CMS 1: 231–244 (2004)

DOI: 10.1007/s10287-004-0014-9

Decision trees for monotone price models

Marina Velikova[1], **Hennie Daniels**[2,3]

[1] Center for Economic Research, Tilburg University, P.O. Box 90153, 5000 LE, Tilburg, The Netherlands (e-mail: M.Velikova@uvt.nl)
[2] Center for Economic Research, Tilburg University, The Netherlands
[3] ERIM Institute of Advanced Management Studies, Erasmus University Rotterdam, The Netherlands

Abstract. In economic decision problems such as credit loan approval or risk analysis, models are required to be monotone with respect to the decision variables involved. Also in hedonic price models it is natural to impose monotonicity constraints on the price rule or function. If a model is obtained by a "unbiased" search through the data, it mostly does not have this property even if the underlying database is monotone. In this paper, we present methods to enforce monotonicity of decision trees for price prediction. Measures for the degree of monotonicity of data are defined and an algorithm is constructed to make non-monotone data sets monotone. It is shown that monotone data truncated with noise can be restored almost to the original data by applying this algorithm. Furthermore, we demonstrate in a case study on house prices that monotone decision trees derived from cleaned data have significantly smaller prediction errors than trees generated using raw data.

Keywords: Data mining, monotone decision trees, price models

MSC code: 90-08 (Computational methods)

1 Introduction

Although, in data mining literature, the main emphasis is put on the analysis and interpretation phase, there are more aspects such as data selection and data preprocessing, which determine the successful implementation of any data mining system. The right description of the domain as well as data cleaning, data integration and data transformation can significantly improve the efficiency of the data mining process. Apart from limitations regarding data quality, there can also be problems in the application of the model if the mining process is conducted by blind search (Breeze, Goldman and Wellman, 1994). Frequently, the models derived are incompatible with business regulations. These shortcomings can often be

© Springer-Verlag 2004

resolved by integrating expert knowledge in the data mining process. Another problem that may occur is the lack of interpretability of the model. In general, human decision makers require models that are easy to understand and do not accept black box models, for example neural networks or very complex decision trees.

Therefore, there is a need for integration of the knowledge discovered by standard data mining algorithms with the knowledge based on intuition and experience of the domain experts (Garofalakis, Rastogi and Shim, 2002; Rajagopalan and Isken, 2001). In this paper, we explicitly describe the implementation of a special form of a prior knowledge that is typical in economic decision problems, namely the monotonicity of classification or prediction rules.

In recent years, several researchers became interested in the incorporation of monotonicity constraints in different data mining methods. In Daniels and Kamp (1999) and Wang (1994) classes of monotone neural networks are introduced. Also, in the application of decision trees, several methods have been developed to solve classification problems with monotonicity constraints. In Ben-David (1995), a new splitting measure for constructing a decision tree was proposed including a non-monotonicity index and standard impurity measure such as entropy. In this way, monotonicity properties of the tree and the classification error can be balanced. Potharst (1999) provides a study for building monotone decision trees from monotone data sets. This constraint limits the applicability of the methods because real databases are often non-monotone due to noise. Frequent occurring causes are errors at data entry, inconsistencies after merging data sets, discrepancies due to the change of data over time, etc.

In this paper, we consider cases where the response variable is continuous as opposed to the discrete case which is treated in Daniels and Velikova (2003). The remainder of this paper is organised as follows. Firstly, in Sect. 2 we formulate monotonicity constraints for regression and classification problems. Then, in Sect. 3 we construct measures to check if data are monotone by defining indicators for the degree of monotonicity. By comparing the value of the indicators with benchmark data sets, one can verify the monotonicity of the data under study.

The main contribution of this paper is the algorithm presented in Sect. 4. This algorithm transforms non-monotone data into monotone data by a relabeling process. This leads to improvements in transparency, smaller error rates and lower variance of the decision models. It is shown that noise added to artificially generated monotone data sets can be removed to a large extent by the algorithm. In a case study (Sect. 5) on house pricing, we show that monotone trees outperform ordinary decision trees and monotone trees generated from cleaned (monotone) data outperform trees generated from the original data, in the sense that the former are more stable upon repeated sampling than the latter and the out-of sample error (on the original data) is smaller.

