Cent. Eur. J. Comp. Sci. • 1(2) • 2011 • 52-67
DOI: 10.2478/s13537-011-0017-x

VERSITA

Central European Journal of **Computer Science**

Correlation based splitting criterionin multi branch decision tree

Research Article

Nima Salehi-Moghaddami*, Hadi Sadoghi Yazdi†, Hanieh Poostchi‡

Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

Received 01 Feb 2011; accepted 07 Jun 2011

Abstract: One of the most commonly used predictive models in classification is the decision tree (DT). The task of a DT is to map observations to target values. In the DT, each branch represents a rule. A rule's consequent is the leaf of the branch and its antecedent is the conjunction of the features. Most applied algorithms in this field use the concept of Information Entropy and Gini Index as the splitting criterion when building a tree. In this paper, a new splitting criterion to build DTs is proposed. A splitting criterion specifies the tree's best splitting variable as well as the variable's threshold for further splitting. Using 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 at each node. Then, the idea of maximizing the margin between classes in a support vector machine (SVM) is used to find the best classification threshold on the selected variable. This procedure will execute recursively at each node, until reaching the leaf nodes. The final decision tree has a shorter height than previous methods, which effectively reduces useless variables and the time needed for classification of future data. Unclassified regions are also generated under the proposed method, which can be interpreted as an advantage or disadvantage. The simulation results demonstrate an improvement in the generated decision tree compared to previous methods.

Keywords: decision tree • splitting criterion • support vector machine • correlation • unclassified region

1. Introduction

A decision tree (DT) is one of the most important knowledge representation approaches for building top-down models in simplified subspaces. They attempt to perform this task 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. More specifically, a splitting criterion in used in the construction of a tree to select the best splitting variable and its threshold for splitting. In classification, the DT model is used to increase system prediction to classify oncoming data. Despite strong competitors like Linear Regression and Artificial Neural Networks (ANNs), DTs have

* *E-mail: nima.salehi@stu-mail.um.ac.ir*
† *E-mail: h-sadoghi@um.ac.ir (Corresponding author)*
‡ *E-mail: hanieh.poostchi@stu-mail.um.ac.ir*

several advantages; they are simple to understand and interpret, inexpensive to compute and capable of dealing with noisy data [1].
The CART algorithm [2] selects splits by providing the Gini and 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 splitting criterion. It does not apply any pruning procedure and cannot handle numeric values and missing data. In the C4.5 algorithm [4], the evolution of ID3, the splitting process ceases when the number of instances to be split is below a certain threshold. C4.5 uses error-based pruning and can handle numeric attributes. It can also induce from a training set with missing values. Due to the extensive number of application domains, numerous researchers focus on improving these principal algorithms, or even propose new splitting criteria with exclusive characteristics.
Numerous old and new algorithms work very differently when compared with the principal DT algorithms already introduced. The CHAID algorithm [5], as an old one, applies statistic procedures to generate a novel decision tree. It uses the statistical p—value concept to perform splitting effectively. Another algorithm, the LMDT [6] algorithm, 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. It uses ANOVA F-test, 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; an earlier version of QUEST, called the FACT algorithm, can be found in [8]. A profound comparison of these classic algorithms and some 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 classification of some sample databases than famous DT algorithms like C4.5 and C5. The ROC-tree algorithm, uses the area under the Receiver Operating Characteristics (ROC) curve to help determine decision tree characteristics, such as node selection and stopping criteria.

1.1. Improvements of principal DT algorithms

Many researchers have focused on the improvement of the principal DT algorithms. Efficient C4.5 [13] is an improvement over C4.5 in which three strategies are suggested to choose the best information gain. The first and second strategies, use quick and counting sort, respectively, and the third strategy 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 trees are promoted by inserting linguistic terms rather than 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 decision tree while updating data set size and DT depth. Another paper presented several approaches to the induction of decision trees for Hierarchical Multi-label Classification (HMC), as well as an empirical study of 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 in a hierarchical structure.
Some researchers have focused on improving classic DT characteristics; like minimization of DTs, improving the DT pruning procedure or changing splitting measures. DT 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 be approximated.

1.2. Applications of DTs

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 by [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] utilize 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 in 2009 they combined the benefits of the WM algorithm with those of DTs to propose a new DT based initialization method

for regression problems. In 2010 they optimized 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 [23]. These examples are just some of many cases where categorization or classification is required and decision trees have been used with great success.

1.3. Commonly used splitting criteria

An overview of DTs and related works are presented in the above. 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 several commonly used standard splitting criterions, which are used in a variety of classification algorithms.

