The Knowledge Engineering Review, Vol 6:4, 1991, 307–333

Principles of induction and approaches to attribute based induction

G. KALKANIS[1] and G. V. CONROY

Department of Computation, University of Manchester Institute of Science and Technology, Manchester M60 1QD, UK

Abstract

This paper presents a survey of machine induction, studied mainly from the field of artificial intelligence, but also from the fields of pattern recognition and cognitive psychology. The paper consists of two parts: Part I discusses the basic principles and features of the machine induction process; Part II uses these principles and features to review and criticize the major supervised attribute-based induction methods. Attribute-based induction has been chosen because it is the most commonly used inductive approach in the development of expert systems and pattern recognition models.

1 Introduction

Over the years of development of information technology, a number of tasks have been considered for automation such as character recognition, image analysis, medical diagnosis, structure elucidation for molecules in chemistry, speech recognition, and others. In order to automate any of these tasks, it is necessary to endow a machine with a large body of knowledge about the problem domain. The knowledge should be encoded in a systematic and precise way, and should also integrate well with other possibly preexisting knowledge, so that it will enhance the machine skills.

However, the task of knowledge acquisition and knowledge integration has been recognized to be a bottleneck in the development of any automatic recognition program (Feigenbaum, 1981) (e.g., for expert systems). The main difficulty is that human experts are often unable to make explicit their domain expertise in a manner precise enough to enable the transfer of expertise from the human mind to the machine's memory.

On the other hand, recent progress in data processing technology has made the accumulation and systematic organization of large volumes of data a routine activity. Therefore, the automatic development and refinement of knowledge models is now possible when the machine is given as input a large (or, sometimes, even a small) amount of data that express instances of a certain task domain. In other words, the machine may be endowed with an inductive learning component that automatically translates data into knowledge. This knowledge may be used either to build a model on which the recognition of new instances will be based, or to refine and update a pre-existing recognition model.

The automated induction problem had been considered by the early 1950s in the field of statistical pattern recognition (Chow, 1957). Later, progress in the field of artificial intelligence made it possible to build systems that extracted knowledge from data, and this knowledge was represented in an explicit symbolic form which was more humanly understandable than a complex mathematical model (Winston, 1975). The inductive knowledge extraction approach has been successfully applied to various real world problems (such as the development of cleavage rules for

[1]The author acknowledges the financial support for this work provided by the State Scholarships Foundation of Greece.

molecules in chemistry [Buchanan & Mitchell, 1978], soyabean disease diagnosis [Michalski & Chilauski, 1980], credit card assessment [Carter & Catlett, 1987], routing of steel products in a job shop [Buntine & Stirling, 1989]), and is currently envisaged as a routine commercial practice for problem domains which are very complex, and for which any humanly supplied knowledge would be incomplete and unreliable (Michie, 1988).

PART I: PRINCIPLES OF INDUCTION

Here we aim to establish a framework useful for the study of various inductive methods (such methods are examined in Part II).

Section I.1 attempts a brief review of the concept of induction, as expressed by various philosophers, and a general formulation of the machine induction problem. Section I.2 emphasizes the necessity of inductive bias as a means of making induction a tractable process. Section I.3 presents a mathematical (probabilistic) formulation of induction. Section I.4 focuses on various data representation formalisms. Section I.5 briefly discusses the major inductive learning paradigms. Finally, section I.6 deals with dimensions that characterize inductive algorithms, such as incrementality, experimentation capability and search strategies.

I.1 Characterization of induction

The concept of *induction*[2] dates back to Aristotle (5th century B.C.). Since then, many authors have suggested various definitions of induction. According to the most important of these definitions, induction can be regarded as:

- *Generalization* by complete enumeration of all the objects that this generalization subsumes (Aristotle).
- *Theory formation* from observational facts (Francis Bacon).
- *Concept formation* by aggregation of objects into classes. Each class of objects has an intension and an extension. The *intension* consists of the properties that an object should possess in order to be a member of a class. The *extension* consists of the objects that possess these properties (empiricists, such as Locke and Hume).
- The operation of discovering and providing empirical propositions (John Stuart Mill).
- The inverse process of deduction (Jevons).
- Generalization from a number of instances, by considering the positive and negative analogies between them (Keynes, 1921).
- Information compression (Watanabe, 1972).
- Determination of a model that prescribes decisions for the undertaking of certain actions. The model should be in accordance with the behaviour of the input instance (behavioural interpretation of induction [Newman, 1957]).

The adoption of different definitions leads to different formulations of the inductive learning problem. For the purposes of this paper, inductive learning will be formulated as:

Given

- A set of data[3] described in a certain instance language
- Background knowledge that defines the language of the hypotheses to be generated, the quality criteria used to choose among competitive hypotheses, and any other constraints imposed on the data set and on the candidate hypotheses

[2]Induction is the translation of the Greek word *επαγωγή*

[3]Synonymous words are *instances*, *examples*, *events*, *observations*.

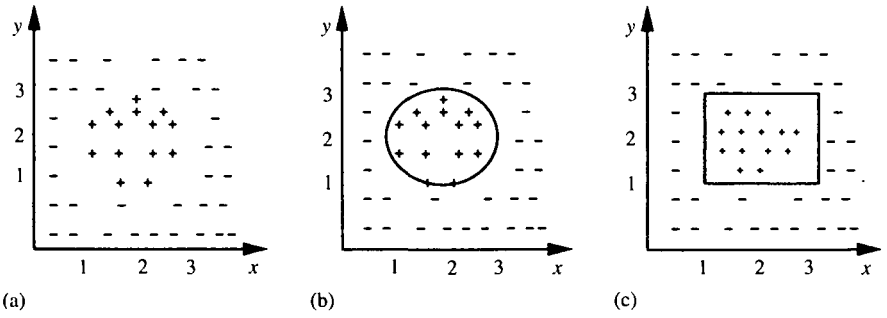

Figure 1 An illustration of the inductive bias. If the aim of induction is to produce a rule which determines whether a randomly chosen point in the above two-dimensional board, is positive (+) or negative (−) (on the basis of the data set given in Figure 1a), then there are two alternative hypotheses that perfectly fit the above data set:

(i) **if** (x,y) belongs to the circle with centre (2,2) and radius $R = 2$ **then** point (x,y) is (+) **else** point (x,y) is (−) (Figure 1b)
(ii) **if** $[1 \leq x \leq 3]$ **and** $[1 \leq y \leq 3]$ **then** point (x,y) is (+) **else** point (x,y) is (−) (Figure 1c)

The choice of one of the above hypotheses is the result of the inductive bias. If the bias favours the preference of circle-based hypotheses, then hypothesis (i) has to be chosen. If the bias is in favour of rectangles then the inductive algorithm chooses hypothesis (ii)

Determine

- A hypothesis that strictly or weakly fits the input data[4] and also satisfies the background knowledge.

I.2 The role of the inductive bias

Induction is not a process deterministically producing a unique outcome. It is possible for a given set of data to induce more than one hypothesis which fits that set. This issue introduces ambiguity into inductive inference, and the question of which hypothesis should be finally chosen from the set of well-fitted hypotheses. The problem was initially made explicit by Mitchell (1980), who introduced the concept of *bias* as any basis for choosing one hypothesis over another, other than the degree of fit with the input data.

Utgoff (1986) identified two major sources of bias:

1. *Linguistic bias*: this bias is imposed by the language(s) used to represent the data and the hypotheses to be created, and also by the background knowledge.
2. *Algorithmic bias*: this bias is due to the nature of the algorithm used, whereby search restrictions are placed on the space of the candidate hypotheses, in order to make induction tractable and efficient.

Moreover, there are two important features of the inductive bias:

The strength of bias expresses the degree of restriction that a bias imposes on the search for a hypothesis. the stronger the bias, the more restrictions it imposes on the number of candidate hypotheses that have to be examined. The strength of bias can be formulated as the sum of that due to the linguistic source plus that due to the algorithm (Rendell, 1986).

The correctness of bias A bias is said to be correct if it includes the target hypothesis in its scope.

An example that illustrates the role of the inductive bias, is given in Figure 1.

[4]The fit of a hypothesis to the data set is strict when all the data are consistent with the hypothesis, while the fit is weak when a high proportion of data (not necessarily all) is consistent with the hypothesis.

In practical induction problems the input data are expressed in a definite (instance) language; sometimes, however, it is not known whether this language is adequate to completely specify the problem domain and/or whether there are errors(noise) in the data set. What is available is an algorithm that induces hypotheses from data; but it is still unknown whether the linguistic and algorithmic biases are correct. The bias pertaining to an inductive system might be:

(i) Fixed (Mitchell, 1982);
(ii) adjusted by user supplied parameters that define preference criteria (Michalski, 1983);
(iii) Shifted during hypothesis formation in order to achieve better quality descriptions (Utgoff, 1986; Schlimmer, 1987; Pagallo, 1989);
(iv) Decided by consideration of problem characteristics in relation to previous problems, and by selection of the most appropriate bias type from a set of available alternatives (this approach assumes the existence of a Variable Bias Management System (VBMS) as described in Rendell et al., 1987).

I.3 Mathematical formulation of induction

Following the previous section, induction can be regarded as an inference process over a space of candidate hypotheses which are constrained by the input data as well as by the inductive bias. To model the inductive inference problem, one can define a probability function[5] over the hypothesis space $\mathcal{H}$. In this way, induction becomes the determination of a hypothesis $H \in \mathcal{H}$ that maximizes the probability $P(H/E,B)$, *where* E is the evidence provided by the input data set, and B is the bias.

According to Bayes law, $P(H \mid E,B) = P(E \mid H,B).P(H \mid B)/P(E \mid B)$. The term $P(E \mid B) = \sum_{H \in \mathcal{H}} P(E \mid H,B).P(H \mid B)$, does not depend on any particular hypothesis H, and will be treated as a constant.

The term $P(E \mid H,B)$ expresses the probability (deductive) of obtaining evidence E (arising from the input data set) given the hypothesis H (with respect to bias B). The choice of the evidence criteria drawn from the input data set varies from one problem to another. Nevertheless, a fundamental source of evidence is the observed error of the hypothesis H (as observed over the input data set). The evidence E can be expressed as $E = (\hat{e},f)$, where $\hat{e}$ is the observed error and f represents the collection of any other source of evidence derived from the input data (e.g., similarities between them). The f component of evidence is particularly important in unsupervized learning (clustering) approaches, where the input instances do not have a class label; in that case, there is no observed error $\hat{e}$ to calculate.

The term $P(E \mid H,B)$ can be rewritten as $P(E \mid H,B) = P(\hat{e},f \mid HB) = P(\hat{e} \mid H,B).P(f \mid \hat{e},H,B)$. The probability $P(\hat{e} \mid H,B)$ expresses the reliability of hypothesis H and is high when the observed error of this hypothesis is close to the real error of it (with respect to the target concept). This aspect has not been made explicit by most existing inductive systems. Moreover, inductive methods require that the error of the induced hypothesis is low. A sufficient condition for achieving hypotheses with low real error is to choose among the hypotheses with high reliability the ones that have low observed error on the input data set.