2 Monotonicity and measures for monotonicity

In many economic classification and regression problems, it is known that the dependent variable has a distribution that is monotone with respect to the independent variables. Economic theory would state that people tend to buy less of a product if its price increases (ceteris paribus), so there would be a negative relationship between price and demand. The strength of this relationship and the precise functional form are, however, not always dictated by economic theory. Another well-known example is the dependence of labour wages as a function of age and education (Mukarjee and Stern, 1994). In loan acceptance, the decision rule should be monotone with respect to income for example, i.e., it would not be acceptable that a high-income applicant is rejected, whereas a low-income applicant with otherwise equal characteristics is accepted. Monotonicity is also imposed in so-called hedonic price models where the price of a consumer good depends on a bundle of characteristics for which a valuation exists (Harrison and Rubinfeld, 1978).

The mathematical formulation of monotonicity is straightforward. We assume that y is the dependent variable and takes values in $\boldsymbol{Y}$, the vector of independent variables is x and takes values in $\boldsymbol{X}$. In the applications discussed here, $\boldsymbol{Y}$ is a one-dimensional space of prices or classes and $\boldsymbol{X}$ is a k-dimensional space of characteristics of products or customers. Furthermore, a data set $D = (y^n, x^n)$ of n points in $\boldsymbol{Y}^*\boldsymbol{X}$ is defined, which can be considered as a random sample of the joint distribution of (y, x). In a regression problem, the goal is to estimate the average dependence of y given x, $\mathbf{E}(y|x)$. $\mathbf{E}(y|x)$ depends monotonically on x, if

$$x^1 \geq x^2 \Rightarrow \mathbf{E}\left(y|x^1\right) \geq \mathbf{E}\left(y|x^2\right),$$

where $x^1 \geq x^2$ is a partial ordering on $\boldsymbol{X}$ defined by $x_i^1 \geq x_i^2$ for $i = 1, 2, \ldots, k$.

In this paper, y states for the price of a house and it is estimated by using a monotone decision tree on the basis of the vector of characteristics of the house, x.

It can be shown in many cases that monotone models perform better than non-monotone models if monotonicity is present in the problem. This is mainly due to the fact that monotone models suppress overfitting. Some data mining algorithms can be applied to cases where the data set is partially monotone (Bioch and Popova, 2002; Daniels and Kamp, 1999; Potharst and Feelders, 2002) whereas other are restricted to the cases where the data set is totally monotone (see Definition 2) (Makino, Suda, Ono and Ibaraki, 1999; Potharst and Feelders, 2002).

There are quite a number of contributions in the literature that discuss monotonicity and measures for monotonicity of models derived from data. In Daniels and Kamp (1999), a monotonicity index to measure the degree of monotonicity of a neural network with respect to each input variable is defined. The value of this index is between zero, indicating a non-monotone relationship, and 1, indicating a monotone relationship. To test whether a given decision tree is monotone or not, Potharst (1999) describes a procedure using the maximal and minimal elements

of the leaf nodes of the decision trees. The degree of the non-monotonicity of the tree is computed as percentage of non-monotone leaf nodes respective to the total number of leaves. In Ben-David (1995), another measure for the degree of non-monotonicity of a decision tree is proposed, which gives equal weight to each pair of non-monotone leaf nodes. A modification of this measure is given in Potharst and Feelders (2002).

All these measures express the degree of monotonicity after a model has been derived from data. However, in practice, one would like to check whether or not a given data set is monotone beforehand in order to verify the assumptions of theory. One obvious question, therefore, is how to measure the degree of monotonicity of a data set. A straightforward method is to compute the fraction of monotone pairs with respect to the total number of pairs in a data set. Another measure is the number of monotone points or the number of label changes required in the algorithm of Sect. 4. Apart from measures we also need benchmark data sets to compare with. In the next section, we define a class of benchmark data used further to compare with the data set under study.

3 Benchmark data sets

Suppose B_π^N denotes the ensemble of samples of N points drawn from a probability distribution $\pi(x, \ell)$. Here π is defined in $X \times L$, where X is a subset of the k-dimensional space $\Re^k$ and L is a totally ordered set of labels. In the discrete case we have $L = \{1, 2, \ldots, \ell_{max}\}$ and in the continuous case $L \in \Re^k$ or $L \in \Re^+$ (pricing model). A sample D (the data under study) drawn from B_π^N is a data set of N points and we use the notation $D = (x^n, \ell(x^n))_{n=1}^N$ throughout the paper. Given $D \in B_\pi^N$, we define a class of benchmark data, $B_{\pi_0}^N \in B_\pi^N$, the set of samples of N points drawn from probability distribution $\pi_0(x, \ell)$. For this class of benchmark data sets, we assume that the explanatory variables and the labels are independent i.e.

$$\pi_0(x, \ell) = \pi_1(x) \,.\, \pi_2(\ell).$$

Furthermore, we assume that π_1 has a probability distribution with density ρ_1, defined on a subset of X and π_2 is a discrete probability measure if L is discrete or has a density ρ_2 if L is continuous.