1.3.1. Information Gain and Gain Ratio

The ID3 algorithm [3] uses entropy based information gain as the split measure to construct a DT. The purpose of this 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 data that belong to class c. Then, the residual information is computed for all the features as below:

$$I_{res}(V) = -\sum_{v} p(v) \sum_{c} p(c|v) \log_2 p(c|v), \tag{2}$$

where v is the distinct values of the feature V. Feature A that maximizes the Gain will be selected as the root of the tree by (3).

$$Gain(A) = I - I_{res}(A). \tag{3}$$

This algorithm only works on categorical-value features and cannot handle uncertainty or noisy data. C4.5 is the improvement of this algorithm that can also work with continuous-value features, such as percentage values. If the data is continuous, first, the data is sorted based on the desired feature and then, the gain 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 of the tree, which prevents over-fitting and removes noisy or missing data, is the most superior attribute of the C4.5 algorithm. A problem with this criterion is that if a given feature has a large number of samples belonging to some class against, then this feature will have the highest Gain when compared with the other features [24]. Quinlan resolved this problem by the following, which is known as Gain Ratio:

$$GainRatio(A) = \frac{Gain(A)}{I(A)}, \tag{4}$$

where, $I(A)$ is the potential information obtained by (2) when feature A is used for categorization.

1.3.2. Gini Index

Gini Index [2] is used as the split measure in the SLIQ algorithm developed by the Quest team at IBM [25]. The Gini Index is defined as:

$$I_{gini} = 1 - \sum_{j} p(c_j)^2, \tag{5}$$

where, $p(c_j)$ is the relative frequency of cases belong to class c_j. Then the information gain is computed as below for all the features:

$$G = I_{gini} - I_{gini}(v), \tag{6}$$

where, $I_{gini}(v) = \sum_{j} \frac{c_j}{c} I_v$ and I_v are computed by (5) when feature v is used for categorization. The feature that maximizes G is chosen as the splitting feature.

1.3.3. Chi-Square statistic

A different decision tree induction method evaluates splitting feature candidates by measuring the association between variables in a contingency table, at each split of the tree node. The chi-square (χ^2) statistic compares the expectation values, i.e. if there were no association between the variables, with the observed values by computing the following equation for each feature i,

$$\chi^2 = \sum_{ij} \frac{(x_{ij} - E_{ij})^2}{E_{ij}}, \tag{7}$$

where x_{ij} and E_{ij} are the observed and expected values in the contingency table, respectively. The feature that results in the largest statistic value indicates greater association and will be selected as the splitting feature [26].

1.3.4. The G-Statistic

Another splitting statistic based on information theory is G-statistic, which also makes use of a contingency table. Using the information measure obtained from equation (1), the chi-square statistic can be approximated by

$$G = 2N \times I, \tag{8}$$

where N is the number of examples. As N remains constant through the entire features, G and I will always have the same ordering. Again, the feature with largest G will be selected as the splitting feature [27].
In [28, 29] the comparison of properties and empirical results of some traditional node splitting measure based decision trees are discussed.
Recently new node splitting measures have been proposed. In [30], if the data is numerical, after presorting the attribute values, the splitting measure computed successive middle of distinct values for each attribute as follows

$$\text{Measure (Splitting point } p) = \frac{|S_1|}{|R|} \times \frac{C_{S_1}}{C_R} \times \sum_{i=1}^{C_{S_1}} \frac{n_i(S_1)}{n_i(R)} + \frac{|S_2|}{|R|} \times \frac{C_{S_2}}{C_R} \times \sum_{i=1}^{C_{S_2}} \frac{n_i(S_2)}{n_i(R)}, \tag{9}$$

where R includes all the data samples that are available at the current node, S_1 and S_2 are sequentially the subsets from above (right child) and below (left child) the splitting point p; $|.|$ represents the cardinality of a set. C_R, C_{S_1} and C_{S_2} denote number of distinct classes in R, S_1 and S_2 respectively, and $n_i(x)$ is the number of samples in set x that belong to class i. If the data contains a categorical attribute, the number of terms in equation (9) is equal to the number of distinct values that the attribute can take. The attribute with least splitting measure is selected as the splitting attribute.
In 2005, SVMT, a binary classification tree algorithm was proposed [31]. Initially, LLE clustering partitions the entire data set into two disjoint subsets in each node and then a SVM will be trained on each subset. The tree is generated by the continued splitting of data and the algorithm is terminated when the data in a node has distinct labels. This data-driven approach leads to over fitting and huge tree size [32]. To resolve this problem, a modified version of SVMT was constructed in [32] which applies depth-first and breadth-first spanning order preferences.

1.4. Organization

In this paper, a new splitting criterion for building decision trees is proposed. A splitting criterion specifies a tree's best splitting variable as well as the variable's threshold for further splitting. For this purpose, we make use of the correlation concept to find the best splitting variable and use a 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 which is obtained from a SVM. This new decision tree algorithm, named Correlation based Multi Branch Support Vector (C-MBSV), will be presented in Section 3. During the tree generation, a series of empty leaves will be created for data that cannot be classified,these leaves will be called unclassified regions. The causes

of unclassified regions, strategies to reduce or eliminate these regions, and 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 shorter tree height and use of fewer features. We should note that, to the best of our knowledge, no decision tree can be introduced as the universal 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 algorithm uses the combination of the correlation concept to find the best splitting variable and a SVM to specify its threshold as the splitting criterion, we introduce these methods in brief.

