Multi-Test Decision Trees for Gene Expression Data Analysis

Marcin Czajkowski[1], Marek Grześ[2], and Marek Kretowski[1]

[1] Faculty of Computer Science, Bialystok University of Technology, Wiejska 45a, 15-351, Bialystok, Poland
{m.czajkowski,m.kretowski}@pb.edu.pl

[2] School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, N2L 3G1, Canada
mgrzes@cs.uwaterloo.ca

Abstract. This paper introduces a new type of decision trees which are more suitable for gene expression data. The main motivation for this work was to improve the performance of decision trees under a possibly small increase in their complexity. Our approach is thus based on univariate tests, and the main contribution of this paper is the application of several univariate tests in each non-terminal node of the tree. In this way, obtained trees are still relatively easy to analyze and understand, but they become more powerful in modelling high dimensional microarray data. Experimental validation was performed on publicly available gene expression datasets. The proposed method displayed competitive accuracy compared to the commonly applied decision tree methods.

Keywords: Decision trees, classification, gene expression, univariate tests.

1 Introduction

Decision trees represent one of the most popular classification techniques [19,20]. Their chief advantage is the fact that they are easy to understand by humans which makes them particularly useful when the aim of modelling is to understand the underlying processes of the environment. Decision trees are also applicable when the data does not satisfy rigorous assumptions required by more traditional methods [15,7]. However, existing attempts to apply decision trees to the classification of gene expression data showed that standard decision tree algorithms are not sufficient for inducing competitive classifiers [14].

In this paper, we introduce a new type of decision trees that allow testing more than one feature in a single node of the tree. Every split of such trees is composed of a set of univariate tests and is called a multi-test split. Trees which are based on such tests are called Multi-Test Decision Trees ($MTDT$).

1.1 Background and Motivation

Gene expression data is extremely challenging for computational tools and mathematical modelling [21]. Each observation is described by a high dimensional feature vector with a number of features reaching even a few dozens of thousands, whereas the number of observations is rarely higher than one hundred. This high ratio of variables/cases requires new era computational tools to extract significant and meaningful rules from this kind of data. Existing decision tree learning algorithms can easily find a split which separates the training data very well at a given level in the tree, but such a split can correspond to noise only. This situation is more probable at intermediate and lower levels of the tree.

A short example will illustrate this problem. Assuming that at a given level of the tree there are 20 observations (10 from class A and 10 from class B) and 2×10^5 features, the number of possible partitions of this training set (the number of combinations of choosing 10 out of 20 instances) is smaller (the exact number is $184,756$) than the number of available features. This makes it very easy to find a split, i.e., an attribute and its corresponding threshold, which can split this data perfectly. When there are only 10 observations in the node, the number of combinations is only 250 whereas the number of attributes is 3 orders of magnitude higher. When there is only one univariate test that splits the data, there is a very high risk of choosing faulty splits which correspond to noise.

In this paper we tackle the problem of improving the performance of decision trees on gene expression data. Our focus is on univariate trees since they are a 'white-box' technique and this fact makes them particularly interesting for scientific modeling. They are much easier to understand than trees with multivariate splits and much easier to learn from data. However traditional algorithms, for example, $C4.5$ [23] or $CART$ [4], fail to produce decision trees with high classification accuracy on gene expression data. Our previous work with various univariate decision tree algorithms showed that these algorithms produce considerably small trees which classify the training data perfectly but fail in classifying unseen instances [14]. Only a small number of attributes is used in such trees and their model complexity is low (high bias) therefore they underfit the training data [20]. Producing bigger trees using standard algorithms such as $C4.5$ does not solve the problem in the case of gene expression data because small trees often classify the training data perfectly [14].

This indicates that the issue of split complexity could be advocated here since not much can be gained from bigger univariate decision trees on this kind of data. Standard techniques of improving the performance of classification algorithms, e.g., ensemble methods when applied to decision trees result in complex classifiers that are almost impossible to understand by humans [8,24]. There are also algorithms which apply multivariate tests [6] based mostly on a linear combination of features. These kinds of decision trees with multivariate splits or bagging/boosting methods often outperform existing univariate algorithms on gene expression data [25,16]. They generate, however, more complex classification rules that from the medical point of view are more difficult to understand and analyze.

Some feature selection should be taken into account especially in the context of microarray data. Providing a group of genes that contributes most to the classification task like in [1,10] may significantly improve the performance of decision trees.

