ASSISTANT 86: A Knowledge-Elicitation Tool for Sophisticated Users.

Conference Paper · January 1987

Source: DBLP

CITATIONS
443

READS
786

3 authors, including:

Bojan Cestnik
Jožef Stefan Institute
72 PUBLICATIONS **1,965** CITATIONS

SEE PROFILE

Ivan Bratko
University of Ljubljana
210 PUBLICATIONS **4,638** CITATIONS

SEE PROFILE

ASSISTANT 86: A Knowledge-Elicitation Tool for Sophisticated Users

Bojan Cestnik, Igor Kononenko & Ivan Bratko

ABSTRACT. Learning of decision rules from examples enables automating of the knowledge acquisition for expert systems. In this paper we describe the system ASSISTANT 86 for induction of decision trees. It was developed from Quinlan's ID3. We improved the basic algorithm to handle incomplete and noisy data, continuous and multivalued attributes, and incomplete domains, i.e. domains where attributes do not suffice for exact classification. ASSISTANT 86 was tested on several medical domains where it reached the classification accuracy of medical experts. It was also used for constructing an expert system that controls a certain industrial process. ASSISTANT 86 is implemented on IBM-PC and has numerous facilities that support its interactive use in the rule elicitation process.

1. INTRODUCTION

Induction learning is one approach to learning from examples [for example, Quinlan, 1986, Patterson and Niblett, 1983, Kononenko, 1985, Breiman et al, 1984]. In this paper we present a system for induction of decision trees from examples called ASSISTANT 86 and results of experiments in 4 medical domains.

ASSISTANT 86 is a member of the family of learning systems also called the TDIDT (Top Down Induction of Decision Trees) family [Quinlan, 1986]. A common characteristic of this family (shown in fig. 1) is the knowledge representation in the form of a decision tree. An example decision tree for the prognosis in hepatitis is shown in fig. 2. ASSISTANT 86 is an implementation of ASSISTANT [Kononenko, 1985] on IBM-PC under the MS-DOS operating system. All improvements over ID3 [Quinlan, 1979] developed in ASSISTANT are also included in ASSISTANT 86, although some of them have been slightly modified.

The basic algorithm of ID3 for constructing a decision tree is as follows:

```
if all the learning examples belong to a single class
  then terminate with a leaf labelled with that class
else
  begin
    on the basis of the learning set choose the most
      informative attribute (using the entropy measure) for
      the root of the tree and partition the learning set
      into subsets according to the values of the selected
      attribute;
    for each value of the attribute do
      recursively construct a subtree with the corresponding
        subset of examples
  end
```

We improved the basic algorithm in several ways in order to handle incomplete and noisy data, continuous and multivalued attributes, and *incomplete domains*, i.e. domains where attributes do not suffice for exact classification. More precisely, a domain is said to be *incomplete* if its attribute set is insufficient to distinguish between all objects that belong to different classes. In the following sections we present these improvements and the flexibility of ASSISTANT 86 which makes it powerful knowledge elicitation tool for sophisticated users.

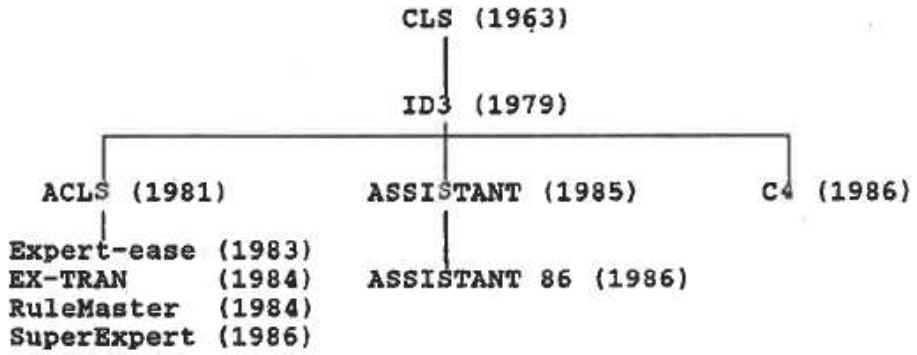

Fig. 1. The TDIDT (Top-down Induction of Decision Trees) family tree. CLS is described in [Hunt et al, 1966], ID3 in [Quinlan, 1979], C4 in [Quinlan et al, 1986], ACLS, Expert-ease, EX-TRAN, RuleMaster and SuperExpert are products of ITL, Glasgow.

