Two polynomial algorithms for relabeling non-monotone data

Ad Feelders

Marina Velikova

Hennie Daniels

Department of Information and Computing Sciences, Utrecht University

Technical Report UU-CS-2006-046

www.cs.uu.nl

ISSN: 0924-3275

Two polynomial algorithms for relabeling non-monotone data

Ad Feelders* Marina Velikova† Hennie Daniels†

October 6, 2006

Abstract

We describe and compare two algorithms for making data sets monotone with as few label changes as possible. These algorithms are useful for preparing data sets for the analysis with data mining algorithms that require monotone data. Also, the relabeled data set can be viewed as the monotone classifier with minimum error rate on the training data. The first algorithm is a greedy algorithm that runs in $O(LN^3)$ time, where L is the number of distinct class labels, and N is the number of observations, but it is not guaranteed to perform the minimum number of label changes required. The second algorithm is an exact algorithm based on a translation of the problem to finding a maximum weight independent set in the so-called monotonicity violation graph. This problem, in turn, is solved by solving a minimum flow problem in a transportation network, and runs in $O(N^3)$ time. Experiments on artificial and real data show that the greedy algorithm is typically very close to the optimal solution.

1 Introduction

In many prediction problems it is reasonable to assume a monotone relationship between the predictors and the response variable: an increase (decrease) in a predictor should (all else equal) lead to an increase (decrease) in the response. In house pricing, for instance, one would expect the price of a house to increase with the house area, volume and number of rooms. In practice, however, the observed data are often non-monotone due to noise, or the omission of important predictors. We present two algorithms for relabeling such datasets in order to make them monotone. The objective of both algorithms is to do this by changing as few labels as possible. Such an algorithm can be used for several purposes:

1. Some data mining algorithms for constructing monotone models require monotone data.

*Department of Information and Computing Sciences, Universiteit Utrecht, ad@cs.uu.nl
†Department of Information Systems and Management, Tilburg University

2. Some data mining algorithms for constructing monotone models work better with monotone data.

3. The relabeled data set may itself be viewed as a non-parametric monotone classifier that minimizes the error rate on the training data. Since this classifier is defined on the observed data points only, it would have to be extended with a rule to label other points from the input space.

The remainder of the paper is organized as follows. In the next section we introduce some notation and definitions, which are used in the discussion throughout the paper. In section 3 we discuss two polynomial algorithms for relabeling: the greedy algorithm introduced in (Velikova and Daniels 2004; Daniels and Velikova 2006) and an exact algorithm based on solving a minimum flow problem. We show that the the greedy algorithm is not guaranteed to produce a monotone data set with the minimum number of label changes. In section 3.2 we describe a transformation of the relabeling problem into a minimum flow problem in a transportation network. In section 4 we compare the algorithms on a number of artificial and real data sets. Section 5 concludes the paper.

2 Preliminaries

Let $\mathbf{X}$ denote the vector of predictors (attributes), which takes values $\mathbf{x}$ in a k-dimensional input space $\mathcal{X} = \times\mathcal{X}_i$, and let Y denote the class variable which takes values y in a one-dimensional space $\mathcal{Y}$. Let $D = \{(\mathbf{x}^i, y^i)\}_{i=1}^N$ denote the set of observed data points in $\mathcal{X} \times \mathcal{Y}$. We also use the alternative representation $U = \{(\mathbf{x}^i, y^i)\}_{i=1}^n$, of n *distinct* points in $\mathcal{X} \times \mathcal{Y}$ together with a vector of weights $w^i = n(\mathbf{x}^i, y^i), \;\; i = 1, \ldots, n$, where $n(\mathbf{x}^i, y^i)$ denotes the number of observations in D with $\mathbf{X} = \mathbf{x}^i$ and $Y = y^i$. Clearly, we have $N = \sum_{i=1}^n w^i$. Furthermore, we assume a partial order on $\mathcal{X}$ and a total order on $\mathcal{Y}$. Typically, the partial order on $\mathcal{X}$ is the product order induced by total orders on $\mathcal{X}_i$, that is

$$\mathbf{x} \leq \mathbf{x}' \Leftrightarrow x_i \leq x_i' \qquad \forall i = 1, \ldots, k,$$

but at no point do we require this to be the case.

