Induction of Ordinal Decision Trees:

An MCDA Approach

Jan C. Bioch Viara Popova

ERIM REPORT SERIES *RESEARCH IN MANAGEMENT*	
ERIM Report Series reference number	ERS-2003-008-LIS
Publication	January 2003
Number of pages	19
Email address corresponding author	popova@few.eur.nl
Address	Erasmus Research Institute of Management (ERIM) Rotterdam School of Management / Faculteit Bedrijfskunde Rotterdam School of Economics / Faculteit Economische Wetenschappen Erasmus Universiteit Rotterdam P.O. Box 1738 3000 DR Rotterdam, The Netherlands Phone: +31 10 408 1182 Fax: +31 10 408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

ERASMUS RESEARCH INSTITUTE OF MANAGEMENT

REPORT SERIES

RESEARCH IN MANAGEMENT

BIBLIOGRAPHIC DATA AND CLASSIFICATIONS		
Abstract	This paper focuses on the problem of monotone decision trees from the point of view of the multicriteria decision aid methodology (MCDA). By taking into account the preferences of the decision maker, an attempt is made to bring closer similar research within machine learning and MCDA. The paper addresses the question how to label the leaves of a tree in a way that guarantees the monotonicity of the resulting tree. Two approaches are proposed for that purpose - dynamic and static labeling which are also compared experimentally. The paper further considers the problem of splitting criteria in the context of monotone decision trees. Two criteria from the literature are compared experimentally - the entropy criterion and the number of conflicts criterion - in an attempt to nd out which one ts better the specifics of the monotone problems and which one better handles monotonicity noise.	
Library of Congress Classification (LCC)	5001-6182	Business
	5201-5982	Business Science
	HD 30.23	Decision making
Journal of Economic Literature (JEL)	M	Business Administration and Business Economics
	M 11 R 4	Production Management Transportation Systems
	C 49	Econometric and Statistical methods: Other
European Business Schools Library Group (EBSLG)	85 A	Business General
	260 K 240 B	Logistics Information Systems Management
	255 A	Decision theory
Gemeenschappelijke Onderwerpsontsluiting (GOO)		
Classification GOO	85.00	Bedrijfskunde, Organisatiekunde: algemeen
	85.34 85.20	Logistiek management Bestuurlijke informatie, informatieverzorging
	31.80	Toepassingen van de wiskunde
Keywords GOO	Bedrijfskunde / Bedrijfseconomie	
	Bedrijfsprocessen, logistiek, management informatiesystemen	
	ordinale gegevens, multicriteria analyse, beslissingen	
Free keywords	ordinal classication, multicriteria decision aid, multicriteria sorting, monotone decision trees, noise, pruning, labeling, splitting criteria, entropy	

Induction of Ordinal Decision Trees: An MCDA Approach

Jan C. Bioch Viara Popova

Dept. of Computer Science
Erasmus University Rotterdam,
P.O. Box 1738, 3000 DR Rotterdam.
{bioch,popova}@few.eur.nl

Abstract

This paper focuses on the problem of monotone decision trees from the point of view of the multicriteria decision aid methodology (MCDA). By taking into account the preferences of the decision maker, an attempt is made to bring closer similar research within machine learning and MCDA.

The paper addresses the question how to label the leaves of a tree in a way that guarantees the monotonicity of the resulting tree. Two approaches are proposed for that purpose - dynamic and static labeling which are also compared experimentally.

The paper further considers the problem of splitting criteria in the context of monotone decision trees. Two criteria from the literature are compared experimentally - the entropy criterion and the number of conflicts criterion - in an attempt to find out which one fits better the specifics of the monotone problems and which one better handles monotonicity noise.

Keywords: ordinal classification, multicriteria decision aid, multicriteria sorting, monotone decision trees, noise, pruning, labeling, splitting criteria, entropy

1 Introduction

Ordinal classification considers problems that are monotone i.e. all attributes have ordered domains (the attributes are criteria) and if x, y are data points such that $x \leq y$ ($x_i \leq y_i$ for each attribute i) then their labels should satisfy $\lambda(x) \leq \lambda(y)$). Ordinal classification therefore aims at generating classifiers that satisfy the same constraint. In practice monotone problems appear in different domain area. An example can be given with credit rating where if one applicant for credit outperforms another on all criteria then he should be given at least the same chance for being approved. Other examples can be given from the areas of financial management (e.g. bankruptcy prediction, bond rating), marketing, human resources management, etc.

The general methods for generating classifiers cannot guarantee that the monotonicity constraint will be satisfied, therefore different methods are necessary. For example well-known algorithm such as CART and C4.5 are not guaranteed to construct monotone trees. Ordinal classification has been studied in the context of logical analysis of data, decision trees, decision lists, rough sets theory, etc. by a number of authors, e.g. [8], [1], [10], [5], [9], [2], [3], [11], [7], [4].

