A Method for Inductive Cost Optimization

Floor Verdenius
RIKS, Postbus 463, 6200 AL Maastricht, The Netherlands*

Abstract

In this paper we present a Method for Inductive Cost Optimization (MICO), as an example of induction biased by using background knowledge. The method produces a decision tree that identifies those setpoints that enable the process to produce in as cost-efficient a manner as possible. We report on two examples, one idealised and one real-world. Some problems concerning MICO are reported.

Keywords: Machine Learning, Biased Inductive Learning, Cost Optimization, Decision Trees

1. Introduction

Inductive learning, deriving knowledge from a real-world dataset, has been the subject of much work in artificial intelligence. Top Down Induction of Decision Trees (TDIDT) is a well known class of induction algorithms. Algorithms in this class recursively partition a dataset, forming a tree that describes decision points; a typical example being ID3 (Quinlan 1986). ID3 generates decision trees for nominal valued data-records. The resulting tree gives a representation that can be used to classify new instances. One of the derivatives of ID3 is NPPA (Talmon, 1986) which generates binary decision trees for real-valued data. Both ID3 and NPPA use information gain as the criterion to find the best partition. In this paper we will use NPPA as our standard induction algorithm.

TDIDT techniques originally are developed as tools for automatic knowledge acquisition. They are also used as a tool for data-analysis in research and industry, for example Cosemans & Samyn (1990) used induction to identify the most significant parameters in several industrial production processes. They showed that induction can achieve good practical results, where statistical techniques are very time-consuming, and sometimes unable to identify all the determining parameters.

---

* Now at: SMR, Jacob van Lennepkade 334 m, 1053 NJ Amsterdam, The Netherlands

It has been reported (e.g. Shapiro, 1987; Nunez, 1988; Tan & Schlimmer, 1989; Bergadano et. al., 1990) that the applicability of induction is improved by using background knowledge. Particularly, it results in more realistic decision trees that provide appropriate answers to specific questions. In this paper we discuss one application of background knowledge in induction: optimization of production costs in industrial environments. The structure of this text is as follows: section 2 introduces the use of background knowledge as an instrument to improve induction and survey existing literature on this topic. In section 3 we introduce cost optimization as one example of the use of background knowledge in induction. In section 4 an application of this algorithm is discussed. We end with conclusions and plans for further research.

2. Context

Let us start with an example of a company, mass-producing some product in a batch production process. Two process parameters, the percentage P1 of some chemical element in a raw material and the temperature P2 in an oven are measured. Both these values can be controlled. At the end of the process the products are labeled relative to their quality. They are either saleable or unsaleable. The unsaleable product is considered to be waste, and has no economic value. The relation between the parameter-values and the labels is unknown.

Given this situation, different departments in this company typically require answers to different questions. For example:

(a) the research department: what are the the optimal setpoints for P1 and P2 from the point of view of quality control?
(b) the technical department: how can we diagnose a production failure as soon as possible, if we know the values of P1 and P2?
(c) the technical department again: and how can we diagnose a production failure as cheap as possible, including the costs of measuring P1 and P2?
(d) the production department: we have some heuristics about how P1 and P2 determine the quality label of our product. If we assume these heuristics to be correct, what additional knowledge can we derive from a dataset?
(e) the financial department: what setpoints for P1 and P2 should be used to produce with the lowest cost price and what is this cost price?

An important point to note is that in establishing this relationships no one analysis is sufficient for all requirements. Induction may answer the questions (a) and (b) (e.g. Cosemans & Samyn, 1990). It establishes relationships between input and output, e.g. between attributes and (quality) labels. By using background knowledge to bias induction, the questions (c) and (d) can be answered as showed by Nunez (1988) and Bergadano et. al. (1990) respectively. As we will show in this paper, by using background knowledge on production costs in induction, we can answer questions such as (e).

TDIDT programs search in the space of all possible decision trees that describe a given dataset. An algorithm chooses one specific tree out of this set. Utgoff (1986) shows that several factors influence the selection of the decision tree, e.g. the formalism or language in which hypothesis are described and the problem space considered. All influencing factors constitute the total bias of induction. These biasing factors can't be avoided, but they can be influenced. It is well known that information gain has a specific bias: it favours attributes with many different values above attributes with only a few different values. Criteria that are closely related to information gain, such as covariability (Cosemans & Samyn, 1990) or geometric distance (e.g. Spangler et. al., 1989) select trees that may be close to the one that is found using information gain. We must use the appropriate bias to be able to answer specific questions.