2.1. Correlation based feature selection

According to literatures [33–35], 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 variables (features) and the independent variable (label) is examined. 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 block-wise selection yields to different regression equation. The classical model-selection method known as Forward selection [36] and its enhanced versions like Forward Stagewise and LARS [37], 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 has its own distinct steps and computation formulas, at the first step, they act the same. By giving a collection of dependent variables, they begin with an empty equation and select the one having the largest absolute correlation with the independent variable at the first step. Variables of greater theoretical importance are entered first.
Applying the idea from forward selection methods, the first step is utilized to select the splitting feature at each node of the DT. Hence, feature j having the most absolute correlation with the output vector (class label) is chosen as the following:

$$j = \arg\max_i |r_i| = \{|\langle X_i, Y\rangle|\}\,, \tag{10}$$

where, X_i and Y are vectors containing values of the $i'th$ feature and the class label of each data sample respectively. $\langle .,. \rangle$ denotes the dot product of two vectors. As in LARS algorithm, normalization is done before correlation computation so that $\left\|X_i\right\|_2 = 1, \forall i$ and $\left\|Y\right\|_2 = 1$, where $\left\| \ . \ \right\|_2$ is the L2-norm of a vector. We refer the reader to [37] for more details on why normalization is necessary.

2.2. Support vector machines

SVMs [38] are very popular because of their abilities to classify input patterns with minimized structural misclassification risk and find the optimal separating hyper-plane between two classes in the feature space. They are also powerful in learning systems because of the utilization of a kernel machine in linearization, which provides good generalization properties. A non-separable case of a SVM is in the following form:

$$\begin{aligned} f &= \min \frac{1}{2}\left\|w\right\|^2 + C\sum_{i=1}^{n}\xi_i \\ s.t \quad & y_i\left(w^T x_i + b\right) \geqslant 1 - \xi_i,\ \xi_i \geqslant 0,\ i = 1, \ldots, n, \end{aligned} \tag{11}$$

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 $w_0^T x + b_0 = 0$, where $w_0 = [w_{01}, w_{02}, \ldots, w_{0m}]^T$ 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.

2.2.1. Multi-Class SVMs

The SVM formulation was originally developed for binary classification problems and finding the direct formulation for multi-class case is not easy and still an on-going research issue. There are two ways we can have a multi-class SVM classifier: directly consider all data in one optimization formulation, or decompose the multi-class problems to several binary problems. The second solution is a better idea and has been considered more than the first approach [39–41]. This is because binary classifiers are easier to implement and some powerful algorithms such as SVMs are inherently binary [42]. Two major decomposition implementations are “one-against-all” and “one-against one”.

The one-against-all [38] method constructs n SVMs where n is the number of classes. The i'th SVM is trained to separate the i'th class from remaining classes. The one-against-one [40] (pair-wise SVMs) instead, constructs $\frac{n(n-1)}{2}$ decision functions for all the combinations of class pairs. In determination of a decision function for a class pair, the training data for the corresponding two classes is used. Thus, in each training session, the amount of the training data is reduced considerably compared to the one-against-all approach, which use all the training data. Experimental results in [43] indicate that the one-against-one is more suitable for practical use.

The idea of maximizing the margin between classes in a SVM is used to find the best threshold at each node via pair-wise SVMs in the proposed DT algorithm.

3. The proposed method

Recently, methods of rule extraction from decision trees have been of interest to the authors of [44, 45] and [23] for the purpose of data classification. Nevertheless, DTs have their own restrictions. Finding best feature at each node in a DT requires the calculation of Entropy or Gini index for all features, which is time consuming. Additionally, as shown in Fig. 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, e.g. if n distinct sorted values exists in the database for a special feature, Gain should be computed $(n - 1)$ times to obtain the best threshold.

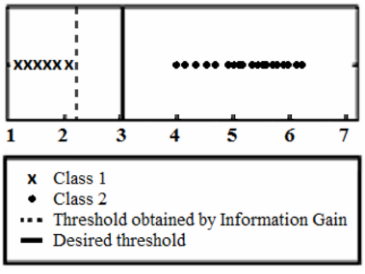


Figure 1. Disparity of threshold obtained using Information Gain with the desired one.

In this paper, we use the concept of correlation from 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. 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 make use of this idea in data classification is SVM. A binary SVM classifies samples into two classes such that the margin $\frac{2}{\|w\|}$ is maximized, or in the other word minimizing $\frac{\|w\|}{2}$ as in (11).

Using this idea makes the threshold more general and draws it towards 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 shifts the split threshold closer to the desired one. In contrary the C4.5 and CART decision tree algorithms can have more than one threshold because they are dealing with numerical data. Some privileges of the proposed method are variable branching factor, improvement in determining the split threshold, and more efficiency in dealing with noisy data. The proposed method is called the Correlation based Multi Branch Support Vector (C-MBSV) algorithm.

Section 2 discussed selection of the splitting feature based on correlation and the calculation of the splitting threshold by using a SVM. In this section, by combining C4.5, SVM and correlation concept, we proposed a simpler and more efficient method for data classification. The structure of the proposed classifier is shown in Fig. 2.

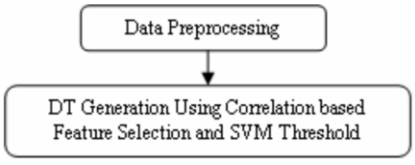


Figure 2. Structure of the proposed method.

