The Knowledge Engineering Review, Vol. 12:4, 1996, 341–367

A review of machine learning

DAVID M. DUTTON[1] and GERARD V. CONROY[2]

[1]*Building Research Establishment, Bucknalls Lane, Garston, Herts WD2 7JR. 01923 664110, duttond@bre.co.uk.*
[2]*Department of Computation, UMIST, PO BOX 88, Manchester M60 1QD. 0161 236 3311.*

Abstract

This paper reviews Machine Learning (ML), and extends and complements previous work (Kocabas, 1991; Kalkanis and Conroy, 1991). Although this paper focuses on inductive learning, it at least touches on a great many aspects of ML in general. In addition, incremental induction is also reviewed. Therefore, a general review of ML is presented, but specific detail which has been covered previously is omitted, although other relevant references are noted, and later material is commented upon.

1 Introduction

In a general sense, learning can be defined as the process of improving one's ability to do a task. For this paper, we refer to *cognitive acquisition* (such as learning to recognise an object by acquiring a description of it), rather than *motor skill refinement*, such as learning to ride a bicycle (e.g. Langley et al., 1987).

Michalski (1994) refers to learning as the combined processes of *inferencing* and *memorising*, where inference is deemed to be any kind of *reasoning* or *knowledge transformation*. That is, inference results in the production of new knowledge. Functionally speaking, learning will often consist of filtering data to obtain 'interesting' information and then refining this in some way so that 'useful' facets remain. Thus data is distilled into 'knowledge', whence it is stored for future use.

ML's objective is two-fold: to learn about the spectrum of learning methods (e.g., human methods); and to endow computers with the same ability. According to Shavlik and Dietterich (1990), a system can learn either by acquiring new knowledge from an extrinsic source, or by modifying its own knowledge such that it becomes more effective. In the former case, learning entails the acquisition of knowledge without the act of directly programming it into a computer. In the latter case, the type of learning is termed *speed-up learning* or *skill acquisition* (Shavlik and Dietterich, 1990; Dietterich, 1986). Often the objective for a system is to acquire a description, or *intensional definition* of a given *target* concept. A *concept* can be thought of as a description of the common properties of a group of instances, or as a subset of a universe of objects, i.e., an *extensional* definition.

General overviews of the subject are given by Carbonell et al. (1983), Michalski (1983), Michalski et al. (1983, 1986b), Rendell (1986), Michalski (1986), Kodratoff (1988), Carbonell (1989), Kodratoff and Michalski (1990), Shavlik and Dietterich (1990), Kocabas (1991), Weiss and Kulikowski (1991) and Michalski and Tecuci (1994), which one should consult for more detail. Important early work is summarised in Dietterich and Michalski (1983), and includes work by Winston (Blocks World), Vere (THOTH), Hayes-Roth (SPROUTER) and Buchanan and Feigenbaum (DENDRAL). An excellent monograph is given by Angluin and Smith (1983). See also Kibler and Langley (1988).

Acquired knowledge can fall into many categories, such as theoretical, methodological, factual and technical (Kocabas, 1991). Inductive algorithms can be used to detect patterns, trends and structure in many domains, e.g., discover patterns in examples of chess end-games, without

reference to the rules of chess (Quinlan, 1983). The knowledge acquisition stage of expert system development has been recognised as a potential bottleneck (e.g., Michalski, 1983, 1986; Quinlan, 1979), and any system which can automate the process, with minimal help from domain experts, is to be encouraged. Indeed, ML may actually improve on expert written knowledge (Michalski and Chilausky, 1980), and experts themselves (Cestnik et al., 1987; Kononenko and Bratko, 1991). Similar techniques can be employed to refine knowledge-bases to improve completeness, consistency, simplicity and so on. For example, Shapiro (1987) uses *structured induction*, decomposing a problem into subproblems and using induction to solve each subproblem from the bottom up, building on previous solutions; Muggleton (1987) uses simplifying transformations to generalise data into a classificatory hierarchy; Michalski and Stepp (1983) discover structure in a domain by clustering like objects together hierarchically. Muggleton (1994) reports on the discovery of *new* knowledge, previously unpublished.

Other uses include fault diagnosis, condition monitoring, quality control and learning behaviour or properties (Donald, 1994). More recently *learning apprentice* algorithms have been used to update an expert system's knowledge base as it operates (e.g., Bareiss et al., 1990). A similar approach is used by Winkelbauer and Fedra (1991), where different learning modules may be 'plugged into' an *Automatic Learning Environment*. Recent research has seen the integration of several algorithms, together with knowledge representation and refinement methods, into the "Machine Learning Toolbox" (Sleeman, 1994). See also MLC++ (Kohavi, et al., 1994). Multi-strategy learning, where two or more disparate methods combine to augment each other, is presented in Michalski and Tecuci (1994), for example. Other systems directly integrate learning elements within an expert system (Wang and Chen, 1991). *Data mining* is a more recent development which entails the discovery of knowledge within databases (e.g., Holsheimer and Siebes, 1994; Fayyad and Uthurusamy, 1995; Piatetsky-Shapiro and Frawley, 1991; Fayyad et al., 1996; Cai et al., 1991; Zytkow and Baker, 1991). Thus, a fundamental goal of AI is the succinct representation and efficient application of complex knowledge (Hinton, 1990), and ML entails the acquisition, organisation and refinement of such knowledge.

This paper is organised along the following lines. A general review of the many influences on ML is intended, although many sub-parts are given in outline only, and as mentioned, the paper has a major emphasis on inductive methods, especially *incremental TDIDT*. Section 2 describes a number of knowledge representations as, after all, AI applications *must* address this issue as it has such an influence on algorithmic accuracy and efficiency, etc. (e.g., see the section on *bias* below). Section 3 describes one of many frameworks used for classifying and comparing ML algorithms via the *knowledge representation* and/or *level of abstraction* involved. This gives a brief overview of the field of ML. Section 4 then concentrates on *symbolic learning*, which is sub-divided into *supervised* and *unsupervised* learning. Again, supervised learning is given more in-depth treatment, and the major paradigms are described in Section 5. Then *decision tree induction*, with an emphasis on *incremental* approaches, is presented.

2 Knowledge representation

The use of an 'appropriate' knowledge representation is fundamental to all AI algorithms, as it can have such a profound effect. It dictates how efficiently and effectively an algorithm tackles a task, e.g. how much detail can be learned (i.e., sufficient expressiveness exists to permit the differentiation between concepts); whether concept definitions can be recorded succinctly and accurately; and whether concept definitions can be learned in polynomial time. A brief overview is therefore given of the typical representations used in ML. Knowledge representation has been well covered elsewhere (e.g., Kocabas, 1991; Kalkanis and Conroy, 1991), and so the following comments are intended to complement and extend such references. The following representations are touched upon: propositional and predicate logic, decision lists and trees, neural nets, genetic algorithms, semantic nets, frames and a number of others.

Many systems use propositional logic. Extensions aim to improve the expressive power of the language, and thus increase the space of representable concepts or descriptions. Such extensions include First Order Predicate Calculus (e.g., Vrain and Lu, 1988), but one might circumscribe the language and, say, learn simplified formulae, such as Horn clauses (e.g., Quinlan, 1990a; Muggleton, 1994). Others include Michalski's Annotated Predicate Calculus (APC) (1983) and variable-valued logics VL_1 and VL_2 (1980). Bain and Muggleton (1987) use *non-monotonic logic* to aid incremental processing. *Fuzzy logic* (Zadeh, 1994) adds degrees of 'imprecision' to predicates. Production rules in their simplest form, shadow propositional logic conditionals. Examples include Michalski and Chilausky (1980), Michalski et al. (1986a), Clark and Niblett (1987), Laird et al. (1984) and Donald (1994).

Decision lists are a more structured form of 'if-then' rules, i.e., one can think of them as a list of "if-then-else" rules. Rivest (1987) depicts each antecedent as a conjunct of k terms, and each consequent in $\{true, false\}$. Such lists can be represented as trees which have two outcomes per internal node (the right-hand child (by convention) leads to a decision, or leaf, whilst the left-hand node leads to a subtree or leaf, e.g., GREEDY3 in Pagallo and Haussler, 1989). See Shen (1992) for an incremental version.

A decision tree consists of a directed acyclic graph of nodes and arcs (e.g., Breiman et al., 1984; Quinlan, 1979, 1986a, etc.). Another representation formalism for trees can be seen in Cockett and Zhu (1989), where binary trees are represented *algebraically*. For a *binary encoding* technique for trees, see Quinlan and Rivest (1989). A more general structure, where successive levels are positive and negative exceptions can be seen in Bhandaru and Murty (1991). Donoho and Rendell (1995) utilise AND-OR trees, where branches are either AND'ed or OR'ed together and the leaves portray a concept description in DNF. *Set-enumeration* trees are described in Rymon (1993). Shlien (1992) generates a number of decision trees, and uses a meta-classifier to select the best ones for use in prediction. See also Swain and Hauska (1977).

Other representations include: neural networks (e.g., Sethi, 1991; Towell and Shavlik, 1994; Rumelhart et al., 1986); genetic algorithms (e.g., Goldberg, 1989; Vafaie and De Jong, 1994); frames (e.g., Blythe, 1988; Lenat, 1983; Núñez, 1988; and Manago and Kodratoff, 1991); semantic networks (e.g., Winston's 'Blocks World'—see Dietterich and Michalski, 1983); finite state automata (e.g., Angluin and Smith, 1983); exemplars (e.g., Aha et al., 1991; Bareiss et al., 1990); probabilistic approaches (e.g., Fisher's COBWEB 1987a, b), Schlimmer and Granger's STAGGER (1986a, b), also Quinlan (1987a); taxonomies/partitions (e.g., Decaestecker, 1991), and so on. For an in-depth discussion, see Ringland and Duce (1988).

An enlarged syntax increases expressiveness, but also computation (through complexity). On the other hand, greater expressiveness will often allow more compact representations, and thus reduce computation through reduced 'dimensionality' of the data. For a given 'input' language, attributes which consist of categorical values have the effect of discretising the example space. One can think of this as taking larger steps through the concept space.

Holte (1989) draws the distinction between a representation which is efficient with regard to the learning process, but renders the *performance task* inefficient (or even intractable), and *vice versa* (e.g., exemplars and trees). It should now be apparent that optimisation of efficiency and expressiveness is often mutually exclusive and represents a trade-off to be satisfied, possibly per domain. Rendell (1986, 1987a, b) makes use of a probabilistic representation for learning, and a simpler, more concise Boolean variation for subsequent transmission and/or explanation purposes (see also Kubat and Pavlickova, 1991).

Many systems employ fixed and implicit knowledge, i.e., a designer has assumed a problem domain and an associated performance system, thereby providing necessary constraints on the learning process. As Keller notes (1987), this constitutes *compiled* knowledge which is correspondingly difficult to change. To paraphrase Marr (1982), any particular representation makes explicit certain information, at the expense of other information which is 'pushed into the background' and consequently may be harder to recover.

3 Levels of abstraction for learning

This section offers an overview of a number of learning paradigms and briefly mentions 'subsymbolic' learning. Learning can be modelled on a number of different levels of abstraction. For the purposes of this paper, a simple taxonomy is used which models learning as symbolic or subsymbolic (numeric), according to the nature of the manipulations (reasoning/inference) which take place whilst learning. Dietterich (1986) concludes that there are two types of learning, *knowledge-level* learning, which equates to *knowledge acquisition*, and *symbol-level* learning which equates to *speed-up* learning (extant knowledge is used more efficiently). Kocabas (1991) also talks about learning at different levels, viz. the *knowledge*, *symbol* and *device* levels. These epithets are used in a different sense to Dietterich in that Kocabas uses this trichotomy to classify algorithms according to their *predominant knowledge representation*. Below, subsymbolic learning is briefly reviewed followed by symbolic learning, in the subsequent section.