The decision trees that are constructed using information gain are very efficient, but they lack logical structure for experts (Arbab & Michie, 1985). Shapiro (1987) and Buntine & Stirling (1988) show how expert background knowledge can be used to bias the induction, producing trees that are easier to understand for humans. These methods, however, need the presence of an expert during the process of induction, and may be time consuming. Nunez (1988) automated the use of knowledge on diagnosis cost in the induction, and showed that by using this knowledge the constructed trees are easier to understand. We think, in accordance with the other authors mentioned, that the use of background knowledge in induction may enable us to answer the questions mentioned above and that it may produce trees with a structure that is close(r) to the way an expert reasons.

3. Cost Optimization

3.1 Why Cost Optimization

Several authors report the application of induction to data from industrial processes (e.g. Cosemans & Samyn, 1990; Bergadano et. al., 1990). Most examples are concerned with quality control. As we showed in section 2 however, not all questions that may be asked in industry can yet be answered by means of induction. We use induction to answer question (e) (the one that asks for the cheapest products). The reason for choosing to deal with this particular problem is threefold. Firstly, this question may be interesting for industry. Identifying the optimal setpoints from a cost price point of view enables industry to optimize profit. Second, background knowledge on cost optimization (in the way we use it) can be represented formally. Such a representation may not be available for some other types of background knowledge. Finally, it is a clear example of how the use of background knowledge may influence the partition criterion.

Our main goal was to develop an algorithm that was able to find a tuning of the process delivering cheap end products, by means of induction. This implied three changes in the process of induction. Decision trees on real valued parameters, as produced by TDIDT algorithms, contain choice points of the form: either make $P_x < p$, or $P_x > p$. Most process controllers work differently. A process set-point can be set, being the mean value to be reached in the process, and the controller has a specific accuracy, being the (standard) deviation of the values in the process. We adopt this process control method of representing choice points. In addition the partition criteria considers cost gain instead of information gain, and the produced tree contains an optimal path.

3.2 Assumptions

In accordance with other induction algorithms we assume that:
• the dataset is representative for the problem space
• the noise is kept within reasonable norms
• the dataset contains no missing values
• all quality determining parameters are represented in the dataset

In addition we assume that:
• all parameters in the dataset can be controlled
• controlling a parameter consists of controlling both the mean value and standard deviation
• the costs involved in controlling a parameter P_i can be represented in two functions, μP_i and σP_i, for controlling the mean value and the standard deviation of P_i respectively. These functions, which represent the functional relation between costs and setpoints, form the background knowledge to be used in the partition criterion.

3.3 Cost Gain as a Partition Criterion

To start with, we need the following data for optimizing the costs:
• A set of parameters $P : \{ p_1, p_2, p_3, \ldots\}$.

- Mean and standard deviation cost functions $\mu P_i(\mu_o, \mu_n)$ and $\sigma P_i(\sigma_o, \sigma_n)$, one of them for each parameter P_i. These functions indicate how the cost of one single end-product is influenced when the mean value and standard deviation of parameter P_i are changed from μ_o to μ_n and σ_o to σ_n respectively.
- A set of data-records D : { d_1, d_2, d_3, ...}. Each data-record d_x consists of: {k_x, v_{1x}, v_{2x}, ..., l_x}, with k_x a key for d_x, v_{jx} being the value of parameter j for record d_x, and l_x being the label of record d_x. The label is a member of a set L : { saleable, unsaleable }. An unsaleable product represents waste.
- The initial costs of one individual product IC, as it is derived for the entire dataset.

The result of the algorithm should be a cost decision tree (CDT). In a CDT a path from the root to a leaf node represents a tuning of the production process (In this paper, for ease of understanding only the tuning of the most optimal path is shown). Such a tuning consists of a set of parameter/setpoint/accuracy combinations. Also attached to each tuning is a cost price C_i of a saleable product.