Multicriteria sorting, an approach within the family of methods of Multicriteria Decision Aid (MCDA), considers the same problem as ordinal classification but from a different point of view. In general classification focuses on the model and the ways to extract automatically a model from the available data. The focal point in MCDA is the decision maker (DM) who is responsible for choosing the best alternative out of a set of solutions to the problem. That shifts the requirements for the methods in order to fit the decision making process. The method should be able to produce a model that is sensitive and flexible enough to incorporate explicitly the knowledge and preferences of the DM. The model should also be understandable to the DM and provide some explanation of why the recommended solution is considered the best. That might give the DM the possibility to trace back and further refine the model. For an overview of the MCDA methodology see [14].

Multicriteria sorting is one of the directions in MCDA which considers assigning the alternatives in predefined groups ordered by preference - starting from the most preferred alternatives to the least preferred ones. In this way it tries to solve the same problem as ordinal classification while taking into account the additional requirements considered by the MCDA approach. This similarity between the two area is known but little is done to bridge the gap and bring the methodologies closer. From the ordinal classification point of view one way to achieve that is to try to satisfy the special requirements of the DM for more transparent and flexible models that can implement their knowledge and experience.

One of the classification methods that are usually quoted to generate easier to understand models is the decision tree generation. This paper discusses some aspects of the monotone decision trees (MDT) generation which solves the similar problem within the ordinal classification area. While it inherits the transparency of the classical decision tree algorithms, a lot can still be done to increase the flexibility of the methods and the sensitivity to the preferences of the DM.

This paper focuses on the problem of providing possibility for increasing the flexibility of the MDT generation process by allowing the DM to choose from a number of alternative methods for controlling the outcome model. We consider different methods for controlling the size of the tree and a number of approaches for labeling the leaves. We also discuss the problem of choosing a splitting criterion. Experiment were done to compare the different methods with the goal of determining how they perform and which one fits better the problem of extracting MDT from noisy data.

Section 2 gives a brief introduction to the MDT algorithm. Section 3 presents the extension for dealing with monotonicity noise. Section 4 discusses the problems of pruning and labeling MDT. Two approaches for labeling are presented in subsections 4.1 and 4.2 and compared in 4.3. Section 5 discusses two splitting criteria - the entropy criterion and a more 'consistency-oriented' criterion we refer to as 'number of conflicts'. Section 6 gives the results of the experiments performed on the comparison of the two labeling approaches(subsection 6.1) and on the comparison of the two labeling approaches (subsection 6.2). Section 7 gives some conclusions and some possible directions for future research.

2 Monotone decision trees

An extension of the classical decision tree algorithm for dealing with ordinal data was first proposed in [10] for 2-class problems. A more general approach applicable to k-class problems is proposed in [5, 11]. In [4] a number of methods were presented for dealing with noise with respect to monotonicity.

The classical decision tree algorithm can be characterized by three rules: a splitting rule which defines how to split a node, a stopping rule defining when to stop growing a branch and turn the current node to a leaf and a labeling rule defining how to label the new leaf. The algorithm for MDT generation presented in [5, 11] extends the original algorithm by means of adding one more rule, the update rule, which takes care of preserving the monotonicity property of the tree.

Let T be a node of the tree $\mathcal{T}$ generated thus far on the data set $D \subseteq \mathcal{X}$ where $\mathcal{X}$ is the input space. T can be represented as $T = \{x \in \mathcal{X} : a(T) \leq x \leq b(T)\}$ where $a(T), b(T)$ are called the lower/upper corners of T respectively. In the following the corners will be denoted by simply a and b when no ambiguity occurs. The original MDT algorithm is given in figure 1. In order to guarantee the monotonicity of the tree, an update procedure is performed on the data set by adding at most 2 new data points with consistent labels. The update procedure is performed every time a node is considered for splitting and the added points are the lower and the upper corners of the node (if they are not already present in the data set). The labels are chosen to be $\lambda_{max}(a)$ and $\lambda_{min}(b)$ respectively, which are defined in the following, where c_{min}/c_{max} are the lowest/highest class in the data set:

$$\downarrow x = \{y \in \mathcal{X} : y \leq x\}$$

$$\uparrow x = \{y \in \mathcal{X} : y \geq x\}$$

$$\downarrow D = \bigcup_{x \in D} \downarrow x$$

$$\uparrow D = \bigcup_{x \in D} \uparrow x$$

$$\lambda_{min}(x) = \begin{cases} \max\{\lambda(y) : \ y \in D \cap \downarrow x\} & \text{if } x \in \uparrow D \\ c_{min} & \text{otherwise.} \end{cases}$$

x	a_1	a_2	a_3	a_4	a_5	a_6	λ
1	0	0	1	1	0	1	0
2	1	1	2	1	3	1	0
3	0	1	1	0	0	1	0
4	2	3	1	3	3	1	1
5	1	0	2	2	3	1	1
6	0	0	0	3	2	2	1
7	2	2	1	1	1	2	1
8	2	4	2	2	2	3	2
9	1	1	2	1	3	2	2
10	3	2	1	0	0	1	2
11	3	2	2	1	2	2	3
12	3	3	4	1	2	2	3
13	4	2	3	3	3	3	3
14	3	3	3	4	1	3	3
15	4	4	2	3	0	1	3