The benchmark data sets, denoted by Benchmark-1, are defined as the collection of all data sets D° generated with the same structure of independent variables as D and labels drawn randomly from the labels in D. So, $D^\circ = (x^n, m(x^n))_{n=1}^N$, where $\{m(x^n)\}_{n=1}^N$ is a randomly generated permutation of $\{\ell(x^n)\}_{n=1}^N$.

For this class of benchmark data we consider two measures for the degree of monotonicity, namely the expectation value of the fraction of monotone pairs and the expected number of monotone points (see Definition 1). By comparing these measures with the respective indicators obtained from a real data set, one can verify the degree of monotonicity of the data under study.

Definition 1. We call z a monotone point if for all $y \in D$:

$$y \geq z \Rightarrow \ell(y) \geq \ell(z) \text{ and } y \leq z \Rightarrow \ell(y) \leq \ell(z)$$ □

Definition 2. A data set is monotone if all points are monotone. □

An alternative measure for the degree of monotonicity is the number of label changes necessary to convert a non-monotone into monotone data set. The number of label changes is computed in the algorithm for relabeling in Sect. 4. The algorithm transforms non-monotone into a monotone data set by relabeling some of the points. There are several reasons for doing this. One reason is to remove noise, another is to adjust incomplete data in order to capture and describe better the latent relationship between the dependent variable and missing information. For instance, in the case study described in Sect. 5, factors not listed in Table 1 such as availability of shopping centre, sport facilities, etc., may play important role in determining the house price, which is not explicitly shown in the recorded data. This can lead to obtaining unreliable models with high variance on unseen data. Therefore, an appropriate modification of the data set such as transformation of a non-monotone into monotone data set can capture better implicit dependences between completely missing information and the decision variable as well as to reduce the model variance.

4 Algorithm for relabeling

The objective of the algorithm is to transform a given data set into monotone one by changing the value of the dependent variable. The idea is to reduce the number of non-monotone pairs by relabeling one data point in each step. In order to do this, a data point is chosen for which the increase in correctly labelled points is maximal (this is not necessarily the point which is involved in the maximal number of non-monotone pairs). The process is continued until the data set is monotone (see Definition 2).

Let $D = (x^n, \ell(x^n))_{n=1}^N$ denote the initial data set and $Q(D)$ denotes the set of all non-monotone points in D.

For each data point $x \in Q(D)$, we define:

$$Down(x) = \{y \in X | y < x\},$$
$$Up(x) = \{y \in X | y > x\}.$$

Hence, all points $y \in Down(x) \cup Up(x)$ are incomparable to x.

Furthermore, we define $A_\alpha(x) \subset Down(x)$ and $B_\beta(x) \subset Up(x)$ by:

$$A_\alpha(x) = \{y < x | \text{ label } (y) = \alpha\} \text{ for } \alpha \in L,$$
$$B_\beta(x) = \{y > x | \text{ label } (y) = \beta\} \text{ for } \beta \in L.$$

Let a_α and b_β denote the number of points in $A_\alpha(x)$ and $B_\beta(x)$, respectively and c_x denotes the number of the points in $Down(x) \cup Up(x)$, i.e. this is the number of all points comparable to x.

Furthermore, we define

ℓ_{min}, ℓ_{max} – the minimum and maximum of the labels in D,
ℓ_{maxDn} – the maximum of the labels in $Down(x)$,
ℓ_{minUp} – the minimum of the labels in $Up(x)$ and
N_{ℓ_x} – the total number of points correctly labelled with respect to x for the current label of x,ℓ_x, i.e.,

$$N_{\ell_x} = a_{\ell_{\min}} + \ldots + a_{\ell_x} + b_{\ell_x} + \ldots + b_{\ell_{\max}}.$$

We assume that all data points in the data set D are unique, i.e., no points are represented twice. For each data point $x \in Q(D)$ we compute the maximal increase, I_{max}, in the number of correctly labelled points with respect to x, if the label of x is changed into ℓ', where $\ell' \in L$. If there is more than one label with the same maximal increase in correctly labelled points, we choose the closest label to the current label of x. Finally, we select a point $x \in Q(D)$ for which I_{max} is the largest and change its label. This process is repeated until the data set is monotone. The algorithm outline is given in the Appendix.

