Methodologies from Machine Learning in Data Analysis and Software

D. MICHIE
The Turing Institute, George House, 36 North Hanover Street, Glasgow G1 2AD

In the last few decades the rise of computing and telecommunications has flooded the worlds of government, business, medicine and engineering with unprecedented volumes of stored data. These databases provide the raw material for information supply but have been largely impenetrable as potential sources of expert knowledge. Computer-oriented techniques can now be used, however, in integration with established methods from classical statistics to generate rule-structured classifiers which not only make a better job of classifying new data sampled from the same source but also possess the quality of clear explanatory structure. New developments in the computer induction of decision rules have contributed to two areas, multivariate data analysis and computer assisted software engineering. Practical connections between the two are thereby coming to light. This paper reviews some of the more significant of these developments.

Received July 1990, revised December 1990

1. INTRODUCTION

Computer induction of decision rules from sample multivariate data was already known a quarter of a century ago. But Hunt, Marin and Stone’s CLS (Concept Learning System) initially aroused interest only among cognitive psychologists.1 In recent years, however, new developments have contributed to two applied disciplines, namely (1) multivariate data analysis and (2) computer-assisted software engineering. Practical connections between (1) and (2) are also thereby coming to light. This paper reviews some of the more significant of these developments.

In 1977 Friedman independently incorporated the essentials of CLS in an algorithm suitable for inducing decision trees from statistical-type (i.e. ‘noisy’) data. For this he introduced an important innovation, automatically pruning the tree’s more distal nodes under the control of a user-supplied parameter. At about the same time Quinlan devised and demonstrated an algorithm, ID3 (Iterative Dichotomizer 3), capable of extracting complete logical descriptions from large files of multivariate noise-free data, adverse to analysis for reasons of logical rather than statistical complexity. The data-analytic era of rule induction was consolidated with the pioneering work Classification and Regression Trees by Breiman, et al.3

The opening passage of the above treatise contains the statement: ‘An important criterion for a good classification procedure is that it not only produces accurate classifiers but that it also provides insight and understanding into the predictive structure of the data.’ In the last few decades the rise of computing and telecommunications has flooded the worlds of business, government, medicine and engineering with unprecedented volumes of stored data. These databases provide the raw material for information supply, but have been largely impenetrable as potential sources of expert knowledge. However, computer-oriented techniques can now be used in integration with established methods from classical statistics to generate rule-structured classifiers which not only make a better job of classifying new data sampled from the same source, but also possess the quality of clear explanatory structure emphasised in the above-quoted passage. Applicability has been shown both to data derived from databases (real case histories) and from simulators (model case histories), as in the following knowledge-intensive application areas.

Synthesis from simulation data
- aerospace
- instrumentation
- manufacturing
- pharmaceuticals
- electronics trouble-shooting
- interpretation of biomedical monitoring
- generating software from specifications

Synthesis from captured data
- nuclear engineering
- gas and oil processing
- circuit fault diagnosis
- steel and chemical process industries
- seismic measurement interpretation
- clinical diagnosis
- credit control
- stockmarket assessment

In what follows the essentials of rule-based techniques drawn from machine learning are summarised in the context of earlier approaches.

2. NATURE OF THE PROBLEM

Given. A ‘training set’, or estimation sample, of case-descriptions, each in the form of a list of attribute-values (e.g. age, duration of pregnancy, number of previous births, number of previous pregnancies, marital status, etc.), together with a classification of each case into, say, {YES, NO}, or more generally {class₁, class₂,...,classₙ}, (e.g. Did patients with these characteristics elect to have an amniocentesis test? Did loan applicants giving these pattern of meteorological measurements followed by thunderstorms? Which category of fault was identified from the observed engine test results?)

Required. A classifier (i.e. some formula or rule defined over the attributes) which can classify new cases sampled from the same population.
2.1. Two approaches

We demand of a classifier: (1) that it should predict with high accuracy; (2) that it should be simple and easy to understand.