Table 1: The example data set

$$\lambda_{max}(x) = \begin{cases} \min\{\lambda(y) : \; y \in D \cap \uparrow x\} & \text{if } x \in \downarrow D \\ c_{max} & \text{otherwise.} \end{cases}$$

Using this way of labeling we guarantee that the lower corner gets the minimal label possible for the node and the upper corner gets the maximal possible label. At the same time the new points remain consistent with the rest of the data, therefore the monotonicity constraint is not violated.

```
split(node $T$):
    **update($T$)**;
    if $T$ is homogeneous
        label $T$;
    else
        split $T$ into disjoint $T_L$ and $T_R$;
        split($T_L$);
        split($T_R$);
```

```
**update(node $T$)**:
    if $a \notin D$
        $\lambda(a) = \lambda_{max}(a)$;
        add $a$ to $D$;
    if $b \notin D$
        $\lambda(b) = \lambda_{min}(b)$;
        add $b$ to $D$;
```

Figure 1: The monotone decision tree algorithm

In order to illustrate how the algorithm works we use the example data set from table 1. In the first step, the root of the tree (containing the whole data set) is considered for splitting. The update rule fires and the corners $a = (0,0,0,0,0,0)$ and $b = (4,4,4,4,3,3)$ are added with labels $\lambda(a) = 0$ and $\lambda(b) = 3$. Further $a_1 > 2$ is chosen for a test of the node and the data

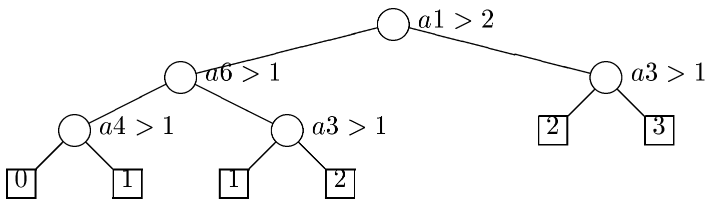

Figure 2: The MDT generated for the example data set

set is divided between the two children. Following the left-depth-first strategy we consider the left child for splitting. The corners are $a = (0,0,0,0,0,0)$ and $b = (2,4,4,4,3,3)$ where a is already present and b is added with a label $\lambda(b) = 2$. The algorithm continues with finding the best split, etc. The full monotone tree generated from this data set is given in figure 2.

3 Monotone decision trees from noisy data

The MDT algorithm requires a monotone data set. In practical applications, however, noise can cause inconsistencies of the type: data points $x \leq y$ with labels $\lambda(x) > \lambda(y)$. The extension of the algorithm proposed in [4] allows the generation of monotone trees from non-monotone data.

The extension of the MDT algorithm for handling noisy data alters the update rule to not only add the corners if they are not present but also to relabel them with consistent labels if they are present. In this way the algorithm tries to repair the inconsistencies caused by monotonicity noise. The new update rule, for a node T with lower/upper corners respectively a and b, is given in figure 3.

```
update(node $T$):
    $l_1 = \lambda_{max}(a)$; $l_2 = \lambda_{min}(b)$;
    if $a \in D$
      relabel $a$: $\lambda(a) = l_1$;
    else
      label $a$: $\lambda(a) = l_1$; add $a$ to $D$;
    if $b \in D$
      relabel $b$: $\lambda(b) = l_2$;
    else
      label $b$: $\lambda(b) = l_2$; add $b$ to $D$;
```

Figure 3: The extended update rule

Theorem: *The MDT algorithm with the extended update rule of figure 3 always generates a monotone tree.*

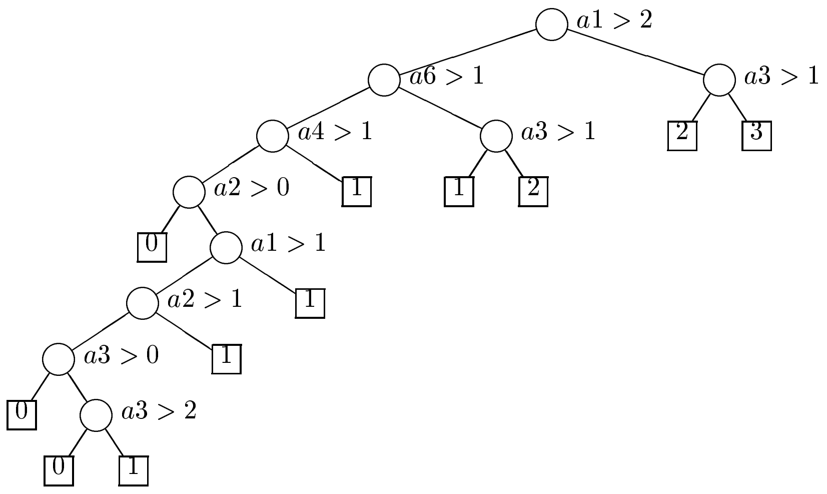

Figure 4: MDT on the non-monotone data set

(See [4] for a proof.)