In general, the points correctly labelled with respect to x are all points incomparable to x as well as the points in $A_{\ell_{\min}} \cup \ldots \cup A_{\ell_x}$ and $B_{\ell_x} \cup \ldots \cup B_{\ell_{\max}}$. Since the number of the points incomparable to x is constant and they do not contribute to I_{max}, we may completely ignore them.

The correctness of the algorithm follows from Lemma 1 and Lemma 2. Lemma 1 states that it is always possible to reduce the number of non-monotone pairs by changing the label of only one point, as long as the data set is non-monotone. In Lemma 2, it is shown that there is a canonical choice for the new label for which a maximal reduction can be obtained. There may be more than one label for which this can be achieved but these are all smaller or all larger than the current label of the point, so the closest one is chosen, which is unique.

Lemma 1. *Let D^k denote the data set D after k-iterations. If $Q(D^k) \neq \emptyset$ there is at least one point $x \in Q(D^k)$ that can be relabelled such that the number of non-monotone pairs is reduced.*

Lemma 2. *Suppose that the maximal increase I_{max} in correctly labelled points with respect to x can be obtained by at least two labels r and s, $r < s$. Then $r < s < \ell_x$ or $\ell_x < r < s$, where ℓ_x is the label of x.*

The proofs of the lemmas are given in Daniels and Velikova (2003).

Here, we would like to make two remarks with regard to the efficiency of the algorithm.

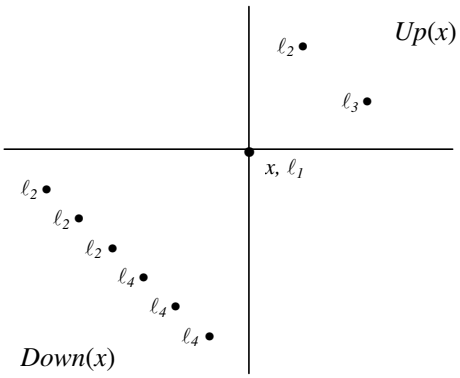

Fig. 1. Distribution of all points in D with respect to x

Remark 1 (Efficiency improvements). It is possible to reduce the number of label checks in the relabeling process for each point $x \in Q(D)$. Depending on the values of ℓ_{maxDn}, ℓ_{minUp} and ℓ_x we consider three cases for the range of labels, ℓ', that have to be checked as candidates.

1. IF $\ell_{maxDn} < \ell_{minUp}$ AND $\ell_x < \ell_{maxDn}$
   THEN $\ell' = \ell_{maxDn}$.
2. IF $\ell_{maxDn} < \ell_{minUp}$ AND $\ell_x > \ell_{minUp}$
   THEN $\ell' = \ell_{minUp}$.
3. IF $\ell_{maxDn} > \ell_{minUp}$
   THEN $\ell' \in [\ell_{minUp}, \ell_{maxDn}]$.

In all these cases, it can be shown that the maximal increase in the number of correctly labelled points with respect to x can be obtained only for the values given above.

The algorithm can be further improved by reducing the number of candidate points considered for relabeling. To see this, compute the number of points comparable to each point $x \in Q(D)$, c_x, and sort $Q(D)$ in descending order by c_x. Then, the maximal increase in correctly labelled points with respect to x, $I_{max}(x)$, is computed. Now, all points $y \in Q(D)$ with $c_y < I_{max}(x)$ can be skipped in the next step because $I_{max}(y) < I_{max}(x)$.

Remark 2 (Example). In general, there is no a straightforward way to find directly the point with a maximal increase in the number of correctly labelled points. All labels in the range defined in Remark-1 have to be considered for relabeling because the dependence of the increase in correctly labelled points on the label can be arbitrary. This is illustrated in the following example.

Let $D = (x^n, \ell(x^n))_{n=1}^9$ denote a data set of 9 points (Fig. 1). We focus on the point x with label ℓ_1.

We now compute the increase/decrease in the number of correctly labelled points if we relabel x with $\ell' \neq \ell_1$. The results are shown in Fig. 2.

