

Learning Multicriteria Classification Models from Examples: Decision Rules in Continuous Space

J. Dombi, Á. Zsíros *

Department of Applied Informatics, University of Szeged, Hungary

Abstract

The classification problem statement of multicriteria decision analysis (MCDA) is to model the classification of the alternatives/actions according to the decision maker's preferences. These models are based on outranking relations, utility functions or (linear) discriminant functions. Model parameters can be given explicitly or learnt from a preclassified set of alternatives/actions.

In this paper we propose a novel approach, the Continuous Decision (CD) method, to learn parameters of a discriminant function, and we also introduce its extension, the Continuous Decision Tree (CDT) method, which describes the classification more accurately.

The proposed methods are results of integration of Machine Learning methods in Decision Analysis. From a Machine Learning point of view, the CDT method can be considered as an extension of the C4.5 decision tree building algorithm that handles only numeric criteria but applies more complex tests in the inner nodes of the tree. For the sake of easier interpretation, the decision trees are transformed to rules.

Key words: Multiple criteria analysis, Classification, Artificial intelligence, Decision trees, Fuzzy sets

* Corresponding author. Address: Department of Applied Informatics, University of Szeged, H-6701 Szeged, P.O. Box 652, Hungary

Email addresses: dombi@inf.u-szeged.hu (J. Dombi),
zsiros@inf.u-szeged.hu (Á. Zsíros).

1 Introduction

The decision process has two major aims: first, to explain decisions and second, to give recommendations how to make a decision under specific circumstances. These are very similar to the goals of the Inductive Learning[1,2] approach in the field of Artificial Intelligence (AI), where the first step is to establish a model from previous experience which is later applied to predict future situations. This parallelism leads to the possible application of AI methods in Decision Support Systems.

In Multicriteria Decision Analysis (MCDA), the inputs are usually described by numerical (continuous) criteria, such as value functions and orderings on a real interval. However, most learning methods in the field of artificial intelligence can detect relationships among elements of an input dataset described by a set of categorical (discrete) criteria (like hierarchical classifiers, decision trees). In order to apply these methods in MCDA, they should be extended to work on numerical criteria/attributes. In this paper, we propose such an extension. Our novel Continuous Decision (CD) and Continuous Decision Tree (CDT) methods help understand the structure of the input dataset described by a set of numerical criteria in the form of a discriminant function or a decision tree, respectively, which could be transformed into decision rules.

In this article, some shortcomings of the well-known ID3 and C4.5 decision tree building methods are discussed and solutions are proposed. An alternative measure is described instead of the entropy function, which builds on the measure of fuzziness using a monotone fuzzy operator. In the proposed methods, continuous criteria are handled without discretization of their values. In the geometric aspect of the problem, our method allows the separation of the decision space with arbitrary figures, such as hyperplanes, spheres, etc., which can also be straightforwardly interpreted by the decision maker.

The paper is organized as follows. In Section 2, a short introduction to the classification problem of MCDA and the (linear) discriminant function model are discussed. The concept of the decision tree and the well-known ID3 and the C4.5 decision tree building methods are described in Section 3. The proposed modifications on the measure of separation is given in Section 4. The derived CD and CDT methods, and ruleset generation are discussed in Section 5. The strength of our methods is demonstrated on a few examples in Section 6, followed by the applicability of the methods in MCDA in Section 7. Finally some conclusions and further research plans are mentioned in Section 8.

2 MCDA: The classification problem

In this section we consider the problem statements of MCDA, list some methods and define the discriminant model we will use later.

When considering a discrete set of alternatives described by some criteria, the following problem statements can be distinguished in order to provide support for decision makers (DM): identifying the best alternative or selecting the best ones (choice problem), assigning alternatives to predefined categories (classifying or sorting problem), ordering alternatives (ranking problem) and providing the performance tableau, identifying the major distinguishing features of the alternatives (description problem)[3,4]. However, both classification and sorting refer to assignment of alternatives into predefined categories, and they differ only in the way the groups are defined. In the classification problem, groups are defined in a nominal way while in the sorting problem statement they are defined in an ordinal way, i.e. the order of the classes are important[5].

To solve the above mentioned tasks, many different methods have been developed. Outranking methods aim to construct a binary relation (often non-transitive, non-complete) among actions/alternatives, using pairwise comparisons. They are frequently called as ELECTRE type methods[6–8,4]. In a value focused approach, the aim is to construct a unique function aggregating the partial preferences on multiple criteria. For example, Multiattribute Utility Theory (MAUT)[9], Utility Additive Multicriteria Method (UTA)[10], and Analytic Hierarchy Process (AHP)[11] are value focused approaches. Many such methods construct additive utility functions, though multiplicative or other utility functions can also be derived. Most commonly applied procedures are inferred from these methods (see also [12,8,4]). A very interesting critique of decision models is found in [13], where a number of models are considered and the existence of a universal model is discussed. Other models have also been suggested for decision support such as Rough Sets[14,15] and methods inferring models from examples with machine learning techniques, for example, decision rules, decision trees or artificial neural nets.

In many such methods, the analyst has to determine values of several parameters to tune the model and to fit it to the decision maker's preferences. Since it is not realistic to assume that the decision maker is able to express his/her preferences in parameters, several methods are developed which infer the parameters through analysis of assignment examples[6]. Our proposed method also falls into this group: it recommends a solution on the problem statement of multicriteria classification. It infers a model from examples, preclassified by the decision maker.