To illustrate the algorithm we introduce monotone inconsistency in the example data set of table 1 - we change the label of data point x_3 from 0 to 1. Thus we introduce an inconsistent pair of data points (x_2,x_3). The output of the algorithm on the new data set is given in figure 4.

Note that a simple criterion for checking the monotonicity of a tree [11] can be defined as follows.

Let $\mathcal{L}$ be the set of leaves of a tree $\mathcal{T}$ and $\mathcal{N}$ be the set of nodes of $\mathcal{T}$. We define a relation on $\mathcal{N}$: for $T, T' \in \mathcal{N}$

$$T \leq T' \Leftrightarrow a(T) \leq b(T').$$

Let $T, T' \in \mathcal{L}$ such that

$$T = \{x \in \mathcal{X} : a(T) \leq x \leq b(T)\}$$

and

$$T' = \{x \in \mathcal{X} : a(T') \leq x \leq b(T')\}.$$

Then the tree is monotone if for any choice of T and T':

$$T \leq T' \Rightarrow \lambda(T) \leq \lambda(T').$$

4 Pruning and labeling rules that guarantee the monotonicity

The decision maker often prefers to work with rules which are easier to understand, apply and explain the decision. The decision tree can easily be translated

```
label leaf $T$:
    $\lambda(T) = L(T)$;
    $\lambda(a(T)) = \lambda(T)$;
    $\lambda(b(T)) = \lambda(T)$;
```

Figure 5: The dynamic labeling rule

to rules, however when the tree is big, the rules are long too and that obstructs the intuitiveness and understandability that appealed to the DM in the first place. Therefore the DM should also be given tools for controlling the size of the tree. Pruning is the first choice in this direction.

Pruning can be performed in two ways: pre-pruning which prematurely stops the growth of the tree when a predefined threshold is reached and post-pruning which first grows the full tree and then cuts branches from it until a predefined criterion is fulfilled.

In both situations it is important to apply consistent rules for giving labels to the new leaves which in the case of MDT should guarantee that the resulting tree is monotone. Dynamic labeling refers to giving labels while the tree is being generated, as soon as a node is turned to a leaf. It can be used together with pre-pruning. Static labeling refers to the process of giving consistent labels to the tree that is already generated. It can be used on any trees with non-homogeneous leaves, e.g. trees generated with pre- and post-pruning as well as trees generated with other methods.

This section discusses the two labeling methods.

4.1 Dynamic labeling

One important difference between the static and the dynamic way of labeling is how much of the information in the tree is already available. While in the static case the tree is grown and all the information about the shape of the tree, the corners of the leaves and the labels of these corners is available, in the dynamic case we only have a part of the tree built and the new label should be based on partial information. Therefore, for dynamic labeling, an important factor is the search strategy used for building the tree, i.e. which part of the tree is expected to be built already and what kind of information will influence the new labels. In the following we assume the depth-first strategy for growing the tree. First we note an observation that holds for this strategy.

Lemma: *Let $T, T' \in \mathcal{L}(\mathcal{T})$ in the monotone tree $\mathcal{T}$ generated with depth-first strategy. Let $T \leq T'$. Then leaf T is generated before leaf T'.*

(See [4] for a proof.)

The general form of the dynamic labeling rule for a non-homogeneous leaf T is given in figure 5 where L is the labeling function. Two possible forms are proposed for the labeling function as follows:

$$L \in \{L_{min}, L_{max}\}$$

$$L_{min}(T) = max\{\lambda(a(T'))|T' \leq T\}$$
$$L_{max}(T) = min\{\lambda(b(T'))|T \leq T'\}$$

Note that for $T' \not\equiv T$ in the above formulas, $a(T') = b(T') = \lambda(T')$.

Theorem: *Let $\mathcal{T}$ be a tree generated using the extended MDT update rule for a threshold of at least m points in a leaf, $m \geq 1$. Let the leaves be labeled using the dynamic labeling rule of figure 5 where one of the following strategies is applied:*

1. $L(T) = L_{min}(T), \forall T \in \mathcal{L}$,
2. $L(T) = L_{max}(T), \forall T \in \mathcal{L}$.

Then $\mathcal{T}$ is monotone.

Proof: Let the tree be generated using the left-depth-first strategy. The newly generated leaf to be labeled is denoted by T. If T is the first leaf then the current set of labeled leaves is monotone.

