

Prior knowledge in economic applications of data mining

A.J. Feelders

Tilburg University
Faculty of Economics
Department of Information Management
PO Box 90153
5000 LE Tilburg, The Netherlands
A.J.Feelders@kub.nl

Abstract. A common form of prior knowledge in economic modelling concerns the monotonicity of relations between the dependent and explanatory variables. Monotonicity may also be an important requirement with a view toward explaining and justifying decisions based on such models. We explore the use of monotonicity constraints in classification tree algorithms. We present an application of monotonic classification trees to a problem in house pricing. In this preliminary study we found that the monotonic trees were only slightly worse in classification performance, but were much simpler than their non-monotonic counterparts.

1 Introduction

The estimation of economic relationships from empirical data is studied in the field of econometrics. In the model specification stage of econometric modelling the relevant explanatory variables and the functional form of the relationship with the dependent variable are derived from economic theory. Then the relevant data are collected and the model is estimated and tested. Applied econometric work does not conform to this textbook approach however, but is often characterized *specification searches* ([Lea78]).

Data mining is often associated with the situation where little prior knowledge is available and an extensive search over possible models is performed. Of course one has to have some prior beliefs, for how else does one for example decide which explanatory variables to include in the model? But often the algorithm is able to select the relevant variables from a large collection of variables and furthermore flexible families of functions are used. Even though data mining is often applied to domains where little theory is available, in some cases useful prior knowledge is available, and one would like the mining algorithm to make use of it one way or the other.

One type of prior knowledge that is often available in economic applications concerns the sign of a relation between the dependent and explanatory variables. Economic theory would state that people tend to buy less of a product if its price increases (*ceteris paribus*), so price elasticity of demand should be negative. The

strength of this relationship and the precise functional form are however not always dictated by economic theory. The usual assumption that such relationships are linear are mostly imposed for mathematical convenience.

This paper is organized as follows. In section 2 we focus the discussion on prior knowledge for applications in economics. One of the most common forms of prior knowledge in economics and other application domains concerns *monotonicity* of relations. This subject is explored further in section 3, where we discuss monotonicity in classification tree algorithms. In section 4 we present an application of monotonic classification trees to a problem in house pricing. Finally, in section 5 we draw a number of conclusions from this study.

2 Prior knowledge in economic applications

Regression analysis is by far the most widely used technique in econometrics. This is quite natural since economic models are often expressed as (systems of) equations where one economic quantity is determined or explained by one or more other quantities.

The a priori domain knowledge is primarily used in the *model specification* phase of the analysis. Such a priori knowledge is supposed to be derived largely from *economic theory*. Model specification consists of the following elements:

1. Choice of dependent and explanatory variables.
2. Specification of the functional form of the relation between dependent and explanatory variables.
3. Restrictions on parameter values.
4. Specification of the stochastic process.

In applied econometrics usually alternative specifications are tried, and the specification, estimation and testing steps are iterated a number of times (see [Lea78] for an excellent exposition of different types of *specification-searches* used in applied work). As a historical note, the search for an adequate specification based on preliminary results has sometimes been called “data mining” within the econometrics community [Lea78,Lov83]. In principle, there is nothing wrong with this approach, its combination however with classic testing procedures that do not take into account the amount of search performed have given “data mining” a negative connotation.

We shall give an example from empirical demand theory to illustrate how different types of domain knowledge may be used in the model specification phase. Empirical demand theory asserts that *ceteris paribus* an individual’s purchases of some commodity depend on his income, the price of the commodity and the price of other commodities. This is reflected in a simple demand equation [Lea78]:

$$\log D_i^o = a + b \log P_i^o + c \log Y_i + d \log P_i^g + \varepsilon_i$$

where D^o denotes the purchases of oranges, P^o the price of oranges, Y denotes income, P^g denotes the price of grapefruit, and index i stands for different households. The log-linear specification is chosen primarily for convenience. It allows

us to interpret the estimated coefficients as elasticities, e.g. the estimate of b is interpreted as the price elasticity of demand, and the estimate of c as the income elasticity of demand. A priori we would expect that $b < 0$ (if price increases, *ceteris paribus* demand decreases). Likewise we would expect $c > 0$ and $d > 0$ (since grapefruits are a substitute for oranges).