Now, we describe the classification and sorting problems formally. Assume

that a finite number of observed *alternatives* (actions, objects): $A = \{a, b, \dots\}$ are assigned into predefined groups (classes). Furthermore, the alternatives are described by a set of *criteria* or *attributes*: g_1, \dots, g_m and any two alternatives can be compared according to the attributes: $g_j : A \rightarrow \mathfrak{R}$ (see Table 1).

In MCDA, the task of the analyst is to understand the preference relation of the decision maker and to develop a model that describes this relation and helps to make decisions in further situations. The model can be an outranking relation, a utility function or a discriminant function. In a multi-attribute utility/value model, the decision maker's preference relation on the set of alternatives relates to values given by a function $u : A \rightarrow \mathfrak{R}$:

$$a \succsim b \iff u(a) \geq u(b)$$

The value function u is the result of the aggregation of the value functions assigned to each criteria (u_i):

$$u(a) = M(u_1(g_1(a)), \dots, u_m(g_m(a)))$$

The simplest, and most commonly used, aggregation function is the weighted sum:

$$u(a) = \sum_{i=1}^m w_i u_i(g_i(a)) \tag{1}$$

but other aggregation functions can also be applied[16].

Note that in this simple decision-aid model, both the values of alternatives on a criterion (g_i) and the importance of the criteria (w_i) are given by numerical (continuous) values.

The parameters of the aggregation function (the weights and the utility functions) are set either in a direct or an indirect way. In the direct approach, weights are given by the user, while in an indirect method, preference information is collected through a pairwise comparison of the previously selected alternatives. The weights and the utility functions are estimated using results of the comparisons. An alternative way of setting parameters of the model is using Machine Learning methods as in the modified UTA method[10], in Rough Sets[14] and in our proposed CDT method.

In order to apply the model (1) for classification, a threshold δ have to be set to separate the alternatives:

Table 1

Description of the input dataset. Alternatives are described by a set of attributes and the class they belong to.

	attribute 1	...	attribute m	class
	(w_1)		(w_m)	
alternative a	$g_1(a_1)$...	$g_m(a_m)$	$c(a)$
alternative b	$g_1(b_1)$...	$g_m(b_m)$	$c(b)$
\vdots	\vdots		\vdots	\vdots

$$\sum_{i=1}^m w_i u_i(g_i(a)) > \delta. \quad (2)$$

The geometric interpretation of (2) is the following: get an alternative a and check whether the point $(g_1(a), \dots, g_m(a))$ in \mathfrak{R}^m is above the hyperplane defined by the normal vector (w_1, \dots, w_m) and the coefficient δ or not.

Discriminant function models are formally very similar to the utility function models

$$f(\mathbf{a}) = \sum_{i=1}^m w_i g_i(a_i). \quad (3)$$

Here, however, all attributes are quantitative, thus the qualitative attributes have to be quantified. On the other hand, the weights/coefficients are unrestricted in sign. In this sense, our CD method is close to this type of multicriteria models.

In the following section, we define the concept of the decision tree and describe some tree building methods.

3 Related work

In this section, we consider classifiers from a Machine Learning point of view, and define the concept of the decision tree. We also mention some tree building algorithms (ID3, C4.5) with their properties. Our CDT method, introduced in the following section, is based on these approaches.

3.1 Decision tree classifiers

Classification models can be grouped by the way they are constructed: they are either made by human experts or are inductively obtained from a set of examples. The induced model is either non-hierarchical (e.g. instance-based

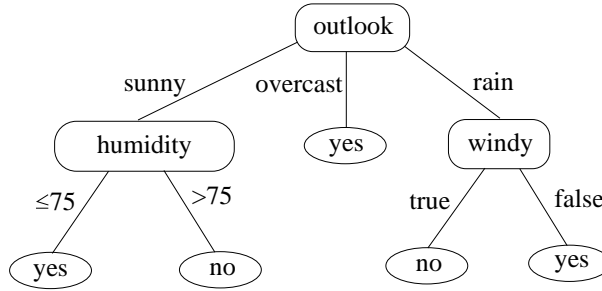


Fig. 1. Decision tree built up from the training set in Table 2 using the C4.5 method

classifiers, or models obtained from a neural network, genetic algorithm, statistical method) or hierarchical, such as decision trees (for a short overview and further references see [17,18]). We will derive a hierarchical classifier which is constructed inductively.

In the following, T denotes the given set of elements with their class information (training set) from which a decision tree is induced:

$$T = \{(\mathbf{x}, c(\mathbf{x})) | \mathbf{x} \in X\},$$

where $\mathbf{x} \in X$ is described by a sequence of attribute values: $\mathbf{x} = (x_1, \dots, x_m)$, x_i is the value of the i th attribute, m is the number of attributes. $c(\mathbf{x})$ indicates the class of element \mathbf{x} . Denote C_1, \dots, C_k the possible classes of elements in T .

From a machine learning point of view, two different types of attributes are distinguished: an attribute is either discrete (categorical), i.e. its value comes from a predefined finite set, or continuous (numerical), i.e. it is an element of an (real) interval.

