Multi Branch Decision Tree: A New Splitting Criterion[1]

Hadi Sadoghi Yazdi[1,2] and Nima Salehi-Moghaddami[1]

[1]*Department of Computer Engineering,*
Ferdowsi University of Mashhad, Mashhad, Iran
[2]*Center of Excellence on Soft Computing and Intelligent Information Processing,*
Ferdowsi University of Mashhad, Mashhad, Iran
h-sadoghi@um.ac.ir

Abstract

In this paper, a new splitting criterion to build a decision tree is proposed. Splitting criterion specifies the best splitting variable and its threshold for further splitting in a tree. Giving the idea from classical Forward Selection method and its enhanced versions, the variable having the largest absolute correlation with the target value is chosen as the best splitting variable in each node. Then, the idea of maximizing the margin between classes in SVM is used to find the best threshold on the selected variable to classify the data. This procedure will execute recursively in each node, until reaching the leaf nodes. The final decision tree has a comparable shorter height than the previous methods, which effectively reduces more useless variables and the time of classification for future data. Unclassified regions are also generated, which can be interpreted as an advantage or disadvantage for the proposed method. Simulation results demonstrate this improvement in the proposed decision tree.

Keyword: *Decision Tree, Splitting Criterion, Support Vector Machine, Correlation, Unclassified Region*

1. Introduction

Decision tree (DT) is one of the most important knowledge representation approaches which attempts to build a top-down model in a simplified subspace by projecting high-dimensional data down to a low-dimensional space. The process of projection is usually done by eliminating duplicated or redundant attributes or neglecting less important ones. Specifically, Splitting criterion in a tree is specified by selecting the best splitting variable and its threshold for the further split. In classification, DT model is used to increase the system prediction for classifying oncoming data. Despite strong competitors like Linear Regression and ANN, DTs have several advantages; they are simple to understand and interpret, inexpensive in computation and capable of dealing with noisy data [1].

CART algorithm [2] selects splits using the Twoing Criteria and prunes the tree by Cost-Complexity pruning. CART has the ability to generate regression trees. The very simple decision tree algorithm (ID3) was first proposed by [3] as a classifier. ID3 uses information gain as the splitting criterion. It does not apply any pruning procedure and cannot handle numeric values and missing data. In C4.5 algorithm [4], evolution of ID3, the splitting process ceases when the number of instances to be split is below a certain threshold. C4.5

[1] This project supported by grant No. 2/15363 from the Ferdowsi University of Mashhad, Engineering Department

uses error-based pruning and can handle numeric attributes. It can also induce from a training set with missing values. Due to the extensive application domain, numerous researchers focus on improving these principal algorithms, or even propose a new splitting criterion with exclusive characteristics.

There are numerous old and new algorithms that work totally different from the common introduced DT algorithms. CHAID algorithm [5] as an old one applies statistic procedures to generate a novel decision tree. It uses the concept of statistical $p-$value to perform splitting effectively. Another algorithm named LMDT [6] constructs a decision tree based on multivariate tests, which are linear combinations of the attributes. The QUEST algorithm [7] supports uni-variate and linear combination splits. This algorithm uses ANOVA F-test or Levene's test or Pearson's chi-square test to compute the association between each input attribute and the target attribute. The QUEST applies Quadratic Discriminant Analysis (QDA) to find the optimal splitting point; earlier version of QUEST was named FACT and can be found in [8]. A profound comparison of these classic algorithms and others algorithms like CAL5 [9] and PUBLIC algorithm [10] have been conducted in [11]. The new DT algorithm proposed in [12] has better simulation results on classifying sample databases than famous DT algorithms like C4.5 and C5. The algorithm named ROC-tree is based on considering the area under the Receiver Operating Characteristics (ROC) curve, to help determine the decision tree characteristics, such as node selection and stopping criteria.

Many researchers focus on improving the principals of DT algorithms. Efficient C4.5 [13] is an improvement over C4.5 in which three strategies are suggested to choose the best information gain. First and second strategies, consequently, use quick and counting sort, and the third one computes the local threshold of C4.5 using the main memory version of the Rainforest algorithm, which does not need sorting. In [14] classic decision tree is promoted by inserting linguistic terms instead of numerical quantities in tree node labels.

The evolutionary method that is presented in [15], allows decision tree flexibility by using co-evolving competition between the decision tree and the training data set. The algorithm mutates the decision tree while updating data set size and DT depth. Another paper presented several approaches to the induction of decision trees for HMC (Hierarchical Multi-label Classification), as well as an empirical study on their use in functional genomics [16]. In [17] a classification algorithm for learning decision tree classifiers from data with hierarchical class labels proposed; as many practical classification problems concern data with class labels that are naturally organized as a hierarchical structure.

Some researchers have focused on improving classic DTs characteristics; like minimization of DT, improving the DT pruning procedure or changing splitting measures. Decision tree pruning is improved in [18] through Automatic Programming and results in rewriting the code of the error based pruning, which is an important part of C4.5 decision tree learning algorithm. In [19] a detailed discussion about minimization of DT and its importance in decision making speed is presented. The author has demonstrated that, the Minimization of decision trees is hard to approximate.