The term $P(H \mid B)$ is the prior probability of hypothesis H (with respect to bias B) which expresses subjective preference criteria for one hypothesis over another, and is independent of any particular data set. These criteria are called "extra-evidential" by some authors (Watanabe, 1985; Rendell, 1986). It is worth mentioning that the subjectivity of the prior probability has been disputed by authors like Keynes (who believed that the only factors that determine the prior probability are the similarities and differences observed among the input data) (Keynes, 1921), and Carnap (who believed that the only important factor in the determination of the probability

[5]The model discussed here presents the authors' views on the induction task. Many other probabilistic formulations have been reported in the literature (Mortimer, 1988).

$P(H \mid E)$ is the causal relation between H and E expressed by the probability $P(E \mid H)$) (Carnap, 1950).

I.4 Representation formalisms

In induction, when speaking about representation, one refers to:

- The formalism used to describe the input data.
- The formalism(s) used to describe the output hypotheses, as well as any intermediate hypotheses considered during the induction process.
- The formalism(s) used to describe the background knowledge.

The background knowledge is either implicit to the instance and hypothesis languages, or can be described in any of the hypothesis representation formalisms; background knowledge representation formalisms are not, therefore, considered separately.

Hypothesis representation formalisms are common to those used in other artificial intelligence applications. Such formalisms include logic, graph-based representations, production rules, frames, algebraic expressions and grammars. This paper will not discuss hypothesis representation formalisms, since they have been adequately examined elsewhere (Kocabas, 1991).

I.4.1 Data representation formalisms

I.4.1.1 Attribute-based descriptions

In an attribute-based domain, every instance is represented as a vector of attribute values. Values may belong to an unordered, partially ordered or totally ordered set; furthermore, the value set may be finite, numerably infinite (i.e. integer numbers) or innumerably infinite (e.g., real numbers)[6]. Attributes define global properties of an object, and are used to connect the meaning (the intension) of a concept to the objects (the extension) that fit that meaning (Mervis & Rosch, 1981). Inference over an attribute space consists of assessing similarity among instances according to their syntactic distance. However, such "syntactic" similarity may not be semantically sound if the attributes are not the necessary and sufficient ones with which to describe the target concept. Therefore, a background theory is additionally needed that indicates how similarity has to be assessed on the basis of the domain attributes (Medin & Wattenmaker, 1986). The attribute-based representation has been adopted by the statistical pattern recognition approaches (Devijver & Kittler, 1982), and also by many artificial intelligence-based inductive systems (Gams & Lavrac, 1987).

I.4.1.2 Higher order descriptions

Attribute-based representations are awkward for describing domains in which objects are composite structures of various components. This aspect was emphasized by the *structuralists* at the beginning of this century; they believed that objects are decomposable into more elementary parts that provide explanations for them (Medin & Wattenmaker, 1986). Such a view has been adopted by the syntactic pattern recognition paradigm (Fu, 1974), in which objects are represented as composite structures (of atomic components) that are generated with respect to a grammar. The introduction of artificial intelligence techniques into inductive learning accounted for a variety of data representations, such as first order logic (Hayes-Roth, 1974), semantic nets (Winston, 1975) and frames (Blythe, 1988). In particular, first order logic has become a very popular formalism for data representation, and has been used both for theoretical studies on the inductive learning

[6]For example, the attributes *colour* (whose possible values belong to the unordered set {blue,brown,red}) and *age* (whose possible values belong to the ordered set [0, 100]) define a two-dimensional attribute space. The vector (*blue*,18) is an instance of that attribute space, where [colour = blue] and [age = 18].

problem (Plotkin, 1971) and for the construction of inductive systems that induce clauses from data, such as the CIGOL (Muggleton, 1988), FOIL (Quinlan, 1989 b) and GOLEM (Muggleton, 1991) systems.

I.5 Induction in machine learning

Machine learning is concerned with developing computational theories of learning and constructing learning systems. Learning refers to the ability of a human (or machine) to increase its knowledge and improve its skills. Induction is a way of learning, and has received considerable attention in the machine learning community. This section outlines the two major inductive learning paradigms, namely *Supervised Inductive Concept Learning* and *Unsupervised Inductive Concept Learning*. Other machine learning paradigms that use induction to some extent, are[7] connectionist learning, genetic algorithms, learning search heuristics and learning by analogy.

I.5.1 Supervised inductive concept learning

In this paradigm the learner is presented with a set of instances where each instance is labelled with a unique class that belongs to a set of classes. The task is to develop a model that fits the input instances, satisfies any background knowledge provided, and can be used to classify new unlabelled instances.

The first approaches, based on statistical decision theory, appeared within the pattern recognition field (Sebestyen, 1962; Nilsson, 1965), and pursued the minimization of the misclassification rate. Classification models were mainly mathematically-based, such as discriminant functions, density estimation models and distance-based models.

In the early 1970s, there appeared artificial intelligence-based approaches (also called symbolic) (Michalski, 1973; Winston, 1975). These approaches, highly influenced by cognitive science, were mainly focused on issues such as knowledge representation and human understandability, while often requiring 100% consistency of the training data to the generated hypotheses (Mitchell, 1982). The output hypotheses were described by logic-based or network models rather than by mathematical expressions.

Supervised inductive concept learning is the most widely studied machine learning paradigm, and includes problems such as prediction of a law that governs a sequence of observations (Angluin & Smith, 1983; Dietterich & Michalski, 1985) and inference of a grammar from positive and negative instances of it (Bierman & Feldman, 1972).

I.5.2 Unsupervised inductive concept learning

Unsupervised inductive concept learning can be divided into two major categories: scientific discovery and clustering.

The task of scientific discovery is the determination of a set of qualitative or quantitative laws that govern the generation of the input observations. Examples of scientific discovery systems are the AM system (Lenat, 1983) (that discovered laws of set and number theory), and the BACON system (Langley et al., 1984) (that discovered quantitative laws of chemistry).

In the clustering problem, the learner is given a set of unlabelled input instances and is asked to partition the instance set into groups (clusters) so that members of the same group will exhibit a high degree of similarity (Anderberg, 1973). There exist hierarchical clustering methods (that form a hierarchy of clusters at various levels of abstraction) and non-hierarchical ones (that form a "flat" partition of the input instances' set). The clusters can be presented either extensionally, by the instances corresponding to each of them, or they can be given intensional descriptions (e.g., each cluster can be expressed as a conjunctive concept); this latter type of clustering is called *Conceptual*

[7]These paradigms have been surveyed in detail in Kocabas (1991).

clustering (Michalski & Stepp, 1983; Fisher, 1987). Similarity between instances is assessed as a function of the descriptors used to specify them. Nevertheless, similarity may also take into account the conceptual coherence of instances according to the specific type of concept (e.g., conjunctive concepts), which is pursued in the case of conceptual clustering. Supervised inductive concept learning can itself be viewed as a clustering process which also takes into account the teacher provided class membership of the input instances.

I.6 Dimensions that characterize inductive algorithms

I.6.1 Incrementality

Single step inductive systems require all the training data to be available before the inductive algorithm is used. When new training data become available, these systems have to discard previously existing hypotheses and start the induction task from the beginning (with the enriched data set).

Alternatively, there are *incremental systems* that allow new instances to integrate with previously existing hypotheses by suitably updating these hypotheses without the need to discard them and start the induction process from the beginning. Incremental learning is inherent in the way humans learn; humans use a partial memory learning mechanism while keeping in mind some important events.

Single step inductive systems are more resistant to noise, but they are also slower when an update is required. Nevertheless, there have recently appeared incremental learning systems that exhibit good noise immunity because of the use of a numerical model that controls inductive rule generation (Schlimmer, 1987; Gross, 1988).

I.6.2 Experimentation capability

Many inductive algorithms passively receive fully specified instances in order to induce hypotheses. However, there are algorithms that possess active experimentation capability which allows them to propose partially specified instance descriptions to an oracle (which may be either the system user or another part of a more general system) and ask for the complete specification of them. Experimentation capability helps to speed up the learning process, and to receive precise information about instances whose concept membership is not clear (Gross, 1988). Furthermore, experimentation enlarges the scope of hypotheses' types for which formal convergence properties about their learning behaviour can be obtained (Amsterdam, 1988 a). Experimentation has often been used in systems that learn search heuristics. These systems generate their own problems and pass them to the heuristic problem solver which (by searching for a solution) provides positive and negative instances of the operators' application to the inductive algorithm.

I.6.3 Search strategies

Inductive learning can be viewed as the process of searching the space of candidate hypotheses that fit the input data and the background knowledge (Mitchell, 1982). Such a search is characterized by three dimensions:

1. *The strategy that controls hypotheses development.* The most common strategies encountered in the induction literature are depth first (Winston, 1975), breadth first (Hayes-Roth, 1974) beam search (Clark & Niblett, 1987) and best first (Muggleton, 1987).
2. *Induction by generalization vs. induction by discrimination.* Some systems start with the most specific hypotheses and gradually generalize them. Hypotheses obtained in this way give good characterizations of particular concepts, but they are susceptible to errors of omission (resulting from too specific generalizations) and errors of commission (resulting from over-

generalizations). On the other hand, some systems start with an overgeneral hypothesis and specialize it to discriminate between various concepts. Induction by discrimination is appropriate for probabilistic domains where there may be noice or incomplete input information. However, the results produced by discrimination systems may not always be humanly understandable.

3. *Data driven vs. model driven search.* Data driven systems use specific instance descriptions for the hypothesis generation process (e.g., the AQ system [Michalski, 1973] uses seed instances to form decision rules). Model driven systems suggest candidate hypotheses independently of the particular instances pertaining to the data set; then use the input data to evaluate these hypotheses and choose the best among them. For example, the CN2 system (Clark & Niblett, 1987) (one of whose intentions was to make a noise tolerant variant of AQ) considers all the candidate terms and evaluates the quality and significance of each of them according to the fit they exhibit with the input data set. Model driven systems are more appropriate for probabilistic and noisy domains.

PART II: APPROACHES TO SUPERVISED ATTRIBUTE-BASED INDUCTION

The general characteristics of the induction process, which were specified in Part I, will be illustrated in the supervised attribute-based induction approaches examined here. In the (supervised) attribute-based induction (ABI) problem, the learner is given a set of data (instances of the domain), each of these described by a vector of attribute values and labelled with a unique decision class that belongs to a set of classes. The aim of ABI is the construction of a hypothesis (which is called a *classifier*) that will be used to classify new, unlabelled, instances. A simplified example of the ABI process is provided in Figure 2.

We will view various approaches that have appeared in statistical pattern recognition and artificial intelligence, under a unified framework. Particular attention is given to the conceptual interpretation of these approaches rather than to the details or the jargon pertaining to specific systems.