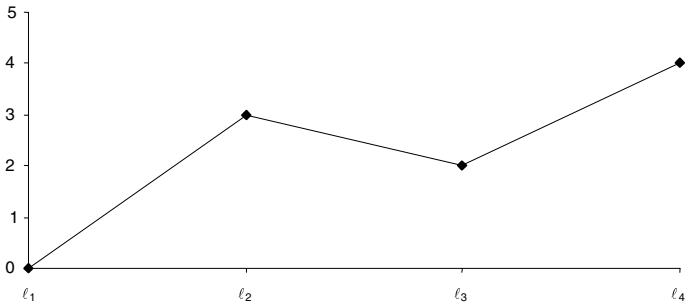

Fig. 2. The increase/decrease in the number of correctly labelled points with respect to x for $\ell' \in [\ell_1, \ell_4]$

It is obvious that the maximal increase is obtained for ℓ_4 and $I_{max} = 4$. Furthermore, it is easily seen that for all other points in D, $I_{max} \leq 3$. Consequently, x is the point with the maximal increase.

In order to examine the performance of the algorithm, an experimental study was conducted using artificially generated data. We firstly generated a data set, D_1, with random points uniformly distributed between 0 and 1 and computed the label of each point by applying a monotone function to the independent variables. Then, the monotone data set was converted into a non-monotone one, D_2, by adding random noise to the labels and the algorithm for relabeling was applied to the modified data to obtain D_3. In the next step, mean-squared error (MSE) is used as a performance measure to check to what extent the algorithm restores the original data. We compute

$$\mathbf{MSE_{mon}} = \frac{1}{N}\sum_{i=1}^{N}\left(\ell_i^{D_3} - \ell_i^{D_1}\right)^2 \text{ and } \mathbf{MSE_{nonmon}} = \frac{1}{N}\sum_{i=1}^{N}\left(\ell_i^{D_2} - \ell_i^{D_1}\right)^2,$$

where N is the number of points in the data set, ℓ^{D_j} is the label set in the data set D_j. This experiment was repeated 10 times with different numbers of points, independent variables and percentages of noise ranging from 7% to 15%. Depending on the number of the explanatory variables, several monotone functions were used to construct the initial label such as $x_1 * sin\frac{\pi}{2}x_2$ based on two variables x_1 and x_2. The results, summarized in Table 1, show that the cleaned data are much closer to the original one than the noisy data. The algorithm for relabeling was implemented in MATLAB, a powerful language providing an interactive environment for algorithm development, data analysis, simulation and technical computing.

5 Case study on house pricing

In this section, we apply the methods described above on house price prediction. In the first part, we briefly present the data set used here. Monotonicity of the data is inspected using the measures and benchmark data introduced in Sect. 3. Then

Table 1. Results after implementation of the algorithm on artificially generated data sets

# points in a data set	# independent variables	# label changes	Noise	MSE_{mon}	MSE_{nonmon}
100	2	10	10%	0.0008	0.0211
100	2	15	15%	0.0015	0.0240
100	3	5	12%	0.0014	0.0055
100	3	12	15%	0.0029	0.0133
100	5	6	14%	0.0079	0.0292
200	2	12	7%	0.0009	0.0224
200	2	29	15%	0.0006	0.0518
200	3	14	10%	0.0044	0.0250
200	5	6	12%	0.0152	0.0222
200	5	15	15%	0.0244	0.0885

Table 2. Definition of the model variables

Symbol	Definition
TOTSPACE	Total flat area
LIVSPACE	Living room area
KITSPACE	Kitchen
DISTKM	Distance in km from the center
ROOMS	Number of rooms
BRICK	Brick flat or not

Table 3. The correlation coefficients between the explanatory variables and the flat price

Variable	TOTSPACE	LIVSPACE	KITSPACE	DISTKM	ROOMS	BRICK
Corr. coef.	0.88	0.85	0.65	−0.37	0.74	0.42

the algorithm for relabeling is applied to obtain monotone data. Decision trees are constructed based on the modified and original data. Finally, the performance of the trees derived from the monotone and non-monotone data are compared.