Applications of classification-based DTs predominate in different fields of science and medicine. For example, a simple decision tree to classify clinical specimens of patients who suffer from lung cancer, as Diseased or Non-diseased was applied [20]. In [21] a DT is used to predict radon gas concentration from other environmental factors, which leads to possible future earthquake predicting system. Since DT algorithms choose relevant variables and create a moderate number of rules, they can be used as a preprocessing phase in many related applications. For example, authors in [22] used C4.5 to generate rules, which were later used to build an optimal fuzzy classifier. They applied the multi-objective evolutionary algorithm $NSGA-II$ to minimize the size and number of fuzzy rules in sample data. In their work at 2009 they combined the benefits of WM algorithm and DT to propose a new decision-tree based initialization method for regression problems. At 2010 they optimize their system by using dynamic constraints to tune fuzzy membership functions and genetic operators, which modify antecedents of rules. This new method led to better accuracy and made their algorithm suitable for high dimensional regression problems which can be applied to classification problems too [23]. These examples are just some of many cases where categorization or classification are required, and decision trees have been used with great success.

1.1. Commonly Used Splitting Criteria

An overview of DTs and related works were presented in the previous section. Different criteria have been proposed in order to effectively split each DT node. As improving this criterion is the main innovation of this paper, we summarize here two commonly used standard splitting criteria, which are used in a variety of classification algorithms.

1.1.1 Gain Ratio

ID3 algorithm [3] uses entropy based information gain as the split measure to construct a DT as a classifier. The purpose of the algorithm is to reduce the height of the tree by decreasing the irregularity at each node. At first, the irregularity is computed for all the features as the following:

$$I = -\sum_{c} p(c)\log_2 p(c) \tag{1}$$

where $p(c)$ is the proportion of the data that belong to class c. Then, the feature which has the most gain ratio will be selected as the root of the tree.

$$Gain(A) = I - I_{res}(A) \tag{2}$$

where I_{res} is the remaining irregularity at all the classes when feature A were used and compute as the following

$$I_{res} = -\sum_{v} p(v)\sum_{c} p(c \mid v)\log_2 p(c \mid v) \tag{3}$$

This algorithm is just working on numerical data and can't handle with uncertainty or noisy data. C4.5, is the improvement of this algorithm and can also work with continues data. If the data is continues, first the data is sorted based on the desired attribute and then, the gain ratio is computed for every case that the sorted data can be classified. If the data is numerical the algorithm is the same as the ID3 algorithm. Pruning the tree is the most superior attribute of the C4.5 algorithm that prevents over fitting and removes noisy or missing data.

1.1.2 Gini Index

Gini Index [2] is used as the split measure in SLIQ algorithm developed by the Quest team at IBM [24]. The Gini Index is defined as:

$$I_{gini} = 1 - \sum_{j} p(c_j)^2 \qquad (4)$$

where, $p(c)$ is the relative frequency of cases that belong to class c_j. Then the information gain is computed as:

$$G = I - \sum \frac{c_j}{c} I_c \qquad (5)$$

The feature that maximize G is chosen as the splitting feature.

In this paper, a new splitting criterion to build decision trees is proposed. Splitting criterion in a tree is specified by selecting the best splitting variable and its threshold for the further split. For this purpose, we take use of the correlation concept to find the best splitting variable and Support Vector Machine (SVM) to specify its threshold. The idea of the correlation concept is taken from regression analysis. SVMs are a set of learning methods used for classification and regression; which are applied effectively in the new decision tree algorithm. These two methods will be explained in section 2 as the background. At each node in the DT the best feature which has the most absolute correlation with the class labels will be selected so that a new branch of the tree will be expanded based on this feature and its specified threshold, that is obtained from SVM. This new decision tree is named Correlation based Multi Branch Support Vector (C-MBSV) and will be presented in section 3. During the tree generation, a series of empty leaves will be created, that can take neither of the class labels and will be called as unclassified regions. Causes of generation and strategies to reduce or eliminate these regions, along with the experimental results of the proposed method in comparison with other famous decision tree algorithms are discussed in section 4. As will be shown, the advantages of the new algorithm are having shorter height and making use of fewer features. We should note that, to the best of our knowledge, no decision tree can be introduced as the optimized data mining tool and deciding which decision tree algorithm is suitable to apply on a given problem is very difficult.

2. Background

As the proposed DT uses the combination of the correlation concept to find the best splitting variable and Support Vector Machine (SVM) to specify its threshold as the splitting criterion, here we introduce our inspired methods for feature selection and threshold specification in brief.

2.1 Correlation Based Feature Selection

According to literatures, correlation analysis is only used in determining whether a linear relationship exists between two variables. To quantify the strength of the relationship, calculation of the correlation coefficient is necessary. Furthermore, in regression analysis, the nature of the relationship itself between the dependent (feature) variables and the independent variable (label) is interested. The analysis is exploiting the relationship between variables and consists of two phases: selection and fitting an appropriate model by the model of least squares. This model can estimate the expected response for a given value of the independent variable. Selection allows the construction of an optimal regression equation along with an investigation into specific predictor variables. The aim of selection is to reduce the set of predictor variables (features) to the necessary number that accounts for nearly as much of the variance that is accounted by the total set. In essence, selection helps to determine the level of importance of each predictor variable.

Usage of different selection procedures, such as forward selection, backward elimination, stepwise selection, and blockwise selection yields to different regression equation. Classical model-selection method known as Forward selection [25] and its enhanced versions like Forward Stagewise and LARS [26], are all greedy forward stepwise procedures whose forward progress is determined by compromising among the currently most correlated covariates. Although, each of the aforementioned algorithms have their own distinct steps and computation formulas, at the first step, they act the same. By giving a collection of variables, they begin with an empty equation and select the one having the largest absolute correlation with the dependent variable at the first step. Variables of greater theoretical importance are entered first.

Getting the idea, at each node in DT we choose the feature which it's vector values has the most absolute correlation with the vector values of the class labels.