A pair of points $(\mathbf{x}^i, y^i)$ and $(\mathbf{x}^j, y^j)$ from U (or D) is called non-monotone if

$$\mathbf{x}^i \leq \mathbf{x}^j \text{ and } y^i > y^j \tag{1}$$

We define the *monotonicity violation graph* to be the directed graph $G = (V, E)$, with $V = \{1, 2, \ldots, n\}$ and $(i, j) \in E$ if $\mathbf{x}^i \leq \mathbf{x}^j$ and $y^i > y^j$. The monotonicity violation graph is the graph of a strict partial order, since it is

1. Anti-symmetric: $(i, j) \in E \Rightarrow (j, i) \notin E$.

2. Transitive: $(i, j) \in E$ and $(j, k) \in E \Rightarrow (i, k) \in E$.

These properties follow immediately from the ordering on the class labels. We associate with each node $i \in V$ the weight w^i. Finally, we define the downset $\downarrow_{(i,S)}$ and the upset $\uparrow_{(i,S)}$ for any $S \subseteq V$ and $i \in V$:

$$\downarrow_{(i,S)} = \{j \in S | \mathbf{x}^j \leq \mathbf{x}^i\} \text{ and } \uparrow_{(i,S)} = \{j \in S | \mathbf{x}^i \leq \mathbf{x}^j\}.$$

3 Two polynomial relabeling algorithms

3.1 The greedy algorithm

In (Daniels and Velikova 2003), the authors present an algorithm for relabeling, which will be referred to as the *greedy algorithm* in the remainder of this paper. The idea is to reduce the number of non-monotone pairs by relabeling at least one data point in each step. To do this, a data point is chosen such that the increase in consistently labeled points is maximal.

The process is continued, until the data set is monotone. In the original version of the algorithm introduced in (Daniels and Velikova 2003), the authors assume that the data consist of unique points only, which may not be the case in practice (e.g., for discrete predictor variables). Here we drop this assumption and present a formal description of the generalized algorithm.

We call a point non-monotone if it violates the monotonicity constraint with at least one other point. Let $V = \{1, \ldots, N\}$, and let Q denote the set of indices of all non-monotone points in D. For each data point $i \in Q$ and for each $y \in \mathcal{Y}$, we define

$$\begin{aligned} A(i,y) &= \{j \in \downarrow_{(i,V)} | \mathbf{x}^j \neq \mathbf{x}^i \wedge y^j = y\} \\ B(i,y) &= \{j \in \uparrow_{(i,V)} | \mathbf{x}^j \neq \mathbf{x}^i \wedge y^j = y\} \\ C(i,y) &= \{j \in V \setminus \{i\} | \mathbf{x}^j = \mathbf{x}^i \wedge y^j = y\} \end{aligned}$$

Let a^i_y, b^i_y, and c^i_y denote the number of points in $A(i,y)$, $B(i,y)$, and $C(i,y)$, respectively. Furthermore, we define

- $y_{\min}, y_{\max}$ – the minimum and maximum value of the class labels,
- $N(\mathbf{x}^i, y^i)$ – total number of points consistently labeled with respect to $\mathbf{x}^i$ for the current label y^i, i.e.,

$$N(\mathbf{x}^i, y^i) = a^i_{y_{\min}} + \ldots + a^i_{y^i} + b^i_{y^i} + \ldots + b^i_{y_{\max}} + c^i_{y^i}.$$

For each $i \in Q$ we compute the maximal increase, I^i_{max}, in the number of consistently labeled points with respect to $\mathbf{x}^i$ if the label of $\mathbf{x}^i$ is changed into y as follows

$$I^i_{max} = \max_{y \in \mathcal{Y}} N(\mathbf{x}^i, y) - N(\mathbf{x}^i, y^i)$$

If there is more than one label with the same maximal increase in consistently labeled points, then we choose the label closest to the current label of $\mathbf{x}^i$. This choice is unique

as shown in Lemma 3 in (Daniels and Velikova 2003). Finally, we select a point $j \in Q$ for which I^j_{max} is the largest, and change its label and the labels of all the points $(\mathbf{x}^k, y^k) \in D$ that are identical to $(\mathbf{x}^j, y^j)$. If there is more than one point with the same maximal increase, the choice is made arbitrarily. This process is repeated until the data set is monotone. The correctness of the algorithm follows from Lemma 2 in (Daniels and Velikova 2003). The algorithm outline is given in Algorithm 1.

