

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

J. Dombi and Á. Zsiros
Department of Applied Informatics,
University of Szeged, Hungary
H-6701 Szeged, P.O. Box 652
{dombi,zsiros}@inf.u-szeged.hu

DRAFT

Abstract

The classification problematic of multicriteria decision analysis is to model classification of alternatives/actions according to the preferences of the decision maker. These models are based on either outranking relations or utility functions or (linear) discriminant functions. Parameters of these models could be learnt from a preclassified set of alternatives/actions. The proposed CD method gives a recommendation to this problem, while its extension which produces a decision tree could describe the classification in a more complex way, more accurately.

In MCDA usually numerical (continuous) inputs are handled, like value functions or orderings on a real interval. On the other hand most learning methods, from the field of artificial intelligence, can detect relationship 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 attributes or criteria. Our method helps realizing the structure of the input dataset described by a set of numerical criteria in the form of a decision tree, that might be transformed into decision rules.

Some difficulties of the well-known ID3 and C4.5 decision tree building methods are mentioned and solutions are suggested on them. An alternative measure is suggested instead of the entropy function, which comes from the measure of fuzziness using a monotone fuzzy operator. In the proposed continuous decision tree (CDT) method, the continuous criteria are handled without discretization of their values. In the geometric aspect of the problem, our method allows separation of the decision space with arbitrary figures, like hyperplanes, spheres, etc, and these can also be interpreted to the decision maker. The power of our new method is going to be demonstrated on a few examples.

Keywords: Multicriteria Decision Analysis; Classification problematic, Artificial intelligence; Decision tree; C4.5; Fuzzy operators

1 Introduction

The aim of a decision process is to answer two basic questions. The first one is to explain decisions and the second one is to give recommendations how to make a decision under specific circumstances. These are very similar to the goals of the inductive learning[27, 18] approach in the field of artificial intelligence (AI), where in the first step a model is established from past-experiences and later it is applied to predict future situations. This parallelism leads to the possible application of AI methods in decision support systems.

According to Roy[26, 8], when considering a discrete set of alternatives described by some criteria, the following problematics can be distinguished in order to provide support to decision makers (DM): identify the best alternative or selecting the best ones (choice problematic), assigning alternatives to predefined categories (classifying or sorting problematic), ordering alternatives (ranking problematic) and providing the performance tableau, identify the major distinguishing features of the alternatives (description problematic). However both classification and sorting refer to assignment of alternatives into predefined categories, they differ with respect to the way the groups are defined. In the classification problematic groups are defined in a nominal way while in the sorting problematic they are defined in an ordinal way[30].

To solve the above mentioned problematics many different methods have been developed. Outranking methods aim to construct a binary relation (often non-transitive, non-complete) among actions/alternatives, proceeded by pairwise comparisons of them. These are frequently mentioned as ELECTRE[25] type methods[19]. See also [21, 8]. 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)[15], Utility Additive (UTA) multicriteria method[1] and Analytic Hierarchy Process (AHP)[28]. Many such methods construct additive utility functions, but multiplicative or other utility functions can also be derived. For other methods and further references see also [7, 21, 8]. A very interesting critique of decision models is found in [2] where many commonly used models are considered and the existence of a universal one is questioned. Most commonly employed procedures come from the preceding typed ones, however others have also been suggested like Rough Sets[9, 22] or other methods inferring models from examples with machine learning techniques: decision rules, decision trees or artificial neural nets.

In many above mentioned methods the analyst have to determine values of several parameters to tune the model, fit it to the preferences of the decision maker. While it is not realistic to assume that the decision maker is able to express his/her preferences in parameters, there are methods which infer them through analysis of assignment examples[19]. Our proposed method also falls in this group, it recommends a solution on the multicriteria classification problematic. It infers a model from examples, preclassified by the decision maker.

Multicriteria classification and sorting methods have numerous practical applications. They are applied in medicine, pattern recognition, marketing, finan-

cial management, human research management and many other fields[30].