2.2 Support Vector Machines

Support Vector Machines (SVMs) [27] are very popular and powerful in learning systems because of the utilization of a kernel machine in linearization, providing good generalization properties, their ability to classify input patterns with minimized structural misclassification risk and finding the optimal separating hyper-plane between two classes in the feature space. Non-separable case of SVM is in the following from

$$\begin{aligned} f &= \min \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ s.t \ \ & y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, \ \ \xi_i \geq 0, \ \ i = 1,\ldots,n \end{aligned} \tag{6}$$

where $S = \{(x_i, y_i)\}_{i=1}^{n}$ is a set of n training samples, $x_i \in R^m$ is an m-dimensional sample in the input space, and $y_i \in \{-1,1\}$ is the class label of x_i. Inputs to the SVM system are the training data and the constant C. The system will calculate proper slack variables $\xi_i, i = 1,\ldots,n$, and will determine the separating hyper-plane. ξ_i is the training error corresponding to data sample x_i. SVM finds the optimal separating hyper-plane with minimal classification errors. Let w_0 and b_0 denote the optimum values of the weight vector and bias respectively. The hyper-plane can be represented as $w_0^T x + b_0 = 0$, that $w_0 = [w_{01}, w_{02}, \ldots, w_{0m}]^T$ and $x = [x_1, x_2, \ldots, x_m]^T$, w_0 is the normal vector of the hyper-plane, and b_0 is the bias that is a scalar.

The idea of maximizing the margin between classes in SVM is used to find the best threshold at each node in the proposed DT.

3. The Proposed Method

Recently, methods of rule extraction from decision trees have been of interest to some authors [28], and [29] and [23]. Nevertheless, DTs have their own restrictions. Finding best feature at each node in DTs needs calculation of Entropy or Gini index for all features, which is time consuming. And also, as shown in Figure 1 the determined threshold is not convenient; the threshold based on Information Gain has many disparities in comparison with the desired one. In the continuous feature space, we should compute Gain for each value to obtain the threshold.

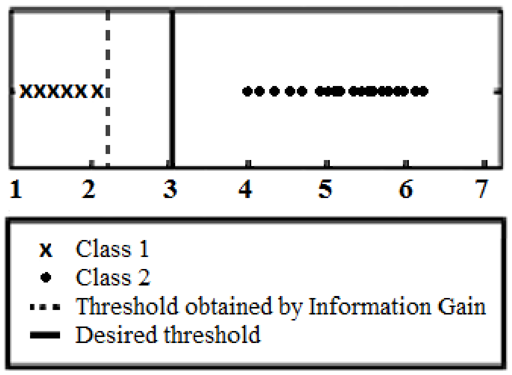


Figure 1. Disparity of Threshold Obtained using Information Gain with the Desired One

In this paper, we use the concept of correlation in the family of classic forward selection regression methods and choose the feature having the most absolute correlation with class labels as the best feature at each node. Indeed, instead of calculating the Entropy or Gini index for all the features and finally computing the Gain, the computations of correlation runs just one time on all the features. To solve the problem of finding the threshold, the idea of maximizing the margin between classes has been used. One of the methods that makes use of this idea in data classification is Support Vector Machine (SVM). SVM classifies samples into two classes such that the margin $2/\|w\|$ is maximized, or in the other word minimizing $\|w\|/2$ as in (6). Using of this idea, make the threshold more general and close it to the ideal one.

Maximizing the margins between classes is used to determine the Split Threshold on the feature that its selection was based on correlation. Selection of this cost function accedes the Split Threshold to the desired one. In contrary with C4.5 and CART decision tree algorithms, dealing with numerical data the branching factor at the proposed method can have more than one threshold. Variable branch factor, improvement in determining the split threshold and more efficiency in dealing with noisy data are some privileges of the proposed method. The proposed method in this paper is called; Correlation based Multi Branch Support Vector (C-MBSV).

Chapter 2 discussed SVM and Feature Selection based On Correlation. In this chapter, by combining C4.5, SVM and correlation concept, we proposed a simpler and more efficient method for data classification. After the generation of the DT, the rules will be extracted from the tree. The structure of the proposed classifier is shown in Figure 2.

C-MBSV is applied only on numerical and continuous feature values. Non numerical values must first change to numerical values. E.g. a discrete feature with three values of Morning, Midday and Night can be represented by values of 1, 2 and 3 in sequence. In the following, we explain different parts of C-MBSV method in details.

3.1 DT Generation Using Correlation Based Feature Selection and SVM Threshold Function

The procedure of tree branching at each node of the DT is started by determining the best splitting feature which has the most absolute correlation with class labels. After choosing the best feature, data samples of that node will be partitioned by using the splitting threshold/s obtained from SVM. If one of the end conditions is satisfied at this node, the algorithm will be terminated. Otherwise, the algorithm will be called recursively for data samples of this node. This procedure is demonstrated in Figure 3.

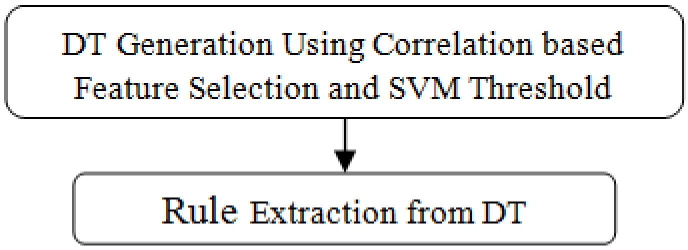

DT Generation Using Correlation based Feature Selection and SVM Threshold
Rule Extraction from DT