According to economic theory there should be absence of money illusion, i.e. if income and all prices are multiplied by the same constant, demand will not change. If we believe in the “absence of money illusion” we can add the restriction that the demand equation should be homogeneous of degree zero. For the log-linear specification this leads to the constraint $b + c + d = 0$.

3 Domain knowledge in trees

Tree-based algorithms such as CART[BFO84] and C4.5[Qui93] are very popular in data mining. It is therefore not surprising that many variations on these basic algorithms have been constructed to allow for the inclusion of different types of domain knowledge such as the cost of measuring different attributes and misclassification costs. Another common form of domain knowledge concerns monotonicity of the allocation rule.

Let us formulate the notion of monotone classification more precisely. Let (x_1, x_2, \dots, x_p) denote a vector of linearly ordered features. Furthermore, let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_p$ be the feature space, with partial ordering \geq , and let \mathcal{C} be a set of classes with linear ordering \geq . An allocation rule is a function $r : \mathcal{X} \rightarrow \mathcal{C}$ which assigns a class from \mathcal{C} to every point in the feature space. Let $r(\mathbf{x}) = i$ denote that an entity with feature values \mathbf{x} is assigned to the i^{th} class.

An allocation rule is monotone if

$$\mathbf{x}_1 \geq \mathbf{x}_2 \Rightarrow r(\mathbf{x}_1) \geq r(\mathbf{x}_2),$$

for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$.

A classification tree partitions the feature space \mathcal{X} into a number of hyperrectangles (corresponding to the leaf nodes of the tree) and elements in the same hyperrectangle are all assigned to the same class. As is shown in [Pot99], a classification tree is non-monotonic if and only if there exist leaf nodes t_1, t_2 such that

$$r(t_1) > r(t_2) \text{ and } \min(t_1) \leq \max(t_2),$$

where $\min(t)$ and $\max(t)$ denote the minimum and maximum element of t respectively. A dataset (\mathbf{x}_n, c_n) , is called monotone if

$$\mathbf{x}_i \geq \mathbf{x}_j \Rightarrow c_i \geq c_j,$$

for all $i, j = 1, \dots, n$.

Potharst [Pot99] provides a thorough study for the case that the training data may be assumed to be monotone. This requirement however makes the algorithms presented of limited use for data mining. For example, in loan evaluation the dataset used to learn the allocation rule would typically consist of

loans accepted in the past together with the outcome of the loan (say, defaulted or not). It is very unlikely that this dataset would be monotone.

A more pragmatic approach is taken by Ben-David [BD95], who proposes a splitting rule that includes a non-monotonicity index in addition to the usual impurity measure. This non-monotonicity index gives equal weight to each pair of non-monotonic leaf nodes.

A possible improvement of this index would be to give a pair t_1, t_2 of non-monotonic leaf nodes weight $p(t_1) \times p(t_2)$, where $p(t)$ denotes the proportion of cases in leaf t . 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. The reader should note that $p(t_1) \times p(t_2)$ is an upperbound for the degree of non-monotonicity between node t_1 and t_2 because not all their elements have to be non-monotonic with respect to each other.

The use of a non-monotonicity index in determining the best split has certain drawbacks however. Monotonicity is a *global* property, i.e. it involves a relation between different leaf nodes of a tree. If the degree of monotonicity is measured for each possible split during tree construction, the *order* in which nodes are expanded becomes important. For example, a depth-first search strategy will generally lead to a different tree than a breadth-first search. Also, a non-monotonic tree may become monotone after additional splits. Therefore we consider an alternative, computationally more intensive, approach in this study. Rather than *enforcing* monotonicity during tree construction, we generate many different trees and *check* if they are monotonic. The collection of trees may be obtained by drawing bootstrap samples from the training data, or making different random partitions of the data in a training and test set. 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. This approach has the additional advantage that one can estimate to what extent the assumption of monotonicity is correct. In the next section we apply this idea to an economic data set concerning house prices.

