Nearest Neighbour Classification with Monotonicity Constraints

Wouter Duivesteijn and Ad Feelders

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

Abstract. In many application areas of machine learning, prior knowledge concerning the monotonicity of relations between the response variable and predictor variables is readily available. Monotonicity may also be an important model requirement with a view toward explaining and justifying decisions, such as acceptance/rejection decisions. We propose a modified nearest neighbour algorithm for the construction of monotone classifiers from data. We start by making the training data monotone with as few label changes as possible. The relabeled data set can be viewed as a monotone classifier that has the lowest possible error-rate on the training data. The relabeled data is subsequently used as the training sample by a modified nearest neighbour algorithm. This modified nearest neighbour rule produces predictions that are guaranteed to satisfy the monotonicity constraints. Hence, it is much more likely to be accepted by the intended users. Our experiments show that monotone kNN often outperforms standard kNN in problems where the monotonicity constraints are applicable.

1 Introduction

Monotonicity of relations between a response variable and predictor variables is a form of prior knowledge that is available in many application areas of machine learning. For example, in house pricing, the price of a house typically increases with the lot size, and decreases with the distance to the city center. Other examples of monotonicity constraints can be found in medicine [8,25], finance [16], and law [18].

Monotonicity may also be an important model requirement with a view toward explaining and justifying decisions, such as acceptance/rejection decisions. Pazzani et al.[21], report on an application of rule induction algorithms to early detection of dementia, and prediction of mild mental retardation. They show that the rules learned with monotonicity constraints were significantly more acceptable to medical experts than rules learned without the monotonicity restrictions.

While human experts tend to feel uncomfortable expressing their knowledge and experience in terms of numeric assessments, they typically are able to state their knowledge in a semi-numerical or qualitative form with relative conviction and clarity, and with less cognitive effort [10]. Experts, for example, can often easily indicate which of two probabilities is smallest. In addition to requiring less cognitive effort, such relative judgements tend to be more reliable than direct numerical assessments [19].

Hence, monotonicity constraints occur frequently in machine learning problems and such constraints can be elicited from subject area experts with relative ease and reliability. This has motivated the development of learning algorithms that are able to enforce such constraints in a justified manner. Several machine learning techniques have been adapted to be able to handle monotonicity constraints in one form or another. Examples are: classification trees [7,12,22], neural networks [3,26], and Bayesian networks [1,13].

In this paper we present an algorithm for nonparametric monotone classification. Our approach consists of two steps. In the first step, the training data is made monotone by relabeling as few cases as possible. This relabeled data set may be viewed as the monotone classifier with the smallest error rate on the training data. In the second step, we use a modified nearest neighbour rule to predict the class labels of new data in such a way that the monotonicity constraints are satisfied.

The paper is organized as follows. In the next section, we establish some notation and definitions that will be used throughout the paper. In section 3, we discuss the problem of relabeling a non-monotone data set, and give an algorithm to make it monotone with as few label changes as possible. Subsequently, we present in section 4 a monotone variant of the k-nearest neighbour rule to predict the class labels of new data points. Related work on nonparametric monotone classification is discussed in section 5. In section 6 we present the results of experiments in which we compare the monotone nearest neighbour rule with standard nearest neigbour prediction. Finally, we draw conclusions in section 7.

2 Notation and Preliminaries

Let $\mathbf{X}$ denote the vector of predictors (attributes), which takes values $\mathbf{x}$ in a p-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 $w_i = n(\mathbf{x}_i, y_i), \;\; i = 1, \dots, n$, 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 $\mathcal{Y} = \{1, 2, \dots, c\}$, where c is the number of class labels. Typically, the partial order on $\mathcal{X}$ is the product order induced by total orders on $\mathcal{X}_i$, that is

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

but at no point do we require this to be the case. The objective is to learn from data an allocation rule $f : \mathcal{X} \to \mathcal{Y}$ such that $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$:

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow f(\mathbf{x}) \leq f(\mathbf{x}'), \tag{1}$$

that is, a lower ordered input is not allowed to have a higher class label. A pair of points $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$ from U (or D) is called non-monotone if

$$\mathbf{x}_i \leq \mathbf{x}_j \text{ and } y_i > y_j \tag{2}$$

We define the *monotonicity violation graph* (MVG) 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 $y_i > y_j$. We note that 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 order 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 Relabeling Non-monotone Data

The first step in our approach is to relabel the training data in order to remove all monotonicity violations, using as few label changes as possible. The relabeled data set can be viewed as a monotone classifier that minimizes the error rate on the training data.