Figure 2. Structure of the Proposed Method

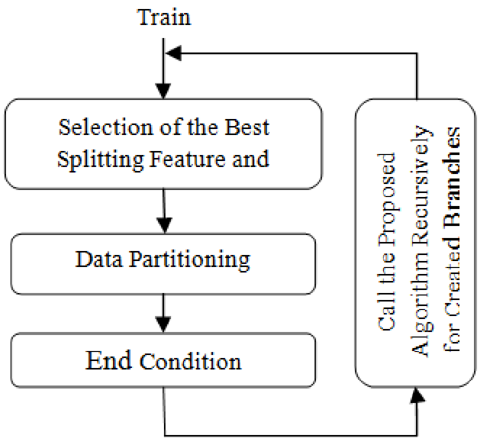

Train
Selection of the Best Splitting Feature and
Data Partitioning
End Condition
Call the Proposed Algorithm Recursively for Created Branches

Figure 3. Procedure of DT Generation using Feature Selection based On Correlation and SVM Threshold Function

1) Selection of the Best Splitting Feature and Computing its Splitting Threshold/s

Getting the idea from greedy forward stepwise procedures whose forward progress is determined by compromise among the currently most correlated covariate, at each node in DT, we compute the inner dot product of every feature vector values with the class labels vector values only on unclassified samples that have been travelled to this branch of the tree at the last call. According to

$$j = \arg\max_{i} \; |r_i| = \{|< X_i, Y >|\} \tag{7}$$

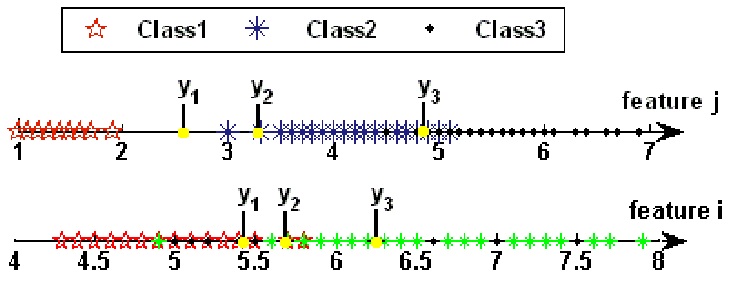
where X_i is feature vector i and $< X_i, Y >$ is the inner dot product among X_i and Y, feature X_j which has the most absolute correlation r_j with class labels Y is determined as the dominant one. Then, the dominant feature's splitting threshold/s obtained from

$$Y^j = \begin{bmatrix} y_{11}^j & y_{12}^j & \cdots & y_{1n}^j \\ y_{21}^j & y_{22}^j & \cdots & y_{2n}^j \\ \vdots & \vdots & \vdots & \vdots \\ y_{n1}^j & y_{n2}^j & \cdots & y_{nn}^j \end{bmatrix}, \quad j = 1, \ldots, m \tag{8}$$

where y_{ab}^j represents the splitting criterion between class a and b at the j'th feature, which obtained from

$$(w^t)_{ab} y^j{}_{ab} + b_{ab} = 0, \qquad j = 1, \ldots, m, \qquad a, b = 1, \ldots, n \tag{9}$$

in which $(w^t)_{ab}$ and b_{ab} explain the splitting hyper-plane between class a and b at the j'th feature in SVM. Y that is computed for the dominant feature is a symmetric matrix with zero diagonal elements. In Figure 4 feature j



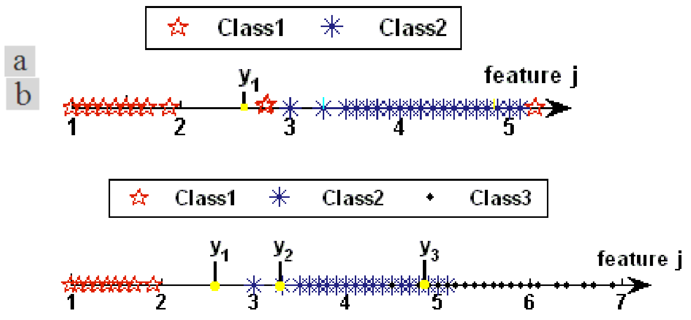
$$Y^j = \begin{bmatrix} \circ & 2.55 & 3.31 \\ 2.55 & \circ & 4.86 \\ 3.31 & 4.86 & \circ \end{bmatrix}$$

Figure 4. Selection of the i'th Feature Causes Minimum Irregularity within DT

which has the most absolute correlation defeat feature i, matrix Y^j is also shown for j'th feature. Maximum number of splitting thresholds at each node in the tree is calculated as follows:

$$b = \frac{n \times n - n}{2} \tag{10}$$

where n and b are the number of classes and the maximum number of splitting thresholds at each node respectively. Maximum branching factor at each node can be equal to $b+1$. For example, if the data in a node have three classes then $b = \frac{3 \times 3 - 3}{2} = 3$, and if the data in a node have two classes then b is 2. These partitions are shown in Figure 5 along with the splitting threshold/s. In Figure 5(a) the data contains two distinct classes and therefore there exists just one splitting threshold, but in Figure 5(b) since the data contains three distinct classes, three splitting thresholds obtain.



$$Y^j = \begin{bmatrix} \circ & 2.61 \\ 2.61 & \circ \end{bmatrix}$$

$$Y^j = \begin{bmatrix} \circ & 2.55 & 3.31 \\ 2.55 & \circ & 4.86 \\ 3.31 & 4.86 & \circ \end{bmatrix}$$

Figure 5. The Classified Data in a Node on j'th Feature. (a) The data have two classes; therefore, there is one splitting threshold. (b) The data have three classes; therefore, there are three splitting thresholds.

Next part describes data partitioning based on the dominant feature X_j and its obtained splitting threshold Y^j.

