Isotonic Classification Trees

Rémon van de Kamp, Ad Feelders, and Nicola Barile

Utrecht University, Department of Information and Computing Sciences,
P.O. Box 80089, 3508TB Utrecht, The Netherlands,
{rpkamp,ad,barile}@cs.uu.nl

Abstract. We propose a new algorithm for learning isotonic classification trees. It relabels non-monotone leaf nodes by performing the isotonic regression on the collection of leaf nodes. In case two leaf nodes with a common parent have the same class after relabeling, the tree is pruned in the parent node. Since we consider problems with ordered class labels, all results are evaluated on the basis of L_1 prediction error. We experimentally compare the performance of the new algorithm with standard classification trees.

1 Introduction

In many applications of data analysis it is reasonable to assume that the response variable is increasing (or decreasing) in one or more of the attributes or features. For example, the sale price of a house - all else equal - increases with lot size, and according to economists people tend to buy less of a product if its price increases. Such relations between response and attribute are called monotone. Monotonicity constraints can, for example, also be found in medicine [20, 6] and law [12]. Besides being plausible, monotonicity may also be a desirable property of a decision model for reasons of explanation, justification and fairness. Pazzani et al.[17], show that rules learned with monotonicity constraints were significantly more acceptable to medical experts than rules learned without the monotonicity restrictions.

Because the monotonicity constraint is quite common in practice, many data analysis techniques have been adapted to be able to handle such constraints. In this paper we present a new algorithm, called ICT, for learning monotone classification trees for problems with ordered class labels. Our approach differs from earlier monotone tree algorithms such as [5, 18, 11] in that we adjust the probability estimates in the leaf nodes in case of a violation. This is done in such a way that, subject to the monotonicity constraint, the sum of absolute prediction errors on the training sample is minimized. Another new element of our algorithm is that we can also handle problems where some, but not all, attributes have a monotone relation with the response. The performance of the new algorithm is evaluated through experimental studies on real life data sets.

This paper is organized as follows. In the next section we introduce the basic concepts and notation that will be used throughout the paper. Since the isotonic regression is an important technique for our algorithm, we discuss it shortly in section 3. In section 4 we discuss the main contribution of this paper, the Isotonic Classification Tree (ICT) algorithm. ICT is evaluated in section 5 where we present the results of experiments on real data. Section 6 concludes.

2 Preliminaries

Let $\mathcal{X}$ be a *feature space* $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_p$ consisting of vectors $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ of values on p features or attributes. We assume that each feature takes values x_i in a linearly ordered set $\mathcal{X}_i$. The partial ordering $\preceq$ on $\mathcal{X}$ will be the ordering induced by the order relations of its coordinates $\mathcal{X}_i$: $\mathbf{x} = (x_1, x_2, \ldots, x_p) \preceq \mathbf{x}' = (x'_1, x'_2, \ldots, x'_p)$ if and only if $x_i \leq x'_i$ for all i. Furthermore, let $\mathcal{Y}$ be a finite linearly ordered set of *classes*. Without loss of generality, we assume that $\mathcal{Y} = \{1, 2, \ldots, k\}$ where k is the number of classes.

A *monotone* classification rule is a function $c : \mathcal{X} \to \mathcal{Y}$ for which

$$\mathbf{x} \preceq \mathbf{x}' \Rightarrow c(\mathbf{x}) \leq c(\mathbf{x}') \tag{1}$$

for all instances $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. A data set $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$ is *monotone* if for all i, j we have $\mathbf{x}_i \preceq \mathbf{x}_j \Rightarrow y_i \leq y_j$.

The classification rules we consider are univariate binary classification trees. For such trees, at each node a split is made using a test of the form $X_i < d$ for some $d \in \mathcal{X}_i, 1 \leq i \leq p$. Thus, for a binary tree, in each node the associated set $t \subset \mathcal{X}$ is split into the two subsets $t_\ell = \{\mathbf{x} \in t : x_i < d\}$ and $t_r = \{\mathbf{x} \in t : x_i \geq d\}$. The classification rule that is induced by a decision tree T will be denoted by c_T.