A subset of the vertices of a graph is an *independent set* if no two vertices in the subset are adjacent. As Rademaker et al. [23] 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 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. Although finding a maximum independent set in an arbitrary graph is known to be NP-hard [17], we make use of the fact that this is not the case for *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, see [20,14]. As we noted, the monotonicity violation graph is a comparability graph, so we have an $O(n^3)$ algorithm that minimizes

$$\sum_{i=1}^{N} I(y_i \neq f(\mathbf{x}_i)),$$

subject to

$$\mathbf{x}_i \leq \mathbf{x}_j \Rightarrow f(\mathbf{x}_i) \leq f(\mathbf{x}_j), \tag{3}$$

for an arbitrary partial order on $\mathcal{X}$, and for an arbitrary number of linearly ordered class labels.

We next describe the transformation of the monotonicity violation graph $G = (V, E)$ to the corresponding transportation network $G' = (V', E')$. Let V^- denote the set of nodes in V with non-zero degree. 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:

$$V' = \bigcup_{i \in V^-} \{i_a, i_b\} \cup \{s, t\},$$

where s is the source, and t is the sink of the transportation network. The edge set E' contains edges (i_a, i_b) for all $i \in V^-$, and edges (i_b, j_a) for all $(i, j) \in E$. Furthermore, E' contains edges (s, i_a) for all minimal points $\mathbf{x}_i$, and edges (j_b, t) for all maximal points $\mathbf{x}_j$. The edges $(i_a, i_b) \in E'$ 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 $+\infty$. 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 [15], $\mathtt{f}_{val}^{min}$ equals the maximum capacity of an s, t-cut (or maximum cut) in G', that is,

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

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

Obviously, $\mathtt{f}_{val}^{min}$ must be positive, and therefore an optimal cut S, T contains no edges $(v, w) \in E'$ with $v \in T$ and $w \in S$, since $uc(v, w) = +\infty$ for each such edge. This implies that the set of vertices

$$A = \{i \in V \mid (i_a, i_b) \in E', i_a \in S, i_b \in T\} \tag{4}$$

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_a and $j_a \in S$, and i_b and $j_b \in T$. Since i and j are comparable, we have either $(i_b, j_a) \in E'$ or $(j_b, i_a) \in E'$ 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

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

we have that the minimum flow value in G' equals the maximum weight of an antichain in G [20].

Although in the worst case, the run time of the algorithm is cubic in the number of distinct observations, it may be quite fast in practice because of two reasons. First, all points that are not involved in any monotonicity violations can be disregarded, because they will never be relabeled. Since they are not connected to any other point in the MVG, they will belong to every maximum independent set. Hence the restriction to nodes in V^- in the transportation network. If the data generating process is indeed monotonic, and any monotonicity violation is caused by noise, then it is reasonable to assume that most points are not involved in a monotonicity violation. Secondly, we can apply a divide-and-conquer strategy by finding a maximum independent set for each

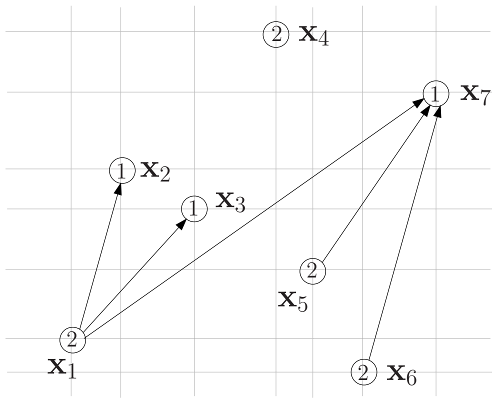

Fig. 1. Example Monotonicity Violation Graph

connected component of the MVG separately. The union of these sets will then be a maximum independent set for the complete graph.

The relabeling algorithm is summarized in Algorithm 1. It takes the training set and its monotonicity violation graph as inputs, and returns the relabeled training set. In line 6 the actual relabeling takes place. The function *select* picks a value from the interval of allowed class labels $[y_{\min}, y_{\max}]$; this interval always contains at least one element, since the set of points with index in M is always consistent. Which element it picks is arbitrary from the viewpoint of error-rate minimization.

Algorithm 1. relabel($U, G = (V, E)$)

```
1: $M \leftarrow$ maximum independent set($G$)
2: $R \leftarrow V \setminus M$
3: for all $j \in R$ do
4:     $y_{\min} \leftarrow \max\{y_i | i \in \downarrow_{(j,M)}\}$
5:     $y_{\max} \leftarrow \min\{y_i | i \in \uparrow_{(j,M)}\}$
6:     $y_j \leftarrow$ select($y_{\min}$,$y_{\max}$)
7:     $M \leftarrow M \cup \{j\}$
8: end for
9: return $U$
```

As an example, consider the data set with monotonicity violation graph depicted in Figure 1. Here $\mathbf{x}_1, \ldots, \mathbf{x}_7$ are plotted as points in the plane, and their observed class labels are given inside the points. If both x_1 and x_2 are known to have a positive influence on y, then the appropriate ordering on the input points is given by

$$\mathbf{x} \leq \mathbf{x}' \Leftrightarrow x_1 \leq x_1' \wedge x_2 \leq x_2'$$