A classifier is a model built from the training dataset, and is applied later to predict class values of unseen elements. The model is based on the attribute values of the elements. A typical classifier is the decision tree (see Figure 1):

Definition 1 *A decision tree is a special rooted tree, in which a class identifier is associated to each leaf node (it determines the class of elements which reached the node), and each internal (or decision) node specifies a test, with one branch and subtree for each outcome of the test.*

Methods for constructing decision trees might be grouped by the variety of tests used in their inner nodes. In some approaches, only single attribute selection is allowed as a test, for example, in the CART method[18], in Quinlan's ID3 method and in its extension to handle continuous attributes, the C4.5 method[17]. In other methods, some mixture of the attributes are also allowed as tests, for instance, in Cios's CID3[19], in Oblique Decision Trees[20], and

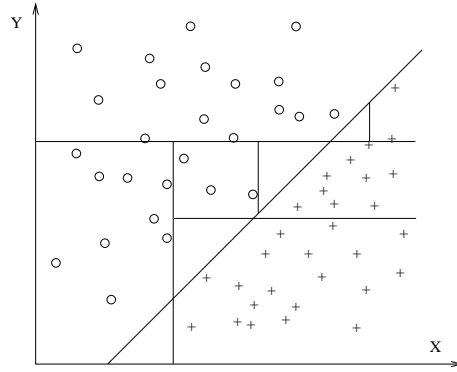


Fig. 2. Geometric interpretation of classification. Elements of the two classes could be separated either by a set of C4.5-type axes-parallel lines or by only one line in arbitrary position as in the CDT method.

in our CDT method.

In the geometric aspect of the problem, the application of tests based on single attribute selection can be interpreted as a partition of the decision space – which is described by a set of continuous attributes – with hyperplanes whose edges are parallel to the axes. A mixture of attributes leads to hyperplanes in arbitrary positions. In our CDT method, arbitrary figures can be applied for separation of the space while in C4.5 only axis-parallel hyperplanes are used (Figure 2).

In order to construct a decision tree, a training set is given, where elements are described by some (discrete or continuous) attributes, and a tree is searched for in the hypothesis space which fits the elements of the training set. It is easy to find one that is consistent with the training set[1]: for example, a tree which contains only one element of the training set in all leaf nodes. This tree, however, does not tell us anything about the structure of the original problem and has very poor predictive power. According to the principle of Occam's razor, it is worth looking for one of the smallest decision trees. The motivation is that a simple decision tree (less complex model) might perform better on unseen elements than a more complicated one[21,2]. However, the problem of finding the smallest decision tree consistent with a training set is NP-Complete[22].

To cope with this computational problem, usually a greedy divide-and-conquer algorithm is used to build a decision tree consistent with the training set (which, of course, does not lead to the smallest one in the general case)[23,17]. The generic algorithm is presented in Figure 3. Initially, the one-node decision tree is considered (containing all elements of the training set). As the algorithm proceeds, in each step a test is chosen and is applied to the examples that have reached the examined node. This test is then assigned to the node and branches are created according to the outcomes of the selected test. Finally,

```

function BuildTree(examples,  $C_{\text{def}}$ )
  return a decision tree
  input: examples: set of examples with
           classes  $C_1, C_2, \dots, C_k$ 
            $C_{\text{def}}$ : default class
  if all examples have class  $C_j$  then
    return leaf with title  $C_j$ 
  else if examples is empty then
    return leaf with title  $C_{\text{def}}$ 
  else
    test := select a test to separate examples
    make an inner node with test
    for each outcome  $O_i$  of test
      examplesi := elements of examples which outcome of test is  $O_i$ 
      determinate value of  $C_{\text{def}}$ 
      subtreei := BuildTree(examplesi,  $C_{\text{def}}$ )
    return tree

```

Fig. 3. The greedy algorithm used to build a decision tree from a training set of examples

the same method is recursively applied to the newly created branches. The crucial point of this algorithm is the test selection criterion. The methods discussed in the present paper (ID3, C4.5, CDT) differ at this point. In all three approaches, the test selection criterion is based on the homogeneity of the training set according to the class values. In ID3 and C4.5 the measure of homogeneity is based on the entropy function[24], while in CDT a different measure is applied, derived from fuzzy conjunction operators.

In the following, the commonly applied ID3 and C4.5 decision tree building algorithms are discussed.

3.2 The ID3 and the C4.5 methods

As mentioned earlier, the crucial points of the tree-building algorithm are the variety of possible tests and the test selection criterion. In the ID3 and the C4.5 methods, tests are based on single attribute selection, so the possible tests are related to the possible attributes.

Test selection examines the training examples and finds the attribute that separates the examples most perfectly considering their class membership. The ID3 algorithm uses a function from the field of information theory, the entropy, to measure how separated the elements are in the original training set

Table 2

The training set that specifies when to go playing

outlook	Temp(F)	Humidity(%)	Windy?	Class
sunny	75	70	true	Play
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play
rain	65	70	true	Don't Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

and in the subsets after partitioning. Formally, the criterion is the following:

$$\max_A \{Gain(A)\} \quad (4)$$

where

$$Gain(A) = I(T) - E(A, T) \quad (5)$$

gives the improvement in the entropy, the gain,

$$I(T) = - \sum_{j=1}^k \frac{|C_j|_T}{|T|} \cdot \log_2 \left(\frac{|C_j|_T}{|T|} \right) \quad (6)$$

is the entropy function ($|C_j|_T$ denotes the number of elements of T belong to class C_j) and

$$E(A, T) = \sum_{i=1}^m \frac{|T_i|}{|T|} \cdot I(T_i) \quad (7)$$

is the weighted sum of the entropies in the subsets. Note that the application of entropy function and the *Gain* criterion, which is just the simple difference of $I(T)$ and $E(A, T)$, has no strong theoretical background, and they are chosen subjectively as one of many possible solutions.