For any node or leaf t of T, the subset of the instance space corresponding to that node can be written

$$t = \{\mathbf{x} \in \mathcal{X} : \mathbf{a} \preceq \mathbf{x} \prec \mathbf{b}\} = [\mathbf{a}, \mathbf{b}) \tag{2}$$

for some $\mathbf{a}, \mathbf{b} \in \overline{\mathcal{X}}$ with $\mathbf{a} \preceq \mathbf{b}$. Here $\overline{\mathcal{X}}$ denotes the extension of $\mathcal{X}$ with infinity-elements $-\infty$ and ∞. In some cases we need the infinity elements so we can specify a node as in equation (2).

Below we will call $\min(t) = \mathbf{a}$ the *minimal element* and $\max(t) = \mathbf{b}$ the *maximal element* of t. Together, we call these the *corner elements* of node t. If $\min(t) \prec \max(t')$ then node t contains points that are smaller than some points in node t', hence the monotonicity constraint requires that the label assigned to node t should not be bigger than the label assigned to node t'. Therefore, we call a pair of leaves t, t' non-monotone if $\min(t) \prec \max(t')$ and $c_T(t) > c_T(t')$ [19]. A tree is non-monotone if it contains at least one non-monotone leaf pair.

It is customary to evaluate a classifier on the basis of its error-rate or 0/1 loss. For classification problems with ordered class labels this choice is less obvious. It makes sense to incur a higher cost for those misclassifications that are “far” from the true label, than to those that are “close”. One loss function that has this property is L_1 loss:

$$L_1(i, j) = |i - j| \qquad\qquad i, j = 1, \ldots, k \tag{3}$$

where i is the true label, and j the predicted label. We note that this is not the only possible choice. One could also choose L_2 loss for example, or another loss function that has the desired property that misclassifications that are far from the true label incur a higher loss. Nevertheless, L_1 loss is a reasonable candidate, and in this paper we confine our attention to this loss function.

To illustrate the concepts introduced, we discuss a small example. Let $\mathcal{Y} = \{1, 2, 3\}$ and suppose we have a tree with two input attributes with $X_1, X_2 \in [0, 1]^2$. The tree is given in Figure 1 on the left. The corresponding partitioning of the input space is depicted in Figure 1 on the right.

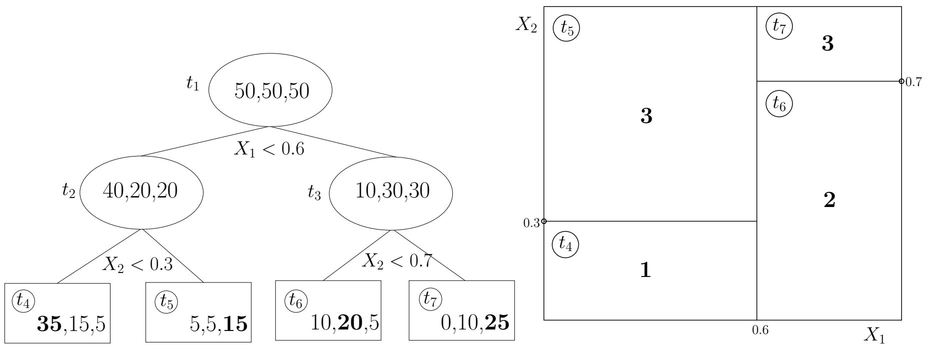

Fig. 1. Left: Classification tree for three-class problem. Numbers in nodes are the counts for class 1, 2 and 3 respectively. In the leaf nodes, the counts of the median class are shown in boldface. The circled labels in the leaf nodes correspond to the labeled regions in the picture on the right. Right: Partitioning of input space corresponding to the tree on the left. The class labels assigned to the different rectangles are shown in boldface. Rectangle t_5 and t_6 form a nonmonotone leaf pair.

To minimize L_1 loss we allocate to the median in leaf nodes, which leads to the class labels as shown in boldface in Figure 1 on the right. Leaf node t_5 and t_6 form a nonmonotone pair, since t_5 has a higher class label but contains points that are smaller than some points in t_6: the lower left corner of t_5 is smaller than the upper right corner of t_6.

3 The Isotonic Regression

In this section we give a general description of the isotonic regression. In section 4 we discuss its application to making trees monotone.