Our aim is to minimize C_i. Let us first calculate C_i for a certain node i. We assume that this node contains x saleable and y unsaleable products. Then C_i is defined as follows:

(1) $$C_i = \frac{(x+y) * (IC + C_c)}{x},$$

where C_c are the accumulated control costs of each node from the root of the tree down to the current node. To calculate C_c we have to calculate the control costs generated by a branching node, C_{ci}. We assumed before that there are costs related to controlling the mean value and standard deviation of a parameter. If P_i is a parameter, then $\mu P_i(\mu_o,\mu_n)$ is the function that gives the total costs of controlling P_i from the old setpoint μ_o to the new setpoint μ_n. Accordingly, $\sigma P_i(\sigma_o,\sigma_n)$ gives the costs resulting from changing the control accuracy from σ_o to σ_n. The control costs in a node i can now be expressed as:

(2) $$C_{ci} = \mu P_i(\mu_o,\mu_n) + \sigma P_i(\sigma_o,\sigma_n)$$

Our aim, minimizing the unit product costs for a saleable product can be reformulated as maximizing the cost gain. If C_i is the cost at branching node i, by partitioning this node for $P_x = p$, we can calculate the new cost price for the child nodes: C_{left} and C_{right}. The costgain then becomes:

(3) $$CG\ (P_x = p) = C_i - \text{Min}\ (C_{left}, C_{right})$$

Our algorithm now should find a parameter P_x and related value p such that CG is maximized.

Once a CDT has been derived, we have to find the optimal leaf. This of course is very simple: the path from the root to the leaf with the minimal cost price indicates the most optimal group of setpoints. In the branching points the setpoints are available, both as border values (as is the case in ID3/NPPA) and as mean and sigma values.

3.4 An Example

To illustrate the algorithm we generated a simplified two-dimensional space belonging to the example of section 2. Both the variables P1 and P2 take values from 0 to 9. The labels were attached: saleable (+) and unsaleable (-). The initial cost of a product, IC, was set to 11. This resulted in a cost price of a good product for the entire dataset, C_i to be 14.86. A representation of the instance space is presented as a 10x10 grid (Figure 1). Figure 1a shows the areas that were recognized by NPPA. The region with the minimal cost price (region 1) has a cost price of 13.52, about 9 % cheaper than the cost price of the initial dataset.

Table 1 shows the cost functions for mean and standard deviation. The functions may stand for a process where the costs are inversely proportional to the standard deviation of the parameter values, and quadratic/linearly dependent on the mean value of the parameters (in real-world situations these functions may be far more complicated!). The partitioning MICO finds for this situation is shown in Figure 1b. A part of the related CDT can be found in Figure 2. MICO finds a partitioning in which only one unsaleable product is represented. The final cost price is 11.88, about 20 % cheaper than the original result, and 11% cheaper than the solution NPPA comes up with.

By varying cost functions and fixed product-costs, which are the background knowledge, the partitioning can be influenced. As might be expected induction biased this way follows certain rules: when the initial costs IC of one produced entity are relatively high, MICO seeks for an area with as little 'bad' products as possible (as in region 1 of Figure 1b). Lower IC leads to less partitioning: more 'bad' products are tolerated (as in region 1 of Figure 1c). This however also decreases the realative cost price reduction achieved by the partitioning (5% in Figure 1c). This may be illustrated by the following example: when producing airplanes (high IC), you want to be very sure that at the end of your process, a good plane is delivered, and so you are prepared to invest in process control. When you are producing chewing gum (low IC) you can afford to throw away some of your end products. The initial costs IC thus weights the importance of the unsaleable instances related to the saleable. Though the same effect can be reached in other ways, e.g. by duplicating the saleable and unsaleable instances so that their ratio is according to some weight, using the initial costs IC seems a better solution.

The cost functions weigh the preference for controlling certain parameters. If we vary the cost functions, we influence these weights. In Figure 1d we can see what happens when the cost functions for parameter P1 of Figure 1b are changed. The functions $\mu P1$ and $\sigma P1$ are changed so that the effect of changing the setpoint and the standard deviation of P1 is 10 and 5 times as influential as in Figure 1b.