2) Data Partitioning

In this part, the procedure of tree branching and its generation in a recursively manner is explained. Three major steps involved in branching procedure are **dividing data, pruning tree and end condition**. Since depending on data different cases may appear during this procedure, and the three steps are highly dependent to each other, they are explained together.

After choosing the best feature, training data samples of that node will be partitioned into $b+1$ branches by using the obtained splitting threshold/s, and the data partitioning conditions would be written upon that node. For example Figure 6 illustrates data partitioning of i'th feature of Figure 4. After dividing the data to the branches, one of the following two cases may occur.

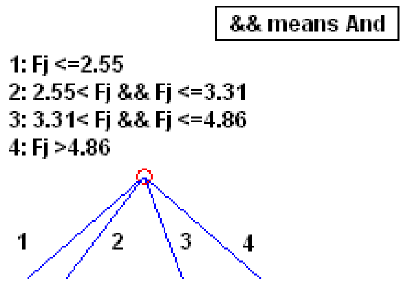


Figure 6. Labeling Nodes in C-MBSV Algorithm

i) All branches have data

In this case, as shown in Figure 7(a), all branches have data and one of the two conditions below would occur for each node of this sub tree:

a) If all the data belongs to the i'th class in a node then it takes the i'th class as the leaf label as shown in Figure 7(b) and the algorithm is terminated for this branch.

b) If all the data doesn't belong to a class in a node then the C-MBSV algorithm is called recursively for this node, as shown in Figure 3.

ii) Some branches have data

In this case, at least one of the tree branches does not have any data and for each node of this sub tree one of the two following conditions may happen:

a) If only one node has data and all the data belongs to the i'th class in a node then it takes the i'th class as the leaf label (Figure 7(b)) and the algorithm for this branch is terminated. Otherwise, if all the data does not belong to one distinct class, then the node takes the label of the most frequently used class in the data node. As shown in Figure 8(a) there exist two data samples with different labels in the marked leaf which are considered as errors. The cause of this error is that data on each leaf is projected based on one feature, and so they are placed on each other. In the other words, data are not linearly separable based on one feature.
At this stage, because only one leaf has data and other leaves are empty, as shown in Figure 8(b), the tree is pruned and the algorithm for this branch is terminated.

b) At least two leaves have data, and the rest are empty. In this case, the parent node includes some empty leaves that are called Unclassified Region in this paper and are marked in Figure 9. In the next section, the causes of generation of these regions and methods of reducing or eliminating them are discussed. It should be considered that if all data does not belong to one class in a leaf, then the C-MBSV algorithm is **called recursively for that data**, as shown in Figure 3. Otherwise, as previous all the data belongs to the i'th class and takes the i'th class label as the leaf label.

3.2 Rule Extraction from Decision Tree

In the last chapter, we describe the procedure of the DT construction in details. Rule extraction from DT will be discussed in the following. We use depth first traversal to extract rules from the DT, in which for every path we start from the root and continue parsing until we reach a leaf node and then convert that to a rule. The following rule is extracted from the path that is marked in Figure 10.

$$IF \quad F3 \le 9.90 \quad and \quad F3 \le 8.97 \quad and \quad F1 \le 53.78 \quad Then \quad y \quad is \quad Class2 \tag{11}$$

Test data samples will be classified by the extracted rules.

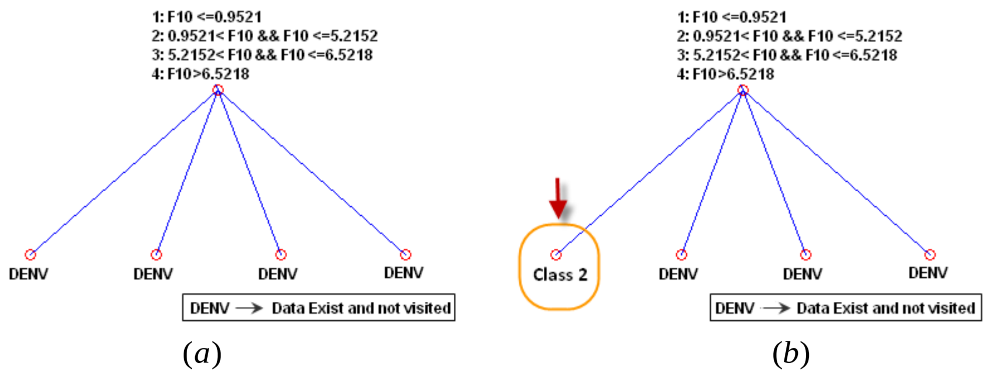

1: F10 <=0.9521
2: 0.9521< F10 && F10 <=5.2152
3: 5.2152< F10 && F10 <=6.5218
4: F10>6.5218
DENV
DENV
DENV
DENV
DENV → Data Exist and not visited
(a)
1: F10 <=0.9521
2: 0.9521< F10 && F10 <=5.2152
3: 5.2152< F10 && F10 <=6.5218
4: F10>6.5218
Class 2
DENV
DENV
DENV
DENV → Data Exist and not visited
(b)

Figure 7. Data are Divided into Four Branches. (a) All branches have data. (b) The left branch data belongs to the 2nd class.