4 Den Bosch housing data

In this section we discuss the application of monotonic classification trees to the prediction of the asking price of a house in the city of Den Bosch (a medium sized Dutch city with approximately 120,000 inhabitants). 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 ([HR78]). The price of the good is determined by a combination of these valuations:

$$P = P(x_1, \dots, x_p)$$

In the case at hand the variables x_1, x_2, \dots, x_p are characteristics of the house. 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 1.

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
STOR	number of storeys
GARD	type of garden, four categories ranked from bad to good
GARG	1. no garage 2. normal garage 3. large garage

Table 1. Definition of model variables

It is a relatively small data set with only 119 observations. Of all 7021 distinct pairs of observations, 2217 are comparable, of which 78 are non-monotonic. For the purpose of this study we have discretized the dependent variable (asking price) into the classes “below median” (fl. 347,500) and “above median” (the current version of the algorithm can only handle binary classification). After this discretization of the dependent variable only 9 pairs of observations are non-monotonic.

The tree algorithm used is in many respects similar to the CART program as described in [BFOS84]. The program only makes binary splits and use the gini-index as splitting criterion. Furthermore it uses cost-complexity pruning [BFOS84] 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. The algorithm has been written in the Splus language [VR97].

In order to determine the effect of application of the monotonicity constraint we repeated the following experiment 100 times. The dataset was randomly partitioned (within classes) into a training set (60 observations) and test set (59 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 upperbound for the degree of monotonicity (as described in section 3) was computed.

Out of the 100 trees thus constructed, 61 turned out to be monotone and 39 not. The average misclassification rate of the monotonic trees was 14.9%, against 13.3% for the non-monotonic trees. Thus, the monotonic trees had a slightly worse classification performance. A two-sample t-test of the null hypothesis that monotonic and non-monotonic trees have the same classification error yielded a p-value of 0.0615 against a two-sided alternative. The average degree of non-monotonicity of the non-monotonic trees was about 1.7%, which is quite low, the more if we take into consideration that this is an upper bound. Another interesting comparison is between the average sizes of the trees. On average, the monotonic trees had about 3.13 leaf nodes, against 7.92 for the non-monotonic trees. Thus, the monotonic trees are considerably smaller and therefore easier to understand at the cost of only a slightly worse classification performance.

5 Conclusion

Monotonicity of relations is a common form of domain knowledge in economics as well as other application domains. Furthermore, monotonicity of the final model may be an important requirement for explaining and justifying model outcomes. We have investigated the use of monotonicity constraints in classification tree algorithms.

In preliminary experiments on house pricing data, we have found that the predictive performance of monotonic trees was comparable to, but slightly worse than, the performance of the non-monotonic trees. On the other hand, the monotonic trees were much simpler and therefore more insightful and easier to explain. This provides interesting prospects for applications where monotonicity is an absolute requirement, such as in many selection decision models.

References

- [BD95] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19:29–43, 1995.
- [BFOS84] L. Breiman, J.H. Friedman, R.A. Olshen, and C.T. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, California, 1984.
- [HR78] O. Harrison and D. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 53:81–102, 1978.
- [Lea78] E. Leamer. *Specification Searches: Ad Hoc Inference with Nonexperimental Data*. Wiley, 1978.
- [Lov83] Michael C. Lovell. Data mining. *The Review of Economics and Statistics*, 65(1):1–12, 1983.
- [Pot99] R. Potharst. *Classification using decision trees and neural nets*. PhD thesis, Erasmus Universiteit Rotterdam, 1999. SIKS Dissertation Series No. 99-2.
- [Qui93] J.R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.
- [VR97] W.N. Venables and B.D. Ripley. *Modern Applied Statistics with S-PLUS (second edition)*. Springer, New York, 1997.