1.2 Related Work

One of the approaches that addresses the issue of the test complexity in decision trees was explored by Berzal et al. [3] who proposed multi-way decision trees using multi-way splits. In [3], a hierarchical clustering of attribute values is combined with the standard greedy decision tree algorithm. The author reduces the tree complexity (in terms of the number of nodes) by using multiple thresholds in each split on a single numerical attribute. This will potentially increase the branching factor of such splits, however such tests will be more expressive and the overall number of nodes in the corresponding decision tree will be smaller. In contrast to our solution, multi-way splits used in [3] are based on a single attribute which is not sufficient to overcome the high ratio of features/observations in the gene expression data.

The specific character of gene expression data and its influence on the process of building decision trees was investigated by Li et al. [18]. This solution was focused on using committees of trees to aggregate the discriminating power of a bigger number of significant rules and make more reliable predictions. Firstly, all features are ranked according to the gain ratio [23]. In the next step, the first tree using the first top-ranked feature in the root node is built. Next, the second tree using the second top-ranked feature in the root node is built and the process continues until the k-th tree using the k-th top-ranked feature is obtained. The classification of the final committee of k decision trees is governed by weighted voting. It was observed that:

- well performing rules often contain features which are globally low-ranked;
- if the construction of a tree is confined to a set of globally top-ranked features, the rules in the resulting tree may be less accurate than rules derived from the entire feature space;
- alternative trees often outperform or compete with the performance of the greedy tree.

This work also supports our approach to use many univariate splits in multi-test decision tree induction algorithm. In particular, our aim is to make use of features which are globally lower-ranked and use them jointly in multi-tests. However, our aim is also to preserve simplicity of final decision trees, which is not the case in [18].

The rest of the paper is organized as follows. In the next section, an algorithm to learn multi-test decision trees is presented. In Section 3, the proposed approach is experimentally evaluated on real gene expression data. The paper is concluded in the last section and future work is also discussed.

2 Multi-Test Decision Trees

Regardless which approach to construct decision trees is used, one of the dimensions by which decision trees can be characterized is the number of features which are tested at each node. Standard algorithms, such as C4.5 [23], use univariate splits, which means that only one feature is checked in each internal node of the tree. In this paper, we introduce a new type of decision trees that allow testing more than one feature in a single internal node of the tree. Every split of our trees is composed of a set of univariate tests and is called a multi-test split. The fact that these elementary splits are univariate and the way they are combined show that our approach is substantially different from multi-variate, e.g. oblique, spits. Trees which are based on our approach are called Multi-Test Decision Trees (*MTDTs*), because several univariate elementary tests can be applied in every internal node of the tree. Every univariate test of the multi-test corresponds basically to one split from classical algorithms such as C4.5, and our extension is about combining such individual tests into more complex multi-test splits. The reminder of this section introduces our algorithm for learning and applying for classification such multi-test splits.

Decision trees can be constructed using different methods among which top-down induction is the most common. In what follows, it is assumed that such top-down induction is used, and the further description focuses on our novel idea of multi-test splits, which could be essentially used with other types of decision tree learning methods as well.

2.1 Learning Multi-Test Splits

Let M be a number of training instances $X = \{x_1, x_2, \ldots, x_M\}$ in a given node. Each instance is described by P attributes denoted as $F = \{f_1, f_2, \ldots, f_P\}$. Let $x_{i,j}$ denote the value of the attribute j of the instance i. Each non-terminal node contains a set of W multi-tests denoted as MT ($MT = \{mt_1, mt_2, \ldots, mt_W\}$) from which only one will be chosen in order to split the training instances into two groups and create a branch for each outcome of the test. Each i-th multi-test is composed of a group of no more than N univariate tests in which one is called a *primary splitter* (PS_i) and the rest $N-1$ *surrogate splitters* ($S_{i,j}$ where $1 \leq j < N$). The parameter denoted as N represents the maximum number of one-dimensional tests that constitute the multi-test.

The $MTDT$ splitting criterion is directed by the majority voting mechanism where the result of each test constitutes a single vote. For this reason, surrogate tests have considerable impact on decisions of multi-tests because they can out-vote the primary splitters. It should be noted that this impact can be positive as well as negative and effects the gain ratio for the entire multi-test. Additionally, we create not one but W multi-tests that can compete with each other. Therefore, the best multi-test that will be used as a splitting criterion may not contain the test with highest gain ratio (PS_1). This can happen when a competitive multi-test mt_i ($1 < i \leq W$) has a higher gain ratio than mt_1. The illustration of finding the splitting rule for an internal node of the $MTDT$ is showed in Fig. 1.