The paper is organized as follows. In section 2, a very short introduction to the notions of multicriteria decision analysis (MCDA) is given and the connection between the classification problematic of MCDA and instance-based learning methods for building classifiers is described. In section 3 the task of classification and some types of classification models are considered. The well-known ID3 and the C4.5 decision tree building methods are described in section 4. The proposed modifications on the measure of separation is given in section 5 and the derived CD and CDT methods for applicability in MCDA are described in section 6 and 7. Some examples are given to show the power of our new method in section 8, the rule generation and applicability in MCDA are investigated in section 9 followed by some conclusions and further plans in section 10.

2 MCDA: the classification problematic

In the classification and sorting problematics of MCDA a finite set of observed *alternatives* (actions, objects): $A = \{a, b, \dots\}$ are assigned into predefined groups. In classification problems these groups are nominal, while in sorting problems they are ordinal. The alternatives are described by a set of *attributes* or *criteria*: g_1, \dots, g_m depending upon the domain is unordered or ordered and any two alternatives can be compared by any attribute/criteria: $g_j : A \rightarrow \mathfrak{R}$ (see table 1).

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

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

The u value function is 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 commonly used, aggregation function is the weighted sum:

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

but other more complex aggregations could also be considered.

It is worth to notice that in this simple decision-aid model both values of alternatives on a criterion (g_i) and importances of criteria (w_i) are given by numerical (continuous) values.

The parameters of the aggregation function are set either on a direct or an indirect way (for more about aggregation in the mathematical point of view

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

Figure 1: Description of the input dataset. Alternatives are described by a set of attributes and class they are fall in.

see [17]). In a direct method weights are asked from the user directly, while in an indirect one preference information is collected by comparison of selected alternatives, and weights/utility functions are estimated using results of these pair-comparisons. An alternative way is setting parameters of the model from examples using machine learning methods as in the modified UTA method[1] or Rough sets[9] or in our proposed CDT method.

In order to apply a model (1) for classification a threshold w_0 have to be set for separation of alternatives:

$$\sum_{i=1}^m w_i u_i(g_i(a)) > w_0. \quad (2)$$

The geometric interpretation of (2) as follows: get an alternative a and check whether the point $(g_1(a), \dots, g_n(a))$ in \mathfrak{R}^n is above the hyperplane given by vector (w_0, w_1, \dots, w_n) or not.

The discriminant function models are very similar to the utility function models

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

but all the attributes are quantitative (all the qualitative ones have to be quantified) and the weights/coefficients are unrestricted in sign. In this sense our CDT method is closer to this type of multicriteria models.

In the following section we are going to describe the problem of classification and that frequently arises both in artificial intelligence and data mining tasks. One of the commonly used classifier, the decision tree is also introduced.

3 Classifiers, Decision Tree

A lot of applications on the area of artificial intelligence and data mining lead to a similar task: constructing a classification model based on our knowledge. Classification models help us to understand the underlying structure of a given problem and later they can be used to predict classes of unseen elements.

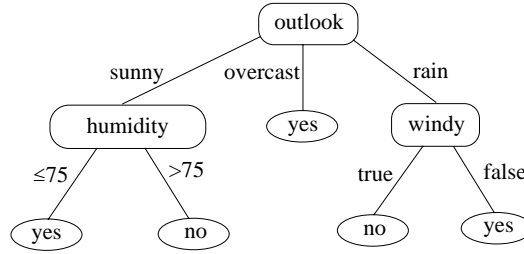


Figure 2: Decision tree built up from the training set on table 1 using the C4.5 method

Classification models could be grouped by the way of construction: they are made by human experts or obtained from a set of examples, inductively. The built up model may be a non-hierarchical one (like a instance-based classifier, or a model obtained from a neural network, a genetic algorithm and a statistical method) or a hierarchical one like a decision tree (for a short overview and further references see [23, 3]). We will mention hierarchical classifiers built up inductively.

