Enforcing monotonicity of decision models: algorithm and performance

A case study of hedonic price model

Marina Velikova[1] and Hennie Daniels[1,2]

[1]Tilburg University, CentER for Economic Research,Tilburg, PO Box 90153,
5000 LE The Netherlands, phone: +31 13 466 8721, fax: +31 13 466 3377,
email:M.Velikova@uvt.nl
[2]Erasmus University Rotterdam, ERIM Institute of Advanced Management Studies,
Rotterdam, The Netherlands.

Abstract
The objective of data mining is the derivation of knowledge from databases, for example to produce decision rules. In practice one often encounters difficulties with models that are constructed purely by search, without incorporation of knowledge about the domain. In economic decision making like for example credit loan approval, or risk analysis one often requires models that are monotone with respect to the decision variables involved. If the model is constructed by a blind search on the data it does mostly not have this property even if the underlying data are monotone. In this paper we present methods to enforce monotonicity of decision models. We propose measures to express the degree of monotonicity of the data and an algorithm to clean non-monotone data sets. In addition we show that the performance of the models obtained in this way is better. This is illustrated using artificially generated data and a real case study.

Keywords: data mining, domain knowledge, monotonicity, monotonic datasets, decision trees

1. Introduction

Data mining has attracted a lot of interest in recent years due to the growing amount of data collected in business and the need to turn this data into useful knowledge. The objective of a data mining system is to derive valuable knowledge implicitly present in large databases. Although, in data mining literature, the main emphasis is put on the analysis and interpretation phase, there are more aspects such as data selection and data pre-processing, which determine the successful implementation of any data mining system. The right description of the domain as well as data cleaning, data integration and data transformation can significantly improve the efficiency of the data mining process.

Apart from limitations regarding data quality there are also problems in the application of the model if knowledge discovery is conducted by blind research. Frequently the models are incompatible with business regulations. When the rules must be enforced in the business process it can be a problem if knowledge is derived using data mining algorithms.
Another problem is the lack of interpretability of the model. In general, human decision makers require that the model is easy to understand and do not accept black box models.
Finally, data mining algorithms may produce models, which are hard to manage by human decision makers due to their huge complexity.
Therefore, there is a need for integration of the knowledge discovered by standard data mining algorithms with the knowledge based on intuition and experience of the domain experts.

In this paper, we explicitly consider the implementation of a special form of a prior knowledge that is typical in economic decision problems, namely the monotonicity of relationship between the dependent and explanatory variables.

In recent years, several researchers have become interested in incorporation of monotonicity constraints in different data mining methods. In ([Dan, 99]) a class of a neural network that is monotone by construction is described. This class is obtained by considering multilayer neural networks with non-negative weights. In ([Wang, 94]) the monotonicity of the neural network is guaranteed by enforcing constraints during the training process.
Also data analysis methods have been developed for classification problems with monotonicity constraints. In ([Ben-David, 95]), a new splitting measure for constructing a decision tree was proposed including a non-monotonicity index and standard impurity measure such as entropy. In this way, Ben-David balances monotonicity and classification error. Potharst ([Pot, 99]) provides a study for building monotonic decision trees using only monotonic data sets. He presents algorithms for construction monotone tree by adding the corner elements of a node with an appropriate class label to the dataset as well as by repairing any minor local non-monotonicities. Rather than enforcing the monotonicity during the tree construction, Potharst and Feelders ([Pot, 02]) consider an alternative approach that generates many different trees by resampling the training data and selects a monotonic tree. This approach allows the use of a standard tree algorithm except that the minimum and maximum elements of the nodes have to be recorded during tree construction, in order to be able to check whether the final tree is monotone.

In practice the data recorded for some transactions can be non-monotone, even if the underlying business process is supposed to be monotone. This is due to the noise in the data recorded, for example human or computers errors at data entry, inconsistencies after merging datasets, discrepancies due to the change of data over time, etc. Noisy data can cause confusion for the mining procedure, resulting in unreliable output. Particularly, in monotonic classification problems, this result could be incompatible with policy rules and business regulations.