Let $Z = \{z_1, z_2, \ldots, z_m\}$ be a nonempty finite set of constants and let $\preceq$ be a partial order on Z. Any real-valued function f on Z is *isotonic* with respect to $\preceq$ if, for any $z, z' \in Z$, $z \preceq z'$ implies $f(z) \leq f(z')$. We assume that each element z_i of Z is associated with a real number $g(z_i)$; these real numbers typically are estimates of the function values of an unknown isotonic function on

Z. Furthermore, each element of Z has associated a positive weight $w(z_i)$ that typically indicates the precision of this estimate. An isotonic function g^* on Z now is an *isotonic regression* of g with respect to the weight function w and the partial order $\preceq$ if and only if it minimizes the sum

$$\sum_{i=1}^{m} w(z_i)\ [f(z_i) - g(z_i)]^2 \tag{4}$$

in the class of isotonic functions f on Z. Brunk [8] proved the existence of a unique g^*.

Any real-valued function f on Z is *antitonic* with respect to $\preceq$ if, for any $z, z' \in Z$, $z \preceq z'$ implies $f(z) \geq f(z')$. The antitonic regression of g is defined completely analogous to the isotonic regression as the function that minimizes (4) within the class of antitonic functions. The *isotonic* regression with respect to a partial order is equivalent to the *antitonic* regression with respect to the inverse of that order.

The best time complexity known for an exact solution to the isotonic regression problem for arbitrary partial order is $O(m^4)$ [16]. It is based on a divide-and-conquer strategy that involves solving at most m maximal flow problems.

4 Isotonic Classification Trees

The ICT algorithm can in principle be combined with any standard classification tree algorithm. Here we modify a CART-like algorithm [7] to incorporate the monotonicity constraints. The main principle of ICT is that it makes trees monotone by relabeling its leaf nodes. This is done in such a way that of all monotone trees that can be obtained by relabeling the leaf nodes, the one produced by ICT has lowest absolute error on the training data. The relabeling is not computed directly, but is obtained by first adjusting the probability estimates in the leaf nodes (using the isotonic regression), and then allocating each leaf node to the (smallest) median of its estimated class distribution.

We first discuss the growing of trees in ICT. Then we discuss the adjustment of probability estimates in the leaf nodes, and the corresponding relabeling, which may lead to pruning the tree. We also discuss the incorporation of partial monotonicity constraints in ICT.

4.1 Growing trees

Let $\tilde{T}$ denote the collection of leaf nodes of tree T, $n(t, j)$ denote the number of observations in t with class label j, and let

$$\hat{P}_j(t) = \frac{n(t,j)}{n(t)}, \qquad t \in \tilde{T}$$

denote the relative frequency of class label j in node t. Furthermore, let

$$\hat{F}_i(t) = \sum_{j \leq i} \hat{P}_j(t), \qquad t \in \tilde{T}$$

denote the unconstrained maximum likelihood estimate of

$$F_i(t) = P(y \leq i \mid \mathbf{x} \in t), \qquad t \in \tilde{T}.$$

Because the median is known to minimize L_1 loss, we allocate to the (smallest) median of the estimated class distribution:

$$c(t) = \min_i : \hat{F}_i(t) \geq 0.5, \tag{5}$$

The standard tree growing algorithm is modified in such a way that it records the corner elements of each node. Since the aim is to minimize L_1 loss, the risk for each node is set to be the mean absolute error for that node:

$$r(t) = \sum_{i:\mathbf{x}_i \in t} \frac{|y_i - c(t)|}{n(t)},$$

where $c(t)$ denotes the class allocated to node t and $n(t)$ denotes the number of observations in node t.

To compute the impurity of a node, we use the gini index combined with absolute error:

$$i(t) = \sum_{j \neq k} |j - k| \hat{P}_j(t)\hat{P}_k(t), \qquad j, k \in \mathcal{Y}$$

As usual, ICT chooses the split that maximizes impurity reduction.