3.1 Neural nets

Overviews of this topic are given in Rumelhart et al. (1986), Blum (1992) and Sethi (1991). Sethi (1991) describes nets with two hidden layers as being able to form *decision boundaries* of any complexity. The first hidden layer partitions the feature space into several regions whilst the second hidden layer ANDs regions together to form convex regions for each class. The output layer then ORs output from the previous layers together to form disjoint regions of arbitrary shape. During learning, networks may distribute a concept representation across many units, or may dedicate neurons to individual 'subtasks' (Hinton 1990; Carbonell 1989).

3.2 Genetic algorithms

Genetic algorithms (GAs) are intended to correspond to the natural selection procedures observed in real life, and are basically an optimising search technique (Goldberg, 1989). For ML, the idea is to induce the best concept description, therefore individuals often start as examples of concepts and "grow" into hypotheses in the concept-space. Fixed length binary strings represent attribute lists, and also concepts. Often a *don't care* value is also used, e.g., "#", and so one can form generalisations. Goldberg (1989) calls these *schemata*, which are templates describing subsets of strings.

Spears and De Jong (1990) do not rate fixed length representations highly for general symbolic concepts and instead stipulate that their algorithm will learn fixed length rules. That is, the antecedents of rules will have a fixed number of attributes but each attribute has a variable number of possible values, represented by a bit string (one bit per *value*). Each population individual is a variable length *list of separate rules*. An alternative approach would have each possible rule as an individual in the population, and this kind of approach can be seen in Quinlan (1988b). Another use is demonstrated in Vafaie and De Jong (1994), where a GA is used to select feature subsets for use in discrimination rules. Wnek and Michalski (1994) use a GA to help learn rule antecedents and associated strengths[1].

3.3 Statistical approaches

Statistical techniques entail the construction of a probabilistic model to describe a 'sample', from which it is possible to draw conclusions about a population in general. Estimating the parameters of a given probability density function for a variable is one typical application, e.g., mean and variance (see Duda and Hart, 1973; Hawkins and Kass, 1982; Breiman et al., 1984). Statistical work which is

[1] Some authors prefer to think of GAs as symbolic (e.g., Carbonell, 1989; Kocabas, 1991), and others as subsymbolic, (e.g., Wnek and Michalski, 1994; Vafaie and De Jong, 1994). Spears and De Jong (1990) state that GAs are general purpose, and can be used in either setting.

of interest to the ML community mostly concerns *inductive inference*, where one forms hypotheses based on data. Here, inference is based more directly upon probabilities within a given *sample* of observations.

Numerous techniques exist for investigating relationships between variables[2]. For example, *regression* attempts to determine the nature of a relationship between two variables (e.g., fit a curve to a scatter diagram of data). *Bayesian techniques* describe decisions, etc., in terms of *losses* or *risks*, e.g., one incorrect prediction may be more costly than another (e.g., Hoel 1984; Duda and Hart, 1973). A Bayes rule classifier is said to be optimal in that it depicts the minimum error one can expect in a domain.

Cluster analysis is described in Hawkins and Kass (1982) and Duda and Hart (1973), and covers many techniques used to sort and classify unlabelled examples into categories, possibly hierarchically, on the basis of 'similarity' measures. Clustering is basically an iterative optimisation problem to find the best partition, and is computationally intensive. A similar approach can be seen in *nearest neighbour methods* (see also Weiss and Kulikowski, 1991; Duda and Hart, 1973).

Statistical methods are often *parametric* in that they make assumptions about the data in use. For example, the form of the probability density function is given *a priori*, and even that it is "normal". Some parameters may be estimated from the current sample, often with good results, provided a sample is large enough for underlying assumptions to remain justifiable (e.g., see Fatti et al., 1982). Parametric methods therefore leave themselves open to criticism concerning the validity of assumptions and their generality.

3.4 Reinforcement learning

Reinforcement learning is a paradigm where an 'episode' of temporally ordered states occurs. At each step the algorithm performs an action which is then evaluated for its quality and the algorithm receives a reinforcement, either 'reward' or 'punishment', according to the quality of the action (Cichosz, 1995). Transition to the next state is then effected and the process reiterates (transitions and reinforcements may be stochastic), and the goal is to learn to predict a future outcome. *Temporal difference* learning, or $TD(\lambda)$, aims to extract information from such sequences to predict future outcomes, and λ is a factor which determines how many states are used, and their relative weight. For example, $TD(0)$ uses the current state only (Dayan and Sejnowski, 1994). Such techniques are often used in "control" to track the movement of an entity for instance, and may be implemented in a neural net. Dayan and Sejnowski (1994) prove the convergence of $TD(\lambda)$ algorithms generally but note that the speed of this convergence may be too slow in the "real-world". Traditionally, learning would wait until an 'outcome' is known, and having kept track of all predictions, calculate the difference between actual and predicted states to compute an error on which to 'train'. In $TD(\lambda)$, one does not wait for an outcome, and the 'error' between two states is used to speed learning (Cichosz, 1995).

3.5 Hybrid approaches

Attempts have been made to combine symbolic and subsymbolic algorithms, such as neural nets and decision trees (Sankar and Mammone, 1991; Utgoff, 1988b) and others (Towell and Shavlik, 1992; Sethi, 1991). Utgoff and Brodley (1990) use *linear threshold units* (LTU) at each tree leaf. A LTU is a linear combination of (so far unused) attribute values, each with an associated weight. The weighted sum resulting from the input of an example is compared against a threshold. If the result is less than the threshold, the example is deemed to be negative, otherwise positive (the polynomial defines a hyper-plane).

Some algorithms select the number of neurons required for a neural net as they train (e.g., Sankar and Mammone, 1991). It is possible to translate symbolic rules into neural nets (Towell and Shavlik,

[2] Discussion is limited to two variables, but multivariate analysis is commonplace.

1992), refine the nets and then retrieve symbolic rules (Towell and Shavlik, 1994), thereby overcoming some criticisms of neural nets (e.g., opacity and inability to justify solutions). Cios and Liu (1992) say a decision tree corresponds to a hidden layer in a neural net. Sethi (1990, 1991) notes that a decision tree path represents an ANDing of tests, whilst several paths represent an ORing of terms. Sethi adds that neural nets perform similar operations across layers, and so a tree can be converted into a net using transformation rules. Other *multistrategy* approaches may be seen in Michalski and Tecuci, (1994), e.g., Vafaie and De Jong (1994). One might also include statistical packages here, e.g., CART (Breiman et al., 1984) and AID (Hawkins and Kass, 1982), designed for statistical analysis, but adaptable to ML. It would appear too that Schlimmer and Granger's STAGGER (1986a) is a combination of symbolic and subsymbolic paradigms as it uses symbolic terms together with weights to describe a given concept.

3.6 Linear separability

Another way in which distinctions can be drawn between the different paradigms is with regard to the complexity of the decision regions produced. With real data especially, an algorithm's ability to delineate inter-class boundaries determines its ability to learn accurate and concise concept descriptions. In N-space, boundaries comprise *surfaces* of $N-1$ dimensions (a *hyper-plane* if the describing function is linear). Problems are said to be *linearly separable* if the example space can be linearly dichotomised (i.e., a linear function describes a separating boundary). One potential disadvantage with decision trees is their reliance on hyper-rectangular regions. That is to say, decision boundaries are orthogonal to the example space axes. Other algorithms within the symbolic paradigm may use more complex decision functions, e.g., constructive induction (Breiman et al., 1984; Murthy et al., 1994). In comparison, some neural nets *can* represent non-linear functions of attributes and thus decision regions in the concept-space are no longer restricted to hyper-rectangles.

Numerous authors have compared the symbolic decision tree algorithm ID3 and the neural net algorithm Back-Propagation (BP) as exemplars of both paradigms (e.g., Fisher and McKusick, 1989). Fisher and McKusick conclude that ID3 is quick to learn and quick to reach a relatively high level of accuracy, whilst BP's learning curve is more gradual. Thus, one might surmise that symbolic algorithms take bigger steps in the search space but may over- or under-step a solution.

Quinlan (1993b) describes neural nets, GAs and instance-based methods as "distributed" representations whilst decision trees and logic are not. The implication is that the former types are less susceptible to small alterations, e.g., classification does not rely on one small part of a description. Hence distributed representations *may* be more robust in the presence of noise.

4 Symbolic learning

Together with subsequent sections, this part represents a major emphasis of this paper, i.e., a review of symbolic induction. With previous sections in mind, the reader should now be in a position to compare the topics below according to the fundamental knowledge representation and reasoning method(s) used.

There are a number of types of symbolic learning which, in a theoretical sense, differ in the amount of knowledge required and the level of inference performed. Obviously, the less information required to start the process the better, and the greater the level of inference, the less emphasis is placed on a user, or 'teacher'; and the greater the level of inference, the more potentially complex and/or voluminous the derived knowledge becomes. The basic fields are: rote, learning by being told, deduction and induction. These have again been covered elsewhere (e.g., Kocabas, 1991), and additional or complementary material only is covered below.

4.1 Deduction

Deduction is a basic logical "truth preserving" operation. Given a 'rule' such as $P \rightarrow Q$ and a fact P, then one can deduce that Q is also true. Such rules form the basis of a *theory* (set of axioms, facts etc.). Herein lies the rub; one must have a theory with which to operate. Consequently, any theory must be applicable to the domain of application and ideally be complete and correct. Deductive theories are necessarily tailored towards a particular domain and the utility of a theory will diminish as the current domain progressively diverges from the intended domain.

This type of learning is also known as *analytical learning* (Carbonell, 1989). It includes *macro-operator* creation (e.g., STRIPS in Fikes and Nilsson, 1971), *chunking* (Laird et al., 1984), *explanation-based learning*, and others (Kocabas, 1991). *Macro* learning consists of learning aggregated operators, i.e., a sequence of steps which can be applied to speed up reasoning. *Explanation-based learning* consists of building a proof and modifying this to give a general explanation (e.g., Mitchell et al., 1990). *Chunks* are sections of search paths used in the solution of a problem, similar to the above macro-operators. These chunks are substituted for more low-level operator sequences when they are applicable[3].

4.1.1 Explanation-based learning

Many researchers hold that, to learn complex concepts, etc., one must start with a substantial amount of knowledge (see also Kocabas, 1991). This is intuitively appealing, if only from an observation of one's own experiences. In this paradigm, complex background knowledge, or theory, is applied to few examples (typically one), to generate an *explanation* of why an example satisfies the definition of the concept in question. The general mechanism views an example with reference to the concept to be learned, and generates an explanation in terms of the extensive background knowledge. The relevant features of the example are identified, and the example can then be generalised (this is termed explanation-based *generalisation* (EBG), e.g., Mitchell et al., 1990), to form the target concept definition. See also Bareiss et al. (1990). Minton (1988) uses EBL to learn search control knowledge, i.e., knowledge used to direct or constrain the search for a solution. Acquiring knowledge has an associated cost and it must be effective knowledge otherwise the cost will outweigh any gain it entails. Minton terms this the *utility* problem, i.e., there is a trade-off between search and knowledge.

Tadepalli (1989) notes that it may not be possible to *prove* that an example belongs to a target concept in some complex domains and thus one is confronted by the *intractable theory problem*. In such a case, *theory revision* may be possible.

4.1.2 Analogy

Analogical and *case-based* reasoning (CBR) (Kolodner, 1993; Watson and Marir, 1994) have many similarities and will be discussed together, in common with Sleeman (1994). CBR is reasoning based on previous experience adapted to new situations to solve similar problems, and can thus be seen to cover the whole reasoning cycle (Kolodner, 1993). *Derivational analogy* can capture extra information during problem solving (e.g., justifications at decision points), which can provide constraints to adaptation during subsequent use (see also Kolodner, 1993; Watson and Marir, 1994). CBR can be seen to include both deductive and inductive learning (e.g., Michalski, 1986, 1994; Kocabas, 1991). See also Bareiss et al. (1990).

4.2 Induction

Induction is often characterised as "learning as search" (see Mitchell, 1982; Michalski, 1983; Angluin and Smith, 1983). Michalski (1994) defines learning as a search through a *knowledge space*[4]

[3] *Chunking* is used as a general mechanism for learning in SOAR (Laird et al., 1984).
[4] Also *hypothesis* or *concept* space.

as defined by the particular knowledge representation used. Laird et al. (1984) hypothesise that all complex behaviour can be characterised as search, and that this search is heuristic and goal-directed, i.e., there is some performance task in mind.