The basic principle of a hedonic price model is that the consumption good is regarded as a bundle of characteristics for which a valuation exists (Harrison and Rubinfeld, 1978). The price of the good is determined by a combination of these valuations:

$$P = P(x_1, x_2, \ldots, x_n).$$

In the case study presented below we want to predict the flat price given a number of characteristics of the house. The data set consists of 150 observations of flats in the city of Moscow. In the original data set, there are 10 explanatory variables and for each of them, the correlation coefficient with the flat price is calculated. For the purposes of the current case study, six variables with the highest correlation are chosen (Table 2).

The correlation matrix (Table 3) suggests, for example, that the total flat area (TOTSPACE) and the number of rooms (ROOMS) are one of the most important determinants of the housing value. The direction of influence corresponds to common sense: more area and rooms will, in general, result in a higher flat value. In

Table 4. Degree of monotonicity of the Moscow data compared with a benchmark

Indicators	Moscow data	Benchmark-1
Number of points	150	150
Percentage of monotone pairs	97.17%	92.39%
Monotone points	25	6

addition, for computational and analytical convenience, the variable DISTKM is transformed in order to synchronize the direction of influence of all explanatory variables on the flat price.

Of all 11175 distinct pairs of observations, 1699 are comparable, and 316 are non-monotone. To verify the monotonicity of this data set, we compare the degree of monotonicity with the benchmark data defined in Sect. 3. The figures are presented in Table 4.

The results show the existence of a monotone relationship between the dependent and independent variables in the data set under study. Note, that the expected percentage of monotone pairs for a random generated data set with the same number of explanatory variables (6) and the same label set where both distributions are uniform is 98.45%. However, the number of comparable pairs (~ 316) in a random data set with large number of labels and uniform distribution of the points and labels is significantly less than that of the Moscow data, which explains the larger degree of monotonicity in a random data set.

The algorithm for relabeling applied to the Moscow data set leads to 54 label changes. We now generate decision trees from the original and cleaned data. This is done by using a modification of the tree-based algorithm presented in Potharst and Feelders (2002). The algorithm was developed in S-PLUS, where the original program was implemented and in many respects it is similar to the CART program described in Breiman, Friedman, Olshen and Stone (1984). The program only makes binary splits using mean-squared error (MSE) as the splitting criterion. The split of each node is determined by one selected attribute, x, say. If x is continuous the split is of the form $x < c$ or $x \geq c$, for some constant c. If x is categorical, the split is of the form $x \in S$ or $x \notin S$, where S is a non-empty subset of x's possible categories. As described in Breiman, Friedman, Olshen and Stone (1984), the partitioning process is applied recursively to each leaf continuing until all leaves are pure, i.e., contain cases with a unique label. The final tree is denoted by T_{max}. Since this tree almost certainly overfits the data, cost-complexity pruning is applied to generate a nested sequence of minimizing subtrees ($T_{max} > \ldots > \{t\}$, where $\{t\}$ is the root node of the tree). The basic idea is to assign a complexity penalty, determined by a parameter α, to the size of the tree and then to find the sequence of smallest minimizing subtrees at different values of α. From this sequence, the best monotone subtree is selected on the basis of validation set performance (explained below), which is the main difference from the algorithms presented in Breiman, Friedman, Olshen and Stone (1984) and Potharst and Feelders (2002), where the choice of the best subtree is based only on the test set performance irrespective of the

type of the tree (monotone/non-monotone). The monotonicity of a tree is checked by comparing the minimum and maximum element of the leaf nodes (Potharst and Feelders, 2002). Finally, the generalisation (prediction) error of the chosen model is computed using separate test set.

To obtain a statistically sound assessment of the two methods, the following experiment was carried out 15 times. The original data set is randomly partitioned into a construction set of 113 observations (75%) and test set of 37 observations (25%). The construction set was further randomly separated into a training set of 75 observations (50%) and validation set of 38 observations (25%). The training set was used to generate a tree of maximal size as explained above and to construct a sequence of subtrees using cost-complexity pruning. From this sequence, the best monotone tree was selected on the basis of the error rate computed as the mean-squared error on the validation set (in case of a tie, the smallest tree was chosen). The monotonicity of a tree is checked as explained above. The random partition into training and validation sets was repeated 5 times resulting in a sequence of 5 trees, from which the one with the lowest error was chosen as a final tree. In order to evaluate the performance of the final tree, the generalisation error was computed on the test set. The algorithm outline is given in the Appendix.