Many of the improvements have already been described elsewhere [Kononenko, 1985, Bratko and Kononenko, 1986]. They are included here in section 2 for the sake of completeness. In this paper the emphasis is on section 3 where we describe the pruning of decision trees as implemented in ASSISTANT 86. The discussion in

the sequel assumes the reader's general familiarity with the basic concepts and terminology of ID3-based tree induction [for example, Bratko and Kononenko, 1986, Quinlan, 1986].

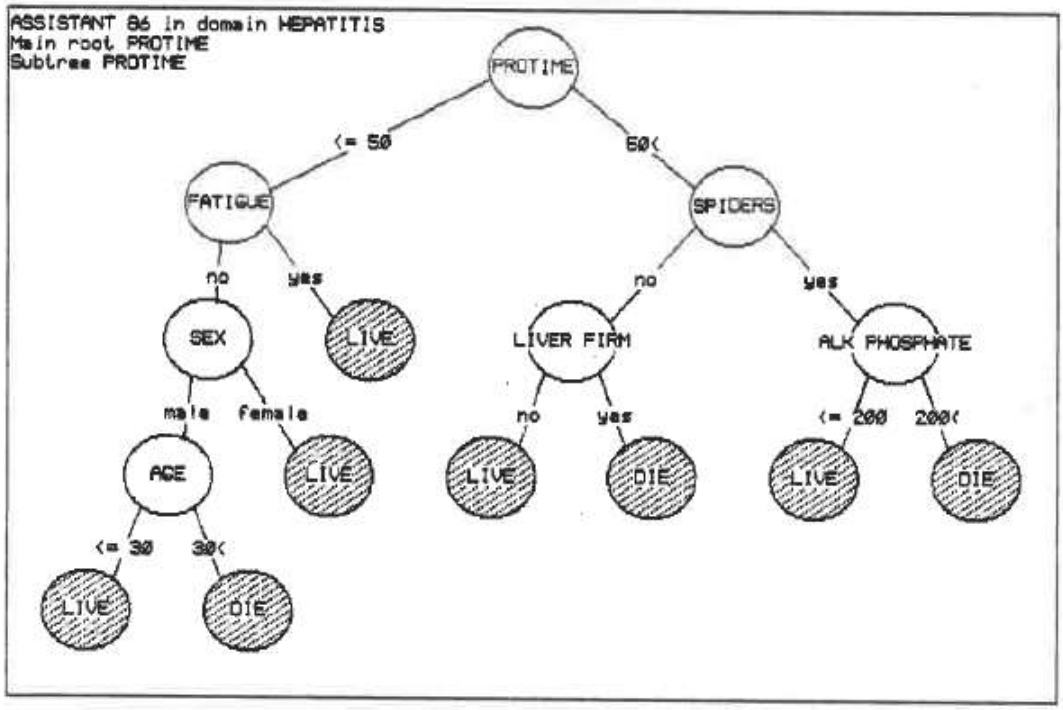

Fig. 2. A decision tree from the domain of hepatitis as constructed and displayed by ASSISTANT 86.

2. THE DIFFERENCES BETWEEN ID3 AND ASSISTANT 86

2.1. Decision tree construction without iterations

A problem of computational efficiency arises when the number of learning examples is too large to fit into the memory. To improve the efficiency ID3 constructs a decision tree by using the 'windowing' mechanism:

```
Randomly select a 'window set' from learning examples;
repeat
  Generate a decision tree for the selected set;
  Test the tree over the rest of examples;
  Add the mis-classified examples to the 'window set'
until there are no mis-classified examples
```

All probabilities used to estimate the informativity of attributes in a decision tree are approximated with relative

frequencies taken from the learning set. When the learning set is larger the approximations tend to be better. The best results are obtained when the program learns from the whole set.

Therefore, as an alternative solution to 'windowing' we can keep all the examples on disk, while in the memory we have only the statistics. This is reasonable, since the operation of scanning the examples consumes relatively small time portion in comparison with the search for the best attribute. In our experiments running the program with examples in the memory was only twice as fast as the one with examples on disk.