One can characterise a search procedure in terms of *operators* for manipulating the current search state. These operators typically entail specialisation or generalisation of the current hypothesis. Thus, concept descriptions are partially ordered by generality (Fisher, 1987b). Michalski (1983) also uses a *reformulation* operator whilst Schlimmer and Granger (1986b) may *invert* (negate) part of a description.

Langley et al. (1987) extol the virtues of *hill-climbing* as a plausible model for learning. Gennari et al. (1989) describe this method as 'cheap' (in terms of memory), in that it avoids combinatorial explosions by maintaining only one current hypothesis, but that it is susceptible to example ordering, operator step size and local minima.

4.2.1 Unsupervised induction

Unsupervised induction, (also known as *concept formation*) implies an algorithm is supplied with, or acquires, unclassified examples, and is required to decide which examples belong to which classes, and indeed, what the classes are. This area can again be subdivided into *clustering* and *discovery* (Shavlik and Dietterich, 1990). Gennari et al. (1989) review concept formation in more depth.

4.2.1.1 Conceptual clustering. Michalski and Stepp (1983) consider *taxonomic description*, or *conceptual clustering*. Clusters are groups of examples which have high intra-class similarity and low inter-class similarity. The problem, then, is to cluster examples and then to characterise them, which gives rise to a two-phase search: firstly through a space of clusters; and then a space of concepts (Fisher, 1987b). Hierarchies often stipulate a generality ordering (decreasing downwards) of concept definitions, and thus inheritance of properties from one level to another is assumed (e.g., Langley et al., 1987). Note that in contrast, *partitions* are flat, i.e., non-hierarchical (Decaestecker, 1991). Clustering algorithms typically use a *distance* function to judge the difference, or similarity, between examples and/or clusters. Fisher (1987b) briefly reviews similar systems.

Rendell et al. (1987) describe an incremental clustering algorithm, PLS1, which defines regions in the instance space from the top-down. One starts with a large "hyper-rectangle" defined by all training instances and successively refines this into smaller regions on the basis of *dissimilarity* (similar to entropy).

Fisher's COBWEB (1987a, b) constructs probabilistic, hierarchical concepts with a view to improving *inferential ability*, i.e., the prediction of missing values during classification. COBWEB uses a statistical evaluation function known as *category utility* to find the classes inherent in the training examples. COBWEB uses nominal attributes whilst CLASSIT uses numerical values (Gennari et al., 1989). CLASSIT may handle noise better as it does not necessarily propagate all examples to leaves, and thus aims to avoid overfitting. An augmented version of COBWEB, known as ECOBWEB is described by Thrun et al. (1991).

A similar approach is seen in UNIMEM (Lebowitz, 1987, 1988), which can use both numeric and nominal attributes. UNIMEM also attaches 'certainty' type factors to descriptions for use in pruning and "fixing", i.e., once a description exceeds a given threshold, it cannot be pruned, and if it should fall below another, it will be pruned away. UNIMEM also allows non-disjoint partitions allowing greater flexibility.

A similar approach is seen in ADECLU (Decaestecker, 1989, 1991), where concept descriptions (one per node as above) consist of the collection of attribute values common to all examples in a node. As for COBWEB, each new example might cause further refinement of the concept at that node, via merge, split or create operators (*create* results in a new concept populated solely by the example). A subsequent version, ADECLU2 allows the selection of attribute value subsets in descriptions (COBWEB uses all).

Michalski and Stepp (1983) note that most algorithms do not take the context into account, i.e., surrounding examples in the example space. They suggest the use of a concept-sensitive algorithm

(CLUSTER/2) which also uses an externally defined set of concept definitions (i.e., the algorithm is supplied with a set of *allowed* clusters). Thus the *conceptual cohesiveness* of two *points* depends upon those points, the surrounding points and the set of concepts available to describe them. See also Iba et al. (1988).

4.2.1.2 Learning by discovery. This type of learning is also known as *learning from observation*. An algorithm observes its environment and induces 'concepts' by itself. Algorithms in this class can either simply observe the environment (passive), or they can *cause* changes in it and observe the effects (active). *Learning by experimentation* can be added here, e.g., LEX (Mitchell et al., 1983) generates problems in symbolic integration, solves them, and then analyses the solution for improvements (i.e., it learns heuristics in the form of production rules).

Schoenauer and Sebag (1990) describe a discovery system which performs operations on problem attributes reminiscent of constructive induction. In the example given, the values of two attributes are found to vary in a similar fashion and so are combined into one, whereupon examples are redescribed in this extra attribute. Subsequent similarities allowed further accretion of this composite attribute with lone attributes to the extent that a mathematical law was "rediscovered".

Lenat's (1982, 1983) work on AM and EURISKO is included here. AM gathered empirical data, "noticed" regularities within it, and then defined new concepts which related the observations. It uses a large body of heuristics ("compiled hindsight"), provided as background knowledge, to recognise areas for investigation. See also Langley et al. (1987), Shrager and Langley (1990), and Valdes-Perez (1996, 1995).

4.2.2 Supervised induction

In contrast to the previous section, *supervised* induction relies upon a set of *pre-classified* training examples which are (ideally) indicative of the concept(s) to be learned. An algorithm 'learns' to discriminate between the *given* concepts, in contrast to the often characteristic-type descriptions resulting from unsupervised induction (Reinke and Michalski, 1988; Carbonell, 1989). This section introduces a number of factors of general concern to supervised induction, viz. learning theory, bias, concept quality and noise.

4.2.2.1 Computational learning theory. Angluin and Smith (1983) and Valiant (1984) have instigated much work on the theoretical aspects of learning, e.g., Valiant's *Probably Approximately Correct* (PAC) learning model. Others develop this topic further (Angluin and Laird, 1988; Natarajan, 1991; Kearns, 1990). These considerations are beyond the scope of this paper. See also Kalkanis and Conroy (1991).

4.2.2.2 The role of bias in learning. It has been noted (Michalski, 1983; Rendell et al., 1987) that, computationally speaking, induction without bias is impossible, it is the *sine qua non* of induction. Thus one requires to shrink or *prune* the search space. One should note, then, that bias may affect the generality of an algorithm. Utgoff (1986) notes that a *strong* bias will focus an algorithm on a relatively small number of consistent hypotheses, and a *correct* bias allows the algorithm to discover the intended target concept. Buntine (1990) decomposes bias further into *functional components*, viz. a *hypothesis space bias* (see below), an *ideal search bias* (which defines what an algorithm should search for), and an *application-specific bias* (for the application, or purpose, in mind). An appropriate bias can facilitate "inductive leaps" through the concept space (Mitchell, 1982), and thus aid inductive efficiency. Sleeman (1994) adds that the ultimate purpose or objective of the learning process should provide the most important bias for any system. Rendell (1987b) refers to *multi-layered* learning in which example, hypothesis, and bias *spaces* are integrated so as to exploit advantages offered by each. Rendell et al. (1987) describe bias as either *fixed, parameterised* or *dynamic*. Fixed bias is set within an algorithm, whereas parameterised bias can be adjusted at the start of a learning session. Dynamic bias is alterable during a run. Further flexibility is afforded by maintaining many biases, associated with different problems. See also Kocabas, 1991; Kalkanis and Conroy, 1991.

If bias is implicit then it is not revisable, whereas explicitly encoded bias can be, in effect allowing an algorithm to reason with it. In SOAR, Laird et al. (1984) include as much information as possible in an explicit, declarative sense. Utgoff (1986) details the algorithm STABB which will detect an *inappropriate* bias and shift to a 'better' one. Similar work can be seen in Rendell et al. (1987), where a variable bias management system (VBMS) is presented, which also learns to associate biases with problem classes.

Michalski (1983) makes use of a *preference criterion*, i.e., a set of explicit bias terms altered by a user to influence the choice of consistent descriptions. Núñez (1991) augments decision trees with other background knowledge in the form of *is-a* hierarchies (a generalisation hierarchy connecting attributes) and attribute "costs". The generalisation process is extended from the basic decision tree method of "dropping conditions" to include "climbing the generalisation tree" and "adding alternatives" (via background knowledge). This extra flexibility may decrease bias but may also hamper speed. See also Haussler (1986).

4.2.2.3 The quality of learned concepts. There are a number of dimensions over which one can judge quality, and these include accuracy, clarity, efficiency, applicability and so on. Safavian and Landgrebe (1991) attempt to optimise *decision trees* with respect to error rate, path length, number of nodes and information gain. The generation of optimal decision trees is NP-complete[5] (Quinlan, 1990c; Hyafil and Rivest, 1976).

Cheng et al. (1988) suggest six measures for evaluating descriptions in *tree* form, including: the number of tree nodes, the error rate, the number of leaves, the number of internal nodes, the average example support per leaf, the number of decisions per test set example, etc. Fayyad and Irani (1990) perform a similar analysis of concept evaluation criteria, and that, whilst accuracy is normally the most important criterion, its measurement can be time-consuming and error prone. They contend that minimising the number of leaves in a tree will bring an improvement in all other dimensions above (and that most of these are related). Probabilistically, reducing the number of leaves can be expected to reduce the error rate of a tree (Quinlan, 1990c; Fayyad and Irani, 1990). This is in fact borne out by Murphy and Pazzani (1994), where error rate increased monotonically for *most* increasing tree sizes (except those near to the minimal size)[6]. It is generally believed that a smaller tree will be more comprehensible (to experts), and will better generalise the training data (e.g., Bohanec and Bratko, 1994).

Various methods exist for determining accuracy: *resubstitution*, *train-and-test, random sampling, repeated random sampling, K-fold cross-validation* and *0.632 bootstrap*. See Kocabas (1991), Kalkanis and Conroy (1991), Donald (1994), Breiman et al. (1984) and Safavian and Landgrebe (1991).

Wnek and Michalski (1994) use *exact error rate* which is defined as *exact error* divided by the size of the example space. Exact error is the number of differences between the learned and the target concepts. Concepts are represented diagrammatically as groups of "cells", where one such cell represents one position in the event space (one unique combination of all attributes and values). Errors can be errors of *omission* (false negatives, "FN"), and errors of *commission* (false positives, "FP"), but see Wnek and Michalski (1994) and Weiss and Kulikowski (1991). Correct classifications are similarly labelled "true positives" (TP) and "true negatives" (TN) (Kononenko and Bratko, 1991)[7]. An example use of this is for judging classifier sensitivity, e.g., $TP/TP+FN$. Indeed, Kononenko and Bratko analyse such facets of the "gross" error rate estimate to determine performance more precisely. Other factors involved concern the use of prior class probabilities in calculating performance, i.e., the misclassification of an example of a more likely class is worse than for a less likely class. Kononenko and Bratko use an entropy-based calculation to more accurately gauge the (probabilistic) amount of information contained in an 'answer'.

[5] For whatever dimension, e.g., size.
[6] Note that this only applies to trees consistent with the training set.
[7] Assuming two classes.

Haussler (1986) uses two performance measures, viz. *convergence rate* and *computational efficiency*. He suggests that these dimensions are mutually exclusive, and thus represent a trade-off. *Convergence rate* is defined as the size of the example set required to learn a hypothesis of a given error rate and confidence (probability). In Wnek and Michalski (1994), it is defined as the number of examples seen before the concept prediction accuracy reaches 95%. *Computational efficiency* is the time taken to the final hypothesis. Concept or hypothesis *significance* is another method (see Muggleton et al., 1992).