Decision formulae derived from standard multivariate statistics, such as discriminant analysis or 'naïve Bayes' methods, have the form of sets of positive and negative weights, scoring attribute-values for their individual contributions (assumed independent) to an ACCEPT-versus-REJECT preference for each decision class. Such lists of numbers mean less to the user than they do to the machine.

Logical decision formulae ('rules') get away from the arithmetic. Instead of the operators addition, multiplication etc., decision-tree rules for example use the logical operator 'if...then...else' for combining relevant subsets of attributes into classifying expressions. The above-referenced treatise by Leo Breiman and his colleagues opens with the following illustration:

At the University of California, San Diego Medical Center, when a heart attack patient is admitted, 19 variables are measured during the first 24 hours. These include blood pressure, age, and 17 other ordered and binary variables summarising the medical symptoms considered as important indicators of the patient's condition.

The goal of a recent medical study was the development of a method to identify high risk patients (those who will not survive at least 30 days) on the basis of the initial 24 hours.

The tree-structured classification rule which these authors obtained from their data is shown below.

\[
\text{if the minimum systolic blood over the initial 24-hour period } \leq 91 \\
\quad \text{then risk is HIGH} \\
\text{else if age } \leq 62.5 \\
\quad \text{then risk is NOT-HIGH} \\
\text{else if sinus tachycardia is present} \\
\quad \text{then risk is HIGH} \\
\text{else risk is NOT-HIGH}
\]

The reason for using the term 'tree-structured' is clear from the graphical representation of this same rule, as shown in Fig. 1.

Breiman and his colleagues comment on this rule that its simplicity raises the suspicion that standard statistical classification methods may give classification rules that are more accurate. When these were tried, the rules produced were considerably more intricate, but less accurate. During the six years which have elapsed since then, decision-tree induction from data has been subjected to field trials in various countries by both academic and industrial groups. The results have been in conformity with the above-cited observation. The chief advantages have been that:

1. the amount of calculation is much smaller;
2. the classifiers produced are easier to understand, and can even be published as miniature predictive theories;
3. filtering out irrelevant attributes is done automatically;
4. decision-tree classifiers induced from data in the style of Breiman and his colleagues have been found in practice to be usually more accurate than classifiers formed by adding up discriminant scores.

![Decision tree](image-url)
2.2. Reasons for improved accuracy

When applied to the kinds of data which make difficulties for standard statistical analysis, decision-tree methods gain improved accuracy in two ways.

(1) They can handle both numerical and non-numerical attributes with equal ease.

(2) They do not suffer any loss of discriminant power when some of the attributes violate the simplistic assumption of mutual independence. Consider the four items of decision data (cases) in Table 1 which, if read as statements from an 'oracle' instead of passively as data, collectively define the exclusive-or relation between two binary attributes.

Table 1. A simple example of a data-set which gives trouble to linear numerical estimation methods. For rule-induction procedures (including standard Boolean simplification procedures) the problem is trivial

<table>
<thead>
<tr>
<th>Case</th>
<th>a1</th>
<th>a2</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+1</td>
<td>+1</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>+1</td>
<td>-1</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>+1</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>-1</td>
<td>False</td>
</tr>
</tbody>
</table>

Linear scoring systems such as discriminant analysis are powerless to find scoring coefficients for a1 and a2 such that they can be multiplied by a1's and a2's values, added up, and compared with a threshold in order reliably to distinguish true and false cases. If the reader cares to try it he will find that it cannot be done. But in common with other logic-based induction algorithms, such as those routinely employed by electrical engineers for circuit simplification, decision-tree induction trivialises the problem:

- if a1 = +1
  - then if a2 = +1
    - then false
    - else true
  - else true
else if a2 = +1
  - then true
else false

To construct from such data a classifying expression using numerical multivariate analysis would require an excursion into non-linear regression equations, or (equivalently, see Angus) into multilayer neural networks. Moreover, decision-rule methods are not limited, as are boolean simplification techniques, to problems where attributes are guaranteed to be of logical type. Modern rule-induction techniques are equally at home with inputs which include numerical attribute-values. For this, the standard approach converts input values to logical type by automatically splitting numerical ranges into intervals according to an entropy-minimisation principle. Among other sources, the book by Breiman and his colleagues can be consulted for details.

