Artificial Intelligence in Medicine 61 (2014) 35–44

Contents lists available at ScienceDirect

Artificial Intelligence in Medicine

journal homepage: www.elsevier.com/locate/aiim

Multi-test decision tree and its application to microarray data classification

Marcin Czajkowski [a,*], Marek Grześ [b], Marek Kretowski [a]

[a] Faculty of Computer Science, Bialystok University of Technology, Wiejska 45a, 15-351 Bialystok, Poland
[b] School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 24 June 2013
Received in revised form 11 January 2014
Accepted 30 January 2014

Keywords:
Decision trees
Univariate tests
Underfitting
Gene expression data

ABSTRACT

Objective: The desirable property of tools used to investigate biological data is easy to understand models and predictive decisions. Decision trees are particularly promising in this regard due to their comprehensible nature that resembles the hierarchical process of human decision making. However, existing algorithms for learning decision trees have tendency to underfit gene expression data. The main aim of this work is to improve the performance and stability of decision trees with only a small increase in their complexity.
Methods: We propose a multi-test decision tree (MTDT); our main contribution is the application of several univariate tests in each non-terminal node of the decision tree. We also search for alternative, lower-ranked features in order to obtain more stable and reliable predictions.
Results: Experimental validation was performed on several real-life gene expression datasets. Comparison results with eight classifiers show that MTDT has a statistically significantly higher accuracy than popular decision tree classifiers, and it was highly competitive with ensemble learning algorithms. The proposed solution managed to outperform its baseline algorithm on 14 datasets by an average 6%. A study performed on one of the datasets showed that the discovered genes used in the MTDT classification model are supported by biological evidence in the literature.
Conclusion: This paper introduces a new type of decision tree which is more suitable for solving biological problems. MTDTs are relatively easy to analyze and much more powerful in modeling high dimensional microarray data than their popular counterparts.

1. Introduction

Decision trees [1,2] are one of the most popular classification techniques in data mining and machine learning. Due to their comprehensible nature, they are particularly useful when the aim of modeling is to understand the underlying processes of the environment. Decision trees are also useful when the data do not satisfy the rigorous assumptions required by more traditional methods [3]. Tree-based classifiers can be successfully applied to solving biological problems [4–6]. Popular techniques for microarray data involve decision tree ensembles like random forest [7] and boosted decision trees [8]. However, existing attempts to apply decision trees to classification using gene expression data showed that single tree algorithms are not sufficient for inducing competitive classifiers [9,10].

In this paper, we tackle the problem of improving the performance of decision trees on gene expression data, with the constraint of preserving simplicity of decision trees. Standard techniques for improving the performance of classification algorithms, e.g., ensemble methods, do not satisfy this constraint when applied to decision trees because resulting classifiers become complex and almost impossible to understand [11,12]. We propose a multi-test approach to decision trees in which several univariate tests can be used to create a single splitting rule in every non-terminal node of the classification tree. We also search for alternative, lower-ranked features in order to obtain more stable and reliable predictions.

1.1. Gene expression data analysis

Cells represent basic organizational units of all living organisms. Each cell contains instructions for the creation of proteins and the regulation of processes in a living body. This collection of instructions is contained in the DNA. Each protein has a corresponding gene which can be seen as a recipe for how to create a given protein. If the gene is expressed, a corresponding protein will be produced [13]. A significant step in genomic research was the ability to monitor the expression level of genes in living cells. Specifically,

* Corresponding author. Tel.: +48 85 746 91 63; fax: +48 85 746 90 57.
E-mail address: m.czajkowski@pb.edu.pl (M. Czajkowski).

0933-3657/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.artmed.2014.01.005

cDNA microarray and high-density oligonucleotide chips allow the expression level of thousands of genes to be monitored simultaneously [14]. The outcome of these diagnostic tests is known as gene expression (or microarray) data.

Microarray data allows for numerous analyses of living organisms. The application of a mathematical apparatus and computations tools is indispensable here, since gene expression observations are represented by high dimensional feature vectors. The important questions are what kind of outcomes can be expected and what kind of questions can be answered using these tools. The answer comes from two fundamental approaches to mathematical modeling, which are equally important in the case of gene expression data. Scientific modeling attempts to understand the true model that is behind the data generated according to that model. In the case of gene expression data, it is concerned with problems of causal relationships between, for example, genes, or genes and proteins. Technological modeling has different aims. Here, the purpose is to build a model from past data that would be good at predicting future data regardless of whether the model is close to reality or not [15]. Discriminant analysis is an example of this kind of modeling in a general sense. It has also been widely used in post-genome cancer research studies [16,17].

Gene expression data poses many research challenges, and is not limited to research areas that are concerned with living organisms. This kind of data is also extremely challenging for computational tools and mathematical modeling [18]. Each observation is described by a high dimensional feature vector with a number of features that reach into the thousands, but the number of observations is rarely higher than 100. Therefore, this kind of data requires new computational tools to extract significant and meaningful rules, and some feature selection should be taken into account. Providing a group of most relevant genes may significantly improve classification performance [19].

1.2. Decision trees

Decision trees (also known as classification trees) represent one of the main techniques for discriminant analysis in data mining and knowledge discovery. They predict the class membership (dependent variable) of an instance using its measurements of predictor variables.

The most popular algorithms for decision tree induction are based on top-down greedy search [20]. First, the test attribute (and the threshold in the case of continuous attributes) is decided for the root node. Instances are split through the tree from the root node to a leaf node, which provides classification of a given instance. At each non-terminal node through which the instance passes, one (or more) attribute of the instance is tested and the instance is moved down to the branch that corresponds to an outcome of the test. The process is recursively repeated for each branch. When to stop partitioning and create a leaf node is still one of the major problems in the area.

Classification trees have many advantages that make them applicable in various scenarios, particularly when the data does not satisfy the rigorous assumptions required by more traditional methods. In this paper, the following facts are significant:

- learning of decision trees is fast, even with huge data sets, due to greedy search;
- classification is very fast, flexible, and allows for straightforward approaches to the problem of missing values;
- decision trees are easy to understand and analyze, as they reflect a hierarchical way of human decision making. They are thus the opposite of the 'black-box' approaches where model parameters are not understandable.

This introduction applies to cases in which tests in internal nodes of trees are based on one attribute. There are also algorithms which apply multivariate tests [21,22] based mostly on linear combination splits. Decision trees that allow the testing of multiple features at the node are potentially smaller than those limited to single univariate splits. Additionally, when only one attribute at each node is tested, it may cause replication of specific subtrees in the decision tree [23]. In effect, some features may be tested more than once in the decision tree. However, trees with simple tests are still desirable because experts can understand them. This fact is explicitly emphasized in the related literature. Brodley and Utgoff [24] say: "*A small tree with simple tests is most appealing because a human can understand it. There is a tradeoff to consider in allowing multivariate tests: using only univariate tests may result in large trees that are difficult to understand, whereas the addition of multivariate tests may result in trees with fewer nodes, but the test nodes may be more difficult to understand*". Our focus is therefore on univariate trees, since they are a 'white-box' technique, which makes them particularly interesting for scientific modeling. It is easy to find explanation for decisions of univariate classification trees.