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

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

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, and $c(\mathbf{x})$ shows the class of element \mathbf{x} . Denote C_1, \dots, C_k the possible classes of elements in T .

In the machine learning point of view usually two different types of attributes are distinguished: an attribute is discrete (categorical), if its value comes from a predefined finite set; and continuous (numerical), if it may be any element of a (real) interval.

A classifier is a model built from the training dataset and it 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.

Definition 1 *Decision tree is a special rooted tree, in which a class identifier is associated to each leaf node (that determinates the class of elements reached that node), and each internal (or decision) node specifies some test, with one branch and subtree for each outcome of the test. (See for example figure 2.)*

Methods for constructing decision trees might be grouped by the variety of tests used in their inner nodes. In some methods only single attribute selection is allowed as a test, for example in Quinlan's ID3 method and in its extension to handle continuous attributes, the C4.5 method[23]. In other ones some mixture

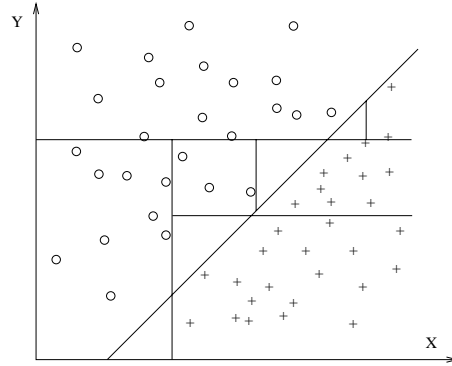


Figure 3: Geometric interpretation of classification. Elements of the two classes could be separated either with a set of C4.5-type axes-parallel lines or with only one line in arbitrary position as in the CDT method.

of attributes are also allowed as tests, for instance in Cios’s CID3[4], in Oblique Decision Trees[20] and in our CDT method.

In the geometric aspect of the problem the application of tests based on single attribute selection or mixture of them can be interpreted as partition of decision space – which is described by a set of continuous attributes – with geometric figures either only with axis parallel hyperplanes or ones in arbitrary position. In our CDT method, arbitrary figures can be applied for separation of the space while in C4.5 only axis-parallel hyperplanes are used, see figure 3.

In order to construct a decision tree a training set given, where elements are described by some (discrete or continuous) attributes and a tree is searched in the hypothesis space which fits to the elements of the training set. It is easy to find such that consistent with the training set[27], take one in which all leaf nodes contain only one element of the training set. But this tree does not tell us anything about the structure of the original problem and have very poor predictive power. According to the principle of Occam’s razor, it worths to look for one of the smallest decision trees. This philosopher idea says that a simple decision tree(less complex model) might perform better on unseen elements than a more complicated one (see [14, 18]). However, *the problem of finding the smallest decision tree consistent with a training set is NP-Complete*[11] as Hyafil and Rivest have shown.

To cope with this computational problem, usually a greedy, divide-and-conquer algorithm is used to build up a decision tree consistent with the training set (which, of course, usually leads not the smallest one), see figure 4. Initially the one-node decision tree is taken (containing all elements of the training set). Later, in each step a test is chosen and applied on the examples reached the examined node. Then this test is assigned to the node and branches are created according to the outcomes of the selected test. Finally, the same method is applied on these newly created branches, recursively. The critical point of

```

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

```

Figure 4: The greedy algorithm used to build a decision tree from a training set of examples

this algorithm is the test selection criterion. The following methods (ID3, C4.5, CDT) differ at this point. In all three methods the test selection criterion is based on the homogeneity of the training set according to the class values. In ID3 and C4.5 it bases on the entropy function[29] while in CDT a different measure is applied, derived from fuzzy conjunction operators. Formally both measures arises in the fundamental book of decision tree construction and application, in the CART method[3].

In the following section the commonly applied ID3 and C4.5 decision tree building algorithms are examined.

4 The ID3 and the C4.5 methods

In this section we are going to give a short description of the ID3 method, of its extension to handle continuous attributes (C4.5) and we are going to demonstrate their work on an example.