The same experiment was carried out with the cleaned data. The only difference is in the way the error is computed. Instead of using a test set of 37 observations from the cleaned data, we computed the generalisation error on the basis of the same 37 observations from the original data, which were used as the test set in the previous experiment. Thus, the model is constructed from the cleaned data, whereas the performance is measured on the original data.

In our experiments it turned out that the non-monotone trees generated by the tree construction algorithm have comparable performance but are much larger than monotone trees (as shown in Fig. 3 in the Appendix). This has been also found in a previous study (Potharst and Feelders, 2002). Furthermore, for problems where monotonicity properties are present in the domain such as house pricing, monotone models are easier to understand than their non-monotone counterparts as they are in accordance with the decision makers' expertise. In other words, non-monotone models are much harder to interpret as they present inconsistent and less intuitive dependencies. Therefore, only monotone trees are selected in the tree-based algorithm applied to the present case study. A summary of the results from the experiments is given in Table 5.

Table 5. Results from the experiments with the monotone and non-monotone Moscow data

Indicators	Mean		Variance	
	Monotone data	Non-monotone data	Monotone data	Non-monotone data
Error rate on test set	0.49	1.06	0.43	0.89
Number of monotone trees generated after pruning a large tree	4.9	2.4	0.89	0.83
Number of leaf nodes	4.9	2.2	1.84	2.89

Table 6. The p-values yielded from the statistical t-tests for each of the indicators

Indicators	P-value
Error rate on rest set	0.3%
Number of monotone trees generated after pruning a large tree	0.0%
Number of leaf nodes	0.0%

To check the significance of the results we performed three t-tests. Since the test set in both experiments is the same, there is a natural pairing of the error rates estimated from monotone and non-monotone data. To test the mean difference of the error rates, we use a paired t-test of the null hypothesis that the trees derived from both data sets have the same classification error against the one-sided alternative. For the other two indicators (number of monotone trees generated after pruning a large tree and number of leaf nodes) we use standard t-tests of the null hypotheses that the means are equal against the one-sided alternatives. The p-values obtained from the three tests are reported in Table 6.

Table 6 shows that the first null hypothesis (error of trees) can be rejected at 5% significance level. Furthermore, the results show that the average number of monotone trees derived from the monotone data is significantly larger than that for non-monotone data. In 50% of cases, the only monotone tree generated by the non-monotone data is the root, which explains the smaller number of leaf nodes for the monotone trees generated by the raw data. Finally, the significantly lower or comparable variances of the indicators in Table 5 show that the monotone trees constructed from the cleaned data are more stable than that generated from the raw data.

6 Conclusion

In the present paper, we have developed methods for the derivation of monotone decision models from non-monotone data. Measures are defined to express the degree of monotonicity of data sets and to compare data with simulated benchmark data. An algorithm to transform non-monotone into monotone data is constructed. In a simulation study, using artificially generated monotone data with noise, it is shown that the algorithm is capable of reducing the noise level considerably. The performance of the methods developed is tested on a real data set on house pricing. Decision trees are constructed on the base of the cleaned (monotone) and original data. The results show that the error of the trees generated from the monotone data is significantly smaller than the classification error of the trees generated from the non-monotone data. Furthermore, the cleaned data yield more monotone trees and trees are more stable compared to the trees generated from the raw data.