Unfortunately, the *Gain* criterion is biased towards discrete attributes with more outcomes. To remedy the situation, the *Gain ratio* test selection criterion

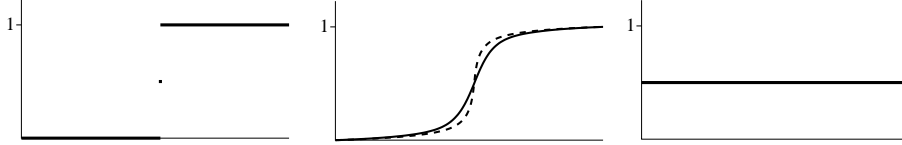


Fig. 4. Membership functions with different sharpness

is proposed[17], which decreases the *Gain* in case of many-valued discrete attributes.

The ID3 method is applicable only in problems described by a set of discrete attributes. In its extension to handle continuous attributes (C4.5), the attribute values are discretized, treated as discrete ones with two outcomes in the following way: different values of the candidate continuous attribute are ordered: $v_1 \leq v_2 \leq \dots \leq v_n$ and all

$$m_i = \frac{v_i + v_{i+1}}{2}, \quad i = 1, \dots, n - 1 \quad (8)$$

midpoints are checked as possible thresholds to divide the training set into two partitions. This test selection criterion in C4.5, however, is biased towards continuous attributes with numerous distinct values. Its analysis and a modified criterion are presented in [25].

To demonstrate the ID3 and C4.5 methods, Figure 1 shows a decision tree constructed from a small training set (Table 2). The example, well-known from the literature[17,2], shows when to play a specific game. The test selection criterion is maximization of *Gain*, so first the attribute 'outlook' is chosen. Then the algorithm is called recursively to the first and third branches of the tree. Finally, a decision tree is built in which all training elements are classified correctly.

In the following section, we discuss the selection criterion applied in the CD and the CDT methods.

4 A new measure of separation

This section, first, deals with the basic concepts of the general class of fuzzy operators and the measure of fuzziness. Next, we propose a new measure replacing the standard entropy-based measures, which leads to our new model, CDT. Finally, the similarity of the two measures is demonstrated on a small example.

Fuzzy theory has two main classes of associative subidempotent conjunction

operators (t-norms): (i) the class of strictly monotonic and (ii) the class of nilpotent operators[26,16]. The strictly monotonic conjunction operators can be written in form $c(x, y) = f^{-1}(f(x) + f(y))$ where $f(x) : (0, 1] \rightarrow \mathfrak{R}^+$, $\lim_{x \rightarrow 0^+} = \infty$, $f(1) = 0$ and $f(x)$ is a strictly decreasing function. The nilpotent operators have the form $c(x, y) = f^{-1}([f(x) + f(y)])$ where $[x]$ is the identity function restricted to $[0, 1]$:

$$[x] = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x \leq 1 \\ 1 & \text{otherwise} \end{cases}$$

and $f(x) : [0, 1] \rightarrow [0, 1]$ is a strictly decreasing function with $f(0) = 1$, $f(1) = 0$. These general forms define a fuzzy conjunction operator for each appropriate generator function $f(x)$.

The fuzziness of a (membership) function may differ significantly. For example, Figure 4 shows some different membership functions. The first function switches very sharply from 0 to 1, so it is not considered fuzzy. In the second graph, the dashed line shows a less fuzzy while the straight line shows a more fuzzy function. The third graph presents a completely fuzzy function. In the case of membership functions defined only at discrete points, fuzzyness is measured by the following formula[26]:

$$S = \frac{1}{n} \sum_{i=1}^n F(x_i) \tag{9}$$

where

$$F(x) = \frac{1}{c\left(\frac{1}{2}, \frac{1}{2}\right)} \cdot c(x, 1 - x) \tag{10}$$

and $c(x, y)$ is a conjunction operator (t-norm). This function measures how much the elements of the training set are separated. It returns small values (close to 0) in the first case in Figure 4 and large (close to 1) ones in the third case.

The entropy function, which is used in the ID3 and the C4.5 methods, can be considered as a special measure of fuzziness[27]. It is obtained from the general formula (9) of fuzziness measures as a special case, where $c(x, y)$ is a nilpotent conjunction operator. Therefore, the entropy function is consistent with nilpotent fuzzy operators[28].

Our measure is derived from (9) using the weighted form of Dombi's conjunction operator (which is a strictly monotonic operator)[26]:

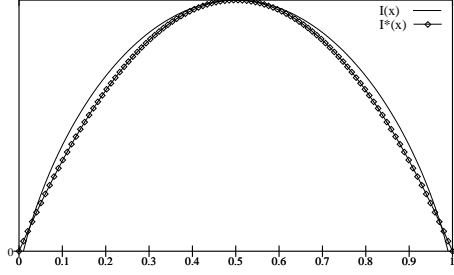


Fig. 5. Entropy function for two classes and the new $4x(1-x)$ measure

$$c_\lambda(x, u; y, v) = \frac{1}{1 + \left(u \left(\frac{1-x}{x} \right)^\lambda + v \left(\frac{1-y}{y} \right)^\lambda \right)^{\frac{1}{\lambda}}} \quad (11)$$

where $u + v = 1$ and λ influences the type of the operator. If $\lambda > 0$ this operator behaves as a conjunction and if $\lambda < 0$ it behaves as a disjunction.