Therefore, a pre-processing step is necessary to "clean" the data by removing the noise and resolving the inconsistencies. In the present paper we propose a technique for dealing with noisy data in a non-monotonic dataset in order to change it into monotonic one. This is an algorithm for relabeling the dependent variable in a dataset. Furthermore, we derive measures for the degree of non-monotonicity of arbitrary datasets. In this way randomly generated datasets can be used as benchmarks for real datasets with the same structure.
The algorithm is applied to artificially generated data and to a case study of predicting house prices. Using the artificially generated datasets we show that the algorithm is capable of removing noise. In the real case study we show that the monotonic datasets produce models, which are more reliable and outperform the models derived from raw data.

In the next section, we formulate the monotonicity constraints in regression and classification problems. A measure for the degree of non-monotonicity in a randomly generated dataset is derived in section 3 using it later as a benchmark for comparison with real datasets. The algorithm for transformation of a non-monotonic dataset into monotonic one is introduced in section 4. In section 5, we provide some simulation results received after implementation the algorithm on artificially generated datasets. In order to determine the effect of using a monotonic dataset in real problems, in section 6, we consider a case study of house pricing where we implement the algorithm and compare the performance of the decision models obtained from the original and transformed datasets. Conclusions and final remarks are given in section 7.

2. Monotonicity

In many economic classification and regression problems it is known that the dependent variable has a distribution that is monotonic with respect to the independent variables. Economic theory would state that people tend to buy less of a product if its price increases (ceteris paribus), so there would be a negative relationship between price and demand. The strength of this relationship and the precise functional form are however not always dictated by economic theory. Another well-known example is the dependence of labour wages as a function of age and education ([Muk, 94]). In loan acceptance the decision rule should be monotone with respect to income for example, i.e. it would not be acceptable that an applicant with high income is rejected, whereas another applicant with low income and otherwise equal characteristics is accepted. Monotonicity is also imposed in so-called hedonic price models where the price of a consumer good depends on a bundle of characteristics for which a valuation exists ([Har, 78]).

The mathematical formulation of the monotonicity rule is straightforward. We assume that y is the dependent variable and takes values in $\boldsymbol{Y}$ and the vector of independent variables is x and takes values in $\boldsymbol{X}$. In the applications discussed here, $\boldsymbol{Y}$ is a one-dimensional space of prices or classes and $\boldsymbol{X}$ is a n-dimensional space of characteristics of products or customers for example. Furthermore we assume that we have a dataset (y^p, x^p) of points in $\boldsymbol{Y}*\boldsymbol{X}$, which can be considered as a random sample of the joint distribution of (y, x). In a regression problem we want to estimate $E(y \mid x)$. $E(y \mid x)$ depends monotonically on x, if

$$x^1 \geq x^2 \Rightarrow E(y \mid x^1) \geq E(y \mid x^2) \qquad (1)$$

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

In cases where we are dealing with a classification problem we have an classification rule $r(x)$ that assigns a class to each vector x in $\boldsymbol{X}$. Monotonicity of r is defined by:

$$x^1 \geq x^2 \Rightarrow r(x^1) \geq r(x^2) \qquad (2)$$

3. Measure and benchmark for the degree of non-monotonicity in a dataset

Several researchers propose various measures to check the degree of monotonicity/non-monotonicity in different data mining tools. In [Dan, 99], Daniels and Kamp define a monotonicity index to measure the degree of monotonicity of a neural network with respect to each input variable, x_i as follows:

$$\text{mon}(x_i) = \frac{1}{n}\left|\sum_{p=1}^{n} I^{+}\left(\frac{\partial f}{\partial x_i}(x_p)\right) - I^{-}\left(\frac{\partial f}{\partial x_i}(x_p)\right)\right|$$

where $I^{+}(z) = 1$ if $z > 0$ and $I^{+}(z) = 0$ if $z \le 0$ and $I^{-}(z) = 1$ if $z \le 0$ and $I^{-}(z) = 0$ if $z=0$, n is the number of observations, x_p is the pth observation (vector) and f denotes the neural network solution. The value of this index is between zero, indicating a non-monotonic relationship, and 1, indicating a monotonic relationship. The value of sign indicates whether the relation of f with respect to x is increasing or decreasing.

