
Quality Technology & Quantitative Management
Vol. 11, No. 1, pp. 133-147, 2014

QTQM

Efficient Construction of Decision Trees by the Dual Information Distance Method

Irad Ben-Gal[1,*], Alexandra Dana[2], Niv Shkolnik[1] and Gonen Singer[3]

[1]Department of Industrial Engineering, Tel-Aviv University, Israel
[2]Department of Biomedical Engineering, Tel-Aviv University, Israel
[3]Department of Industrial Engineering, Afeka College of Engineering, Israel
(*Received* July 2013, *accepted* November 2013)

Abstract: The construction of efficient decision and classification trees is a fundamental task in Big Data analytics which is known to be NP-hard. Accordingly, many greedy heuristics were suggested for the construction of decision-trees, but were found to result in local-optimum solutions. In this work we present the *dual information distance* (DID) method for efficient construction of decision trees that is computationally attractive, yet relatively robust to noise. The DID heuristic selects features by considering both their immediate contribution to the classification, as well as their future potential effects. It represents the construction of classification trees by finding the shortest paths over a *graph* of partitions that are defined by the selected features. The DID method takes into account both the *orthogonality* between the selected partitions, as well as the *reduction of uncertainty* on the *class partition* given the selected attributes. We show that the DID method often outperforms popular classifiers, in terms of average depth and classification accuracy.

Keywords: Average path length, Big-Data, C4.5, decision trees, online classifiers.

1. Introduction

The construction of efficient decision trees is one of the fundamental problems in data mining [5]. Decision trees or classification trees (we use the two terms interchangeably) represent a simple and comprehensible way to describe a decision making process that relies on past knowledge. Although decision trees are not necessarily the most accurate classifiers, they provide a way for an "online" classification that is important when only part of the features are used to classify new instances. The construction of decision trees often uses similarity-based metrics in order to classify newly introduced instances to one out of the predefined classes. However, construction of the optimal decision tree is known to be an NP-hard task [7]. To deal with this constraint, many heuristics have been suggested, which generally are greedy by nature, to build classification-tree models in a linear or close to linear time complexity.

Most of these methods are recursive and use a top-down approach: at each stage of the tree construction they often choose the best attribute with respect to some predefined optimality criterion, which evaluates the attribute's "potential contribution" for a successful classification. Some of these popular greedy algorithms (*e.g.,* ID3 and C4.5) have been shown to perform quite well in practical problems [13], yet they do suffer from the usual flaws of greedy approaches, such as convergence to local optima. In order to reduce the "greediness effects", other heuristics suggest implementing a *look-ahead* approach by

* Corresponding author. E-mail: bengal@tau.ac.il

considering more than a single attribute at some of the construction stages. The obtained trees are called *k-steps look-ahead* trees when considering the best among all possible combinations of k attributes. However, the look-ahead approach is computationally time consuming for large k values. For example, even for the popular 2-steps look-ahead tree, the number of computations required to select the best split is $O(mn^3)$, where n denotes the number of attributes and m denotes the number of available instances. Other classification problems can be solved only with deeper look-ahead procedures, such as the 3-steps look-ahead trees, where the number of computations needed to select the best split increases to $O(mn^7)$ [14]. Moreover, in a counter-intuitive manner, it was suggested [6], [13] that in some cases the look-ahead trees often have less appealing convergence properties than the simplest greedy trees. Thus, in many cases, the k-steps look-ahead procedures do not outperform decision trees that are associated with more greedy approaches.

In this work we suggest a method for improving the greedy construction of decision trees, while taking advantage of their simplicity, tractability and their low time construction complexity. The suggested method obtains a low complexity, since it is based on a one-step look-ahead approach. Furthermore, it selects features (attributes) based on a distance criterion that takes into consideration not only the immediate contribution of potential attribute in the current step, but also its potential contribution to obtain an efficient classification tree in future steps. This dual-measure criterion relies on information theoretic metrics of the tree construction problem, and hence is called the ***d**ual **i**nformation **d**istance* (DID) method. We show that a good balance between these two distance components leads to a tractable, yet a "less greedy" construction approach. Such a feature is potentially appealing when the selected features are subject to noise and distortion, as often happens in real-life cases.

The framework of the suggested method uses a graph model and represents the construction of classification trees by finding the shortest paths over a *graph* of attribute's partitions. Nodes in the graph represent the possible partitions of the dataset by the selected attributes, while the edges are labeled by different selected attributes and their values. Selecting a path of edges leading to a specific node is equivalent in the proposed representation to a selection of attribute values that lead to a certain partition of the dataset. The partitions in the graph are defined as the states of the classification process. In each construction stage of the tree, the selected attribute values define the current partition of the dataset. Then, in the next stage, a selection of a new attribute values in a deeper tree level usually refines the current partition and so forth. This process of consecutive refinements ends when a proper stopping condition is satisfied. For example, when the obtained partition is at least as refined as the class partition, *i.e.* all instances of each of the obtained partition belong to a single class, or when no attributes are left for further selection. We show that the proposed modeling approach can represent known classification algorithms, such as ID3 and C4.5, as well as the proposed DID method, thus providing a good framework for comparisons.

We formulate the tree construction problem using concepts of information theory. Namely, at different construction stages we seek those attributes that maximize the mutual information about the classification variable. Motivated by the search for short paths over the graph of partitions, and inspired by the chain rule of mutual information, we introduce the DID method. Similarly to other popular construction algorithms, the proposed DID method constructs the classification tree in a recursive top-down manner. Yet, unlike these algorithms, the DID method takes into account two distance terms for attributes selection, as indicated above. The first term measures the *orthogonality* between the *current partition*

(current state of the classification process) and the *potential partitions* that can be potentially selected in the next steps of the algorithm. The second distance term refers to the *reduction* of the conditional entropy (measuring the uncertainty) about the *class partition* given the *partition* that is being selected. This is an equivalent criterion to the one used by the ID3 / C4.5 algorithm. As noted above, the first term is used to reduce the "greediness effects" of the proposed method. By weighting these two terms in a single objective function, the classification problem is modeled as a search for short paths over the partitions graph. In particular, the Rokhlin information metric [16] was found to be very effective in representing both the conditional entropy and the orthogonality measures. Based on publically available databases, we show that the DID method often outperforms the ID3 and the C4.5 popular classifiers, mainly in terms of robustness to *features removal*. This is an appealing characteristic of the proposed approach in cases where not all the features are available or known during the construction of the tree.