Section II.1 discusses the major stages involved in attribute-based induction. Section II.2 indicates the difficulties of the ABI process. Section II.3 discusses various criteria employed to assess the quality of the induced concepts, such as misclassification error, concept simplicity, concept usability and ability of the concept to detect regularities pertaining to the problem domain. Section II.4 criticizes the most important methods developed in the field of statistical pattern recognition, such as discriminant analysis, exemplar-based methods and Bayesian classifiers. Section II.5 presents and comments on the characteristics of artificial intelligence-based (also called *symbolic*) methods, such as AQ and version spaces. Section II.6 surveys decision tree classifiers, which incorporate features of both statistical and symbolic methods. Section II.7 outlines the Valiant learning framework which attempts to determine convergence rates and to

Data Set:

age	job	hair	class
25	student	brown	+
18	student	red	+
32	secretary	blond	−
27	nurse	dark	−
34	doctor	red	+

Some Candidate Hypotheses:

1) IF [job = student] OR [hair = red] THEN [class = +] ELSE [class = −]
2) IF [job = student OR doctor] THEN [class = +] ELSE [class = −]
3) IF [hair = brown OR red] THEN [class = +] ELSE [class = −]
4) IF [age ≤ 25] OR [hair = red] THEN [class = +] ELSE [class = −]

Figure 2 A simplified example of attribute-based induction (Attributes: age, job, hair, class)

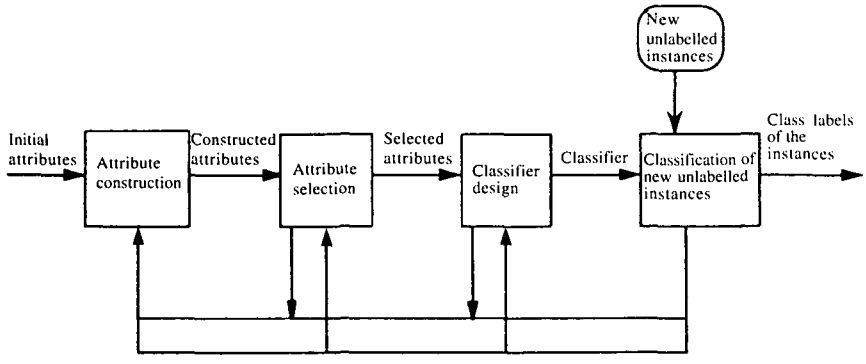

Figure 3 The attribute-based induction life cycle

analyse the complexity of ABI systems. Section II.8 discusses the applicability of attribute-based induction in knowledge engineering, and also the associated difficulties and limitations of it.

II.1 Stages of attribute-based induction

The ABI problem involves four stages (see also Figure 3).

STAGE 1: Attribute construction

The initial set of attributes may not be adequate to account for a good quality classifier, and a new attribute set, derived as a function of the initial attributes, may be needed. New attributes are constructed either by the human expert or automatically by the system. When the induction process involves the automatic construction of new attributes from the existing ones, it is called *constructive induction* (Michalski, 1983). constructive induction is useful in the following circumstances:

1. *To merge disjunctive regions in the original attribute space.* When a class of the domain is dispersed into many disjunctive (i.e., non-connected) regions of the original attribute space, it is very difficult and inefficient to discriminate all these regions by a hypothesis that only uses the initial attributes. Nevertheless, when these regions are systematically placed on the attribute space, it is possible to construct new, higher level attributes, that make explicit that systematic placement. This process has been called *disjunctive construction* (Rendell, 1988). Disjunctive construction simplifies the final hypothesis description, improves the performance of the resulting classifier over new instances, and helps in the prediction of new disjunctive regions pertaining to the original attribute space.
2. *To shift the bias implied by the initial attributes.* The initial attributes, together with the hypothesis representation language, impose a bias concerning the form of the output hypothesis. It ofen happens that the target hypothesis cannot be expressed in the given hypothesis representation language using the initial attributes, while the same hypothesis can be expressed in that hypothesis representation language when using new attributes constructed as functions of the initial ones. As an example, consider a two class domain (with classes + and −), where each instance is represented as a pair of two real valued attributes (x,y), and the output hypothesis should be represented as a conjunction of ranges of attribute values. If the target hypothesis for class + is a circle with centre (x_0,y_0) and radius R, then the initial attributes are unable to express that target hypothesis (w.r.t. the hypothesis representation language). However, if the new attribute z = *distance*$[(x,y)$ *from* $(x_0,y_0)]$ is constructed, the target hypothesis can be expressed in the required hypothesis representation language, as follows:

if $[z \leq R]$ **then** [class = +]
else [class = −]

3. *To compress a hypothesis into a more compact form.* This is necessary in large domains (e.g. chess endgames). In these domains the target hypothesis could be represented by the initial attributes. However, it would be highly desirable to have a more concise hypothesis description by using new, higher level attributes. Thus, the problem domain is structured from the initial attributes to the higher level ones, and eventually to the decision classes. In this context, constructive induction makes the output hypothesis more understandable to the human expert. Furthermore, it increases the efficiency of searching through that hypothesis, in order to classify new, unlabelled instances.

   An example of hypothesis compression is the structured induction approach (Shapiro, 1987). Structured induction creates a hierarchy of attributes in which the initial attributes lie in the terminal nodes. At each non-terminal node, the value of the corresponding attribute is obtained to be the class label resulting from the application of a classification rule which has been induced using as attributes those that correspond to the children of the current node. Therefore, induction takes place at each non-terminal node of the attribute hierarchy, in order to generate rules that express the association of lower level attributes to the higher level ones. The root of the hierarchy is associated with the set of the decision classes. Structured induction was a successful approach for complex chess endgames. However, the attribute hierarchy was created manually by the domain experts, and any attempt to automate it did not prove promising (Paterson, 1983).

   Other examples of hypothesis compression approaches are the DUCE system (Muggleton, 1987), which creates new terms when a common conjunction of attribute values appears in many instances or rules, and the MIRO system (Drastal et al., 1989) that minimizes the number of disjuncts in an AQ-like hypothesis description by amalgamating different attribute values into a common higher level attribute value through background knowledge rules that express attribute value dependencies.

STAGE 2: Attribute selection

The initial (or the constructed) attribute set contains unnecessary attributes whose presence introduces problems, such as high dimensionality and/or noise. In this case, the system (or, sometimes, the human expert) must select the attributes that are relevant for the design of the classifier. The problem of attribute selection was initially addressed by Lewis (1962), and various techniques for automatic selection of relevant attributes have been suggested by Jain and Dubes (1978), Devijver and Kittler (1982) and Baim (1988).

STAGE 3: Classifier design

This process (also called *learning*) is carried out automatically by performing induction on the set of the training instances of the problem domain. There are various design methods which will be discussed in the forthcoming sections. What is worth emphasizing is that the attribute construction and selection stages do not necessarily precede but often occur during the learning stage. Moreover, it is possible to iteratively perform induction by backtracking from a particular stage to a previous one. Any induced classifier should be tested on a testing data set in order to assess its performance. Furthermore, it may be necessary for the classifier to be refined and converted to a new representation formalism before it is used for the classification of unlabelled instances.

STAGE 4: Classification (recognition) of unlabelled instances

During this stage, the classifier is used for the class identification of new, unlabelled instances. The classification stage requires a rule interpreter that searches through the classifier to decide the class with which the unlabelled instance will be labelled. The rule interpreter should take into account the way classification rules have been generated from the training data.

The recognition process is often performed by an expert system that uses the classifier as its knowledge base, the rule interpreter as its inference engine, and which also contains a user

interface that enables the system to interact with the user and explain its reasoning. The expert system may, also, provide a facility for the modification of its knowledge base, when necessary.

II.2 Difficulties of attribute-based induction

ABI is not a simple process. The main problems encountered during ABI are:

1. *Inadequacy of the representation language to describe the target concept.* One way to alleviate this problem (which has been called *incorrect bias* in Part I) is to construct new attributes as functions of the initial ones.
2. Related to the previous problem is *the incapability of the current approaches to detect what type of classifier is most suitable for a particular application.* This makes ABI both an art (the selection of the most appropriate classifier is, still, intuitively performed by the user) and a science (in the methodological derivation of a classifier from a data set by the use of an inductive algorithm).
3. *Small size data sets and also biased data acquisition techniques* render classifiers unreliable.
4. *The presence of few attributes* (inadequate to account for perfect classification) *or too many* (that are redundant, irrelevant and increase, often exponentially, the complexity of the induction process) (Kanal & Chandrasekaran, 1971; Jain & Dubes, 1978).
5. *Noise in the data set* which is due to erroneous or unreliable measurements. There are two types of noise: noise in the attribute values and noise in the class label of instances. Noise is very difficult to theoretically detect and analyse (Boucheron & Sallatin, 1988). To overcome noise, systems use either statistical techniques (which aim at increasing the reliability of the classifier) or additional domain knowledge (Clark & Niblett, 1987).

II.3 Quality criteria for classifier assessment

II.3.1 Error

The *error* of a classifier can be defined as the probability of misclassification of a randomly chosen instance of the attribute space. This probability cannot be exactly computed, but can only be estimated from the input data set. The most commonly used error estimation methods are:

1. *Resubstitution estimate* estimates error on the basis of the percentage of misclassifications occurring in the training data set. This method has been found to be optimistically biased because the classifier has been constructed to fit the training data. However, if the number of data tends to infinity, the resubstitution error estimate converges to the true error of the classifier.
2. *Holdout estimate* estimates the error of a classifier by the percentage of misclassifications occurring to a test data set independent from the training set. This method has been found to be pessimistically biased (Toussaint, 1974). Another problem associated with the method is the determination of the optimum split of an input data set into training and testing sets. A common strategy is to use 70% of the input data for training and 30% for testing the resulting classifier (Breiman et al., 1984; Cestnik et al., 1987). However, for a particular problem, (Highleyman, 1962) found that the size of the test set should never be less than that of the training set for the error estimates to be reliable.
3. *Cross-validation estimate* partitions the input data into v equal size subsets, and each time one of these subsets is kept as test set, while the other subsets are used together for training a classifier. The above process is repeated v times, and the results, after testing the classifier on the corresponding test sets, are averaged. This method becomes unbiased as v increases; increasing the value of v, however, increases the computational cost (since v classifiers have to be trained), and the variance of the error estimate (Glick, 1978). A particular case of cross-validation is the

leave one out (or *U*) *method*, in which *v* equals the number of input data. For small samples, a method of sampling with replacement, called bootstrapping (Efron, 1983), is more appropriate for error estimation because of its low variance.