Note that the pliant operator $c_\lambda(x, u : y, v)$ is derived from the generalized form of mean values[29], which is closely related to the general form of fuzzy operators:

$$f^{-1} \left(\sum_{i=1}^m w_i f(x_i) \right), \text{ where } \sum_{i=1}^m w_i = 1.$$

The following measure of fuzziness is derived from (10) using the pliant conjunction operator with $u = v = \frac{1}{2}$ and $\lambda = 1$:

$$I^*(x) = \frac{1}{c\left(\frac{1}{2}, \frac{1}{2}\right)} \cdot c(x, 1-x) = 4x(1-x). \quad (12)$$

This formula is simpler than the entropy function and it behaves in a very similar way. Graphs of the two functions are shown in Figure 5. It is easy to prove that, in the case of concept learning (i.e. when the examples fall into two classes), these two functions lead to the same result, since on the interval $[0, 1]$ both functions have maxima at $\frac{1}{2}$ and minima at $0, 1$, and both are strictly monotonic on $[0, \frac{1}{2}]$ and on $[\frac{1}{2}, 1]$.

The calculations are demonstrated on the training set (Table 2), where the 14 elements are described by four attributes (two discrete and two continuous) and the examples fall into two classes.

The entropy of the whole training set is

$$I(S) = -\frac{9}{14} \log \frac{9}{14} - \frac{5}{14} \log \frac{5}{14} = 0.9403.$$

Applying the new $I^*(x)$ measure gives

$$I^*(S) = 4 \cdot \frac{9}{14} \cdot \frac{5}{14} = 0.9184.$$

According to the algorithm in Figure 3 and the test selection criteria (5), the *Gain* of all possible separations has to be calculated. For the attribute 'outlook', the entropies of the three subsets (sunny, overcast, rain) are $I(S_{sunny}) = 0.9709$, $I(S_{overcast}) = 0$, $I(S_{rain}) = 0.9709$, so the weighted sum of these values (average entropy) is $E(outlook, S) = \frac{5}{14}0.9709 + \frac{4}{14}0 + \frac{5}{14}0.9709 = 0.6935$. Therefore, $Gain(outlook) = 0.9403 - 0.6935 = 0.2468$, which means that 0.2468 is gained if attribute 'outlook' is selected. The new measure leads to the following values: $I^*(S_{sunny}) = 0.96$, $I^*(S_{overcast}) = 0$, $I^*(S_{rain}) = 0.96$, and the weighted sum of these values is $E^*(outlook, S) = 0.6857$. Therefore, $Gain^*(outlook) = 0.2327$.

In the following steps, the *Gain* of all other attributes have to be calculated in order to find the attribute with the highest *Gain* value. Since the attributes 'temp' and 'humidity' are continuous (numeric), the discretization step (8) of the C4.5 method has to be applied. The attribute 'outlook' leads to the highest *Gain*, consequently three branches of the root node have to be created. The algorithm is called recursively for all new branches and finally the tree in Figure 1 is obtained.

In the following section, we discuss the soft-counting technique, the Continuous Decision and the Continuous Decision Tree methods.

5 The CD and the CDT methods

In this section, we propose a new method, the Continuous Decision (CD) method, which learns the coefficients of a (linear) discriminant function (3), followed by its extension, the CDT method, that is applicable in complex problems. Finally, the process of ruleset generation is discussed.

5.1 Soft counting and the CD method

The CD method is based on a global optimization task, that looks for the hyperplane that separates elements of the training set. We assume, that the elements belong to two classes, denoted by positive (+) and negative (-). In order to apply continuous optimization methods, soft-counting of elements is used.

Instead of the simple addition of the number of elements on one side of the separating hyperplane (strict counting) the sigmoid function is applied:

$$\sigma_1(\mathbf{x}, \mathbf{a}) = \frac{1}{1 + e^{-\lambda f(\mathbf{x}, \mathbf{a})}} \quad (13)$$

where $f(\mathbf{x}, \mathbf{a}) = 0$ defines the separating function in question, vector \mathbf{a} contains the parameters of the separating function and λ gives the sharpness at the boundaries of the function. As a result of soft-counting, examples close to the boundary are counted for both sides (with different membership values). We use the sigmoid function, well known in Artificial Neural Networks, for it is easy to interpret and calculate.

The ratio of the examples belonging to class $C \in \{+, -\}$ and falling in one side of the considered separating function $f(\mathbf{x}, \mathbf{a})$ is the following:

$$S_1^C(\mathbf{a}) = \frac{1}{|C|_T} \sum_{c(\mathbf{x}_i)=C} \sigma_1(\mathbf{x}_i, \mathbf{a}). \quad (14)$$

Soft-counting of examples leads to a continuous global optimization problem. The extreme points of $S_1^C(\mathbf{a})$ are searched, where all variables \mathbf{a} are continuous and preferred to be bounded. If the elements of the training set are linearly separable, and $f(\mathbf{x}, \mathbf{a})$ is linear, this method gives coefficients of a separating hyperplane, otherwise it gives a hyperplane that approximately separates the positive and negative training examples.

The hyperplane representing the linear discriminant function is described by bounded parameters in the following way: all training examples given by m continuous attributes, are represented as points in \mathfrak{R}^m (Figure 6). R_{max} denotes the distance of the farthest points of the bounding box from the center. Each inner point P of the bounding sphere determines a hyperplane, with normal vector \vec{OP} and the point P lying on the hyperplane. This hyperplane separates elements of the training set into two subsets:

$$f_s(\mathbf{p}, \mathbf{o}, \mathbf{x}) = \mathbf{n}(\mathbf{x} - \mathbf{p}) = \frac{\mathbf{p} - \mathbf{o}}{\|\mathbf{p} - \mathbf{o}\|_E}(\mathbf{x} - \mathbf{p}) = 0$$

where $p_i \in (O_i - R_{max}, O_i + R_{max})$.