Algorithm 1 Greedy relabeling

Initialization: Compute Q on the basis of D
while $Q \neq \varnothing$ **do**
  for all $i \in Q$ **do**
    $I^i_{max} = \max_{y \in \mathcal{Y}} N(\mathbf{x}^i, y) - N(\mathbf{x}^i, y^i)$
    $\Lambda = \arg\max_{y \in \mathcal{Y}} N(\mathbf{x}^i, y) - N(\mathbf{x}^i, y^i)$
    Form a triple (i, I^i_{max}, y^i_*) where $y^i_* \in \Lambda$ is the closest label to y^i
  end for
  $r = \arg\max_{j \in Q} I^j_{max}$ (choose randomly if not unique)
  $R = \{k \in Q \mid (\mathbf{x}^k, y^k) = (\mathbf{x}^r, y^r)\}$
  for all $j \in R$ **do**
    $D(\mathbf{x}^j, y^j) \rightarrow D(\mathbf{x}^j, y^r_*)$
  end for
  Update Q on the basis of the modified data set D
end while

Next, we analyze the computational complexity of this algorithm. Let D denote a data set of N points and L distinct labels. At each iteration of the algorithm, we compute $Q(D')$ where D' denotes the modified data set after a number of label changes. Suppose there are p points in $Q(D')$. Then, for each step described in the algorithm, the effort is computed as follows:

$$\begin{aligned}
p\,\frac{N(N-1)}{2} &\quad \text{to compute } Q(D') \\
p\,L(N-1) &\quad \text{to compute } I_{max} \\
L &\quad \text{to compute } \Lambda \\
p\,L &\quad \text{to form the triples} \\
p &\quad \text{to find the triple with maximal } I_{max} \\
\frac{p(p-1)}{2} &\quad \text{to compute } R
\end{aligned}$$

Hence, the total effort, $C(p)$, for one iteration is

$$C(p) = p\,\frac{N(N-1)}{2} + p\,L(N-1) + L + p\,L + p + \frac{p(p-1)}{2}.$$

In the worst case when there are N non-monotone points in the data set and $Q(D)$ decreases by only one point at each step, the complexity is

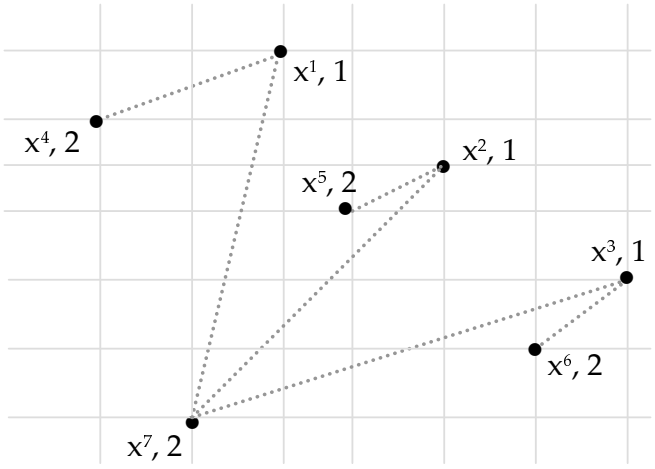

Figure 1: Data set of seven points in a two-dimensional input space

$$\sum_{p=2}^{N} C(p) = O(N^3 L).$$

This result shows that the greedy algorithm is polynomial in the number of points and labels in the data. However, the greedy algorithm does not guarantee a minimum number of label changes. To illustrate this, we consider the following example.

Figure 1 represents the structure of a data set of seven points with their labels, in a two-dimensional input space. Obviously, the data set is non-monotone; the dotted lines show the non-monotone relationships between the points. In order to make these data monotone, we apply the greedy algorithm for relabeling. First, we compute the maximal increase I^i_{max} in the number of consistently labeled points with respect to each point as follows.