So, for example, we have $\mathbf{x}_1 \leq \mathbf{x}_2$, but $\mathbf{x}_2$ and $\mathbf{x}_3$ are incomparable points. The corresponding monotonicity violation graph is $G = (V, E)$ with $V = \{1, 2, 3, 4, 5, 6, 7\}$,

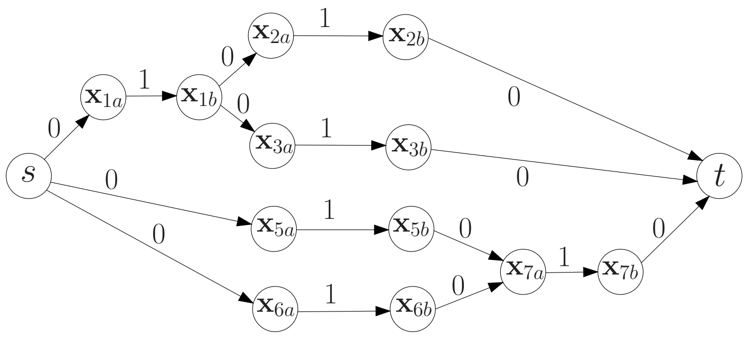

Fig. 2. Transportation network based on the Monotonicity Violation Graph in Figure 1

$E = \{(1,2),(1,3),(1,7),(5,7),(6,7)\}$, and $V^- = V\backslash\{4\}$. Figure 2 depicts the transportation network associated with G. Each of the points in V^- is represented by an edge with a lower capacity of one (all data points happen to be unique in this example), and an 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 $+\infty$. For the network flow in Figure 2, we find $\mathtt{f}_{val}^{min} = 4$, that is, the weight of the maximum weight antichain A is 4. This set is obtained by the S,T-cut, where $S = \{s, \mathbf{x}_{1a}, \mathbf{x}_{1b}, \mathbf{x}_{2a}, \mathbf{x}_{3a}, \mathbf{x}_{5a}, \mathbf{x}_{6a}\}$, $T = V'\backslash S$. Using equation (4), we find $A = \{2,3,5,6\}$. Adding $\mathbf{x}_4$ gives $M = \{2,3,4,5,6\}$. Finally, the set complement to the maximum weight independent set is the set of points that need to be relabeled to get monotone data. Hence, we find that the set of points that need to be relabeled to make D monotone is $R = V\backslash M = \{1,7\}$.

Because the class label is binary, there is only one alternative label for each point, so relabeling is automatic. For binary classification problems, we can also accomodate different misclassification (relabeling) costs. Let $C(j,k)$ denote the cost of relabeling an example from class j to class k. Define weights

$$w_i = \begin{cases} n(\mathbf{x}_i, y_i)C(1,2) \text{ if } & y_i = 1 \\ n(\mathbf{x}_i, y_i)C(2,1) \text{ if } & y_i = 2 \end{cases}$$

We now obtain a minimum cost relabeling by finding a maximum weight independent set in the MVG, and relabeling its complement. To illustrate, consider the case where $C(1,2) = 1$ and $C(2,1) = 3$. Hence, all points in Figure 1 with class label 1 receive a weight of 1, and all points with class label 2 receive a weight of 3. The reader can verify that the maximum weight independent set is $M = \{1,4,5,6\}$ and hence $R = \{2,3,7\}$: it has become cheaper to relabel both 2 and 3, instead of relabeling 1. Unfortunately, this straightforward approach can not be extended to the non-binary case, because we then have more than one relabeling option, and we can therefore not associate a unique weight with each node.

The relabeled data set can be viewed as a monotone classifier that minimizes the error-rate on the training data. This classifier is however only defined on the observed data points. In case we have just a few discrete input variables, these might cover the entire input space, but in general this will not be the case. Hence, the classifier has to be extended to the entire input space in such a way that the monotonicity constraints are satisfied, and the information contained in the observed data points is used to its full extent to classify new cases. This problem is discussed in the next section.

4 Monotone kNN

In order to satisfy the monotonicity restrictions, it is clear that the class label assigned to a new data point $\mathbf{x}_0$ is constrained to lie in the the interval $[y_{\min}, y_{\max}]$, where

$$y_{\min} = \max\{y|(\mathbf{x}, y) \in D \wedge \mathbf{x} \leq \mathbf{x}_0\},$$

and

$$y_{\max} = \min\{y|(\mathbf{x}, y) \in D \wedge \mathbf{x}_0 \leq \mathbf{x}\},$$

where D is the relabeled data set. The choice of a value from this interval is however free, and hence it makes sense to make further use of the observed data to guide this choice. We consider two variants on the standard nearest neighbour rule:

1. Take the k nearest neighbours of $\mathbf{x}_0$ from D and predict the label from $[y_{\min}, y_{\max}]$ that occurs most often among these k points. If none of the k labels are allowed, choose at random from $[y_{\min}, y_{\max}]$.
2. Take the k nearest neighbours of $\mathbf{x}_0$ from D with label in $[y_{\min}, y_{\max}]$ and predict the label by majority voting.