4.2 Making the tree monotone

Let $(\tilde{T}, \preceq)$ be the partial order $\preceq$ over $\tilde{T}$ with

$$t \preceq t' \Leftrightarrow \min(t) \prec \max(t'), \qquad t, t' \in \tilde{T},$$

that is, t precedes t' if t contains points that are smaller than some points in t'. Define

$$F_i^*(t), \qquad i = 1, 2, \ldots, k-1; \quad t \in \tilde{T}$$

as the antitonic regression of $g(t) = \hat{F}_i(t)$ with weights $w(t) = n(t)$ and partial order $(\tilde{T}, \preceq)$, for each value $i = 1, 2, \ldots, k-1$. Of course, $F_k^*(t) = 1$ for all $t \in \tilde{T}$. Note that F^* satisfies the stochastic order constraint

$$t \preceq t' \Rightarrow F_i^*(t) \geq F_i^*(t'), \qquad i = 1, \ldots, k. \tag{6}$$

Subsequently, we allocate t to the smallest median of $F^*(t)$:

$$c^*(t) = \min_i : F_i^*(t) \geq 0.5 \tag{7}$$

Because F^* satisfies (6), we have that c^* satisfies the monotonicity constraint [14]

$$t \preceq t' \Rightarrow c^*(t) \leq c^*(t')$$

Furthermore, it can be shown [10, 3] that $c^*(t)$ minimizes L_1 loss

$$\sum_{i=1}^{n} |y_i - c(t(\mathbf{x}_i))|$$

within the class of monotone integer-valued functions $c(\cdot)$ on $\tilde{T}$, where $t(\mathbf{x}_i) = t \in \tilde{T} : \mathbf{x}_i \in t$. In other words, of all monotone classifiers on $\tilde{T}$, c^* is among the ones (there may be more than one) that minimize L_1 loss on the training sample.

To illustrate, we continue the example introduced in section 2. The monotonicity violation between t_5 and t_6 is resolved by performing the antitonic regression on $\hat{F}_1(t)$, and on $\hat{F}_2(t)$, $t \in \{t_4, t_5, t_6, t_7\}$. Note that we have the total order $t_4 \preceq t_5 \preceq t_6 \preceq t_7$ on the leaf nodes in this particular case. The solution is to average $\hat{F}_1(t_5)$ with $\hat{F}_1(t_6)$ and $\hat{F}_2(t_5)$ with $\hat{F}_2(t_6)$ with $n(t_5)$ and $n(t_6)$ as weights:

$$F_1^*(t_5) = F_1^*(t_6) = \frac{5+10}{25+35} = \frac{1}{4} \qquad F_2^*(t_5) = F_2^*(t_6) = \frac{10+30}{25+35} = \frac{2}{3}$$

Using these revised estimates we assign to the median, meaning we assign to class 2 in both t_5 and t_6. Note that the L_1 error of the original nonmonotone tree is: 25+15+15+10=65. By relabeling leaf node t_5 to class 2, the error increases to 70. This is the best possible monontone relabeling of the leaf nodes; for example, relabeling t_6 to class 3 would increase the error to 90.

4.3 ICT Pruning

After relabeling the leaf nodes, there may be pairs t_ℓ and t_r with common parent t that have been assigned the same class label. In that case, the tree is pruned in t, and we apply the normal allocation rule (5) to t. This may result in a nonmonotone tree, in which case the leaf nodes are relabeled again. Also note that pruning may create a new pair t'_ℓ, t'_r with the same class label and a common parent t' in which case the algorithm will prune in t'.

4.4 Algorithm outline

The ICT algorithm is summarized in Algorithm 1. The algorithm takes as input a tree T, and returns a monotone tree T' which is a possibly relabeled and pruned version of T.

If the tree is monotone, it is returned unchanged. However, if the tree is nonmonotone, the antitonic regression computes the new probability estimates in line 4-6. In line 7-10 the leaf nodes are subsequently relabeled. Line 11-14 prune away leaf nodes with the same class label and common parent. The resulting tree may be nonmonotone, hence the recursive call to ICT in line 15.

It should be noted that we combine the ICT algorithm with cost-complexity pruning in the following way. Let $T_1 > T_2 > \ldots > \{t_1\}$ denote the tree sequence

Algorithm 1 ICT(T)

```
if $T$ is monotone then
    return $T$
else
    for $i \in \{1, \dots, k-1\}$ do
        $F_i^* \leftarrow$ AntitonicRegression($\tilde{T}$,$\preceq$,$n(t) : t \in \tilde{T}$,$\hat{F}_i(t) : t \in \tilde{T}$)
    end for
    for all $t \in \tilde{T}$ do
        $F_k^*(t) \leftarrow 1$
        $c_T(t) \leftarrow \min_i F_i^*(t) \geq 0.5$
    end for
    while there are $t_\ell, t_r \in \tilde{T}$ with common parent $t$ and $c_T(t_\ell) = c_T(t_r)$ do
        $T \leftarrow$ prune $T$ in $t$
        $c_T(t) \leftarrow \min_i \hat{F}_i(t) \geq 0.5$
    end while
    $T' \leftarrow$ ICT($T$)
    return $T'$
end if
```