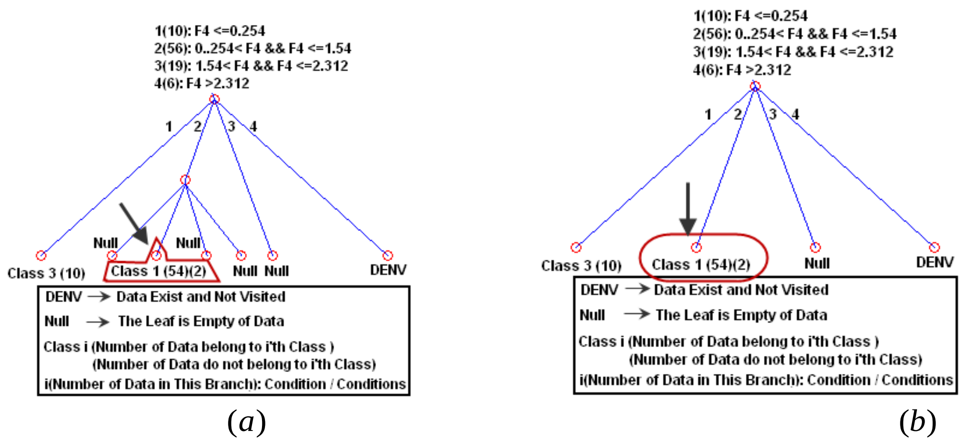

1(10): F4 <=0.254
2(56): 0..254< F4 && F4 <=1.54
3(19): 1.54< F4 && F4 <=2.312
4(6): F4 >2.312
1
2
3
4
Null
Null
Class 3 (10)
Class 1 (54)(2)
Null
Null
DENV
DENV → Data Exist and Not Visited
Null → The Leaf is Empty of Data
Class i (Number of Data belong to i'th Class)
(Number of Data do not belong to i'th Class)
i(Number of Data in This Branch): Condition / Conditions
(a)
1(10): F4 <=0.254
2(56): 0..254< F4 && F4 <=1.54
3(19): 1.54< F4 && F4 <=2.312
4(6): F4 >2.312
1
2
3
4
Class 3 (10)
Class 1 (54)(2)
Null
DENV
DENV → Data Exist and Not Visited
Null → The Leaf is Empty of Data
Class i (Number of Data belong to i'th Class)
(Number of Data do not belong to i'th Class)
i(Number of Data in This Branch): Condition / Conditions
(b)

Figure 8. Some branches have data. Fifty-four data samples in the marked leaf belong to the 1st class and two data samples belong to other classes. In this case, the leaf has two errors. (a) Tree before pruning. (b) Tree after pruning.

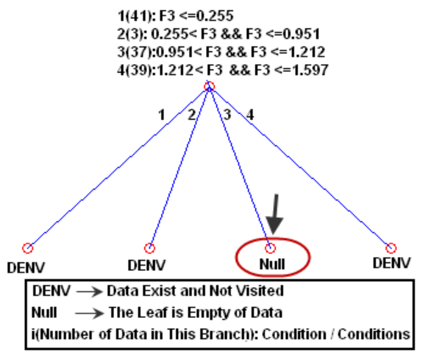

1(41): F3 <=0.255
2(3): 0.255< F3 && F3 <=0.951
3(37):0.951< F3 && F3 <=1.212
4(39):1.212< F3 && F3 <=1.597
1
2
3
4
DENV
DENV
Null
DENV
DENV → Data Exist and Not Visited
Null → The Leaf is Empty of Data
i(Number of Data in This Branch): Condition / Conditions

Figure 9. Some Branches have Data and the Marked Leaves are called Unclassified Region

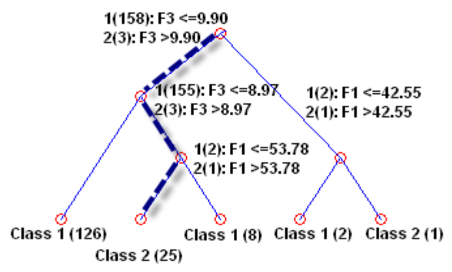


Figure 10. Rule Extraction from DT with Depth First Traversal

4. Discussion

We separate our discussion in two parts. First we discus Unclassified Regions that were introduced in the previous sections. Cause of generation and methods to reduce or eliminate these regions will be explained. Then, we explain our experimental results.

A. Unclassified Regions

The previous section showed that a series of leaves created during the tree generation were named Unclassified Region. In this part, causes of generating and strategies of reducing or eliminating these regions will be discussed. The main problem with these regions is that they cannot make a decision for the test data samples that were placed in them, but this is also an advantage of these regions because they will not make a wrong decision for the data. In this case human knowledge is required for labeling the data.

Why unclassified regions are generated?

There are two main reasons for generating these regions:

a) Because of using one feature at each node for categorizing, data projection exists in this algorithm. The result of data projection is one dimensional and the data may have overlaps based on one feature.
b) Due to inappropriate distribution of training data over feature space, adequate training data is unavailable. The solution is adding training data, but since all possible cases of training pattern may not be available in the reality this solution is impossible or time consuming. This problem is solved by increasing training data while keeping the generality of the data.

Solutions to reduce or eliminate the generation of unclassified regions:

There are several solutions for eliminating or reducing unclassified regions. One simple method for labeling the test data in the Unclassified Region is using another classification algorithm like KNN algorithm [30] or even getting the data labels from a user. The fundamental solution to reduce or eliminate unclassified regions is to solve the data projection problem. For this purpose, more features at each node can be selected for categorizing the data. With this choice, since the data is projected to more than one dimension the unclassified regions are eliminated or reduced. If the number of selected features at each node is equal to the total number of train data features, then the C-MBSV algorithm is similar to the multiclass SVM and the tree height is equal to one. If the number of selected features at each node is equal to one, then the C-MBSV algorithm is similar to the C4.5, C5 or CART. In section 5, KNN algorithm which has highly improved decision making performance is used to solve the problem of unclassified regions.