As it has been 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 C4.5 method tests are based on single attribute selection, so the possible tests are about the possible attributes.

The idea of test selection is to examine training examples and to find the attribute that separates the examples most perfectly considering their class membership. The ID3 algorithm uses a function coming from the field of information theory, the entropy, to measure how separated the elements in the original

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

Table 1: The training set that specifies when to go playing

training set and in the subsets after partitioning. Formally the criterion is the following (where $|C_j|_T$ denotes the number of elements of T belong to class C_j):

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

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 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. We also remark, 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, it is chosen subjectively as one of many possible solutions.

Unfortunately, the *Gain* criterion is biased towards discrete attributes with more outcomes, thus it has a modification, the *Gain ratio* test selection criterion (see [23]) which reduces the value of *Gain* in case of many valued discrete attributes.

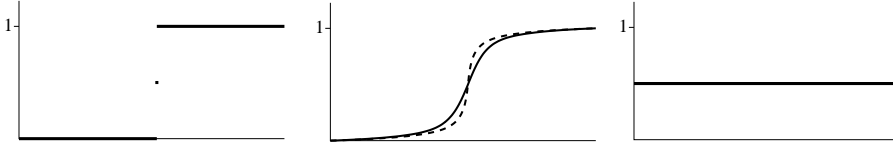


Figure 5: Membership functions with different sharpness

The ID3 method is applicable only on problems described by a set of discrete attributes. Its extension to handle continuous attributes (C4.5) the attribute values are discretized, handled like 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 partition. This test selection criterion in C4.5 is biased towards continuous attributes with numerous distinct values. Its examination and a modified criterion are found in [24].

To demonstrate this method, a decision tree built up from a small training set (see table 1) is given on figure 2. This example is well-known from the literature [23, 18] and 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 the test selection criterion applied in the CDT method and its derivation is introduced.

5 The measure of separation in CD and CDT method

In this section first a few basic concepts are mentioned about the general class of fuzzy operators and the measure of fuzziness. Later the proposed measure is introduced, replacing entropy in CDT. The similarity of the two measures is demonstrated on a small example.

There are two main classes of associative, subidempotent conjunction operators (t-norms) in the fuzzy theory: the class of strictly monotonic and the class of nilpotent operators, see [5, 17]. 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; and nilpotent operators in form $c(x, y) = f^{-1}([f(x) + f(y)])$ where $[x]$ is the identity

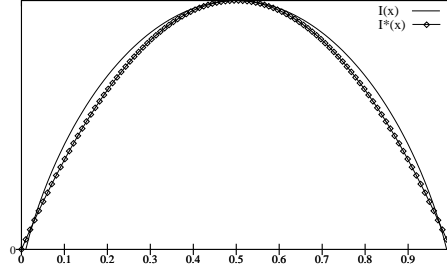


Figure 6: Entropy function for two classes and the new $4x(1-x)$ measure

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

The fuzziness of a (membership) function could differ significantly. For example in figure 5 some different membership functions are shown. The first one makes a very sharp switch from 0 to 1, so it is not considered as fuzzy, on the second one the dotted line shows a less fuzzy while the other one shows a more fuzzy function, while on the third one a completely fuzzy function is given. In case of membership functions defined only at discrete points this property is measured with the following formula[5]:

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

where

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

and $c(x, y)$ is a conjunction operator (t-norm). This function measures how separated the elements of the training set are. It gives small values (close to zero) in the first case in figure 5. and large (close to one) vales in the third case.

The entropy function, which is used in the ID3 and the C4.5 methods, is a special measure of fuzziness[16]. It can also be obtained from the general form (9) of fuzziness measures as a special case, where $c(x, y)$ is a nilpotent conjunction operator. It refers to the fact, the entropy function is consistent with nilpotent fuzzy operators (Dombi).