To test whether a given decision tree is monotone or not, Potharst [Pot, 99] proposes an approach using the maximal and minimal elements of the leaf nodes of the decision trees. For all pairs of leaves, t_1 and t_2, it is checked whether there is a pair that satisfies one of the following conditions:

$r(t_1)>r(t_2)$ and $\min(t_1) \le \max(t_2)$ or $r(t_1)<r(t_2)$ and $\max(t_1) \ge \min(t_2)$.

In case there exists such a pair the decision tree is called non-monotonic. The degree of the non-monotonicity of the tree is computed as percentage non-monotonic leaf nodes of the total number of leaves.

The non-monotonicity index proposed by Ben-David ([Ben, 95]) is another measure for the degree of non-monotonicity, which gives equal weight to each pair of non-monotonic leaf nodes. A modification of this measure, given in [Pot, 02] is to weight the different leaves according to their probability of occurrence. The idea behind this is that when two low-probability leaves are non-monotonic with respect to each other, this violates the monotonicity of the tree to a lesser extent than two high-probability leaves.

All these measures for the degree of monotonicity/non-monotocity are rather based on the models derived from data mining tools such as neural networks and decision trees than on the dataset itself. In this section, we derive a benchmark for the degree of non-monotonicity in a given dataset considering a randomly generated dataset. Using this benchmark we can compare it with the degree of non-monotonicity in a real dataset computed as the proportion of the number of non-monotonic pairs from the total number of pairs. If the latter is significantly less than the benchmark this implies the presence of monotonicity in the dataset and one suitable tool for transformation the non-monotonic dataset into monotonic one could be the algorithm introduced in the next section.

Lemma 1:

For a randomly generated dataset with points drawn from uniform distribution, k-independent variables and L uniformly distributed labels, the expectation value of the fraction of non-monotonic pairs, denoted by Nm, is:

$$E\{Nm\} = 2^{-k}\,\frac{L-1}{L} \qquad (3)$$

Proof: *It will be provided in the final version of the paper*

4. Algorithm for relabeling

A dataset is defined to be monotone if for all possible combinations of data points the relation defined in (2) holds. The objective of the algorithm is to transform a given non-monotonic dataset into monotonic one by changing the value of the dependent variable. This process is called *relabeling*. The idea is to reduce the number of non-monotonic pairs by relabeling one data point in each step. In order to do this we choose a data point for which the increase in correctly labelled points is maximal (this is not necessarily the point which is involved in the maximal number of non-monotonic pairs). The process is continued until the dataset is monotone.

The correctness of the algorithm is proved by Lemma 2 and Lemma 3. In Lemma 2 we show that it is always possible to reduce the number of non-monotonic pairs by changing the label of only one point as long as the dataset is non-monotonic. In Lemma 3 it is shown that there is a canonical choice for the new label for which a maximal reduction can be obtained. There may be more than one label for which this can be achieved but these are all smaller or all larger than the current label of the point.
Let us first introduce some notations. The initial dataset of n points is denoted by $D = (x_n, \ell_n)$, where x_n is a vector of independent variables and ℓ_n is a label (dependent variable) with range 1,2,…,L. For each dataset D, Q(D) denotes the set of all points that participate in at least one non-monotonic pair.
For each data point $x \in Q(D)$, we define

$A_i(x) = \{y < x \mid \text{label } (y) = i\}$,
$B_i(x) = \{y > x \mid \text{label } (y) = i\}$,
a_i and b_i denote the number of points in $A_i(x)$ and $B_i(x)$, respectively
N_ℓ denotes the total number of points correctly labelled with respect to x for the current label of x, ℓ, i.e.
$N_\ell = a_1+a_2+\ldots+a_\ell+b_\ell+\ldots+b_L$

Remark 1:
We assume that all data points in the dataset D are unique i.e. no points are represented twice.