II.3.2 Simplicity

Simplicity refers to the structure of the classifier. Simple structures are preferred to complex ones because they tend to be better understandable to human beings (Michalski, 1983). Moreover, logic based descriptions (e.g., production rules) are far more understandable than mathematical models or complex graph representations. This is the reason why artificial intelligence methods often sacrifice statistical soundness and error minimization for the sake of simplicity. Nevertheless, the preference of simple hypotheses to complex ones (also known as Occam's razor or as the principle of parsimony) often increases classification accuracy on non-training instances (Michalski et al., 1986 b; Mingers, 1989 b; Blumer et al., 1987), because a complex hypothesis may be the result of overfitting the training set. Moreover, simplicity is affected by the inductive bias. If the inductive bias does not allow the target hypothesis to be included in the set of the candidate hypotheses, then the induced hypothesis is often complex and overfits the training data.

II.3.3 Usability

A necessary requirement for a classifier is for it to be effectively used when a new unlabelled instance is to be classified. Effective use implies optimization in the search performed in order to classify that new instance. Optimization is achievable by a well structured classifier (e.g., decision tree) and/or by an appropriate rule interpreter. To increase usability it is often necessary to convert an existing classifier into a more structured representation. Such an approach has been adopted by the ELIS system (Leigh, 1984) that transforms a set of production rules into a directed acyclic graph.

II.3.4 Regularity detection

The ideal aim of induction would be the determination of the target concept (the model that underpins the selection of the input data). However, this aim is unlikely to be realized, unless the learner possesses knowledge about the functional form of the target concept (e.g., the learner might have been told that the target concept is a conjunction of attribute values). Such knowledge is, in general, unavailable, but even its existence does not imply the possibility of identifying the target concept in polynomial time (Kearns et al., 1987).

A minor aim of induction is the detection of interactions occurring among the input attributes. Such interactions express regularities pertaining to the problem domain, and could be indicated through the structure of the classifier. However, one should be very cautious in interpreting a classifier's structure. In a series of experiments, Breiman et al. (1984) showed that tree classifiers with slightly different accuracy, gave rise to totally different tree structures, for the same problem.

II.4 Statistically oriented methods

These methods have been developed within the statistical pattern recognition paradigm (Nilsson, 1965; Duda & Hart, 1973; Devijver & Kittler, 1982), and their main characteristics are:

1. They are not flexible in handling a mixed set of discrete and continuous attributes. Thus, they either convert continuous attributes to discrete ones by grouping intervals, or they convert discrete attributes to dummy continuous ones; both alternatives (especially the latter) may lead to serious side effects and deviations from the desired performance of the classifier.
2. They view induction as a statistical decision theory problem. This is an advantage, in the sense

that rigorous learning algorithms have appeared with interesting convergence properties. However, the resulting classifiers are represented in the form of mathematical models which are hard for human beings to understand. Thus a lot of information regarding the structure of the domain remains implicit.

The most important statistical methods[8] are discriminant analysis, Bayesian classifiers, nearest neighbour classifiers (Duda & Hart, 1973) and belief networks (Pearl, 1985).

II.5 Symbolic methods

Symbolic methods have been developed by the artificial intelligence community, and they view induction as a search in which the learner is given the input instances (as initial state), the background knowledge (that defines rules of inference which enable moving from one state to the next) and is required to reach an output hypothesis (final state) (Michalski, 1983). Moves from one state to the next are guided by generalization, specialization and reformulation operators. The representation formalisms involved in the inductive process (e.g., logic, graphs, frames, etc.) are common to those employed in other areas of artificial intelligence applications. The objective of symbolic approaches is to induce hypotheses which are represented in a humanly understandable form (this is the reason why mathematical models are rejected) and can be used by an expert system in order to classify new unlabelled instances. These methods often require the generation of a hypothesis that does not misclassify any training instance. Therefore, symbolic methods are appropriate only for deterministic domains. However, various versions of the methods can be adapted to cater for probabilistic and/or noisy domains by the introduction of statistically oriented heuristics to the learning process.

The following subsections discuss the two major existing symbolic methods, namely the version spaces and the AQ algorithm. Other methods are exemplified by the CHARADE system (Ganascia, 1987), the DUCE system (Muggleton, 1987), the STAGGER system (Schlimmer, 1987) and its descendant, the HILARY system (Iba et al., 1988).

II.5.1 Version spaces

The version space of a training data set, with respect to bias B, is defined by Mitchell (1982) as the set of the hypotheses (resulting from bias B) which are consistent with all the training instances. The hypotheses of the version space are partially ordered by the generality relation. Thus, the version space is delimited by two sets: the set Ⓢ of the most specific hypotheses, and the set Ⓖ of the most general hypotheses. Learning on a version space consists of incrementally accepting training instances and updating the sets Ⓢ and Ⓖ until both shrink to the same unique hypothesis, which is the target concept (see also Figure 4). Mitchell called the above learning process *candidate elimination*. Failure to reach the target hypothesis means that this is not included in the hypothesis space resulting from bias B (i.e., the bias is not correct, to borrow the terminology introduced in section I.2).

Mitchell (1982) has demonstrated that on a discrete attribute space with conjunctive target concept and without noise in the training set, the candidate elimination approach is able to identify this target concept. However,the cardinality of the set Ⓖ may increase exponentially to the number of data (Haussler, 1988). Furthermore, in a continuous attribute space, no finite training data set can account for the exact determination of the target concept. Thus Haussler (1988) introduced the idea of approximating the target concept by error at most ε with confidence at least $1 - \delta$ and computing the number of instances required for that approximation (provided that the target concept is conjunctive).

[8]This paper will not enter into the details of statistical methods because they are not widely used in knowledge engineering.

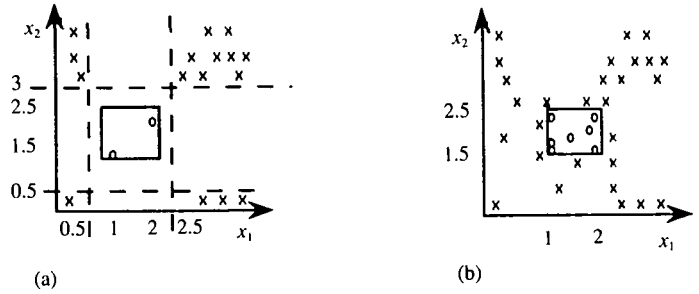

Figure 4 An illustration of the version space method. The attribute space contains two numerical attributes (x_1,x_2) and two decision classes (o and x). The aim of induction is to determine a hypothesis (which is a conjunction of attribute values) that covers all o points and no x point. In Figure 4a the algorithm has accepted some data and has formed the sets Ⓖ and Ⓢ to be the following:

The set of the most general hypotheses Ⓖ contains two hypotheses (marked by the dashed lines in Figure 4a), namely:

$$Ⓖ = \{[0.5 \leq x_1 \leq 2.5], \quad [0.5 \leq x_2 \leq 3]\}$$

The set of the most specific hypotheses Ⓢ contains only one hypothesis, namely:

$$Ⓢ = \{[1 \leq x_1 \leq 2] \textbf{ and } [1.5 \leq x_2 \leq 2.5]\}$$

In Figure 4b the algorithm has accepted more data and the sets Ⓢ and Ⓖ have both shrunk to the unique hypothesis: $[1 \leq x_1 \leq 2]$ **and** $[1.5 \leq x_2 \leq 2.5]$.
Now, Ⓢ = Ⓖ, therefore the hypothesis:

if $[1 \leq x_1 \leq 2]$ **and** $[1.5 \leq x_2 \leq 2.5]$ **then** class = o
else class = x

is the required target hypothesis

A serious problem arises when the target concept contains disjunctions. Murray (1987), in his HYDRA system, suggested that one could minimally specialize hypotheses in Ⓢ by introducing all possible disjunctions, if no conjunctive concept is fully consistent with the training instances. This approach is guaranteed to produce a disjunctive concept consistent with the training data but not in polynomial time. Other approaches which generate version spaces by introducing minimal disjunction criteria, are discussed in Kodratoff (1988). Bundy et al. (1985) considered the version space problem as a focusing problem where the aim is to minimize the uncertainty about the target concept by incrementally adjusting the concept boundaries (by the acceptance of new training instances) until these stabilize to the target concept. Thornton (1987) showed that the advantage of the focusing method is that it is capable of discovering the target concept (in a discrete attribute space) while other methods (e.g., decision trees) are not. However, he pointed out that the efficiency of focusing decreases if the target concept contains disjunctions.

The version spaces approach does not seem useful for practical learning problems (e.g., knowledge base construction) for three main reasons:

1. It is restricted to two class domains, and no attempt has been made so far to extend it to multi-class domains.
2. It works satisfactorily only with conjunctive target hypotheses or with target hypotheses in which the number of disjuncts is very few.
3. It is computationally expensive.

II.5.2 The AQ algorithm

The AQ algorithm (Michalski, 1973) was inspired by non-hierarchical seed-based clustering methods (Anderberg, 1973). The objective of the algorithm is to group the training instances of each decision class into a number of clusters (which are not necessarily disjoint) and assign to each such cluster a description that is a conjunction of attribute values which cover all the training instances of that cluster and no training instance that belongs to another decision class.

The algorithm can be described as follows:

For each *class* of the domain DO[9]

1. Select a seed instance *e* from the class you wish to describe.
2. Generate a set of conjunctive descriptions such that each description covers the seed instance *e* and does not cover any instance of another class.
3. Select the best conjunctive description © according to a *quality criterion*.
4. Remove the training instances that this description covers. Add © as a new disjunct in the class description.
5. **If** there are still uncovered training instances of this class **Then** go to Step 1 in order to generate new conjunctive descriptions to cover these remaining instances.
6. **Else** Form the decision rule:

   If C_1 **Or** C_2 .. **Or** C_r **Then** class
   where C_i $(1 \leq i \leq r)$ are the conjunctive descriptions (clusters) that cover all the training instances of *class*.

A simple example, illustrating how induction proceeds using the AQ algorithm, is given in Figure 5.