Variant 1 uses at most k neighbours in the majority voting, variant 2 always uses exactly k neighbours. The two variants are equivalent if the class label is binary, since in that case there is a choice of label only when both labels are allowed, but then both variants are the same as the standard nearest neighbour rule.

If we want predictions to be consistent among themselves as well (and not just with the training sample), then we should store the points with their predicted class labels to be used in subsequent predictions. It is clear that in this case, the order of arrival of points to be predicted makes a difference.

To visually illustrate the difference between standard nearest neighbour and monotone nearest neighbour, we consider a small example. Suppose the training data consists of the three points plotted in Figure 3. Next to each data point, its (x_1, x_2) coordinates and class label are given. The figure also gives the partitioning of the input space according to the 1-nearest neighbour rule, the so-called Voronoi diagram. It is clear that the resulting allocation rule is not monotone. In Figure 4 we have given the allocation rule for the *next prediction* of the monotone 1-nearest neighbour rule. Since all points smaller than (4,8) can not get a class label bigger than 1, the allocation rule has been adjusted accordingly. Note that this allocation rule is not monotone in general, but it is monotone with respect to the three points in the training sample. If predictions do not have to be monotone among themselves, then Figure 4 gives the monotone 1-nearest neighbour allocation rule. Otherwise, it may have to be updated after each prediction.

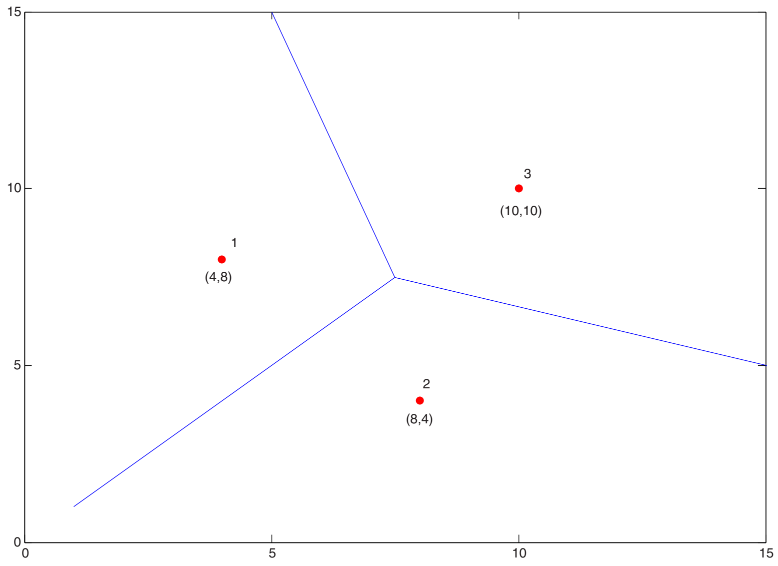

Fig. 3. Allocation rule of 1 nearest neighbour. For each data point its (x_1, x_2) coordinates and class label are given.

5 Related Work

As mentioned in the introduction, several machine learning methods have been adapted to incorporate monotonicity constraints. In this section, we restrict our attention to work that is relevant specifically to the *nonparametric* monotone classification problem that we are considering in this paper.

The earliest work in this area known to us is the Ordinal Learning Model (OLM) of Ben-David [5,6]. They construct a so-called *rule base* from a set of training examples. The rule-base R is a subset of the training examples, and is composed of *consistent* and *irredundant* examples. Consistency here refers to the monotonicity requirement. The algorithm sequentially adds examples from the training set to the rule-base, but if an example violates the monotonicity restriction with one or more examples already present in R, then it is discarded. Due to the prediction rule of OLM, examples may also be redundant with respect to the current rule base. OLM allocates a new case $\mathbf{x}_0$ to the largest class among the points in R that precede it:

$$f_{\text{OLM}}(\mathbf{x}_0) = \max\{y \mid (\mathbf{x}, y) \in R \wedge \mathbf{x} \leq \mathbf{x}_0\} \tag{5}$$

As a consequence, if $(\mathbf{x}, y) \in R$ then any $(\mathbf{x}', y)$ with $\mathbf{x} \leq \mathbf{x}'$ does not affect the labeling of new instances. Hence $(\mathbf{x}', y)$ is redundant with respect to $(\mathbf{x}, y)$. If there is no $(\mathbf{x}, y) \in R$ with $\mathbf{x} \leq \mathbf{x}_0$, then $\mathbf{x}_0$ is allocated to the class of the point in R nearest to it, that is, according to the 1-nearest neighbor rule. We note that the composition of the final rule-base critically depends on the order in which the examples are processed. In particular, when an unfortunate choice is made for the first example, then many of