produced by standard cost-complexity pruning, where t_1 denotes the root node of the tree, and $T_j > T_k$ means T_k is obtained by pruning T_j in one or more nodes. We apply ICT pruning to every tree in this sequence (except the root of course) to obtain a sequence of monotone trees $T_1' > T_2' > \ldots > \{t_1\}$. This sequence may be shorter than the original sequence, since sometimes two trees from the original cost-complexity sequence are pruned back to the same tree by ICT.

4.5 Partial monotonicity

In many applications there will be attributes for which there is no reason to assume that they have a monotone relation with the class label. Therefore we extended the ICT algorithm to be able to handle such cases.

The ICT algorithm for partial monotonicity is largely the same as it is for complete monotonicity. We just need to change the partial order used in the antitonic regression and the check that determines if two leafs are non-monotone.

First we define a partially monotone classification rule. Let $\mathcal{X}$ be defined as before, and let $\mathcal{Z} = \times \mathcal{Z}_i$, $i = 1, \ldots, q$. The values $\mathcal{Z}_i$ may be either ordered or unordered. A classification rule $c : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$ is monotone in $\mathbf{X}$ iff

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \forall \mathbf{z} \in \mathcal{Z} : \mathbf{x} \preceq \mathbf{x}' \Rightarrow c(\mathbf{x}, \mathbf{z}) \leq c(\mathbf{x}', \mathbf{z})$$

Our orginal ordering on $\tilde{T}$ was defined in such a way that $t \preceq t'$ if node t contained elements that were smaller than some elements of t'. This was the case when $\min(t) \prec \max(t')$. Now we have to add the constraint that t and t' should have overlapping values on $\mathbf{Z}$. Hence, we define a new partial order $(\tilde{T}, \preceq)$ with $t \subset \mathcal{X} \times \mathcal{Z}$ as follows:

$$t \preceq t' \Leftrightarrow \min(t_{\mathbf{X}}) \prec \max(t'_{\mathbf{X}}) \wedge t_{\mathbf{Z}} \cap t'_{\mathbf{Z}} \neq \varnothing, \qquad t, t' \in \tilde{T}.$$

Here $t_{\mathbf{X}}$ denotes the projection of t on the monotone attributes $\mathbf{X}$.

5 Experiments

In order to evaluate the proposed algorithm, we performed a number of experiments. This section contains information on the datasets, how we pre-processed the data, the experiments and their results. The programs were implemented in `R`[1].

5.1 Datasets

We selected a number of datasets where monotonicity constraints are likely to apply. We used the KC4, PC3, PC4 and PC5 datasets from the NASA Metrics Data Program [15], the Acceptance/Rejection, Employee Selection, Lecturers Evaluation and Social Workers Decisions from A. Ben-David [4], the Windsor Housing dataset [1], the Den Bosch Housing dataset [9], as well as several datasets from the UCI Machine Learning Repository [2]. All datasets except Den Bosch Housing are publicly available. Table 1 lists all the datasets used.

5.2 Pre-processing of the data

ICT makes the harmless assumption that all monotone attributes have an increasing relation with the response. This means that if the actual relation is decreasing, the attribute values have to be inverted. We tested this by looking at the correlation between the attribute and the response. In case of a negative correlation between some attribute x and the response, we transformed the values of x as follows:

$$x_i = x_{max} - x_i + x_{min}, \qquad i = 1, \ldots, n \tag{8}$$

with $x_{max} = \max(x)$, and $x_{min} = \min(x)$.

For datasets with a numeric response that is not a count (Auto MPG, Boston Housing, CPU Performance, Windsor Housing and Den Bosch Housing) we discretized the response values into four separate intervals, each interval containing roughly the same number of observations.