Let T be not the first leaf and let the current set of labeled leaves be monotone. We label $\lambda(T) = L(T)$.

Let $L = L_{min}(T)$. Then

$$\lambda(T) = max\{\lambda(a(T'))|T' \leq T\}.$$

Therefore, for each $T' \leq T$, $\lambda(T') \leq \lambda(T)$. Therefore the state of the tree is still monotone. The proof is analogous for $L = L_{max}$.

In the same way it can be proved for right-depth-first strategy. $\square$

The following observations can help speeding up the computation:

Observation 1: *For the left-depth-first strategy the following holds:* $L_{max}(T) = \lambda(b(T))$.

Observation 2: *For the right-depth-first strategy the following holds:* $L_{min}(T) = \lambda(a(T))$.

The experiments suggest that $L_{max}(a)$ tends to favor the lower classes while $L_{min}(b)$ tends to favor the higher classes. Therefore they provide a choice to the decision maker for a more pessimistic against a more optimistic prediction.

4.2 Static labeling

As mentioned before, static labeling is performed on the fully generated tree, where the information about all the leaves is already available.

The earlier defined relation over the nodes/leaves of a tree is not transitive. We define now a transitive closure relation, denoted by $\trianglelefteq$, in the following way. For $T', T'' \in \mathcal{N}$: $T' \trianglelefteq T'' \Leftrightarrow \exists T_1, T_2, ..., T_m \in \mathcal{N}$ such that $T' \leq T_1 \leq T_2 \leq ... \leq T_m \leq T''$.

In our implementations we used the algorithm of Warshall [13] for computing the transitive closure.

We define Λ_{min} and Λ_{max} over the set of leaves $\mathcal{L}$ as follows:

$$\Lambda_{min}(T) = max\{\lambda(a(T_1))|T_1 \in \mathcal{L}, T_1 \trianglelefteq T\}$$

$$\Lambda_{max}(T) = min\{\lambda(b(T_2)) | T_2 \in \mathcal{L}, T \trianglelefteq T_2\}$$

Then Λ is defined as: $\Lambda \in \{\Lambda_{min}, \Lambda_{max}\}$.

It can be shown that $\Lambda_{min} \leq \Lambda_{max}$ and they are monotone labelings, i.e. for leaves $T_1 \leq T_2$, $\Lambda_{min}(T_1) \leq \Lambda_{min}(T_2)$ and $\Lambda_{max}(T_1) \leq \Lambda_{max}(T_2)$.

Theorem: *Let $\mathcal{T}$ be a tree generated without labeling. Let the leaves be visited following either left-depth-first or right-depth-first order. The leaves are labeled using one of the following strategies:*

1. $\lambda(T) = \Lambda_{min}(T), \forall T \in \mathcal{L}$,
2. $\lambda(T) = \Lambda_{max}(T), \forall T \in \mathcal{L}$.

Then the resulting tree is monotone.

Proof: Let $\Lambda = \Lambda_{min}$ and the order of visiting the leaves is left-depth-first. Let the last leaf labeled is T. If T is the first labeled leaf then the current set of labeled leaves is monotone.

Let T be not the first labeled leaf and let the current set of labeled leaves be monotone.

$\lambda(T) = \Lambda_{min}(T) = max\{\lambda(a(T')) | T' \trianglelefteq T\}$

Let us assume that the resulting set of labeled leaves is no longer monotone. Using left-depth-first means that there are no labeled leaves T' such that $T \trianglelefteq T'$. Therefore $\exists T' \trianglelefteq T, \lambda(T') > \lambda(T)$. Since $\lambda(T) \geq \lambda(a(T'))$, then $\lambda(T') > \lambda(a(T'))$

$$\Rightarrow \exists T'' : T'' \trianglelefteq T', \lambda(T') = \lambda(a(T'))$$

$$\Rightarrow T'' \trianglelefteq T \Rightarrow \lambda(T) \geq \lambda(a(T'')) = \lambda(T')$$

which is a contradiction with the assumption.

The proof is analogous for Λ_{max} and for right-depth-first order. $\Box$

4.3 Comparison

In order to get more insight in the performance of the two presented approaches for labeling, experiments were conducted. The main goal was to compare the results from the four possible combinations of settings: dynamic labeling with L_{min}, dynamic labeling with L_{max}, static labeling with Λ_{min} and static labeling with Λ_{max}.

The trees were generated using left-depth-first search strategy with pre-pruning at predefined thresholds for the minimal number of points in a node. Therefore the size of the trees was almost always the same.

The results showed that:

- In general the dynamic labeling produces trees with lower misclassification rate on unseen data. The reason for that is most probably in the fact that the tree changes dynamically and it is possible to generate trees that have different shape while in the static case the shape of the tree is already fixed and the only feature that changes is the labels of the leaves.

- The difference between the two labeling functions for both approaches was not so clear-cut and no conclusion can be made yet on which one is preferable.

