Induction of Classification Rules from Examples: A Critical Review

Professor M.A.Bramer

Artificial Intelligence Research Group
Department of Information Science
University of Portsmouth
Milton
Southsea
PO4 8JF

Tel: +44-1705-844444 Fax: +44-1705-844006
Email: bramerma@cv.port.ac.uk

1. Introduction

In recent years the considerable commercial potential of the subfield of Machine Learning known as Data Mining has increasingly become recognised.

One of the key technologies of data mining is the automatic induction of rules from examples, particularly the induction of classification rules. There are two principal motivations for work of this kind:

(a) the need to analyse the ever-growing volume of data held by many organisations, where the aim is to discover hidden relationships in the data

(b) the well-documented 'knowledge acquisition bottleneck' in the development of rule-based expert systems, where the supplying of significant case examples to an inductive learning program has frequently been advocated as a more natural means of knowledge transfer than the conventional techniques of Knowledge Engineering.

In advocating the use of rule induction techniques, Michie (1990) pointed out that two of the three largest Expert Systems in the world at that time (GASOIL and BMT) had been produced using rule induction techniques. Comparative data on these and two other celebrated Expert Systems (MYCIN and XCON) is shown in Figure 1.

	Application	No. of Rules	Develop. Man-Years	Maintenance Man-Years Per Year	Inductive Tools
MYCIN	**Medical Diagnosis**	**400**	**100**	**N/A**	**N/A**
XCON	**VAX Computer Configuration**	**8,000**	**180**	**30**	**N/A**
GASOIL	**Hydrocarbon Separation System Configuration**	**2,800**	**1**	**0.1**	**ExpertEase and Extran 7**
BMT	**Configuration of Fire Protection Equipment in Buildings**	**>30,000**	**9**	**2.0**	**1st Class and RuleMaster**

Figure 1. Taken from Michie (1990) Machine Executable Skills from 'Silent' Brains

There are two particularly influential families of rule induction algorithms: the ID3 family originated by Quinlan (1979) and the AQ family due to Michalski (see for example (Michalski and Chilausky, 1980)).

In this paper the ID3 family of systems for the inductive learning of classification rules will be described and used to illustrate some of the technical issues associated with the use of this technology. Although many of the same issues arise with the AQ family of systems, in the interests of clarity of explanation such systems will not be described here explicitly.

2. Induction of Classification Rules from Examples

2.1 Top-down Induction of Decision Trees

The basic technique used by the ID3 family of algorithms to induce classification rules from a database of examples (which may be complete or partial) is known as Top-Down Induction of Decision Trees (TDIDT).

In the standard formulation of the algorithm, the database is known as the training set and each example in the database is known as an instance.

It is assumed that there is a universe of objects, each of which belongs to one of a set of mutually exclusive classes.

Each instance in the training set must be a description of an object, together with the class to which it belongs.

Each object is described by the values of a collection of its attributes (the same attributes being used for each instance), each attribute taking one of a set of discrete, mutually exclusive values.

It is customary to depict the training set by means of a two-dimensional table, with one row per instance, and each column holding the values of a particular attribute or the class to which the object belongs.

The aim is to develop classification rules in the form of a decision tree which enables the class to which any object in the training set belongs to be determined from the values of its attributes.

As an example, Table 1 below is a training set containing data about 26 university students, linking the results for five final-year subjects, coded as SoftEng, ARIN, HCI, CSA and Project, with the degree classification (in this case either FIRST or UPPER).

SoftEng	ARIN	HCI	CSA	Project	Class
A	B	A	B	B	UPPER
A	B	B	B	B	UPPER
A	A	A	B	B	UPPER
B	A	A	B	B	UPPER
A	A	B	B	A	FIRST
B	A	A	B	B	UPPER

A	**B**	**B**	**B**	**B**	**UPPER**
A	**B**	**B**	**B**	**B**	**UPPER**
A	**A**	**A**	**A**	**A**	**FIRST**
B	**A**	**A**	**B**	**B**	**UPPER**
B	**A**	**A**	**B**	**B**	**UPPER**
A	**B**	**B**	**A**	**B**	**UPPER**
B	**B**	**B**	**B**	**A**	**UPPER**
A	**A**	**B**	**A**	**B**	**FIRST**
B	**B**	**B**	**B**	**A**	**UPPER**
A	**A**	**B**	**B**	**B**	**UPPER**
B	**B**	**B**	**B**	**B**	**UPPER**
A	**A**	**B**	**A**	**A**	**FIRST**
B	**B**	**B**	**A**	**A**	**UPPER**
B	**B**	**A**	**A**	**B**	**UPPER**
B	**B**	**B**	**B**	**A**	**UPPER**
B	**A**	**B**	**A**	**B**	**UPPER**
A	**B**	**B**	**B**	**A**	**FIRST**
A	**B**	**A**	**B**	**B**	**UPPER**
B	**A**	**B**	**B**	**B**	**UPPER**
A	**B**	**B**	**B**	**B**	**UPPER**

Table 1. Degree Classification Data

Given this data the natural question is: what determines who is classified as FIRST or as UPPER?

(Note that in the general case there can be any number of classes, not just two as here. However the classes must be mutually exclusive.)

The basic TDIDT algorithm can be described informally as follows:

IF all cases in the training set belong to the same class

THEN return the value of the class

ELSE

(a) select an attribute A to split on

(b) sort the instances in the training set into non-empty subsets, one for each value of attribute A

(c) return a tree with one branch for each subset, each branch having a descendant subtree or a class value produced by applying the algorithm recursively for each subset in turn.

It is important that in selecting attributes at step (a) the same attribute must not be selected more than once in any branch.

Applying this algorithm to the data in Table 1 will produce a number of possible decision trees depending on the method used to choose attributes at step (a). One possible tree is shown below.