3. MACHINE LEARNING AS DATA ANALYSIS

Multivariate statistical methods were developed from mathematical and scientific foundations by pioneers such as Francis Galton, Karl Pearson and Ronald Fisher. These men made it possible to give precise answers to questions phrased within the statistical paradigm. 'Within what limits of error can this or that classifier be expected to perform?' or 'How much will the error be narrowed by a given increase in the size of estimation sample?' Studies in the computational theory of learning are directed towards building scientific foundations for the machine-learning approach as an extension of classical probability and statistics (in the case of decision-tree induction see, for example Refs 7, 16, 17 and 18).

The connection between inductive learning and statistical data analysis can be explained as follows.

Approximately two decades of exposure to data turns a baby into a mentally capable adult. Evidently the developing brain extracts something of continuing and incremental value. We call the process of extraction 'learning'. For the something which is extracted, stored, refined, built upon and exploited, we have variations on notions of 'knowledge'. We speak about behaviour-patterns, habits, skills, hypotheses, beliefs, models, descriptions, concepts, theories. These structures have one thing in common: they all act as classifiers. Empirical scientists here recognise something familiar. They too are concerned with extracting theories from data, alternating with the theory-guided sampling of new data. In this cycle of extraction and testing the scientist commonly calls on the aid of a special breed of numerical craftsman, the data analyst.

According to one rather undemanding definition, the statistical data analyst's fitting of models to data would qualify as a form of machine learning. This definition says:

- a learning system uses sample data (the training set) to generate an up-dated basis for improved classification of subsequent data from the same source.

Notice that the definition, although phrased strictly in terms of classification, logically extends to acquisition of improved performance on tasks which do not look at all like classification. Iterative situation-action tasks come to mind such as riding a bicycle, solving an equation, or parsing a sentence. The extension becomes obvious when for the decision classes we choose names which refer to partitions of the space of situations as 'suitable for action A', 'suitable for action B', etc.

4. MACHINE LEARNING AS PROBLEM SOLVING

To illustrate the way in which iterative situation-action problems can be coded as classification, consider the task of solving simple equations in schoolroom algebra (see Ref. 20). Thus 3x + 1 = 2 is transformed by an appropriate action ('collect like terms on same side of the equation') into 3x = 1 (by 'combine like terms') and thence into the final 'solution', x = 1/3 (by 'divide by the coefficient of the unknown'). This is the solution or 'goal'.

Using the attributes/classes format, the problem-description is given in Table 2. Line 1 gives the number of attributes, lines 2–7 describe the attributes and the last two give the number and names of the classes.

The key to the six attribute names is:
(a1) Does the equation have a common factor? 
(comfact)

(a2) Are there like terms on opposite sides? (likeopp)

(a3) Are there any bracketed terms? (bracket)

(a4) Does either side have like terms? (likesam)

(a5) Is exactly the same present on both sides? 
(samebrm)

(a6) Is there only one unknown term and is its coefficient equal to one? 
(xcoeff)

Table 2. Problem description for inductive derivation of an equation-solving rule

<table>
<thead>
<tr>
<th>ID no</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>collect</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>combine</td>
</tr>
<tr>
<td>3</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>divcx</td>
</tr>
<tr>
<td>4</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>stop</td>
</tr>
</tbody>
</table>

The seven class names given earlier have the following interpretations:

(1) Divide the equation by its common factor (divcf).
(2) Collect like terms on the same side of the equation (collect).
(3) Multiply out bracketed terms (multbr).
(4) Combine like terms (combine).
(5) Cancel out a term that appears on both sides (cancel).
(6) Divide by the coefficient of the unknown (divcx).
(7) Stop because the equation is solved (stop).