4.2.2.4 Learning in the presence of noise. Noise generally implies that the particular attribute is inadequate for the learning and classification tasks. According to Weiss and Kulikowski (1991), any attribute which is no more predictive than chance can be considered to be noisy. However, it is noted that an attribute may appear noisy in isolation, but actually be highly predictive (of a class, etc.) when used in conjunction with others (i.e., polythetic analysis, e.g., see Seshu 1989). Mingers (1989) mentions *residual variation* where inadequate attributes mean that class-attribute relations cannot be "properly" explained. Schaffer (1991) notes that attribute noise has a greater effect on an algorithm's performance than class noise. Natarajan (1991) describes *malicious noise* where whole examples are replaced. Similar problems occur when attribute values are missing. Muggleton et al. (1992) use *compression* to identify noisy data. That is, a generalisation allows data to be 'compressed' into a hypothesis and the degree to which this is possible can be used to indicate its *significance*. Incompressible data is thus deemed to be noisy. See also Quinlan (1986b, 1987b, etc.).

4.3 Discussion

The above paradigms can be used to draw up a taxonomy of learning algorithms. The level of inference undertaken in each has been described, and is the basis of the said taxonomy. One can view an alternative taxonomy based on the amount of initial, or *a priori* knowledge built into an algorithm. Thus, neural networks might be considered relatively knowledge-free, whilst EBL is knowledge-rich.

Induction works best when there are many examples, whilst EBL algorithms require very few. On the other hand, induction algorithms require relatively little background knowledge, or domain theory, and are thus easier to implement, easier to use and are potentially applicable to a wider range of domains. Sleeman (1994) suggests that in some complex domains, the definition of an extensive theory, or bias, might not be possible. Alternatively, Manago and Kodratoff (1991) do in fact use complex background knowledge and deduction within a mainly inductive setting.

According to Kocabas (1991), "similarity-based" techniques (e.g., ID3) are limited by their inability to generate *justified* generalisations. EBL systems address this issue and both techniques may be complementary. Effective learning in many domains could result from integration (see also Michalski and Tecuci, 1994, and cf. Bareiss et al., 1990). However, as intimated above, the requirement of an extensive domain theory can be a severe limitation, both to an EBL system's effectiveness and its range of applicability. EBL systems can suffer from incomplete domain theories, inconsistent theories and intractable theories (Kocabas, 1991). A significant proportion of Michalski and Tecuci (1994) is given over to *theory revision*. Furthermore, the integration of context-specific cases/exemplars together with general rules and models may be *required* to improve the robustness and "quality" of future systems (e.g., see David et al., 1993).

5 Major paradigms for supervised induction

This section describes a number of paradigms in the sub-field of supervised induction. These include *inductive logic programming, instance-based* and *rule-based* learning and *decision trees*. A major emphasis is placed upon the decision tree paradigm, and this is explained in some depth.

5.1 Inductive logic programming

Algorithms in the ILP paradigm learn Prolog clauses, from examples and background theory (clauses) (e.g., Quinlan, 1990a; Muggleton, 1994; Cameron-Jones and Quinlan, 1994). Groups of these clauses then form *logic programs* (concept definitions). ILP has its roots in the CIGOL algorithm (Muggleton, 1988; Muggleton and Buntine, 1988). Muggleton (1994) holds that this formalism is more general than the "classic" empirical learning due to the use of first-order relational logic, and also differs from EBL due to its use of (possibly) incomplete and/or incorrect background theory. Thus, it can deal with structured objects, relations, k-ary predicates, and so on. Logical theorem-proving forms the basis of such program derivations, and if formulae are constrained to Horn clauses, a Prolog interpreter can suffice. Such generality leads to large concept spaces and *extra-logical* constraints may be required to maintain efficiency. Statistical constraints may be used to rank hypotheses, whilst language bias restricts expressiveness, e.g., by providing rule models or templates to describe clause form. A hypothesis H cannot be *deduced* from background theory B and positive example set E^+ (Muggleton, 1994), but one can rewrite $B \,\&\, H \models E^+$ as $B \,\&\, \bar{E}^+ \models \bar{H}$ and then use (deductive) theorem-proving to get the negation of H.

Quinlan's FOIL (Quinlan, 1990a; Cameron-Jones and Quinlan, 1994) is more closely related to ID3, i.e., top-down induction using a heuristic based on entropy. The emphasis is on an accurate, compact definition rather than one which is exact. FOIL uses a "covering" method to learn Horn clause relations from examples of the relation. As long as a positive example remains and a logical clause to cover (satisfy) it can be found, the clause is added to the current definition, and any positive examples covered are removed. Finally, the definition is pruned to remove redundant clauses. Quinlan makes use of the Minimum Description Length Principle (Quinlan and Rivest, 1989) to infer *compact* definitions, and a later version mFOIL, adds noise tolerance (Thrun et al., 1991). Muggleton et al. (1992) use hypothesis compressibility to identify and 'avoid' noise.

5.2 Exemplar/nearest neighbour algorithms

These are also known as Instance-Based Learning (IBL) algorithms (e.g., Kibler and Aha, 1990; Aha et al., 1991; Aha, 1992; Salzberg, 1990; Quinlan, 1993a; Bareiss et al., 1990); (cf. statistical *nearest neighbour*, and CBR).

An IBL algorithm consists of (Aha et al., 1991):

- a *similarity function* which calculates the similarity between an example and the examples in a concept definition;
- a *classification function* which takes the similarity function's output and performance records of examples in the description and produces a class for the current example;
- a *concept description updater* which maintains performance records and decides which examples to include in a description.

Kibler and Aha identify a number of models:

- the proximity model, where all examples are stored, with no abstraction;
- the best examples model, which assumes the existence of prototypes, and thus only stores typical examples;
- the selected examples model, which does not assume the existence of only one prototype, and thus may store more general examples.

Weiss and Kulikowski (1991) describe three types of distance measure (for calculating similarity), such as *absolute distance*, where the differences between the values of each attribute are summed. Others include *Euclidean distance* and *normalised distance*, that is, some attributes may be scaled differently, i.e., have different sized units, so one normalises all ranges to be within certain limits.

Salzberg (1990) uses "nested generalised exemplars", where some generalisation is allowed. A generalisation is a hyper-rectangle in the instance space. These hyper-rectangles can be nested within

one another, and thus concept definitions can have exceptions which is an efficient, space saving addition. Each exemplar in memory has an associated probability that can be used in prediction, i.e., each exemplar has a record of its successes and failures as a predictor. Many exceptions lead to a fragmented concept space. Thus, to cope with noise, exemplars are allowed to 'decay' over time, and will eventually disappear if not 'reinforced'.

5.3 Rule-based algorithms

Rules are closely related to decision trees[8], i.e., it is a simple task to reform trees as rules (e.g., Quinlan, 1987c), but many algorithms learn rules from the outset. For example, see Michalski (1980, 1983), Schoenauer and Sebag (1990), Shen (1992) and Wnek and Michalski (1994). In CN2 (Clark and Niblett, 1987, 1989), for example, rules consist of conjunctions of attributes which predict class membership. CN2 generates rules one-by-one, by searching for a rule, adding it to the rule-base, and then removing examples covered by (satisfying) it. The rule search entails looking for conjunctions of attributes that are satisfied by as many examples of *one* class as possible. The search starts with all possible most general rules, i.e., consisting of one attribute each. Each rule is evaluated for quality, statistical significance and the possibility of specialisation. Specialisation, generalisation and reformulation operators for rule manipulation are given in Michalski (1980, 1983).

5.4 TDIDT

In the following sections, TDIDT is covered in more depth, and issues such as attribute selection and decision function complexity are discussed. The section concludes with a description of constructive and incremental induction.

Quinlan (amongst others) gives in depth analysis of many decision tree facets (e.g., 1986a, b, 1987a, b, c, 1988a, etc.). Top-down induction of decision trees is, again, covered in Kalkanis and Conroy (1991) and Kocabas (1991). In addition, Quinlan (1993b), Breiman et al. (1984), Crawford (1989), Utgoff (1995), Sethi (1990), Weiss and Kulikowski (1991) and Mingers (1989), amongst others, offer useful discussions and extensions.

5.4.1 Attribute selection

Attribute selection is the method by which trees are generated and attributes are often selected on the basis of example set *impurity* (with respect to example classes). There are a number of methods for defining *impurity functions* and Fayyad and Irani (1992) list desirable properties:

1. Maximise inter-class separation.
2. Minimise intra-class separation.
3. Be sensitive to permutations in class distributions.
4. Be smooth (differentiable) and symmetric with respect to classes.

Such functions include: entropy (Quinlan, 1986a), AMIG (Sethi, 1990; Sethi and Sarvarayudu, 1982), Gini diversity index, Twoing criterion (Breiman et al., 1984), Lopez de Mantaras (1991), and so on.

Quinlan and Rivest (1989) introduce the Minimum Description Length Principle (MDLP) as applied to the construction of decision trees. Trees are encoded in bit form with a view to minimising the "cost", i.e., the description length of both the tree and any misclassified examples. It is as if one wished to communicate this data in the shortest possible message. The method characterises the inherent trade-off between tree accuracy and tree size, i.e., one wants the smallest tree, but also the most accurate. The MDLP is used in place of entropy to select attribute tests, and subsequently to prune away subtrees. A fully grown tree is repeatedly pruned back, i.e., whenever the removal of a node improves total communication cost for tree plus misclassifications. Variations and others are

[8] If based on propositional logic.

given in Murthy et al. (1994), Núñez (1991), and so on. Mingers (1989) gives an excellent survey of this area, including the use of χ^2 for attribute selection (see also White and Liu, 1994), and Lopez de Mantaras (1991) uses a "distance-based" selection metric. Ben-David (1995) criticises 'entropy-based' trees for "non-monotonic" behaviour with respect to classification, for domains with ordered classes (e.g., credit ratings). Non-monotonic trees, for example, may grant a high credit rating to an applicant when another, more suitable one is refused. Ben-David adds a weighted item to the usual entropy calculations such that a measure of "ambiguity" is included, which is designed to minimise non-monotonicity. *Ambiguity* is defined as the log of the ratio of the number of actual non-monotonic branch pairs, to the total possible number of such pairs. A branch pair is a node test plus a leaf class (e.g., in a tree path with three attributes A_1, A_2, A_3, and class $+$, one gets three pairs: $\{A_1, +\}, \{A_2, +\}, \{A_3, +\}$), and these pairs are checked against all others for non-monotonic behaviour.

5.4.2 TDIDT decision functions

This section discusses the types of decision "boundaries" resulting from the attribute selection process; it follows on from Section 3.6. Fisher has described attribute selection in trees as *monothetic*, as only one attribute is considered at a time. *Polythetic* classifiers on the other hand, can consider groups of attributes for a given decision (e.g., Fisher, 1987a, b; Schlimmer and Granger, 1986b). Typical decision tree algorithms look for relationships between one attribute and a class, i.e., they assume that attributes are independent of one another (Quinlan, 1988b; Cios and Liu, 1992; Hawkins and Kass, 1982).

ID3 suffers from limited lookahead in that it selects the 'best' attribute for splitting, according to entropy. ID3 could improve this situation by performing a k-ply lookahead, and thus maximise information gain over k attribute tests (levels of a tree). This would be of benefit in "j-of-k" problems (e.g., parity; see Seshu, 1989), where a single attribute is useless for classification. Extensive lookahead is computationally infeasible in all but the smallest of attribute sets. ID5 (Utgoff, 1988a), for example, rectifies this situation to a certain extent by adding (quasi-) backtracking which can 'undo' suboptimal choices. IDX (Norton, 1989) is based on ID3 but uses limited lookahead, as dictated at run-time, to select tests in each tree level. Trees become more balanced, and average tree depth is consistently reduced. IDX also took note of individual test costs, and is able to delay or ignore the more expensive ones. See also Murthy and Salzberg (1995).

Other approaches use combinations of attributes at a node, such as constructive induction. Utgoff and Brodley (1990) use LTUs (linear threshold units) to learn multivariate decision functions at tree nodes. LTUs consist of the untested attributes and weights, and training moves the hyperplane they represent about the example space, attempting to dichotomise the example classes. Such a hyperplane is of any orientation. Murthy et al. (1994) use *oblique trees* where a hyperplane is 'situated' at each tree test node. Breiman et al. (1984) use Boolean and linear combinations.