2.2. Incompletely specified learning instances

In real domains it often happens that for some instances the value of a certain attribute is missing. In such cases in ID3 a special value 'unknown' must be added to the value set of that attribute. Introducing the value 'unknown' causes decision trees to increase in size. Also we have to provide an additional set of learning examples in order to cover all the branches in the tree and to avoid additional *NULL leaves* (i.e. leaves that are not covered by any learning example). If a testing instance falls into the NULL leaf the tree is not able to classify it.

Therefore, ASSISTANT 86 handles missing values differently. All possible values are assigned to an attribute with a missing value, each value weighted with a conditional probability. If the given instance belongs to a class C, then the probability of an attribute A having a value V is:

$$p(V|C) = \frac{p(V\&C)}{p(C)}$$

Probabilities $p(V\&C)$ and $p(C)$ are approximated with relative frequencies from the set of instances in a current node of the decision tree.

In classification an object with a missing value of an attribute A is classified by following the branches that correspond to all possible values of A, weighted by prior probabilities of the corresponding values of A.

2.3. Classification in combination with Bayesian principle

If a problem domain is incomplete, i.e. if a set of attributes is not sufficient for exact classification of all instances, then a

decision tree is necessarily unreliable. The situation becomes even worse if we have to learn from a small number of examples. The confidence in a decision tree may depend on the number of instances in the leaves of the tree. For example, if the learning examples are drawn statistically and may contain errors and if there is only one instance in a leaf, then the classification in this leaf is unreliable. However, it is also possible that a leaf is empty - NULL leaf (with no corresponding learning example). If a testing instance falls into a NULL leaf, the classification with the tree is impossible.

To overcome this problem, in ASSISTANT 86 the Bayesian classification principle is used whenever a testing instance falls into a NULL leaf. Conditional probabilities for each class are computed considering only the attributes along the branch from the root to the NULL leaf. The leaf is labelled with the most probable class.

2.4. Binarisation of attributes

The main purpose of the binarisation is to normalize the informativity of all the attributes with respect to the number of values. As a result we get smaller and more accurate decision trees. Binarisation prevents over-splitting of the learning set. Thus the attribute selection becomes more reliable even in lower levels of the tree where the number of examples is small.

Although binary construction is computationally less efficient and sometimes generates trees that are not well structured, they tend to be significantly smaller and thus easier to use and understand.

Continuous attributes can have any real value within a fixed interval (as for example temperature, blood pressure, etc.). In order to use such attributes in a decision tree we have to group the values into subintervals. ASSISTANT 86 makes continuous attributes binary by selecting a boundary value that divides the interval of an attribute's possible values into two subintervals, that maximize the attribute's informativity.

To overcome the overestimation of multivalued attributes all the attributes (not just continuous ones) in a decision tree are made binary (for a discussion on informativity overestimates of multivalued attributes see, for example, Bratko and Kononenko, 1986). This is done by grouping the values into two subsets that maximize attribute's informativity. If some of the attribute values have no corresponding instance they form a third subset,

leading to the NULL leaf. The reason for the appearance of this third branch from a node lies either in the weakness of the learning set (small number of examples) or because such values are logically impossible.

Binarisation of discrete attributes is a combinatorial problem. Considering all possible splits is reasonable only if the number of attribute values is small enough. The computation time grows exponentially in the number of values. ASSISTANT 86 uses a heuristic algorithm for binarisation of attributes with more than 4 values (if the number of attribute values is less or equal 4, it does exhaustive search). The algorithm for the binarisation of a discrete attribute A is as follows:

```
Split the values of A into two subsets S1 and S2 such that
  S1 contains only one value V1. V1 is selected so as to
   maximize the informativity of A binarised with respect to
   S1 and S2;
repeat
  for every value V1 in set S2 do
    begin
      delete V1 from set S2 and put it in S1;
      if the informativity of A binarised did not increase
        then move V1 back from S1 to S2
    end
until no changes in S1 and S2 are made
```