The rest of the paper is organized as follows. Section 2 presents the information theoretic analysis of the tree construction problem that motivates the characteristics of the DID method. Section 3 outlines the proposed DID method. Section 4 describes the experiments that were conducted by implementing the DID approach to known datasets and comparing the results to those obtained by known algorithms. Section 5 summarizes the work and proposes future research directions.

2. Information Measures of Attribute Partitions

Shannon [17] introduced the chain rule of the mutual information that reflects the conditional information, which is obtained from a series of random variables about a target random variable (Cover and Thomas [2]). When applying this rule to measure the mutual information between a list of attributes $A_1,\ldots,A_n$, that are selected sequentially, and a class target Y, the chain rule expression can be divided into two summation terms of conditional entropies

$$\begin{aligned} I\left(A_1,A_2,\ldots,A_n;Y\right) &= \sum_i I(A_i;Y|A_{i-1},\ldots,A_1) \\ &= \underbrace{\sum_{i=0}^{n} H\left(A_i|A_{i-1},\ldots,A_1\right)}_{\text{Attributes Orthogonality}} - \underbrace{\sum_{i=0}^{n} H\left(A_i|A_{i-1},\ldots,A_1,Y\right)}_{\text{Remaining Uncertainty}}. \end{aligned} \quad (1)$$

The first term represents the *orthogonality* among the attributes independently on Y, while the second term represents the uncertainty in each attribute about the class target, given the attributes that were previously selected. Since entropies are non-negative quantities, in order to maximize the mutual information between this list of attributes $A_i,\ldots A_1$ and the class attribute Y, one needs to maximize the first summation term and minimize the second summation term. The proposed DID approach, which is introduced in the next chapter, analogously aims to follow this direction.

This process of sequential selection of attributes during the construction of a decision tree can be described in terms of data partitions. In particular, each selection of an attribute partitions the dataset of instances according to the values of the selected attributes [4]. Accordingly, let us consider the information metrics of the partitions in the following manner:

Let X be a sample set of instances (examples) $X=\{x_1,x_2,\ldots,x_n\}$, which are classified (tagged) by the target attribute Y with probabilities

$$p(x_i) \geq 0, \sum_{x \in X} p(x) = 1 .$$

and let partition α_i denote a partition resulting from the selection of attribute i and let α_i^j denote a subset of such partition satisfying the condition that the value of attribute A_i is equal to j, such that

$$\alpha_i = \left\{ \alpha_i^j \middle| \alpha_i^j \in \alpha_i, \alpha_i^j \cap \alpha_i^k = \phi, \bigcup_j \alpha_i^j = \alpha_i \right\}. \quad (2)$$

Then let the multiplication between two partitions be defined as follows:

$$\alpha_i \vee \alpha_j = \left\{ \alpha_i^l \cap \alpha_j^k \mid l = 1,2,\ldots,|\alpha_i|, k = 1,2,\ldots,|\alpha_j| \right\}. \quad (3)$$

The process of sequential partitioning can be performed recursively. To understand how this process is reflected on the partitions space, let us consider the principle of restrictions:

Let $\alpha = \{\alpha^1, \alpha^2, \ldots, \alpha^m\}$ be an attribute partition (we omit the sub-index without loss of generality). Let us denote by $\alpha|_\beta$ a restriction of the attribute partition α by a partition $\beta \subseteq X$ and define it as follows:

$$\alpha|_\beta = \left\{ \alpha^1 \cap \beta, \alpha^2 \cap \beta, \ldots, \alpha^m \cap \beta \right\}. \quad (4)$$

Therefore $\alpha \vee \beta = \bigcup_{\beta_j} \alpha|_{\beta_j} = \bigcup_{\alpha_i} \beta|_{\alpha_i}$.

2.1 Entropy Between Partitions

Recall that $X = \{x_1, x_2, \ldots, x_n\}$ is the sample space and let $p: X \to [0,1]$, $\sum_{i=1}^n p(x_i) = 1$ be a probability mass function defined over *X*. Let α be a partition of *X*, and let

$$p(\alpha^j) = \sum_{x \in \alpha^j} p(x), \alpha^j \in \alpha, \alpha^j \cap \alpha^k = \phi \ \ \forall j \neq k, \sum_{\alpha^j \in \alpha} p(\alpha^j) = 1 . \quad (5)$$

Then, the entropy (to the base 2) of the partition α is computed as follows:

$$H(\alpha) = -\sum_{\alpha^j \in \alpha} p(\alpha^j) \log_2 p(\alpha^j) \ . \quad (6)$$

Let β be another partition of *X*. The *Conditional entropy of partition* α given partition β is defined as follows:

$$H(\alpha \mid \beta) = -\sum_{\beta^k \in \beta} \sum_{\alpha^j \in \alpha} p(\alpha^j, \beta^k) \log_2 p(\alpha^j \mid \beta^k) . \quad (7)$$

where

$$p(\alpha^j, \beta^k) = p(\alpha^j \cap \beta^k) \ \text{ and } \ p(\alpha^j \mid \beta^k) = \frac{p(\alpha^j \cap \beta^k)}{p(\beta^k)} . \quad (8)$$

For more information about the properties of the entropy of partitions see [18].