Cheng et al. (1988) and Fayyad and Irani (1992) present a generalised version of ID3 named GID3. Some values emanating from a node may be irrelevant to classification (identified statistically) and so these are grouped into a single "default" branch. A similar algorithm is seen in PRISM (Cendrowska, 1987; Thrun et al., 1991) and C4.5 (Quinlan, 1993b), which recognise that some attribute-value pairs might be highly relevant, and some not so. As Murthy et al. (1994) note, the use of such decision functions can significantly decrease tree size, but they may become more opaque in the process.

Other systems use weights to identify the more important and/or costly attributes in a domain (e.g., Núñez, 1988, 1991), allowing an induction algorithm to have a more informed choice, and thus economise on performance system resources. Tan and Schlimmer (1990) reduce cost during learning *and* use. Their algorithm CS-ID3 must be able to ignore features that have not yet been evaluated, and select ones with least cost (as well as greatest information gain). A similar approach is described by Norton (1989). Murthy et al. (1994) select subsets of attributes on which to learn, in

order to ignore irrelevant ones. To a certain extent, the use of frames (e.g., Manago and Kodratoff, 1991) will also achieve this, by recording only the necessary attributes per object. (KATE is a variant of ID3 which uses frames and can thus cope with complex structured objects.) Indeed, irrelevant attributes can seriously hamper learning efficiency and effectiveness. AQ17-HCI is able to remove such attributes from example descriptions, and possibly replace them with others (Wnek and Michalski, 1994).

Safavian and Landgrebe (1991) note that trees can result in overlapping classes and therefore inefficient use of computing resources. That is, if two test nodes contain one or more common classes then the nodes are said to *overlap*, and the same attribute tests may be replicated in separate subtrees (but see Van de Velde, 1989, 1990).

5.4.3 Constructive induction

Constructive induction introduces new terms into the output language. As Muggleton states (1987), the idea is to introduce *meaningful* terms which will then aid the classification process. In Utgoff's terminology (1986), constructive induction constitutes *dynamic* bias (see also Rendell et al., 1987). Ideally, a constructive induction algorithm can overcome inadequacies stipulated by inappropriate bias (i.e., representation language). New terms can lead to more concise concept descriptions but can also lead to a large, complex search space. Examples include Michalski (1980, 1983), DUCE (Muggleton, 1987), AQ15 (Michalski et al., 1986a), CIGOL (Muggleton, 1988; Muggleton and Buntine, 1988), FRINGE and GREEDY3 (Pagallo and Haussler, 1989; Pagallo, 1989), CITRE (Matheus, 1990), variations on AQ17 (Wnek and Michalski, 1994; Thrun et al., 1991; Schlimmer, 1987b and Donoho and Rendell, 1995).

Issues such as *when* to use constructive induction and how to *guide* it are addressed by Watanabe and Elio (1987). Searching for new attributes can be a computationally expensive process. The algorithm LAIR (Watanabe and Elio, 1987), relies on the knowledge-base to constrain construction of new predicates, e.g., if the current hypothesis contains a predicate P, and the knowledge-base contains a rule $P \rightarrow Q$ then replace P with Q. The knowledge-base is partly provided (*a priori*), partly learned and is used as above to deductively constrain induction.

Pagallo and Haussler (1989) and Matheus (1990) note that trees can suffer from the *replication* problem where attribute test sequences are repeated across subtrees. This can lead to excessive fragmentation of the example set and thus the need for more examples to ensure reliable induction. FRINGE (Pagallo, 1989) builds a tree using the primitive attributes, then examines the test sequences near the positive leaves (fringes) and constructs Boolean combinations of these tests to form new attributes. Examples are then redescribed in the new terms, whence the tree is rebuilt and the process reiterates. This iteration halts when no more attributes can be added or a maximum is reached. In CITRE (Matheus, 1990), constructed features are evaluated and 'useless' ones are discarded to constrain search times and so forth. Evaluation uses performance criteria such as accuracy and frequency of use in hypotheses, etc. STAGGER (Schlimmer and Granger, 1986b) adds new attributes to a language which correspond to the concept descriptions attained part way through its incremental learning. Donoho and Rendell (1995) use constructive induction to repair a deficient *theory*.

5.4.4 Incremental induction

ML has been recognised as a method for alleviating the knowledge engineering bottleneck. It seems prudent therefore, that knowledge-bases be *incrementally* updated, not regenerated from scratch as new information becomes available (Schoenauer and Sebag, 1990). Incremental concept learning is the process by which concepts are induced over time, with a possibly incomplete training set which is periodically enlarged as more data becomes available. Langley et al. (1987) are concerned with "plausible" models of learning for humans and propose three dimensions to theories of learning, including *incrementality*. Humans must learn incrementally because of the sequential nature of information, i.e., one rarely has a complete set of facts before reasoning is required (Reinke and Michalski, 1988). Michalski (1986) states:

> "The meaning of human concepts is dynamic; it changes with time and adapts to new contexts and requirements. Novel concepts are continuously being created and developed, and some are being outgrown."

One is also limited by a finite memory, requiring information to be processed as and when it is acquired. Thus learning is *non-monotonic* (Bain and Muggleton 1987; Schoenauer and Sebag, 1990), as descriptions are *revisable*.

> "... knowledge revision is typically much less expensive than knowledge creation" (Utgoff, 1995).

Particular benefit is to be found with large data sets, especially those with many complex and noisy features. An inefficient algorithm can make induction in these domains all but intractable (e.g., see Shen, 1992). Algorithm efficiency is also tackled in Sutton and Whitehead (1993) and Musick et al. (1993) and Webb (1995), among others. For an in-depth investigation of incremental efficiency, see Dutton (1996).

Examples include AQ15 (Michalski et al., 1986a; Kodratoff and Michalski, 1990), AQ11 (Michalski and Chilausky, 1980), GEM (Reinke and Michalski, 1988), LEX (Mitchell et al., 1983), CIGOL (Muggleton, 1988; Muggleton and Buntine, 1988), ID4 (Schlimmer and Fisher, 1986), VBMS (Rendell et al., 1987), COBWEB (Fisher, 1987a, b), CLASSIT (Gennari et al., 1989), ID5(R) (Utgoff, 1988a, 1989a, b), IDL (Van de Velde, 1989, 1990), STAGGER (Schlimmer and Granger, 1986a, b), UNIMEM (Lebowitz, 1987, 1988), PT2 (Utgoff and Brodley, 1990), IB*x* (Aha et al., 1991; Aha, 1992), ITI (Utgoff, 1995) and others (Vrain and Lu, 1988; Sebag and Schoenauer, 1991a, b; Holte, 1989; Schoenauer and Sebag, 1990; Cai et al., 1991; Kubat and Pavlickova, 1991; Conroy and Dutton, 1994, 1995; Dutton 1996).

Advantages offered by an incremental approach include its ability to allow new information to be added to a concept representation without having to restart the induction process. Rebuilding a whole tree when, in all likelihood, a small change is required, is an obviously inferior approach given the often copious, complex and volatile nature of real life data. Incrementality can thus allow an induction algorithm to circumscribe the combinatorial explosion that can be inherent in large, multi-attribute example sets (e.g., Musick et al., 1993). That is, non-incremental algorithms are typically restricted to "one-shot" learning and may attempt to examine the whole example set at once. With incrementality, examples are handled a few at a time instead of in toto (Manago and Kodratoff, 1991; Schoenauer and Sebag, 1990; Sebag and Schoenauer, 1991a; Lebowitz, 1988; Shifu et al., 1992), thus obviating the need for a "window" on the example set (Quinlan, 1979). As training sets grow, being restricted to a subset, or window, for computational reasons is not the ideal answer and has been shown to increase run-times and is of no real benefit (Wirth and Catlett, 1988)[9]. Blythe (1988) suggests the combination of both non-incremental and incremental learning, that is, the initial learning from a large collection of examples followed by the incremental adjustment of the concept definition.

Less concern about the number and size of examples (e.g., Sebag and Schoenauer, 1991a) can relax the need for background knowledge required to constrain the search for an effective representation, thus making the algorithm more generally applicable. Tadepalli (1989) describes *Lazy-EBL*, where an incomplete explanation is formed and then refined incrementally. This results in a reduced learning cost, first of all from the reduced deduction and secondly because partial plans are only corrected if they prove to be inadequate for the performance task.

Incremental learning with perfect memory (Michalski et al., 1986a), means that all examples seen so far are retained. Partial memory systems will retain say, only prototypical examples, exceptions, or even merely concept descriptions. One can usually discard previously seen data if back-tracking is allowed, as there is no need to retain it for a possible "rerun", a distinct advantage if machine space is limited.

Combined with some appropriate form of pruning, incremental learning can track *concept drift* (Schlimmer, 1987a; Schlimmer and Granger, 1986a, b), or *non-stationary concepts* (Sutton and

[9]In fact, with a few restrictions (e.g., only iterate if accuracy increases) and alterations (e.g., pick uniform class distributions), Quinlan (1993b) notes that accuracy may be improved, but time also increases considerably.

Whitehead, 1993; Kubat and Pavlickova, 1991). Utgoff (1989a) suggests that new examples should replace older ones where appropriate (i.e., if a new example has a different class).

Incremental learning can help to induce smaller trees (Utgoff, 1989b) and focus a learning algorithm on misclassified examples, or exceptions, e.g. ADECLU (Decaestecker, 1991), $\widehat{ID5}$ (Utgoff, 1988a). Indeed, ID3 can result in "empty" leaves (if there are more than two classes – Quinlan, 1990c), as all attribute values are assumed to exist in all branches of a tree. This does not occur in incremental settings as branches are added as needed.

Learning from 'near misses' represents an increase in efficiency (e.g., Schlimmer and Fisher, 1986; Utgoff, 1988a; Gemello and Mana, 1989), i.e., the algorithm is forced to focus on areas where further information is needed. Near misses are typically counter-examples that are 'accepted' by the current hypothesis, and help to avoid over-generalisation by ensuring consistency (and *vice versa* for rejected positive examples).

Incrementality can also allow bi-directional searching which is important when poor choices may affect a concept and allows an algorithm to recover from them, e.g., ID4, ID5. As Schoenauer and Sebag (1990) note, incremental algorithms can allow the improved handling of noise (decisions are revisable) but convergence upon a solution may not be possible due to the oscillatory application of operators (e.g., ID4 and Decaestecker, 1991). Bi-directional searching allows quasi-backtracking without its inherent overheads (Langley et al., 1987). The ability to backtrack can thus render an algorithm less susceptible to the order of presentation of examples and to the effects of noise (Lebowitz, 1988; Gennari et al., 1989; Decaestecker, 1991; etc.).

5.4.5 Incremental TDIDT algorithms

ID3 (Quinlan, 1979) is a well known tree induction algorithm which has been extended in a number of ways (e.g., Quinlan, 1993b). Two of these extensions, viz., ID4 (Schlimmer and Fisher, 1986) and ID5 (Utgoff, 1988a), are said to be incremental in that examples can be added piecemeal to an existing tree. In this section, we explore in more detail these, and other algorithms.

5.4.5.1 ID4. ID3 can be made to process incrementally simply by restarting the whole tree generation process from scratch whenever new examples arrive. Obviously, this is a worst-case scenario, and significant improvement in tree update time is desirable. This worst-case algorithm is termed "INCREMENTAL-ID3". In fact, Gennari et al. (1989) deny that such extensive reprocessing of training data can be labelled as incremental. A truly incremental algorithm, ID4, is as follows. Consider a tree built by ID3 where each test node has a number of potential test attributes, i.e., the algorithm is faced with a choice for the test attribute. In ID4, for each potential test attribute at a node, a count is maintained for each attribute value, in terms of the example classes. In other words, if one assumes two classes (positive and negative), then there are two entries for each attribute value per decision node. Each entry is a count of the number of examples seen so far that have the particular attribute value in mind. Mingers (1989) terms these *contingency tables*. Subtrees are grown as in ID3, but with an added statistical *significance* test (χ^2). This is a form of pre-pruning which will only allow a "split" if the implied relationship is not thought to have arisen by chance (see also Quinlan, 1986a). Each time a new example is added to the tree, it starts at the root and follows a path according to its value for the current test attribute. At each node, the entropy calculations which first picked the current, or *installed* (Utgoff, 1995) test attribute are recalculated. Thus, one can immediately determine whether the current test attribute is still the "best" choice. If it is, the tree is left unchanged and the example propagation continues to the next level in the tree path. Otherwise, the current test attribute is deemed to be suboptimal (with respect to entropy), i.e., another attribute would give greater information gain at that point in the tree. The subtree is discarded and replaced by the new test attribute. The result is to replace several tests with one. To rebuild a discarded subtree to its previous detail, further training instances which will be propagated to this new leaf are required.