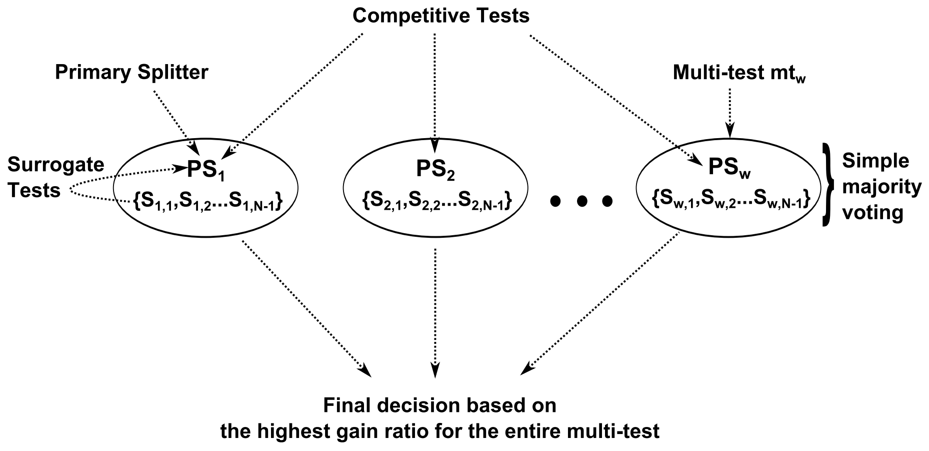

Fig. 1. An example of finding the best multi-test from the set of multi-tests for a non-terminal node in *MTDT*

First, the algorithm searches for the best possible thresholds. This process is similar to the search function in the *C*4.5 algorithm. Next, the *W* multi-tests are calculated and the one with the highest gain ratio is chosen to split the training instances. The algorithm that finds the splitting rule (the best multi-test) for a given node of the *MTDT* during top-down induction is presented below.

```
Inputs:
    M - number of training instances X={x1,x2,...,xM} in a node
    P - number of attributes F={f1,f2,...,fP}
    W - number of multi-tests
Initialize:
    V - Vector of pairs: {threshold h, gain ratio gr}
    MT - Empty vector MT={mt1,mt2,...,mtW}
Training:
    FOR i in {1,...,P}
        FOR j in {1,..,M-1}
            h_(i,j) = 0.5*(x_(i,j) + x_(i,j+1))
            IF IsCandidateThreshold(h_(i,j)) is True
                gr = gain ratio of h_(i,j)
            add pair {h_(i,j),gr} to V
        ENDIF
        ENDFOR
    ENDFOR
    Sort V decreasingly according to the highest gain ratio
    MT[1] = BuildMultitest(V[1].h)
    FOR i in {2,...,W}
        h = FindCompetitive(V,MT)
        MT[i] = BuildMultitest(h)
    ENDFOR
    FOR i in {1,...,W}
        calculate gain ratio for MT[i]
    ENDFOR
RETURN multi-test with the highest gain ratio from MT
```

Specific functions, such as *IsCandidateThreshold*, *BuildMultitest* and *FindCompetitive*, are discussed in detail in subsequent sections.

IsCandidateThreshold. Function *IsCandidateThreshold* guides the search process of the possible thresholds. At the beginning of the algorithm, we search for a vector V that contains pairs of a threshold and a gain ratio that is calculated from the univariate test obtained from that threshold. If the attributes are nominal, the set of possible values an attribute can take is limited and usually small. Finding the potential set of tests for the continuous-valued attributes is somewhat more difficult. In this case, one needs to calculate and rank all tests that involve one feature only. Each single test compares the value of an attribute f_j $(1 \leq j \leq P)$ against a threshold $h_{k,j}$: $f_j \geq h_{k,j}$ where $h_{k,j}$ denotes the value of the k-threshold $(1 \leq k < M - 1)$ on the attribute j. To formulate the test, we sort the training instances based on the values of an attribute f_j in order to obtain a finite set of values $\{x_{1,j}, x_{2,j}, \ldots, x_{M,j}\}$.