2.2 The Rokhlin Metric

Let χ be the set of all possible partitions of X, and let $\alpha, \beta \in \chi$ be two partitions defined over X. The *Rokhlin distance* between partitions α and β or the *Rokhlin metric* on χ, which was first introduced by [16], is defined as follows [11], [16], [18]:

$$d_{Rok}(\alpha, \beta) = H(\alpha|\beta) + H(\beta \mid \alpha) . \quad (9)$$

It is important to note that the Rokhlin distance follows the required mathematical properties of a metric [18], namely:

$$d(\alpha,\beta)\geq 0\,,\quad d(\alpha,\alpha)=0\,,$$

and

$$d(\alpha,\beta)\leq d(\alpha,\gamma)+d(\gamma,\beta),\quad \forall\alpha,\beta,\gamma\in\chi\,.$$

Let us clarify the relation between the Rokhlin metric $d(\alpha,\beta)$ and the mutual information $I(\alpha,\beta)$:

Let $X=\{x_1,x_2,\ldots,x_n\}$ and $Y=\{y_1,y_2,\ldots,y_m\}$ be two finite sets and $p: X\times Y \rightarrow[0,1]$ be a joint probability mass function defined over these sets. The *mutual information* in X about Y is defined as follows [17]:

$$I(X;Y)=H(Y)-H(Y|X)=\sum_{i=1}^{n}\sum_{j=1}^{m}p(x_i,y_j)\log\left(\frac{p(x_i,y_j)}{p(x_i)p(y_j)}\right). \tag{10}$$

Let $\alpha,\beta\in\chi$ be two partitions of X. The mutual information in β about α is given by the following expression:

$$I(\alpha;\beta)=H(\alpha)-H(\alpha|\beta)=\sum_{\alpha^j\in\alpha}\sum_{\beta^k\in\beta}p(\alpha^j,\beta^k)\log\left(\frac{p(\alpha^j,\beta^k)}{p(\alpha^j)p(\beta^k)}\right). \tag{11}$$

Therefore, the relation between the mutual information $I(\alpha;\beta)$ and the Rokhlin metric $d(\alpha,\beta)$ is the following [9]:

$$d(\alpha,\beta)=H(\alpha,\beta)-I(\alpha,\beta)\,. \tag{12}$$

Thus, minimization of the Rokhlin distance is equivalent to simultaneously maximizing the mutual information and minimizing the joint entropy.

Other useful properties of the entropy of partition are the following. Let $\alpha,\beta\in\chi$. If β is a refinement of α, $\beta\succ\alpha$, then $H(\beta|\alpha)=H(\beta)-H(\alpha)$. Backwards, if $H(\beta|\alpha)=H(\beta)-H(\alpha)$, then $\beta\succ\alpha$. Thus, $H(\alpha|\beta)=0$ if and only if $\alpha\prec\beta$. If the Rokhlin metric $d(\alpha,\beta)=0$, then $\alpha=\beta$ (for details see [9], [18]).

Popular classification algorithms, such as the ID3 and the C4.5, take into account the uncertainty in the class attribute Y following the selection of some attribute A_i, and measure it by the conditional entropy, $\mathrm{H}(Y|A_i)$, *i.e.* they look for an attribute A_i that minimizes the conditional entropy measure. Note from Equation (9) that when using the Rokhlin distance measure, we take into account not only $H(Y|A_i)$ but also $H(A_i|Y)$. A relevant question is whether the second term of conditional entropy contributes to the proper selection of A_i. The logic behind the second conditional entropy term is less explicit but can be well understood by using a small illustrative example in Figure 1. Let us consider the following six-instances dataset that consists of a class attribute Y and two input attributes, A_1 and A_2. As clearly seen, each of the attributes A_1 or A_2 alone classify the class attribute Y, *i.e.,* $H(Y|A_1)=0$ and $H(Y|A_2)=0$. However, A_1 partitions the dataset to six subsets, $\{\alpha_1^1,\alpha_1^2,\ldots,\alpha_1^6\}$, while A_2 partitions the dataset to two subsets only $\{\alpha_2^1,\alpha_2^2\}$. This fact is well reflected in the second conditional entropy term of the attributes given Y, *i.e.*, $H(A_1|Y)=1.58$ and $H(A_2|Y)=0$. By aiming to minimize the Rokhlin distance, which also considers the second conditional entropy term,

one will prefer the attribute A_2 over A_1, thus choosing an attribute that partitions the dataset as little as required, or in other words, staying as close as possible to the class attribute, while avoiding unnecessary splitting.

A_1	A_2	Y
1	1	X
2	1	X
3	1	X
4	2	O
5	2	O
6	2	O

Figure 1. Small data example of 6 instances. A_1 , A_2 depict the attributes and Y depicts the class label of each record. Each attribute alone classifies the class attribute Y , however $H(A_1|Y)=1.58$ and $H(A_2|Y)=0$.

Under practical considerations, obtaining a more refined partition with respect to the class partition is sometimes unnecessary and costly. An example for such a practical situation can be found in the area of medical tests, where some types of tests are more expensive than others. Thus, the first type of tests might only determine whether the tissue is contaminated or not and is relatively cheap. The second type of test, on the other hand, also distinguishes between the possible sources of the contamination. If the rough partitioning, which is achieved by the first type of test, is enough for the classification purpose, it will obviously be preferable by the Rokhlin distance.

It is important to note that other distance measures can be also used between different partitions. Not all the distances measures are proper metrics in terms of the mathematical requirements; however, many such measures could possibly lead to good classification results. The optimal selection of distance measure for various classification and clustering problems is a subject of ongoing research work (see [9], [15]).

3. The Proposed DID Approach

As a general framework for the proposed classification method, we use an iterative version of Korf's Learning Real Time Algorithm (LRTA*) [10]. The suggested DID tree is constructed by using a sequential partitioning scheme, where in each stage of the construction, the selected attributes up to this stage define the current partition of the sample space. In the next construction stage, a selection of a new attribute in a new tree level refines (in most cases) the current partition. This process of continuous refinements ends when the obtained partition of the sample space is at least as refined as the class partition, thus, all instances in all the sub-partitions belong to the same class, or when no attributes are left for selection.