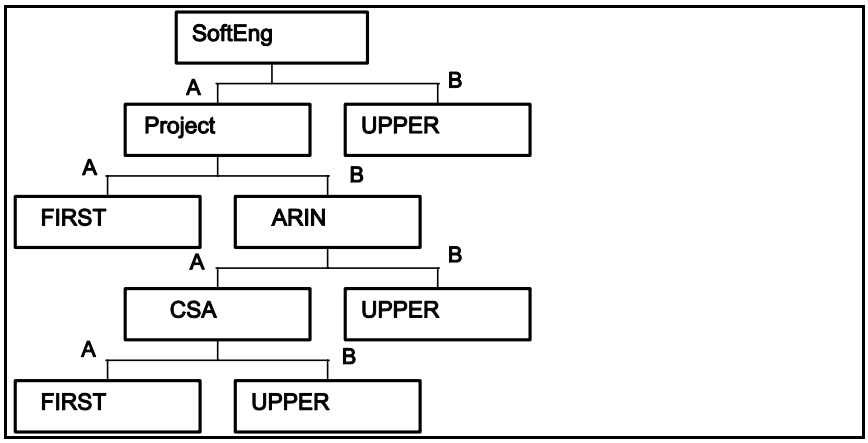

Figure 2. Decision Tree for Degree Classification Data

Provided the set of attributes is adequate, i.e. no two different instances have the same values of all the attributes but belong to different classes, any method of choosing attributes at step (a) will suffice to produce a decision tree, providing of course that the same attribute is not chosen twice in the same branch. As the number of attributes is finite, the algorithm is certain to terminate however the attributes are chosen.

2.2 Decision Trees Viewed as Rules

The induced decision tree can be regarded as a set of classification rules, one corresponding to each branch of the tree. Thus, working from left to right through the branches of Figure 2 gives the five rules:

if SoftEng = A and Project = A then Class = FIRST

if SoftEng = A and Project = B and ARIN = A and CSA = A
then Class = FIRST

if SoftEng = A and Project = B and ARIN = A and CSA = B
then Class = UPPER

if SoftEng = A and Project = B and ARIN = B then Class = UPPER

if SoftEng = B then Class = UPPER

The 26 instances in the training set can themselves be written in the form of rules, one per instance, each having a conjunction of five terms in its premise, e.g. the first instance can be written as:

if SoftEng = A and ARIN = B and HCI = A and CSA = B
and Project = B then Class = UPPER

Even for this simple example and with an (effectively) arbitrary choice of attribute to split on at each stage, it is significantly simpler to find the class to which any member of the training set belongs by using the induced set of rules than by examining the original data. An alternative way of viewing an induced decision tree is therefore as a more compact means of storing the data in the training set and in practice a considerable compression of data is frequently possible.

Using the number of terms contained in the complete set of rules (or equivalent) as a crude measure of complexity, the original training set scores 26 x 5 = 130 whereas the induced decision tree scores only fourteen (an average of 2.8 terms per rule).

2.3 The ID3 Family of Rule Induction Algorithms

The ID3 algorithm of Ross Quinlan (Quinlan 1979, 1983) is an extension of the basic TDIDT algorithm described in Section 2.1. ID3 aims at developing classification rules in the form of decision trees of the kind previously described. It was originally developed for domains in which no 'noise' is present in the data but Quinlan (1986) also describes experiments with noisy data.

In the original ID3 experiments (Quinlan, 1979) the choice of attribute to split on in the basic induction algorithm was based on the use of an ad hoc formula. This was subsequently

replaced by an entropy formula derived from information theory which amounts to choosing at each stage the attribute which maximizes the expected gain of information from applying the additional test. Details are given in Quinlan (1986).

An important point in the original design of ID3 was the need to construct decision trees for large (or very large) training sets, where the basic induction algorithm could not be used directly. The approach to this problem adopted in ID3 was an iterative one based on the use of windows, i.e. relatively small subsets of the whole training set.

The basic ID3 algorithm is as follows:

(1) choose a subset of the whole training set to work with as an initial window

(2) perform the TDIDT algorithm on the instances in the window, using entropy to determine which attributes to split on at each stage

(3) find all cases where instances in the whole training set are wrongly classified by the rules induced by (2)

(4) select a replacement window (incorporating some misclassified instances)

(5) go back to step (2)

The algorithm continues until there are no incorrectly classified instances at step (3).

There are two alternative approaches to choosing the replacement window at step (4) described in Quinlan (1979): the fixed window, where a number of instances in the window are replaced by the same number of incorrectly classified instances outside the window at each stage, and the growing window, where the window grows by the addition of (previously incorrectly classified) instances at each stage. The role of windowing in ID3 will be discussed further in Section 3.2.

2.4 Early Extensions of ID3

The principal developments from the original ID3 algorithm up to the mid-1980s were reviewed by Quinlan (1986), from which the 'family tree' shown in Figure 3 is taken.

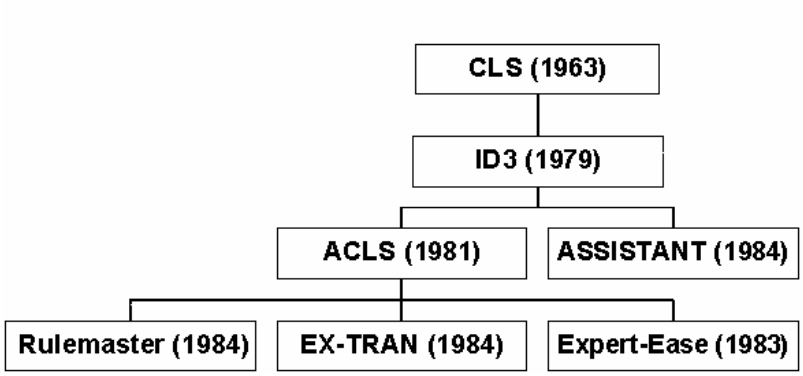

Figure 3. The ID3 Family of Rule Induction Systems: Early History

This figure also shows the line of descent of ID3 from Hunt's Concept Learning System (Hunt, Marin and Stone, 1966) which constructs a decision tree that attempts to minimize the cost of classifying an object.

In describing his own work, Quinlan comments: "ID3 ... is one of a series of programs developed from CLS .... it embeds a tree-building method in an iterative outer shell, and abandons the cost-driven lookahead of CLS with an information-driven evaluation function".