For all datasets from the NASA Metrics Data Program the attribute `ERROR_COUNT` was used as the response. All attributes that contained missing values were removed. Furthermore, the attribute `MODULE` was removed because it is a unique identifier of the module and the `ERROR_DENSITY` was removed because it is a function of the response variable. On the remaining attributes we used the function `stepAIC` with backward elimination in `R` to fit a linear model; attributes that did not occur in the final model were removed from the dataset. Since the distribution of `ERROR_COUNT` was highly skewed (most modules have zero errors) we sampled the modules with zero errors to create a more balanced

[1] http://www.r-project.org/

distribution. High error counts are less frequent than low error counts. In order to increase frequencies, the higher counts were merged into a single class. For example, for KC4, all class labels greater than five were set to five.

For the CPU Performance dataset the machine cycle time in nanoseconds was converted to clock speed in Khz, in order to make it positively correlated with the class label. From this dataset the attributes `Vendor Name`, `Model Name` and `ERP` were removed.

From the Den Bosch Housing dataset the independent attributes `year`, `x-coordinate` and `y-coordinate` were removed.

5.3 Relabeling toward monotonicity

Besides enforcing a monotone model, one can also use prior knowledge about monotonicity by relabeling the dataset to make it monotone. As shown in [22, 13], models learned on relabeled datasets on average perform better than models learned with the original class labels.

Therefore, we also tested ICT on relabeled versions of the original datasets. We computed y^* as the relabeling of the observations that minimizes

$$\sum_{i=1}^{\mathrm{ntrain}} |y_i - y_i'|$$

within the class of monotone relabelings y'. Here ntrain denotes the number of observations in the training sample. The test data was not relabeled in the experiments.

Table 1 summarizes for all datasets the cardinality, the number of attributes after pre-processing, the number of distinct class labels and the L_1 distance between y and y^*. For example, to make the Australian Credit data monotone we have to relabel for a total absolute error of 14. Since Australian Credit has a binary class label, this means that 14 observations have to be relabeled.

5.4 Experimental results

Each of the datasets was randomly divided one hundred times into a training set consisting of four fifth of the data and a test set consisting of the remaining one fifth of the data. On every training set a tree was grown, after which cost complexity pruning was applied to obtain a sequence of trees. The ICT algorithm was applied to each tree in this sequence to obtain a sequence of monotone trees. Subsequently, the test set was used to select the best tree from the original sequence and to select the best tree from the monotone sequence. The test errors of the best standard trees and the best monotone trees were averaged over the one hundred repititions of the experiment.

Table 2 shows the results of the experiments on all datasets. The errors are indicated as the mean absolute error on the test sample. For each column the mean error and the standard deviation of this mean error are indicated, separated

Table 1. Dataset charasterics and relabeling information

Dataset	cardinality	#attributes	#labels	$\sum \lvert y_i - y_i^* \rvert$
Australian Credit	690	14	2	14
Auto MPG	392	7	4	23
Boston Housing	506	13	4	28
Car Evaluation	1728	6	4	21
Den Bosch Housing	119	8	4	10
Empoyee Rej/Acc	1000	4	9	1161
Employee selection	488	4	9	104
Haberman survival	306	3	2	55
Lecturers evaluation	1000	4	5	364
CPU Performance	209	6	4	26
Pima Indians	768	8	2	53
Social Workers Decisions	1000	10	4	375
Windsor Housing	546	11	4	134
KC4	122	4	6	80
PC3	320	15	5	1
PC4	356	16	6	6
PC5	1032	21	6	141

by a ± sign. The lowest error and the lowest number of leafs for each dataset are printed in boldface.

First we consider the results with the original class labels. In that case ICT almost always has a slightly lower error than the standard tree. The two exceptions are the Car evaluation data and dataset PC5. There is no clear winner on the tree size criterion.

On the relabeled data the conclusions are quite different. Now there is no clear winner in terms of the error but ICT clearly has the smaller trees.

Comparing the error on the relabeled and original data, we can conclude that it is beneficial to relabel the training data, since it tends to reduce the error.

It should be noted that the differences found were small, and nowhere significant. Nevertheless it is safe to conclude that in the datasets studied, enforcing a monotone model does not lead to a degradation of the predicitive performance. Hence, when a monotone model is required, or just preferred, such a model can be obtained without loss of predicitive accuracy.

Finally, we discuss our experiments with partially monotone trees. One of the important problems is how to determine which attributes are to be constrained, and which are not. In practice such information may be obtained from domain experts. Here we used a data-based test proposed by [21]. This sometimes resulted in the removal of the constraint for a particular attribute, but the results did not improve compared to complete monotonicity, and are therefore not reported.