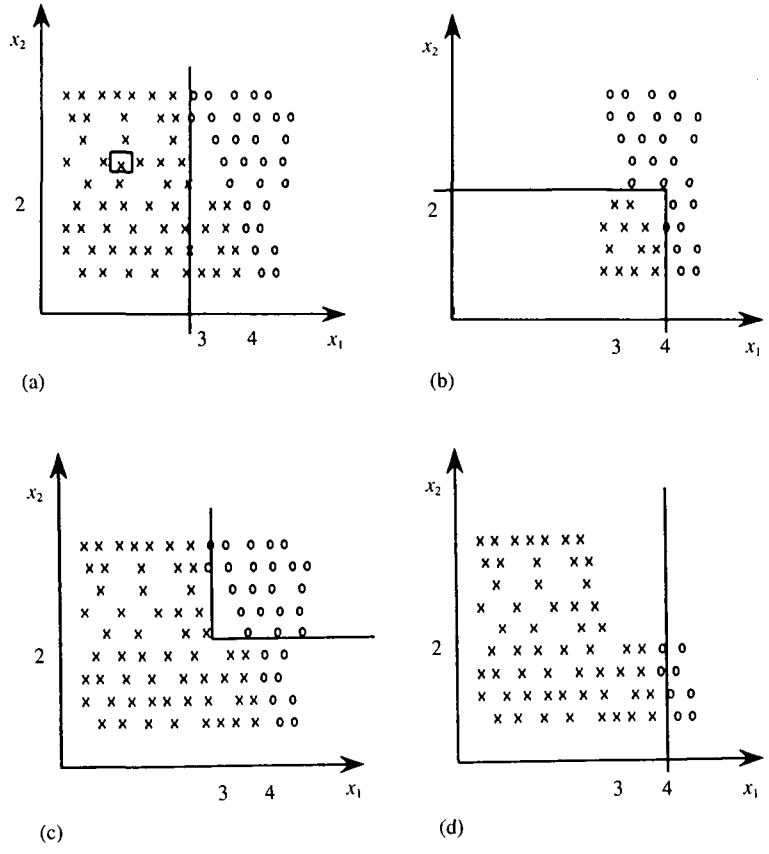

Figure 5 The AQ algorithm. The attribute space contains two numerical attributes (x_1,x_2) and two decision classes (x and o). In the beginning, AQ looks for a conjunctive description of the instances of class x. Picking as seed instance the $\boxed{x}$ point of Figure 5a, it induces the expression: $[x_1 < 3]$. Removing the training instances for which $[x_1 < 3]$ (Figure 5b), there have still remained training instances of class x. To cover these instances, AQ induces the conjunctive expression $[x_1 < 4]$ **and** $[x_2 < 2]$ (Figure 5b). Now all instances of class x have been covered; therefore, the rule for class x is:

if $[x_1 < 3]$ **or** $([x_1 < 4]$ **and** $[x_2 < 2])$ **then** class = x

Following the same process, AQ generates the following rule for the class o (Figures 5c and 5d):

if $([x_1 \geq 3]$ **and** $[x_2 \geq 2])$ **or** $[x_1 \geq 4]$ **then** class = o

[9]If the domain contains only two decision classes, then it is sufficient to determine a rule only for the one class and any instance that does not match that rule will be automatically classified to the other class.

Conjunctive descriptions generated during Step 2 are assessed by a set of criterion tolerance pairs ($\langle A_1,T_1\rangle$, ... $\langle A_m,T_m\rangle$) where A_i defines the criterion (e.g., number of terms in the conjunction) and T_i defines a threshold of acceptance of a description with respect to that criterion. The (conjunctive) descriptions are first assessed by criterion A_1; those that score above threshold T_1 are assessed by criterion A_2 and the same process continues over the next criteria until either the list of criteria has been exhausted or a single description has remained as a candidate (which is finally chosen).

The above algorithm can also account for incremental learning and various versions have been produced, such as AQ11 (Michalski & Chilauski, 1980), AQ15 (Michalski et al., 1986 b) and GEM (Reinke & Michalski, 1988). The general idea behind the AQ algorithm (i.e., seed-based clustering) was also employed in the unsupervised learning (clustering) system CLUSTER/2 (Michalski & Stepp, 1983) which creates clusters from a training set of unlabelled instances and intensionally describes each cluster as a conjunction of attribute values.

AQ is useful for constructing the knowledge base of an expert system, because it works with any number of classes. Furthermore, the output hypothesis is represented as a set of conjunctive **If..then** rules, which is the most common representation for expert systems. AQ has been applied in the construction of an expert system for soyabean diagnosis. That system showed superior performance to an expert system constructed by traditional knowledge acquisition techniques for the same domain (Michalski & Chilauski, 1980). However, AQ is not appropriate for noisy and probabilistic domains because it requires 100% accuracy of the induced rules on the training set. A version of AQ, called AQ15 (Michalski et al., 1986), was built in order to induce probabilistic rules. Nevertheless, AQ15 showed inferior performance compared to decision tree systems (Cestnik et al., 1987). Another problem associated with AQ is that of rule interpretation. The rule induction algorithm does not impose any constraints on the rule interpreters applicable to the resulting set of rules. This can be regarded as an advantage, in the sense that any available rule interpreter (i.e., expert system shell) can be used for the classification of new instances. However, the induced set of rules does not possess any structure (unlike a decision tree, for instance); thus, searching that rule set for the classification of new instances might be inefficient.

II.6 Decision tree methods

Decision tree methods are a hybrid of statistical and symbolic methods. The statistical component of decision tree methods concerns the tree induction process which is guided by statistically-based criteria. The symbolic component concerns the decision tree structure and the conversion of a decision tree into a set of production rules. For their predominant position in the inductive learning literature, decision trees are dealt with in more detail than the other methods.

Decision trees are non-parametric classifiers that recursively partition regions of the attribute space into subregions. A decision tree is a collection of nodes. Each node of the tree corresponds to a region of the attribute space, and the root of the tree corresponds to the entire attribute space. Each terminal node is assigned to a class. This class is assigned by the classifier to any instance that belongs to the corresponding region. Each non-terminal node of the tree is associated with a test with I possible outcomes. The test partitions the corresponding region of the attribute space into subregions, where each subregion corresponds to a child of the original region.

A new instance is classified by the decision tree, by traversing a path from the root of the tree to a terminal node. At each non-terminal node along the path, the instance is subjected to the test associated with that node. The outcome of the test, which depends on the instance's attribute values, determines the respective child node where the instance should pass next. When that instance reaches a terminal node, it is classified to the class assigned to that node.

A generic decision tree development algorithm proceeds as follows:

Tree development algorithm

0. Initialize: A : = The root of the tree

Dataset : = The entire training set

1. Expand-tree (A, Dataset)

Proceed expand-tree (A, Dataset)

If the *stopping criterion* is satisfied for *Dataset* **Then**

— Declare A as a terminal node

— Assign A to the majority class

Else

— Select a split from the set of *candidate splits* subject to a *splitting criterion*.

— Partition *Dataset* into m subsets $Dataset_i$ ($1 \leq i \leq m$) so that :

$\forall$ @$\in$ Dataset, @ $\in$ $Dataset_i$ iff the attr. values of d classify it to child node (subregion) B_i.

— **For** each child node B_i resulting after splitting node A *Do* Expand-tree (B_i, $Dataset_i$)

The decision tree development process (which is illustrated by an example in Figure 6) is mainly characterized by four issues (Breiman et al., 1984):

1. A set of candidate tests (splits) whose outcomes determine the way a particular region (node) is partitioned into subregions (child nodes). Most of the existing systems employ splits based on the values of one attribute (as shown in Figure 6). For numerical (i.e., real or integer valued) attributes, a region is usually split into two subregions[10]. For discrete attributes, some algorithms create binary splits, while others create a separate child node for each distinct attribute value. Some recent approaches have considered splits based on the conjunction of several attribute values (Pagallo, 1989; Matheus & Rendell, 1989). Another alternative is to consider splits based on linear combinations of numerical attributes (Breiman et al., 1984); however, these types of splits result in decision trees that are not easily understandable to human experts.
2. A criterion that determines which split to choose from the set of candidate splits. A region of the attribute space is split into subregions, in order to reduce the uncertainty concerning the class membership of the instances that belong to that region. That uncertainty is quantified by the introduction of an impurity function $f(A)$ associated with each region A. $f(A)$ is a function of the class probabilities at region A. These class probabilities are usually estimated by the class

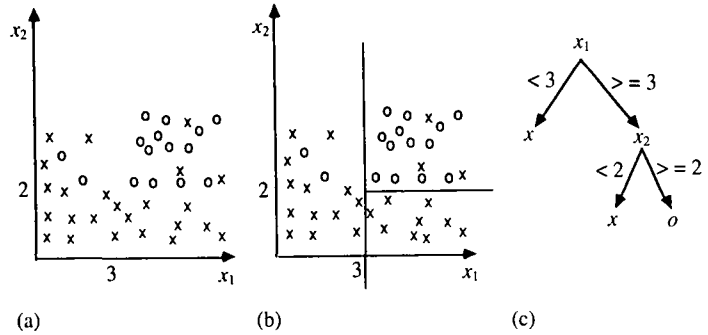

Figure 6 The decision tree development process. The attribute space contains two numerical attributes (x_1,x_2) and two decision classes (o and x) (Figure 6a). The attribute space is first partitioned by a line parallel to the x_2 axis. Region ($x_1 < 3$) is not further partitioned, and it is assigned to class x. Region ($x_1 \geq 3$) is further partitioned into two subregions by the line ($x_2 = 2$). The first subregion ($x_2 < 2$) is assigned to class x (since all the training data corresponding to it belong to class x). The second subregion contains few data belonging to class x; we decide to stop further refinement and assign this subregion to class o (Figure 6b). Figure 6c shows the decision tree representation of the feature space partitioning of Figure 6b

[10]However, there are decision tree algorithms in which a split based on the values of a numerical attribute may partition a region into more than two subregions (Kass, 1980).

relative frequencies observed in the training set. The value of $f(A)$ is low when the probability of one class is very high, while the probabilities of the other classes are very low. After splitting region A into subregions $B_1, \ldots, B_l$, the impurity becomes $\sum_{i=1}^{l} P(B_i) \cdot f(B_1)$ (where $P(B_i)$ is the probability that a randomly chosen instance from region A belongs to subregion B_i). Therefore, the reduction in the impurity after splitting is $\Delta f = f(A) - \sum_{i=1}^{l} P(B_i) \cdot f(B_1)$. A decision tree algorithm chooses the split that maximizes Δf from amongst all the candidate splits.

The most frequently used impurity functions are (Breiman et al., 1984):

(i) The entropy of the class probabilities p_j (where $1 \leq j \leq k$ for a k class domain) at region A:

$$f(A) = -\sum_{j=1}^{k} p_i . \log_2 p_i$$

(ii) The quadratic entropy (also called Gini diversity index) of the class probabilities at region A:

$$f(A) = 1 - \sum_{j=1}^{k} p_i^2$$

(iii) The error of region A, if A was declared as terminal and assigned to the highest probability class:

$$f(A) = 1 - \max_{1 \leq j \leq k} p_j$$

An alternative to the impurity function-based splitting criteria is the use of statistical tests, such as the χ^2 contingency test, that measure whether a candidate split significantly reduces the impurity of a region (Kass, 1980). It has been shown that the entropy impurity function and the χ^2 test criteria are asymptotically equivalent (Mingers, 1989 a). Furthermore, several variants of the splitting criteria presented above have been considered to induce better decision trees for particular problems (Mingers, 1989 a).

3. A criterion that determines whether to stop further partitioning of a region (and declare it as terminal), and a criterion that decides whether to prune an already formed tree in order to increase its reliability over the classification of future unlabelled instances. A standard criterion for stopping further expansion of a node is that all the training instances corresponding to that node should belong to the same decision class. However, such a criterion may result in large trees whose terminal nodes correspond to very few training data. Such trees show high misclassification error when classifying non-training instances, while the observed misclassification error over the training set is very low. In order to increase the accuracy of the trees over non-training data, it is necessary either to establish stricter stopping criteria or to prune an already developed tree, by rendering some non-terminal nodes (regions) into terminal ones, subject to a pruning criterion (Mingers, 1989 b).
4. A rule that assigns a class to the instances that belong to a terminal node (region). The most common rule is to assign the class of the majority of the training instances that belong to that node.