Quinlan goes on to describe the other systems shown in the diagram as follows: "ACLS (Paterson and Niblett, 1983) is a generalisation of ID3. CLS and ID3 both require that each property used to describe objects has only values from a specified set. In addition to properties of this type, ACLS permits properties that have unrestricted integer values .... ASSISTANT (Kononenko, Bratko and Roskar, 1984) also acknowledges ID3 as its direct ancestor. It differs from ID3 in many ways .... ASSISTANT further generalizes on the integer-valued attributes of ACLS by permitting attributes with continuous (real) values. Rather than insisting that the classes be disjoint, ASSISTANT allows them to form a hierarchy, so that one class may be a finer division of another. ASSISTANT does not form a decision tree iteratively in the manner of ID3, but does include algorithms for choosing a 'good' training set from the objects available. The bottom-most three systems in the figure are commercial derivatives of ACLS.

While they do not significantly advance the underlying theory, they incorporate many user-friendly innovations and utilities."

Another early extension of ID3 was the 'Structured Induction' approach of Shapiro (1983) which incorporated an initial problem decomposition stage.

Although the ID3 algorithm has been very influential (both in its own right and as an important component of at least two commercial Case-Based Reasoning shells, ESTEEM and KATE) there are some significant problems associated with the basic form as described above and there are many variants of the basic algorithm aimed at overcoming these. The technical issues associated with the induction of classification trees and some of the approaches to overcoming them are discussed in Section 3.

3. Induction of Classification Rules: Technical Issues

3.1 Generality

(a) The basic form of the ID3 algorithm as described in the last section is quite restrictive. Some of these restrictions were eased even in the early derivatives of the original algorithm described in Section 2.4, e.g. by allowing attributes to have continuous values (in ASSISTANT) or by allowing rules to have components involving inequality relations for integer-valued variables (in the Expert-Ease package).

However, more fundamental restrictions remain. The most important of these concerns representation. ID3 can only express (in the form of a decision tree) conjunctive propositional rules of the general form

if $att_1 = val_1$ and $att_2 = val_2$ and ... then class = c

i.e. conjunctions of simple terms each concerning a constant value of a single attribute (where '=' can be replaced by an inequality relation in some versions).

It is therefore not possible to construct rules relating different attributes of an object, such as 'Length = Breadth' or 'Force = Mass x Acceleration', or rules relating properties of different objects in a domain.

The basic method also cannot be used if the value of an attribute is unknown or uncertain (e.g. it is known to be either A or B).

Although there are some systems which permit more general forms of rule to be generated, particularly those which can be expressed in First Order Predicate Logic, the discussion in this paper will be restricted to the generation of conjunctive propositional rules or the equivalent in the form of a decision tree.

(b) This paper is primarily concerned with the generation of classification rules, but some systems such as CART (Breiman, 1984), Cupid (Mallen, 1995) and C4.5 (Quinlan, 1993) have taken the broader objective of finding any pattern of interest in the data, known as Generalised Rule Induction.

(c) The rules produced by ID3 and most related systems are exact, but some systems, such as CN2 (Clark and Niblett, 1989) and the commercial system Knowledge Seeker (Struhl, 1995), permit the generation of probabilistic rules, of the kind found in many expert systems.

3.2 Use of Windowing in ID3

The use of windowing in ID3 has a strong intuitive appeal. When termination of an iterative process of this kind occurs it seems reasonable to believe that the induced rules will have a high level of predictive power for instances not in the training set, since rules 'explaining' all

the instances in the training set will already have been generated from only a subset of it.

However, one of the earliest papers on ID3 (Quinlan, 1979) strongly suggests that windowing should be seen simply as a practical way of coping with a large training set. He remarks "We assume that the training set available to the experimenter is substantial - it may even contain all possible instances. Such a training set will usually be too large to present to the induction algorithm, requiring either too much memory or excessive computation. The approach taken ... was to select a subset of it called a window and to modify this subset in the light of the rule[s] discovered using it".

The use of windowing has a number of associated problems. First, there is no guarantee at all that the iteration will converge. In the case of a fixed window, since instances which have once been discarded from the window can subsequently become 'exceptions' (i.e. wrongly classified instances) and be reintroduced, it is perfectly possible that the algorithm will loop. For a growing window, O'Keefe (1983) has shown that the iteration is only certain to converge in the 'degenerate' case where the window can grow to include the full training set. In practice, ID3 users have frequently dispensed with windowing altogether, the induction algorithm being applied directly to the entire training set.

Most of those who have used windowing would seem to have chosen the growing window technique, but no consensus has emerged concerning the most appropriate choice of parameters for doing so. There seems to have been little systematic investigation of how the choice of parameters such as the window size and the criteria by which instances are added to/deleted from the window may affect the final induced decision tree. It may well be that in a particular case certain choices of window size, replacement criteria etc. would lead to infinite looping whilst other not very dissimilar choices would lead to a successfully generated decision tree.

Wirth and Catlett (1988) report a series of experiments aimed at investigating the merits of the windowing technique used in ID3. In nearly every experiment the use of windowing considerably increased the processing requirements of ID3 but produced no significant benefits.

They state that 'Windowing is an intuitively appealing idea which seems to reduce the problems of scale to a more manageable level. It successfully demonstrates that it is possible to capture a concept represented by a large training set with a subset of those examples, sometimes a relatively small subset. However the effort of finding such a subset is usually greater than that of learning using the whole set from the start. We have shown that the cost of using windowing with ID3 is almost always a significant increase in running time or no improvement'. They conclude that windowing should be avoided, particularly in noisy domains (see Section 3.6 below).

A modified form of the ID3 windowing technique is employed in Quinlan's later system C4.5 (Quinlan, 1993), the basis of the commercial rule induction system Clementine.

3.3 Attribute Selection

The selection of attributes is a vital issue in inductive learning. In ID3 it is carried out using the Information Theoretic measure entropy. However it is not self-evident that this or any similar measure will necessarily lead to a decision tree which is either meaningful to subject experts or of significant predictive value in the open world case of a partial training set, however valuable it may be for purposes of data compression in the closed world case of a complete training set.

In one sense, the selection of the most appropriate attribute to split on at each stage of the basic induction algorithm is arbitrary, since any choice will still lead to a decision tree. In practice it is likely to be of crucial importance if the induced rules are to have any predictive value in the normal 'open world' case, where the effect of even one bad choice of attribute to split on when constructing a decision tree can be quite substantial.

As an extreme example, if a set of attributes contains one that is completely irrelevant to the classification and by chance this one is chosen to split on first, the eventual decision tree at worst may be completely meaningless and at best might well include a number of identical subtrees - one for each of the values of the irrelevant attribute.