Specifically, at each step the algorithm acts on the restricted partitions as follows. Let $\alpha=\{\alpha^1,\alpha^2,...,\alpha^m\}$ be an attribute partition. Let us denote by $\alpha|_\beta$ a *restriction* of the attribute partition α by a partition $\beta\subseteq X$, as defined by Equation (4). Assume that a probability mass function is defined over the sample space X and that the probabilities of the instances are, without loss of generality, equal. In addition, let us assume that attribute α_Y is the desired class attribute partition.

Let us denote the *current partition* of the dataset by $\alpha_c = \left\{\alpha_c^1, \alpha_c^2, ..., \alpha_c^m\right\}$. Then, the recursive DID creates *m* threads. In the *i*-th thread, *i*=1,…,*m,* the current partition α_c^i is a set of elements having normalized probabilities with respect to the sum of elements of α_c^i. The selection of the next partition is taken from the current partition neighborhood $N\left(\alpha_c^i\right)$, which includes the following partitions:

$$N\left(\alpha_c^i\right) = \left\{\alpha_j \mid_{\alpha_c^i}\right\}, j \in F_c,$$

where $F_c = \{j : A_j \textit{ not selected by the algorithm yet}\}$ and where the probabilities are normalized over each sub-partition.

If the classification is achieved in a subset, then the thread terminates. If no further attributes are left for selection, then the thread terminates and the classification is determined by the major class in the sub-partition. If the cardinality of the chosen sub-partition is not greater than 1, then the thread terminates and classification is achieved. Otherwise, the chosen set defines the current partition for this thread and the process continues. Analogously to the presented concepts of the chain rule of information in Equation (1), the proposed objective function for selecting the partitioning attribute in each step is to maximize the orthogonality between the current partition α_c and the next candidate partition α_j, while minimizing the distance between the considered (restricted) partition α_j and the class partition α_Y. Using the Rokhlin distance metric in Equation (9), this objective is defined as follows:

$$\min_{\alpha_j}\left(w_1 \left\langle Rokhlin\left(\boldsymbol{\alpha_c}, \boldsymbol{\alpha_j}\right)\right\rangle + w_2 \left\langle Rokhlin\left(\boldsymbol{\alpha_j}, \boldsymbol{\alpha_Y}\right)\right\rangle\right). \tag{13}$$

For simplicity purposes, let us mark the above two distances with d_1 and d_2, resulting in:

$$\min_{\alpha_j}\left\{w_1 d_1\left(\alpha_c, \alpha_j\right) + w_2 d_2\left(\alpha_j, \alpha_Y\right)\right\}, \tag{14}$$

where $w_1 \le 0$ and $w_2 > 0$.

The term $d_1\left(\alpha_c, \alpha_j\right)$ denotes the orthogonality measure distance between the current partition and the next chosen partition, which we try to maximize, analogous to the first term in Equation (1); the highest the orthogonality between the next restricted partition and the current partition, the better. High orthogonality values often lead to a situation where future selections of attributes result in a refined partition with respect to the class partition.

Let us note that measuring the orthogonality between attributes is not a new idea. In fact, this is a basic principle used in design of experiments for example by the D-optimality criterion [12]. In the *data mining* society, measuring the orthogonality between attributes is sometimes performed as part of the pre-processing stage (*e.g.,* [19]) by considering the entire attribute vector. In our approach, however, we measure the orthogonality during the construction of the classification tree – only when it applies to a sub-set of the attribute. Such an approach enables us to reveal the effective orthogonal relations between the restricted partitions of interest. In other words, we look for orthogonality in sub-partitions where they are most effective, rather than using is as a general feature-selection measure.

The second distance $d_2\left(\alpha_j, \alpha_Y\right)$ denotes the information distance, therefore measures the difference between the considered partition α_j – obtained by selecting an additional attribute that refines the current partition – and the final class partition α_Y, *i.e.* this term measures the remaining uncertainty in the considered partition α_j about the final class partition α_Y, which is analogous to the second term in Equation (1). By using this distance,

one seeks to find a partition which gives the maximum information about the class partition.

For a better algorithmic flexibility, we allow to combine these distance measures by using different weights, w_1 and w_2, in a single objective function. Such a weighting enables refining the constructed classification tree to better suit the data and address issues of variance-bias effects, where w_1 is usually selected as a negative integer (maximizing the orthogonality distance), and w_2 is usually selected as a positive integer (minimizing the information distance). The exact values of the weights can be determined in the learning phase. The pseudo code of the recursive DID algorithm is given below (see also [9]).

Algorithm (DID)

Input:

(i) set of weights w_1, w_2.

(ii) attributes partitions $\alpha_1, \alpha_2, ..., \alpha_n$.

(iii) class partition α_Y.

Initialization:

(i) Initialize current partition $\alpha_c \leftarrow \alpha_0$.

(ii) Define E to contain the used attributes for partitioning. Init $E = \{\phi\}$.

(iii) Define F to contain the unused attributes for partitioning. Init $F = \{1, 2, ..., n\}$.

Step:

For each sub-partition $\alpha_c^i \in \alpha_c$ such that:

(i) $\left|\alpha_c^i\right| > 1$.

(ii) $\alpha_Y\left|_{\alpha_c^i}\right.$ is not yet classified.

(iii) F is not empty.

start the **Search** procedure for the sub-partition α_c^i.

Function Search: Given set α_c^i, E and F.

(1) Initialize current partition α_c by α_c^i.

(2) *init* $E_c \leftarrow E; F_c \leftarrow F$ where instances in the sub-partition α_c^i are considered equi-probable and the probabilities of all instances of the sub-partition sum up to 1.

(3) Normalize probabilities of the elements of α_c^i.