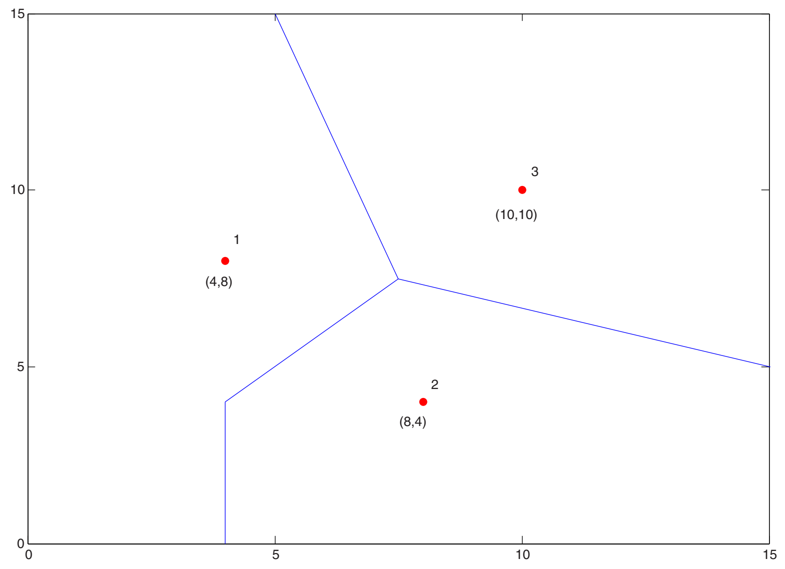

Fig. 4. Allocation rule of monotone 1-nearest neighbour. For each data point its (x_1, x_2) coordinates and class label are given.

the training samples will have to be discarded. Furthermore, as pointed out in [9], non-monotone prediction results are possible due to the 1-nearest neighbor rule used in case no points smaller than $\mathbf{x}_0$ are contained in the rule-base. The most important difference between OLM's prediction rule and ours, is that OLM does not make use of the class labels of nearby points in making its predictions: as shown in equation (5) it simply takes the maximum label of all points that are smaller than the point to be predicted.

Cao-Van [9] presents an algorithm called Ordinal Stochastic Dominance Learner (OSDL), which learns a collection of probability distributions over the class variable, under the restriction that

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow \sum_{j=1}^{i} \Pr(y = j \mid \mathbf{x}) \geq \sum_{j=1}^{i} \Pr(y = j \mid \mathbf{x}'), \tag{6}$$

for $i = 1, 2, \ldots, k-1$. In words, if $\mathbf{x}$ precedes $\mathbf{x}'$ in the ordering, then the distribution of Y in $\mathbf{x}'$ must be stochastically larger than the distribution of Y in $\mathbf{x}$. Although the interpretation of the monotonicity constraint in terms of stochastic dominance is a useful one for probabilistic classifiers, an allocation rule that assigns an input point to the mode of this distribution will not in general be monotone, unless the class variable is binary. In case an outright assignment to a class is required, OSDL therefore takes the median class value according to $\widehat{\Pr}(Y|\mathbf{x})$. This could be interpreted as an attempt to minimize L_1 loss, although this is not stated explicitely. The conditional probabilities $\Pr(Y|\mathbf{x})$ are estimated in a nonparametric way; for details we refer to [9].

Dykstra et al.[11] propose a nonparametric monotonic classification procedure that minimizes L_1 loss

$$\sum_{i=1}^{N} |y_i - f(\mathbf{x}_i)|,$$

subject to

$$\mathbf{x}_i \leq \mathbf{x}_j \Rightarrow f(\mathbf{x}_i) \leq f(\mathbf{x}_j),$$

where the class labels are numbered $\{1, 2, \ldots, c\}$. Their algorithm requires the performance of $c - 1$ isotonic regressions [24] to find an optimal solution. They also provide an algorithm that minimizes L_2 loss that requires the performance of a single isotonic regression. Note that in the important special case of binary classification, minimizing either of these loss functions results in minimal 0/1 loss as well. This is however not the case if there are more than two class labels. The use of squared error loss or absolute error loss presupposes more than an ordering of the class values. Even though these values may be numbered $1, 2, \ldots, c$ for convenience, this does not imply that performance of numerical operations on them is meaningful. On the other hand, it does make sense to presume that classifying a class 1 observation as class 5, is worse than classifying it as class 2.

Dykstra et al.[11] indicate possibilities to extend the relabeled training data to a monotone prediction rule for the entire input space, but like OLM without using any information in the training data beyond the ordering of data points.