As an example, Table 2 shows four attributes recorded for each of 12 University students together with the 'class' to which each belongs - in this case whether each is a member of the University's Rugby club or its Netball club. (For simplicity, it will be assumed that there is a strict rule that each student must belong to one and only one of these two clubs.)

eyecolour	married	sex	hairlength	class
brown	yes	male	long	rugby
blue	yes	male	short	rugby
brown	yes	male	long	rugby
brown	no	female	long	netball
brown	no	female	long	netball
blue	no	male	long	rugby
brown	no	female	long	netball
brown	no	male	short	rugby
brown	yes	female	short	netball
brown	no	female	long	netball
blue	no	male	long	rugby
blue	no	male	short	rugby

Table 2. Club Membership

One possible decision tree which can be induced from this data is Figure 4 below (the figures in parentheses indicate the number of instances to which each branch of the tree corresponds).

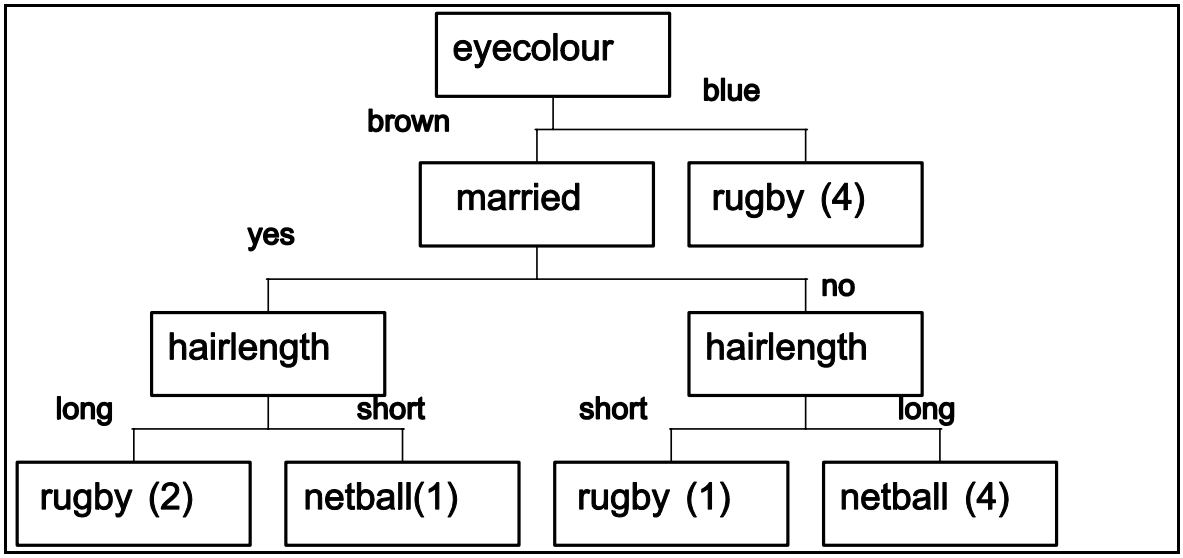

Figure 4. Club Membership: One Possible Decision Tree

With a different algorithm for attribute selection a different decision tree can be induced from the same data as follows.

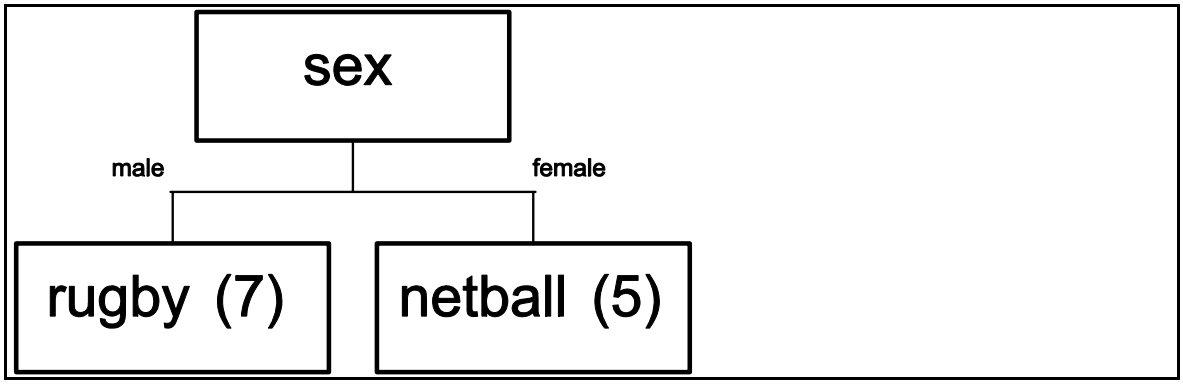

Figure 5. Club Membership: Another Possible Decision Tree

It looks suspiciously as if the real distinction between members of the two clubs is their sex, but this may or may not be the case. If it is, Figure 5 is certain to predict perfectly for every instance not in the training set and the predictive power of all other possible decision trees is likely to be extremely low (unless they include a test of the value of attribute 'sex' in every branch).

On the other hand, if there are (say) male students who are members of the netball club but who are not included in the training set, the decision tree shown in Figure 5 will certainly fail to make the correct prediction for them but the decision tree shown in Figure 4 might.

The simplicity of the decision tree shown in Figure 5 may look convincing, but whether or not it correctly captures the underlying causality of the domain from which the training set was drawn depends on the nature of the training set, in particular whether all 'critical cases' are included.

Unfortunately, in many real-life situations, where data is collected on a partially self-selecting basis (e.g. from patients admitted to a particular hospital or from passers-by being interviewed in the street) it is extremely difficult to ensure that all critical cases are included as well as to avoid an unintended bias. Example cases selected by experts may include critical cases (even ones which occur very rarely in practice) but are extremely unlikely to avoid (statistical) bias without a great deal of care. It is clearly not likely that a training set will both be a random sample and include all necessary critical cases.

3.3.1 Alternatives to Entropy

A serious fundamental weakness of the entropy formula used to select attributes in ID3 is that it is essentially an 'unintelligent' process. There is nothing to prevent an attribute of little or no relevance being selected to split on by chance, in preference to one of high relevance.

