Improving Geometrical Structure of Data Using Geometric Decision Tree

Miss. Shubhangi C. Nasare1, Dr. A. B. Bagwan2, Dr. Pradeep K. Deshmukh3

1Student, Dept. of CSE, RSCOE, Pune, India
2Dr, HOD, Dept. of CSE, RSCOE, Pune, India
3Dr, Dept. of CSE, RSCOE, Pune, India

ABSTRACT

For representing the classifier Decision trees are the great approach. This paper presents a very newfound algorithm for acquiring a Oblique decision trees. For acquiring a decision tree in top-down fashion maximum decision tree algorithms trust on impurity related measures to evaluate the goodness of hyper planes at every node while developing a decision tree in top-down fashion. These impurity measures are not properly represent the geometric Structures in the data. Our algorithm implements a Strategy for evaluating the hyper planes in such a way that the Geometric structure in the data is filled into account. At every node Of the decision tree, There is the clustering hyper planes for both the classes and implement their angle bisectors as the split rule at that particular kind of node. We mention through experimental studies that this idea guides to small decision trees and for better performance. We also show some analysis to represent that the angle bisectors of clustering hyper planes that we implement as the split rules at every node are problem solutions of an eliciting optimization problem and hence represent that this is a high principled method of acquiring a decision tree.

Keywords: Decision trees, multiclass classification, Oblique decision tree, generalized eigen value problem.

1. INTRODUCTION

For the classification, A decision tree is a well-known and widely used method. As a classification rule decision trees popularity is because of its simplicity and easy clearness. In a decision tree classification, every non leaf node is associated with a so-called decision function or a split rule, that is a function of the feature vector and is binary valued. Every leaf node in the tree is specified with a class label. For classification a feature vector by using a decision tree, at each non leaf node that we run across (starting with the root node), we combine to one of the sibling of that node completely based on the value expected by the split rule of that node on the specified feature vector. This activity follows a way in the tree, and when we arrive at a leaf, the class label of the leaf is what is appointed to that feature vector. In this paper, we shows the difficulty of learning an oblique decision tree, given a set of labeled training ingest s. We mention a algorithm that attempts to create the tree by representing the geometric structure of the all class regions.

There are two classified ways of decision trees i.e., axis parallel and oblique [1]. In the axis-parallel decision tree, the split rule at every node is a function of only one of the attempt of the vector. When all features are nominal then Axis-parallel decision trees are particularly attractive; in such type of cases, we have a non binary tree where, at every node, we check one feature value, and the node can have as so many children as the values presumed by that feature [2]. However, in many general situations, we have to resemble even arbitrary linear segments in the conditions of class boundary with many axis-parallel pieces; hence, the area of the resulting tree becomes large. The oblique decision trees, on the other hand, follow a decision function that totally depends on a linear combination of all components. Thus, an oblique decision tree is a binary tree where we mention a hyperplane with every node. For classification of pattern, we use a path in the tree by considering the left or right child at every node based on which align of the hyperplane (of that node) the vector falls in.

There are two broad categories to classify the approaches for learning oblique decision trees. In one set of approaches, the structure of the tree is constant beforehand, and we try to acquire the optimal tree with this constant structure. Several researchers has been adopted this methodology, and very different optimization algorithms have been proposed [3]–[8]. The main problem with these methods is that they are applicable only in that situation where we know the structure of the tree a priori, which is not the case. The other classes of methods have been more popular because of their versatility.

Top-down methods are recursive algorithms for creating the tree in a top-down fashion. We move with the given training data and determine on the “best” hyperplane, which is allocated to the root of the tree. Then, we divide the training examples into two sets that go to the left child and the right child of the root node by using current hyperplane. Then, at each & every of the two sibling nodes, we repeat the same procedure. The recursion exits when the all set of examples that come to a node is in clean form, that is, all these instruction models are of the similar class. Then, we create it a leaf node and allocate that class to the leaf node. A elaborate survey of top-down decision tree algorithms is available in [9]. There are two main problems in top-down decision tree learning algorithms: 1) given the training samples at a node, how to
give rating on different hyperplanes that can be associated with this node and, 2) given a rating function, how to calculate the optimal hyperplane at each and every node.