Details on the experimental setting and the results are given in section 6.

5 Splitting criteria for MDTs

One of the important rules in building a decision tree is the splitting rule which defines how to split the current node in two branches. For a binary tree a split is a tuple of the type $\langle a_i, v_j \rangle$ where a_i is the attribute to split on and v_j is the cut-off value. There is a lot of research done on finding good criteria for choosing good splits for the classical DT algorithms. One of the most successful approaches is to choose the split which produces the highest decrease in the entropy or the highest information gain. These two notions are usually defined as follows. The entropy of a node T is:

$$Ent(T) = -\sum_{i=1}^{n} p_i log_2 p_i$$

where p_i is the proportion of data points with class i in T for an n-class problem. Then the information gain of an attribute a_i and cut-off value v_j is:

$$Gain(T, a_i, v_j) = Ent(T) - \sum_{k \in \{L,R\}} \frac{|T_k|}{|T|} Ent(T_k)$$

where T_L/T_R are respectively the left and the right child of T when split on $\langle a_i, v_j \rangle$.

However for the specific case of MDT it is not clear which splitting criteria are good. Intuitively they should not only attempt to produce smaller trees but also provide fast decrease in the inconsistencies in the tree. One such criterion within the MCDA methodology was suggested in [7] although no experimental results were presented on its performance compared to other criteria. The criterion aims at reducing the number of non-monotone pairs of points in the resulting branches. It chooses the split with the least number of inconsistencies/conflicts.

Let the current node T be split into the following non-overlapping subsets:

$$T' = \{a(T') \le x \le b(T')\}$$

and

$$T'' = \{a(T'') \le x \le b(T'')\}.$$

Let T' be the left branch. Therefore $T' \le T''$. If for all points $x \in T', y \in T''$ it is true that $\lambda(x) \le \lambda(y)$ then the split is monotone. There might be however points such that $\lambda(x) > \lambda(y)$. The number of those inconsistent pairs is counted for all possible splits of the current node and the one with the lowest count is chosen.

The experiments that were performed for this paper aim at giving more insight in the performance of the two mentioned criteria in the context of MDT. The two main aspects for comparison are the size and the accuracy of the generated trees. The experimental results point at the following observations.

- On monotone data sets no criterion is systematically better than the other.
- With the increase of monotonicity noise in the data, the entropy criterion tends to generate smaller trees.
- With the increase of monotonicity noise in the data the conflicts criterion generates more accurate trees with lower misclassification rate on unseen data.

Intuitively the reason for the difference is probably in the orientation of the two criteria. The entropy criterion strives at generating smaller trees by reducing the diversity of classes in the leaves as fast as possible. The conflicts criterion, on the other hand, only cares about the consistency/monotonicity of the tree and therefore produces trees that are larger but better fit the 'character' of the data. In this way it handles more successfully monotonicity noise but at the cost of generating bigger trees.

Details about the data used, the performed experiments and the experimental results are given in section 6.

6 Experiments

As it was mentioned in sections 4 and 5, experiments were performed in two directions. The first direction aims at comparing the dynamic and the static approach for labeling with their variations. The results are presented in section 6.1. The other direction of experiments aims at comparing the entropy and the conflicts splitting criteria in the context of monotone decision trees. The experimental setting and the results are presented in section 6.2. In all experiments left-depth-first search strategy was used.

For all experiments, the starting data sets were taken from UCI Machine Learning Repository [6]. The Nursery data set is a real-world monotone data set which represents applications for a nursery school, contains 12960 instances described by 8 attributes and covers the whole input space. The Cars data set is an artificial set describing cars by their properties and classifying them according to their acceptability for a buyer. The original data was not strictly monotone and for that reason one of the values of one attribute was removed. The resulting set was monotone. It contains 1153 instances described by 6 attributes. Since both data sets cover the whole input space, they were also used as test sets in the experiments.

For the experiments random samples of size 200 points were drawn from both data sets. Monotone inconsistencies were introduces in the data in the following way: a pair of comparable data points (such that either $x \leq y$ or $x \geq y$) from different classes was chosen at random and the labels were switched. That results in one or more inconsistent pairs. The procedure can be repeated to introduce more noise.

Since both data sets cover the whole input space it was possible to test the misclassification rate on the whole data instead of using separate test samples.

6.1 Comparison of the dynamic and the static labeling

For the comparison of the two labeling approaches, four samples were used - one from the cars data set and three from the nursery data. For each the algorithms were applied for the thresholds of minimum 5, 10, 15 and 20 points in a node. For the static case the leaves were left unlabeled and the labeling was performed at the end. Eight different combinations of settings were tested: entropy versus conflicts splitting criteria, L_{min} versus L_{max} and Λ_{min} versus Λ_{max}. The results about the splitting criteria are discussed in section 6.2 and for the purpose of the current discussion the results will the averaged over the two criteria.