In a 'well behaved' environment (cf. the parity problem), the tree stabilises from the top-down, level by level until convergence upon an ideally (quasi-) optimal tree occurs. Schlimmer and Fisher

(1986) analyse the efficacy of their algorithm in a number of ways. In a tree with root T_0, level T_i must stabilise before T_{i+1} can and this leads to a worst case estimate of using all E training examples, *per level*, to construct a given tree (ID3 would require E in total).

Schlimmer and Fisher also make use of *smarter* versions of ID4 and incremental ID3, namely $\widehat{ID4}$ and $\widehat{ID3}$. These change a tree only when a *misclassification* occurs, i.e., a training example arrives at a leaf of a different class. This utilises "near-miss" examples assumed to be close to the (geometric) concept boundary in concept-space[10].

Utgoff notes (1989a) that ID4 is unable to learn some concepts (in the context of unlimited examples and a desired accuracy). This occurs when ID4 cannot decide in which order test attributes should be arranged and repeatedly discards subtrees, thus preventing the tree from stabilising.

5.4.5.2 ID5. Usually, not all of a discarded subtree needs to be changed, and so ID4 is at a disadvantage. In other words, if the tree were rebuilt from scratch then one could find that most of the tests in the purportedly suboptimal subtree may well remain as they were. ID5 (Utgoff, 1988a) rectifies this by not discarding subtrees but *restructuring* them. The idea is to identify the best test attribute as before and then attempt to *pull* this attribute up from a lower position in the tree to become the new subtree root. Again, ID5 achieves incrementality by maintaining statistics at each decision tree node that summarise class distributions, in terms of attribute values. As with ID4, ID5 can then judge whether a subtree has become suboptimal with respect to entropy and then restructure to keep the most informative attributes nearer the root of the tree.

ID5 is a *perfect memory* algorithm in that all examples are retained. This is achieved largely implicitly through the use of the tree structure. Therefore, at a given leaf, only the attributes not tested so far are required to be stored, the rest are stipulated by the path back to the root.

As an example is added to a node, the usual counts are incremented (termed *instance count additions* or ICAs, by Utgoff), and the potential test attributes are rechecked to see which is best. If no revision is needed, the algorithm progresses to the next level, until a leaf is reached. If there is only one class, the portion of the example not implicit in the tree is stored. If the leaf is impure, the tree is grown as in ID3. If at some stage a revision is needed, a subtree is first expanded to include the new 'ideal' test attribute in all paths. Examples are expanded to include the implicit parts. The new best attribute is then recursively pulled to the root of each subtree of the old test attribute, whereupon it is swapped with its parent, that is, the suboptimal test attribute is pushed down one level. Duplicated subtree roots are then merged and subtrees of one class are contracted to a leaf. Thus ID5 has the operators "split" (tree growth), "prune" (remove redundant nodes from the merging process) and "transpose" (restructure) (Van de Velde, 1990).

In the first version (Utgoff, 1988a), the process would now stop, possibly leaving suboptimal subtrees below the "swapped" test. That is, the subtree paths are not rechecked to see if there are more restructures which should be performed. In a subsequent version, i.e., ID5R (Utgoff, 1989a), the subtree below the new test attribute is recursively rechecked to ensure the most favourable sequence of tests is in place. This ensures ID3-equivalent trees and thus the most informative attributes occupy the highest positions in the subtree.

Again, versions which update only on misclassifications in order to save on computation were also used, i.e., $\widehat{ID5}$ and $\widehat{ID5R}$. It is noted (Utgoff, 1989a) that all incremental algorithms built trees smaller than ID3 (for the multiplexor domain), although more examples were used.

5.4.5.3 IDL. ID3 is biased towards the induction of small trees, and the design of IDL (Van de Velde, 1989, 1990) reiterates this. IDL is based on ID5(R), and is thus incremental. An additional heuristic enables IDL to find trees which are often optimal in size[11], in many domains, requiring less computation and fewer training examples. This search for a *topologically minimal* tree stems from similar motivations as tree *pruning*, i.e., redundant tests. IDL searches from the leaves up, using a

[10]And thus more informative for specifying the said concept boundary.
[11]i.e. contain fewest nodes.

heuristic based on a tree's structure, not on a statistical method, and aims to avoid the obsolete repetition of tests within trees (see also Pagallo, 1989).

Van de Velde claims (1990) that ID4 and ID5(R) often find suboptimal trees (with respect to size)[12]. An example is *covered* by a tree path if all tests on the path are satisfied by the example, including the class at the leaf. Such a path is unique for the example, but there are other *partial* paths which may cover part of the example. These are found in a bottom-up fashion by starting at leaves of the *same class* as the example in question. One then climbs successively upwards if the parent decision node test is satisfied by the example. The *topological relevance* (TR) for an attribute A is then defined as the number of times A appears in the partial and full paths covered by the example in question. TR gives an indication of the importance of A for classification. Thus attributes with high TR values would be preferable in higher positions in a tree and may render other tests obsolete. That is, highly discriminatory tests often result in smaller trees. The tree manipulation procedures used in ID5(R) are now used to push preferred attributes further up the tree and consequently minimise TR scores. IDL starts in the same way as ID5, etc. with an empty tree which is grown incrementally using measures such as entropy. IDL uses training examples to guide its search for less efficient tests, which are then pushed down the tree and possibly pruned away. Using the current example keeps this search relevant. Van de Velde conjectures that in most cases IDL will find a topologically minimal tree and will then adhere to it, unlike ID4/5. If no such tree exists, IDL will not converge to a unique tree, and thus *limit-cycles* through equivalent, suboptimal trees (i.e., of the same size). As noted above, IDL sometimes converges to non-topologically-minimal trees (Elomaa and Kivinen, 1990), although this may be dependent upon example ordering. Elomaa and Kivinen conjecture that the failure to converge to a topologically minimal tree may lead to a failure to converge to *any* tree if input is repeated indefinitely.

IDL was also found to reduce run-time, entropy calculations, tree growths, pruning, and especially tree restructures over ID5R. Van de Velde notes that IDL may well be sensitive to noise due to its reliance on single examples.

5.4.5.4 ITI. Utgoff (1995) describes two algorithms for efficient restructuring of decision trees, one of which (ITI) is incremental, and is descended from ID5R. The other (DMTI), aims to use measures of tree quality more explicitly. Utgoff utilises binary trees, which also list attributes, values and their classes, in binary *search* trees at each decision node, to aid efficient update. ITI implements multi-phase example addition, as elsewhere (Conroy and Dutton, 1994, 1995). Utgoff notes that such a policy allows a number of "training modes". A "lazy" mode allows one to add several examples and only update counts (etc.) once, at the end. This in itself could save much processing, i.e., one only rechecks the tree for revisions when an up-to-date tree is required for some purpose. Use is made of the Minimum Description Length Principle to prune subtrees which are insufficiently compact and/or accurate. However, any pruned subtrees are not discarded, but simply marked as pruned. This "virtual pruning" preserves information for further updates, but provides a (hopefully) less over-fitted subtree for use in classification.

Use is also made of a "stale marker" at each decision node (similar to Conroy and Dutton 1994, 1995). When an example is added to a path, each node is marked as stale. This marker is only removed if the recalculation of the impurity function indicates that a tree restructure is required. Utgoff also describes an efficient method for processing for continuous attributes. Each value is listed in order at a node, tagged by the class of the example from whence it came. Then each pair of adjacent values of a different class gives rise to a possible cut-point, midway between them.

DMTI is a non-incremental tree induction algorithm again described in Utgoff (1995). This algorithm builds a tree using a typical impurity function (gain ratio) and then evaluates the whole tree, possibly revising it, based on a global "direct metric", or quality measure. This allows one to alter the bias required for a tree, i.e., one simply alters the direct metric.

[12]Although it has been noted that so too can IDL (Elomaa and Kivinen, 1990).

5.4.5.5 Other incremental algorithms. Cockett and Zhu (1989) use a similar approach to ID5 and IDL to reduce trees in algebraic form to a minimum size. The algorithm attempts to push non-essential attributes down the tree and removes them from the leaves using *syntactic* relations (e.g., $q(x, x) = x$, says that a node q with identical children can be reduced to a single leaf). Cockett and Zhu note however, that such an *irreducible* tree is not guaranteed to be *simple* as some redundancy can still remain.

Crawford (1989) introduces some extensions to the CART algorithm (Breiman et al. 1984), enabling incremental processing of *binary* trees. A new example is added to the tree and the new impurity is calculated at each node. The optimum split may now have changed, which may cause the example to be sent down the *alternate* branch, i.e. not the one it would be sent down with the current split. So, CART computes the decrease in impurity for both cases, the current split s where the example behaves as it should, and a hypothetical split s', where the example takes the alternate branch. If the impurity decrease for s (with the extra example) is less than that for s' (plus example), CART searches for a new test attribute. If the new attribute is not s, the subtree is pruned and replaced with the new attribute, whereupon CART *rebuilds* both child subtrees.

Crawford's experiments revealed a large number of tree restructures (many unnecessary), at least partially caused by attributes being dependent upon one another and by noise, with new trees often being virtually the same as the old. In light of this, Crawford amends the procedure with extra statistical calculations to determine whether a new suggested split is actually worthwhile, with improved performance.

Bhandaru and Murty (1991) use a general tree-like structure called an *HC-Expression*, where successive levels are positive and negative exceptions. To classify an example, one starts at the root and finds a lower node which "covers" the example, then one checks to see if the example is included in the exceptions one level below. The example's class is then the class of the last covering node. That is, each node is described by a "cover", as in the AQ algorithm. The *quality* of nodes is judged by an accuracy-like measure ($\frac{e_i - e_{i+1}}{e_i}$), where e_i is the number of examples satisfying node i, and e_{i+1} is the number satisfying the node below). If node quality is poor, then a node is split (into value subsets). The algorithm also updates on exceptions only, and the authors hold that the structure is easy to change, and so is useful for incremental learning.

5.5 Theory revision

Theory revision is an area perhaps more commonly associated with EBL and/or ILP, as both have explicitly represented background knowledge or domain theory. However, according to Donoho and Rendell (1995), theory revision "...integrates inductive learning and background knowledge by combining training examples with a coarse domain theory to produce a more accurate theory". Donoho and Rendell aim to change a theory's *structure* and/or its *representation* if either prove "restrictive". That is, a representation may be inappropriate for a new theory and may need to be changed (e.g., rules to neural nets: Towell and Shavlik: 1994), or a theory's structure may need more than "local" changes.

5.6 Discussion

Bareiss et al. (1990) criticise algorithms' sole reliance on *higher-level* generalisations of examples. They state that the varied uses of concept descriptions necessitates more flexible representations which reflect inherent 'vagueness' (i.e., polymorphism in natural domains). That is, the use of inappropriate knowledge representations, into which a description has been 'compiled' may adversely affect the efficient use of this knowledge. Hence they propound exemplars and "lazy generalisation" which does less abstraction on fewer attributes, if examples are 'sufficiently' similar, and claim that this results in more generally applicable descriptions.

Quinlan (1993a) notes that IBL algorithms can lead to relatively complex and inaccurate

concepts if irrelevant attributes exist (as compared to algorithms which determine explicit generalisations). Indeed, Quinlan combines IBL and "model-based" methods (e.g., decision trees), and uses such models to adjust the k prototypes selected for k-nearest neighbour matching (i.e., adjust the prediction made by a prototype).