B. Experimental Results

Our method is validated using 8 datasets, all downloaded from *UCI Machine Learning Repository* [31], involving different number of feature variables and one output label (see Table 1). For all datasets, 15-fold cross validation was used. The original data is randomly partitioned into folds for all the datasets, and take constant during running of all different DT algorithms. Our method is compared against C4.5 and CART. C4.5 algorithm was implemented in Linux [32]; CART and C-MBSV algorithms were implemented in Windows XP using MATLAB 7.10 running on a PC with an Intel Core2 Duo CPU processor 3GHz and

1 GB RAM. The comparison results of applying C-MBSV, C4.5 and CART on this data set are shown in Tables 2 to 6. During the results, imbalance and imbalance/stopping condition/s were considered. Imbalance problem happens in a data set when the number of data samples of a class is much larger than the other one. In this condition, SVM usually computes incorrect hyper plane, which settles down closer to the class with smaller data samples or even crosses over it. To handle this problem, consider a case where class z_1 and z_2 have n_1 and n_2 samples respectively. Parameter C in equation (6) for all data samples of class z_1 will be considered as $C_1 = \frac{n_2}{n_1 + n_2}$ and $C_2 = \frac{n_1}{n_1 + n_2}$ for class z_2. This penalty setting moves the hyper plane towards the correct place. Stopping condition is another consideration, which happens when in a node executing the algorithm alternatively, made no new results or in other words, it falls in a loop. So, in this condition we stop running at the specified node and leave it without any more processing as a leaf node.

Table 2 shows the error rates. The training error rate is worse in some of the data sets, but we should say that the CART and C4.5 algorithms that we use, make use of pruning and improvement algorithms. The results on test error rate show that our DT usually has better or the same test error rate in comparison with CART and C4.5. In Chapter 4 we also discuss about unclassified regions which we use KNN to handle them. It's obvious that when there exists no test data in unclassified regions or no such regions exists the results are the same as the case that not using KNN. Height of the decision tree is one of the most comparative tools that can be used to compare different DTs. Table 3 compares the average height of our DT against two other algorithms. It's obvious our DT is superior in height, which causes lower time of induction for future data. At last, Table IV contains the number of needless features that does not enter our tree in comparison to C4.5 DT. This number is another advantage of our algorithm that leads to just most related and fewer features (while having the less or more test error rate). This advantage can be used as feature reduction or preprocessing phase of applications that need to work with fewer features. Consequently, by the obtained results it seems that the proposed algorithm outperforms in terms of smaller height and fewer number of useful features in the final decision tree.

Table 1. The Properties of the Data Sets

Data	Num. of Attributes	Num. of Instances	Classes
Balance-Scale	4	625	3
Glass	10	214	6
Haberman	3	306	2
Ionosphere	34	351	2
Iris	4	150	3
Pima	8	768	2
Wine	13	178	3
Zoo	17	101	7

Table 2. Error Rate

	Training Error Rate			Test Error Rate			
Data	C-MBSV	CART	C4.5	C-MBSV	C-MBSV(KNN)	CART	C4.5
Balance_Scals	30.72	24.37	25.69	39.19	39.19	33.78	35.06
Glass	5.24	11.85	7.20	32.29	**31.81**	31.97	32.79
Haberman	17.58	13.82	23.62	25.79	25.79	33.60	30.30
Ionosphere	8.61	1.59	1.36	11.71	11.71	11.97	9.45
Iris	1.00	2.33	1.96	3.33	3.33	6.00	4.00
Pima	24.83	7.20	15.73	25.12	25.12	29.04	25.27
Wine	0.04	1.68	1.08	6.77	6.77	13.99	7.82
Zoo	0.00	3.89	0.00	0.00	0.00	5.71	2.07

Table 3. Average Height of the Tree

Data	C-MBSV	CART	C4.5
Balance_Scals	6.67	8.00	8.00
Glass	6.53	10.33	9.80
Haberman	4.73	13.13	5.00
Ionosphere	4.00	9.40	9.00
Iris	2.07	4.00	3.87
Pima	2.20	12.87	9.40
Wine	3.67	4.27	3.07
Zoo	1.00	5.00	5.00

Table 4. Number of Needless Features

Data	C-MBSV (QP)	C4.5
Balance_Scals	0.00	0.00
Glass	1.00	0.13
Haberman	0.34	1.33
Ionosphere	26.80	23.20
Iris	1.94	2.00
Pima	5.94	1.20
Wine	5.80	9.40
Zoo	16	11.80

6. Conclusion

In this paper, a new splitting criterion to build decision trees is proposed, and named as Correlation based Multi Branch Support Vector(C-MBSV). In the proposed DT, the correlation concept is used to find the best splitting variable, and then it's relevant thresholds is specified by using Support Vector Machine (SVM). Since the algorithm can return multi thresholds, the tree is divided to one or more branches at each node. If the instances in a sub-node belong to one class, then the sub-node is regarded as a leaf node, else we continue to split the sub-node until all leaf nodes are generated. Some leaves that do not lead to any decisions were also generated and named as *Unclassified Regions*. The obtained results seem to outperform in terms of shorter height and fewer number of useful features in the final proposed decision tree against compared algorithms.

7. Future Work

In future, we want to enhance our tree by using more than one variable as the splitting feature at each node. Furthermore, we want to use the proposed method on classification applications in the field of Image Processing.

References

[1] Y. S. Kim, "Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size", Expert Systems with Applications, vol. 34, **(2008)**, pp. 1227-1234.

[2] L. Breiman, J. Friedman, C. J. Stone and R. A. Olshen, "Classification and Regression Trees. Monterey", CA: Wadsworth and Brooks, **(1984)**.