1.3. Background and motivation

As stated in the previous section, decision trees with univariate splits are convenient. They are much easier to understand than trees with multivariate splits, and it is much easier to learn from the data. However, traditional algorithms, for example, C4.5 [25] or CART [26], fail to produce decision trees with high classification accuracy of gene expression data. Our previous work with various univariate decision tree algorithms showed that these algorithms produce considerably small trees that perfectly classify the training data but fail to classify unseen instances [10]. 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 [2]. 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 [10]. This indicates that the issue of split complexity could be advocated here, since not much can be gained from bigger univariate decision trees with this kind of data. This line of research is pursued in our paper.

We are motivated by the fact that univariate decision tree induction represents a white-box approach and improvements of such algorithms have considerable potential for genomic research and scientific modeling of underlying processes. Thus, our goal is to improve the classification accuracy of decision trees and imply more informative analysis of microarray data in a way that will make them still easy to understand. Decision trees with multivariate splits or bagging/boosting methods often outperform existing univariate algorithms on gene expression data [9,27,28]. However, those approaches generate complex rules that from a medical point of view are more difficult to understand and analyze. Our goal is to increase the complexity of univariate decision trees to the extent that makes them easy to understand and more competitive in terms of classification accuracy. We believe that the use of individual univariate splits may cause the classifier to underfit the learning data, since it leads to trees that are not robust enough and do not take information about other most relevant attributes into account. Our novel technique uses several univariate tests in each internal node to avoid these problems. As multi-test nodes are based on univariate tests, trees learned with this approach will be much easier to analyze than trees with classical multivariate splits.

In this paragraph, we attempt to justify why our approach is suitable for gene expression data and why this may lead to high classification accuracy. Gene expression data is characterized by a very high ratio of features to observations, which poses serious

problems for standard univariate splits. The learning algorithm can easily find a test that separates the training data very well at a given level in the tree, but this split can correspond to noise only. This situation is even more likely at intermediate and lower levels of the tree. For example, 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 finding a test likely, i.e., an attribute and its corresponding threshold, which can split this data perfectly. When there are only 10 observations in the node (even distribution), the number of possible splits is only 252, but the number of attributes is 3 orders of magnitude higher. When the split contains only one univariate test, there is a very high risk of choosing tests that correspond to noise. Thus, our approach is to have more univariate tests in each internal node and to base splitting decisions on a larger number of univariate tests not necessarily those tests that yield the highest value of the gain ratio [25] or the Gini index [26].

1.4. Related work

This paper addresses an issue of test complexity in decision trees. A standard approach in the case of discrete attributes is to associate a branch with each categorical attribute value. Another possibility is to group some attribute values in order to reduce the branching factor. When all values are grouped into two clusters, a binary tree is obtained (e.g., in CART [26]). In the case of continues attributes, binary splits are used. Here, the standard split compares the value of the attribute with a threshold and the outcome of such a comparison is binary. Thus, a straightforward way to reduce the tree complexity (in terms of the number of nodes) is to use multiple thresholds in each split on a 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. This approach was explored by Berzal et al. [29] who proposed multi-way decision trees using multi-way splits.

In [29], a hierarchical clustering of attribute values is combined with the standard greedy decision tree algorithm. Initially, each separate attribute value is treated as an individual interval, and the two most similar adjacent intervals are merged in each step. This process can be repeated until only two intervals are left; this would lead to a binary decision tree. However, the clustering process can be stopped before that. Each time two adjacent intervals are merged, the impurity measure associated with the decision tree is checked. The current interval set is determined according to the highest measure of impurity. This technique is similar to the splitting criterion used to evaluate alternative splits like the C4.5 gain ratio or the Gini index of diversity. The Berzal approach was not evaluated in terms of gene expression data, and, due to the nature of single attribute multi-way tests, it would not be sufficient to overcome the high ratio of features/observations in this kind of data.

The specific character of gene expression data and its influence on the process of building decision trees was investigated by Li et al. [30]. This solution focused on using committees of trees to aggregate the discriminating power of a bigger number of significant rules and to make more reliable predictions. First, all features are ranked according to the gain ratio. Then, 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 kth tree using the kth top-ranked feature is obtained. The classification of the final committee of k decision trees is governed by weighted voting. It was observed that:

- significant rules often contain features that 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 whole feature space;
- alternative trees often outperform or compete with the performance of the greedy tree.

This work also supports our decision to use many univariate tests in our multi-test decision tree induction algorithm. In particular, our aim is to make use of features that are globally low-ranked and use them jointly in multi-tests. However, our aim is to preserve the simplicity of final decision trees, which is not the case in [30].

Our previous work [10] in which standard decision trees were evaluated on gene expression data led us to the conclusion that the high ratio of variables/cases may cause the learning algorithm to be misled by randomly chosen dependencies in the training data. This may be disastrous for learned trees due to the hierarchical nature of the classification process of decision trees. Performed experiments revealed that the size of decision trees built with traditional classification methods, such as C4.5, is relatively small and does not capture all of the structure available in the data and is additionally misled by noise.

The rest of the paper is organized as follows. In Section 2, we introduce a novel representation for decision trees. Then, our algorithm that learns decision trees in the new representation is presented in Section 3. In Section 4, 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

This paper introduces multi-test decision trees (*MTDT*) – a new, richer language to represent a decision tree. The overall structure of a multi-test tree does not differ from a standard decision tree, e.g., C4.5 [25]. In multi-test tree, every split in non-terminal nodes is composed of a set of univariate tests and is called a multi-test split. These elementary tests are univariate and are combined in a way that shows our approach is substantially different from typical multivariate, e.g. oblique, splits.

During classification, the *MTDT* splitting criterion is directed by the majority voting mechanism where all univariate test components have the same importance.

Fig. 1 illustrates a multi-test with three individual attribute tests, $\{(f_1 \leq 2), (f_2 \leq 5), (f_3 \leq 8)\}$, that splits the data in the node into two subsets: *Class A* and *Class B*. In this particular example, as a result of the majority voting rule, at least 2 out of 3 univariate tests

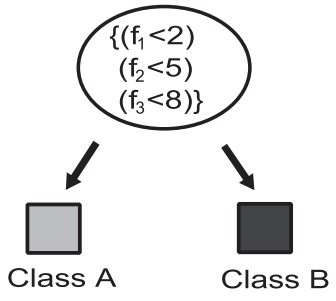

Fig. 1. An example of a multi-test split which contains a set of univariate tests.

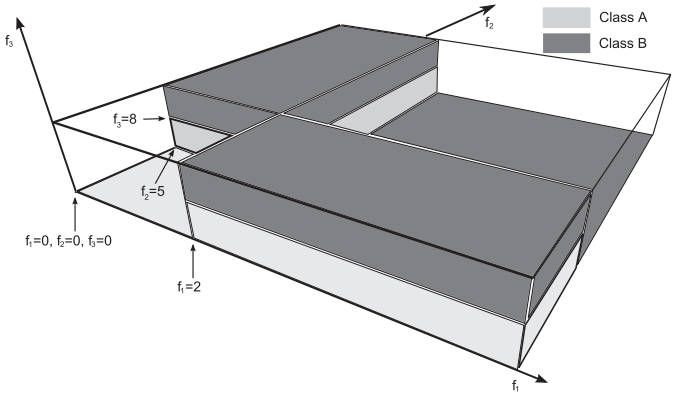

Fig. 2. Graphical representation of a multi-test split that contains 3 single attribute tests: $\{(f_1 \leq 2), (f_2 \leq 5), (f_3 \leq 8)\}$.