$\mathbf{x}^1$,$\mathbf{x}^2, \mathbf{x}^3$	$\mathbf{x}^4$	$\mathbf{x}^5, \mathbf{x}^6$	$\mathbf{x}^7$
$N(\mathbf{x}^i, 1) = 0$	$N(\mathbf{x}^i, 1) = 1$	$N(\mathbf{x}^i, 1) = 1$	$N(\mathbf{x}^i, 1) = 5$
$N(\mathbf{x}^i, 2) = 2$	$N(\mathbf{x}^i, 2) = 0$	$N(\mathbf{x}^i, 2) = 1$	$N(\mathbf{x}^i, 2) = 2$
$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
$I^1_{max} = I^2_{max} = I^3_{max} = 2$	$I^4_{max} = 1$	$I^5_{max} = I^6_{max} = 0$	$I^7_{max} = 3$

The results show that the maximal increase in the number of consistently labeled points, $I_{max} = 3$, is obtained for $\mathbf{x}^7$; so, this will be the point chosen to be relabeled at the first step. In the next steps, the algorithm needs to relabel three other points to

make the data monotone. In other words, the algorithm will make four label changes in total. However, if we relabel only the three points $\mathbf{x}^1$, $\mathbf{x}^2$, and $\mathbf{x}^3$, we also obtain a monotone data set. This indicates that the greedy algorithm for relabeling could make a sub-optimal choice for the set of points to be relabeled.

3.2 An optimal flow network algorithm for relabeling

A subset of the vertices of a graph is an *independent set* if no two vertices in the subset are adjacent. The second algorithm is based on finding a maximum weight independent set in the monotonicity violation graph. As Rademaker, De Baets, and De Meyer (2006) observe, a maximum weight independent set in the monotonicity violation graph, corresponds to a maximum size monotone subset of the data. Relabeling the complement of the maximum weight independent set results in a monotone data set with as few label changes as possible; it is important to note that it is always possible to find a consistent relabeling of this complement. Although the monotonicity violation graph is defined to contain a node for all distinct points in D, it is obvious that points that do not violate the monotonicity constraint with any other point will always be part of the maximum weight independent set, and can be ignored in the sequel.

Although finding a maximum independent set in an arbitrary graph is known to be NP-hard (Garey and Johnson 1979), this is not the case for so-called *comparability graphs* (the graph of a partial order). For such graphs, a maximum independent set corresponds to a maximum antichain in the corresponding partial order, and can be computed in $O(N^3)$ time by solving a minimum flow problem on a transportation network that is easily constructed from the comparability graph (Möhring 1985). Since we have already shown that the monotonicity violation graph is the graph of a partial order, we have a polynomial time exact algorithm to compute a maximum weight independent set of G.

Möhring (1985) shows that the maximum weight independent set of a comparability graph equals the minimum flow value in a transportation network that can be obtained by a simple transformation of the graph. Hence, our problem is finally reduced to finding this value. We next describe the transformation of the monotonicity violation graph $G = (V, E)$ to the corresponding transportation network $G' = (V', E')$. Because the monotonicity violation graph has weights associated with the vertices rather than the edges (as is assumed by standard network flow algorithms), we transform vertices to edges, by so-called vertex splitting:

$$V' = \bigcup_{i \in V} \{i_1, i_2\} \cup \{s, t\},$$

where s is the source, and t is the sink of the transportation network. The edge set E' contains edges $i_1 i_2$ for all $i \in V$, and edges $i_2 j_1$ for all $ij \in E$. Furthermore, E' contains edges $s i_1$ for all minimal points $\mathbf{x}^i$, and edges $j_2 t$ for all maximal points $\mathbf{x}^j$. The edges $i_1 i_2 \in E'$ are assigned lower capacities w^i and upper capacities $+\infty$. All remaining edges of E' are assigned lower capacities of zero and upper capacities of $+\infty$.

To illustrate this transportation network construction, we consider the data depicted in Figure 1. The set of lines E connecting the points indicate the monotonic-