In this paper, we mention a fresh decision tree learning algorithm, which is based on the plan of capturing, to a few level, the geometric structure of the essential class regions. For this, we make use of ideas from some recent variants of the support vector machine (SVM) method, which are fairly first-rate at capturing the (linear) geometric structure of the data. In all-purpose, there will be two angle bisectors; we choose that which is superior based on an impurity measure. Thus, the algorithm combines the facts of linear tendencies in data and purity of nodes to find improved decision trees. We also present some examination to take out some interesting properties of our angle bisectors that can give details why this may be a good method to learn decision trees.

The representation of any top-down decision tree algorithm depends on the measure used to price different hyperplanes at every node. The problem of having a suitable algorithm to find the hyperplane that optimizes the selected rating function is also significant. For example, for all impurity measures, the optimization is tricky because finding the incline of the impurity function with respect to the parameters of the hyperplane is not potential. Motivated by these considerations, here, we recommend a new criterion function to measure the suitability of a hyperplane at a node that can capture the geometric construction of the class regions. For our principal function, the optimization obscurity can also be solved more straightforwardly.

We first explain our method by bearing in mind a two-class problem. Given the set of training patterns at a node, we first find two hyperplanes, i.e. one for every class. Every hyperplane is such that it is neighboring to all patterns of one class and is outermost from all patterns of the other class. We entitle these hyperplanes as the clustering hyperplanes. We mention the hyperplane that bisects the angle between the clustering hyperplanes as the split rule at this node. Selecting any of the angle bisectors as the hyperplane of a tree at the root node to split the data results into subsequent separable classification problems at both child nodes. Thus, we see here that our scheme of using angle bisectors of two clustering hyperplanes really captures the right geometry of the classification problem. This is the reason we name our approach “geometric decision tree (GDT).”

We also note here that neither of our angle bisectors scores high on any impurity based measure; if we use either of these hyperplane as the split rule at the root, both child nodes would contain roughly equal number of patterns of every class. This instance is only for clearing up the motivation behind our approach. Not all classification problems have such a nice symmetric structure in class regions.

2. RELATED WORK

Most studies describes the use of classic algorithms In [2], two parallel hyperplanes are cultured at each node such that one surface of each hyperplane contains points of only one class and the gap between these two hyperplanes contains the points that are not detachable. A small alternative of the aforesaid algorithm is planned in [3], where only one hyperplane is cultured at each decision node in such a way that one surface of the hyperplane contains points of only one class. The decision tree learning for a special multiclass classification problems using linear-classifier-based approaches is discussed in [4], [5]. As an alternative of finding a linear classifier at each node, Cline[6], which is a family of decision tree algorithms, uses a variety of heuristics to decide hyperplanes at each node. Conversely, they do not supply any results to show why these heuristics help or how one chooses a scheme. A Fisher-linear-discriminant-based conclusion tree algorithm is proposed in [7]. All the previously discussed approaches produce crisp decision limitations. The decision tree loom giving probabilistic decision border is discussed in [8]. Its blurry variants are discussed in [9] and [10]. The Gini index and entropy are some of the regularly used impurity measures [1]. Evolutionary approaches are broad-minded to noisy evaluations of the rating function and also make easy optimizing multiple rating functions concurrently [11], [12]. In [13], a different advance is recommended, where the function for ranking hyperplanes gives elevated values to hyperplanes, which endorse the “the degree of linear separability” of the set of patterns corridor at the child nodes. It has been found experimentally that the decision trees educated using this criterion are more compressed than those using impurity method. The charge function is not differentiable with respect to the parameters of the hyperplanes, and the method uses a stochastic search technique called Alopex [14] to find the optimal hyperplane at each node.

3. IMPLEMENTATION DETAILS.

3.1 System Architecture

![Diagram](Figure 1: System Architecture)
3.2 Data flow of System

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

3.3 Mathematical Model

Finding Clustering Hyper planes

\( S = \{(x_i, y_i) | x_i \in [-1,1], i = 1...n\} \) be the training data set. Let \( C^+ \) be the set of points in which \( y_i = 1 \). In addition, let \( C^- \) be the set of points for which \( y_i = -1 \). \( w_1^T x + b_1 = 0 \) and \( w_2^T x + b_2 = 0 \) be the two clustering hyperplanes.

