (or multiple hyper planes) separating the instances of data belonging to different sets, setting the hyper-plane's position so that it maximizes the margin of each class from the hyper plane, while minimizing the number of points lying on the "many" side of the hyper-plane[11]. The result of the SVM technique is a hyper plane described by the equation $wTx + w_0 = 0$. This approach brings high accuracy in the separation of the data.

It combines generalization control as a technique to control dimensionality. The kernel mapping enables comparisons by providing a common base for most of the commonly employed model architectures. In classification problems generalization control is obtained by maximizing the margin and minimization of the weight. The set of support vectors i.e. the solution can be sparse. The minimization of the weight vector with a modified loss function can be used as a criterion in regression problems.

```

To find a split of a set of examples T:

Find the best axis-parallel split of T. Let I be impurity of this
split.

Repeat R times:

Choose a random hyper plan H.

(for the first iteration. Initialize H to be the best axis -parallel
split.)

Step 1: Until the impurity measures does not improve do;

    Perturbe each of coefficient of H in sequence .

Step 2: Repeat(at most j times)

    Choose a random direction and attempt to perturb H in
    that direction.

If this reduces the impurity of H.go to step 1.

Let  $I_1 =$  the impurity of H . if  $I_1 < I$  ,then set  $I = I_1$ .

Output the split corresponding to I
    
```

Figure 4. Overview of the OC1 algorithm for a single node of a decision tree

2.4 GDT

For Given set of training patterns at a node, we first find two hyper planes, i.e., one for each class .Each hyper plane 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 hyper planes as the clustering hyper planes (for the two classes). Because of the way they are defined, these clustering hyper planes capture the dominant linear tendencies in the examples of each class that are useful for discriminating between the classes. Hence, a hyper plane that passes in between them could be good for splitting the feature space. Thus, we take the hyper plane that bisects the angle between the clustering hyper planes 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 Gini index. 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[7].

At each node

Find clustering hyperplanes for each class.

Clustering hyperplanes are found using Multisurface Proximal SVM

$$(\mathbf{w}_1, b_1) = \operatorname{argmin}_{\mathbf{w}, b} \frac{\frac{1}{\|\mathbf{w}\|^2} \sum_{\mathbf{x}_i \in C_1} |\mathbf{w}^T \mathbf{x}_i + b|^2}{\frac{1}{\|\mathbf{w}\|^2} \sum_{\mathbf{x}_i \in C_2} |\mathbf{w}^T \mathbf{x}_i + b|^2}$$

$$(\mathbf{w}_2, b_2) = \operatorname{argmin}_{\mathbf{w}, b} \frac{\frac{1}{\|\mathbf{w}\|^2} \sum_{\mathbf{x}_i \in C_2} |\mathbf{w}^T \mathbf{x}_i + b|^2}{\frac{1}{\|\mathbf{w}\|^2} \sum_{\mathbf{x}_i \in C_1} |\mathbf{w}^T \mathbf{x}_i + b|^2}$$

Find angle bisectors

$$\text{Let } \tilde{\mathbf{w}}_a^* = (\mathbf{w}_a^*, b_a^*) \text{ and } \tilde{\mathbf{w}}_b^* = (\mathbf{w}_b^*, b_b^*)$$

be parameters of angle bisector then

$$(\mathbf{w}_a^*, b_a^*) = \left(\frac{\mathbf{w}_1}{\|\mathbf{w}_1\|} + \frac{\mathbf{w}_2}{\|\mathbf{w}_2\|}, \frac{b_1}{\|\mathbf{w}_1\|} + \frac{b_2}{\|\mathbf{w}_2\|} \right)$$

$$(\mathbf{w}_b^*, b_b^*) = \left(\frac{\mathbf{w}_1}{\|\mathbf{w}_1\|} - \frac{\mathbf{w}_2}{\|\mathbf{w}_2\|}, \frac{b_1}{\|\mathbf{w}_1\|} - \frac{b_2}{\|\mathbf{w}_2\|} \right)$$

Find one which best improves purity

Split the dataset

Repeat above till child nodes become pure (almost)

2.5 C4.5

C 4.5 can be used to construct Multivariate decision trees with the Linear Machine approach, using the Absolute Error Correction and also the Thermal perceptron rules. Considering a multiclass instance set, we can represent the multivariate tests with a Linear Machine (LM)[12].

- LM: Let if an instance description consisting of 1 and the n features that describes the instance. Then each discriminant function has the form consisting of vector of n + 1 coefficients. The LM infers instance to belong to class
- Absolute Error Correction rule: One approach for updating the weight of the discriminant functions is the absolute error correction rule, which adjusts w_i , where i is the class to which the instance belongs, and w_j , where j is the class to which the LM incorrectly assigns the instance.
- Thermal Perceptron: For not linearly separable instances, one method is the “thermal perceptron”, that also adjusts w_i and w_j , and deals with some constants.

3. ANALYSIS OF OBLIQUE DECISION TREE CLASSIFIERS

3.1 Analysis of CART-LC

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 ϵ , but because impurity may increase and decrease, the algorithm can spend arbitrarily long time at a node.

3.2 Analysis of OC1

OC1 uses multiple iterations which improves the performance. The technique of perturbing the entire hyperplane 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 hyperplane at a time [10]. Breiman et al. report no upper bound on the time it takes for a hyperplane to reach a optimal position, where as OC1 accepts limited no of perturbations.

3.3 Analysis of SVM-ODT

SVM-ODT that exploits the benefits of multiple splits over single attributes and combines them with Support Vector Machine (SVM) techniques to take 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 [11].

3.4 Analysis of GDT

Classifier obtained with GDT is as good as that with SVM, whereas it is faster than SVM. The performance of GDT is comparable to that of SVM in terms of accuracy. GDT 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 GDT approach and OC1 for 4×4 checkerboard data, It shows that GDT 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, GDT is faster than SVM on majority of the cases[7]. Time wise GDT algorithm is much faster than OC1 and CART

3.5 Analysis of C4.5

The C4.5 algorithm basically used to implements Univariate DT's. AS Mutivariate DTs are advantageous than univariate 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[12].

4. 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.

5. 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
- [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 Tees"