Any threshold $h_{k,j}$ between $x_{i,j}$ and $x_{i+1,j}$ $(1 \leq i < M)$ will have the same effect when dividing the training instances, so we need to check only $M - 1$ possible thresholds for each numerical attribute f_j. In Fig. 2, it can be observed that some regular thresholds should not be considered, for example, $h_{1,j}$, $h_{4,j}$ and $h_{M-1,j}$. Tests performed on those thresholds are useless for creating new tests because they split two training instances from the same class. Therefore in order to optimize the performance, we consider only the relevant threshold called candidate threshold [11]. The proposed algorithm performs this optimization using the *IsCandidateThreshold* function. All candidate thresholds are added to the vector V and sorted according to the highest gain ratio. In our work, the gain ratio criterion is used to determine the best possible threshold, and the midpoint, $h_{k,j}$, of the interval $[x_{i,j}, x_{i+1,j}]$ is applied as the value of this threshold: $h_{k,j} = \frac{x_{i,j}+x_{i+1,j}}{2}$. It differs slightly from the implementation in the $C4.5$ algorithm, where the threshold is set to the largest value of f_j in the entire training set that does not exceed the above interval midpoint.

BuildMultitest. Let us consider the first multi-test mt_1. Let PS_1 be a single univariate test performed on a threshold from the input parameter of BuildMultitest function for mt_1. In this particular case, the PS_1 will have the highest gain ratio in the node because it was built on the best possible threshold $(V[1].h)$. However we believe that applying a single test based on one attribute may cause the classifier to underfit the learning data due to low complexity of such a classification rule. For this reason, the multi-test is composed of a group of N univariate tests. The parameter denoted as N represents the maximum number of one-dimensional tests that constitute the multi-test in each non-terminal node.

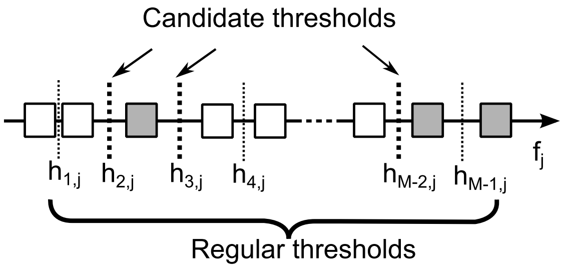

Fig. 2. Candidate thresholds on attribute f_j

Those tests will support the division of the training instances made by the primary splitter PS_1. In other words, the remaining tests of the multi-test should, using the remaining features, branch the tree in similar way to PS_1.

In order to determine surrogate tests, we have adopted a solution proposed in the *CART* system. The use of the surrogate variable at a given split results in a similar node impurity measure. It also mimics the chosen split itself in terms of which and how many observations goes to the corresponding branch. Therefore, the measure of similarity between the primary splitter and remaining tests of the multi-test is the number of observations classified in the same way. In our method, we also consider tests that classify instances in a inverse (opposite) way to PS_1. For such tests, we reverse the relation between attribute and interval midpoint, and recalculate the score. The primary splitter PS_1 and up to $N-1$ surrogate tests $S_{1,l}$ $(1 \leq l < N)$ constitute the single multi-test denoted as mt_1.

FindCompetitive. Function searches for a threshold that will be applied in the *BuildMultitest* function for mt_i where $1 < i \leq W$. This threshold will be used to build an *i primary splitter* (PS) for the i multi-test. The process of obtaining the first multi-test denoted as mt_1, whose primary splitter PS_1 has the highest gain ratio, was shown in previous paragraph. Here we describe alternative multi-tests that are also built in each non-terminal node and which compete with mt_1. The process of building multi-tests, mt_i $(1 < i \leq W)$, requires finding new primary splitters PS_i which together with their surrogate tests may outperform mt_1.

Two factors should be taken into consideration while choosing PS_i. Firstly, the primary splitters PS_i should be competitor splitters to PS_1. Competitor splitters, alike surrogate splitters $S_{1,i}$, yield high gain ratio but are not as good as the primary splitter PS_1. A significant difference between these splits is the way variables are ranked. Surrogate splitters are not evaluated on how much improvement they yield in reducing node impurity but rather on how closely they mimic the split determined by the primary splitter. Competitor splits are runners-up to the primary split and are ranked according to the highest gain ratio. We denote splitters as competitor splitters if their gain ratio is higher than $q\%$ of the best gain PS_1 (the default value equal to 95%). Using more competitor tests in the search process for the primary split (low q value) may lead to the selection of tests with low gain ratio. However, decreasing the number of competitor tests (high q) may cause the PS_i be too similar to PS_1. To sum up: the surrogate splitters are similar to the primary splitter, whereas competitor splitters are those which have highest gain ratio.

The second element that should be taken into consideration is that the same variable is often listed as both a competitor and a surrogate. It may result in obtaining alternative multi-tests, mt_i, that contain similar or identical univariate tests and do not provide any improvement. Therefore competitor splits should be diversified to make the alternative multi-tests also diversified. Function *FindCompetitive* finds the primary splitter PS_i in a loop for $W - 1$ multi-tests. Each PS_i must be a competitor splitter to PS_1 and be the worst average surrogate to all primary splitters PS_j where $j < i$. The next step is to build multi-test mt_i according to PS_i in the same way as in Section 2.1.