Our measure is derived from (9) using the weighted form of Dombi's con-

junction operator (which is a strictly monotonic operator)[5]:

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

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

This pliant operator is derived from the generalized form of mean values[10], 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) with the pliant conjunction operator in case of $u = v = \frac{1}{2}$ 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 very similar way. Graphs of the two function are shown on figure 6. It is easy to prove, that these two functions lead to the same result (in case of concept learning, when examples falls into two classes), since on interval $[0, 1]$ both functions have maxima at $\frac{1}{2}$, minima at 0 and 1, both are strictly monotonic on $[0, \frac{1}{2}]$ and on $[\frac{1}{2}, 1]$.

The calculations are demonstrated on the training set on table 1 where the 14 elements training set is described by four attributes (two discrete ones and two continuous ones) and the examples falls into two classes[23].

The entropy of the whole training set in table 1 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

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

According to the algorithm on figure 4 and the test selection criteria (5) the *Gain* of all possible separations have to be calculated. For 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) $E(outlook, S) = \frac{5}{14}0.9709 + \frac{4}{14}0 + \frac{5}{14}0.9709 = 0.6935$. 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 chose the attribute with the highest *Gain* value. Attributes 'temp' and 'humidity' are continuous (numeric) therefore the discretization step of C4.5 method (see (8)) have to be applied. 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 on figure 2 is obtained.

In the following sections the proposed new Continuous Decision and Continuous Decision Tree methods are described in order to learn coefficients of a linear discriminant function and – in case of a more complex problem – to learn a decision tree with all inner nodes a linear discriminant function, respectively.

6 Learning linear discriminant models, the CD method

In this section learning of coefficients of a linear discriminant function (see 3 is examined and a new, Continuous Decision (CD) method is proposed.

The CD method is based on a global optimization method that looks for the hyperplane that separates elements of the training set falling into two different classes (denoted by positive ('+') and negative ('-') in the following). In order to be able to apply continuous optimization methods, soft-counting of elements is chosen.

The hyperplane, which represents the linear discriminant function, is described with bounded parameters in the following way. All training examples, given by n continuous attributes, might be represented as points in \mathbb{R}^n (see figure 7). Denote R_{max} 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 \overrightarrow{OP} and a point P lying on the hyperplane, that 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 $p_i \in (O_i - R_{max}, O_i + R_{max})$.

Instead of simple addition of elements on one side of the separating hyperplane (strict counting) the well-known sigmoid function is applied for soft-counting:

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

where $f(\mathbf{x}, \mathbf{a}) = 0$ describes the considered hyperplane, vector \mathbf{a} contains the parameters of the hyperplane and λ gives the sharpness at the bounds of the hyperplane. This soft-counting leads to that examples close to the separating

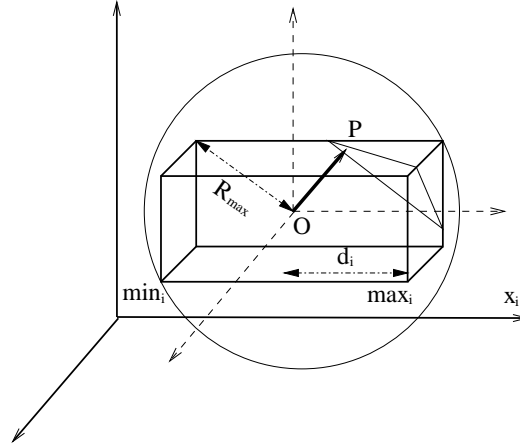


Figure 7: Description of a hyperplane, that separates the training elements, with bounded parameters.

hyperplane are counted into both sides (with different membership values). The applied sigmoid function is well known in artificial neural networks and it is chosen, because it has nice – easy to interpret and easy to calculate with – properties.