2.5. Filtering of learning examples

If an attribute set is not sufficient for exact classification of all instances, then there are some leaves in the decision tree with more than one corresponding class (*SEARCH leaves*). The reason for this situation might also be noise, present in the learning examples. To improve the classification accuracy and decrease the complexity of a tree, especially in noisy domains, one idea is to construct a tree, taking into account only 'good' training instances. The algorithm for the selection of 'good' instances in ASSISTANT 86 is as follows:

```
Build a statistics over all training instances;
for all training instances do
  if an instance is correctly classified with Bayesian
    principle then select it for learning
  else reject it
```

Decision trees constructed by using the filtering option are typically smaller and more accurate. They do not contain noise

but sometimes they exclude 'good' exceptions. Two other methods, pre-pruning and post-pruning, discussed in the next section, produce almost the same effect concerning the size and accuracy but the exceptions are incorporated in the decision tree. This is the main reason why these two methods are preferred to filtering.

3. TREE PRUNING

In incomplete domains classification cannot be exact. But the basic induction algorithm tries to construct an exact decision tree. So at the lower levels of a tree it tries to split a subset of learning examples rather randomly by choosing attributes that have no classification power. Decision tree pruning is the mechanism that aspires to prevent the construction of subtrees if they are unreliable. It can serve as a very useful feature for reducing the size of a decision tree and improving its classification accuracy. It is also clear that it is less useful in complete domains with training sets that are large enough.

In other systems different approaches to tree pruning are used [Kononenko, 1985, Quinlan et al, 1986, Quinlan, 1986].

To alleviate difficulties with incomplete set of attributes and noise in the learning data, two types of pruning are available in ASSISTANT 86:

- *pre-pruning* - decide whether to stop or not at a certain node while constructing a decision tree.
- *post-pruning* - after a decision tree is constructed estimate the classification errors in the nodes and then decide whether to prune certain subtrees or not.

3.1. Pre-pruning

Pre-pruning takes place while constructing the tree. It is used to stop branching in a node where the classification bias of a learning set becomes unreliable. To estimate the reliability we use three parameters which are computed in each node of the decision tree:

- *attribute suitability* $\frac{Inf(Am)}{E} * Wi$ (1)

where *Am* is the most informative attribute,

E is the entropy of the learning set in a current node,
Wi is the weight of the learning set in the node, and
Inf(Am) is the informativity of *Am*, i.e. the reduction of the entropy of the learning set after applying *Am*.

- *class frequency* $p(Cm) * 100\ (\%)$ (2)

where $p(Cm)$ is the probability of the majority class in the learning set of the node.

- *node weight* $\frac{Wi}{Wr} * 100\ (\%)$ (3)

where *Wi* is the weight of the learning set in the node and *Wr* is the weight of the learning set in the root node.

The first parameter (1) can be split into two parts. In the first part we estimate the *attribute's success* (the share of information provided by the attribute *Am*):

$$suc(Am) = \frac{Inf(Am)}{E} \quad (4)$$

For (4) the following holds:

$$0 \leq suc(Am) \leq 1 \quad (5)$$

If the value of *suc(Am)* is closer to 0 it shows that the success of the attribute *Am* is rather poor. On the other hand, if the value of *suc(Am)* is closer to 1 the attribute *Am* gains a considerable share of the whole information from the learning set.

The weight *Wi* of the learning set is related to the confidence into success of *Am*. If we multiply *suc(Am)* with *Wi* we get a heuristic estimate called the attribute suitability (1). When the value of (1) falls under a certain threshold, supplied by the user before the tree construction, ASSISTANT 86 will prune the decision tree at the current node (insert a LEAF and terminate branching). The threshold is set by the user as a domain parameter. If we do not want to prune (for example, in complete domains) the value of the threshold should be 0. In our

experiments (see section 4) we used the threshold value 3, which turned out to be good in some real world domains. Of course, depending on the domain it is possible to achieve better classification accuracy by varying the threshold value.

The second parameter (2) shows the share of the majority class in the learning set. If, in a certain node, it grows over a specified threshold, the tree expansion in this node is stopped. For example, if we specify the threshold value of 80%, then whenever the share of the majority class grows over 80% the leaf will be inserted and the remaining 20% will be considered as hard-to-distinguish minority. In our experiments in section 4 we excluded the tree pruning due to this criterion by setting this threshold to 100%.