(4) Create local class partition $\alpha_Y\left|_{\alpha_c}\right.$.

(5) Generate neighborhood partitions: $N(\alpha_c) = \left\{\alpha_j\left|_{\alpha_c}\right.\right\}, j \in F_c$.

(6) Normalize probabilities of neighbors and of the class partition.

(7) Obtain distance measures by calculating $d_2\left(\alpha_c, \alpha_j\left|_{\alpha_c}\right.\right)$ and $d_2\left(\alpha_j\left|_{\alpha_c}\right., \alpha_Y\left|_{\alpha_c}\right.\right), j \in F_c$.

(8) Choose next partition $\alpha_{next} \leftarrow argmin_{\alpha_j \in N(\alpha_c)}\left\{w_1 d_1\left(\alpha_c, \alpha_j\left|_{\alpha_c}\right.\right) + w_2 d_2\left(\alpha_j\left|_{\alpha_c}\right., \alpha_Y\left|_{\alpha_c}\right.\right)\right\}$.

(where ties are resolved arbitrary).

(9) Update E_c and F_c (move j from F_c to E_c).

(10) Move to next partition: $\alpha_c \leftarrow \alpha_{next}$.

(11) For each sub-partition $\alpha_c^i \in \alpha_c$ such that

(i) $\left|\alpha_c^i\right| > 1$.

(ii) $\alpha_Y\Big|_{\alpha_c^i}$ is not yet classified.

(iii) F_c is not empty.

start the **Search** procedure (for sub-partition α_c^i, F_c, E_c).

(12) Stop condition

(i) If $\alpha_Y\Big|_{\alpha_c}$ is classified, return.

(ii) If $\left|\alpha_c\right| = 1$ classify according to the instance's class value, return.

(iii) If $F_c = \{\phi\}$ classify according to the most common value of the class attribute, return.

The suggested DID tree can also be pruned according to different criteria. In our study, if the tree is chosen to be pruned, the train dataset could be divided into two different datasets – the first is used for building the tree, and the second dataset to validate the pruning efficiency. Our strategy was to prune the different nodes until the average accuracy of all records in the second dataset could not be improved.

4. Experiments and Analysis

In this section the proposed approach is tested on several different known classification problems, and compared against the popular ID3 and C4.5 classifiers.

Our testing methodology uses a training set to construct the classification tree model. The partitioning process is then performed until each reached subset (leaf in the tree) is fully classified, when the predefined stopping criterion is reached[1], or when no further attributes are left for selection (then the classification is defined according to the majority rule of the instances in this sub-set). The validity of the classification model is performed over a test set, often by various holdout datasets. Some of the published datasets are already partitioned into a training set (sometimes more than one) and a test set. In these cases we used the sets as represented. In cases where the datasets were not split into training and test sets, we randomly divided the data into training and test alien sets. In such cases, the splitting procedure was replicated several times (usually between 5 to 10 times) to calculate the average performance values, as well as the standard deviation[2].

The programmed application can handle different distance metrics, as well as assigning different weights for these distances. The obtained results rely particularly on the following performance measures, based on the classification tree and the test set:

- The tree average depth (known also as the average path length).
- True classification percentage (the accuracy) of the test set by the classification.
- The number of leaves in the classification tree.
- The number of decision nodes in the classification tree.

---

[1] The programmed application can handle different stopping rules, such as stopping the partitioning process when a sub-partition reaches some predefined size.

[2] The standard deviation was calculated for the *average depth* of the tree.

Two other measures that were analyzed were the minimum and the maximum number of decisions required for the classification model. In practice, the average number of decisions is often associated with the implemented test procedure that requires an economic justification and affects cost and time measures.

Table 1. Summarizing Comparison between ID3, C4.5 and DID decision trees.

Dataset	Domain	Size		ID3		C4.5		DID		
		Number of instances in dataset (and the test set)	Number of Attributes	Average Depth	True Classification Percentage	Average Depth	True Classification Percentage	Average Depth	True Classification Percentage	(w1, w2)
Monk's-1	Robotics	124 (432)	6	3.21	82%	3.32	82%	2.66	96.7%	(-5,1)
Monk's-2	Robotics	169 (432)	6	4.34	70.4%	4.6	75%	4.2	66%	(-2,1)
Monk's full Random set	Robotics	216 (216)	6	1.93	100%	2.04	100%	1.8	100%	(-2,1)
Connect4	Games	67,557	42	5.85	73.8%	10.16	79.4%	5.64	75%	(-5,1)
SPECT Heart	Medicine	80 (187)	22	9.6	75.1%	10.2	80.3%	9.3	76%	(-5,1)
Voting	Social	435	16	1.8	96%	2.2	96.6%	2.1	96%	(-1,1)
Balance Scale	Social	625	4	3.4	76.3%	3.4	78.6%	3.3	76.6%	(-2,1)
Cars	General	1728	6	2.82	77.1%	2.83	77%	2.77	78.5%	(-1,1)
Tic-Tac-Toe	Games	958	9	4.62	80.6%	4.62	80.4%	4.6	76.2%	(-1,1)
Soy Beans	Life	47	35	1.35	100%	2.37	97%	1.32	97%	(-1,1)
Lymphography	Medicine	148	18	2.71	75.1%	6.51	77.3%	2.6	72.6%	(-2,1)

In Table 1, we outline a summarized comparison for nine published problems (using eleven configurations). For each problem we present the dataset size, the number of attributes available for classification, the average depth of the obtained classification tree and the true classification rate over the test set (i.e., the percentage of instances correctly classified).

As the construction of the DID tree depends on the weights of the distances, these parameters are also listed in the last column of the table. By analyzing several combinations of weights over the train set, we searched for the most promising weights combination that could result primarily in a relatively small tree with a lower average depth, yet with relatively high classification accuracy.