Hyperplane \( h_1 \) is to be closest to all points of class \( C^+ \) and farthest from points of class \( C^- \).

\[
\begin{align*}
D_+(w,b) &= \frac{1}{n^+} \sum_{x_i \in C^+} \|w^T \tilde{x}_i\|^2 \\
&= \frac{1}{n^+} \sum_{x_i \in C} \tilde{x}_i \tilde{x}_i^T w = \frac{1}{\|w\|^2} \tilde{w}^T G \tilde{w} \\
\tilde{w}_1 &= \arg \min_{\tilde{w} \neq 0} \tilde{w}^T G \tilde{w}
\end{align*}
\]

\[
\begin{align*}
D_-(w,b) &= \frac{1}{n^-} \sum_{x_i \in C^-} \|w^T \tilde{x}_i\|^2 \\
&= \frac{1}{n^-} \sum_{x_i \in C} \tilde{x}_i \tilde{x}_i^T w = \frac{1}{\|w\|^2} \tilde{w}^T G \tilde{w} \\
\tilde{w}_2 &= \arg \min_{\tilde{w} \neq 0} \tilde{w}^T G \tilde{w}
\end{align*}
\]

Gini index/ information gain finding

\[
\begin{align*}
\text{Gini}(\tilde{w}_i) &= \frac{n^{l'}}{n} \left[ \frac{(n^l n^{l'})^2}{n^l n^{l'} + n^{r'} n^{r''}} \right] + \frac{n^{r'}}{n} \left[ \frac{(n^l n^{r'})^2}{n^l n^{r'} + n^{l'} n^{r''}} \right] \\
&= \tilde{w}_3 + \tilde{w}_4
\end{align*}
\]

Where \( n^l = n^{l'} + n^{l''} \) is the number of points in \( S^l \). In addition, \( n^l = n^{l'} + n^{l''} \) is the number of points falling in set \( S^l \) and \( n^{r'} = n^{l'} + n^{r''} \) is the number of points falling in set \( S^r \). We choose \( \tilde{w}_3 \) or \( \tilde{w}_4 \) to be the split rule \( S^l \) based on which of the two gives lesser value of the gini index.

When the clustering hyperplanes are parallel (that is when \( w_1 = w_2 \)), we choose a hyperplane given by \( w = (w, b) = (w_1(b_1 + b_2)/2) \) as the splitting hyperplane.

Grow tree method for making tree by algorithm.

\[
I_E(f) = -\frac{1}{m} \sum_{i=1}^{m} f_i \log_2 f_i
\]

Algorithm: GDT

Input: \( S \) = \( \{(x_i, y_i)\}^{n}_{i=1} \) Max-Depth, E1
Output: Pointer to the decision trees root
begin
    Root = Growtreemulticlass(S);
    return Root;
end
Growtreemulticlass(St)
Input: set of patterns at node \( t \) (St)
Output: Pointer to a subtree
begin
    Divide set St in two parts, ie S_l^t and S_r^t ;
    S_l^t contains points of the majority class, and S_r^t contains points of the remaining classes;
    Find matrix A corresponding to the points of S_l^t ;
    Find matrix B corresponding to the points of S_r^t ;
    Find \( \vec{w}_l \) and \( \vec{w}_r \), which are the solutions of optimization problems;
    Find angle bisectors \( \vec{w}_l \) and \( \vec{w}_r \); choose the angle bisectors having lesser gini index value. call it \( \vec{w} \);
    Let \( \vec{w}_l \) denotes the split rule to node t. give
    \( \vec{w}_l \rightarrow \vec{w}_r \)
    Let \( S_l^t = \{ X_i \in S^t | \vec{w}^T \vec{x} < 0 \} \) and \( S_r^t = \{ X_i \in S^t | \vec{w}^T \vec{x} \geq 0 \} \)

Let Define n1(st) = (max( \( n_1 \) \( n_2 \) \( n_3 \) \( n_4 \) ) / ( \( n_1 \) ));
if(Tree-Depth = max- Depth) then
    Get a node tl, and make tl a leaf node;
give the class label related to the majority class to tl;
    Make tl the left child of t;
else if (n1(stl)>1-e1) then
    Get a node tl, and make tl a leaf node;
give the class label related to the popular class in set Stl to tl;
    Make tl the left child of t;
else
    tl=Growtree multiclass(Stl);
    Make tl the left child of t;
end if(Tree-Depth = max- Depth) then
    Get a node tr, and make tr a leaf node;
give the class label related to the popular class to tr;
    Make tr the right child of t;
else if (n1(str)>1-e1) then
    Get a node tr, and make tr a leaf node;
give the class label related to the popular class in set Str to tr;
    Make tr the right child of t;
else
    tr=Growtree multiclass(Str);
    Make tr the right child of t;
end
return t
end

4. RESULT SET
In this section, we present experimental results to show the usefulness of our decision tree learning algorithm. We check the performance of our algorithm on several imitation and real data sets.