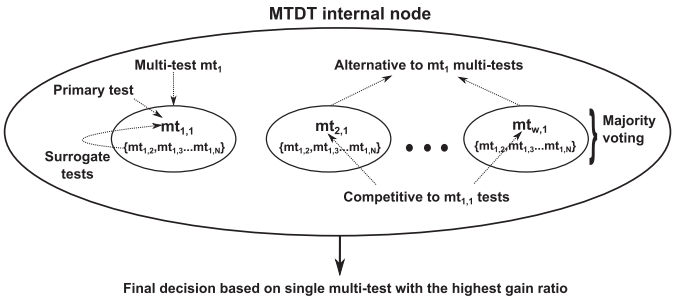

Fig. 3. An example search process which determines the best multi-test for a non-terminal node of multi-test decision tree (*MTDT*) from the set of potential multi-tests (*MT*).

determine the decision of the actual multi-test split at the node. The graphical representation of the multi-test example is shown in Fig. 2. Each test that uses feature f_i can split an instance space but only with a boundary that is orthogonal to the f_i axis. In our example, if $f_1 < 2$ and $f_2 < 5$, then regardless of the decision of f_3, the decision is *Class A* (light gray region). If $f_1 > 2$ and $f_2 > 5$, then regardless of f_3, the decision is *Class B* (dark gray region). If, on the other hand, f_1 and f_2 yield a contradiction, the final decision is determined by f_3 where $f_3 > 8$ leads to *Class B* and $f_3 < 8$ to *Class A*. Certainly, univariate tests can be evaluated in any order. The fact that only univariate tests are used in multi-test splits ensures that *MTDT* can be treated as a univariate decision tree despite more than one being used in each split.

3. Learning multi-test decision trees

The previous section introduced the idea of multi-test decision trees. In principle, this decision tree learning can take various approaches. In this section, we propose one particular method for learning trees that is based on greedy construction of multi-test splits. In what follows, it is assumed that decision trees learning uses top-down induction, where, at every level of recursion, the top-down algorithm constructs multi-test splits to be used in non-terminal nodes of the decision tree. The procedure to construct those multi-test splits is the core of this section and our contribution. It should be noted that multi-tests could also be used with other types of decision tree learning methods, i.e., algorithms that are not top-down. The concept of top-down induction was introduced in Section 1.2.

3.1. Building multi-test splits

Top-down decision tree learning algorithms have to choose a split (or terminate recursion and create a terminal node with a decision) at every level of recursion, given a subset X of training instances. For this reason, our procedure in Algorithm 1 takes X, the current set of instances, and returns the best multi-test for splitting instances in X. Note, that the cardinality of X is non-increasing with every recursive call of the top-down procedure. Additional parameter W, determines the number of alternative multi-tests that are considered by the procedure before the best multi-test is returned. Specifically, our procedure constructs a set $MT = \{mt_1, mt_2, \ldots, mt_W\}$ of W alternative multi-tests and returns the best one according to the gain ratio criterion (Line 9) in Algorithm 1.

Algorithm 1. Multi-test construction.

```
CreateMultiTest(X,W, N)
1: V ← create all candidate thresholds using X
2: best_primary = argmax_{v∈V} GainRatio(v,X)
3: mt_1 = BuildMulti-test(best_primary, V, X, N)
4: for i ∈ {2, ..., W} do
5:    MT = {mt_1, ..., mt_{i−1}}
6:    next_primary = NextPrimary(V, MT, X)
7:    mt_i = BuildMulti-test(next_primary, V, X, N)
8: end for
9: return arg max_{mti} GainRatio (mt_i, X)
```

The first step of algorithm determines the set of univariate tests for further evaluation. We only consider the relevant thresholds, called the candidate thresholds [31], which split instances from different classes. In existing algorithms with univariate tests, once the set of possible thresholds is computed (univariate tests correspond to thresholds when continues features are present), the best threshold is selected according to some priority measure (e.g., the gain ratio criterion), and the univariate test with highest evaluation is returned. This standard procedure would return the univariate test computed in Line 2 of the algorithm. Our algorithm does additional computation in order to build splits with multiple univariate tests.

Each ith multi-test is composed of no more than N univariate tests; the first one is called a *primary test* ($mt_{i,1}$), and all remaining $N-1$ tests are called *surrogate tests* $mt_{i,j}$ where $1 < j \leq N$. The parameter denoted as N represents the maximum number of univariate tests that constitute the multi-test. Fig. 3 illustrates tests that are considered in every execution of Algorithm 1.

The set $\{mt_1, \ldots, mt_W\}$ is constructed as follows. First, mt_1 is constructed in Line 3 using the primary univariate test found in Line 2. The actual multi-test is built in the BuildMulti-test function, which identifies which candidate tests should be used as surrogate tests of the primary test that is provided as a parameter. This step is explained in detail in Section 3.1.1. mt_1 is a special multi-test because its primary test is the best univariate test according to the priority measure. Primary tests for remaining multi-tests have to be selected in a way that would diversify created multi-tests. This process takes place in the NextPrimary function which is executed in Line 6 and described in Section 3.1.2. Once the primary test for a new multi-test is identified, the BuildMulti-test procedure can be used again (Line 7).

Because of the majority voting mechanism applied during classification, surrogate tests have a considerable impact on multi-test decisions because they can outvote the primary test. It should be noted that this impact can be positive and negative, and it affects the gain ratio of the entire multi-test. Therefore, it is possible that the best multi-test that will be returned by Algorithm 1 may not contain the original univariate test with highest gain ratio ($mt_{1,1}$).

This can happen when voting components of competitive multi-tests mt_i ($1 < i \leq W$) have higher gain ratio taken as a whole than mt_1 despite the fact that $mt_{1,1}$ is the univariate test with the highest gain ratio. This fact justifies our decision to use multi-test decision trees, since it can provide better and more robust against noise classification.

3.1.1. Multi-test construction

When function BuildMulti-test is executed, the primary test provided in the first parameter constitutes the first univariate test that will be included in the multi-test, and the goal of this function is add $N - 1$ surrogate tests. The reason for adding more tests is that applying a single primary test based on one attribute may cause the classifier to underfit the learning data due to low complexity of the classification rule. Surrogate tests should support the division of the training instances made by the primary test. In other words, the remaining tests (the surrogate tests) of the multi-test should, using the remaining features, branch the tree in a similar way to their primary test.

In order to determine surrogate tests, we have adopted a solution proposed in the *CART* system [26]. The use of the surrogate variable at a given split results in a similar node impurity measure. It also mimics the chosen split in terms of which and how many observations go to the corresponding branches. Therefore, the measure of similarity between the primary test and surrogates of the multi-test is given by the number of observations classified in the same way. The parameter b equals the percent of decisions made by surrogate tests that differ from the primary splitter. The parameter is described in more detail in the next section. In our method, we also consider tests that classify instances in an inverse (opposite) way to their primary test (high value of the parameter b). For such tests, we reverse the relation between attribute and interval midpoint, and recalculate the score. However, this only works if we a have binary classification problem.