Induction of decision trees proceeds top-down, by splitting a region into small subregions according to a split selection criterion (therefore, decision tree induction is model driven). Furthermore, most of the existing inductive algorithms perform non-backtracking depth-first search over the space of possible splits of a certain region. The induction of a decision tree that optimizes a specific criterion (e.g., minimization of terminal nodes) can only proceed bottom up,

but the search time is exponential in the number of the domain attributes (Payne & Neisel, 1977). Therefore, bottom up methods are not practically useful.

Decision tree systems are not incremental, since the whole training set is required to be available from the beginning of the induction process. Some incremental versions of decision tree systems that appeared in the literature, such as the algorithms ID4 (Schlimmer & Fisher, 1986) and ID5 (Utgoff, 1988), update the tree only when the introduction of a new training instance significantly changes the statistics associated with each node. However, both ID4 and ID5 need to keep all the training instances in memory in order to update the tree when necessary. Another important alternative is to use a subset (a *window*) of the training set for learning a decision tree, and then to examine the tree classification performance over the remaining training data; if the tree misclassifies a significant amount of them, then the misclassified data are added to the existing window to form a new window which will be used for the generation of a new tree. This process is repeated until a tree with good classification behaviour on the entire training set is generated. Windowing has been used by the ID3 (Quinlan, 1986) and C4 (Quinlan et al., 1986) systems; nevertheless, experiments show that windowing does not offer significant advantages compared to single step learning with the entire training set (Wirth & Cattlet, 1988).

An important problem pertaining to decision tree systems is how they handle missing attribute values. Missing attribute values occur in three levels (Quinlan, 1989 a), and various methods have been suggested to cope with them:

1. When computing the score of split selection criteria.
2. When assigning instances with unknown values on the splitting attribute(s) to the appropriate branch(es).
3. When classifying instances that miss values of attributes encountered along the path from the root of the tree to a terminal node.

The decision tree structure offers an excellent top-down inference method for classifying new instances. However, most existing expert system shells are rule-based. In this case, the decision tree can be converted into a set of conjunctive **if** Condition **then** class production rules, each rule corresponding to a distinct path of the tree. Furthermore, it is possible to prune the initial set of production rules by removing some conditions from the **if** part of these rules (Quinlan, 1987 a). This type of pruning was found to be very effective, especially in cases where traditional tree pruning techniques are unable to yield satisfactory results (Quinlan, 1987 b). However, a rule set obtained from the paths of the decision tree destroys the tree structure and renders inference ineffective. A method that isomorphically maps a decision tree into a set of production rules, is described in Kalkanis (1989). This method results in larger rule bases than those obtained by forming a separate rule from each path of the tree; on the other hand, it makes inference as effective as with the original decision tree.

Decision tree induction is based on divide and conquer methods, since each region of the attribute space is divided (split) into subregions, and each of these subregions may be further divided. An alternative to a divide and conquer method is a separate and conquer method. Separate and conquer methods induce decision lists which are specific types of decision trees. A *decision list* is represented as a list of pairs $((T_1,c_{k1}), \ldots, (T_r, c_{kr})$, *default class*), where $T_i = T_i(x_1, \ldots, x_d)$ is a Boolean function of the domain attributes, and c_{ki} is a class assigned to the instances that belong to the subregion in which $T_i = 1$ (see also Figure 7). If T_i is a function of a single attribute, then the list is called a *linear decision tree* (Arbab & Michie, 1985).

Learning on decision lists mainly involves the determination of the appropriate term T_i at level $i - 1$ of the list. In the approaches existing so far, this term is a conjunction of attribute values. There are two main alternatives for choosing T_i:

1. The class for which T_i is true is prespecified—this is the case in systems like PRISM (Cendrowska, 1987) and GREEDY3 (Pagallo & Haussler, 1988).

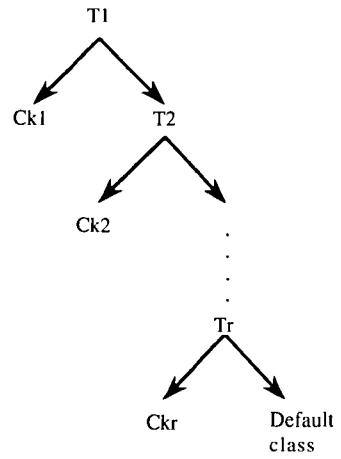

Figure 7 Separate and conquer methods determine a term that accounts for a satisfactory classification of a part of the domain. The remaining part of the domain has to be recursively subjected to the same process

2. The class for which T_i is true is not prespecified, but is dynamically determined as the class of the best candidate term—this approach has been followed by systems like CN2 (Clark & Niblett, 1987) and GROVE (Pagallo & Haussler, 1988).

The search methods for the derivation of T_i are either depth first (e.g., PRISM, GREEDY3, GROVE) or beam search (e.g., CN2). the evaluation functions used by the current systems, to control search, are either information theoretic criteria (CN2, PRISM, GROVE) or criteria maximizing the target class probability of the instances that satisfy the term T_i (GREEDY3). Furthermore, CN2 applies a stopping criterion when the addition of further conjunctive terms to T_i does not significantly improve classification quality, while GROVE and GREEDY3 perform pruning by dropping some conjunctive terms after the entire T_i has been created.

Decision lists are easily converted to production rules of the form **if** T_i **then** c_{ki} which have to be examined in the sequence they appear in the list, when a new instance comes to be classified. Therefore, decision lists tend to be more understandable than decision trees.

In the long run, decision tree methods possess the following advantages over purely symbolic methods:

1. They are applicable to both deterministic and probabilistic domains. Furthermore, they show a satisfactory performance in the presence of noisy data.
2. The terminal nodes of the tree (and, subsequently, the resulting production rules) may be accompanied by a certainty factor that indicates the probability of the classes assigned to these nodes. Alternatively, each terminal node may be accompanied by a confidence interval estimate of the error rate of the corresponding production rule. Confidence interval estimates can be obtained from the training set (Kalkanis, 1991). The advantage of confidence interval estimates over point estimates is that they incorporate the confidence component which expresses the probability that the error rate of the resulting rule lies within the predicted interval.
3. The tree induction process is computationally efficient, and is guided by statistically sound criteria.
4. The structure of the decision tree offers a simple rule interpreter which is very efficient for the classification of new instances.

The major drawback of decision tree methods is a result of the single attribute splits that most of the currently existing approaches use. Using only single attribute splits makes tree induction quick because the set of candidate splits of a node is highly constrained. However, that constraint, combined with the non-backtracking depth first search employed by most tree induction algorithms, renders decision trees inappropriate for some problem domains. Decision lists relax the

single attribute constraint by allowing binary splits based on a conjunction of attribute values, nevertheless the requirement that one of the resulting two child nodes should be terminal might not always be the best policy. To combine the advantages of decision tree and decision list approaches, Chan (1989) suggested an approach in which each node is split by a conjunction of terms derived by using decision list induction techniques. Both of the resulting child nodes are then considered for further splitting.

II.7 The valiant learning framework

The Valiant learning (or *Probably Approximately Correct* (PAC) *Learning*) framework, was introduced by Valiant (1984) as an attempt to analyse the complexity of the induction task and provide convergence rates for various classes of hypotheses[11]. Learnability, in the sense of Valiant, can be defined as follows (Haussler, 1987):

Definition A class of hypotheses (concepts) $\mathcal{H}$ is (polynomially) learnable if there exists an algorithm L that runs in polynomial time w.r.t. all the associated input parameters, and also that:

Given:

1. A set of training data coming from any probability distribution and any target concept $H \in \mathcal{H}$.
2. A maximum error tolerance ε and a confidence parameter δ.

The algorithm induces a hypothesis H_L in which[12]: $P\{\text{Error}\ (H_L) \leq \varepsilon\} \geq 1 - \delta$. Furthermore, the number of data required for inducing hypothesis H_L is polynomial w.r.t ε, δ and any other parameters pertaining to the problem domain (e.g., number of attributes, complexity of the target concept, etc.).

The Valiant learning framework was first applied to Boolean attribute-based domains, and gave some positive learnability results for various classes, such as conjunctive concepts, k-CNF, k-DNF concepts, while it was found that the general Disjunctive Normal Form (DNF) class was not learnable (Kearns et al., 1987). The above results were extended to be valid for any attribute-based two class domain, by using results based on convergence rates of a generalization of the Glivenko–Cantelli theorem given in Vapnik and Chervonenkis (1971). Such extensions have been reported in Blumer et al. (1986) and Haussler (1988). Furthermore, various variants of the PAC learning framework have appeared, intending to examine learnability under different learning protocols, such as learning by experimentation (Amsterdam 1988 a), learning in the presence of background knowledge (Milosavljevic, 1987), learning in the presence of attribute noise (Angluin & Laird, 1988), and incremental learning with memory bounds (Haussler, 1988 b).

However, the Valiant learning framework is very restricted. The main limitations of it are outlined below:

1. It requires that the learner knows the class where the target concept belongs (i.e., the concept bias) before induction begins. Christman (1989) extended the PAC learning framework so as to perform a test and report whether the output hypothesis H_L (produced by algorithm L) does meet the learnability requirements. That test guarantees (with confidence $1 - \delta$) that any class of hypotheses learnable by algorithm L will be reported as learnable by the test; while hypotheses belonging to classes that do not guarantee learnability may also be PAC learned by L.

[11] A class of hypotheses is a set of hypotheses that share a common representation formalism. For instance, all the hypotheses that are written as disjunctions of conjunctions of terms (where each conjunction contains at most k terms), form the class of k-DNF (k-Disjunctive Normal Form).

[12] Error (H_T) is a random variable of the training sample, and is defined with respect to the target concept.

2. The training set size required to PAC learn a concept H is valid for all concepts of a concept class and for any probability distribution over the attribute space. However, in a practical induction problem, the input data are drawn from a particular (although unknown) distribution, and the target concept is not necessarily the most difficult to learn among all the concepts of the concept class. Therefore, the actual problem required size of the data set is often much less than that predicted by the PAC learning model. Buntine (1989) considered PAC learnability in a Bayesian framework and produced more realistic figures.
3. The Valiant learning framework currently tackles two class deterministic domains, although most real world domains are multi-class probabilistic ones (Amsterdam, 1988 b). Moreover, it requires that the concept H_T produced by algorithm L, is 100% consistent with the training data. An attempt towards relaxing these constraints would be possibly to determine rates of convergence for algorithms that minimize the resubstitution error rather than requiring the latter to be zero (Vapnik, 1989).