The systematic comparison of many algorithms is presented in Thrun et al. (1991) for the "Monks Problems" which one should consult for further detail, and Wnek and Michalski (1994), and others (e.g., Gams and Lavrac, 1987; Clark and Niblett, 1989; Kocabas, 1991; Kalkanis and Conroy, 1991). Rendell (1986, 1987a) describes and reviews logic, trees, and exemplars.

6 Conclusion

This paper has reviewed the field of ML. It extends and complements earlier work and thus omits certain details which have been covered previously. Nevertheless, additional and subsequent references are noted and/or discussed, and a substantial bibliography is given. Emphasis is given to inductive ML, and indeed, to symbolic, supervised induction. Furthermore, extra detail is presented on incremental decision tree algorithms which have yet to be substantially reviewed. Such algorithms may be *required* for practical induction, both for automatic knowledge acquisition and data mining, as the proliferation of data continues apace. Efficient (with respect to run-time and storage), robust and accurate algorithms, which are mindful of concept intelligibility, are necessary for further advancement of this paradigm in mainstream computing.

References

Aha, D, Kibler, D and Albert, M, 1991. "Instance-based learning algorithms". *Machine Learning Journal*, **6**(1) 37–66.

Aha, D, 1992. "Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms". *International Journal of Man-Machine Studies* **36**(2) 267–287.

Angluin, D and Smith, C, 1983. "Inductive inference: theory and methods". *ACM Computing Surveys* **15**(3) 237–270.

Angluin, D and Laird, P, 1988. "Learning from noisy examples". *Machine Learning Journal* **2** 343–370.

Bain, M and Muggleton, S, 1987. "Non-monotonic learning". In: Hayes, J., Michie, D. and Tyugu, E. eds., *Machine Intelligence 12: Towards An Automated Logic Of Human Thought*, 105–119.

Bareiss, E, Porter, B and Weir, C, 1990. "PROTOS: an exemplar-based learning apprentice". In: Kodratoff, Y and Michalski, R, eds., *Machine Learning: An AI Approach*, vol. 3. Morgan-Kaufmann.

Ben-David, A, 1995. "Monotonicity maintenance in information theoretic machine learning algorithms". *Machine Learning Journal* **19** 29–43.

Bhandaru, M and Murty, M, 1991. "Incremental learning from examples using HC-expressions". *Pattern Recognition* **24**(4) 273–282.

Blum, A, 1992. *Neural Networks in C++ – An Object-Oriented Framework for Building Connectionist Systems*. Wiley.

Blythe, J, 1988. "Constraining search in a hierarchical discriminative learning system". *Proceedings ECAI88, European Conference on AI* 378–383.

Bohanec, M and Bratko, I, 1994. "Trading accuracy for simplicity in decision trees". *Machine Learning Journal* **15**(3) 223–250.

Breiman, L, Friedman, J, Olshen, R and Stone, C, 1984. *Classification and Regression Trees*. Wadsworth International Group.

Buntine, W, 1990. "Myths and legends in learning classification rules". *Proceedings 8th National Conference on AI, vol. 2*, 736–742, AAAI/MIT Press.

Cai, Y, Cercone, N and Han, J, 1991. "Attribute oriented induction in relational databases". In: Piatetsky-Shapiro, G and Frawley, W, eds., *Knowledge Discovery in Databases*, 213–228. AAAI/MIT Press.

Cameron-Jones, R and Quinlan, J, 1994. "Efficient top-down induction of logic programs". *SIGART Bulletin* **5**(1) 33–42.

Carbonell, J, Michalski, R and Mitchell, T, 1983. "An overview of machine learning", In: Michalski, R, Carbonell, J and Mitchell, T, eds., *Machine Learning: An AI Approach*, Morgan-Kaufmann.

Carbonell, J, 1989. "Introduction: paradigms for machine learning". *Artificial Intelligence* **40**(1–3) 1–9.

Cendrowska, J, 1987. "PRISM: an algorithm for inducing modular rules". *International Journal for Man-Machine Studies*, **27**(4) 349–370.

Cestnik, B, Kononenko, I and Bratko, I, 1987. "ASSISTANT 86: a knowledge elicitation tool for sophisticated users". In: Bratko, I and Lavrac, N, eds., *Progress in Machine Learning – Proceedings of the 2nd European Working Session on Learning* 31–45. Sigma Press.

Cheng, J, Fayyad, U, Irani, K and Qian, Z, 1988. "Improved decision trees: a generalised version of ID3". In: Laird, J, ed., *Proceedings 5th International Conference on Machine Learning*. Morgan-Kaufmann.

Cichosz, P, 1995. "Truncating temporal differences: on the efficient implementation of TD(λ) for reinforcement learning". *Journal of AI Research* **2** 287–318.

Cios, K and Liu, N, 1992. "A machine learning method for generation of a neural network architecture: a continuous ID3 algorithm". *IEEE Transactions on Neural Networks* **3**(2) 280–290.

Clark, P and Niblett, T, 1987. "Induction in noisy domains". In: Bratko, I and Lavrac, N, eds., *Progress in Machine Learning – Proceedings of the 2nd European Working Session on Learning* 11–30. Sigma Press.

Clark, P and Niblett, T, 1989. "The CN2 induction algorithm". *Machine Learning Journal* **3** 261–283.

Cockett, J and Zhu, Y, 1989. "A new incremental learning technique for decision trees with thresholds". *Applications Of Artificial Intelligence 7*, SPIE, International Society For Optical Engineering, vol. 1095, no. 2.

Conroy, G and Dutton, D, 1994. "JITTER: a Lazy machine's guide to induction". *Technical Report, UMIST-COM-AI-94-4*, Department of Computation, UMIST, PO BOX 88, Manchester M60 1QD, UK.

Conroy, G and Dutton, D, 1995. "The effects of noise on efficient incremental induction (extended abstract). In: Lavrac, N and Wrobel, S, eds., *Machine Learning: ECML-95, Proceedings 8th European Conference on Machine Learning* 275–278. Lecture Notes in AI Series, Springer Verlag.

Crawford, S, 1989. "Extensions to the CART algorithm". *International Journal of Man-Machine Studies* **31** 197–217.

David, J, Krivine, J and Simmons, R, eds., 1993. *Second Generation Expert Systems* Springer-Verlag.

Dayan, P and Sejnowski, T, 1994. "TD(λ) converges with probability 1". *Machine Learning Journal* **14** 295–301.

Decaestecker, C, 1989. "Incremental concept formation via a suitability criterion". *Proceedings Conference on Data Analysis, Learning Symbolic and Numeric Knowledge*. 435–442.

Decaestecker, C, 1991. "Incremental classification: a multidisciplinary viewpoint". *Proceedings Conference on Symbolic-Numeric Data Analysis and Learning*, Diday, E and Lechevallier, Y, eds., Nova Science, 283–295.

Dietterich, T and Michalski, R, 1983. "A comparative review of selected methods for learning from examples". In: *Machine Learning: An AI Approach*, Michalski, R, Carbonell, J and Mitchell, T, eds., vol. 1, Morgan-Kaufmann, 41–81.

Dietterich, T, 1986. "Learning at the knowledge level". *Machine Learning Journal* **1** 287–316.

Donald, JH, 1994. "Rule induction – Machine learning techniques". *Computing and Control Engineering Journal* **5**(5) October, 249–255.

Donoho, P and Rendell, L, 1995. "Representing and restructuring domain theories: a constructive induction approach". *Journal of AI Research* **2** 411–446.

Duda, R and Hart, P, 1973. *Pattern Classification and Scene Analysis*. Wiley.

Dutton, D, 1996. "An investigation into the efficiency of incremental decision tree induction and alternative sources of bias. PhD thesis, Department of Computation, UMIST, PO Box 88, Manchester M60 1QD, UK.

Elomaa, T and Kivinen, J, 1990. "On inducing topologically minimal decision trees". *Proceedings 2nd International IEEE Conference on Tools for AI*, 746–752.

Fatti, L, Hawkins, D and Liefde Raath, E, 1982. "Discriminant Analysis". In: *Topics in Applied Multivariate Analysis*, Hawkins, D, ed., Cambridge University Press, pp. 1–71.

Fayyad, U and Irani, K, 1990. "What should be minimised in a decision tree". *Proceedings 8th National Conference on AI*, vol 2, AAAI/MIT Press.

Fayyad, U and Irani, K, 1992. "The attribute selection problem in decision tree generation". *Proceedings 10th International Conference on AI*, vol 1, AAAI/MIT Press, 104–10.

Fayyad, U and Uthurusamy, R, 1995. *Proceedings 1st International Conference on Knowledge Discovery and Data Mining*. AAAI Press.

Fayyad, U, Piatetsky-Shapiro, G, Smyth, P and Uthurusamy, R, 1996. *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press.

Fikes, R and Nilsson, N, 1971. "STRIPS: a new approach to the application of theorem proving to problem solving". *Artificial Intelligence* **2** 189–208.

Fisher, DH, 1987a. "Conceptual clustering, learning from examples and inference". *Proceedings 4th International Workshop on Machine Learning*, Morgan-Kaufmann.

Fisher, DH, 1987b. "Knowledge acquisition via incremental conceptual clustering". *Machine Learning Journal* **2** 139–172.

Fisher, D and McKusick, K, 1989. "An empirical comparison of ID3 and back-propagation". *Proceedings 11th International Joint Conference on AI*, Morgan-Kaufmann.

Gams, M and Lavrac, N, 1987. "Review of five empirical learning systems within a proposed schemata".

Progress in Machine Learning – Proceedings of EWSL87: 2nd European Working Session on Learning, Bratko, I and Lavrac, N (eds.), Sigma Press, 46–66.

Gelfand, S and Delp, E, 1991. "On tree structured classifiers". *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections*, Sethi, I and Jain, A (eds.), Elsevier, 51–70.

Gemello, R and Mana, F, 1989. "An integrated characterisation and discrimination scheme to improve learning efficiency in large data sets". *Proceedings 11th International Joint Conference on AI*, Sridharan, N ed., vol 1, Morgan-Kaufmann.

Gennari, J, Langley, P and Fisher, D, 1989. "Models of incremental concept formation". *Artificial Intelligence* **40**(1–3).

Goldberg, D, 1989. *Genetic Algorithms in Search, Optimisation, and Machine Learning*. Addison-Wesley.

Haussler, D, 1986. "Quantifying the inductive bias in concept learning". *Proceedings 5th National Conference on AI* 485–489, Morgan-Kaufmann.

Hawkins, D and Kass, G, 1982. "Automatic interaction detection". In: *Topics in Applied Multivariate Analysis*. Hawkins, D (ed.), 269–302. Cambridge University Press.

Hinton, G, 1990. "Preface to the special issue on connectionist Symbol Processing, *Artificial Intelligence*, **46**(1–4) 1–4.

Hoel, P, 1984. *Introduction to Mathematical Statistics*. Wiley.

Holsheimer, A and Siebes, A, 1994. "Data Mining – The Search For Knowledge in Databases". *Technical Report CS-R9406*, CWI, PO BOX 94079, 1090 GB Amsterdam, Netherlands.

Holte, R, 1989. "Alternative information structures in incremental learning systems, In: *Machine and Human Learning – Advances in European Research*, Kodratoff, Y and Hutchinson, A (eds.), 121–142. Kogan Page.

Hyafil, L and Rivest, R, 1976. "Constructing optimal binary decision trees is NP-complete". *Information Processing Letters* **5**(1) 15–17.

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

Kalkanis, G and Conroy, G, 1991. "Principles of induction and approaches to attribute based induction". *The Knowledge Engineering Review* **6**(4) 307–333.

Kearns, MJ, 1990. *The Computational Complexity of Machine Learning*. MIT Press.

Keller, R, 1987. "Concept learning in context". In: *Proceedings 4th International Workshop on Machine Learning*, Langley, P (ed.). Morgan-Kaufmann.

Kibler, D and Langley, P, 1988. "Machine learning as an experimental science". In: *EWSL88 – Proceedings of the 3rd European Working Session on Learning*, Sleeman, D (ed.), 81–92. Pitman.