3.1. Data preprocessing

C-MBSV is applied only on numerical (continuous or discrete) feature values because SVM and the correlation algorithms need real values to compute. Ordered categorical values must first transfer to numerical values in a preprocessing phase. E.g. a discrete feature with three values of Morning, Midday and Night can be represented by 1, 2 and 3 in sequence. If there is no ordering present in the categorical values, chi-square can be used to determine the correlation. Features with larger chi-square values have larger association, which means of larger correlation [46].

3.2. DT Generation using correlation based feature selection and a 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 at this node. This procedure is shown in Fig. 3 and the pseudo-code is also illustrated as Algorithm 1.

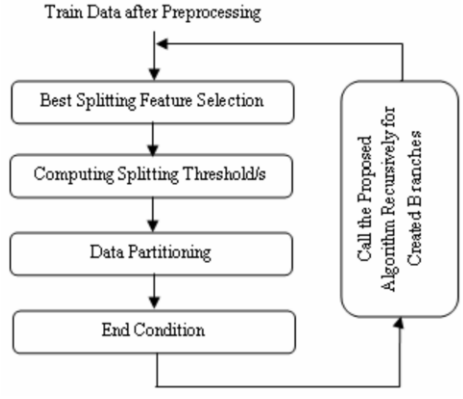


Figure 3. Procedure of DT generation using Correlation based Feature Selection and SVM Threshold function.

Algorithm 1 DT Generation.

```
1: procedure BUILD TREE(Data D)
2:     if all samples in D are belong to a distinct class then
3:         the node is labeled and return;
4:     end if
5:     // f is index of feature that has the most absolute correlation with classes label
6:     f = Best Splitting Feature Selection (D);
7:     // T is matrix of thresholds and D_f is a vector containing values of feature f in D
8:     T = Computing Splitting Threshold (D_f);
9:     // S_i is subset i after partitioning D with (f, T) into b = |T| + 1 branches (where |.| is computed by (14))
10:    [S_1, S_2, ..., S_b]=Add child nodes to decision tree D(f, T);
11:    // Data Partitioning
12:    if only one S_i has data then
13:        // i=1, ..., b
14:        Prune children and labeled the parent node with the most frequently label of classes used in D;
15:    end if
16:    if some S_i between S_k and S_m that 0 < k < b, 0 < m < b, k < m, k and m ≠ i has no data then
17:        Labeled S_i as Unclassified Region;
18:    end if
19:    for B = 1 to b do
20:        if S_B does not take label then
21:            Build Tree(S_B);
22:        end if
23:    end for
24: end procedure
```

A: Best splitting feature selection

As discussed previous in Section 2.1, by applying the idea from greedy forward stepwise procedures, correlation is used to select the best splitting feature. At each current node in the DT, the inner dot product between two vectors containing values of the $i'th$ feature and the class label of data samples is computed by (10) for $i = 1, \ldots, N$, where N is the number of features. Note that at each node, (10) is calculated just on the unclassified data samples that have been assigned to this branch of the tree. Feature f that leads to the larger value is selected as the best splitting feature.

B: Computing splitting threshold/s:

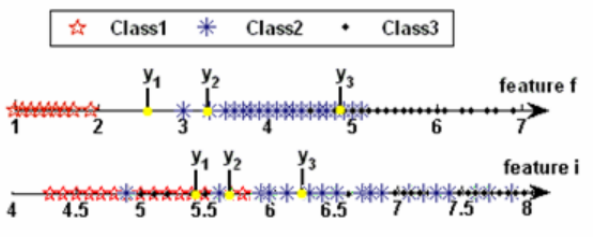
After selection of the dominant splitting feature f, the splitting threshold/s are computed on D_f (where D_f is a vector containing values of feature f and D is the entire data samples) as follows:

$$T^f = \begin{bmatrix} t^f_{11} & t^f_{12} & \cdots & t^f_{1n} \\ t^f_{21} & t^f_{22} & \cdots & t^f_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ t^f_{n1} & t^f_{n2} & \cdots & t^f_{nn} \end{bmatrix}, \quad f \text{ is index of dominant feature and } t^f_{ww} = 0, \qquad w = 1, \cdots, n, \tag{12}$$

where n is the number of classes and t^f_{ab} represents the splitting criterion between class a and b at the f'th feature, which obtained from

$$\left(w^T\right)_{ab} t^f_{ab} + b_{ab} = 0, \quad f \text{ is index of dominant feature and } a, b = 1 \ldots, n. \tag{13}$$

In the above equation, $\left(w^T\right)_{ab}$ and b_{ab} are the same parameters w^T and b in equation (11) which are obtained by solving the dual programming of the SVM between class a and b when $x_i = D_f$ [38] or by applying Sequential Minimal Optimization (SMO) [47], which is faster than solving the dual programming of SVM. T^f that is computed for the dominant feature is a symmetric matrix with zero diagonal elements. In Fig. 4, a classification example with two features (i and f) and three classes is demonstrated. In this example, feature f which has the most absolute correlation defeat feature i as the dominant feature. Then, matrix T^f is calculated for feature f where the values are also shown in Fig. 4.