2.2 Multi-Test Size and Prediction

The size of the multi-test has a critical impact on its performance and a splitting decision. The parameter denoted as N represents the maximum number of univariate tests in a multi-test and is defined by the user. To classify observations, simple majority voting mechanism is employed in which each test has an equal vote. In the case of a draw, the decision is made in accordance with the primary splitter. In order to determine the final decision, the gain ratio for each of W splits determined by multi-tests, mt_i $(1 \leq i \leq W)$, is calculated and compared. The multi-test with the highest gain ratio is then applied in a given node.

The exact size of the multi-test depends on the difference between the primary splitter and surrogate tests. The main idea of the *MTDT* is to use a group of similar tests in a single node instead of one test as in the classical approach to univariate decision trees. If there are tests that do not have a right substitute, surrogate tests should not be added in order to avoid discrepancy in the multi-test. An inappropriate set of surrogate tests may dominate the primary splitter and deteriorate the splitting criterion. Therefore, surrogate tests added to the multi-test should not be different from the primary splitter more than b percent. When $b = 0\%$, it means that no surrogates are accepted, which is equivalent to setting $N = 1$. In this case, the decision tree would become similar to the tree generated by the $C4.5$ algorithm as only one attribute will be used in each multi-test. When $b = 100\%$, it means that all $N - 1$ surrogates join the multi-test. The threshold, b, can be defined by the user (default value equal 10%).

3 Experimental Results

In this section the proposed solution is experimentally verified using real microarray datasets. The results of the *MTDT* algorithm were compared with several popular decision tree systems.

3.1 Setup

The performance of the *MTDT* classifier was investigated using publicly available microarray datasets described in Table 1. These datasets are from the Kent

Table 1. Kent Ridge Bio-medical gene expression datasets

Datasets	Attributes	Training Set	Testing Set
Breast Cancer	24481	34/44	12/7
Central Nervous System	7129	21/39	-
Colon Tumor	6500	40/22	-
DLBCL Standford	4026	24/23	-
DLBCL vs Follicular Lymphoma	6817	58/19	-
DLBCL NIH	7399	88/72	30/50
Leukemia ALL vs AML	7129	27/11	20/14
Leukemia MLL vs ALL vs AML	12583	20/7/20	4/3/8
Prostate Cancer	12600	52/50	27/8

Ridge Bio-medical Dataset Repository [17] and are related to studies of human cancer, including: leukemia, colon tumor, breast and prostate cancer. For datasets that were not pre-divided into the training and testing parts, the 10-fold stratified cross-validation was applied[1]. Leave-one-out cross-validation was also considered however no significant influence on classification accuracy was observed. To ensure stable results, the average score of 10 runs is presented in all experiments.

The classification process for all algorithms was preceded by feature selection using the Relief-F [1] method which is common for microarray data analysis. In the first step, Relief-F draws instances at random and computes their nearest neighbors. Afterwards, Relief-F adjusts a feature weighting vector to give higher weight to those attributes which discriminate the instance from neighbors of different classes. The number of neighbors in Relief-F was equal to 10 and in order to improve the computation time, the number of selected attributes was arbitrary limited to the top 1000. Restriction for the number of attributes has no significant influence on classification accuracy however it speeds the algorithms up.

We have employed two alternative multi-tests mt_2 and mt_3 in addition to the primary test, mt_1, so the number of multi-tests analyzed in each non-terminal node, was equal to 3 ($W = 3$). Performed experiments show that employing a higher number of multi-tests, besides significant increase of the calculation time, did not yield any improvement in classification accuracy. To prevent data over-fitting, $C4.5$-like pessimistic pruning was applied.

3.2 Multi-Test Decision Tree Results

In Table 2, we compare the influence of the multi-test size on the accuracy. Results show that the number of univariate tests N used in a single multi-test has a significant impact on the classifier accuracy. The average score of the

[1] Pre-divided datasets were also tested with cross-validation but since the obtained performance was the same as with the original division into training and testing parts, due to lack of space, we report results with that original division only.