Thus in a medical training set, it might be well understood by experts that the sex of the patient was likely to be of great importance, whereas (say) the patient's country of origin was of much less importance. There is no explicit way this information can be represented or used in ID3, although it could possibly be dealt with by associating costs with attributes as in CLS.

The entropy formula makes use of probabilities and in the open world case these are generally not known, but can only be estimated from the corresponding frequencies of occurrence in the training set. Since the training set will generally not be a random sample from the (usually unknown or unavailable) set of all possible instances, estimating probabilities from frequencies will not necessarily be reliable.

The value of using a formula based on maximizing the expected gain of information is also not entirely clear. It has been pointed out that if the attribute 'name' or 'reference number' is included in a training set (say in a medical classification application) it is certain to be chosen to split on and, indeed, completely determines the classification. Unfortunately, it has no predictive power whatever for instances not in the training set.

ITRule (Smyth and Goodman, 1992) makes use of an alternative information theoretic measure of rule 'interestingness' in order to generate propositional rules. The user specifies the number of rules N to store in a table and also the maximum complexity (order) a rule may reach, measured by the number of attribute-value pairs on a rule's left-hand side. Initially all rules of complexity one are generated and their J values calculated. The right-hand sides of

rules can be a conjunction of the values of several attributes.

The best N rules are stored in a table and ITRule then attempts to 'specialise' the existing rule set by progressively adding 'attribute = value' pairs to the left-hand sides of rules provided that conditions on the value of the J measure continue to be met. The final table then contains the N 'most interesting' rules describing the dataset.

Although the J measure would seem to be considerably less well-known than the entropy measure of information gain, it would seem to be a promising approach and has been followed up in the CUPID system (Mallen and Bramer (1994), Mallen (1995)) which is discussed further in Section 3.11.

Breiman (1984) investigated statistical techniques for constructing classification trees, leading to the development of the CART classification algorithm (subsequently used for inductive retrieval in the REMIND Case-Based Reasoning shell).

The commercial rule induction package Knowledge Seeker uses the statistical chi-square test for attribute selection.

Research by Mingers (1989a), followed up by Buntine and Niblett (1992) and Liu and White (1994) has concentrated on the idea of using a random choice of attributes followed by tree pruning (see Section 3.8 below) in the construction of decision trees in noisy domains.

3.3.2 Inductive Bias

A typical example of a problem set in so-called 'intelligence tests' is the following:

> **Find the next term in the sequence 1, 4, 9, 16, ...**

> **$(-5n^4+50n^3-151n^2+250n-120)/24$**

The problem solver is clearly expected to solve this by a process of inductive reasoning based on finding the underlying formula which generated the sequence.

Although presented as find the next term, any number (or other value) is a valid answer and there are infinitely many formulae which will generate a sequence of any length beginning with the four terms given (obtained, for example, by fitting polynomials of progressively higher degree).

One possible answer for the fifth term is 20, based on a generating formula for the nth term of:

It is interesting to note that this answer is virtually certain to be marked as simply 'wrong' by the examiner of the intelligence test even though it is as justifiable as any other solution. It is clear that there is an inductive bias at work here in favour of certain kinds of solution (ones involving squares or cubes of integers etc.) at the expense of others (such as ones involving fourth-order polynomials).

Although the inductive task may at first seem to be that of finding a solution, finding some solution is in fact trivial and the real task involved is to find the right solution, in the above case the formula which was in the mind of the problem setter.

Inductive bias in rule induction systems can be either a helpful or a harmful influence. The use of entropy in ID3-like systems was shown by Kononenko et al (1984) to introduce a bias towards the selection of attributes with larger ranges of values over those with smaller ranges. Quinlan (1986) discusses a new information theoretic measure which he calls the 'gain ratio' designed to compensate for this bias.

Two recent papers which discuss the issue of bias in information-theoretic measures further are White and Liu (1994) and Kononenko (1995).

3.4 Choosing a Set of Attributes

Choosing the attributes by means of which a set of objects in a domain is described generally involves a process of abstraction from the real world, and thus a loss of some information compared with the full complexity of the world.

The only requirement of the ID3 rule induction algorithm is that the set of attributes chosen has to be adequate, i.e. that no two instances with the same values of all the attributes may belong to different classes.

There will generally be many different ways in which an adequate set of attributes may be chosen and all of these will lead to an eventual decision tree (at least if windowing is not being used). However, with a poor choice of attributes, a large number of rules (almost equivalent to a case-by-case tabulation) are likely to be produced with little or no predictive power.

Although a good choice of attributes is likely to be of crucial practical importance, this may be extremely difficult to make, even for experts. Experts are obviously unlikely to make major errors in their choice of attributes, but it is perfectly possible that at least one important (but not entirely obvious) attribute may be inadvertently omitted from the set chosen or unimportant attributes may be included. Such relatively minor errors of judgment by a

subject expert could easily go entirely unnoticed, with a decision tree produced that looks reasonable but in fact has little or no predictive value.

To take an extreme example, supposing that in the 'club membership' training set shown in Table 2 (Section 3.3), the crucial attribute 'sex' were omitted from the attribute set. The remaining attributes still form an adequate set, and the induction algorithm will still produce a decision tree, such as Figure 4.

However, assuming that a student's sex is what really determines his or her team membership, it is obvious that the rules induced without using that attribute will have little or no predictive value.

The two rules derived from Figure 4 which conclude that the class is 'netball' are:

if eyecolour = brown and married = yes
and hairlength = short then Class = netball

if eyecolour = brown and married = no
and hairlength = long then Class = netball

whereas the single such rule in Figure 5 is:

if sex = female then Class = netball

There is clearly no sense in which the combination of attribute values

eyecolour = brown and married = yes
and hairlength = short
or
eyecolour = brown and married = no
and hairlength = long

is genuinely equivalent to 'sex = female', but that happens to be the case for the training set under consideration. Effectively, in this example where the attribute 'sex' is assumed to have been omitted from the training set given in Table 2, the induction algorithm has found a 'circumlocution' for a value of a crucial but omitted attribute. However, this can hardly be of general applicability or useful for predicting the class membership of instances not in the training set.

Although the problem of an important attribute being omitted is unlikely to arise with the most important attributes in a domain, it is quite probable that it will do with less important attributes and may reduce the value of the rules generated quite considerably.