The point of the third parameter (3) is to stop over-splitting of the learning set. As mentioned earlier, if relative frequencies are taken from a small learning set they are very unreliable. To avoid this problem we can stop further construction of the tree whenever the learning set becomes too small. If we specify the threshold value of 5% it means that whenever there are less than 5% of all weighted learning examples in the current node all subtrees at that node will be pruned. In our experiments in section 4 the threshold value was 4%.

3.2. Post-pruning

Post-pruning mechanism is used to prune the decision tree after it is already created. The method is described in detail in [Niblett and Bratko, 1986]. The general algorithm is defined by procedure *prunetree(Tree, PrunedTree, Error)* below, where *Tree* is the tree to be pruned, *PrunedTree* is the result of pruning and *Error* is the expected classification error of *PrunedTree*.

```
procedure prunetree(Tree; PrunedTree; Error);
begin
  set N to be the root of Tree;
  calculate the static error Es of N;
  if N is a leaf then
    PrunedTree = N and Error = Es
  else
   begin
      for each Subtree_i rooted at N do
        prunetree(Subtree_i , PrunedSubtree_i , E_i );
      calculate the dynamic error Ed of N;
      if Ed ≥ Es then {prune subtrees}
        PrunedTree = N and Error = Es
      else
        PrunedTree = the tree with the root N and subtrees
          PrunedSubtree_i , and Error = Ed
    end
end
```

For every node of the tree we have two errors: *static error Es* and *dynamic error Ed*. The static error *Es* of a node *N* is computed by the formula [Niblett and Bratko, 1986]:

$$Es = (n - n_c + k - 1)/(n + k)$$

where n is the total number of examples in node N,
n_c is the number of examples in the majority class c of the node N, and
k is the number of classes of the domain.

The dynamic error *Ed* of a node *N* is computed considering the error estimates for the subtrees of *N* (in ASSISTANT 86 we have only two subtrees due to the binarisation of attributes):

$$Ed = (n_1 * E_1 + n_2 * E_2)/(n_1 + n_2)$$

where n_i is the number of examples in the i-th subtree and
E_i is the error estimate for the i-th subtree.

If node *N* is a leaf, then dynamic error *Ed* is equal to static error *Es*. The algorithm prunes subtrees of a certain node if dynamic error of that node is greater than or equal to static error.

4. EXPERIMENTAL RESULTS

Here we present results of applying ASSISTANT 86 to 4 medical domains (see TABLE 1) with special reference to tree pruning. In each domain we performed 6 experiments. Each time 70% of the original examples were randomly taken for training and the remaining 30% for testing. In [Breiman et al, 1984] it is noted that usually two thirds of the data are used for learning and one third for testing, although there is no theoretical support for this particular choice.

Domain	No.of Classes	No.of Attributes	Average Values/Attr	No.of Examples
Primary tumour	22	17	2.2	339
Lymphography	4	18	3.3	148
Hepatitis	2	19	3.6	155
Breast cancer	2	10	2.7	288

TABLE 1: Description of four medical domains.

	NO PRUNING		PRE-PRUNING	
Domain	No.of Leaves	Accuracy	No.of Leaves	Accuracy
Primary tumour	90	41%	41	44%
Lymphography	22	76%	13	76%
Hepatitis	11	80%	9	83%
Breast cancer	63	70%	8	77%

TABLE 2: Average complexity and accuracy of decision trees, generated with ASSISTANT 86 without tree pruning and with pre-pruning.

In our experiments the thresholds for pre-pruning parameters (see section 3) were set as follows: 3 for attribute suitability, 100% for class frequency and 4% for node weight. The average complexity of created decision trees is shown in TABLE 2. The results of the experiments are presented in TABLE 3.

Domain	BAYES	ASSISTANT 86 pre-pruning	ASSISTANT 86 post-pruning	Relative reduction
Primary tumour	47% (2.1)	44% (3.0)	34% (3.5)	78% (5.4)
Lymphography	79% (4.9)	76% (5.8)	76% (5.0)	24% (14.4)
Hepatitis	84% (3.3)	83% (3.9)	82% (3.6)	12% (10.0)
Breast cancer	78% (3.8)	77% (2.0)	78% (2.1)	69% (16.4)

TABLE 3: Classification accuracy of ASSISTANT 86 (with pre-pruning and post-pruning) compared to Bayesian principle. The last column is the *relative reduction* (the percentage of the nodes in a pre-pruned tree removed by post-pruning) of the tree size by post-pruning. The number in parenthesis is the standard deviation.