II.8 Applicability of inductive methods to knowledge engineering

Inductive methods are useful for knowledge base construction and knowledge base refinement. Knowledge base refinement is used to:

1. Integrate new knowledge into an existing knowledge base so that the new knowledge is consistent with the existing knowledge.
2. Compress a knowledge base in order to resolve conflicts, eliminate redundant rules and impose a better structure to it. Methods for automatic knowledge base refinement are reported in Lee and Ray (1986) and Wilkins and Buchanan (1986).

Knowledge base construction is the main application area of inductive methods. Attribute-based induction is the most common inductive learning paradigm used for knowledge base construction. Attribute-based induction simplifies the knowledge acquisition task, because the domain expert(s) is not required to explicitly articulate a complete and consistent set of rules that express his/her domain expertise. Instead, the expert should determine a set of necessary and sufficient attributes that characterize the problem domain. Furthermore, the expert should format an available set of past cases (instances) of the problem domain, so that each case is described as a vector of attribute values.

The determination of a set of necessary and sufficient attributes is a non-trivial process. The expert should take care not to include irrelevant or highly correlated attributes. Moreover, the attributes must be formatted in a way directly relevant for the expression of any causal relationship between attribute values and classes. For example, if the difference $x - y$ of two numerical attributes affects the class membership of an instance (while none of x and y affect the class membership individually), the expert should include the attribute $x - y$ in the attribute set, rather than just the primitive attributes x and y. Including only primitive attributes renders classification rules inefficient. Even the most sophisticated constructive induction approach is unable to compensate for the human expertise in the selection of the appropriate set of attributes. Constructive induction approaches are effective once an appropriate set of initial attributes is available. Furthermore, many of the existing constructive induction techniques require a considerable amount of background knowledge in order to suggest useful new attributes.

The formatting of a large set of past cases into vectors of attribute values is a tedious process that requires considerable attention and expertise. Different cases may have been recorded by different experts in the files, and it is very hard to describe all these cases by the predefined set of attributes. Therefore, many revisions in the predefined attribute set are necessary to reach a set of necessary and sufficient attributes and format all the cases as vectors of these attributes' values. Furthermore, there may be missing or erroneous attribute values which degrade the performance of the resulting set of rules. Another problem is that the set of the available past instances may not be fully

representative of the problem domain. In this case, the expert has either to provide a set of additional simulated instances, or has to modify the rules generated by the inductive algorithm in order to compensate for unrepresentative training data.

The rules generated by an inductive algorithm are rarely used immediately as the knowledge base of an expert system. Human experts often refine the induced rules by removing irrelevant conditions and adding other conditions missing from the original rule set. Buntine and Stirling (1989) describe an approach called *interactive induction*, where the following iterative process takes place: a decision tree algorithm generates rules from data. The expert modifies these rules by considering new attributes and/or removing some other attributes from consideration. The training data are reformatted by updating the set of attributes, and the decision tree induction algorithm runs with the updated attribute set. Iterations stop when the expert is happy with the decision tree output. Interactive induction was successfully applied to generating rules for routing steel produced in a job shop.

Above all, attribute-based induction is not appropriate for complex domains where higher order languages (such as predicate logic, graphs or frames) or deep knowledge models are necessary. Nevertheless, learning in complex domains usually requires a large amount of background knowledge to be supplied by the expert in order for the system to generate accurate hypotheses in an efficient manner. However, some attribute-based induction approaches can be extended so as to be applicable to complex domains. Examples of such attempts are the INDUCE (Michalski, 1983) and the FOIL (Quinlan, 1989 b) systems that are first order logic extensions of the attribute-based systems AQ11 and ID3, respectively.

Conclusions

The first part of this paper tried to make clear that induction is a subjective process, since there may be more than one alternative hypothesis that fits a set of input data. The choice of the final hypothesis is a consequence of the inductive bias which incorporates the learner's preference for one alternative over another. The subjective statement of the inductive problem was then quantified and the role of the representation in the induction task was highlighted.

Part II focused on methods for attribute-based induction. In attribute-based induction, most of the background knowledge is incorporated in the semantics of the input attributes, further to that point induction becomes, more or less, a mechanistic process. The basic criteria to assess the quality of the induced rules are their accuracy over the classification of new unlabelled instances and their soundness and understandability to the human experts. Early statistical approaches, developed within the field of pattern recognition, disregarded the issue of soundness and understandability, while many symbolic approaches (developed by artificial intelligence researchers) failed to produce successful classification rules in noisy and uncertain domains. It seems that approaches such as decision trees and decision lists (both of which are easy to convert to production rules) offer the advantage of statistical preciseness together with that of human understandability.

In the long run, it is questionable whether inductive methods can cover the whole knowledge acquisition life cycle. In the authors' opinion, rule induction should be integrated as part of the knowledge acquisition process (which encompasses other stages, such as background knowledge elicitation, validation and refinement of the induced rules by the user, etc.). Efforts for creating such integrated knowledge acquisition methodologies, are still at the research stage (Rappaport & Gaines, 1990).

References

Amsterdam, J, 1988a. "Extending the Valiant learning model" In: *Proceedings of 5th International Workshop on Machine Learning*. Morgan-Kaufmann.

Amsterdam, J, 1988b. "Some philosophical problems with formal learning theory" In: *Proceedings AAAI'88*. Morgan-Kaufmann.

Anderberg, MR, 1973. *Cluster Analysis for Applications*. Academic Press, New York, NY.
Angluin, D and Laird, PD, 1988. "Learning from noisy examples" In: *Machine Learning* 2(4), 343–370.
Angluin, D and Smith, CH, 1983. "Inductive inference: theory and methods" *ACM Computing Surveys* **15**(3), 237–269.
Arbab, B and Michie, D, 1985. "Generating rules from examples" In: *Proceedings 9th IJCAI'85*. Morgan-Kaufmann.
Baim, PW, 1988. "A method for attribute selection in inductive learning systems" *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**(6), 888–896.
Bierman, AW and Feldman, JA, 1972. "A survey of results in grammatical inference" In: S Watanabe (ed.), *Frontiers of Pattern Recognition*. Academic Press, New York, NY.
Blumer, A, Ehrenfeucht, A, Haussler, D and Warmuth, M, 1986. "Classifying learnable geometric concepts with the Vapnik–Chervonenkis dimension" In: *Proceedings 18th ACM Symposium*. ACM.
Blumer, A, Ehrenfeucht, A, Haussler, D and Warmuth, M, 1987. "Occam's razor" *Information Processing Letters* **24**(6), 377–380.
Blythe, J, 1988. "Constraining search in a hierarchical discriminative learning system" In: *Proceedings ECAI'88*. Pitman.
Boucheron, S and Sallantin, J, 1988. "Learnability in the presence of noise" In: *Proceedings EWSL'88*. Pitman.
Breiman, L, Friedman, JH, Olshen, RA and Stone, CJ, 1984. "Classification and regression trees" Wadsworth Int. Group, Belmont, CA.
Buchanan, BG and Mitchell, TM, 1978. "Model directed learning of production rules" In: DA Waterman and F Hayes-Roth (eds.), *Pattern Directed Inference Systems*. Academic Press, New York, NY.
Bundy, A, Silver, D and Plummer, D, 1985. "An analytical comparison of some rule learning programs" *Artificial Intelligence* **27**(2), 137–181.
Buntine, R, 1989. "A Critique of the Valiant model" In *Proceedings 11th IJCAI-89*. Morgan-Kaufmann.
Buntine, W and Stirling, KA, 1989. "Interactive induction" In: J Hayes and D Michie (eds.), *Machine Intelligence* 12. Oxford University Press, Oxford.
Carnap, R, 1950. *The Logical Foundations of Probability*. Chicago.
Carter, C and Catlett, J, 1987. "Assessing credit card applications using machine learning" *IEEE Expert* **2**(3), 71–79.
Cestnik, B, Kononenko, I and Bratko, I, 1987. "ASSISTANT'86: a knowledge elicitation tool for sophisticated users" In: I Bratko and N Lavrac (eds.), *Progress in Machine Learning*. Sigma Press, Wilmslow.
Chan, PK, 1989. "Inductive learning with BCT" In: *Proceedings 6th International Workshop on Machine Learning*. Morgan-Kaufmann.
Chendrowska, J, 1987. "PRISM: an algorithm for inducing modular rules" *International Journal of Man–Machine Studies* **27**(4), 349–370.
Chow, CK, 1957. "An optimum character recognition system using decision functions" *IRE Transactions on Electronic Computers* **EC-6** (December 1957), 247–254.
Chrisman, L, 1989. "Extending the Valiant framework to detect incorrect bias" Technical Report, School of Computer Science, Carnegie Mellon University, CMU-CS-89-137.
Clark, P and Niblett, T, 1987. "Induction in noisy domains" In: *Progress in Machine Learning*. Sigma Press, Wilmslow.
Devijver, PA and Kittler, J, 1982. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, London.
Dietterich, TG and Michalski, RS, 1985. "Discovering patterns in sequences of events" *Artificial Intelligence* **25**(2), 187–232.
Drastal, G, Meunier, R and Raatz, S, 1989. "Error correction in constructive induction" In: *Proceedings 6th International Workshop on Machine Learning*. Morgan-Kaufmann.
Duda, RO and Hart, PE, 1973. *Pattern Classification and Scene Analysis*. Wiley, Chichester.
Efron, B, 1983. "Estimating the error rate of a prediction rule" *Journal of the American Statistical Association* **78**(382), 316–331.
Feigenbaum, EA, 1981. "Expert systems in the 1980s" In: A Bond (ed.), *State of the Art Report on Machine Intelligence*. Pergamon-Infotech, Maidenhead.
Fisher, DH, 1987. "Conceptual clustering, learning from examples and inference" In: *Proceedings 4th International Conference on Machine Learning*. Morgan-Kaufmann.
Fu, KS, 1974. *Syntactic methods in Pattern Recognition*. Academic Press, New York. NY.
Gams, M and Lavrac, N, 1983. "Review of five empirical learning systems within a proposed schemata" In: I Bratko and N Lavrac (eds.), *Progress in Machine Learning*. Sigma Press, Wilmslow.
Ganascia, JG, 1987. "Learning with Hilbert cubes" In: I Bratko and N Lavrac (eds.), *Progress in Machine Learning*. Sigma Press, Wilmslow.
Glick, N, 1978. "Additive estimators for probabilities of correct classification" *Pattern Recognition* **10**(3), 211–222.

Gross, KP, 1988. "Incremental multiple concept learning using experiments" In: *Proceedings 5th International Workshop on Machine Learning*. Morgan-Kaufmann.

Haussler, D, 1987. "Bias, version spaces and the Valiant learning Framework" In: *Proceedings 4th International Workshop on Machine Learning*. Morgan-Kaufmann.

Haussler, D, 1988a. "Quantifying inductive bias: AI learning algorithms and the Valiant learning framework" *Artificial Intelligence* **36**(2), 177–221.