[3] J. R. Quinlan, "Induction of Decision Trees", Machine Learning 1, **(1986)**, pp. 81-106.

[4] J. R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann, San Mateo, California, **(1993)**.

[5] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data", Applied Statistics, vol. 29, **(1980)**, pp. 119-127.

[6] C. E. Brodley and P. E. Utgoff, "Multivariate decision trees", Machine Learning, vol. 19, **(1995)**, pp. 45-77.

[7] W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees", Statistica Sinica vol. 7, **(1997)**, pp. 815-840.

[8] W. -Y. Loh and N. Vanichsetakul, "Tree-structured classification via generalized discriminant analysis (with discussion)", Journal of the American Statistical Association, vol. 83, **(1988)**, pp. 715-728.

[9] W. Muller and F. Wysotzki, "Automatic construction of decision trees for classification", Annals of Operations Research, vol. 52, **(1994)**, pp. 231-247.

[10] R. Rastogi and K. Shim, "PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning", Data Mining and Knowledge Discovery, vol. 4, **(2000)**, pp. 315-344.

[11] T. -S. Lim, W. -Y. Loh and Y. -S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms", Machine Learning, vol. 40, **(2000)**, pp. 203-228.

[12] M. Maruf Hossain, M. Rafiul Hassan and B. J., "ROC-tree: A Novel Decision Tree Induction Algorithm Based on Receiver Operating Characteristics to Classify Gene Expression Data", In the 8th SIAM International Conference on Data Mining (SDM08), **(2008)** April.

[13] S. Ruggieri, "Efficient C4.5," IEEE transaction on knowledge and data engineering, vol. 14, **(2002)** March/April.

[14] Z. Qin and J. Lawry, "Decision tree learning with fuzzy labels", Information Sciences, vol. 172, **(2005)**, pp. 91-129.

[15] M. J. Aitkenhead, "A co-evolving decision tree classification method", Expert Systems with Applications, vol. 34, **(2008)**, pp. 18-25.

[16] C. Vens, J. Struyf, L. Schietgat, S. Džeroski and H. Blockeel, "Decision trees for hierarchical multi-label classification", Machine Learning, vol. 73, **(2008)**, pp. 185-214.

[17] Y. L. Chen, H. W. Hu and K. Tang, "Constructing a decision tree from data with hierarchical class labels", Expert Systems with Applications, vol. 36, **(2009)**, pp. 4838-4847.

[18] S. E. Hansen and R. Olsson, "Improving decision tree pruning through automatic programming," in Proceedings of the Norwegian Conference on Informatics (NIK-2007), pp. 31-40, (2007).

[19] D. Sieling, "Minimization of decision trees is hard to approximate", Journal of Computer and System Sciences, vol. 74, **(2008)**, pp. 394-403.

[20] M. K. Markey, G. D. Tourassi and F. C. E., "Decision tree classification of proteins identified by mass spectrometry of blood serum samples from people with and without lung cancer", Proteomics, **(2003)**, pp. 1678-1679.

[21] B. Zmazek, L. Todorovski, S. Dzˇeroski, J. Vaupoticˇ and I. Kobal, "Application of decision trees to the analysis of soil radon data for earthquake prediction", Applied Radiation and Isotopes, vol. 58, **(2003)**, pp. 697-706.
[22] P. Pulkkinen and H. Koivisto, "Fuzzy classifier identification using decision tree and multiobjective evolutionary algorithms", International Journal of Approximate Reasoning, vol. 48, **(2008)**, pp. 526-543.
[23] P. Pulkkinen and H. Koivisto, "A Dynamically Constrained Multiobjective Genetic Fuzzy System for Regression Problems", IEEE Transactions on Fuzzy Systems, vol. 18, **(2010)** February.
[24] M. Mehta, R. Agrawal and J. Riassnen, "SLIQ: a fast scalable classifier for data mining", in in: Extending Database Technology Avignon, France, **(1996)**.
[25] S. Weisberg, Applied Linear Regression. New York: Wiley, **(1980)**.
[26] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," The Annals of Statistics, vol. 32, **(2004)**, pp. 407-499.
[27] V. N. Vapnik, The Nature of Statistical Learning Theory. New York: Springer, **(1995)**.
[28] N. Barakat and J. Diederich, "Eclectic Rule-Extraction from Support Vector Machines", International Journal of Computational Intelligence, vol. 2, **(2005)**, pp. 59-62.
[29] M. Farquad, V. Ravi and R. S. Bapi, "Rule extraction using Support Vector Machine based hybrid classifier", in IEEE Region 10 Conference TENCON 2008 Masab Tank, Hyderabad, **(2008)**.
[30] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification", IEEE Transactions on Information Theory, vol. 13, **(1967)**, pp. 21-27.
[31] UCI, "http://archive.ics.uci.edu/ml/datasets.html", UCI Repository of machine learning databases and domain theories, **(2010)** May.
[32] R. Quinlan, "Site address: <http://www.rulequest.com/Personal/>", Ross Quinlan's Home Page, **(2010)** May.

Author

Hadi Sadoghi Yazdi is currently an Associate Professor of Computer Science and Engineering at Ferdowsi University of Mashhad (FUM). He received his B.S. degree in Electrical Engineering from FUM in 1994, and received his M.S. and Ph.D. degrees in Electrical Engineering from Tarbiat Modares University in 1996 and 2005, respectively. Dr. Sadoghi Yazdi has received several awards including Outstanding Faculty Award and Best System Design Award in 2007. His research interests are in the areas of Pattern Recognition, Machine Learning, Machine Vision, Signal Processing, Data Mining and Optimization.