Domain	non-specialists	specialists	ASSISTANT 86
Primary tumour	32%	42%	44%
Lymphography	60% (estimate)	85% (estimate)	76%
Hepatitis	-	-	83%
Breast cancer	65%	65%	77%

TABLE 4: Classification accuracy of ASSISTANT 86 compared to medical doctors (non-specialists and specialists).

In TABLE 4 in the primary tumour and breast cancer domains the diagnostic precision of non-specialists and specialists is the average of four internists non-specialists and four oncologists specialists from the University Medical Centre, Ljubljana [Kononenko, 1985]. In lymphography the diagnostic precision of physicians is a specialist's estimate. For the hepatitis problem we have no data about the physicians' performance. Gail Gong has experimented with the same data with statistical methods [Diaconis and Efron, 1983] and has achieved 80% classification accuracy.

5. CONCLUSION

The tree pruning mechanisms enable the system to deal with incomplete and noisy domains. Generated rules are drastically

reduced in size while performance (the classification accuracy) remains approximately at the same level (see TABLE 2 and 3). Post-pruning mechanism developed by Niblett and Bratko (1986) (see section 3.2) seems to make rather pessimistic estimation about the information contained in learning data (it over-estimates the error rate of subtrees) so the pruning is occasionally too drastic and the classification accuracy decreases. From the results above it seems that post-pruning is too drastic when the number of learning examples per class per attribute is low. On the other hand, the resulting trees are smaller and therefore more understandable.

In recent years the main application field of artificial intelligence research is expert systems. The basic problem is assimilation and representation of expert's knowledge in an expert system. The traditional approach of knowledge acquisition by consulting experts of the application domain is time consuming and expensive. The alternative approach is the automatic knowledge acquisition by qualitative modelling [Bratko et al, 1986] and inductive learning from examples. The latter approach seems to be mature enough to be used in constructing expert systems in real world problems [Michalski and Chilausky, 1980, Quinlan et al, 1986].

Our experiments in 4 medical domains show that automatically generated rules achieve the same level of classification accuracy as human experts. The knowledge acquisition process is shortened to few weeks needed for the data preparation (for example, cleaning the data from the medical records of patients) and for experimenting with ASSISTANT 86. We have to select a proper set of parameters to achieve good classification accuracy and the structure of decision trees that is understandable by humans. Generated rules can be integrated in an expert system or can be used without the computer as well (listed on a paper) to help non-specialists in diagnosing new patients and to teach students of medicine.

ASSISTANT 86 was used for the construction of a knowledge base for a medical expert system [Roškar et al, 1987] and for the control of an industrial process [Lavrač et al, 1986]. The latter expert system is routinely used in the Iron & Steel Works Jesenice (Yugoslavia).

ASSISTANT 86 is implemented on IBM-PC. We put a great effort in the design of user interface in order to make the system widely available for use. Menu technique is used to provide a simple and user-friendly man-machine communication. Because of its

flexibility and efficiency the system enables interactive experimenting with the data of an adequately defined domain. ASSISTANT 86 also allows the user to force an attribute selection in nodes while constructing a tree. This is a very useful feature, especially for those who want to incorporate their own knowledge in a generated decision tree. These features make ASSISTANT 86 a powerful and easily accessible tool which can be used by experts for experimenting with data and generating decision rules.

ACKNOWLEDGMENTS

The data for lymphographic investigation, primary tumour and breast cancer recurrence were obtained from the University Medical Centre, Ljubljana. We would like to thank M.Zwitter and M.Soklič for providing the data and for interpreting the results. We thank G.Gong from Carnegie-Mellon University for providing the data for the hepatitis problem.

REFERENCES

Bratko, I., Mozetič, I., Lavrač, N. (1987), Automatic synthesis and compression of cardiological knowledge, Machine Intelligence 11 (eds. J.Hayes, D.Michie, J.Richards), Oxford University Press. Also in Expert Systems: Automating Knowledge Acquisition (AI Masters video handbook, D.Michie, I.Bratko), Addison-Wesley, 1986.