As can be seen in Table 1, the proposed algorithm outperforms both ID3 and C4.5 classifiers in almost all of the considered classification problems in terms of the average depth of the tree. For the Monk's-1 dataset, the proposed algorithm results in a tree model with an average depth of 2.66 decisions, while the ID3 and the C4.5 average depths are 3.21 and 3.32 decisions respectively. The difference is even more significant with respect to the Connect4 dataset – a relatively large dataset, consisting of more than 67,000 instances and 42 available attributes. In this case, the DID classifier results in a tree with 5.64 decisions in average, while the C4.5's model requires 10.16 decisions on average. A significant difference was also discovered with the Lymphography dataset, in which the DID algorithm produced a tree with an average depth of 2.6 decisions, compared with C4.5's average depth of 6.51 decisions. In one of the tested datasets, the DID algorithm resulted in a higher average length than the ID3: in the Congressional Voting dataset, the ID3 produces a slightly lower average than that of the DID algorithm, yet, the DID algorithm was still better than the C4.5's.

With respect to the true classification percentage, the DID algorithm was found to be less accurate. In most of the problems the C4.5 model obtained a better classification rate than the DID and the ID3, yet this difference was relatively marginal. The DID algorithm performs better than the other two algorithms when applied to the Monk's-1 dataset. It produces a model with a true classification rate of 96.7%, compared to a classification rate of 82% produced by both ID3 and C4.5. For example, in the CARS dataset, the DID algorithm results in a true classification rate of 78.5%, slightly better than the ID3 and the C4.5 models. A relatively inferior accuracy for the DID was obtained in the Monk's-2 dataset, in which the DID algorithm succeeded in classifying only 66% of the cases, compared with 70.4% and 75% for the ID3 and the C4.5 algorithms, respectively. In most of the other cases, the DID algorithm ranks second, usually only slightly worse than the best classifying algorithm. The potential effects of the type and features of the database on the success of different classification algorithms, including the DID, will be the subject for future research.

Table 2. Classification accuracy of SMV, C4.5 (J48 implementation) and DID for different problems taken from the UCI Repository.

Case	*No. of features*	*Type of data*	*SVM best accuracy%*	*J48 Best accuracy %*	*J48 pruning coefficient*	*DID best accuracy %*	*DID parameters* (w_1, w_2 , *prune tree*)
australian	14	continuous	55.5	86.2	0.05	86.9	(-0.7, 1, true)
breast	9	continuous	96.5	93.6	0.1	93.5	(-0.65, 1, false)
diabetes	8	continuous	65.1	74.2	0.2	72.6	(-1.6, 1, true)
glass	8	continuous	69.16	50.1	0.05	51.8	(-0.1, 1, false)
glass2	8	continuous	76.68	75.3	0.45	82.1	(-2.05, 1, true)
heart	13	continuous	55.93	79.4	0.45	79.5	(-0.05, 1, true)
iris	4	continuous	96.67	94.4	0.25	95.6	(-0.5, 1, true)
pima	36	continuous	65.1	73.1	0.15	72.2	(-0.8, 1, true)
segment	18	continuous	63.9	94.1	0.05	93.6	(-0.8, 1, false)
shuttle-small	9	continuous	89.41	62	0.3	61.9	(-0.8, 1, false)
vehicle	18	continuous	30.5	69.7	0.25	65.2	(-0.8, 1, false)
waveform-21	21	continuous	86.1	76.3	0.1	73.5	(-0.65,1, true)
cleve	13	mixed	54.73	78.9	0.4	78.0	(-0.45, 1, true)
crx	15	mixed	65.67	87.5	0.35	87.6	(-0.15, 1, true)
german	20	mixed	70	65.2	0.15	66.6	(-0.8, 1, true)
hepatitis	19	mixed	83.55	57.4	0.45	64.2	(-2.8, 1, true)
chess	36	discrete	93.83	99.3	0.05	99.8	(-1.05, 1, false)
corral	6	discrete	96.89	98.1	0.3	98.3	(0, 1 , false)
flare	9	discrete	82.37	61.2	0.05	68.9	(-0.35, 1, true)
mofn-3-7-10	10	discrete	100	100	0.05	100	(-2.45, 1, false)
soybean-large	35	discrete	87.19	95.8	0.05	94.2	(-0.85, 1, false)
vote	15	discrete	95.35	94.7	0.3	95.4	(-1.6,1,true)

4.1. Testing the Robustness of the Algorithms to Features Removal

In the second stage of the analysis, we compared the DID and the C4.5 (using the J48 implementation) classifiers, with respect to the robustness of the classification when key features are removed from the dataset. The order of features' exclusion was set from the most influential feature to the least influential one, as measured by the mutual information gain between the feature and the class attribute. The exclusion of the features was performed as follows: the first classification comparison was based on all the available features; the second comparison was based on all the features except the most influential

one; in the third comparison, we excluded the two most influential features, while the last comparison was based on the least influential feature.

The classifiers were tested on several different known classification problems taken from the UCI Repository [1] and compared to the results obtained by the C4.5 (J48) and SVM [3] classifiers. For each case presented in Table 2, all available records of each evaluated dataset were divided into train and test datasets using a five-folds validation procedure. Attributes with non-discrete values (*i.e.* less than 12 unique values) were discretized into four equal probability intervals. Both trees were assessed in terms of the average classification accuracy (over all folds). The different free parameters of the classification trees (for the C4.5 - the pruning coefficient when implementing its J48 version; and for DID the $w1, w2$ weights and whether to prune the tree) were set to achieve maximal average classification accuracy when considering all available features (see details in Table 2). The results in Table 2 indicate that for part of these cases, such as Glass2 and Flare, the DID algorithm outperforms the existing C4.5 decision tree. For the remaining cases, similar classification accuracies to C4.5 were obtained. In addition, the DID algorithm outperformed the SVM classifier accuracy in 13 out of 21 cases, indicating that this method is comparable to one of the best know classifiers to this date.