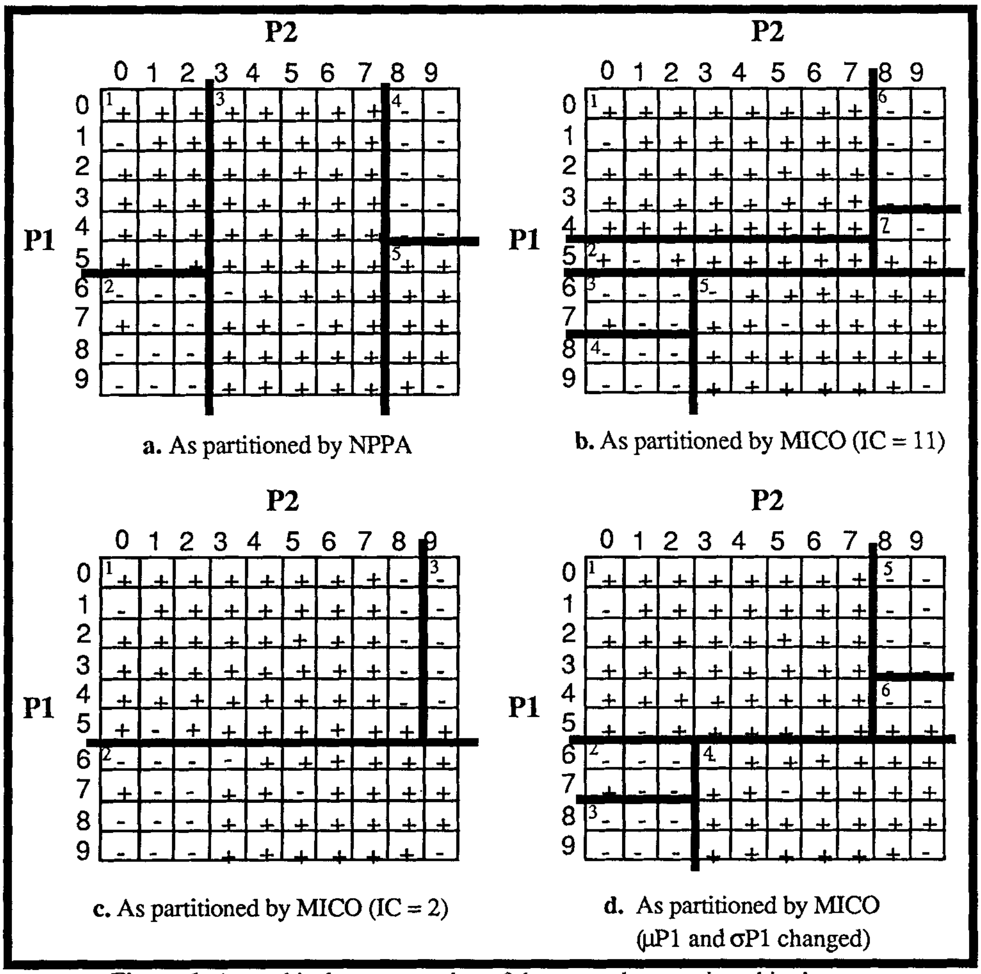
P2
P1
a. As partitioned by NPPA
b. As partitioned by MICO (IC = 11)
c. As partitioned by MICO (IC = 2)
d. As partitioned by MICO (μP1 and σP1 changed)

Figure 1. A graphical representation of the examples mentioned in the text

3.5 Information Gain versus Cost Gain

The performance of information gain and cost gain in some of the above examples can be seen in table 2. We see that the results MICO comes up with are at least as beneficial as the results of NPPA. This holds for all examples we have considered. The partitioning of the instance space as made by NPPA may differ significantly from the one found by MICO (see Figure 1).