3.1.2. Identifying additional primary tests

The NextPrimary function searches for a threshold that will be applied in a BuildMulti-test, which is executed to build multi-test mt_i for $k < i \leq W$ after the first $k \geq 1$ multi-tests are constructed.

Two factors should be taken into consideration when choosing the primary test for mt_i. First, new primary tests should be competitors to all existing primary tests. The competitor tests yield high gain ratio but are not as good as, e.g., the primary test $mt_{1,1}$ used to construct mt_1. A significant difference between these tests and surrogate tests is the way in which they are ranked. As shown in the previous subsection, surrogate tests are not evaluated by how much improvement they yield in reducing node impurity but rather on how closely they mimic the split determined by their primary test.

On the other hand, the competitor tests are ranked according to the highest gain ratio. We denote tests as competitor tests if their gain ratio is in the top q highest gain ratio values where primary tests used in mt_j for $j < i$ are not considered (the default value of q is 10). Performed experiments described in Section 4.2.3 show that using more competitors (high q value) leads to the selection of tests with low gain ratio; this decreases the power of alternative multi-tests. However, decreasing the number of competitor tests (low q) may cause new primary tests be too similar to those already selected.

The second factor that should be considered is that the same attribute is often listed as both a competitor, i.e., as one of the primary tests, and as a surrogate. This may lead to alternative multi-tests, mt_i, that contain similar or identical univariate tests and do not provide any comparable improvement. Therefore, competitor tests should be diversified in order to diversify the alternative multi-tests. For that reason, function NextPrimary receives the list of all multi-tests that were created before mt_i. The diversification problem is then solved by requiring that every new primary split must be a competitor for $mt_{1,1}$, i.e., for the primary test of the first multi-test (determined by the q value introduced in the previous paragraph), and it must also be the worst average surrogate (have the highest average value of parameter b) in all primary tests $mt_{k,1}$ where $k < i$.

To sum up: the surrogate tests are similar to the primary test; the competitor tests are those that have highest gain ratio and are different than all previously selected primary tests.

3.2. Multi-test size and prediction

The size of the multi-test, i.e., the maximum number of single tests that make every multi-test, has a strong impact on its performance and the 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, the 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 test.

The exact size of the multi-test depends on the difference between the primary 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 seen in the classical approach to univariate decision trees. To avoid discrepancies in the multi-test, surrogate tests should not be added to tests that do not have a proper substitute. An inappropriate set of surrogates may dominate the primary test and deteriorate the splitting criterion. Therefore, surrogate tests added to the multi-test should return no more than b% of decisions (default 10%) that differ from the primary test. Using $b = 0\%$ means that surrogate tests can only be added to the multi-test if they split observations in the exactly the same way as the corresponding primary splitter. In practice, setting b to 0% rejects almost all surrogates; therefore, it is equivalent to setting the size of multi-test N to 1. In this event, the decision tree would become similar to the tree generated by the *C*4.5 algorithm because only one attribute would be used in each multi-test. If the value of b is high then all $N - 1$ surrogates join the multi-test.

4. Experimental results

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

4.1. Setup

The performance of the *MTDT* classifier was investigated using publicly available microarray datasets described in Table 1. These datasets are from the Kent Ridge Bio-medical Dataset Repository [32] and are related to studies of human cancer, including leukemia, colon tumor, prostate cancer, lung cancer, breast cancer and lymphoma. For datasets that were not pre-divided into the training and testing parts, the 10-fold stratified cross-validation was applied. By the stratified cross-validation, we mean that each fold contains roughly the same proportion of instances with the same class labels. Leave-one-out cross-validation was also considered; however, no significant difference in results was observed with this type of cross-validation. The average score of 10 runs is presented for cross-validated data.

The classification process for all algorithms was preceded by feature selection using the Relief-F [33] method, which is common for microarray data analysis. In the first step, Relief-F draws instances at random and computes their nearest neighbors

Table 1
Kent Ridge bio-medical gene expression datasets used in experiments.

Dataset	Abbreviation	Attributes	Classes	Training set	Testing set
Breast Cancer	BC	24,481	2	34/44	12/7
Central Nervous System	CNS	7129	2	21/39	–
Colon tumor	CT	6500	2	40/22	–
DLBCL Stanford	DS	4026	2	24/23	–
DLBCL vs. Follicular Lymphoma	DF	6817	2	58/19	–
DLBCL NIH	DNH	7399	2	88/72	30/50
Leukemia ALL vs. AML	AML	7129	2	27/11	20/14
Leukemia MLL vs. ALL vs. AML	MLL	12,583	3	20/17/20	4/3/8
Lung Cancer Dana-Farber	LCD	12,600	5	139/21/20/6/17	–
Lung Cancer Brigham	LCB	12,533	2	16/16	15/134
Lung Cancer Univ. of Michigan	LCU	7129	2	86/10	–
Lung Cancer – Toronto, Ontario	LCT	2880	2	24/15	–
Ovarian Cancer NCI PBSII	OC	15,154	2	91/162	–
Prostate Cancer	PC	12,600	2	52/50	27/8

(default 10). Then, Relief-F adjusts a feature weighting vector to give higher weight to attributes that discriminate the instances from neighbors of different classes. The main benefits of using feature selection are shorter training times, improved model interpretability, and enhanced generalization by reducing overfitting. However, as we mentioned in previous sections, with univariate decision trees using microarray data, one faces the problem of underfitting to the learning data (overfitting is not significant). Hence, there is no need to improve the model interpretability because it is already simple; it is useful to retain a larger number of features and use a less aggressive feature selection. We tested different numbers of top ranked attributes/features 50, 100, 200, 1000, 2000 and also considered no feature selection at all. Reductions in the number of attributes to 200 have no significant influence on the test-sets accuracy of compared classifiers; however, it speeds up the training time of all algorithms. Our multi-test algorithm works well on test-sets without feature selection and those with larger numbers of features (200 and over). When the number of top selected attributes is small, *MTDT* loses its ability to find lower-ranked features (as they were excluded from the data), and its performance is similar to the rest of the tested decision trees. Therefore, the number of selected attributes was arbitrarily limited to the top 1000 to allow *MTDT* to find low-ranked features.

A statistical analysis of all obtained results was performed using the Friedman test and the corresponding Dunn's multiple comparison test (significance level equal to 0.05) recommended by Demsar [34].

4.2. Multi-test decision tree results

4.2.1. Multi-test size

The influence of the multi-test size on the performance of our method was experimentally verified on real gene expression data. Classification algorithms applied to these kinds of data are more likely to underfit because of a small ratio of the number of observations to the amount of attributes. The performance of the *MTDT* classifier was studied with six different values of parameter *N*, which stands for the maximum number of univariate tests in the multi-test. 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. Both algorithms use the gain ratio criterion and pessimistic pruning. There is, however, a slight difference in calculating the exact threshold value; this is described in Section 3.1.