The four lines of examples given at the outset, namely

3x + 1 = 2, 3x = 2 - 1, 3x = 1, x = \frac{1}{3}

would appear in an induction file as:

\begin{verbatim}
6 comfact logical yes no
likeopp logical yes no
bracket logical yes no
likesam logical yes no
samebrm logical yes no
xcoeff logical yes no
3 divcf collect multbr combine
cancel divcx stop
\end{verbatim}

From the output of a skilled human equation-solver on a number of example cases of the above kind, an induction system acquires expertise. From sequences of illustrative solution steps, a solver is built. Application of the solver to actual problems is iterative. In each cycle the output action is applied to the current state of the equation to yield a modified state. This is then fed back to the classifier as a new input, and so on until the output *stop* appears. Although seemingly remote from the task of synthesizing a controller from a recorded behavioural trace of a skilled human operator (see Ref. 19 for a worked example), the logic is the same. Successive 'snapshots' of the state of the dynamical system correspond to successive lines of symbols in equation-solving. The STOP action is of course normally dropped to allow for instabilities and perturbations from the goal state, normally absent from symbol-manipulation tasks.

5. A MORE DEMANDING DEFINITION OF LEARNING

A more demanding definition of learning is now coming from applied artificial intelligence:

a learning system uses sample data to generate an up-dated basis for improved classification of subsequent data from the same source, and expresses the new basis in intelligible symbolic form.

According to this more demanding definition, not only improved performance results from the learning process, but also an explicit set of rules. Decision-tree induction in the algebra domain meets not only the less demanding but also the more demanding criterion. An intelligible symbolic form obtained by the ACLS algorithm from a training set of equation solutions is shown in Fig. 2. Such a form constitutes an operational theory, a prescription which can be followed by any agent able to interpret it, whether human or machine.

Michie et al. found that use of a version of ACLS by high-school children could be an effective mode of self-instruction. Children were asked to teach the machine equation-solving from an elementary algebra text-book by selecting and supplying example solvings. The children's subsequent grasp was tested against that of a group of classmates who had been exposed instead to a conventional CAI algebra package.

6. MACHINE LEARNING AS THEORY-CONSTRUCTION

Scientists of the past have been content if the automated procedures of data analysis satisfied the undemanding definition only, taking on themselves the responsibility of abstracting from the analysis the desired explanatory or predictive theories. AI-based machine learning, combining as it does both logical and statistical idioms, follows the more demanding definition. It thus enters directly into the theory-building process, as was first shown in 1976 by groups working in chemistry at Stanford, USA, and in plant pathology at the University of Illinois. Subsequent studies in the USA, Australia, Yugoslavia and Britain established the tree-structured paradigm of computer induction as the dominant form for rule-based data analysis. The following were the chief advances.

(1) In 1977 J. H. Friedman introduced ways of de-
riving tree structures from data in the style of Earl Hunt’s 1966 scheme but constrained by criteria for pruning unprofitable branches. Decision-tree induction was thus generalised to noisy data, yielding trees with confidence measures associated with their leaves (outcome nodes).

(2) In 1979 J. R. Quinlan published the first of a series of papers describing the ID3 series of decision-tree algorithms, the efficiency and versatility of which led to their widespread adoption as the basis of commercial inductive software engineering. 25 For large problems (the 1979 paper describes a rule-based solution of a problem which in unreduced form constituted some 21/2 million records) extracted decision trees were efficient at runtime but unstructured and hence obscure — the automated equivalent of hand-crafted ‘spaghetti code’.

(3) A. D. Shapiro and T. Niblett 28 elaborated the Quinlan paradigm with a method known as ‘structured induction’, subsequently studied in depth by Shapiro. 27 To supplement the initially given primitive attributes they added separately induced procedural attributes. Structured induction confers on inductive data analysis the benefits of top-down problem decomposition, as in the discipline of structured programming. In its original form, it can only be applied where noise is absent and where the problem can be fully specified in terms of the primitives. Today the structuring steps can in suitable cases themselves be automated (see (7) below).