6 Conclusions

We have presented a new algorithm, called ICT, for learning monotone classification trees from data. ICT differs from existing monotone tree algorithms in that it relabels the leaf nodes of the tree in case of monotonicity violations: ICT produces the monotone relabeling that minimizes absolute error on the training

Table 2. Results of monotone trees (ICT) and standard trees.

Dataset	Label	Error ICT	Error Standard	#Leafs ICT	#Leafs Standard
Australian	y	0.1426±0.0070	0.1431±0.0068	3.4300±1.9553	**3.3500**±2.2490
Credit	y^*	**0.1418**±0.0078	0.1426±0.0071	3.3600±1.9515	3.5700±2.9241
Auto MPG	y	**0.2982**±0.0292	0.3045±0.0282	**9.2800**±2.7746	10.6300±5.3497
	y^*	0.2985±0.0295	**0.2982**±0.0293	10.1100±2.7484	12.3800±4.5964
Boston	y	0.3966±0.0370	0.4050±0.0376	8.3100±3.1065	**7.7100**±4.9222
Housing	y^*	**0.3861**±0.0334	0.3935±0.0339	8.3200±2.7957	8.5700±5.2073
Car	y	0.0871±0.0181	**0.0836**±0.0164	**27.6200**±5.4417	32.4400±8.4271
Evaluation	y^*	0.0897±0.0190	0.0849±0.0184	28.2300±5.3802	32.3100±7.8787
Den Bosch	y	0.4922±0.0832	0.5165±0.0852	5.5200±1.5274	5.6600±1.9396
Housing	y^*	**0.4755**±0.0829	0.4856±0.0841	**5.5100**±1.4106	5.5500±1.6840
Employee Rej/Acc	y	1.2764±0.0407	1.2926±0.0415	10.3400±3.9778	**8.2200**±3.4629
	y^*	1.1773±0.0242	**1.1627**±0.0208	17.0600±2.2103	19.3900±1.1538
Employee	y	0.4348±0.0369	0.4590±0.0395	**23.6900**±4.3128	26.9500±8.4452
Selection	y^*	0.3829±0.0265	**0.3822**±0.0304	23.9100±2.9063	27.6300±3.9457
Haberman	y	0.2585±0.0146	0.2605±0.0139	2.6000±2.1742	**1.9900**±1.6112
Survival	y^*	**0.2482**±0.0164	0.2484±0.0164	3.8300±2.0003	4.3200±2.4980
Lecturers	y	0.4764±0.0267	0.4903±0.0267	**19.9000**±5.1981	19.9200±8.9563
Evaluation	y^*	0.4151±0.0233	**0.3832**±0.0157	21.5300±3.5773	28.7900±3.1311
CPU Performance	y	0.4556±0.0540	0.4773±0.0554	**8.3300**±2.3401	9.2900±4.4818
	y^*	**0.4323**±0.0486	0.4324±0.0453	8.5900±1.9700	9.8800±2.7016
Pima	y	0.2586±0.0145	0.2619±0.0144	5.6300±3.1866	4.6700±3.1464
Indians	y^*	0.2580±0.0130	**0.2576**±0.0117	**4.3900**±2.7299	4.5200±3.9936
Social Workers	y	0.4707±0.0209	0.4772±0.0181	9.3300±4.3579	**7.6100**±4.5436
Decisions	y^*	0.4309±0.0163	**0.4045**±0.0137	12.6700±3.6350	27.4500±4.7298
Windsor	y	0.6244±0.0328	0.6619±0.0364	17.0100±5.1198	**15.5400**±12.8860
Housing	y^*	**0.5992**±0.0333	0.6103±0.0377	18.7900±4.2433	24.9500±12.0566
KC4	y	1.1871±0.1349	1.2358±0.1474	**4.4300**±2.2031	4.6300±3.2649
	y^*	**1.0153**±0.1298	1.0174±0.1349	4.8400±1.5681	5.0900±1.7529
PC3	y	0.5357±0.0440	0.5363±0.0394	2.6000±1.3780	**2.4900**±1.5986
	y^*	**0.5351**±0.0443	0.5359±0.0399	2.6100±1.4695	**2.4900**±1.5986
PC4	y	**0.5735**±0.0492	0.5835±0.0543	4.9500±2.9418	**4.5100**±3.0600
	y^*	0.5886±0.0617	0.5928±0.0632	5.4200±3.2167	5.1900±4.0394
PC5	y	0.4960±0.0188	0.4948±0.0242	6.1600±4.1310	**5.8400**±4.5543
	y^*	0.4939±0.0189	**0.4904**±0.0229	6.1111±4.4006	6.8800±5.1410

sample. Furthermore, in contrast to existing monotone tree algorithms, ICT can also be applied to partially monotone problems.