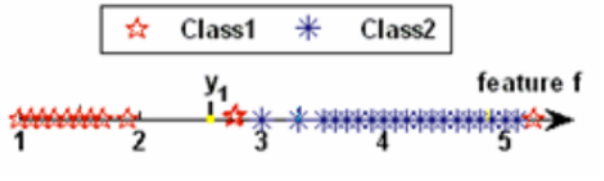
$$T^f = \begin{bmatrix} 0 & 2.55 & 3.31 \\ 2.55 & 0 & 4.86 \\ 3.31 & 4.86 & 0 \end{bmatrix}$$

Figure 4. Selection of the f'th feature causes minimum irregularity within DT.

The maximum number of distinct splitting thresholds at each node in the tree is the number of elements in the upper triangular matrix without considering diagonal elements which is calculated as follows:

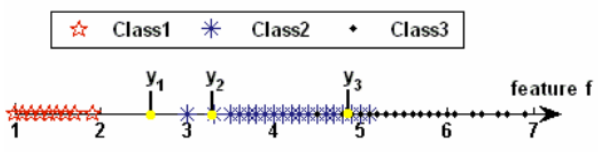
$$\left|T^f\right| = \frac{n \times n - n}{2}, \tag{14}$$

where n and $\left|T^f\right|$ are the number of classes and the maximum number of distinct splitting thresholds at each node respectively. Maximum branching factor at each node is equal to $b = \left|T^f\right| + 1$. For example, if the data in a node has three classes then $\left|T^f\right| = \frac{3\times3-3}{2} = 3$, and if the data in a node has two classes then $\left|T^f\right|$ is 1. These cases are shown in Fig. 5 along with the splitting thresholds. In Fig. 5a the data contains two distinct classes and therefore there exists just one splitting threshold, but in Fig. 5b since the data contains three distinct classes, there are three splitting thresholds.



$$T^f = \begin{bmatrix} 0 & 2.61 \\ 2.61 & 0 \end{bmatrix}$$

a



$$T^f = \begin{bmatrix} 0 & 2.55 & 3.31 \\ 2.55 & 0 & 4.86 \\ 3.31 & 4.86 & 0 \end{bmatrix}$$

b

Figure 5. The classified data in a node on dominant feature. (a) The data with two classes; therefore, there is one splitting threshold. (b) The data have three classes; therefore, there are three splitting thresholds.

The next part describes data partitioning based on the dominant feature and its obtained splitting threshold T^f.

C: Data partitioning

In this part, the procedure of dividing data into branches is explained. After choosing the best splitting feature f, the training data samples are sorting based on this feature. That is, the data samples having lowest values of f come first. The sorted matrix of data samples D will be partitioned into b branches by using the obtained splitting thresholds T^f

in to $S_1, S_2, \cdots, S_b$ subset nodes. For example Fig. 6 illustrates data partitioning of Fig. 4 in corresponding to feature f and splitting thresholds T^f.

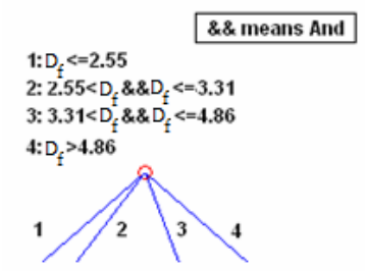


Figure 6. Data partitioning of Fig. 4 in corresponding to feature f and splitting thresholds T^f.

D: End conditions:

After dividing the data into the branches, termination conditions and labeling must be checked. If none of the termination conditions are satisfied the algorithm is called recursively. Three termination conditions are described below.

1. If all the data belongs to the i'th class in a sub node ($S_i,\ i = 1, \cdots, b$) then it takes the i'th class as the leaf label and the algorithm is terminated for this branch. An example of this condition is shown in Fig. 7.

2. If only one $S_i,\ i = 1, \cdots, b$ has data then the label is selected as the most frequent class in the data samples. There exists a reason to choose most frequent class as the label. Data samples on a leaf are projected based on only one feature, and so they may placed on each other. This case occurs when two or more data samples have the same values in feature f. In other words, data samples are not linearly separable based on one feature of the input space. If we call the algorithm again to classify these data samples we get a loop. Fig. 8a is an example of this case, where the parent node has 54 samples of class one and 2 samples of a different class, and SVM cannot find the optimum hyperplane. After determination of the label all S_i are pruned and the parent node takes the label. The pruning of Fig. 8a is shown in Fig. 8b.

3. If some S_i between S_k and S_m such that $0 \prec k \prec b$, $0 \prec k \prec b$, $k \prec m$, k and $m \neq i$ has no data, the corresponding leaf marked as *Unclassified Region*. Fig. 9 shows an example of this case. The causes of these regions and methods of reducing or eliminating them are discussed in the next section.

If none of the termination conditions are satisfied for each of $S_i,\ i = 1, \cdots, b$ the C-MBSV algorithm will be called recursively as shown in Fig. 3.