If we use hyperplane separation, the function (14) has m variables, where m is the number of attributes considered. All the variables are bounded and the soft-counting of elements leads to a continuous function with many small hills. The surface depends on the distribution of the training examples and their classes.

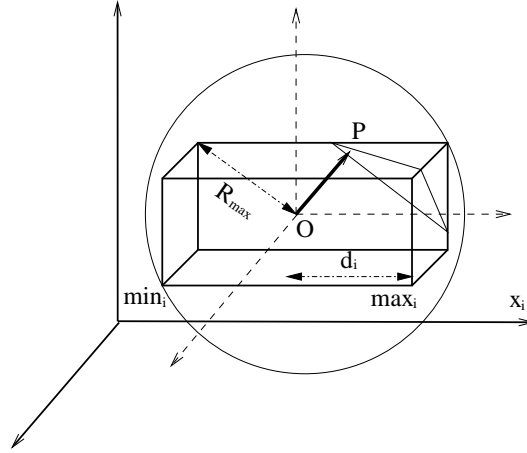


Fig. 6. Description of a hyperplane, that separates the training elements, with bounded parameters.

Many different methods are applicable to solve the described global optimization problem. In the current implementation of the algorithm, a stochastic method is used[30], but further investigation is needed to find the most appropriate approach.

Although we introduced the CD method, which consist of the soft counting of examples and the global optimization approach, using a hyperplane to separate the elements of the training set, the method could be applied to find not only hyperplanes, but any geometric figures: only the function $f(\mathbf{x}, \mathbf{a})$ has to be modified in (13) appropriately. In this way, either quadratic separability or separation by spheres or ellipsoids could be solved.

In the following section the straightforward extension of the CD method is introduced, which leads to a decision tree where all inner nodes have a (linear) discriminant function.

5.2 The CDT method

The CD method, introduced in the previous section, searches a separating hyperplane or any geometric object, depending on the function $f(\mathbf{x}, \mathbf{a})$ in (14). This could be considered as a one-level decision tree. The CDT method is its extension, where each inner node of the tree contains a separation.

The CDT method is based on the greedy decision tree building algorithm in Figure 3 and the *Gain*-type test selection criterion (5)–(7) derived from the measure $I^*(x)$ introduced in Section 4. The result of our calculations for $E^*(t, S)$ is shown in Figure 7.

Recall that in the CD method not only hyperplanes but any geometric figure

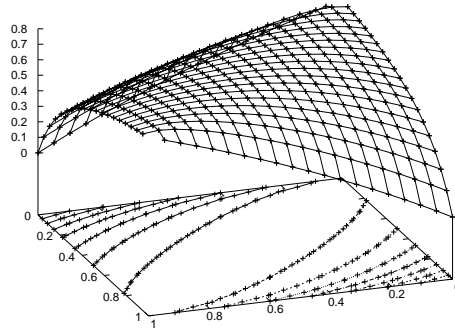
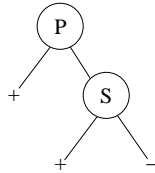


Fig. 7. The test selection criterion. The CDT method walks on this surface during construction of the tree as long as the measure of separation $I^*(x)$ is reduced according to the *Gain* criterion. The axes x and y represent the portion of positive and negative training examples on one side of the geometric figure.



$$\text{Ruleset}(+) = \{P(\mathbf{w}, \delta), \neg P(\mathbf{w}, \delta) \wedge S(\mathbf{o}, R)\}$$

Fig. 8. Ruleset generation from a decision tree. The ruleset $\text{Ruleset}(+)$ is the disjunction of two root-to-leaf paths. The second expression consists of the conjunction of two separations: the first one is a separation by hyperplane P , and the second one is a separation by a sphere S .

can be searched. It also hold for the CDT method, which checks all the possible tests (i.e. all the possible functions $f_i(\mathbf{x}, \mathbf{a})$) and chooses the one with the highest *Gain* value. The selected function will be assigned to the precessed node of the decision tree.

Although the CDT method has the drawback that it is applicable only for concept learning (i.e. elements belong to two classes), it allows the elements of \mathfrak{R}^n to be separated by arbitrary geometric figures. This is the main difference between the C4.5 decision tree building algorithm, which separates the decision space with hyperplanes parallel to the axes, and the CDT method. In the present implementation, soft-counting of elements is used, and the parameters of the figures separating the points are found by a stochastic global optimization approach.

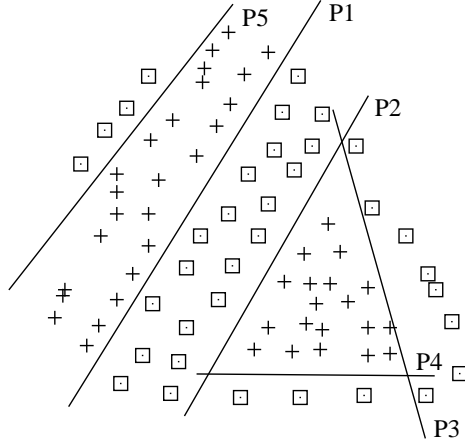


Fig. 9. The CDT method: separation of positive and negative elements in the extended triangle problem

5.3 Ruleset generation

The constructed tree can be transformed into rules in the following way: from the root node to a leaf construct conjunctions of *tests* given in the inner nodes, and make disjunction of expressions assigned to all paths from the root to a leaf. The ruleset that describes the + examples in Figure 8 is the following:

$$\text{Ruleset}(+) = \{P(\mathbf{w}, \delta), \neg P(\mathbf{w}, \delta) \wedge S(\mathbf{o}, R)\}$$

The second expression in a more detailed form:

$$\neg P(\mathbf{w}, \delta) \wedge S(\mathbf{o}, R) = \left(\sum_{i=1}^m w_i x_i > \delta \right) \wedge \left(\sum_{i=1}^m (x_i - o_i)^2 < R^2 \right)$$

The generated ruleset is more readable, easier to understand and it can be applied to make inferences. Note that in Rough Sets theory the results are also transformed into rules[14].