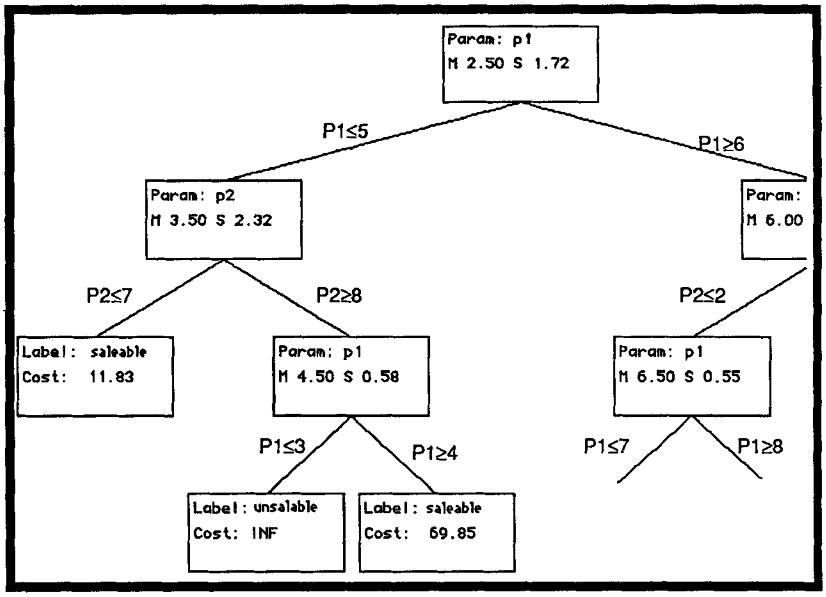

Figure 2. Part of the CDT belonging to Figure 1d (M = Mean value, S = standard deviation). In the branching nodes, the mean value and standard deviation that are needed to reach the descendant node with the minimum Ci are shown. The setpoints for the other branch are available in the program, and can be accessed through the interface

Function type Parameter	Mean value cost function ● μP_x (o_μ, n_μ)	Standard deviation cost function σP_x (o_σ, n_σ)
P1	$0.01 (n_\mu^2 - o_\mu^2)$	$1/n_\sigma - 1/o_\sigma$
P2	$0.3 (n_\mu - o_\mu)$	$10 (1/n_\sigma - 1/o_\sigma)$

Table 1. The mean and standard deviation functions of the example

CDT's produced according to MICO differ from the decision trees as produced by other TDIDT algorithms on several points. The first difference is that a CDT reflects a search space, where an optimal path can be found. The optimal path indicates the optimal tuning of the process. A decision tree on the other hand gives a classification of all instances in the dataset; it makes no sense to talk about an optimum here. When a CDT is used as a classification tree, it behaves as a probabilistic classifier (Quinlan 1987).

Another difference between CDT's and classical decision trees is the information provided by the branching points. In the branching points of NPPA-like decision trees conditions are stored in the form of an inequality on attribute values. In CDT's the branching points store a setpoint and a deviation.

Used version	Initial cost price	Cost price of the NPPA solution	NPPA versus initial	Cost price of the MICO solution	MICO versus initial
IC = 11 (fig 1b)	14.86	13.52	9.0%	11.88	20.0%
IC = 2 (fig 1c)	2.70	4.14	-53.3%	2.54	5.9%
µP1 10 times higher	14.86	12.41	16.49%	11.83	20.4%

Table 2. A comparison of the NPPA and MICO in the examples of Figure 1

3.6 N-class Labeling

In section 3.4 we presented all requirements for the 2-class datasets, where one of the classes was waste. We will now sketch how the algorithm can be generalized to N-class situations and to situations where all products have some value, e.g. as scrap, or as a second class product. For the 2-class problems with one of the classes being waste, we can concentrate on costs. When we want to optimize the problem for more classes, with different values, costs are not sufficient. We have to find another way to compare two situations. This is solved by the introduction of a function representing the value of a product:

- Income-function $I(K_V, l)$; given a class-distribution K_V and a specific label l, this function gives the total income for products of that label.

The partitioning criterion now becomes the maximization of the average benefits of a product. As argued before this does not change the basic algorithm. An extra calculation however is needed. Instead of minimizing the costs of a good product, we are now interested in maximizing the profit of a product. We now have to maximize the benefit gain:

(4) $BG\ (P_x = p) = \mathrm{Max}\ (B_{left}, B_{right}) - B_i$

(with B_i the benefit in the parent node, and B_{left} and B_{right} the benefit for the child nodes). If we find a distribution of $[x_1, x_2, \ldots x_n]$ in a leaf node, the average benefit gain BG on a product can be calculated with the formula:

$$(5) \quad BG = \frac{\left(\sum_{i=1}^{n} x_i * I([x_1, x_2 .. x_n], x_i)\right) - \left(\sum_{i=1}^{n} x_i\right) * (IC + C_c)}{\sum_{i=1}^{n} x_i} .$$

The benefit of one product thus is calculated by taking the difference between total benefit and total cost, divided by the number of products. This extension is straightforward and introduces no new concepts. In many industrial cases 2-class labeling is sufficient.