For each data point $x \in Q(D)$ we compute the label ℓ' for which there is a maximal increase in the number of correctly labelled points with respect to x, if the label of x is changed into ℓ'. The maximal increase is denoted by I_{max}. In case there is more than one label with one and the same maximal increase in correctly labelled points, we choose the closest label to the current label of x. In the next step we select a point $x \in Q(D)$ for which I_{max} is the largest and change its label. This process is repeated until the dataset is monotonic.

Algorithm
Step 1 – Initialisation:
Compute Q(D) on the basis of the dataset D
Step 2 – Main program
Step 2.1 As long as $Q(D) \neq \varnothing$
For each data point $x \in Q(D)$ compute
2.1.1 $I_{max} = \max \{ N_{\ell'} - N_\ell \mid 1 \leq \ell' < L\}$
2.1.2 Λ - set of indices ℓ' for which $N_{\ell'} - N_\ell$ is maximal
2.1.3 Form a triple (x, I_{max}, λ) where $\lambda \in \Lambda$ is the closest label to ℓ, (in Lemma 3 it is shown that λ is unique).
Step 2.2 From all triples choose the one where I_{max} is maximal and change the label into ℓ'.
Step 2.3 Update Q(D) on the basis of the modified dataset D.

Remark 2:
In general, the points correctly labelled with respect to x are all points incomparable to x as well as the points in $A_1 \cup A_2 \cup \ldots \cup A_\ell$ and $B_\ell \cup B_{\ell+1} \cup \ldots \cup B_L$. Since the number of the points incomparable to x is constant and it does not contribute to I_{max}, we may completely ignore it during the computation.

Lemma 2:
Let D^k denote the dataset D after k-iterations. If $Q(D^k) \neq \varnothing$ there is at least one point $x \in Q(D^k)$ that can be relabelled such that the number of non-monotonic pairs is reduced.

Proof: *It will be provided in the final version of the paper*

Lemma 3:
Suppose that the maximal increase I^x_{max} in correctly labelled points w.r.t. x can be obtained by at least two labels r and s, r < s. Then

$r < s < \ell_x$ or $\ell_x < r < s$

where ℓ_x is the label of x.

Proof: *It will be provided in the final version of the paper*

Correctness of the algorithm
In each step the number of points participating in non-monotonic pairs is reduced by at least one (Lemma 2). Since the algorithm can only terminate when Q(D)=0 the resulting dataset is monotonic. By Lemma 3 it follows that there is only one canonical choice for the new labels.

5. Simulation results

In order to check to what extend noise added to a monotone dataset can be removed by the algorithm, we conducted the following experiment. We firstly generated a dataset with random points uniformly distributed between 0 and 1 and computed the label of each point by applying a monotonic function on the independent variables. Then the continuous dependent variable (label) was discretized into finite number of classes. In the next step, we turned the monotonic dataset into non-monotonic one by adding random noise to the discrete labels. After that we applied the algorithm and compared the labels by computing the percentage of correctly restored labels. This experiment was repeated 10 times with different number of points, independent variables and labels as well as different noise levels. The results are summarized in Table 1 below:

# points in a dataset	# independent variables	# labels	Noise	Restoration (%)
100	2	3	15 %	99 %
100	2	3	15 %	98 %
100	2	4	11 %	96 %
100	3	4	15 %	94 %
100	5	3	15 %	88 %
200	2	3	15 %	97 %
200	3	4	16 %	92 %
200	3	5	16 %	92 %
200	5	4	15 %	89 %
200	7	5	15 %	88 %

Table 1: The results of data cleaning

The results show that the algorithm restores to a large extend the original dataset (7 of 10 times the restoration is above 90%). In the rest cases the restoration is less due to the increase of the number of independent variables and labels.