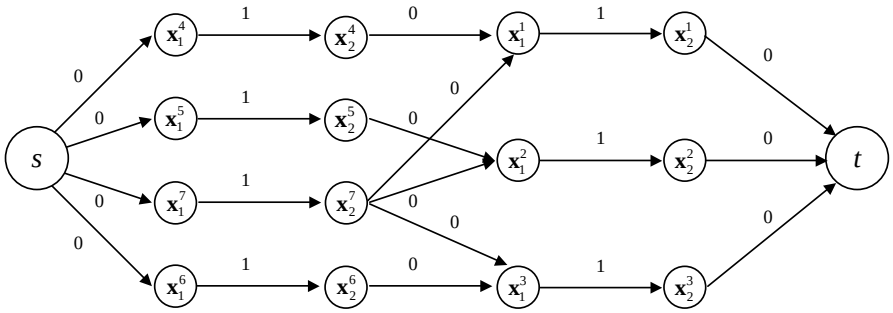

Figure 2: Transportation network based on the non-monotone data in Figure 1

ity violations; the corresponding monotonicity violation graph is $G = (V, E)$ with $V = \{1, 2, 3, 4, 5, 6, 7\}$, and $E = \{(4, 1), (7, 1), (5, 2), (7, 2), (7, 3), (6, 3)\}$. Figure 2 depicts the transportation network associated with G. Each of the seven points is represented by an edge with a lower capacity of one (all data points happen to be unique in this example), and a upper capacity of $+\infty$. The connections to the source and the sink, and the edges representing the monotonicity violations are assigned lower capacities of zero and upper capacities of $+\infty$.

The problem of finding the maximum weight independent set in G can now be solved by finding the minimum flow value $\mathtt{f}_{val}^{min}$ in G'. Furthermore, by the min-flow max-cut theorem (Ford and Fulkerson 1962; Lawler 1976) $\mathtt{f}_{val}^{min}$ equals the maximum capacity of an s, t-cut (or maximum cut) in G', that is,

$$\mathtt{f}_{val}^{min} = \max_{S,T} \left[\sum_{\substack{ab \in E' \\ a \in S, b \in T}} lc(ab) - \sum_{\substack{ab \in E' \\ a \in T, b \in S}} uc(ab) \right],$$

where S, T is an s, t-cut of $G' = (V', E')$, i.e., $V' = S \cup T$, $S \cap T = \varnothing$, $s \in S$, $t \in T$, and where $lc(ab)$ and $uc(ab)$ denote the lower and upper capacity of edge $ab \in E'$.

As Möhring (1985) remarks, $\mathtt{f}_{val}^{min}$ must obviously be positive, and therefore an optimal cut S, T contains no edges $ab \in E'$ with $a \in T$ and $b \in S$, since $uc(ab) = +\infty$ for each such edge. This implies that the set of vertices $A = \{i \in V \mid i_1 i_2 \in E', i_1 \in S, i_2 \in T\}$ corresponding to an optimal cut, is an antichain of G. To see this, suppose that A is not an antichain, that is, it contains comparable points i and j. Then, by the definition of A, we have $i_1, j_1 \in S$ and $i_2, j_2 \in T$. Since i and j are comparable, we have either $i_2 j_1 \in E'$ or $j_2 i_1 \in E'$ which means we would have an edge from T to S in E'. But this contradicts our observation that $\mathtt{f}_{val}^{min}$ must be positive.

Furthermore, since each antichain A in G induces an S, T cut in G' by putting

$$S = \{v \in V' \mid \text{there is a directed path in } G' \text{ from } v \text{ to } i_2 \text{ for some } i \in A\}$$

we have that the minimum flow value in G' equals the maximum weight of an antichain in G (Möhring 1985).

Finally, the set complement to the maximum weight independent set is the set of points that need to be relabeled to get monotone data. For the network flow in Figure 2, we find $\mathtt{f}_{val}^{min} = 4$, i.e., the weight of the maximum weight independent set M is 4. This set is obtained by the S,T-cut, where $S = \{s, 4_1, 5_1, 6_1, 7_1\}$, $T = V' \backslash S$, and $M = \{4, 5, 6, 7\}$. Hence, we find that the set of points that need to be relabeled to make D monotone is $V \backslash M = \{1, 2, 3\}$. The relabeling algorithm can now be described as follows:

1. Find a maximum weight independent set $M \subseteq V$ of G.
2. For all $i \in M$, set $y_i^* = y_i$.
3. Set $R = V \setminus M$
4. For each $j \in R$
   (a) Let
   $$\underline{y}^* = \max\{y_i^* | i \in \downarrow_{(j,M)}\} \text{ and } \bar{y}^* = \min\{y_i^* | i \in \uparrow_{(j,M)}\}$$
   Pick $y_j^* \in [\underline{y}^*, \bar{y}^*]$ and set $M = M \cup \{j\}$, and $R = R \setminus \{j\}$.
   Replace $(\mathbf{x}_j, y_j)$ by $(\mathbf{x}_j, y_j^*)$ in U.
   (b) Stop when R is empty.