6 Experiments

In order to test the proposed classification algorithm, we conducted a number of experiments. In all these experiments, we compared the performance of monotone kNN with that of standard kNN, in order to make sure we are not obtaining monotone models at the expense of predictive accuracy. If a monotone model is really required, then a small increase of the error might be acceptable, but clearly this should be within reasonable limits. On the other hand, if the problem really is monotone, then we might even expect an improvement of the accuracy.

We selected a number of data sets for which the presence of an increasing (or decreasing) relation between the attributes and the response variable was a priori plausible. Table 1 gives an overview of the data sets we used. All data sets have been taken from the UCI machine learning repository [4], except for *Windsor Housing*[1] [2], and *Employee Selection* [2] [6].

As an example, in Table 2 we give the signs of the relations between the attributes and the response that we use for the *AutoMpg* data set.

For the Australian credit approval data, we only used columns 7, 8, 9 and 10 of the attributes from the original data set. For the Boston housing data, we excluded the

[1] Available from the Journal of Applied Econometrics Data Archive at `http://econ.queensu.ca/jae/`

[2] Available at `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`

Table 1. Data sets used in the experiments. The number of attributes given is *after* preprocessing. The column labeled *Comparability* gives the fraction of all pairs of data points $\mathbf{x}, \mathbf{x}'$ for which $\mathbf{x} \leq \mathbf{x}'$, or $\mathbf{x}' \leq \mathbf{x}$.

Data set	# points	# attributes	Target	Comparability
Australian credit approval	690	4	Binary	0.7162
AutoMpg	392	7	Numeric	0.4009
Boston housing	506	12	Numeric	0.1910
Employee Selection	488	4	9 Classes	0.7065
Haberman survival	306	3	Binary	0.3123
Machine (cpu performance)	209	6	Numeric	0.4950
Pima indians diabetes	768	8	Binary	0.0732
Windsor housing	546	11	Numeric	0.2737
Wisconsin breastcancer	683	9	Binary	0.2710

Table 2. Signs used between the target *Miles per gallon* and the different attributes in the Auto-Mpg data set

Attribute	Type	Sign
mpg	continuous	target
cylinders	multi-valued discrete	$-$
displacement	continuous	$-$
horsepower	continuous	$-$
weight	continuous	$-$
acceleration	continuous	$+$
model year	multi-valued discrete	$+$
origin	multi-valued discrete	$+$

Charles River dummy variable. In the experiments we used a number of data sets with a binary target, one with 9 class values (Employee Selection) and some with a numeric target (see Table 1). The numeric targets have been discretized into two and four intervals; the intervals were chosen so that each one contained approximately the same number of cases. In order to check whether our a priori ideas about monotonicity are confirmed by the data, we compared the number of non-monotone pairs present in a data set of size N, to the average number of non-monotone pairs of that same data set, but with the N class labels randomly permuted. The idea is that such a random permutation of class labels represents a non-monotone process, and hence the distribution obtained by computing the number of non-monotone pairs for a great number of such random permutations, can be loosely interpreted as its distribution under the null-hypothesis of a non-monotone process. Table 3 shows the result of these computations for the data sets with binary class label. The last column gives the ratio of the observed number of non-monotone pairs (as given in the first column) to its average for 1000 permuted data sets (as given in the second column). Note that the Haberman data set has by far the highest ratio, and so perhaps the monotonicity assumption is dubious in this case. We don't have a clearcut criterion to decide on that however. Table 4 provides the same information for the data sets with non-binary classes.

Table 3. Monotonicity test results: two classes

Data set	# pairs	Mean	Std.dev.	Ratio
Australian	8087	42649.0	2258.1	0.190
AutoMpg	21	7716.5	733.3	0.003
Boston	309	6113.0	612.5	0.051
Haberman	1784	2848.2	379.4	0.626
Machine	86	2714.0	311.4	0.032
Pima	482	4923.9	569.2	0.098
Windsor	429	10187.0	875.1	0.042
Wisconsin	9	14472.0	1357.4	0.001

Table 4. Monotonicity test results: non-binary classes

Data set	# pairs	Mean	Std.dev.	Ratio
AutoMpg	74	11557.0	816.4	0.006
Boston	691	9175.8	739.4	0.075
Employee	1125	34236.0	1600.1	0.033
Machine	167	4070.2	359.2	0.041
Windsor	1328	15302.0	970.2	0.087

The experiments were performed with 10-fold cross-validation. For monotone nearest neighbour this was done as follows. For each fold, we

1. relabeled the observations in 9 parts of the data to remove any monotonicity violations;
2. used the relabeled data to predict the class labels of the remaining part with the monotone nearest neighbour rule.

For prediction we considered the quasi-monotone prediction rule (predictions have to be consistent only with the training data) as well as the monotone prediction rule (predictions also have to be consistent among themselves). For the monotone prediction rule, the points predicted thus far were used *only* to determine the interval of allowed class labels for a new point; they were not used in voting for the class label of the new point.