In order to determine the performance of the original non-monotonic dataset and the transformed monotonic dataset we applied them in a tree-based algorithm presented in [Pot, 02], that is in many respects similar to the CART program as described in ([Bre, 84]). The program only makes binary splits and uses the Gini-index as splitting criterion. Furthermore cost-complexity pruning is applied to generate a nested sequence of trees from which the best one is selected on the basis of test set performance. During tree construction, the algorithm records the minimum and maximum element for each node. These are used to check whether a tree is monotone.

On the basis of this algorithm we repeated the following experiment 50 times with the first dataset given in Table 1 using both the original and transformed datasets. Each dataset was randomly partitioned (within classes) into a training set of 50 observations and test set of 50 observations. The training set was used to construct a sequence of trees using cost-complexity pruning. From this sequence the best tree was selected on the basis of error rate on the test set (in case of a tie, the smallest tree was chosen). Finally, it was checked whether the tree was monotone and if not, the upper bound for the degree of non-monotonicity was computed by giving a pair t_1, t_2 of non-monotonic leaf nodes weight $2 * p(t_1) * p(t_2)$, where $p(t_i)$ denotes the proportion of cases in leaf i.

The results show that the model yielded from the monotonic dataset has better performance than that yielded from the non-monotonic dataset considering the average error on the trees – the average error rate on monotonic and non-monotonic trees for monotonic dataset is almost twice less that that for non-monotonic dataset. Also the average degree of non-monotonicity for monotonic dataset is very low in comparison with the result for the non-monotonic dataset. All the results are summarized in Table 2 below:

	Monotonic dataset	**Non-monotonic dataset**
# monotonic trees	45	41
# non-monotonic trees	5	9
Average error rate on monotonic trees	0.147	0.283
Average number of leaf nodes on monotonic trees	5.6	3.6
Average error rate on non-monotonic trees	0.156	0.293
Average number of leaf nodes on non-monotonic trees	8.4	12.1
Average degree of non-monotonicity	0.003	0.062

Table 2: Comparison of the results received from monotonic and non-monotonic datasets

6. Case study - Hedonic price model

The basic principle of a hedonic price model is that the consumption good is regarded as a bundle of characteristics for which a valuation exists ([Har,78]). The price of the good is determined by a combination of these valuations:

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

In the case study presented below we want to predict the house price given a number of characteristics. So, the variables $x_1, x_2, ..., x_n$ correspond to the characteristics of the house. The dataset consists of 119 observations of houses in the city of Den Bosch, which is a medium sized Dutch city with approximately 120,000 inhabitants. The explanatory variables have been selected on the basis of interviews with experts of local house brokers, and advertisements offering real estate in local magazines. The most important variables are listed in Table 3.

Symbol	Definition
DISTR	Type of district, four categories ranked from bad to good
SURF	Total area including garden
RM	Number of bedrooms
TYPE	1. Apartment 2. Row house 3. Corner house 4. Semidetached house 5. Detached house 6. Villa
VOL	Volume of the house
GARD	Type of garden, four categories ranked from bad to good
GARG	1. No garage 2. Normal garage 3. Large garage

Table 3: Definition of model variables

Of all 7021 distinct pairs of observations, 2217 are comparable, and 78 are non-monotonic. For the purpose of this study we have discretized the dependent variable (asking price) into three classes with labels '1', '2' and '3'. After the discretization of the dependent variable the number of the non-monotonic pairs was reduced to 25 i.e. the degree of non-monotonicity is 0.36% (number of non-monotonic pairs divided by the total number of pairs). Comparing this result with the result from the benchmark (3) for 3 labels and 7 independent variables, which is 0.52%, we can conclude that the monotonicity is present in the dataset.