The portion of examples belong to class $C \in \{'+', '-'\}$ and falling in one side of the considered hyperplane described by 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 leads to a continuous global optimization problem, where extreme points of $S_1^C(\mathbf{a})$ are searched, where all variables $\mathbf{a} \in \mathbb{R}^m$ are continuous, with bounds. If elements of the training set is linearly separable, this methods gives parameters of a separating hyperplane, otherwise gives a hyperplane that separates the positive and negative training examples approximately. The function (14) to optimize is m dimensional where m is the number of attributes considered. All the variables are bounded with values coming from the description of hyperplanes. It is a continuous function with many small hills – depending upon the distribution of training examples belonging to different classes.

With the introduced CD method elements of the training set is separated with a hyperplane. But 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 set in (13) appropriately. In this way either quadratic separability or separation with spheres or ellipsoids could be solved.

Many different method is applicable to solve the described global optimization problem. In the implementation of the algorithm a stochastic method was

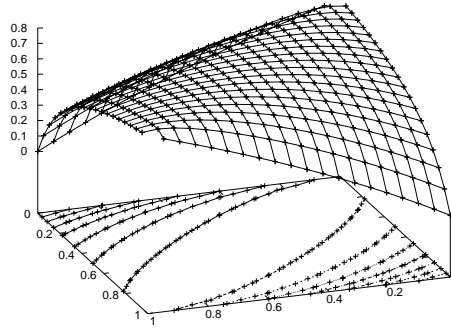


Figure 8: The test selection criterion. The CDT method walks on this surface during construction of the tree while the measure of separation $I^*(x)$ is reduced according to the *Gain* criterion. In the axes x and y the portion of positive and negative training examples on one side of the geometric figure are set.

chosen[12] but further investigation is needed in order to find the most appropriate method to our problem.

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

7 Learning Continuous Decision Trees, the CDT method

The CD method introduced in the previous section finds 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 all inner nodes of the tree contains a separation.

The CDT method is based on the greedy decision tree building algorithm in figure 4 and the *Gain*-type test selection criterion (5)-(7) derived from the measure $I^*(x)$ introduced in section 5. The result of our calculations for $E^*(t, S)$ is shown in figure 8.

The CDT method is applicable only when elements of the training set belongs to two classes, positive and negative ones.

Applying the CDT method, elements of \mathfrak{R}^n can be separated with arbitrary geometric figures. In this algorithm a soft-counting of elements is used and parameters of the figures separating the points are found by global optimization methods. In the following section it is demonstrated on a few examples.

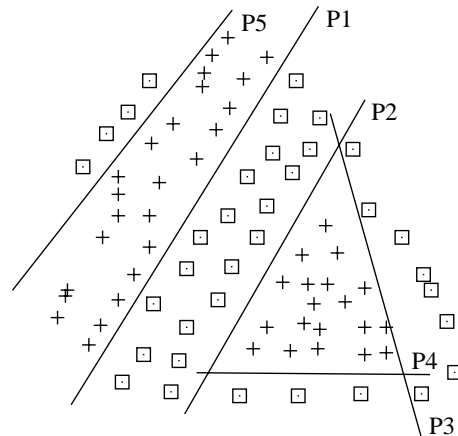


Figure 9: Separation of positive and negative elements with CDT in case of the extended triangle problem.

8 Examples

Some simple examples are considered to demonstrate the applicability of our CDT method introduced in the previous sections.

In the first example, take the training set shown in figure 9. This is the triangle problem extended with a few extra points. In the example only lines are allowed to separate elements of the training set. The separation of the triangle is solved by three lines correctly, while in the C4.5 method (where only axis-parallel lines are possible) about ten decisions are needed. (This example is also examined in [4]).

Another well-known difficult task is the problem of two spirals[4] on figure 10. It is solved by separation of points not only with lines but circles. In the proposed CDT method any type of geometric figures can be inserted easily as described in section 7. On the right side of the figure the first few steps of the tree building method is shown because it leads to a large complex decision tree. Elements of the training set have been quite well separated and the structure of the problem has been realized in these very first steps of the method. The C4.5 method gives a very complex decision tree to solve this task.