Our experiments have shown that ICT usually performed slightly better than standard trees on the original data. After relabeling, the performance of ICT and the standard tree algorithm was virtually identical. It should be noted however that a standard tree algorithm applied to monotone data does not necessarily produce a monotone tree. Therefore, if a monotone model is required, application of a standard algorithm to relabeled data may not be sufficient. Furthermore, on the relabeled data ICT on average produced smaller trees than the standard algorithm. This warrants the conclusion that ICT trees are easier to understand than their somewhat larger and possibly non-monotone counterparts.

References

1. P.M. Anglin and R. Gençay. Semiparametric estimation of a hedonic price function. *Journal of Applied Econometrics*, 11(6):633–648, 1996.
2. A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

3. N. Barile and A. Feelders. Nonparametric monotone classification with MOCA. In F. Giannotti, editor, *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM 2008)*, pages 731–736. IEEE Computer Society, 2008.
4. A. Ben-David, L. Sterling, and Y. Pao. Learning and classification of monotonic ordinal concepts. *Computational Intelligence*, 5:45–49, 1989.
5. Arie Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19:29–43, 1995.
6. D.A. Bloch and B.W. Silverman. Monotone discriminant functions and their applications in rheumatology. *Journal of the American Statistical Association*, 92(437):144–153, 1997.
7. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification And Regression Trees*. Chapman and Hall, 1984.
8. H.D. Brunk. Conditional expectation given a σ-lattice and applications. *Annals of Mathematical Statistics*, 36:1339–1350, 1965.
9. H.A.M. Daniels and B. Kamp. Application of MLP networks to bond rating and house pricing. *Neural Computing & Applications*, 8(3):226–234, 1999.
10. R. Dykstra, J. Hewett, and T. Robertson. Nonparametric, isotonic discriminant procedures. *Biometrika*, 86(2):429–438, 1999.
11. A. Feelders and M. Pardoel. Pruning for monotone classification trees. In M.R. Berthold, H-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, editors, *Advances in Intelligent Data Analysis V*, volume 2810 of *LNCS*, pages 1–12. Springer, 2003.
12. J. Karpf. Inductive modelling in law: example based expert systems in administrative law. In *Proceedings of the third international conference on artificial intelligence in law*, pages 297–306. ACM Press, 1991.
13. W. Kotlowski and R. Slowinski. Statistical approach to ordinal classification with monotonicity constraints. In *ECML PKDD 2008 Workshop on Preference Learning*, 2008.
14. S. Lievens, B. De Baets, and K. Cao-Van. A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Annals of Operations Research*, 163:115–142, 2008.
15. J. Long. NASA metrics data program [http://mdp.ivv.nasa.gov/repository.html]. 2008.
16. W.L. Maxwell and J.A. Muckstadt. Establishing consistent and realistic reorder intervals in production-distribution systems. *Operations Research*, 33(6):1316–1341, 1985.
17. M.J. Pazzani, S. Mani, and W.R. Shankle. Acceptance of rules generated by machine learning among medical experts. *Methods of Information in Medicine*, 40:380–385, 2001.
18. R. Potharst and J.C. Bioch. Decision trees for ordinal classification. *Intelligent Data Analysis*, 4(2):97–112, 2000.
19. Rob Potharst. *Classification using Decision Trees and Neural Nets*. PhD thesis, Erasmus University Rotterdam, 1999.
20. P. Royston. A useful monotonic non-linear model with applications in medicine and epidemiology. *Statistics in Medicine*, 19(15):2053–2066, 2000.
21. M. Velikova. *Monotone Models for Prediction in Data Mining*. PhD thesis, Tilburg University, 2006.
22. M. Velikova and H. Daniels. Decision trees for monotone price models. *Computational Management Science*, 1(3-4):231–244, 2004.