Therefore, in the next step, we applied the algorithm for relabeling described above, which led to the label change of 5 points.

Again, in order to determine the performance of the original non-monotonic dataset and the transformed monotonic dataset, we applied them in a tree-based algorithm and repeated 100 times the experiment described in section 5. The results are shown in Table 4:

	Monotonic dataset	Non-monotonic dataset
# monotonic trees	57	49
# non-monotonic trees	43	51
Average error rate on monotonic trees	0,247695	0,289519
Average number of leaf nodes on monotonic trees	4,47	4,16
Average degree of non-monotonicity	0,012339	0,022985

Table 4: Comparison of the results received from monotonic and non-monotonic house pricing datasets

In the next step, we held a two-sample t-test of the null hypothesis that average error rate on monotonic trees is one and the same for the monotonic and non-monotonic datasets against one-sided alternative that the former is less than the latter. The test yielded a p-value 0.00000452, which leads to rejection of the null hypothesis and respectively to the conclusion that the average error on monotonic trees for the monotonic datasets is significantly less than that for non-monotonic datasets.
Furthermore, the average degree of non-monotonicity for monotonic datasets is almost twice less than that for non-monotonic datasets, which along with the result that monotonic datasets yield more monotonic decision trees than non-monotonic datasets, shows that the model yielded from the monotonic dataset has better performance and produces more reliable model.

7. Conclusion

In the present paper, we have shown that the incorporation of prior knowledge can significantly improve the effectiveness of a data mining process. We explicitly consider a very common form of domain knowledge, which is present in many economic problems, namely the monotonic relationship between dependent variable (label) and explanatory variables. Usually the data sets used for solving monotonic classification problems are non-monotonic due to the noise in the data, which can result in unreliable output and incompatibility of the model with policy rules and business regulations. Therefore, in this paper, we introduce an algorithm for relabeling the dependent variable in a non-monotonic dataset and thus transform it into monotonic one. Using the algorithm in a real case study of predicting house prices, we show that the models derived from the cleaned data show better performance than those derived from the original data.

References

[Ben, 95]: Ben-David, A., "Monotonicity Maintenance in Information-Theoretic Machine Learning Algorithms", *Machine Learning*, **19,** pp. 29-43, (1995).

[Bre, 84]: Breiman L., Friedman J. H. Olshen R. A. and Stone C. T., "Classification and Regression Trees", Wadsford, California, (1984).

[Dan, 99]: Daniels, H. A. M. and Kamp, B., "Application of MLP networks to bond rating and house pricing", *Neural Computation and Applications*, **8**, pp. 226-234, (1999).

[Fee, 00]: Feelders, A., Daniels, H. A. M. and Holsheimer, M, "Methodological and practical aspects of data mining", *Information & Management*, **37**, pp. 271-281, (2000).

[Har, 78]: Harrison, O. and Rubinfeld, D., "Hedonic prices and the demand for clean air", *Journal of Environmental Economics and Management*, **53**, pp. 81-102, (1978).

[Muk, 94]: Mukarjee, H. and Stern, S., "Feasible Nonparametric Estimation of Multiargument Monotone Functions", *Journal of the American Statistical Association*, **89**, no.425, pp. 77-80, (1994).

[Nun 91]: Nunez, M., "The Use of Background Knowledge in Decision Tree Induction", *Machine Learning*, **6**, pp. 231-250, (1991).

[Pot, 99]: Potharst, R., "Classification using decision trees and neural nets", Erasmus Universiteit Rotterdam, *SIKS Dissertation Series* No. 99-2, (1999).

[Pot, 02]: Potharst, R. and A.Feelders, "Classification trees for problems with monotonicity constraints", *SIGKDD Explorations Newsletter*, Volume 4 , Issue 1 (2002)

[Wan, 94]: Wang, S., "A neural network method of density estimation for univariate unimodal data", *Neural Computation & Applications*, **2**, pp. 160- 167, (1994).