6 Examples

In this section, we demonstrate the applicability of the CDT method on a few simple examples.

Consider the training set in Figure 9. This is the triangle problem extended with a few extra points. In the example only lines are allowed to separate

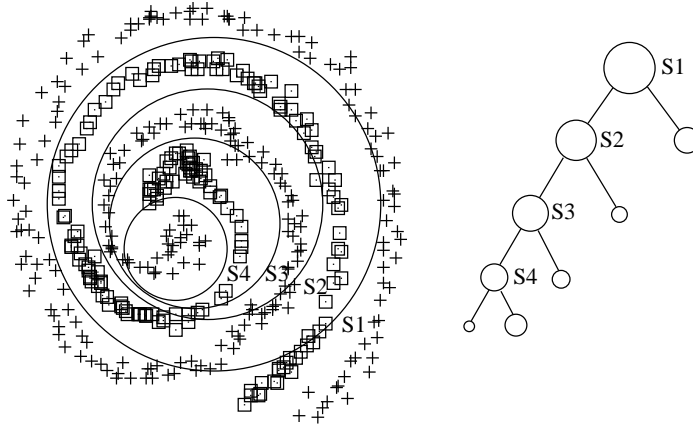


Fig. 10. The CDT method: separation of positive and negative elements by circles in the two spirals problem

elements of the 2-dimensional training set. The separation of the triangle is correctly solved by five lines, while the C4.5 method requires about ten decisions. (A similar example is discussed in [19]).

Another well-known difficult task is the the problem of two spirals[19], illustrated in Figure 10. In the solution of this problem, the points are separated by circles instead of lines. This is the first test to apply the CDT method with other type of geometric figures. On the right hand side of the figure, only the first few steps of the tree building process are shown because the solution leads to a large complex decision tree. Elements of the training set have been quite well separated in the very first steps of the method. The circle type decisions also gives an alternative interpretation of the problem: the spiral can be approximated with a few circles. The C4.5 method gives a very complex decision tree solving the task.

Since in the Iris plant problem (150 elements dataset, 4 continuous attributes and 3 equally distributed classes[31]) the elements of the training set belongs to more than two classes, the CDT method separates elements of one class from the rest of the dataset. Any class is separated from the others only by one hyperplane decision, even though the attributes are correlated. Therefore the CDT method produces a compact tree. This method also gives an impression of the layout of points representing the elements of the training set in \mathbb{R}^4 .

Finally, in a training set, where most of the points are classified into class + and a few points in a small group into class -, in the CDT method the class - elements are separated by a single m -dimensional sphere, where m is the number of attributes. In the C4.5 method $2m$ separating hyperplanes are required to solve this problem, that encapsulates the class - elements into a hyperbox. The CDT method can even find the 'exceptions' or 'errors' in a

dataset.

These simple examples show the applicability of the CDT method in two-class classification problems, when all the attributes are continuous (numeric) or could be transformed to continuous ones.

7 Application in MCDA

In Section 2, the classification problem of MCDA was introduced. The models based on utility functions and the linear discriminant models are very similar to the single hyperplane learning CD method introduced in Section 5.1. It can be applied to learn coefficients of the linear discriminant functions. However, the method is generalized, and any type of discriminant functions is possible to learn – quadratic functions, spheres, ellipsoids, etc.

Moreover, in case of complex problems when the alternatives classified by the decision maker are not separable by a simple model, the decision tree approach is applicable. This approach, leads to a more accurate and more complex model. In this way the classification of alternatives can be approximated arbitrarily with a sequence of interpretable separations.

The hyperplane type decisions might be interpreted as coefficients in a linear discriminant model (3), or as the importance of attributes/criteria in additive utility function models (2). The sphere or ellipsoid type decisions might be interpreted as a typical alternative (center of the sphere or ellipsoid) and its neighbors (radius).

These types of decisions are a slightly more complex than the ones usually applied in MCDA models, but the CDT algorithm leads to a more compact description of a classification problem.

8 Conclusions and future perspectives

This paper presents a novel decision tree building algorithm (CDT) as the generalization of the C4.5 algorithm. In C4.5 the continuous decision space is separated only by hypercubes whose edges are parallel with the coordinate axes, while in our new method arbitrary geometric figures like hyperplanes, spheres and ellipsoids are allowed. The advantage of this improvement has been demonstrated through some examples. Although this modification increases the time-complexity of the algorithm, it also enlarges the variety of possible tests and leads to more compact trees.

The entropy function is replaced with a simpler measure (using a strictly monotone fuzzy operator in the general measure of fuzziness) and the test selection criterion is derived from this measure. This simplification slightly reduces the time-complexity of the algorithm.

In our method, the elements of the training set are soft-counted, which means that the elements close to the boundary of the figure belong to both regions (with different membership values). Therefore, the decision criterion is a continuous function of the parameters of the geometric figures. This property makes the global optimization problem (finding parameters of geometric figure) simpler and more reliable.