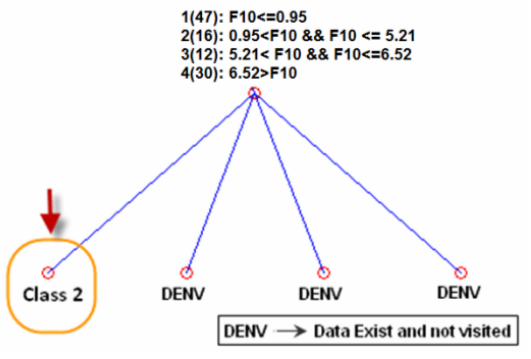


Figure 7. The data is divided into four branches. All the data samples of the left branch belong to class 2. (In Figures 7, 8a, 8b and 9 the presentation of "a(b):Fi" above each nodes means that the number of b data samples interred the a'th branch and the splitting feature was i.)

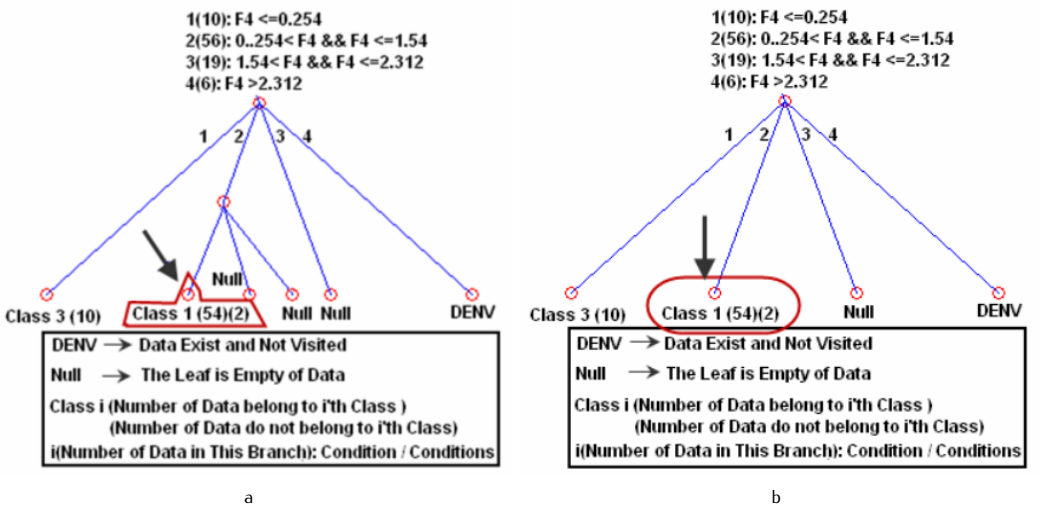


Figure 8. Some branches have data. (a) Fifty-four data samples in the marked leaf belong to the 1'st class and two data samples belong to other classes. (b) Tree after pruning.

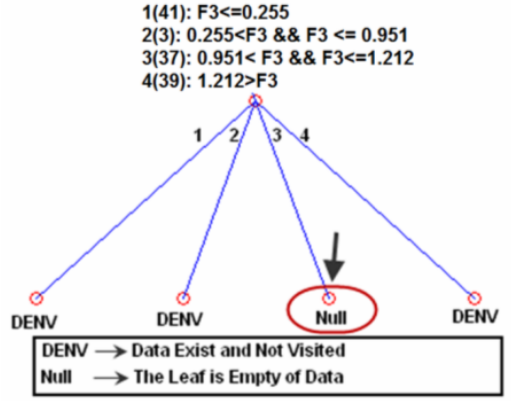


Figure 9. Some branches have data and the marked leaves are called *Unclassified Region*.

4. Discussion

We separate our discussion in two parts. First *Unclassified Regions* that are introduced in the previous section are discussed. The cause of these regions and methods to reduce or eliminate these regions will be explained. Then, we explain our experimental results.

4.1. 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 these regions 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 are Unclassified Regions generated?

There are two main reasons for generating these regions:

a) Because one feature is used at each node for categorizing, data projection exists in this algorithm. The result of this data projection is one dimensional and the data may have overlaps based on one feature.

b) Due to inappropriate distribution of training data over the input space, adequate training data is unavailable. The solution is to add more training data, but since all possible cases of training pattern may not be available, this solution may be impossible or too time consuming. This problem can be solved by increasing the amount of training data while keeping the data general.

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 the K-Nearest Neighbors algorithm [48] 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. Towards this purpose, more features at each node can be selected for categorizing the data. 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 produced tree will have a height 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. The KNN algorithm, which has highly improved decision making performance, was used to classify data from the *Unclassified Regions* in the experimental results.

4.2. Experimental results

Our method is validated using 10 datasets, all downloaded from *UCI Machine Learning Repository*[1], 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 was randomly partitioned into folds for all the datasets, and held constant for all DT algorithms. Our method, C-MBSV, is compared against C4.5 and CART. C4.5 algorithm was implemented in Linux[2]; 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 3 GHz 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 4. During the results, imbalance conditions were considered. Imbalance problems happen in a

Table 1. 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
MAGIC Gamma Telescope	10	19020	2
Pima	8	768	2
Spambase	57	4601	2
Wine	13	178	3
Zoo	17	101	7

[1] *UCI, http://archive.ics.uci.edu/ml/datasets.html, (last accessed: May, 2010), UCI Repository of machine learning databases and domain theories.*
[2] *R. Quinlan, http://www.rulequest.com/Personal/, (last accessed: May, 2010), Ross Quinlan's Home Page*

data set when the number of data samples of one class is much larger than those of another class. In this condition, the SVM usually computes the 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 (11) 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.
Table 2 shows the error rates. The results on test error rate show that our DT algorithm usually has the same or better error rates when compared with CART and C4.5. While the test error rates for C-MBSV are to some extent lower for databases with a large number of features and data samples like Spambase and MAGIC Gamma Telescope, the height of C-MBSV tree is significantly shorter than the trees produced by the CART or C4.5 algorithms. Note that for Spambase and MAGIC Gamma Telescope databases C-MBSV, C4.5 and CART trees are binary.