4. Empirical Results

We have applied the algorithm to data gathered in an industrial company with a mass-production process. We received a small dataset from the factory, containing 52 records of 10 parameters. All parameters are real-valued. The initial costs, valid for the entire dataset, are 170. Given the distribution between saleable and unsaleable products in the dataset of 29 and 23 respectively, this resulted in a cost-price for the saleable products of 304.83. Good cost functions were not directly available. We therefore constructed cost functions on the basis of a factory-model we received from the company and on the basis of expert interviews.

We applied both NPPA and MICO on this dataset. The results are shown in table 3. Using the NPPA algorithm, we constructed a decision tree. The derived ruleset consists of 7 rules. The cheapest solution NPPA came up with was 174.98, which was valid for 6 examples. All examples in this rule were classified correctly. The CDT produced by MICO was larger, containing 10 rules. It contained a partition on a parameter that was not identified as quality determining by NPPA. Most of the rules were valid for a few examples, and thus were not very reliable. The cheapest set of setpoints reduced the total cost price from 304.83 to 171.46 (this rule incorporated 10 examples).

Feature	Initial	NPPA	MICO
Number of rules	-	7	10
Lowest cost	304.83	174.98	171.46
Labeling accuracy (over all)	55%	86.5%	78 %
Labeling accuracy (best)	55%	100 %	100 %

Table 3. Comparison between the initial dataset, NPPA and MICO on real world data

5. Conclusions

5.1. MICO

MICO is an algorithm that performs well in minimizing the costs and optimizing the benefits of a production process. The solutions it finds are at least as good as the ones NPPA finds, at least in two points: they represent production tunings for products with a low cost price, and they are understandable for operators of the production process, because they make choices that make sense.

If we try to understand how the information gain criterion works for these cases we see a reduction in uncertainty on the class label. In NPPA partitioning continues until no significant information gain can be reached. The way our cost optimizing algorithm works is very similar. MICO does not make use of information gain as its partitioning principle. The pragmatic condition of information gain, which is used in NPPA, is replaced by a well-defined one in MICO: the cost functions exactly quantify this criteria.

Algorithms like NPPA control the partitioning of the dataset by calculating some statistical feature of the information gain, e.g. the significance of the reached information gain. These control strategies are highly pragmatic. In MICO the control strategy is meaningful: as long as the cost price decreases, the process continues. The way the cost functions for standard deviation value of a parameter are formulated should guarantee that a partitioning up to individual records should not happen. Another view here is that the cost functions and the initial IC are the parameters controlling the partitioning. In other words this method of biasing induction uses a well-defined meaning in the real-world instead of a pragmatic measure.

5.2 Induction Biased by Background Knowledge

We have presented an algorithm that uses one specific form of background knowledge. Several other ways of using background knowledge however must be distinguished. The cost example uses background knowledge to partition the dataset.

One way of biasing induction is by manipulating the hypothesis space. The approach Utgoff (1986) took with STABB enlarges the space of hypothesis by weakening the bias. In TDIDT this use of weakening the bias can be found in Spangler et. al. (1989). Our approach of biasing induction doesn't weaken the bias, but shifts the bias to another part of the hypothesis space, thereby making it possible for other questions to be answered.

5.3 Acquiring Cost Functions

Applying MICO clearly creates a new problem: the acquisition of cost functions. As we explained in section 3.2 cost functions relate a change in parameter tuning with a change in the production costs of one product. In the industrial environments we visited, there were no such cost functions available, at least not formulated explicitly. However, there are very clear heuristics about them. For example, when talking about controlling a temperature gradient, a typical statement was: ‘controlling this is very expensive, maybe it is cheaper to try influencing the temperature’.