In the Iris plant problem (a 150 element dataset, 4 continuous attributes and 3 equally distributed classes¹) any class can be separated from the others only with one separation. The CDT method produces a compact tree and it also gives an impression of the layout of points representing the elements of

¹This database is available in the machine learning repository at <http://www.ics.uci.edu/~mllearn/MLRepository.html>

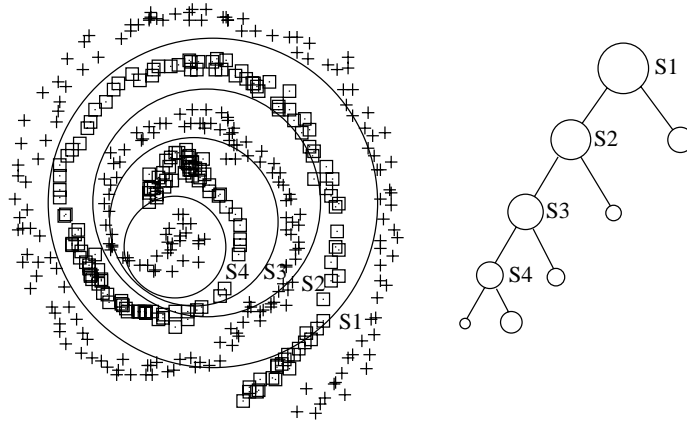


Figure 10: Separation of positive and negative elements with the CDT method by circles in case of the two spirals problem

the training set in \mathbb{R}^d – whether they are linearly separable by a hyperplane or separable by a sphere or not.

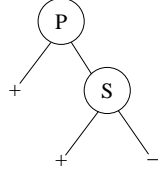
In a training set where most of the points are classified into class '+' and a few points into the other class ('-') in a small group, the '-' elements are separated by a single sphere, while in C4.5 at least $2m$ separating hyperplanes are required.

These simple examples shows 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. In the following section the transformation of trees into rules are introduced and the applicability of the method in the classification problematic of MCDA is investigated.

9 Ruleset generation, application in MCDA

It is possible to transform the results of the construction step, the decision tree, into rules in the following way: from the root node to a leaf get conjunctions of *tests* given in the inner nodes and make disjunction of expressions assigned to all paths from the root to a leaf (see figure 11). The generated ruleset is more readable, easier to understand and it might be applied to make inferences. In the rough sets theory the results are also transformed into rules[9].

In section 2 the classification problematic of MCDA was introduced. The models based on utility functions or the linear discriminant models are very similar to the single hyperplane learning CD method introduced in section 6. It can be applied to learn coefficients of the linear discriminant functions. However



$$\text{Ruleset}(+) = \{P(\mathbf{w}, w_0), \neg P(\mathbf{w}, w_0) \wedge S(\mathbf{o}, R)\}$$

The second expression in a more detailed form:

$$\neg P(\mathbf{w}, w_0) \wedge S(\mathbf{o}, R) = (\sum_{i=1}^m w_i x_i > w_0) \wedge (\sum_{i=1}^m (x_i - o_i)^2 < R^2)$$

Figure 11: Ruleset generation from a decision tree. The ruleset $\text{Ruleset}(+)$ is disjunction of two root to leaf paths. The second expression consist of conjunction of two separations – the first one is a separation by hyperplane P and the second one is a separation by a sphere S .

the method is generalized and any type of discriminant functions are possible to learn – quadratic functions, spheres, ellipsoids.

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 introduced in section 7 which leads to a more accurate and more complex model. In this way the classification of alternatives can be approximated arbitrary with a sequence of interpretable separations.

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

These type of decisions are a bit more complex than the ones usually applied, but the CDT model leads to a more compact description of the classification problematic of MCDA.

10 Conclusions, further plans

Our CDT decision tree building method is generalization of the C4.5 algorithm in the following sense.