As a test set, the full data sets were used as they cover the whole input space (1153 points for the cars data set and 12960 points for the nursery data set). The number of misclassified points per sample is given in table 2. The first column contains the value of the threshold. Columns 2 and 3 give the results for the tree labeled with static labeling for Λ_{min} and Λ_{max} respectively Columns 4 and 5 give the analogous results for dynamic labeling and L_{min} and L_{max} respectively.

As it was mentioned before, the size of the trees was the same for the static and the dynamic case (per sample and threshold). Having this in mind, it can be seen that the number of points misclassified by the trees which was labeled dynamically is systematically lower than (often more than twice as low as) that of the trees labeled statically.

On the other hand no definite answer can be given to the question which labeling function to use: L_{min} versus L_{max} and Λ_{min} versus Λ_{max}. In the static case the choice of labeling function made hardly any difference in the results while in the dynamic case for some samples L_{min} is better and for the other L_{max} gives better results.

These results are also confirmed by the experiments performed by a master's student [12] working under the guidance of the authors of this paper.

6.2 Comparison of the entropy and the conflicts splitting criteria

The experiments were designed to answer two main questions in the comparison of the two splitting criteria:

- Which criterion performs better on monotone data by means of generating smaller and/or more accurate monotone trees?
- Which criterion handles better monotonicity noise/inconsistencies by means of generating smaller and/or more accurate monotone trees?

	static		dynamic	
thr	Λ_{min}	Λ_{max}	L_{min}	L_{max}
5	205	205	93	285
10	350	350	156	407
15	408	408	217	425
20	408	408	219	434
5	2495	2495	1708	1915
10	4012	4012	2172	2154
15	4530	3524	2395	2292
20	4950	4950	2512	2424
5	2436	2436	1821	1694
10	3380	3380	2486	1892
15	4653	4653	2587	2404
20	5055	5055	2967	2404
5	4152	4152	2097	3351
10	5402	5402	2489	3509
15	6892	6892	2983	4219
20	7398	7403	3130	4555

Table 2: Experimental data on the labeling approaches

In order to give some insight on these questions the experiments were conducted in two different settings. In the first part 10 monotone samples were drawn from the nursery data set. MDTs were generated from each of them using the two criteria. The results - the number of misclassified points over the full data sets and the number of tree nodes - are given in table 3. It can be seen that the performance of the two criteria is different on the different samples and no definite conclusion can be made on which one fits better monotone problems.

For the second part of the experiments, 4 samples were chosen - 1 from the cars data set and 3 from the nursery data set (rows 2, 3 and 7 from table 3). For each data set 6 noisy versions (5 for the cars data) were generated by switching the labels of 1 to 7 pairs of points. For each such series, MDTs were generated using the entropy and the conflicts criteria.

The results per series are presented in table 4. The first column contains the number of non-monotone pairs of points in the set. Columns 2 and 3 give the number of misclassified points on the full data set and the number of nodes for the tree generated with the entropy criterion, while columns 4 and 5 give the respective information for the number of conflicts criterion.

It can be seen that with the increase of inconsistencies the number of conflicts criterion gives systematically better misclassification rate while producing bigger trees than the entropy criterion.

entropy		num conflicts	
miscl	nodes	miscl	nodes
1612	321	1512	233
1516	95	1522	211
1520	87	1752	131
1645	85	1458	247
1478	143	1330	159
1354	151	1429	783
1277	163	1718	195
1266	107	1232	149
1087	401	1672	241
1504	343	1622	327

Table 3: Experimental data on the splitting criteria for monotone samples

incons	entropy		num conflicts	
pairs	miscl	nodes	miscl	nodes
0	52	53	47	83
19	90	293	74	171
55	196	311	103	373
86	218	421	172	501
111	225	469	176	521
118	224	473	186	609
0	1516	95	1522	211
2	1500	115	1520	303
21	2326	561	1789	397
28	2496	593	1773	583
31	2366	605	1737	589
40	2432	963	1763	639
42	2586	1787	1831	1069
0	1520	87	1752	131
2	1512	157	1812	125
7	1673	747	1850	317
39	3052	3655	2692	3497
64	3607	3719	3003	3829
66	3685	3353	3327	4339
69	3798	3751	3322	4437
0	1277	163	1718	195
39	3035	2093	3607	3945
59	3518	3755	4283	5645
74	4577	2969	4369	5737
78	4603	3137	4553	5863
81	4547	3125	4504	5887
83	4547	3135	4466	5861

Table 4: Experimental data on the splitting criteria on non-monotone samples

7 Conclusions and further research

This paper presents two approaches for labeling decision trees in a way that guarantees the monotonicity of the resulting trees. For each approach two possible labeling functions are proposed and their performance was compared experimentally. The results indicate that dynamic labeling approach produces more accurate trees than the static labeling for trees of the same size.