4 Experiments

First we conduct experiments with artificial data to compare the results from the application of the greedy algorithm for relabeling and the minimum flow algorithm. To obtain more sound assessment of the results, we generate a number of data sets by combining the following characteristics:

- Number of points: 100 or 500,
- Number of attributes: 2 or 6,
- Number of class categories: 2 or 6,
- Noise level: up to 10%, 30%, or 60%.

Thus we obtain 24 different data sets in total. The attributes (independent variables) in all data sets are drawn from the uniform distribution on [0,1]. The dependent variable is generated by applying monotone functions on all independent variables as follows.

- For data sets with 2 independent variables, we use
$$z = x_1 \sin\left(\frac{\pi}{2} x_2\right)$$

- For data sets with 6 independent variables, we use

$$z = x_1 + x_2 + x_3 + \sin\left(\frac{\pi}{2} x_4 x_5 x_6\right)$$

Next we discretize the continuous dependent variable into a finite number of classes. To do so, we split the z-values into a number of intervals corresponding to the required number N_{cl} (2 or 6) of classes. Each interval i, $i = 1, 2, \ldots N_{cl}$, is defined by $\big[lb(i), lb(i+1)\big)$; $lb(i)$ is the lower bound of the interval computed by

$$lb(i) = \frac{i-1}{N_{cl}} \cdot z_{max},$$

where z_{max} is the maximum z-value.

In the next step, we turn the monotone data set into a non-monotone set by adding random noise to the discrete labels. This is done by changing randomly the labels with certain probabilities; for example, label $y = 2$ does not change with a probability of 90% and change to either $y = 1$ or $y = 3$ with a probability of 5%.

On the non-monotone data thus generated, we apply both the greedy algorithm for relabeling and the minimum flow algorithm to make the data monotone. The results from the experiments with all data sets are reported in Table 1.

The results show that for 18 out of 24 data sets both algorithms make the data monotone with the same number of label changes. We observe differences in this number only for data sets with a high percentage of noise level (see the results in bold in Table 1). For five data sets, the greedy algorithm for relabeling tends to make only from 1 to 3 additional label changes. Exceptionally, in the one case where nearly half of the data points are relabeled, we have a difference of 10 label changes between the algorithms. Based on these results we conjecture that the greedy algorithm for relabeling makes a number of label changes that is close to the minimum.

Next we assess the performance of the both algorithms on real data sets, which represent monotone classification problems. The characteristics of the data sets are given in Table 2. The bond rating data has been used in previous studies (Daniels and Kamp 1999; Daniels and Velikova 2006), whereas the other four data sets are publicly available from the UCI data repository (Newman, Hettich, Blake, and Merz 1998). The original AutoMpg data set represents a monotone regression problem, and to transform it into a classification problem we discretized the dependent variable into two classes: a car has *low* (≤ 28) or *high* (> 28) mileage per gallon (mpg).

Next, for all data sets we check the direction of influence of each attribute with respect to the class variable. The correlation coefficients reveal that for the Pima Indian data all the attributes have a positive influence, whereas for the other data sets we observe both attributes with positive and negative influences. Since we consider increasing monotonicity in this study, we synchronize the direction of influence to be positive of all the attributes with respect to the class variable. To do so, we perform the following linear transformation on each attribute x with a negative correlation coefficient:

$$x^i = x_{max} - x^i + x_{min}, \quad \text{for } i = 1, \ldots, N,$$

Table 1: Results from the experiments with the greedy algorithm for relabeling and the minimum flow algorithm applied on artificial data sets

# points	# attributes	# class categories	Noise level (up to)	Number of label changes	
				Greedy algorithm for relabeling	Minimum flow algorithm
100	2	2	10 %	3	3
			30 %	24	24
			60 %	40	40
		6	10 %	7	7
			30 %	19	19
			60 %	**43**	**41**
	6	2	10 %	1	1
			30 %	7	7
			60 %	12	12
		6	10 %	3	3
			30 %	7	7
			60 %	**14**	**13**
500	2	2	10 %	31	31
			30 %	113	113
			60 %	**214**	**212**
		6	10 %	39	39
			30 %	**134**	**131**
			60 %	**247**	**237**
	6	2	10 %	5	5
			30 %	42	42
			60 %	**94**	**93**
		6	10 %	8	8
			30 %	31	31
			60 %	68	68

where N is the number of data points in a data set, and x_{max}, x_{min} are the maximum and minimum values of x in the data, respectively. This transformation has been applied to the following attributes:

- Bond rating: CF/D, CF, and COV;
- AutoMpg: Cylinders, Displacement, Horsepower, and Weight;
- Breast Cancer Ljubljana: Menopause, Tumor-size, Inv-nodes, Deg-malig, and Irradiat;
- Haberman's survival: Year of operation.

Table 2: Characteristics of real data sets with their degree of monotonicity

Data set	Number of points	Number class categ.	Number of attributes	Number of comparable pairs	Degree of monotonicity
AutoMpg	392	2	7	30 727	99.9 %
Breast Cancer L.	277	2	9	7 586	93.5 %
Bond rating	256	7	5	9 685	98.9%
Hab	306	2	3	14 575	87.7 %
Pima	768	2	8	21 553	97.7 %

Table 3: Results from the experiments with the greedy algorithm for relabeling and the minimum flow algorithm applied on real data sets

Data set	Number of label changes	
	Greedy algorithm for relabeling	Minimum flow algorithm
AutoMpg	7	7
Breast Cancer L.	42	41
Bond rating	28	28
Hab	56	55
Pima	54	53

Furthermore, for each data set the measure for the degree of monotonicity (DgrMon) of the data is computed as the fraction of monotone pairs of all comparable pairs in the data. Although the problems they are derived from are monotone, it is clear that all data sets contain inconsistencies. Therefore we apply the greedy algorithm for relabeling and the minimum flow algorithm to resolve them. The number of label changes made by both algorithms are presented in Table 3. The results clearly indicate that both algorithms make the same or very close number of label changes–the difference in the number is one point only.

5 Conclusions

We have discussed two polynomial algorithms for turning non-monotone into monotone data by changing the labels of as few points as possible. The greedy algorithm is based on previous studies and works by reducing the number of non-monotone pairs by relabeling only one data point in each step. Although the algorithm does not guarantee a solution with a minimum number of label changes, it makes the data monotone in a small number of steps. The second algorithm is based on finding a maximum weight independent set in the monotonicity violation graph. This can be done in polynomial time, because the monotonicity violation graph is the graph of a partial order. Relabeling the complement of the maximum weight independent set gives a monotone data set with a minimum number of label changes. Experiments on artificial and real data show that the number of points relabeled by the greedy algorithm in general is very close to the minimum. On the real data sets, the difference was shown to be at most one point.

References

Daniels, H. and B. Kamp (1999). Application of MLP networks to bond rating and house pricing. *Neural Computing & Applications 8*(3), 226–234.

Daniels, H. and M. Velikova (2003). Derivation of monotone decision models from non-monotone data. Center Internal Report 2003–30, Tilburg University.

Daniels, H. and M. Velikova (2006). Derivation of monotone decision models from noisy data. *To appear in IEEE Transactions on Systems, Man and Cybernetics, Part C*.

Ford, L. R. and D. R. Fulkerson (1962). *Flows in networks*. Princeton, NJ: Princeton University Press.

Garey, M. and D. Johnson (1979). *Computers and Intractability: a Guide to the Theory of NP-Completeness*. New York: Freeman.

Lawler, E. L. (1976). *Combinatorial optimization: networks and matroids*. New York: Holt, Rinehart and Winston.

Möhring, R. H. (1985). Algorithmic aspects of comparability graphs and interval graphs. In *Graphs and Order*, pp. 41–101. Dordrecht: Reidel.

Newman, D. J., S. Hettich, C. L. Blake, and C. J. Merz (1998). UCI Repository of Machine Learning Databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Rademaker, M., B. De Baets, and H. De Meyer (2006). Data sets for supervised ranking: to clean or not to clean. In *Proceedings of the fifteenth Annual Machine Learning Conference of Belgium and The Netherlands: BENELEARN 2006, Ghent, Belgium*, pp. 139–146.

Velikova, M. and H. Daniels (2004). Decision trees for monotone price models. *Computational Management Science 1*(3–4), 231–244.