In C4.5 the continuous decision space is separated only with 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 on some examples. Of course it increases the time-complexity of the algorithm, but 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 bound of the figure belong to both regions (with different membership values). Therefore, the decision criterion is a continuous function in the parameters of the figure. 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 problematic of MCDA. Either importances of criteria in a utility function model or coefficients in a discriminant function can be learnt from preclassified alternatives by the decision makers. Even more complex classification models could be determined that bases on decision trees or their transformation to rules.

A few more possible improvements follows. Sometimes the built up decision trees are too complex, worths to apply some pre- or post-pruning[6, 13] method to simplify the result tree. In the implementation of the algorithm seems to worth replacing the general purpose stochastic global optimizer with a problem specific one. The running time on large datasets might be rather long, so in this case the sampling technique should be used to build up a tree. The experiments shows that the algorithm is quite fast on a few hundred or thousands examples, but gets to be rather slow on larger (more hundred thousand elements) dataset.

Acknowledgements 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 much to improve both the level and the structure of the paper.

References

- [1] M. Beuthe and G. Scannella. Comparative analysis of uta multicriteria methods. *European Journal of Operational Research*, 130:246–262, 2001.
- [2] D. Bouyssou et al. *Evaluation and Decision Models: a critical perspective*. Kluwer Academic Publishers, Boston, 2000.
- [3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.
- [4] Krzysztof J. Cios and Ning Liu. A machine learning method for generation of a neural network architecture: a continuous ID3 algorithm. *IEEE Transactions on Neural Networks*, 3(2):280–291, March 1992.

- [5] J. Dombi. A general class of fuzzy operators, the demorgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets & Systems*, 8:149–163, 1982.
- [6] F. Esposito, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:476–491, 1997.
- [7] S. French. *Decision Theory: An Introduction to the Mathematics of Rationality*. Ellis Horwood, Chichester, 1988.
- [8] T. Gal, T.J. Steward, and T. Hanne, editors. *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications*. Kluwer Academic Publishers, 1999.
- [9] S. Greco, B. Matarazzo, and R. Slowinski. Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research*, 129:1–47, 2001.
- [10] G. Hardy, J. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 2nd edition, 1934.
- [11] L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5:15–17, 1976.
- [12] A.H.G. Rinnoy Kan and G.T. Timmer. Stochastic global optimization methods. *Mathematical Programming*, 39:27–78, 1987.
- [13] J. Kay. Comments on esposito et al. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:492–493, 1997.
- [14] M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, Cambridge, Massachusetts, 1994.
- [15] R.L. Keeney and H. Raiffa. *Decisions with multiple objectives – Preferences and value tradeoffs*. John Wiley and Sons, New York, 1976.
- [16] A.D. Luca and S.A. Termini. Definition of non-probabilistic entropy in the setting of fuzzy sets theory. *Information and Control*, 20:301–312, 1972.
- [17] Jean-Luc Marichal. *Aggregation Operators for Multicriteria Decision Aid*. PhD thesis, University of Liège, 1999.
- [18] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [19] V. Mousseau and R. Slowinski. Inferring an electre tri model from assignment examples. *Journal of Global Optimization*, 12:157–174, 1998.
- [20] S.K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

- [21] D.L. Olson. *Decision Aid for Selection Problems*. Springer, 1996.
- [22] Z. Pawlak and R. Słowiński. Decision analysis using rough sets. *Int. Trans. Opl Res.*, 1:107–114, 1994.
- [23] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
- [24] J. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.
- [25] B. Roy. The outranking approach and the foundations of electre methods. *Theory and Decision*, 1991.
- [26] B. Roy. Decision science or decision aid science? *European Journal of Operational Research*, 66:184–203, 1993.
- [27] S. Russel and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice-Hall, Englewood Cliffs, 1995.
- [28] T.L. Saaty. *The analytic hierarchy process*. McGraw Hill, 1980.
- [29] R. Vetschera. Entropy and the value of information. *CEJOR*, 8:195–208, 2000.
- [30] C. Zopounidis and M. Doumpos. Multicriteria classification and sorting methods: A literature review. *European Journal of Operational Research*, 138:229–246, 2002.