In Table 2, we compare the influence of the multi-test size on accuracy. In all experiments, 1000 attributes were considered, and the algorithm's parameters had their default values of $W = 3$ and $b = 10\%$. These results revealed that the number of univariate tests used in a single multi-test has a significant impact on the classifier accuracy. According to the Friedman test, there is a statistically significant difference (p-value of 0.0003) in the accuracy of all versions. Based on Dunn's Multiple Comparison Test Difference, there is a statistically significant difference in classification quality between the number of tests in the multi-test, *N*, equal to 1, and 7, 9, and 11.

Experimental validation performed on 14 datasets showed that the average accuracy of the multi-test algorithms increased over 3% when $N = 3$, and over 6% when $N = 7$, compared to the base *MTDT* with $N = 1$. On only one dataset (*BC*), the result of the multi-test algorithm was lower than expected, although the overall improvement is noticeable. The reason why results for *BC* were better for $N = 1$ lies in the number of attributes that distinguish classes. For this dataset, only a few genes are considered as markers; therefore, a higher number of surrogates could decrease the *MTDT* accuracy when the tree is overfit.

Considering the results, we conjecture that the underfitting is the main cause of lower classification accuracy of the *MTDT* approach with $N = 1$. Decision trees obtained by standard (single univariate test in a node) algorithm are not complex enough. It was also observed that using too many genes in the multi-test may induce more complex rules and overfit learned trees to the training data.

In order to detect and mitigate the possibility of overfitting in the training phase of our method, we created artificial datasets that were copied from those listed in Table 1; attributes were left exactly the same, but class labels were randomly changed. This technique

Table 2
A comparison of the multi-test decision tree (*MTDT*) accuracy under different numbers of tests (*N*) in the multi-test. Datasets abbreviations are used (Table 1). The highest classifiers accuracy for each dataset was bolded.

Dataset	$N = 1$	$N = 3$	$N = 5$	$N = 7$	$N = 9$	$N = 11$
BC	**68.42**	63.15	57.89	52.63	57.89	57.89
CNS	60.50	71.33	**72.17**	72.00	72.17	**74.33**
CT	80.40	83.14	85.83	**85.97**	85.83	83.92
DS	81.75	85.00	85.25	85.55	85.05	86.60
DF	84.82	82.07	83.42	85.01	**85.57**	85.42
NIH	51.25	60.00	60.00	62.50	**63.75**	62.50
AML	**91.18**	85.29	**91.18**	**91.18**	**91.18**	88.23
MLL	86.67	73.33	**100.00**	**100.00**	93.33	**100.00**
LCD	89.41	90.98	91.60	**92.12**	91.15	90.96
LCB	88.59	95.97	96.64	97.98	**98.66**	**98.66**
LCU	97.48	98.04	98.32	98.93	99.78	**100.00**
LCT	61.42	61.66	63.67	**66.83**	65.67	62.16
OC	97.04	**98.69**	98.02	98.34	98.34	98.18
PC	26.47	58.82	**61.76**	**61.76**	47.06	44.11
Average	76.10	79.11	81.83	**82.20**	81.20	80.93

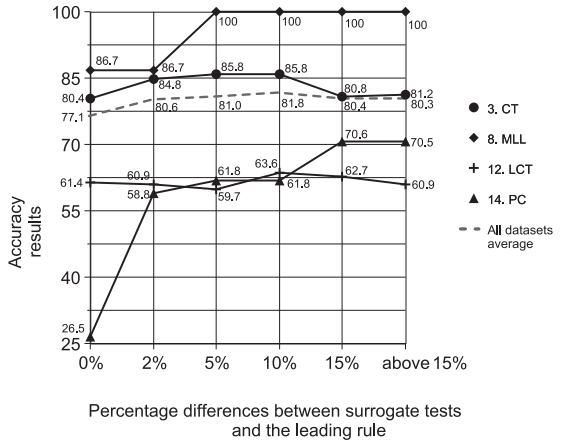

Fig. 4. The influence of the similarity measure *b*, on the classification accuracy of the multi-test decision tree (*MTDT*) algorithm.

is usually referred to as the Y-randomization test [35]. The *MTDT* classification accuracy was significantly lower on the randomized data than on original data (which is good in this situation); this indicates that there is no evidence of overfitting in our method.

4.2.2. Surrogate tests

In Section 3.2, it was explained that surrogate tests, should not differ more than *b*% from primary tests. Performed experiments suggest that surrogate tests added to the multi-test should not differ from the primary test by more than 10%. We consider this a default value in all datasets; however, adequate setting of this parameter may improve classification accuracy. Fig. 4 presents the influence of the similarity parameter, *b*, on the performance of the *MTDT* classifier.

In this figure, $b = 0\%$ means that only surrogates that have the same gain ratio as primary tests are accepted (it is almost equivalent to setting $N = 1$), and a high value of b (in the figure above 15%) means that all $N - 7$ surrogates join the multi-test. Although, a general average on all 14 datasets has the highest accuracy when $b = 10\%$, we may observe that the optimal value of this parameter is different for specific datasets. The score of the *MTDT* algorithms on leukemia (*MLL*) and prostate cancer (*PC*) datasets increases significantly when there is no restriction on choosing surrogate tests. However, in other datasets, when the surrogate tests are more

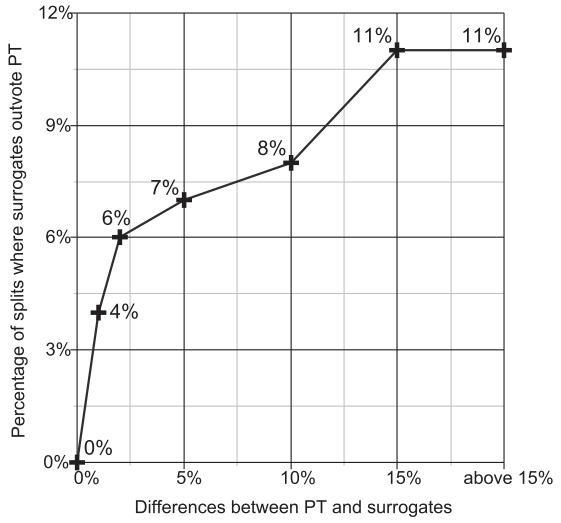

Fig. 5. The influence of the similarity measure *b*, on the decision split in the tree node.

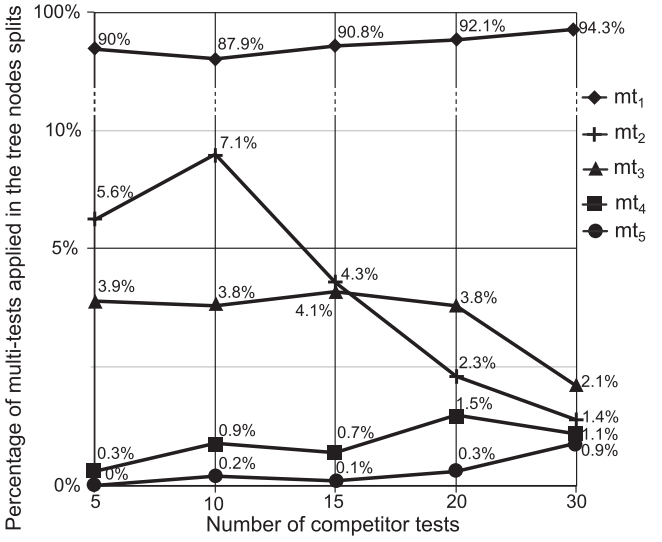

Fig. 6. The influence of the number of competitor tests *q*, on the application of multi-tests as a splitting criterion in the tree node.

similar to the primary test, the results are slightly better. An additional comparison of the *MTDT* performance with a simple baseline based on random selection showed the significant difference in prediction accuracy in favor of proposed solution.