This type of remarks was made very frequently. From that, we searched for formalization of the weights of the parameters. We found it to be hard for operators and engineers in industry to formulate the cost functions we needed in MICO. Our approach in acquiring the functions was to analyse the process on the basis of experts-interviews and to extract cost functions from this analysis. This resulted in cost functions of a type close to the heuristics of process experts.

6. Further research

We will continue developing the MICO algorithm. An interesting question is the effect of the dependency of parameters that appear in the cost functions. More experience is needed in constructing cost functions for real factory situations and we would like to compare MICO more extensively with existing methods for cost optimization as defined in the cost price analysis, and in engineering sciences. We should also search for better ways to represent cost functions. As we explained earlier, we adopted a representation for cost functions which is as close as possible to the heuristics that process experts use. This representation differs from the one used in business economics, where costs are considered to have a fixed and variable part. In this form, cost functions may be formulated more naturally, and better explicit formulation may take place. Other topics of research will be the significance of the cost optimizations as derived by MICO, and the possibilities for post-processing the produced trees (as in Payne & Meisel, 1977). This may enable us to further improve the solutions, of both optimality of costs and classification.

The main subject of our project is to develop appropriate strategies for applying background knowledge in induction. We find that this may enlarge the possibilities for using inductive learning techniques in practical situations, both in knowledge acquisition and in data analysis applications.

Acknowledgements

I am indebted to Jan Talmon for making his NPPA version available for me, and to Hilde Thijsen for building the generic interface for my induction programs. I thank Jan Paredis, Jan Talmon and Gerry Kelleher for the useful suggestions and comments on earlier versions of this paper.

References

Arbab & Michie 1985 **Arbab, B., and Michie, D.**, Generating Rules from Examples, in: Proceedings of IJCAI-85, pp. 631-633, Los Angeles , CA, 1985

Bergadano et. al. 1990 **Bergadano, F., Giordana, A. and Saitta, L.**, Biasing Induction by Using a Domain Theory: An Experimental Evaluation, in: Proceedings of ECAI-90, pp. 84-89, 1990

Buntine & Stirling 1988 **Buntine, W., and Stirling, D.**, Interactive Induction, in: Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications, pp. 320-326, San Diego, 1988

Cosemans & Samyn 1990 **Cosemans, G., and Samyn, J.**, Inductive Analysis of Datasets, paper presented at BeneLearn-90, Leuven 1990

Nunez 1988 **Nunez, M.**, Economic Induction: A Case Study, in:, Proceedings of EWSL-88, pp. 139-146, Glasgow, Scotland 1988

Payne & Meisel 1977 **Payne, H.J., and Meisel, W.S.**, An Algorithm for Constructing Optimal Binary Decision Trees, in: IEEE Transactions on computers, vol 26, pp. 905-916, 1977

Quinlan 1986 **Quinlan, J.R.**, Induction of Decision Trees, in: Machine learning, Vol. 1, No 1, pp. 81-106, 1986

Quinlan 1987 **Quinlan, J.R.**, Decision Trees as Probabilistic Classifiers, in: Proceedings of the 4th International Machine Learning Workshop, pp. 31-37, University of California, Irvine, 1987

Shapiro 1987 **Shapiro, A.**,Structured Induction in Expert Systems, Turing Institute, Glasgow, Scotland, 1987

Spangler et. al. 1989 **Spangler, S., Fayyad, U.M., and Uthurusamy, R.**, Induction of Decision Trees from Inconclusive Data, in: Proceedings of the 6th International Workshop on Machine Learning, pp. 146-150, Ithaca, NY, 1989

Talmon 1986 **Talmon, J.L.**, A Multiclass Nonparametric Partitioning Algorithm, in: Pattern recognition letters Vol. 4, pg 31-38, 1986

Tan & Schlimmer 1989 **Tan, M. and Schlimmer, J.C.**, Cost-Sensitive Concept Learning of Sensor Use in Approach and Recognition,in: Proceedings of the 6th International Workshop on Machine Learning, pp. 392-395, Ithaca, NY, 1989

Utgoff 1986 **Utgoff, P.E.**, Shift of Bias for Inductive Concept Learning, in: Machine Learning, An Artificial Intelligence Approach, Volume II, R.S. Michalski,, J.G. Carbonell and T.M. Mitchell (eds), Morgan Kaufman, Los Altos, California 1986