3.4.1 Constructive Induction

One means of avoiding problems associated with important attributes not being present in the training set is the use of Constructive Induction to create new attributes automatically from existing ones for subsequent use in the induction process. There are two principal approaches to this.

Hypothesis-driven Constructive Induction (HCI) is based on analysis of previously generated inductive hypotheses. Potentially useful concepts in generated rules are extracted and used to define new attributes which, it is hoped, will prove valuable because they explicitly express hidden relationships in the data.

Data-driven Constructive Induction (DCI) methods construct new attributes based on an analysis of the training data itself, for example by combining attributes using a set of arithmetic or Boolean operators.

The two approaches to Constructive Induction are embodied in AQ17-HCI (Wnek and Michalski, 1993) and AQ17-DCI (Bloedorn and Michalski, 1991) respectively.

3.5 Types of Attribute

It is helpful to distinguish between (at least) two different types of attribute, those which are (a) nominal (or 'categorical') and (b) ordinal. Most of the attributes included in a training set are likely to be of one or other of these types, and both can pose problems.

(a) Nominal Attributes

These are attributes which take a (usually small) unordered finite set of possible values, such as sex (with values male and female) or country_of_birth (with values England, Scotland, Wales, Ireland, France and Other). Although the values of such attributes may sometimes be represented by integers for convenience (e.g. country_of_birth = 1, 2, 3, 4, 5 or 6), it is obviously meaningless to use these numbers in conjunction with inequality relations (such as country_of_birth>3).

An important special case of a nominal attribute is one such as name or reference number where every instance has a different value of the attribute. Such an attribute can be used as a unique key (e.g. if reference_number=287 then Class=A) but rules including this kind of attribute will have no predictive power whatsoever and such attributes should never be chosen to split on.

(b) Ordinal Attributes

Ordinal attributes are those such as age (in years), where the possible values form an ordered set. Performing a full split on such attributes, and thus generating rules with components such as age=23, does not seem desirable since no single value of age is likely to be particularly meaningful. An inequality relation such as age>70 may be meaningful, but it is likely to be better to group the attribute values in a meaningful way, such as child, young, middle-aged and old, which effectively converts an ordinal into a nominal attribute.

A common way of dealing with attributes of different kinds, especially those with continuous values, is to convert them to a number of equivalent binary attributes by a technique known as 'optimal splitting'. The method is described by Liu and White (1994).

3.6 Dealing with Noise

The basic ID3 algorithm assumes that all attribute values are known and exact and that all the classifications are correct. However, in many real-world domains it is usual for there to be 'noise' in the data, i.e. for attribute values or classifications to be incorrect.

The most likely outcome of noise in the training set is for the normal 'adequacy' condition to be violated and clashes to occur between two or more instances which have identical attribute values but which belong to different classes. the standard way of dealing with clashes in a noise-free training set is to introduce additional attributes which discriminate between the clashing instances. In large noisy domains this approach can lead to very large decision trees which to a substantial extent are merely fitted to noise in the data.

Quinlan (1986) describes two modifications to ID3 to deal with noise. Clashes are to be resolved by 'probabilistic induction' whereby if a terminal node is generated which has instances belonging to more than one class, the value of the most probable class is assigned to the node. In addition a statistical test is used as a 'stopping rule' to prevent further attributes being added whilst a decision tree is being constructed.

3.7 Incremental Induction

ID3 is a non-incremental learning system, i.e. it requires all of its training set to be available at once.

However, in many learning situations new data will unavoidably become available at intervals (e.g. in the case of medical data as new patients present with particular symptoms). Given additional data the ID3 algorithm would have to start again and build a new tree incorporating all the data then available. Although in some situations the newly built tree can be radically different from the original tree (e.g. if a new attribute is selected at the root) in most cases the effect of a single new instance will only lead to a minor change to part of the tree and it seems wasteful to have to recalculate the entire tree for such a relatively minor gain.

Incremental induction systems are ones which can incorporate training instances after learning has already taken place without recalculating the entire tree.

ID4 (Schlimmer and Fisher, 1986) maintains counts at each node in the original decision tree (number of positive and negative instances of each attribute etc.). When subsequently a new instance arrives the counts are updated and a check is made to see whether the attribute selected at each part of the tree is still the most suitable for partitioning on. If it is not, the subtree is rebuilt.

A revised version of this algorithm is implemented in ID5 (Utgoff, 1988).

3.8 Tree Pruning

A common problem encountered when generating decision trees is that even with a very large training set, whereas some branches may correspond to (say) 10% of the training set, some branches (especially deeper ones in the decision tree) are inevitably likely to correspond to only a small number of instances. These will often result in erroneous classifications when applied to instances not in the training set.

Tree pruning methods based on statistical significance tests are often used to reduce a full-size tree to one that is smaller but likely to give a better classification performance. Mingers (1989b) gives a detailed review of these methods.

3.9 Predictive Value of Rules: Complete Versus Partial Training Sets

There are two significantly different reasons for using an induction algorithm to induce rules from the instances in a training set: data compression and prediction. It is necessary to distinguish between the two cases discussed in Section 3.3.

(a) The training set is complete and the task is to find an efficient way of discriminating amongst the various classes. For this case it is not particularly important (although it may be desirable) for the rules to be meaningful to an outside human observer. It is enough for them to be correct, i.e. to produce the correct classification for every instance in the training set, and to be a more compact representation of the data than the original training set. The induced rules can then replace the training set in subsequent use, frequently giving a considerable 'compression' relative to the raw data in the training set.

(b) The training set is a partial one which forms a sample from a much larger set of possible instances. Here it is not generally of prime importance to find rules to compress the data. Rather, the task is to find a set of rules which can be used to predict with a high degree of reliability the class to which an object not included in the training set would belong, solely on the basis of the instances which were used to construct the decision tree.

The distinction between complete and partial training sets - essentially that between 'closed' and 'open' worlds - is crucial in considering induction algorithms.

Paradoxically, one of the weaknesses of ID3 in the open world case is that the method is too powerful: it can generate rules where they do not exist. For example, given a small training set of weather data (temperature, rainfall etc.) for (say) 20 days, it may well be possible to find an adequate set of attributes but weather-forecasting rules induced from the data are likely to be meaningless.