Fig. 5 presents the impact of surrogates on the decision split. It illustrates the percentage number of splits on the testing data for which primary tests were outvoted by their surrogates. We may observe that for all datasets, for the defined value of parameter *b* equal 10%, the average percentage of splits contrasting the primary test is equal 8%. This impact of surrogate tests, together with alternative multi-tests, improves the *MTDT* average accuracy up to 6%, from 76.1% to 82.2%.

4.2.3. Alterative multi-tests

The parameters of alternative multi-tests were defined empirically through extensive experiments. Fig. 6 presents the average influence of the number of competitor tests *q* on the performance of alternative multi-tests on all datasets. We can observe what percent of multi-tests were applied as a splitting criterion in the tree node. It is not surprising that the tree node splits were mostly determined by the multi-tests (mt_1) that were built on the primary tests. However, for the default value of the parameter $q = 10$, over 12% of all splits were made in accordance to the alternative multi-tests mt_i $(1 < i \leq W)$.

In experiments, we employed two alternative multi-tests mt_2 and mt_3, so the number of multi-tests analyzed in each non-terminal node was equal to 3 ($W = 3$). Additional 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.

4.2.4. Leukemia MLL vs. ALL vs. AML dataset

In one of our experiments, the dataset from Armstrong [36] was evaluated in more detail. The dataset describes the distinction between leukemia MLL and other conventional ALL subtypes. There are a total of 57 three class training samples (20 for ALL, 17 for MLL, and 20 for AML) and 15 test samples (4, 3, and 8 correspondingly). The *MTDT* decision trees with $N = 1$ and $N = 7$, when evaluated on the training instances splits data exactly the same way and for both values of *N*, the classification accuracy is 100%. The actual trees are illustrated in Fig. 7, and the confusion matrix is presented in Table 3.

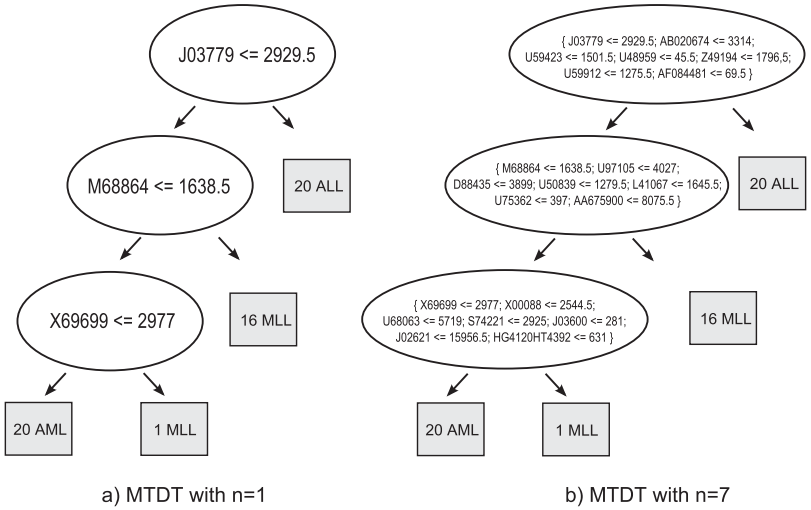

Fig. 7. Multi-test decision tree (*MTDT*) with $N = 1$ and $N = 7$ tests in a single node.

We can observe that although both induced trees have the same structure and classified instances from the training set, their performances on the test set were significantly different. Because both trees have identical primary tests, even when there is no impact on alternative multi-tests, this is a very good example of the strength of the proposed solution. The reason for such a good performance of *MTDT* with $N = 7$ in this example can be explained by the impact of surrogates on the multi-test decision. In 6 out of 15 instances, the surrogate tests $mt_{1,j}$ $(1 < j \leq N)$ in *MTDT* with $N = 7$ have to outvote the primary tests $mt_{1,1}$ in the nodes and correctly classify the instances. In this way, we have improved the classification accuracy for the Armstrong dataset from 86% to 100%.

4.3. Comparison of MTDT to other classifiers

The comparison of *MTDT* to other classifiers was also performed. The following classification algorithms were selected for this analysis:

- Decision trees:
  1. AD Tree (*AD*) – alternating decision tree [38].
  2. BF Tree (*BF*) – best-first decision tree classifier [39].
  3. J48 Tree (*J48*) – pruned C4.5 decision tree [25].
  4. Simple C&RT (*CT*) – version of the C&RT algorithm that implements minimal cost-complexity pruning [26].
- Decision rule classifiers:
  1. JRip (*JR*) – rule learner – repeated incremental pruning to produce error reduction (*RIPPER*) [40].
- 'Black box' meta decision trees:
  1. Random forest (*RF*) – algorithm constructing a forest of random trees [41].
  2. Bagging (*BG*) – reducing variance meta classifier [42].
  3. Adaboost (*ADA*) – boosting algorithm using Adaboost M1 method [43].

Table 3
Results for multi-test decision tree (*MTDT*) with $N = 1$ and $N = 7$ on dataset Leukemia MLL vs. ALL vs. AML.

MTDT $N = 1$			MTDT $N = 7$			Classified as:
(a)	(b)	(c)	(a)	(b)	(c)	
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%			

It is worth noting that besides the 'white box' classifiers, results on meta decision trees are also included. Those methods can generate more complex decision rules and outperform standard approaches. The resulting classifiers are, however, more difficult to understand. Our results show that the proposed *MTDT* algorithm that uses simple, univariate tests is highly competitive with 'black box' solutions.

The implementation of competitive algorithms in the Weka package [44] 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.

Results in Tables 2 and 4 show that *MTDT* with $N = 7$ tests in a single node yielded the best average accuracy: 82.20%, in all classification problems. In general, it can be observed that more complex methods like *RF*, *ADA*, and *BG* performed better than standard non-ensemble algorithms, which generate simpler solutions. The proposed *MTDT* method managed to achieve high accuracy, but comprehensive classification rules were maintained via the univariate tests used in multi-test splits. According to the Friedman test, there is a statistically significant difference (p-value of 0.0215) between tested classifiers. Based on Dunn's Multiple Comparison Test Difference, there is a statistically significant difference in terms of quality between the *MTDT* (with $N = 7$), and *BF* and *j48* trees. The

Table 4
Comparison of classification accuracy of algorithms: AD Tree (AD), BF Tree (BF), J48 Tree (J48), Simple CART (CT), JRip (JR), Random forest (RF), Bagging (BG), Adaboost (ADA).