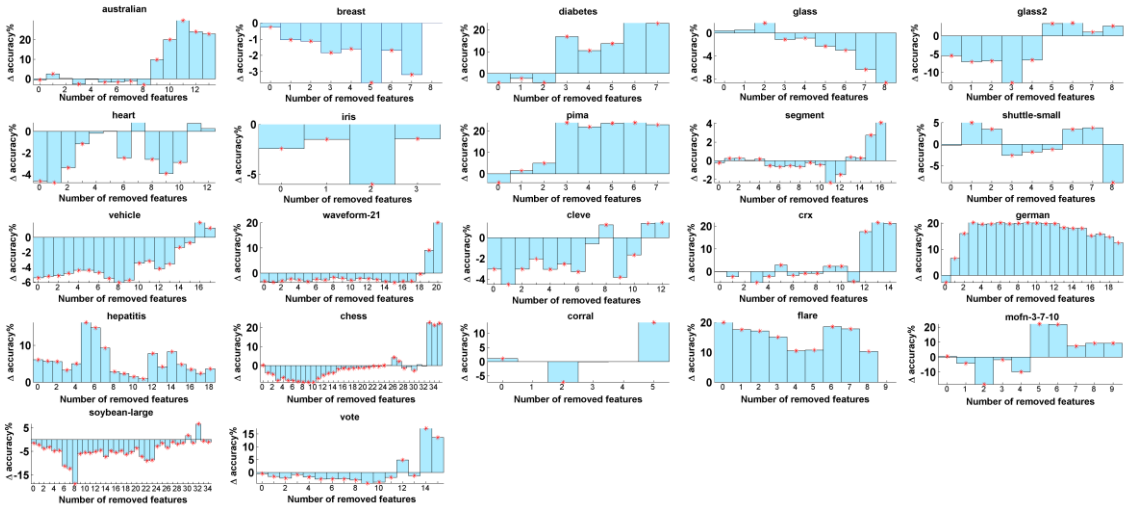

Figure 2. Accuracy of C4.5 (J48) and DID as a function of the number of removed features for different cases taken from the UCI Repository. The blue graphs represent the accuracy percentage difference between DID and J48 (DID–J48), averaged on the five-folds and 100 different runs. Significant differences (unpaired *t*-test, $p<0.01$ are marked with a red *).

Finally, we looked at the robustness of the difference between the classification accuracies of the C4.5 (J48) and the DID trees under features removal conditions for 100 different train-test divisions. Subsequently, we investigated the statistical significance of the accuracy differences under features removal using the unpaired t-tests. Figure 2 shows for each case study the difference between the percentage accuracy between the DID and the C4.5 (J48) algorithms (averaged across all five-folds and 100 different runs). For part of the datasets, the DID algorithm was found to be significantly ($p<0.01$) more robust to features removal than the C4.5 (*e.g.* the *australian*, *diabetes*, *pima*, *german*, *hepatitis* and *flare* datasets).

5. Discussion and Conclusions

In this work we suggested a general framework for the construction of classification trees. The new framework models the classification problem by mapping it to a problem of finding the *shortest-path* over a graph of partitions, while using a dual information distance (DID) measure. The partitions in this graph are constructed by combinations of the input attributes, as well as by the class attribute itself. Any combination of the input attributes can generate a partition which can be examined by the DID measure. The framework of finding the *shortest-path* over a graph of partitions allows the use of different distance metrics. In this paper, we mainly focused on information-related metrics and used the conditional entropy, the information gain and the Rokhlin measures, as the distances measures between the partitions. Using the proposed graphic representation, the proposed DID method can represent known classification algorithms, as well as new and attractive ones, and can be further expanded to address the complexity-optimization tradeoff that exists in these problems. The use of other metrics over the graph of partitions is subject to future research.

The proposed DID algorithm takes into account two distance measures that traditionally were addressed in two distinct scientific fields. The first distance measure examines the orthogonality between the current partition and the next partition. This measure is commonly used in the field of *design of experiments*, for example when implementing D-optimal design. The second distance measure, which is vastly used in *data mining*, considers the information distance (measured by the conditional entropy) between the candidate partition and the class partition. This measure reflects how the candidate partition can contribute to the entropy reduction - a measure that can be found in other decision trees. To obtain algorithmic flexibility we allow implementing different weights to these two distance components that can be adjusted in the learning phase.

The DID algorithm was found to be competitive to the popular C4.5 (J48), ID3, SVM algorithms with respect to both the classification accuracy and average depth (also known as EPL – "Expected Path Length"). The average depth of the tree is an important measure that represents how many tests are required on the average to classify new instances. This is a critical measure for online classification in Big-Data environments, when only part of the features are known. For example, in a healthcare system, with hundreds of patients admitted every day to a hospital, the medical stuff must quickly diagnose each patient and recommend a correct treatment. This occurs when not all possible symptoms are expressed and not all laboratory results are known upfront. Therefore, as opposed to classical classification methods such as SVM, short depth trees (that are used "online") can reduce the time required for treating each patient while using fewer laboratory tests, therefore saving time and money. Another example that reflects the advantage of shallow and relatively accurate decision trees in Big Data environment is related to content-searching sites via the internet. In this case, visitors should reach their desired content quickly without moving to another site. The Nielsen NetView Survey performed in June 2010 [8] indicates that in the US, users spend 819 million hours a month on online related sites (*i.e.*, portals, video/movies, search, software Info, entertainment and auctions). Therefore even a reduction of 1% of this spent time implies a potential saving of more than 8.1M man hours each month. In addition, the true classification rate of the DID was found to be relatively robust to feature removal with respect to the C4.5 (J48 implementation), emphasizing the advantage of the DID in making real-time decisions.