Data Set Description: We generated following synthetic data sets in different dimensions, which are described here. In graphical analysis y axis shows the accuracy and x axis shows the overall (P) positive & (M) negative accuracy. Firstly we enter some input values through data set, for training dataset. and creates a tree after that from training data values it will select 40% randomly values and passes that values from root like passes first line then second line and generate the tested result in testing data. Tested data shows that exact accuracy of your training data because it is tested data and it shows the exact accuracy.

1) 2 X 2 checkerboard data set: 1000 rows & 2 columns are sampled uniformly.
2) 4X4 checkerboard data set: 1000 rows & 2 columns are sampled uniformly.

Experimental Setup:
The performance of GDT is differentiable to that of SVM in terms of accuracy. GDT performs definitely better than SVM on 10 and 100-dimensional synthetic data sets and the Balance Scale data set. GDT performs comparable to SVM on the 2x2 checkerboard. GDT performs worse than SVM on the 4x4 checkerboard data sets.

In the terms of the time taken to acquire the classifier, GDT is quicker than SVM on popular of the cases. At each node of the tree, we are evaluating a comprehensive eigen value crisis that takes time on the order of \((d+1)^3\), where \(d\) is the dimension of the feature space. Thus, in general, when the number of points is bulky compared to the measurement of the characteristic space, GDT learns the classifier quicker than SVM.

Finally, Fig., we show the efficiency of our algorithm in terms of capturing the geometric constitution of the classification problem.

We see that our loom learns the exact geometric structure of the classification boundary, whereas the OC1, which uses the Gini index as adulteration measure, does not capture that.

Even though GDT gets the exact decision boundary for the 4x4 checkerboard dataset, as shown in Fig. Its cross-validation exactness is lesser than that of SVM. This may be because the data here are solid, and hence, numerical round-off errors can change the classification of points close to the boundary. On the other hand, if we allocate some margin sandwiched between the data points and the decision boundary then we observed that SVM and GDT both achieve 99.8% cross-validation accuracy.

In the GDT algorithm explained, is a parameter. If more than 1 fraction of the points go down into the majority class, then we state that node as a leaf node and allocate the class label of the widely held class to that node. As we enlarge \(1\), chances of any node to become a leaf node will enlarge. This leads to lesser sized decision trees, and the learning time also decreases. However, the accurateness will bear.

5. CONCLUSION AND FUTURE WORK
In this paper, we have offered a new algorithm for acquiring oblique decision trees. The originality is in learning hyperplanes that captures the geometric structure of the class regions. At each node, we have established the two clustering hyperplanes and selected one of a angle bisectors as the split rule. We have offered some analysis to gain the optimization problem for which the angle bisectors are the resolution. Based on information gain, we argued that our method of choosing the hyperplane at every node is sound. Through wide experiential studies, we mentioned that the method performs enhanced than the other decision tree approaches in terms of accuracy, space of the tree, and time. We have also exposed that the classifier obtained with GDT is as good quality as that with SVM, whereas it is quicker than SVM. Thus, overall, the algorithm presented here is an excellent and narrative classification method. In future by considering new factors it will also work in very excellent manner.

References


AUTHORS

Shubhangi Chandrakant Nasare is student of Master of Computer Engg., Department of Computer Engineering, JSPM’s Rajarshi Shahu College of Engg, Tathawade, Pune 411033, India. Her research interest is Data Mining.

Dr. A. B. Bagwan, Head of Department of Computer Engineering, RSCOE College, S.NO. 80/3, Tathawade, Pune 411033, India. His research interest is Data Mining.

Dr. Pradeep K. Deshmukh, Professor of Department of Computer Engineering, RSCOE College, S.NO. 80/3, Tathawade, Pune 411033, India. His research interest is Mobile Computing, Networking, Data Mining.