For any given training set, there are many possible decision trees which can be constructed. However, most of these are likely to have a low predictive value. The problem is not to find a corresponding decision tree but to find the best possible decision tree, i.e. the one which best captures the underlying causality of the domain.

Returning to the 'club membership' training set discussed previously, it is surely obvious that whether or not a student belongs to the netball team cannot really be related to the colour of his or her eyes. Finding rules which make such evidently spurious connections is of no predictive value whatsoever.

It is probable that the skill (and commonsense) of the users of ID3 and related systems can be relied on to prevent such obviously worthless decision trees as Figure 4 being constructed (or at least used). However, there are likely to be more subtle cases where spurious decision trees are generated which may be far less easy to detect, especially if the tree is large.

In general, an induced decision tree is only likely to have high predictive power if:

- **the attributes are well-chosen**
- **a good choice is made of the attribute to split on at each stage**
- **the training set is large and includes critical cases.**

From the discussion in earlier sections, it is apparent that each of these conditions is problematic.

3.10 Decision Trees Versus Decision Rules

In a PhD project at the Open University, supervised by the present author, Cendrowska (1987, 1989) strongly criticised the principle of generating decision trees which can then be converted to decision rules, compared with the alternative of generating decision rules directly from the training set.

Using notation such as (e.g.) a_1 to denote that the value of attribute a = 1 and δ_1 to denote that the class value = 1, she comments as follows.

> **"[The] decision tree representation of rules has a number of disadvantages. Firstly, decision trees are extremely difficult to manipulate - to extract information about any single classification it is necessary to examine the complete tree, a problem which is only partially resolved by trivially converting the tree into a set of individual rules, as the amount of information contained in some of these will often be more than can easily be assimilated. More importantly, there are rules that cannot easily be represented by trees.**
>
> **Consider, for example, the following rule set:**
>
> **Rule 1:** $a_1 \wedge b_1 \rightarrow \delta_1$
> **Rule 2:** $c_1 \wedge d_1 \rightarrow \delta_1$
>
> **Suppose that Rules 1 and 2 cover all instances of class δ_1 and all other instances are of class**

δ_2. These two rules cannot be represented by a single decision tree as the root node of the tree must split on a single attribute, and there is no attribute which is common to both rules. The simplest decision tree representation of the set of instances covered by these rules would necessarily add an extra term to one of the rules, which in turn would require at least one extra rule to cover instances excluded by the addition of that extra term. The complexity of the tree would depend on the number of possible values of the attributes selected for partitioning. For example, let the four attributes a, b, c and d each have three possible values 1, 2 and 3, and let attribute a be selected for partitioning at the root node. The simplest decision tree representation of Rules 1 and 2 is shown in [Figure 6].

Figure 6. Taken from Cendrowska (1989)

The paths relating to class δ_1 can be listed as follows:

1. $a_1 \wedge b_1 \rightarrow \delta_1$
2. $a_1 \wedge b2 \wedge c_1 \wedge d_1 \rightarrow \delta_1$
3. $a_1 \wedge b3 \wedge c_1 \wedge d_1 \rightarrow \delta_1$
4. $a_2 \wedge c_1 \wedge d_1 \rightarrow \delta_1$
5. $a_3 \wedge c_1 \wedge d_1 \rightarrow \delta_1$

Clearly, the consequence of forcing a simple rule set into a decision tree representation is that the individual rules, when extracted from the tree, are often too specific (i.e. they reference attributes which are irrelevant). This makes them highly unsuitable for use in many domains."

Cendrowska's own system PRISM induces conjunctive classification rules directly from the training set using an information theoretic measure of the information gain associated with the selection of each possible attribute-value pair during the rule induction process.

This approach would appear to be a promising alternative to the standard techniques of inducing decision trees and Cendrowska reports good results in a series of comparative trials against ID3.

Classification systems which generate decision trees include ID3, CART and C4.5. Systems which induce rules directly include PRISM, ITRule, CN2 and Cupid.

3.11 Using Domain Knowledge to Aid Rule Induction

One approach to avoiding the possibility of an induction system generating rules that are meaningless (see Section 3.9) is to find ways to make use of background knowledge of the domain under consideration that is not included in the training set itself.

Cupid (Mallen, 1995) seeks to improve the effectiveness of the rule induction process (based on the information theoretic J-measure) by making use of two types of domain knowledge.

(a) Attribute values may be grouped together into generalisation hierarchies in order to reduce the amount of search required. For example, towns and cities may be grouped into 'higher' geographical locations such as North and South.

A generalisation hierarchy effectively imposes a partial ordering on the values taken by the attribute. For example, an attribute 'geographic location' may have values such as London, Birmingham, Aberdeen, Plymouth etc. The user can impose a generalisation hierarchy on these values, producing a clustering of attribute values into ranges of interest. For example, locations may be grouped under north/south divisions or coastal, inland or riverside etc. Another example, relating to an 'occupation' attribute, is shown in Figure 7 below.

Figure 7. Taken from Mallen (1995)

(b) The user can define useful combinations of attributes, called intensional attributes in Cupid, which can then be used in the rule generation process. For example, in a medical domain the user may combine the attributes Height, Weight and Diagnosis to form an intensional attribute definition

IF Height=tall AND Weight=heavy AND Diagnosis=fracture
THEN multiple-fracture-likely=true

'Multiple-fracture-likely=true' can then be used as effectively a new attribute in forming further rules such as

IF multiple-fracture-likely=true and age>65 THEN time-in-plaster=2 months.

The effectiveness of these two uses of domain knowledge are analyzed in detail in a series of experiments in (Mallen, 1995). The results would seem to suggest that the use of domain knowledge can be a valuable way of increasing system efficiency and accuracy. One important finding of the work is that whereas good domain knowledge can produce an improvement in the predictive value of the generated rules over those produced by a 'knowledge-free' version of the Cupid system, poor domain knowledge can be identified as such by the system and does not lead to a significant degradation in performance.

4. Summary and Conclusions

Information about some of the systems mentioned in Section 3 is summarised in Table 3 below for convenience.