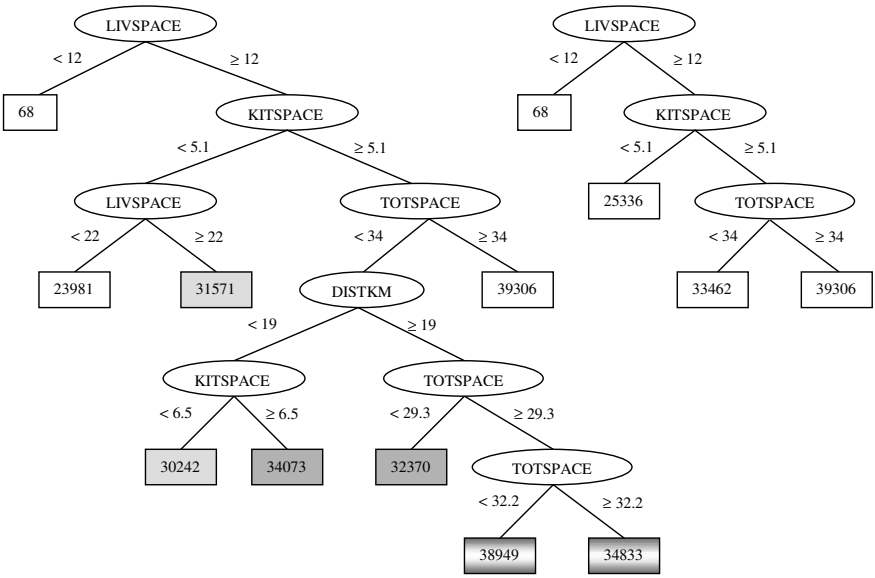

Fig. 3. An example of non-monotone (left) and monotone tree obtained from the original Moscow data. The patterns represent the non-monotone leaf-pairs in the non-monotone tree. The estimated error rate of the non-monotone tree is 1.01 and the estimated error rate of the monotone tree is 1.02

Appendix

Algorithm-1: Algorithm for relabelling

Step 1 – Initialisation:
Compute $Q(D)$ on the basis of D
Step 2 – Main program
Step 2.1 As long as $Q(D) \neq \emptyset$
For each data point $x \in Q(D)$ compute
$I_{max} = \max\{N_{\ell'} - N_{\ell_x} | \ell' \in L\}$.
$\Lambda-$set of indices ℓ' for which $N_{\ell'} - N_{\ell_x}$ is maximal.
Form a triple (x, I_{max}, λ) where $\lambda \in \Lambda$ is the closest label to ℓ_x
(in Lemma 2 it is shown that λ is unique).
Step 2.2 From all triples, choose the one where I_{max} is maximal and change the label into ℓ'.
Step 2.3 Update $Q(D)$ on the basis of the modified data set D.

Algorithm-2: The modified tree-based algorithm used in the house pricing case study

Data := data set of N observations
For i := 1:15 **do**
Construction_set := Random sample of $\frac{3}{4}N$ observations from *Data*
Test_set := Complement set of *Construction_set*
For j := 1:5 **do**
Train_set := Random sample of $\frac{1}{2}N$ observations from *Construction_set*
Validation_set := Complement set of *Train_set*
Construct T_{max} based on *Train_set*
Construct *Trseq* := a nested sequence of minimizing subtrees ($T_{max} > \ldots > \{t\}$) by applying cost-complexity pruning to T_{max}
Bmtr := the best monotone tree from *trseq* selected on the basis of *Validation_set* performance
Classify *Test_set* data using *Bmtr* and determine the performance of the model

References

Ben-David A (1995) Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning* **19**, 29–43

Bioch JC, Popova V (2002) Monotone decision trees and noisy data. *ERIM Internal Report*, Erasmus University Rotterdam, No 206

Breeze J, Goldman R, Wellman M (1994) Introduction to the special section on knowledge-based construction of probabilistic and decision models. *IEEE Transactions on Systems, Man and Cybernetics*, vol 24, pp 1577–1579

Breiman L, Friedman JH, Olshen RA, Stone CT (1984) Classification and regression trees. Wadsworth

Daniels HAM, Kamp B (1999) Application of MLP networks to bond rating and house pricing. *Neural Computation and Applications* **8**: 226–234

Daniels HAM, Velikova M (2003) Derivation of monotone decision models from non-monotone data. *Center Internal Report*, Tilburg University No 2003-30

Garofalakis M, Rastogi R, Shim K (2002) Mining sequential patterns with regular expression constraints. *IEEE Transactions on Knowledge and Data Engineering* **14**: 530–552

Harrison O, Rubinfeld D (1978) Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management* **53**: 81–102

Makino K, Suda T, Ono H, Ibaraki T (1999) Data analysis by positive decision trees. *IEICE Transactions on Information and Systems*, Volume E82-D, No 1, pp 76–88

Mukarjee H, Stern S (1994) Feasible nonparametric estimation of multiargument monotone functions. *Journal of the American Statistical Association* **89**(425): 77–80

Potharst R (1999) Classification using decision trees and neural nets. Erasmus University Rotterdam, *SIKS Dissertation Series* No. 99-2

Potharst R, Feelders A (2002) Classification trees for problems with monotonicity constraints. SIGKDD Explorations Newsletter, vol 4, Issue 1

Rajagopalan B, Isken M (2001) Exploiting data preparation to enhance mining and knowledge discovery. *IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews*, vol 31, pp 460–467

Wang S (1994) A neural network method of density estimation for univariate unimodal data. *Neural Computation & Applications* **2**: 160–167