The results for the quasi monotone prediction rule for problems with two classes are given in Table 6, and for problems with more than two classes in Table 7. They were computed with prediction rule variant 1 as discussed in Section 4. Preliminary experiments showed the results of the two variants were virtually the same, and variant 1 is easier to compute. Likewise, the results for the monotone prediction rule were virtually identical to those for the quasi monotone rule, and are therefore not reported separately. The results have been summarized in Table 6 and Table 7 as follows. We took the best result of kNN for $k = 1, 3, 5$ and compared its error with the error of monotone kNN for that same value of k (usually they had their lowest error rate for the same value of k). The last column indicates what value of k that was. The p-values were

Table 5. Cross-table for comparison of classifiers. For example, b is the number of cases classified incorrectly by kNN but correctly by monotone kNN.

	I_{mkNN}	C_{mkNN}
I_{kNN}	a	b
C_{kNN}	c	d

Table 6. Comparison of error rates of kNN and monotone kNN with quasi-monotone prediction rule for two-class problems

data set	kNN	mkNN	Winner	p-value	k
Australian	18.3%	16.4%	mkNN	0.0984	5
AutoMPG	8.7%	7.9%	mkNN	0.6476	1
Boston	20.6%	18.8%	mkNN	0.1996	1
Haberman	26.5%	28.4%	kNN	0.4050	3
Machine	15.3%	14.8%	mkNN	1	1
Pima	25.7%	25.9%	kNN	0.9050	3
Windsor	26.4%	20.9%	mkNN	0.0001	5
Wisconsin	3.7%	3.5%	mkNN	1	5

computed using an exact binomial test. We computed a cross-table as given in Table 5 and performed a binomial test of b successes on $b + c$ trials under

$$H_0 : \pi = \tfrac{1}{2} \qquad\qquad H_a : \pi \neq \tfrac{1}{2}$$

where π denotes the probability of success. The p-value was computed with the function `binom.test` in the `R` system[3].

Comparing the error rates for the two-class problems (see Table 6), monotone kNN performs better in all cases, except for the Haberman and Pima data sets. The result for Haberman is not surprising, given the relatively high number of nonmonotone pairs we found in our preliminary calculations. For the Pima data, this ratio is also relativey high, but not as high as for the Australian data, and there we *did* find a substantial improvement of monotone kNN over standard kNN. Hence, the computed ratio by itself is not a perfect indicator for the success of the monotone model.

Looking at the problems with more than two classes (see Table 7), the advantage of enforcing the monotonicity constraint appears even more prominent. Monotone kNN has the lower estimated error rate in all cases, and in two cases significantly so. The effect of the monotonicity constraint can be appreciated clearly by looking at the performance for $k = 1$ (see Table 8): it appears to reduce if not prevent the overfitting of standard kNN. The *Employee Selection* data set is a good example: standard kNN breaks down, whereas monotone kNN isn't performing much worse than for higher values of k.

[3] See `www.r-project.org`

Table 7. Comparison of error rates of kNN and monotone kNN with quasi-monotone prediction rule for problems with more than two classes

data set	kNN	mkNN	Winner	p-value	k
AutoMPG	22.2%	21.9%	mkNN	1	1
Boston	49.8%	42.1%	mkNN	7.2×10^{-6}	5
ESL	30.1%	29.7%	mkNN	0.890	5
Machine	37.8%	34.5%	mkNN	0.324	3
Windsor	51.8%	46.5%	mkNN	0.009	3

Table 8. Comparison of error rates of kNN and monotone kNN for problems with more than two classes and $k = 1$

data set	kNN	mkNN	Winner	p-value
AutoMPG	22.2%	21.9%	mkNN	1
Boston	50.0%	44.5%	mkNN	0.0015
ESL	45.1%	30.5%	mkNN	2.231×10^{-11}
Machine	39.7%	33.5%	mkNN	0.066
Windsor	52.4%	46.2%	mkNN	0.0017

We conclude on the basis of these experiments that enforcing the monotonicity constraint does not lead to a deterioration of predictive accuracy, on the contrary, we have found it usually leads to an improvement. In addition, the monotone models are much more likely to be accepted by their intended users, since its predictions are in accordance with their qualitative domain knowledge.

7 Conclusion

We have proposed an adaptation of the k-nearest neighbour rule, to allow for the inclusion of monotonicity constraints. Such constraints can often be elicited reliably from subject area experts. We have shown that the use of monotonicity constraints can give substantial improvements in predictive performance over the standard k-nearest neighbour classifier. More importantly, the resulting models are much more likely to be accepted by their intended users, because their predictions are in accordance with their qualitative domain knowledge.