	ID3	IT Rule	CN2	CART	Prism	Cupid	C4.5	Clem-entine	Know-ledge Seeker
Conjunctive Propositional Rules?	_	_	_	_	_	_	_	_	_
Classification Rules in Form of Decision Tree (T) or Production Rules (R)?	T	R	R	T	R	R	R and T	R and T	T
Classification Rule Induction?	_		_	_	_	_	_	_	_
Generalised Rule Induction?		_		_		_	_	_	_
Probabilistic Rules?			_						_
Information Theoretic/ Statistical Measure Used E entropy J J measure P probability of attribute-value pairs C Chi Square G Gini Coefficient	E	J	E	G	P	J	E	E	C

Table 3. Summary of Information about Selected Induction Systems

In the preceding sections the techniques of automatic induction of classification rules from a training set of examples have been illustrated and the associated technical issues discussed in the context of the ID3 algorithm and its derivatives.

Although a growing number of practical successes testify to the considerable commercial potential of this approach it is clear that there are still significant areas of difficulty associated

with the use of rule induction techniques. However, these do not seem to have prevented their successful use in a number of practical applications, probably because the skill and experience of users of the system has enabled them to bypass potential areas of difficulty. This skill could be applied in several ways, e.g. by making a good choice of attributes or by choosing an appropriate window size or simply by selecting suitable problems to tackle.

As rule induction algorithms become better established and increasingly packaged into systems for commercial sale they are likely to be used increasingly by those without the experience or technical knowledge to avoid the pitfalls of the methods embodied in them. In unskilled hands, rule induction algorithms can easily produce rules that are entirely meaningless and of no predictive value whatsoever.

Without wishing to discourage the use of techniques which are likely to prove of considerable commercial value this may be an appropriate time to issue a warning about the problems which can result from their use by those without sufficient experience and expertise to alert them to the technical pitfalls.

5. References

Bloedorn, E. and Michalski, R.S. (1991). Constructive Induction from Data in AQ17-DCI: Further Experiments. Reports of the Machine Learning and Inference Laboratory, MLI 91-12, Centre for AI, CMU, 1991.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). Classification and Regression Trees. Monterey, CA: Wadsworth.

Buntine, W. and Niblett, T. (1992). A Further Comparison of Splitting Rules for Decision Tree Induction. Machine Learning, 8, 75-86.

Cendrowska, J. (1987). PRISM: An Algorithm for Inducing Modular Rules. International Journal of Man-Machine Studies, 27, 349-370.

Cendrowska, J. (1989). Knowledge Acquisition for Expert Systems: Inducing Modular Rules from Examples. PhD Thesis, The Open University.

Clark, P. and Niblett, T. (1989). The CN2 Induction Algorithm. Machine Learning, 3, 261-283.

Hunt, E.B., Marin, J. and Stone, P.J. (1966). Experiments in Induction, Academic Press.

Kononenko, I. (1995). On Biases in Estimating Multi-Valued Attributes. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montreal, Canada, 1034-1040.

Kononenko, I., Bratko, I. and Roskar, E. (1984). Experiments in Automatic Learning of Medical Diagnostic Rules, Technical Report, Jozef Stefan Institute, Ljubljana, Yugoslavia.

Liu, W.Z. and White, A.P. (1994). The Importance of Attribute Selection Measures in Decision Tree Induction. Machine Learning, 15, 25-41.

Mallen, J.I. (1995). Utilising Incomplete Domain Knowledge in an Information Theoretic Guided Inductive Knowledge Discovery Algorithm. PhD Thesis, University of Portsmouth.

Mallen, J.I. and Bramer, M.A. (1994). CUPID - An Iterative Knowledge Discovery Framework. In Research and Development in Expert Systems XI, SGES Publications, 1994.

Michalski, R.S. and Chilausky, R.L. (1980). Knowledge Acquisition by Encoding Expert Rules Versus Computer Induction From Examples: A Case Study Involving Soybean Pathology. International Journal of Man-Machine Studies, 12, 63-87.

Michie, D. (1990). Machine Executable Skills from "Silent" Brains. In Research and Development in Expert Systems VII. Cambridge University Press, 1990.

Mingers, J. (1989a). An Empirical Comparison of Selection Measures for Decision Tree Induction. Machine Learning, 3, 319-342.

Mingers, J. (1989b). An Empirical Comparison of Pruning Measures for Decision Tree Induction. Machine Learning, 4, 227-243.

O'Keefe, R.A. (1983). Concept Formation from Very Large Training Sets. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Morgan Kaufmann.

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

Quinlan, J.R. (1979). Discovering Rules by Induction from Large Collections of Examples. In D.Michie (Ed.), Expert Systems in the Microelectronic Age, Edinburgh University Press, 168-201.

Quinlan, J.R. (1983). Learning Efficient Classification Procedures and their Application to Chess Endgames, In Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.). Machine Learning: An Artificial Intelligence Approach, Tioga Publishing Company.

Quinlan, J.R. (1986). Induction of Decision Trees. Machine Learning, 1, 81-106.

Quinlan, J.R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann.

Schlimmer, J.C. and Fisher, D. (1986). A Case Study of Incremental Concept Induction. Proceedings of the Fifth National Conference on Artificial Intelligence. Morgan Kaufmann, 496-501.

Shapiro, A.D. (1983). The Role of Structured Induction in Expert Systems, PhD Thesis,

University of Edinburgh.

Smyth, P. and Goodman, R.M. (1992). An Information Theoretic Approach to Rule Induction from Databases. IEEE Transactions on Knowledge and Data Engineering, 4, 301-316.

Struhl, S. (1995). Using Classification Tree Analysis: a Review of the Method and a new Software Package for CHAID/CART. Quirk's Marketing Research Review, November 1995.

Utgoff, P.E. (1988). ID5: An Incremental ID3. Proceedings of the Fifth International Conference on Machine Learning, Michigan, USA. Morgan Kaufmann, 1988, 107-120.

White, A.P. and Liu, W.Z. (1994). Bias in Information-Based Measures in Decision Tree Induction. Machine Learning, 15, 321-329.

Wirth, J. and Catlett, J. (1988). Experiments on the Costs and Benefits of Windowing in ID3. Proceedings of the Fifth International Conference on Machine Learning, Michigan, USA. Morgan Kaufmann, 1988, 87-99.

Wnek, J. and Michalski, R.S. (1993). Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments. Machine Learning, 1993.