Table 2. A comparison of the *MTDT* accuracy under different numbers of tests in the multi-test

Dataset / Classifier	$MTDT\ N = 1$	$MTDT\ N = 5$	$MTDT\ N = 11$
Breast Cancer	**68.42**	57.89	57.89
Central Nervous System	60.50	72.17	**74.33**
Colon Tumor	80.40	**85.83**	83.92
DLBCL Standford	81.75	85.25	**86.60**
DLBCL vs Follicular Lymphoma	84.82	83.42	**85.42**
DLBCL NIH	51.25	60.00	**62.50**
Leukemia ALL vs AML	**91.17**	**91.17**	88.23
Leukemia MLL vs ALL vs AML	86.67	**100.00**	**100.00**
Prostate Cancer	26.47	**61.76**	44.11
Average score	70.16	77.50	75.89

multi-test with $N > 1$ was higher on most of the datasets. On only one dataset (Breast Cancer), the result of the multi-test algorithm was lower than expected, although the overall improvement is noticeable. We conjecture that the main cause of lower classification accuracy of the *MTDT* approach with $N = 1$ was due to under-fitted decision trees. It is worth emphasizing that the *MTDT* with a single one-attribute test in a node, $N = 1$, behaves similarly to the standard $C4.5$ algorithm. It was also observed that using too many genes in the multi-test may not only induce more complex rules but also over-fit learned trees to the training data.

In order to detect and exclude the possibility of over-fitting in the training phase of our method, we created artificial datasets which were copied from those listed in Table 1 where attributes were left exactly the same but class labels were randomly changed. This is usually referred to as the Y-randomization test [27]. The *MTDT* classification accuracy was significantly lower on randomized data than on original data and therefore this indicates that there is no evidence of over-fitting in our method.

Experiments performed on the Dual-Core CPU 1.66GHz machine with 2GB of RAM showed that the proposed solution is scalable and can manage large datasets. Average computation time on analysed datasets for increasing numbers of tests in the multi-test: $N = 1$, $N = 5$, and $N = 11$ was 2.8, 5.3 and 8.8 seconds correspondingly.

Leukemia MLL vs. ALL vs. AML Dataset. In one of our experiments, the dataset from Armstrong [2] was evaluated. Dataset describes the distinction between Leukemia MLL and other conventional ALL subtypes. There are a total of 57 3-class training samples (20 for ALL, 17 for MLL, and 20 for AML) and 15 test samples (4, 3, and 8 correspondingly). *MTDT* decision trees with $N = 1$ and $N = 5$ when evaluated on the training instances have the classification accuracy equal 100%. The actual trees are illustrated in Fig. 3. Although decision trees compared in this figure have the same performance on the training data, there is a significant difference in results on the testing instances. Table 3 shows the

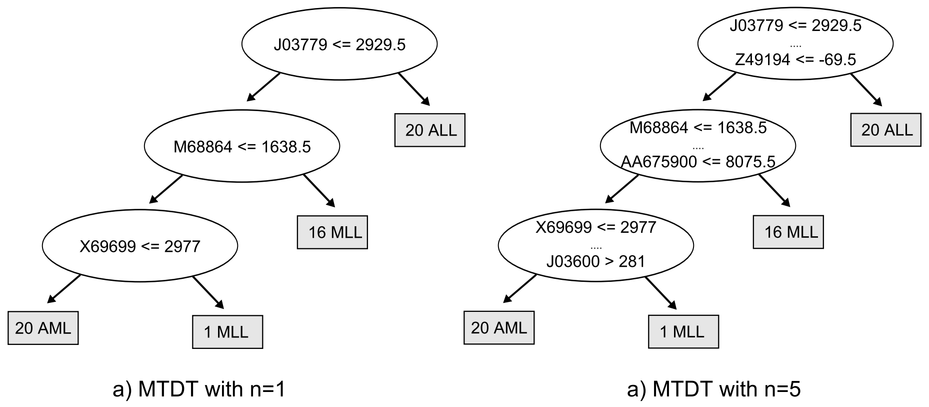

Fig. 3. Multi-Test decision trees with N=1 and N=5 tests in a single node

Table 3. Multi-Test Decision Tree with $N = 1$ and $N = 5$

$MTDT\ N = 1$			$MTDT\ N = 5$			
(a)	(b)	(c)	(a)	(b)	(c)	Classified as:
6	2	0	8	0	0	(a): AML
0	1	2	0	3	0	(b): MLL
0	2	2	0	0	4	(c): ALL
Accuracy 60%			Accuracy 100%			