(4) In 1987 J. R. Quinlan adapted his C4 algorithm for inducing trees from noisy data so as to generate solutions in the form of compact sets of logic rules. 26 After pooling the branches harvested from the trees separately induced from the same data set, the program winnows out redundancy and delivers a compact and intelligible local theory of the data.

(5) Quinlan’s current version, C4.5, has been used to recover from the recorded behaviour of simulator-trained human subjects sets of ‘production rules’ of the type postulated by cognitive psychologists. 19 Such productions, in some neurally encoded form, are believed to underlie decision skills. As compared with the original expert behaviour, induced rules exhibit a ‘clean-up effect’, having shed some of the inconsistencies and noise which even the most highly trained nervous system introduces into the recognise–act cycle.

(6) D. Michie derived, 17 and with A. Al-Attar, partly tested, a formulation of decision-tree induction from the axioms of Bayesian probability. 18 Such productions, in some neurally encoded form, are believed to underlie decision skills. As compared with the original expert behaviour, induced rules exhibit a ‘clean-up effect’, having shed some of the inconsistencies and noise which even the most highly trained nervous system introduces into the recognise–act cycle.

(7) Full automation of structured induction requires algorithms for generating new procedural attributes, rather than depending for this on the knowledge-based insight of human domain specialists. Advances by Muggleton and Buntine and by Bain and Muggleton at the Turing Institute, Glasgow 22 have yielded needed algorithms within the framework of first-order predicate logic (see also Ref. 21).

7. INDUCTIVE PROGRAM GENERATION

The work by A. D. Shapiro referred to above extended data-oriented induction into the realm of computer-assisted software engineering (CASE), and recent extensions to first-order level have established points of contact with the formal methods school. On the practical side, a path has finally been found to circumvent what has sometimes been termed the ‘bottleneck problem’ facing knowledge acquisition from experts.

A few years ago it was hoped that experts — doctors, engineers, etc. — would be able to teach their skills directly to computers, which would then be able to carry out much of their routine diagnostic work. Faults or symptoms would be fed into a computer, which would then give a diagnosis of the problem. Unfortunately, it was found that if explicit how-to-do-it rules are required from them, experts cannot effectively feed their own decision-making processes into computers. The soyabean specialist, the analytical chemist or the cardiologist largely react ‘intuitively’ to data, in ways he or she cannot fully explain. In the realm of evaluating credit-worthiness in the finance industry, L. Sterling and E. Shapiro give a telling account of the phenomenon: 29

The major difficulty was formulating the relevant expert knowledge. Our expert was less forthcoming with general rules for overall evaluation than for rating the financial record, for example. He happily discussed the profiles of particular clients, and the outcome of their credit requests and loans, but was reluctant to generalise.

The observation that specialists transmit their inarticulate skills to trainees by example, rather than by using explicit rules, led in the mid-1980s to the development of commercial programs to exploit ‘teaching by showing’ in the style illustrated with schoolroom algebra. The machine learns ‘how to do it’ from experts who supply examples. A number of large corporations, notably British Petroleum, 29 have begun using this approach as a cost-effective way of building large applications. The resultant programmer productivities exceed current industry standards by an order of magnitude. The world’s largest expert system, BMT, for configuring fire-detection equipment, was built by the German company Brainware using the RuleMaster and 1st Class inductive shells. 10 BMT consists of 150,000 lines of inductively generated C code and is in routine use by the client organisation. The total figure of 9 man-years expended on the project includes management, support staff and domain specialists as well as programmer time. Reductions of software maintenance overheads have also been impressive, as can be seen from the summary results set out in Table 3.