There are some restrictions in the current version of our method. Only continuous attributes are considered. It is also possible to handle ordered discrete attributes, but obviously it is impossible to handle unordered ones in this way. All elements of the training set belong to two classes, so only concept learning is examined. All the tests applied in the inner nodes of the decision tree separate the training set into two groups. Multivalued tests are not possible.

The Continuous Decision Tree method fits to the classification problem of MCDA. Either the importance of criteria in a utility function model or coefficients in a discriminant function can be learnt from preclassified alternatives. Even more complex classification models can be determined based on the decision trees or their transformation to rules.

A few improvements of the CDT method are planned. Sometimes, the constructed decision trees are too complex, so it is beneficial to apply some pre- or post-pruning[32,33] method to simplify the resulting tree. In the implementation of the algorithm, replacing the general purpose stochastic global optimizer with a problem specific one might prove to be useful. The running time on large datasets might be rather long, so sampling technique should be used to construct a tree.

Acknowledgement

We would like to thank all the efforts of the unknown referees and all the useful suggestions and remarks they made on the early version of this paper. It helped us much to improve both the level and the structure of the paper.

References

- [1] S. Russel, P. Norvig, *Artificial Intelligence - A Modern Approach*, Prentice-Hall, Englewood Cliffs, 1995.
- [2] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.
- [3] B. Roy, Decision science or decision aid science?, *European Journal of Operational Research* 66 (1993) 184–203.
- [4] T. Gal, T. Steward, T. Hanne (Eds.), *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications*, Kluwer Academic Publishers, 1999.
- [5] C. Zopounidis, M. Doumpos, Multicriteria classification and sorting methods: A literature review, *European Journal of Operational Research* 138 (2002) 229–246.
- [6] V. Mousseau, R. Slowinski, Inferring an ELECTRE TRI model from assignment examples, *Journal of Global Optimization* 12 (1998) 157–174.
- [7] B. Roy, The outranking approach and the foundations of ELECTRE methods, *Theory and Decision* 31 (1991) 49–73.
- [8] D. Olson, *Decision Aid for Selection Problems*, Springer, 1996.
- [9] R. Keeney, H. Raiffa, *Decisions with multiple objectives – Preferences and value tradeoffs*, John Wiley and Sons, New York, 1976.
- [10] M. Beuthe, G. Scannella, Comparative analysis of UTA multicriteria methods, *European Journal of Operational Research* 130 (2001) 246–262.
- [11] T. Saaty, *The analytic hierarchy process*, McGraw Hill, 1980.
- [12] S. French, *Decision Theory: An Introduction to the Mathematics of Rationality*, Ellis Horwood, Chichester, 1988.
- [13] D. Bouyssou, et al., *Evaluation and Decision Models: a critical perspective*, Kluwer Academic Publishers, Boston, 2000.
- [14] S. Greco, B. Matarazzo, R. Slowinski, Rough sets theory for multicriteria decision analysis, *European Journal of Operational Research* 129 (2001) 1–47.
- [15] Z. Pawlak, R. Słowinski, Decision analysis using rough sets, *International Transactions in Operational Research* 1 (1994) 107–114.
- [16] J.-L. Marichal, *Aggregation operators for multicriteria decision aid*, Ph.D. thesis, University of Liège (1999).
- [17] J. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.
- [18] L. Breiman, J. Friedman, R. Olshen, C. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1984.

- [19] K. Cios, N. Liu, A machine learning method for generation of a neural network architecture: a continuous ID3 algorithm, *IEEE Transactions on Neural Networks* 3 (2) (1992) 280–291.
- [20] S. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, *Journal of Artificial Intelligence Research* 2 (1994) 1–32.
- [21] M. Kearns, U. Vazirani, *An Introduction to Computational Learning Theory*, The MIT Press, Cambridge, Massachusetts, 1994.
- [22] L. Hyafil, R. Rivest, Constructing optimal binary decision trees is np-complete, *Information Processing Letters* 5 (1976) 15–17.
- [23] J. Stefanowski, Classification and decision supporting based on rough set theory, *Foundations of Computing and Decision Sciences* 18 (3-4) (1993) 371–380.
- [24] R. Vetschera, Entropy and the value of information, *Central European Journal of Operations Research* 8 (2000) 195–208.
- [25] J. Quinlan, Improved use of continuous attributes in C4.5, *Journal of Artificial Intelligence Research* 4 (1996) 77–90.
- [26] J. Dombi, A general class of fuzzy operators, the demorgan class of fuzzy operators and fuzziness measures induced by fuzzy operators, *Fuzzy Sets & Systems* 8 (1982) 149–163.
- [27] A. Luca, S. Termini, Definition of non-probabilistic entropy in the setting of fuzzy sets theory, *Information and Control* 20 (1972) 301–312.
- [28] J. Dombi, A fuzzy halmazok operátorainak szerkezete a többtényezős döntések szempontjából, Ph.D. thesis, Attila József University, Szeged (1993).
- [29] G. Hardy, J. Littlewood, G. Pólya, *Inequalities*, 2nd Edition, Cambridge University Press, 1934.
- [30] A. R. Kan, G. Timmer, Stochastic global optimization methods, *Mathematical Programming* 39 (1987) 27–78.
- [31] C. Blake, C. Merz, UCI repository of machine learning databases (1998).
URL <http://www.ics.uci.edu/~mllearn/MLRepository.html>
- [32] F. Esposito, D. Malerba, G. Semeraro, A comparative analysis of methods for pruning decision trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1997) 476–491.
- [33] J. Kay, Comments on esposito et al., *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1997) 492–493.