Let us note that the DID heuristic can be extended in several research directions, such as: i) Allowing different weights in different levels of the tree, thus adding a degree of freedom to adjust the weight parameters along the tree levels, as the orthogonality measure is potentially more important at higher tree levels; ii) Considering different orthogonality and information metrics, such as the *Ornstein* metric as indicated in [9]; iii) Studying different target functions to evaluate the distances between different partitions over the partitions space; and iv) Combining the DID algorithm with look-ahead and look-backward procedures. Such a direction will allow not only k-steps look-head moves, but also k-steps backward moves. Thus, developing a sensitive-enough metric to allow the designer, during the construction of the tree over the graph of partitions, to abandon a path on the graph and go backwards if the potential benefit of this path is not sufficiently promising.

Another potential extension of the DID algorithm is the use of mid-level points (MLP) in the graph of partitions. The MLP refer to nodes over the graph of partitions that represent a promising combination of features under some orthogonality/information criteria. The proposed approach can focus on estimating those "appealing" MLP, enabling to "jump" to these MLP to find the shortest path in the graph.

A concluding note is that integrating concepts of design of experiments, data mining and information theory, as exercised in this work, can be beneficial for the construction of better classification tree models.

Acknowledgements

This research was supported by the Israel Science Foundation (grant No, 1362/10).

References

1. Bache, K. and Lichman, M. (2013). *UCI Machine Learning Repository*, http://archive.ics.uci.edu/ml., School of Information and Computer Science, University of California, Irvine.
2. Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*, 2nd edition, John Wiley & Sons, New York.
3. Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning,* 20, 273-297.
4. De Mántaras, R. L. (1991). A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6, 81-92.
5. Duda, R. O., Hart, P. E. and Stork, D. G. (2001). *Pattern Classification,* 2nd edition, Wiley, New York.
6. Goodman, R. M. and Smyth, P. (1988). Decision tree design from a communication theory standpoint. *IEEE Transactions on Information Theory*, 34(5), 979-994.
7. Hancock, T., Jiang, T., Li, M. and Tromp, J. (1996). Lower bounds on learning decision lists and trees, *Information and Computation,* 126, 114-122.
8. June 2010 Nielsen NetView survey on Internet usage (2010). Retrieved from http://www.nielsen.com/us/en/newswire/2010/what-americans-do-online-social-media-and-games-dominate-activity.html.
9. Kagan, E. and Ben-Gal, I. (2006). An informational search for a moving target. *Electrical and Electronics Engineers in Israel,* 24, 153-155.
10. Korf, R. E. (1990). Real-time heuristic search. *Artificial Intelligence,* 42, 189-211.

11. Martin, N. F. and England, J. W. (2011). *Mathematical Theory of Entropy*, vol. 12, Cambridge University Press, UK.
12. Montgomery, D. C. (2008). *Design and Analysis of Experiments*, 5th edition, Wiley, New York.
13. Murthy, S. K. and Salzberg, S. (1995). Decision Tree Induction: How Effective Is the Greedy Heuristic? *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 222-227.
14. Page, D. and Ray, S. (2003). Skewing: An efficient alternative to lookahead for decision tree induction. *International Joint Conference on Artificial Intelligence*, 601-612.
15. Priness, I., Maimon, O. and Ben-Gal, I. (2007). Evaluation of gene-expression clustering via mutual information distance measure. *BMC Bioinformatics,* 8, 111.
16. Rokhlin, V. A. (1967). Lectures on the entropy theory of measure-preserving transformations. *Russian Mathematical Surveys,* 22, 1-52.
17. Shannon, C. E. and Weaver, W. (1948). *A Mathematical Theory of Communication*, eds. American Telephone and Telegraph Company.
18. Sinaĭ , I. A. G. e. (1976). *Introduction to Ergodic Theory*. vol. 18, Princeton University Press.
19. Yang, J. and Li, Y. P. (2006). Orthogonal relief algorithm for feature selection. *Lecture Notes in Computer Science*, 4113, 227-234.

Authors' Biographies:

Professor Irad Ben-Gal is the chair of the Department of Industrial Engineering at Tel Aviv University, Israel. His research interests include statistical methods and applications of information theory and machine learning to industrial and service systems. He holds a B.Sc. (1992) degree from Tel-Aviv University, M.Sc. (1996) and Ph.D. (1998) degrees from Boston University. Irad is the former chair of the Quality Statistics & Reliability (QSR) society at INFORMS and a member in the Institute of Industrial Engineers (IIE), the European Network for Business and Industrial Statistics (ENBIS) and the International Statistical Institute (ISI). He is a Department Editor in IIE Transactions and serves in the Editorial Boards of several other professional journals. He wrote and edited five books, published more than 70 scientific papers and received several best papers awards. Prof. Ben-Gal supervised dozens of graduate students, led many R&D projects and worked with companies such as Siemens, Intel, Applied Materials, GM, Kimberly-Clark, Proctor and Gamble, IBM, SAS, Oracle and SAP. Irad is a co-founder and an active chair of Context-Based 4casting (C-B4), a software company that provides automated predictive analytics solutions.

Alexandra Dana is a Ph.D. student in the Department of Biomedical Engineering at Tel Aviv University, Israel. Her research primary focuses on developing new computational methods and statistical tools for analyzing large scale biological data. Alexandra received her MSc in Electrical Engineering from Tel Aviv University in 2009 and her BSc in Computers Engineering in 2004 from the Technion Institute of Technology, Israel.

Niv Shkolnik is a former B.Sc. and M.Sc. (2009) student at the Department of Industrial Engineering at Tel Aviv University, Israel.

Dr. Gonen Singer is the Head of the Department of Industrial Engineering and Management, at Afeka-Tel-Aviv Academic College of Engineering, Israel, from 2010. Dr. Singer has extensive expertise in *machine learning techniques* and *stochastic optimal control* and their applications to real-world problems in different areas, such as retail, maintenance, manufacturing and homeland security. His research activities over the last years have focused on the area of production inventory and unstable marketing problems in the framework of stochastic optimal control and development of pattern-based predictive-analytic models and tools for complex and adaptive systems.