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
MAGIC Gamma Telescope	15.38	4.00	9.05	17.26	17.26	17.90	14.93
Pima	24.83	7.20	15.73	25.12	25.12	29.04	25.27
Spambase	7.48	2.06	2.71	9.08	9.08	8.04	7.95
Wine	0.04	1.68	1.08	6.77	6.77	13.99	7.82
Zoo	0.99	5.79	0.22	7.93	7.93	11.11	3.18

In Section 4.1 we also discussed data in the *Unclassified Regions* and the use of the KNN algorithm to handle them. It is obvious that when there exists no test data in *Unclassified Regions*, or no such regions exists, the results are the same as when KNN is not used. The 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 the DTs produced by our algorithm against two other algorithms. It is obvious that our DTs are superior in height, which results in lower induction time for test data. Lastly, Table 4 compares the number of needless features excluded in DTs generated by C-MBSV against those excluded from the trees generated by C4.5. These numbers are another advantage of our algorithm that leads to most related and fewer features (while having a lower or higher test error rate). This advantage can be used as feature reduction or preprocessing phase of applications that need to work with fewer features. Consequently, the results show that the proposed algorithm outperforms the CART and C4.5 algorithms in terms of shorter height and fewer number of needless features in the final decision tree.

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
MAGIC Gamma Telescope	5.26	33.66	21.50
Pima	2.20	12.87	9.40
Spambase	11	30	28.50
Wine	3.67	4.27	3.07
Zoo	5.93	6.13	5.06

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
MAGIC Gamma Telescope	5.67	0.00
Pima	5.94	1.20
Spambase	33.47	17
Wine	5.80	9.40
Zoo	9.14	11.40

5. Conclusion

In this paper, a new DT algorithm is proposed, the Correlation based Multi Branch Support Vector (C-MBSV) algorithm. In the proposed DT algorithm, the correlation concept is used to find the best splitting variable, and then its relevant splitting thresholds is specified by using a SVM. Since the algorithm can return multiple 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, which we called *Unclassified Regions*. The obtained results show that the C-MBSV algorithm seems to outperform the CART and C4.5 algorithms in terms of shorter height and fewer number of needless features.

Future work

In future, we want to enhance our algorithm by splitting on more than one variable at each node. Furthermore, we want to use the proposed method on classification applications.

References

[1] Kim Y.S., 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 SYST APPL, 2008, 34, 1227-1234
[2] Breiman L., Friedman J., Stone C.J., Olshen R.A., Classification and Regression Trees, The Wadsworth statistics/probability series, Wadsworth and Brooks, Monterey, CA, 1984
[3] Quinlan J.R., Induction of Decision Trees, MACH LEARN, 1986, 1, 81-106
[4] Quinlan J.R., C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, California, 1993
[5] Kass G.V., An exploratory technique for investigating large quantities of categorical data, APPL STATIST, 1980, 29, 119-127
[6] Brodley C.E., Utgoff P.E., Multivariate decision trees, MACH LEARN, 1995, 19, 45-77
[7] Loh W.-Y., Shih Y.-S., Split selection methods for classification trees, STAT SINICA, 1997, 7, 815-840
[8] Loh W.-Y., Vanichsetakul N., Tree-structured classification via generalized discriminant analysis (with discussion), J AM STAT ASSOC, 1988, 83, 715-728
[9] Muller W., Wysotzki F., Automatic construction of decision trees for classification, ANN OPER RES, 1994, 52, 231-247