Dataset	AD	BF	J48	CT	JR	RF	BG	ADA
BC	42.10	47.36	52.63	68.42	73.68	68.42	63.15	57.89
CNS	63.33	71.66	56.66	73.33	65.00	75.00	71.66	75.00
CT	74.19	75.80	85.48	75.80	74.19	75.80	79.03	79.03
DS	95.74	80.85	87.23	82.97	74.46	95.74	87.23	89.36
DF	88.31	79.22	79.22	83.11	77.92	88.31	85.71	90.90
NIH	50.00	60.00	57.50	62.50	61.25	52.50	58.75	65.00
AML	91.17	91.17	91.17	91.17	94.11	82.35	94.11	91.17
MLL	[a]	73.33	80.00	73.33	66.66	86.66	100.00	66.66
LCD	[a]	89.65	91.62	88.17	90.14	92.11	90.64	78.32
LCB	81.87	89.65	81.87	81.87	95.97	93.28	82.55	81.87
LCU	96.87	96.87	98.95	96.87	93.75	98.95	97.91	96.87
LCT	69.23	61.53	58.97	58.97	64.10	66.66	61.53	69.23
OC	99.60	98.02	97.23	98.02	98.81	98.02	97.62	99.20
PC	38.23	44.11	29.41	44.11	32.35	29.41	41.17	41.17
Average	74.22	75.66	74.85	77.05	75.89	78.80	79.36	77.26

[a] AD can be applied to data with two classes only.

AD classifier was excluded from statistical analysis, as it could not be applied to a multi-class dataset.

5. Discussion

In some cases, multi-test trees could be treated as a consistent representation of traditional univariate decision trees, but it works in one way only. A *MTDT* tree can be transformed into a traditional decision tree, but it is usually impossible to do it the other way round. Furthermore, even if our formulation of multi-test decision trees and traditional univariate trees were isomorphic (but they are not as we explained above), this would not invalidate our research because we show another representation that is more suitable (according to our results) for the greedy search, which is traditionally employed for learning decision trees. A similar relationship exists between decision trees and decision rules. Even though the hypothesis space of decision rules is a superset of the hypothesis space of decision trees, researchers still investigate decision trees because of various advantages that decision trees can offer.

The importance of particular types of tests that are used to build decision trees may depend on the type of search. Results presented in our paper show that standard greedy top-down learning of decision trees can be significantly improved using multi-test splits. If it would be possible to learn the optimal decision trees for a given test representation (which is infeasible on real-life data because the problem is NP-hard) instead of using a greedy algorithm, then one could check, for example, the imfluence of single and multi-test splits on the exact algorithm. It is likely that single test splits would be more competitive using alternative search strategies but at the same time multi-test splits could lead to further improvements. The current state-of-the-art in decision tree learning uses greedy search in most academic research and industrial applications; thus, our multi-test splits improve learning with that most important type of search. This fact explains, for example, why single test splits in Fig. 7 were weaker than multi-test splits. Multi-test splits were simply more convenient for top-down learning and better trees could be learned. In theory, better single test trees could potentially be obtained for the example in Fig. 7; however, but assuming that such trees exist and could be found, a different search algorithm or special tuning of existing algorithms would be required to find them. The same advantage of multi-test splits was observed on other datasets evaluated in this paper. Theoretically, these observations can be explained using the concept of 'inductive bias' in machine learning, that is, the need to make explicit or implicit assumptions about what kind of model is wanted for a particular problem [46].

In the experiment on the leukemia MLL vs. ALL vs. AML dataset, decision trees with multi-test size $N = 1$ and $N = 7$ have the same structure the same number of nodes. 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 multi-test mt_1 or surrogate tests outvote the primary test. 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.

To the most of datasets described in Table 1 biologists have found and published some marker genes that are highly correlated with class distinction. In order to evaluate whether the *MTDT* results are biological meaningful or not, we explored whether discovered genes from classifier's model are supported by biological evidence in the literature. Our research showed that most of the genes from the *MTDT* model were also identified in biological publications. For this particular dataset, six out of seven genes that built *MTDT* multi-test in the root node were also referred to in article [36] and patent [37]. Attributes that built multi-tests in the lower parts of the *MTDT* tree usually do not appear in publications as they represent only a 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 may also be meaningful when combined.

In the comparison of *MTDT* to other classifiers, 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 *J*48 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 underfitted decision tree model. A slight increase in the number of tests in each split improved the classification accuracy; this can be observed in Table 2. The experimental sections showed that the proposed method leads to highly competitive results. In our tests, *MTDT* outperformed classical decision trees and decision rule classifiers and was highly competitive with more powerful meta learning algorithms.

Even though several interesting questions from the machine learning point of view are still open, we are convinced that the existing version of the algorithm reported in this paper offers a useful tool for molecular biologists doing exploratory analysis of gene expression data. In their work, biologists rarely rely on out of the box solutions, and tuning algorithm's parameters is their normal practice. Therefore, our existing *MTDT* algorithm is a perfect tool for their experiments. By changing the number of components in the multi-tests splits of the *MTDT*, the biologist can obtain a monotone range of decision trees that start with trees corresponding to C4.5 (where one attribute is tested in every node) and proceed by having higher numbers of tests. A known phenomenon in molecular biology is that there often exist groups of genes (or features in general) that behave in a similar way. Biologists call it 'epistasis' [45]. For example, a specific substance, such as melanin, which is produced in living organism, may require several compounds in which each compound is produced by its corresponding gene and compounds that require other compounds in order to be produced. If any of the genes are defective, its compound will not be produced and neither will the substance. A similar phenomenon exists when the features used for data analysis are motifs, i.e., small sequences of DNA. When the numbers of occurrences of motifs are similar in the same piece of the DNA sequence, then features corresponding to motifs become similar. Our idea of identifying surrogate tests relates to this biological phenomenon, and it can identify these relationships exactly.

6. Conclusion

In this paper, we presented a multi-test decision tree approach to gene expression data classification. A new splitting criterion was introduced with the aim of reducing the underfit of decision trees on these kinds of data and improving classification accuracy. The proposed solution outperforms, or was highly competitive with, all tested competitors. Evaluation on real microarray data showed that knowledge discovered by *MTDT* is supported by biological evidence in the literature. Therefore, biologist can benefit from using this 'white box' approach, as it can build accurate and biologically meaningful models for classification and reveal new regularities in biological data. From the machine learning point of view, our rigorous empirical analysis revealed and evaluated the important algorithmic properties of our method.

The standard practical question left open is the autonomous tuning of the test size, N, to particular data. We observed that a

data-specific selection of N can significantly improve the performance of our method, although a general domain independent value was enough to obtain better results than that which existing algorithm can achieve. We are currently working on an algorithm that, through internal cross-validation, can set this parameter automatically for particular 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 analysis showed that the split subsets may have an incorrect size, which can then increase the tree height and lead to data overfit.

Acknowledgments

Authors thank Wojciech Kwedlo for reviewing the paper and providing constructive feedback. This work was supported by the grant S/WI/2/13 and W/WI/1/2013 from Bialystok University of Technology. The second author was supported by a fellowship from the Ontario Ministry of Research and Innovation.