confusion matrix for decision trees. In this experiment, decision trees with multi-test size $N = 1$ and $N = 5$ have the same structure, number of nodes and the same primary splitters. However, for other values of parameter N or different datasets this may not be the case. Differences in the tree structure may occur when alternative multi-tests outperform the mt_1 test or surrogate splits outvote the primary splitters. In spite of an equal tree size between $MTDT$ with $N = 1$ and $N > 1$, a larger number of univariate tests in a multi-test generates more complex nodes. Hopefully, the multi-tests contain only univariate tests which are easy to understand by human experts. For most datasets shown in Table 1, there is a relevant biological literature which identifies marker genes that are highly correlated with the class distinction. In order to evaluate whether the $MTDT$ results are biologically meaningful, we checked if discovered genes from our model match biological finding in the literature. The comparison showed that most of the genes from our $MTDT$ model were also identified in biological publications. For this particular dataset, 4 out of 5 genes that built $MTDT$ multi-test in the root node were also distinguished in article [2] and patent [13]. Attributes that built multi-tests in the lower parts of the $MTDT$ tree usually do not appear in biological publications as they distinguish only small sets of instances. We believe that $MTDT$ is capable of finding not only the most significant groups of marker genes but also low-ranked genes that when combined may also be meaningful.

Table 4. Comparison of classification accuracy

DT/CL	**AD**	**BF**	**J48**	**RF**	**CT**
Breast Cancer	42.10	47.36	52.63	68.42	68.42
Central Nervous System	63.33	71.66	56.66	75.00	73.33
Colon Tumor	74.19	75.80	85.48	75.80	75.80
DLBCL Standford	95.74	80.85	87.23	95.74	82.97
DLBCL vs Follicular Lymphoma	88.31	79.22	79.22	88.31	83.11
DLBCL NIH	50.00	60.00	57.50	52.50	62.50
Leukemia ALL vs AML	91.17	91.17	91.17	82.35	91.17
Leukemia MLL vs ALL vs AML	*	73.33	80.00	86.66	73.33
Prostate Cancer	38.23	44.11	29.41	29.41	44.11
Average score	67.88	69.28	68.81	72.68	72.75

3.3 Comparison of *MTDTs* to Other Classifiers

The comparison of *MTDTs* to other decision trees was also performed. The following classification algorithms were selected for this analysis:

1. *AD Tree* - alternating decision tree [12].
2. *BF Tree* - best-first decision tree classifier [22].
3. *J48 Tree* - pruned *C*4.5 decision tree [23].
4. *Random Forest* - algorithm constructing a forest of random trees [5].
5. *Simple Cart* - *CART* algorithm that implements minimal cost-complexity pruning [4].

The implementation of standard algorithms in the Weka package [26] was used in our evaluation. All classifiers, including the *MTDT* algorithm, were employed with default values of parameters on all datasets. The results are presented in Table 4. AD Tree can be applied only to binary class dataset therefore there are no results for *Leukemia MLL vs ALL vs AML* dataset.

Results in Tables 2 and 4 show that *MTDTs* with $N = 5$ tests in a single node yielded the best average accuracy, 77.50%, over all classification problems. However, the proposed *MTDT* method managed to achieve high accuracy whereas comprehensive decision rules were maintained via univariate tests used in multi-test splits. It is worth emphasizing that the *MTDT* with a single binary test in a node, i.e., $N = 1$, performed similarly to all remaining 'univariate test' methods. It can be compared to the J48 tree algorithm as they both use the gain ratio criterion. Their trees in most cases separated the training data perfectly, but performed considerably worse on testing instances. This may be caused by the under-fitted decision tree model. A slight increase in the number of tests in each split improved classification accuracy which can be observed in Table 2.

4 Conclusion and Future Directions

In this paper, we presented the multi-test decision tree approach to gene expression data classification. A new splitting criterion was introduced with the aim of reducing the under-fit of decision trees on these kind of data and improving classification accuracy. The experimental sections showed that our method led to competitive results as it outperformed the standard decision trees. Additionally, proposed method can be used with incomplete or noisy datasets since it uses internal surrogate tests. The preliminary comparison with the biological literature showed that decision trees learned by the $MTDT$ algorithm have biological interpretation. Therefore, biologists can benefit from using this "white box" approach as it builds accurate and biologically meaningful models for classification. In our future work, we are planning to apply $MTDT$ to solve the problem of missing values.