[10] Rastogi R., Shim K., PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning, DATA MIN KNOWL DISC, 2000, 4, 315-344
[11] Lim T.-S., Loh W.-Y., Shih Y.-S., A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, MACH LEARN, 2000, 40, 203-228
[12] Maruf Hossain M., Rafiul Hassan M., Bailey J., ROC-tree: A Novel Decision Tree Induction Algorithm Based on Receiver Operating Characteristics to Classify Gene Expression Data, 8th SIAM International Conference on Data Mining (SDM08) (April 2008 Georgia), 455-465
[13] Ruggieri S., Efficient C4.5, IEEE T KNOWL DATA EN, 2002, 14, 438-444
[14] Qin Z., Lawry J., Decision tree learning with fuzzy labels, INFORM SCIENCES, 2005, 172, 91-129
[15] Aitkenhead M.J., A co-evolving decision tree classification method, EXPERT SYST APPL, 2008, 34, 18-25
[16] Vens C., Struyf J., Schietgat L., Džeroski S., Blockeel H., Decision trees for hierarchical multi-label classification, MACH LEARN, 2008, 73, 185-214
[17] Chen Y.L., Hu H.W., Tang K., Constructing a decision tree from data with hierarchical class labels, EXPERT SYST APPL, 2009, 36, 4838-4847
[18] Hansen S.E., Olsson R., Improving decision tree pruning through automatic programming, Proceedings of the Norwegian Conference on Informatics (NIK-2007) (November 2007, Holmenkollen Park Hotel Rica, Oslo), 2007, 31-40
[19] Sieling D., Minimization of decision trees is hard to approximate, J COMPUT SYST SCI, 2008, 74, 394-403
[20] Markey M.K., Tourassi G.D., Floyd C.E., Decision tree classification of proteins identified by mass spectrometry of blood serum samples from people with and without lung cancer, PROTEOMICS, 2003, 3, 1678-1679
[21] Zmazek B., Todorovski L., Dzeroski S., Vaupotic J., Kobal I., Application of decision trees to the analysis of soil radon data for earthquake prediction, APPL RADIAT ISOTOPES, 2003, 58, 697-706
[22] Pulkkinen P., Koivisto H., Fuzzy classifier identification using decision tree and multiobjective evolutionary algorithms, INT J APPROX REASON, 2008, 48, 526-543
[23] Pulkkinen H.K.P., A Dynamically Constrained Multiobjective Genetic Fuzzy System for Regression Problems, IEEE T FUZZY SYST, 2010, 18, 161-177
[24] Mitchell T.M., Machine Learning, McGraw-Hill International, New York, 1997
[25] Mehta M., Agrawal R., Riassnen J., SLIQ: a fast scalable classifier for data mining, Extending Database Technology, (March, Avignon, France), 1996, 18-33
[26] Kass G.V., An exploratory technique for investigating large quantities of categorical data, APPL STATIST, 1980, 29, 119-127
[27] Mingers J., Expert systems - rule induction with statistical data, J OPER RES SOC, 1987, 38, 39-47
[28] Mingers J., An Empirical Comparison of Selection Measures for Decision-Tree Induction, MACH LEARN, 1989, 3, 319-342
[29] Shih Y.-S., Families of splitting criteria for classification trees, STAT COMPUT, 1999, 9, 309-315
[30] Chandra B., Varghese P.P., Moving towards efficient decision tree construction, INFORM SCIENCES, 2009, 179, 1059-1069
[31] Pang S., Sr K.D., Bang S.Y., Face membership authentication using SVM classification tree generated by membership-based LLE data partition, vol. 16, ETATS-UNIS: Institute of Electrical and Electronics Engineers, New York, NY, 2005
[32] Pang S., Kasabov N., Encoding and decoding the knowledge of association rules over SVM classification trees, KNOWL INF SYST, 2009, 19, 79-105
[33] Archdeacon T.J., Correlation and Regression Analysis: A Historian's Guide, University of Wisconsin Press, Madison, 1994
[34] Miles J., Shevlin M., Applying Regression and Correlation: A Guide for Students and Researchers, Sage Publications Ltd, London, 2000
[35] Cohen J., Cohen P., West S.G., Aiken L.S., Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences, 3rd ed., Routledge Academic, Mahwah, NJ, 2002
[36] Weisberg S., Applied Linear Regression, Wiley, New York, 1980
[37] Efron B., Hastie T., Johnstone I., Tibshirani R., Least angle regression, ANN STAT, 2004, 32, 407-499
[38] Vapnik V.N., The Nature of Statistical Learning Theory, Springer, New York, 1995
[39] Wu T.-F., Lin C.-J., Weng R.C., Probability Estimates for Multi-class Classification by Pairwise Coupling, J MACH LEARN RES, 2004, 5, 975-1005

[40] Hastie T., Tibshirani R., Classification by pairwise coupling, Proceedings of the 1997 conference on Advances in neural information processing systems 10 (Mahwah, NJ, Denver, Colorado, United States) MIT Press, 1998, 507-513
[41] Allwein E.L., Schapire R.E., Singer Y., Reducing multiclass to binary: a unifying approach for margin classifiers, J MACH LEARN RES, 2001, 1, 113-141
[42] Zhou J., Peng H., Suen C.Y., Data-driven decomposition for multi-class classification, PATTERN RECOGN, 2008, 41, 67-76
[43] Hsu C.-W., Lin C.-J., A comparison of methods for multiclass support vector machines, IEEE T NEURAL NETWOR, 2002, 13, 415-425
[44] Barakat N., Diederich J., Eclectic Rule-Extraction from Support Vector Machines, INT J COMPUT INT SYS, 2005, 2, 59-62
[45] Farquad M., Ravi V., Bapi R.S., Rule extraction using Support Vector Machine based hybrid classifier, IEEE Region 10 Conference TENCON 2008 (2008, Masab Tank, Hyderabad) 2008, 1-6
[46] Reynolds H.T., The Analysis of Cross-Classifications, Free Press, New York, 1977
[47] Platt J.C., Fast Training of Support Vector Machines using Sequential Minimal Optimization, Advances in Kernel Methods – Support Vector Learning, MIT Press, Xiamen, China, 1998
[48] Cover T.M., Hart P.E., Nearest Neighbor Pattern Classification, IEEE T INFORM THEORY, 1967, 13, 21-27