Bratko, I., Kononenko, I. (1986), Learning diagnostic rules from incomplete and noisy data, AI Methods in Statistics, UNICOM Seminar, London, December 1986.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984), Classification and Regression Trees, Belmont, California: Wadsworth Int. Group.

Diaconis, P., Efron, B. (1983), Computer-intensive methods in statistics, Scientific American, Vol 248.

Hunt, E.B., Marin, J., Stone, P.J. (1966), Experiments in induction, New York: Academic Press.

Kononenko, I. (1985), The Design of the ASSISTANT Inductive Learning System, E.Kardelj University: M.SC. Thesis.

Lavrač, N., Varšek, A., Gams, M., Kononenko, I., Bratko, I. (1986), Automatic construction of the knowledge base for a steel classification expert system, the 6th Int. Workshop on Expert Systems and Their Applications, Avignon, April 1986.

Michalski, R.S., Chilausky, R.L. (1980), Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soyabean disease diagnosis, International Journal of Policy Analysis and Information Systems, Vol 4, No.2, pp.125-161.

Niblett, T.B., Bratko, I. (1986), Learning decision rules in noisy domains, Expert Systems 86, Cambridge University Press (Proc. Expert Systems 86 Conf., Brighton 1986).

Patterson, A., Niblett, T.B. (1983), ACLS User Manual, Glasgow: Intelligent Terminals Ltd.

Quinlan, J.R. (1979), Discovering rules by induction from large collections of examples, Expert Systems in the Microelectronic Age (D.Michie ed), Edinburgh University Press.

Quinlan, J.R. (1986), Induction of decision trees, Machine Learning 1, pp.81-106.

Quinlan, J.R., Compton, P., Horn, K.A., Lazarus, L. (1986), Inductive knowledge acquisition: a case study, the New South Wales Institute of Technology, School of Computing Sciences: Technical report 86.4 (presented also at ISSEK Workshop 86, Bled 1986).

Roškar, E., Abrams, P., Bratko, I., Kononenko, I., Varšek, A. (1987), MCUDS - An expert system for the diagnostics of lower urinary tract, Meeting of British Medical Info. Society, London, March 1987.

Progress in Machine Learning

edited by
I. Bratko & N. Lavrač

SIGMA PRESS

Progress in Machine Learning
edited by I. Bratko & N. Lavrač

Progress in Machine Learning

This book contains a selection of papers presented at EWSL 87 – one of a series of annual meetings attended by leading authorities from the AI world.

The papers are presented in camera-ready form to ensure rapid publication, and are grouped into four areas:

Learning from examples in the presence of noise –
how AI systems can function in the real world of imperfect measurements and human subjectivity.

Meta-knowledge and explanation in learning –
the key issues in the transfer of knowledge between humans and machines, and how machines can learn by example.

Studies of special mechanisms for learning –
emerging rule systems and algorithms may soon be applied in practical AI situations.

Learning in complex domains –
how machines can construct 'deep' knowledge bases for solving unusual problems, find new knowledge structures by asking questions, or solve problems that are tough for humans – such as elucidating protein structures.

Ivan Bratko heads the AI laboratory at the J. Stefan Institute and is also a professor in the Faculty of Electrical Engineering at the E. Kardelj University, Ljubljana, Yugoslavia.

Nada Lavrač is a research fellow at the J. Stefan Institute, Yugoslavia.

Published by:

Sigma Press
98a Water Lane
Wilmslow
Cheshire
SK9 5BB

ISBN: 1-85058-088-X

Nada Lavrac
Jozef Stefan Institute, Jamova 39,61000 Ljubljana, Yugoslavia

Igor Mozetic
Jozef Stefan Institute, Jamova 39, 61000 Ljubljana, Yugoslavia

Stephen Muggleton
Turing Institute, 36 North Hanover Street, Glasgow, G1 2AD

Tim Niblett
The Turing Institute, University of Strathclyde

Jean-Francois Puget
Inference and Learning Group, Laboratoire de Recherche en Informatique, U.A. 410 du CNRS, Batiment 490, Université Paris-Sud, 91405 Orsay France.
Tel. 69 41 66 26/69 41 66 29

Carl Uhrik
Dept. of Computer Science, University of Illinois, USA

C J C H Watkins
Philips Research Laboratories, Cross Oak Lane, Redhill, Surrey, RH1 5HA, UK.

CONTENTS