The results we obtained encourage us to explore further possibilities in this direction. One could, for example, investigate how minimization of L_1 or L_2 loss in the relabeling phase (as proposed by Dykstra et al. [11]) would influence the predictive performance of the monotone nearest neighbour rule. Minimization of the error rate on the training sample (as performed by the current relabeling algorithm) does after all not necessarily lead to the lowest error on a test sample. Another possibilty is to look for improvements in the relabelling phase. Currently, points are relabelled to arbitrary values from their allowed intervals. More sophisticated alternatives could be considered here.

References

1. Altendorf, E.A., Restificar, A.C., Dietterich, T.G.: Learning from sparse data by exploiting monotonicity constraints. In: Bacchus, F., Jaakkola, T. (eds.) Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005), pp. 18–25. AUAI Press (2005)
2. Anglin, P.M., Gençay, R.: Semiparametric estimation of a hedonic price function. Journal of Applied Econometrics 11(6), 633–648 (1996)
3. Archer, N.P., Wang, S.: Application of the backpropagation neural network algorithm with monotonicity constraints for two-group classification problems. Decision Sciences 24(1), 60–75 (1993)
4. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
5. Ben-David, A.: Automatic generation of symbolic multiattribute ordinal knowledgebased DSS: methodology and applications. Decision Sciences 23, 1357–1372 (1992)
6. Ben-David, A., Sterling, L., Pao, Y.: Learning and classification of monotonic ordinal concepts. Computational Intelligence 5, 45–49 (1989)
7. Ben-David, A.: Monotonicity maintenance in information-theoretic machine learning algorithms. Machine Learning 19, 29–43 (1995)
8. Bloch, D.A., Silverman, B.W.: Monotone discriminant functions and their applications in rheumatology. Journal of the American Statistical Association 92(437), 144–153 (1997)
9. Cao-Van, K.: Supervised ranking, from semantics to algorithms. PhD thesis, Universiteit Gent (2003)
10. Druzdzel, M.J., van der Gaag, L.C.: Elicitation of probabilities for belief networks: combining qualitative and quantitative information. In: Besnard, P., Hanks, S. (eds.) Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI 1995), pp. 141–148. Morgan Kaufmann, San Francisco (1995)
11. Dykstra, R., Hewett, J., Robertson, T.: Nonparametric, isotonic discriminant procedures. Biometrika 86(2), 429–438 (1999)
12. Feelders, A., Pardoel, M.: Pruning for monotone classification trees. In: Berthold, M.R., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 1–12. Springer, Heidelberg (2003)
13. Feelders, A., van der Gaag, L.: Learning Bayesian network parameters with prior knowledge about context-specific qualitative influences. In: Bacchus, F., Jaakkola, T. (eds.) Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005), pp. 193–200. AUAI Press (2005)
14. Feelders, A., Velikova, M., Daniels, H.: Two polynomial algorithms for relabeling nonmonotone data. Technical Report UU-CS-2006-046, Department of Information and Computing Sciences, Utrecht University (2006)
15. Ford, L.R., Fulkerson, D.R.: Flows in networks. Princeton University Press, Princeton (1962)
16. Gamarnik, D.: Efficient learning of monotone concepts via quadratic optimization. In: Proceedings of the eleventh annual conference on computational learning theory, pp. 134–143. ACM Press, New York (1998)
17. Garey, M., Johnson, D.: Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York (1979)
18. Karpf, J.: Inductive modelling in law: example based expert systems in administrative law. In: Proceedings of the third international conference on artificial intelligence in law, pp. 297–306. ACM Press, New York (1991)
19. Meyer, M.A., Booker, J.M.: Eliciting and Analyzing Expert Judgment: A Practical Guide. Statistics and Applied Probability. ASA-SIAM, Philadelphia (2001)
20. Möhring, R.H.: Algorithmic aspects of comparability graphs and interval graphs. In: Rival, I. (ed.) Graphs and Order, pp. 41–101. Reidel (1985)

21. Pazzani, M.J., Mani, S., Shankle, W.R.: Acceptance of rules generated by machine learning among medical experts. Methods of Information in Medicine 40, 380–385 (2001)
22. Potharst, R., Bioch, J.C.: Decision trees for ordinal classification. Intelligent Data Analysis 4(2), 97–112 (2000)
23. Rademaker, M., De Baets, B., De Meyer, H.: On the role of maximal independent sets in cleaning data for supervised ranking. In: 2006 IEEE International Conference on Fuzzy Systems, pp. 1619–1624 (2006)
24. Robertson, T., Wright, F., Dykstra, R.L.: Order Restricted Statistical Inference. Wiley, Chichester (1988)
25. Royston, P.: A useful monotonic non-linear model with applications in medicine and epidemiology. Statistics in Medicine 19(15), 2053–2066 (2000)
26. Sill, J.: Monotonic networks. In: Advances in neural information processing systems. NIPS, vol. 10, pp. 661–667 (1998)