Haussler, D, 1988b. "Space efficient learning algorithms" University of California, Santa Cruz, UCSC-CRL-88-2.

Hayes-Roth, F, 1974. "Schematic classification problems and their solution" *Pattern Recognition* **6**(2), 105–113.

Highleyman, WH, 1962. "The design and analysis of pattern recognition experiments" *Bell Systems Technical Journal*. March.

Iba, W, Wogulis, J and Langley, P, 1988. "Trading off simplicity and coverage in incremental concept learning systems" In: *Proceedings 5th International Workshop on Machine Learning*. Morgan-Kaufmann.

Jain, AK and Dubes, R, 1978. "Feature definition in pattern recognition with small sample size" *Pattern Recognition* **10**(2), 85–97.

Kalkanis, G, 1989. "A proposal to enhance decision tree based inductive systems" MSc Dissertation, University of Manchester Institute of Science and Technology.

Kalkanis, G, 1991. "The application of confidence interval error analysis to the design of decision tree classifiers" (To appear in *Pattern Recognition Letters*.)

Kanal, LN and Chandrasekaran, B, 1971. "On dimensionality and sample size in statistic pattern recognition" *Pattern Recognition* **3**(3), 225–234.

Kass, GV, 1980. "An exploratory technique for investigating large quantities of categorical data" *Applied Statistics* **29**(2) 119–127.

Kearns, M, Li, M, Pitt, L and Valiant, L, 1987. "Recent results on Boolean concept learning" In: *Proceedings 4th International Workshop on Machine Learning*. Morgan-Kaufmann.

Keynes, JM, 1921. *A Treatise on Probability*. Macmillan.

Kocabas, S, 1991. "Knowledge representation and learning" *Knowledge Engineering Review* **6**(3).

Kodratoff, Y, 1988. *Introduction to Machine Learning*. Pitman.

Langley, P, Simon, HA and Bradshaw, GL, 1984. "Heuristics for empirical discovery" Technical Report, Carnegie Mellon University Robotics Institute.

Lee, WD and Ray, SR, 1986. "Rule refinement using the probabilistic rule generator" In: *Proceedings AAAI'86*. Morgan-Kaufmann.

Leith, P, 1984. "Hierarchically structured production rules" *The Computer Journal* **26**(1), 1–5.

Lenat, DB, 1983. "The role of heuristics in learning by discovery: three case studies" In: RS Michalski, JG Carbonell and TM Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach (vol. I)* Tioga.

Lewis, PW, 1962. "The characteristic selection problem in recognition systems" *IRE Transactions on Information Theory*, IT-8 (February 1962), 171–178.

Matheus, CJ and Rendell, LA, 1989. "Constructive induction on decision trees" In: *Proceedings 11th IJCAI'89*. Morgan-Kaufmann.

Medin, DI and Wattenmaker, WD, 1986. "Categories, cohesiveness, theories and cognitive archaeology" In: *Concepts Reconsidered: The Ecological and Intellectual Bases of Categories*. U Neisser (ed.). Cambridge University Press.

Mervis, CB and Rosch, E, 1981. "Categorisation and natural objects" *Annual Review of Psychology,* **32**(2), 89–115.

Michalski, RS, 1973. "AQVAL/1—computer implementation of a variable-valued logic system and the application to pattern recognition" In: *Proceedings 1st International Joint Conference on Pattern Recognition*. Washington.

Michalski, RS, 1983. "A theory and methodology of inductive learning" In: RS Michalski, JG Carbonell and TM Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach (Vol I)* Tioga.

Michalski, RS and Chilauski, RL, 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* **4**(2), 125–161.

Michalski, RS and Stepp, RE, 1983. "Learning from observation: conceptual clustering" In: RS Michalski, JG Carbonell and TM Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach (vol. I)*. Tioga.

Michalski, RS, Carbonell JG and Mitchell TM, 1983. *Machine Learning: An Artificial Intelligence Approach (vol. I)* Tioga.

Michalski, RS, Carbonell, JG and Mitchell, TM, 1986a. *Machine Learning: An Artificial Intelligence Approach (vol. II)* Morgan-Kaufman.

Michalski, RS, Mozetic, I, Hong, N and Lavrac, N, 1986b. "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains" In: *Proceedings AAAI'86*. Morgan-Kaufmann.

Michie, D, 1988. "Machine learning in the next five years" In: *Proceedings EWSL'88*. Pitman.

Milosavljevic, A, 1988. "Learning in the presence of background knowledge" Technical Report, Univ. Santa Cruz, California, UCSR-CRL-87-27.

Mingers, J, 1989 a. "An empirical comparison of pruning methods for decision tree induction" *Machine Learning* **3**(3), 319–342.

Mingers, J, 1989 b. "An empirical comparison of pruning methods for decision tree induction" *Machine Learning 4*(2), 227–248.

Mitchell, TM, 1980. "The need for biases in learning generalisations" Technical Report CBM-TR-117, Department of Computer Science, Rutgers University.

Mitchell, TM, 1982. "Generalisation as search" In: BL Webber and NJ Nilsson (eds.), *Readings in Artificial Intelligence*. Troga Publishing Company.

Mortimer, H, 1988. *The Logic of Induction* Ellis Horwood.

Muggleton, S, 1987. "Structuring knowledge by asking questions" In: I Bratko and N Lavrac (eds.), *Progress in Machine Learning*. Sigma Press, Wilmslow.

Muggleton, S, 1988. "A strategy for constructing new predicates in first order logic" In: *Proceedings EWSL'88*. Pitman.

Muggleton, S, 1991. "Inductive logic programming" *New Generation Computing* **8**(4), 295–318.

Murray, KS, 1987. "Multiple convergence: an approach to disjunctive concept acquisition" In: *Proceedings 10th IJCAI'87*. Morgan-Kaufmann.

Neyman, J, 1957. "Inductive behaviour as a basic concept of philosophy and science" *Review of the International Statistical Institute* **25**.

Nilsson, N, 1965. *Learning Machines: Foundations of Trainable Pattern Classifying Systems*. McGraw-Hill.

Pagallo, G, 1989. "Learning DNF by decision trees" In: *Proceedings 11th IJCAI-89*. Morgan-Kaufmann.

Pagallo, G and Haussler, D, 1988. "Feature discovery in empirical learning" University of California at Santa Cruz, Technical Report no. UCSC-CRL-88-08.

Payne, HJ and Meisel, WS, 1977. "An algorithm for constructing optimal binary decision trees". *IEEE Transactions on Computers* **C–26** (a) September, 905–916.

Pearl, J, 1985. "Learning hidden causes from empirical data" In: *Proceedings 9th IJCAI'85*. Morgan-Kaufmann.

Plotkin, GD, 1971. "A further note on inductive generalisation" In: *Machine Intelligence 6*. B Meltzer and D Michie (eds.). Elsevier, New York.

Quinlan, JR, 1986. "Induction of decision trees" In: *Machine Learning 1*(1), 81–106.

Quinlan, JR, 1987a. "Generating production rules from decision trees" In: *Proceedings 10th IJCAI'87*. Morgan-Kaufmann.

Quinlan, JR, 1987b. "Simplifying decision trees" *International Journal of Man–Machine Studies* **27**(3), 221–234.

Quinlan, JR, 1989a. "Unknown attribute values in induction" In: *Proceedings 6th International Workshop on Machine Learning*. Morgan-Kaufmann.

Quinlan, JR, 1989b. "Learning relations: comparison of a symbolic and a connectionist approach" Technical Report 346, University of Sydney.

Quinlan, JR, Compton, PJ, Horn, KA and Lazarus, L, 1986. "Inductive knowledge acquisition: a case study" In: *Proceedings Second Australian Conference on Applications of Expert Systems*. Sydney.

Rappaport, AT and Gaines, BR, 1990. "Integrated knowledge base building environments" *Knowledge Acquisition* **2**(1), 51–72.

Reinke, RE and Michalski, RS, 1988. Incremental learning of concept descriptions: a method and experimental results" In: JE Hay and D Michie (eds.), *Machine Intelligence 11*. Oxford Press.

Rendell, L, 1986. "A general framework for induction and a study of selective induction" In: *Machine Learning 1*(2), 177–226.

Rendell, L, 1988. "Learning hard concepts through constructive induction: framework and rationale" Technical Report, Univ. of Illinois at Urbana Champaign, UIUCDS-R- 88-1426.

Rendell, L, Seshu, R and Cheng, D, 1987. "More robust concept learning using dynamically variable bias management" In: *Proceedings 10th IJCAI'87*. Morgan-Kaufmann.

Schlimmer, JC and Fisher, D, 1986. "A case study of incremental concept induction" In: *Proceedings AAAI'86*. Morgan-Kaufmann.

Schlimmer, JC, 1987. "Incremental adjustment of representations for learning" In: *Proceedings 4th International Workshop on Machine Learning*. Morgan-Kaufmann.

Sebestyen, GS, 1962. "Pattern recognition by an adaptive process of sample set construction" *IRE Transactions on Information Theory* **IT–8** September, 82–91.

Shapiro, AD, 1987. *Structured Induction For Expert Systems* Turing Institute Press, Addison-Wesley.

Thornton, C, 1987. "Hypercuboid-formation behaviour of two learning algorithms" In: *Proceedings 10th IJCAI'87*. Morgan-Kaufmann.

Toussaint, GT, 1974. "Bibliography on estimation of misclassification" *IEEE Transactions on Information Theory* **IT–20** (4) July, 472–479.

Utgoff, PE, 1986. *Machine Learning of Inductive Bias* Kluwer.

Utgoff, PE, 1988. "ID5: an incremental ID3" In: *Proceedings 5th International Workshop on Machine Learning*. Morgan-Kaufmann.

Valiant, L, 1984. "A theory of the learnable" *Communications of the ACM* **27**(11), 1134–1142.

Vapnik, VN and Chervonenkis, AY, 1971. "On the uniform convergence of realative frequencies of events to their probabilities" *Theory of Probability and its Applications* **16**(2), 264–280.

Vapnik, VN, 1989. "Inductive principles for the search for empirical dependencies (methods based on weak convergence of probability measures)" In: *Proceedings Second Annual Workshop on Computational Learning Theory*. Santa Cruz. CA. Morgan-Kaufmann.

Watanabe, S, 1972. "Pattern recognition as information compression" In: S Watanabe (ed.), *Frontiers of Pattern Recognition*. Academic Press.

Watanabe, S, 1985. *Pattern Recognition: Human and Mechanical* Wiley.

Wilkins, DC and Buchanan, BG, 1986. "On debugging rule-sets when reasoning under uncertainty" In: *Proceedings AAAI'86*. Morgan-Kaufmann.

Winston, PH, 1975. *The Psychology of Computer Vision* McGraw-Hill.

Wirth, J and Catlett, J, 1988. "Experiments on the costs and benefits of windowing in ID3" In: *Proceedings 5th International Workshop on Machine Learning*. Morgan-Kaufmann.