Even though, our results on the existing version of the algorithm and the current parameter tuning are promising, additional work on the influence of the test size, N, could yield an interesting insight into the behavior of our algorithm. Overall, we observed that the size, N, of the multi-test has significant impact on discovered rules and classification accuracy. We are working at the moment on the algorithm that through internal cross-validation could set this parameter automatically depending on training data. Another improvement concerns the pre-pruning mechanism that will reduce the size of the multi-test in lower parts of the tree. Our observations showed that the split subsets may have an incorrect size which can then increase the tree height and lead to data over-fit. We are planning also to look for adequate values of the percentage threshold b, which measures the similarity between surrogate tests and the primary splitter. We observed that replacing default settings with individually calculated values for each dataset could also improve classification results.

Acknowledgments. We thank Wojciech Kwedlo for reading this paper and providing constructive feedback. This work was supported by the grant S/WI/2/08 from Bialystok University of Technology.

References

1. Aldamassi, M., Chen, Z., Merriman, B., Gussin, D., Nelson, S.: A Practical Guide to Microarray Analysis of Gene Expression. UCLA Microarray Core & Nelson Lab, UCLA Department of Human Genetics (2001)
2. Armstrong, S.A.: MLL Translocations Specify a Distinct Gene Expression Profile that Distinguishes a Unique Leukemia. Nature Genetics 30, 41–47 (2002)
3. Berzal, F., Cubero, J.C., Marín, N., Sánchez, D.: Building multi-way decision trees with numerical attributes. Information Sciences 165, 73–90 (2004)
4. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group (1984)
5. Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001)
6. Brodley, C.E., Utgoff, P.E.: Multivariate Decision Trees. Machine Learning 19, 45–77 (1995)
7. Chen, X., Wang, M., Zhang, H.: The use of classification trees for bioinformatics. Wires Data Mining Knowl. Discov. 1, 55–63 (2011)
8. Dettling, M., Buhlmann, P.: Boosting for tumor classification with gene expression data. Bioinformatics 19(9), 1061–1069 (2003)

9. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
10. Dramiski, M., Rada-Iglesias, A., Enroth, S., Wadelius, C., Koronacki, J., Komorowski, J.: Monte Carlo feature selection for supervised classification. Bioinformatics 24(1), 110–117 (2008)
11. Fayyad, U.M., Irani, K.B.: On the Handling of Continuous-Valued Attributes in Decision Tree Generation. Machine Learning 8, 87–102 (1992)
12. Freund, Y., Mason, L.: The alternating decision tree learning algorithm. In: Sixteenth International Conference on Machine Learning, Bled, Slovenia, pp. 124–133 (1999)
13. Golub, T.R., Armstrong, S.A., Korsmeyer, S.J.: MLL translocations specify a distinct gene expression profile, distinguishing a unique leukemia, United States patent: 20060024734 (2006)
14. Grześ, M., Kretowski, M.: Decision Tree Approach to Microarray Data Analysis. Biocybernetics and Biomedical Engineering 27(3), 29–42 (2007)
15. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. In: Data Mining, Inference and Prediction, 2nd edn. Springer, Heidelberg (2009)
16. Hu, H., Li, J., Wang, H., Shi, M.: A Maximally Diversified Multiple Decision Tree Algorithm for Microarray Data Classification. In: I Workshop on Intelligent Systems for Bioinformatics, ACS (2006)
17. Kent Ridge Bio-medical Dataset Repository, `http://datam.i2r.a-star.edu.sg/datasets/index.html`
18. Li, J., Liu, H., Ng, S., Wong, L.: Discovery of significant rules for classifying cancer diagnosis data. Bioinformatics (19 suppl. 2), 93–102 (2003)
19. Murthy, S.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery 2, 345–389 (1998)
20. Rokach, L., Maimon, O.Z.: Data mining with decision trees: theory and application. Machine Perception Arfitical Intelligence 69 (2008)
21. Sebastiani, P., Gussoni, E., Kohane, I.S., Ramoni, M.F.: Statistical challenges in functional genomics. Statistical Science 18(1), 33–70 (2003)
22. Shi, H.: Best-first decision tree learning, MSc dissertation, University of Waikato (2007)
23. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
24. Tan, A.C., Gilbert, D.: Ensemble machine learning on gene expression data for cancer classification. Applied Bioinformatics 2(3), 75–83 (2003)
25. Tan, P.J., Dowe, D.L., Dix, T.I.: Building classification models from microarray data with tree-based classification algorithms. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 589–598. Springer, Heidelberg (2007)
26. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
27. Wold, S., Eriksson, L.: Statistical Validation of QSAR Results. In: van de Waterbeemd, H. (ed.) Chemometrics Methods in Molecular Design, VCH, pp. 309–318 (1995)
28. Yeoh, E.J., Ross, M.E.: Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell 1(2), 133–143 (2002)