References

[1] Murthy SK. Automatic construction of decision trees from data: a multi-disciplinary survey. Data Mining and Knowledge Discovery 1997;2:345–89.
[2] Rokach L, Maimon O. Data mining with decision trees: theory and applications. Machine perception and artificial intelligence, vol. 69. Singapore: World Scientific Publishing; 2008.
[3] Hastie T, Tibshirani R, Friedman JH. The elements of statistical learning. Data mining, inference and prediction. 2nd ed. New York: Springer; 2009.
[4] Che D, Liu Q, Rasheed K, Tao X. Decision tree and ensemble learning algorithms with their applications in bioinformatics. Software tools and algorithms for biological systems. Advances in Experimental Medicine and Biology 2011;696:191–9.
[5] Chen X, Wang M, Zhang H. The use of classification trees for bioinformatics. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2011;1:55–63.
[6] Czajkowski M, Kretowski M. Top scoring pair decision tree for gene expression data analysis. In: Arabnia HR, Tran Q-N, editors. Software tools and algorithms for biological systems. Advances in experimental medicine and biology, 696. 2011. p. 27–35.
[7] Diaz-Uriarte R, Alvarez de Andres S. Gene selection and classification of microarray data using random forest. BMC Bioinformatics 2006;7:3.
[8] Qu Y, Adam BL, Yasui Y, Ward MD, Cazares LH, Schellhammer PF, et al. Boosted decision tree analysis of surface-enhanced laser desorption/ionization mass spectral serum profiles discriminates prostate cancer from noncancer patients. Clinical Chemistry 2002;48:1835–43.
[9] Ge G, Wong GW. Classification of premalignant pancreatic cancer mass-spectrometry data using decision tree ensembles. BMC Bioinformatics 2008;9:275.
[10] Grześ M, Kretowski M. Decision tree approach to microarray data analysis. Biocybernetics and Biomedical Engineering 2007;27(3):29–42.
[11] Dettling M, Buhlmann P. Boosting for tumor classification with gene expression data. Bioinformatics 2003;19(9):1061–9.
[12] Tan AC, Gilbert D. Ensemble machine learning on gene expression data for cancer classification. Applied Bioinformatics 2003;2(3):75–83.
[13] Kuo WP, Kim E, Trimarchi J, Jenssen T, Vinterbo SA, Ohno-Machado L. A primer on gene expression and microarrays for machine learning researchers. Journal of Biomedical Informatics 2004;37:293–303.
[14] Brown PO, Botstein D. Exploring the new world of the genome with DNA microarrays. Nature Genetics 1999;21:33–7.
[15] Cowell RG, Dawid AP, Lauritzen SL, Spiegelhalter DJ. Probabilistic networks and expert systems: exact computational methods for Bayesian networks. International Statistical Review 2008;76:306–7.
[16] Golub TR, Slonim DK. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 1999;286(5439):531–7.
[17] Yeoh EJ, Ross ME. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell 2002;1(2):133–43.
[18] Sebastiani P, Gussoni E, Kohane IS, Ramoni MF. Statistical challenges in functional genomics. Statistical Science 2003;18:33–70.
[19] Draminski M, Rada-Iglesias A, Enroth S, Wadelius C, Koronacki J, Komorowski J. Monte Carlo feature selection for supervised classification. Bioinformatics 2008;24(1):110–7.
[20] Rokach L, Maimon O. Top-down induction of decision trees classifiers – a survey. IEEE Transactions on Systems, Man, and Cybernetics – Part C 2005;35(4):476–87.
[21] Brown DE, Pittard CL, Park H. Classification trees with optimal multivariate decision nodes. Pattern Recognition Letters 1996;17:699–703.
[22] Murthy S, Kasif S, Salzberg S. A system for induction of oblique decision trees. Journal of Artificial Intelligence Research 1994;2:1–33.
[23] Pagallo G, Haussler D. Boolean feature discovery in empirical learning. Machine Learning 1990;5:71–99.
[24] Brodley CE, Utgoff PE. Multivariate decision trees. Machine Learning 1995;19:45–77.
[25] Quinlan R. C4.5: programs for machine learning. San Mateo, CA, USA: Morgan Kaufmann; 1993.
[26] Breiman L, Friedman J, Olshen R, Stone C. Classification and regression trees. Belmont, CA, USA: Wadsworth International Group; 1984.
[27] Tan PJ, Dowe DL, Dix TI. Building classification models from microarray data with tree-based classification algorithms. In: Orgun MA, Thornton J, editors. AI 2007. Lecture notes in artificial intelligence, vol. 4830. Berlin Germany: Springer; 2007. p. 589–98.
[28] Hu H, Li J, Wang H, Shi M. A maximally diversified multiple decision tree algorithm for microarray data classification. In: Boden M, Bailey TL, editors. WISB 2006, vol. 73. Darlinghurst, Australia, Australia: Australian Computer Society, Inc.; 2006. p. 35–8.
[29] Berzal F, Cubero JC, Marin N, Sanchez D. Building multi-way decision trees with numerical attributes. Information Sciences 2004;165:73–90.
[30] Li J, Liu H, Ng S, Wong L. Discovery of significant rules for classifying cancer diagnosis data. Bioinformatics 2003;19(2):93–102.
[31] Fayyad UM, Irani KB. On the handling of continuous-valued attributes in decision tree generation. Machine Learning 1992;8:87–102.
[32] Kent Ridge bio-medical dataset repository; 2012. http://datam.i2r.a-star.edu.sg/datasets/krbd/index.html (accessed 20.12.12).
[33] Robnik-Siikonja M, Kononenko I, Theoretical. Empirical analysis of relief and relief. Machine Learning 2003;53:23–69.
[34] Demsar J. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 2006;7:1–30.
[35] Wold S, Eriksson L, Clementi S. Statistical validation of QSAR results. Chemometrics methods in molecular design, vol. 5. Weinheim, Germany: Wiley-VCH Verlag GmbH; 2008. p. 309–38.
[36] Armstrong SA. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. Nature Genetics 2002;30:41–7.
[37] Golub TR, Armstrong SA, Korsmeyer SJ. MLL translocations specify a distinct gene expression profile, distinguishing a unique leukemia. United States patent 20060024734; 2006.
[38] Freund Y, Mason L. The alternating decision tree learning algorithm. In: 16th international conference on machine learning ICML99, Bled, Slovenia. San Fransisco, CA, USA: Morgan Kaufmann; 1999. p. 124–33.
[39] Shi H. Best-first decision tree learning. University of Waikato; 2012 (Master's thesis), http://researchcommons.waikato.ac.nz/bitstream/handle/10289/2317/thesis.pdf (accessed 20012.12).
[40] Cohen WW. Fast effective rule induction. In: 12th international conference on machine learning ICML95. San Francisco, CA, USA: Morgan Kaufmann; 1995. p. 115–23.
[41] Breiman L. Random forests. Machine Learning 2001;45(1):5–32.
[42] Breiman L. Bagging predictors. Machine Learning 1996;24(2):123–40.
[43] Freund Y, Schapire RE. Experiments with a new boosting algorithm. In: 13th international conference on machine learning ICML96. 1996. p. 148–56.
[44] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten HI. The WEKA Data Mining Software: an update. ACM SIGKDD explorations newsletter 2009;11(1):10–8.
[45] Cordell HJ. Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. Human Molecular Genetics 2002;11(20):2463–8.
[46] Shalev-Shwartz S, Ben-David S. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press; 2014.