The reader can appreciate the connection with CASE methods by thinking of expert supplied examples as statements in a requirements-specification language: ‘in situations of this kind we require the system to do x; in situations of that kind we require it to do y, etc.’. In this context an induction routine can be thought of as a kind of compiler, translating from a specification consisting of example cases into a program consisting of if–then–else expressions. A mass of partly redundant and partly incomplete specifications becomes a structured set of efficient executable rules. Permissiveness towards redundancy and incompleteness in the user’s tabulation of cases marks the sole but significant departure of rule-induction programming from mainstream decision-table methods. 12,13 Rule induction in a CASE context is indeed
little more than the rebirth of structured decision tables, with this difference: that specification tables, as we may here term them, are initially partial, and are developed incrementally in successive cycles of generate-and-test. Herein lies the key to the extraordinary productivities, illustrated in Table 3, where numbers of the order of 100 lines of installed code per programmer-day are the norm. The even more remarkable gains in software maintainability can be similarly explained. Modern inductive shells automatically flag not only incomplete parts of the rule-base but also those which have been derived from generalisation steps in the induction process. The user can then interactively validate flagged passages, editing specification tables as required, re-inducing at each stage from the edited ‘spec’.1516

General implications for software manufacture were discussed a few years ago in the author’s Royal Society Technology Lecture,14 and have more recently been elaborated in the specific contexts of interactive validation and of inductive logic programming.1621

8. AUTOMATED SYNTHESIS OF OPERATIONAL KNOWLEDGE

The possibility had long been foreseen of using inductive inference to derive operational models (a formal equivalent of ‘skills’) from deep models (a formal equivalent of ‘understanding’). Thus from a numerical model of an aero-engine a qualitative model can in principle be prepared, from which all possible combinations of engine faults may automatically be listed, together with the corresponding sensor readings and test responses. From such a giant tabulation, efficient rules for use in future fault diagnosis may be machine-induced. The wild card in the foregoing is the phrase ‘in principle’. But Yugoslav work using a computer model of the human heart resulted in the discovery by machine of a corpus of new clinical rules for interpreting electrocardiogram patterns.14 Work at Glasgow’s Turing Institute on diagnosing electronic faults in a space satellite confirmed the methodology as fully practicable.24 These synthetic rule-bases comprise new diagnostic knowhow beyond the achievements of human specialists. Their means of construction also automatically guarantees completeness and correctness with respect to the formal models used to generate the exhaustive datasets. Since the latter are logically equivalent to the descriptive specifications from which they were derived, they too may be viewed as formal specifications, in the form of very long sentences written in a ground-level data-description language. This insight, however, logically irrefutable, seems strange to some who approach via formal methods, where conciseness in a specification is of the essence. The Bratko and Pearce syntheses do indeed start from concise (intensional) specifications, but find the path to automated synthesis via an intermediate product, namely a complete extensional form which is its logical equivalent, with the added property of inductive transformability into a set, again concise, of operational rules.

REFERENCES

Announcement

27-29 MAY 1992

Symposium on Assessment of Quality Software Development Tools, New Orleans, Louisiana

Sponsored by Tulane University in cooperation with IEEE Computer Society TCSC (Technical Committee on Software Engineering), with the assistance of IBM Systems and Software Education.

In the last few years there has been a major renaissance in the availability, kinds and scope of software development tools. The growth of the CASE umbrella, repository architectures and the widespread use of workstation technology have changed the nature of tools. Modern tools come in large number and large variety, creating a new challenge to software engineers: how to choose the right tools. There is no clear and simple way today to go about assessing tools and matching them to the needs of development organisations.

This Symposium will review the problems of assessing software development tools soliciting case studies of tool applications and their impact on productivity, and examine strategies for the evaluation of future tools. In particular the symposium will focus on the assessment of tools to assist with productivity and quality in software development.

For further information contact: Ez Nahourali, IBM (786/0899), 6321 San Ignacio Avenue, San Jose, CA 95119, USA. Tel: (408) 281-3741. E-mail: eznah@stlvm7.inuus1.ibm.com.