The experiments used the left-depth-first search strategy for generating the trees. The obvious next step in this direction of research is to investigate whether the results also apply for the right-depth-first strategy.

The paper further investigates the performance of two splitting criteria from the literature - the classical entropy criterion and the more 'monotonicity-oriented' number of conflicts criterion. Although for monotone samples the performance of none of the criteria is systematically better, the experimental results confirm the intuition that with the increase of monotonicity noise the entropy criterion produces smaller trees while the conflicts criterion generates more accurate trees.

These results point at one potential direction for further research - is it possible to combine the good properties of the two criteria in a new criterion that both strives at generating smaller trees and reducing the inconsistencies in the nodes in order to produce trees that fit better monotone data and better handle monotonicity noise.

References

[1] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19:29–43, 1995.

[2] J. C. Bioch. Dualization, decision lists and identification of monotone discrete functions. *Annals of Mathematics and Artificial Intelligence*, 24:69–91, 1998.

[3] J. C. Bioch and V. Popova. Rough sets and ordinal classification. *LNAI, Proceedings of the 11th Int. Conf. on Algorithmic Learning Theory (ALT'2000), Sydney, Springer-Verlag*, pages 291–305, 2000.

[4] J. C. Bioch and V. Popova. Monotone classification and noisy data. Technical Report ERS-2002-53-LIS, Dept. of Computer Science, Erasmus University Rotterdam, http://www.erim.nl, 2002.

[5] J. C. Bioch and R. Potharst. Decision trees for monotone classification. *K. van Marcke and W. Daelmans (eds), Proceedings of the Dutch Artificial Conference on Artificial Intelligence (NAIC'97), Antwerp*, pages 361–369, 1997.

[6] C. L. Blake and C. J. Mertz. Uci repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science [http://www.ics.uci.edu/ mlearn/ MLRepository.html], 1998.

[7] K. Cao-Van and B. De Baets. Growing decision trees in an ordinal setting. *submitted to International Journal of Intelligent Systems*, 2002.

[8] Y. Crama, P. L. Hammer, and T. Ibaraki. Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, 16:299–326, 1988.

[9] S. Greco, B. Matarazzo, and R. Slowinski. A new rough set approach to evaluation of bankruptcy risk. *C. Zopounidis (ed.), Operational Tools in the Management of Financial Risks, Kluwer, Dordrecht*, pages 121–136, 1998.

[10] K. Makino, T. Suda, K. Yano, and T. Ibaraki. Data analysis by positive decision trees. *Proceedings of International Symposium on Cooperative Database Systems for Advanced Applications (CODAS), Kyoto*, pages 282–289, 1996.

[11] R. Potharst and J. C. Bioch. Decision trees for ordinal classification. *Intelligent Data Analysis*, 4:1–15, 2000.

[12] W. van Eikeren. Monotone decision trees and stacked generalization. Master's thesis, Erasmus University Rorrerdam, 2002.

[13] S. Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962.

[14] C. Zopounidis and M. Doumpos. Multicriteria classification and sorting methods: A literature review. *European Journal of Operational Research*, 138:229–246, 2002.

Publications in the Report Series Research[*] in Management

ERIM Research Program: “Business Processes, Logistics and Information Systems”

2003

Project Selection Directed By Intellectual Capital Scorecards
Hennie Daniels and Bram de Jonge
ERS-2003-001-LIS

Combining expert knowledge and databases for risk management
Hennie Daniels and Han van Dissel
ERS-2003-002-LIS

Recursive Approximation of the High Dimensional max Function
Ş. İl. Birbil, S.-C. Fang, J.B.G. Frenk and S. Zhang
ERS-2003-003-LIS

Auctioning Bulk Mobile Messages
S.Meij, L-F.Pau, E.van Heck
ERS-2003-006-LIS

Induction of Ordinal Decision Trees: An MCDA Approach
Jan C. Bioch, Viara Popova
ERS-2003-008-LIS

A New Dantzig-Wolfe Reformulation And Branch-And-Price Algorithm For The Capacitated Lot Sizing Problem With Set Up Times
Zeger Degraeve, Raf Jans
ERS-2003-010-LIS

International Portfolio Choice: A Spanning Approach
Ben Tims, Ronald Mahieu
ERS-2003-011-LIS

Reverse Logistics – a review of case studies
Marisa P. de Brito, Rommert Dekker, Simme D.P. Flapper
ERS-2003-012-LIS

Product Return Handling: decision-making and quantitative support
Marisa P. de Brito, M. (René) B. M. de Koster
ERS-2003-013-LIS

[*] A complete overview of the ERIM Report Series Research in Management:
http://www.ers.erim.eur.nl

ERIM Research Programs:

LIS	Business Processes, Logistics and Information Systems
ORG	Organizing for Performance
MKT	Marketing
F&A	Finance and Accounting
STR	Strategy and Entrepreneurship