Kibler, D and Aha, D, 1990. "Learning representative exemplars of concepts: an initial case study". In: *Readings in Machine Learning* Shavlik, J and Dietterich, T (eds.). (Originally *5th Int. Workshop on Machine Learning*.) Morgan-Kaufmann.

Kocabas, S, 1991. "A review of learning". *The Knowledge Engineering Review* **6**(3) 195–222.

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

Kodratoff, Y and Michalski, R (eds.), 1990. *Machine Learning: An AI Approach*. Vol. 3. Morgan-Kaufmann.

Kolodner, J, 1993. *Case-Based Reasoning*. Kluwer.

Kohavi, R, John, G, Long, R, Manley, D and Pfleger, K, 1994. "MLC++: A machine learning library in C++". *Tools with Artificial Intelligence* 740–743. IEEE Press.

Kononenko, I and Bratko, I, 1991. "Information-based evaluation criterion for classifier's performance". *Machine Learning Journal* **6**(1) 67–80.

Kubat, M and Pavlickova, J, 1991. "System FLORA: learning from time-varying training sets. In: "*Proceedings EWSL91 – European Working Session on Learning* Kodratoff, Y (ed.). Springer-Verlag.

Laird, J, Rosenbloom, P and Newell, A, 1984. "Towards chunking as a general learning mechanism". *Proceedings National Conference on AI* 188–192. Morgan-Kaufmann.

Langley, P, Gennari, J and Iba, W, 1987. "Hill-climbing theories of learning". In: *Proceedings 4th International Workshop on Machine Learning* Langley, P (ed.), 312–323, Morgan-Kaufmann.

Lebowitz, M, 1987. "Experiments with incremental concept formation: UNIMEM". *Machine Learning Journal* **2**(2) 103–138.

Lebowitz, M, 1988. "Deferred commitment in UNIMEM: waiting to learn". In: *Proceedings 5th International Conference on Machine Learning* Laird, J (ed.), 80–86, Morgan-Kaufmann.

Lenat, D, 1982. "The nature of heuristics". *Artificial Intelligence* **19**(2) 189–249.

Lenat, D, 1983. "The role of heuristics in learning by discovery: three case studies". In: *Machine Learning: An AI Approach*, Michalski, R, Carbonell, J and Mitchell, T (eds.), vol 1, 243–306. Morgan-Kaufmann.

Lopez de Mantaras, R, 1991. "A distance-based attribute selection measure for decision tree induction". *Machine Learning Journal* **6**(1).

Manago, M and Kodratoff, Y, 1991. "Induction of decision trees from complex structured data". In: *Knowledge Discovery in Databases* Piatetsky-Shapiro, G and Frawley, W (eds.), 289–306. AAAI/MIT Press.

Marr, D, 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman.

Matheus, C, 1990. "Feature construction: an analytic framework and an application to decision trees, *PhD Thesis*, University of Illinois at Urbana-Champaign.

Michalski, RS and Chilausky, 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 soybean disease diagnosis". *International Journal of Policy Analysis and Information Systems* **4**(2) 125–161.

Michalski, RS, 1980. "Pattern recognition as rule-guided inductive inference". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2**(4) 349–361.

Michalski, RS and Stepp, RE, 1983. "Learning from observation: conceptual clustering". In: *Machine Learning: An AI Approach*, Michalski, R., Carbonell, J and Mitchell, T (eds.), vol 1. Morgan-Kaufmann.

Michalski, R, Carbonell, J and Mitchell, T (eds.), 1983. *Machine Learning: An AI Approach* vol 1. Morgan-Kaufmann.

Michalski, RS, 1983. "A theory and methodology of inductive learning". In: *Machine Learning: An AI Approach* Michalski, R, Carbonell, J and Mitchell, T (eds.), vol 1. Morgan-Kaufmann.

Michalski, RS, Mozetic, I, Hong, J and Lavrac, N, 1986a. "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains". *Proceedings 5th National Conference on AI* 1041–1045. Morgan-Kaufmann.

Michalski, R, Carbonell, J and Mitchell, T (eds.), 1986b. *Machine Learning: An AI Approach* vol 2. Morgan-Kaufmann.

Michalski, R, 1986. "Understanding the nature of learning". In: *Machine Learning: An AI Approach* Michalski, R, Carbonell, J and Mitchell, T (eds.), vol 2. Morgan-Kaufmann.

Michalski, R, 1990. "Learning flexible concepts". In: *Machine Learning: An AI Approach* vol 3. Kodratoff, Y and Michalski, R (eds.). Morgan-Kaufmann.

Michalski, R and Tecuci, G, 1994. *Machine Learning: A Multistrategy Approach* vol 4. Morgan-Kaufmann.

Michalski, R, 1994. "Inferential theory of learning". In: *Machine Learning: A Multistrategy Approach* Michalski, R and Tecuci, G (eds.), vol 4. Morgan-Kaufmann.

Mingers, J, 1989. "An empirical comparison of selection measures for decision tree induction". *Machine Learning Journal* **3** 319–342.

Minton, S, 1988. *Learning Search Control Knowledge – An Explanation Based Approach*. Kluwer Academic.

Mitchell, T, 1977. "Version spaces: a candidate elimination approach to rule learning". *Proceedings 5th International Joint Conference on AI* 305–310.

Mitchell, T, 1982. "Generalisation as search". *Artificial Intelligence* **18** 203–226.

Mitchell, T, Utgoff, P and Banerji, R, 1983. "Learning by experimentation: acquiring and refining problem-solving heuristics". In: *Machine Learning: An AI Approach* Michalski, R, Carbonell, J and Mitchell, T (eds.), vol 1. Morgan-Kaufmann.

Mitchell, T, Keller, R and Kedar-Cabelli, S, 1990. "Explanation-based generalisation: a unifying view". In: *Readings in Machine Learning* Shavlik, J and Dietterich, T (eds.). Morgan-Kaufmann.

Muggleton, S. 1987. "Structuring knowledge by asking questions". In: *Progress in Machine Learning* Bratko, I and Lavrac, N (eds.), 218–229. Sigma Press.

Muggleton, S, 1988. "A strategy for constructing new predicates in first order logic". In: *Proceedings EWSL88 – 3rd European Working Session on Learning* Sleeman, D (ed.). Pitman.

Muggleton, S and Buntine, W, 1988. "Machine invention of first-order predicates by inverting resolution". *Proceedings 5th International Conference on Machine Learning* Ann Arbor, MI, 339–352. Morgan-Kaufmann.

Muggleton, S, Srinivasan, A and Bain, M, 1992. "Compression, significance and accuracy". *Proceedings 9th International Workshop on Machine Learning* 338–347.

Muggleton, S, 1994. "Inductive logic programming: derivations, successes and shortcomings". *SIGART Bulletin* **5**(1) 5–11.

Murphy, P and Pazzani, M, 1994. "Exploring the decision forest: an empirical investigation of Occam's razor in decision tree induction". *Journal of AI Research* **1** 257–275.

Murthy, S, Kasif, S and Salzberg, S, 1994. "A system for induction of oblique decision trees". *Journal of AI Research* **2** 1–32.

Murthy, S and Salzberg, S, 1995. "Lookahead and pathology in decision tree induction". Unpublished manuscript.

Musick, R, Catlett, J and Russel, S, 1993. "Decision theoretic subsampling for induction on large databases". *Proceedings 10th International Conference on Machine Learning*. Morgan-Kaufmann.

Natarajan, BK, 1991. *Machine Learning: A Theoretical Approach* Morgan-Kaufmann.

Niblett, T, 1987. "Constructing decision trees in noisy domains". In: *Progress in Machine Learning – Proceedings of EWSL87: 2nd European Working Session on Learning* Bratko, I and Lavrac, N (eds.), 67–78. Sigma Press.

Norton, S, 1989. "Generating better decision trees". *Proceedings 11th International Joint Conference on AI* vol 1, 800–805, Morgan-Kaufmann.
Núñez, M, 1988. "Economic induction: A case study". In: *Proceedings EWSL88 – 3rd European Working Session on Learning* Sleeman, D (ed.). Pitman.
Núñez, M, 1991. "The use of background knowledge in decision tree induction". *Machine Learning Journal* **6**(3) 231–250.
Pagallo, G and Haussler, D, 1989. "Two algorithms that learn DNF by discovering relevant features". In: *6th International Workshop on Machine Learning* Segre, A (ed.), 119–123.
Pagallo, G, 1989. "Learning DNF by decision trees, In: *Proceedings 11th International Joint Conference on AI* Sridharan, N (ed.), vol 1. Morgan-Kaufmann.
Piatetsky-Shapiro, G and Frawley, W, 1991. *Knowledge Discovery in Databases*. AAAI Press.
Quinlan, J, 1979. "Discovering rules by induction from large collections of examples". In: *Expert Systems in the Micro-Electronic Age*, Michie, D (ed.). Edinburgh University Press.
Quinlan, J, 1983. "Learning efficient classification procedures and their application to chess end games". In: *Machine Learning: An Artificial Intelligence Approach* Michalski, R, Carbonell, J and Mitchell, T (eds.), vol 1, 463–482. Morgan-Kaufmann.
Quinlan, J, 1986a. "Induction of decision trees. *Machine Learning Journal* **1**.
Quinlan, J, 1986b. "The effect of noise on concept learning". In: *Machine Learning: An AI Approach* vol 2, Michalski, R, Carbonell, J and Mitchell, T (eds.). Morgan-Kaufmann.
Quinlan, J, 1987a. "Decision trees as probabilistic classifiers". *Proceedings of the 4th International Workshop on Machine Learning*. Morgan-Kaufmann.
Quinlan, J, 1987b. "Simplifying decision trees". *International Journal of Man-Machine Studies* **27** 221–234.
Quinlan, J, 1987c. "Generating production rules from decision trees". In: *Proceedings 10th International Joint Conference on AI* McDermott, J (ed.), 304–307, Morgan-Kaufmann.
Quinlan, J, 1988a. "Decision trees and multi-valued attributes". In: *Machine Intelligence 11: Logic and the Acquisition of Knowledge* Hayes, J, Michie, D and Richards, J (eds.), 305–318. Clarendon Press.
Quinlan, J, 1988b. "An empirical comparison of genetic and decision-tree classifiers". *Proceedings 5th International Conference on Machine Learning* 135–141. Morgan-Kaufmann.
Quinlan, J and Rivest, R, 1989. "Inferring decision trees using the minimum description length principle". *Information and Computation* **80** 227–248.
Quinlan, J, 1990a. "Learning logical definitions from relations". *Machine Learning Journal* **15**(3) 239–266.
Quinlan, J, 1990b. "Probabilistic decision trees". In: *Machine Learning: An AI Approach* vol 3, Kodratoff, Y and Michalski, R (eds.). Morgan-Kaufmann.
Quinlan, J, 1990c. "Decision trees and decision making". *IEEE Transactions on Systems, Man and Cybernetics* **20**(2) 339–346.
Quinlan, J, 1993a. "Combining instance-based and model-based learning". *Proceedings 10th International Conference on Machine Learning* 236–243. Morgan-Kaufmann.
Quinlan, J, 1993b. *C4.5: Programs for Machine Learning*. Morgan-Kaufmann.
Reinke, R and Michalski, R, 1988. "Incremental learning of concept descriptions: a method and experimental results". In: *Machine Intelligence 11: Logic and the Acquisition of Knowledge* Hayes, J, Michie, D and Richards, J (eds.), 263–288. Clarendon Press.
Rendell, L, 1986. "A general framework for induction and a study of selective induction". *Machine Learning Journal* **1** 177–226.
Rendell, L, 1987a. "Similarity-based learning and its extensions". *Computational Intelligence* **3**(4) 241–266.
Rendell, L, 1987b. "Representations and models for concept learning". *Technical Report*, UIUCDCS-R-87-1324, University of Illinois.
Rendell, L, Seshu, R and Tcheng, D, 1987. "More robust concept learning using dynamically variable bias". In: *Proceedings 4th International Workshop on Machine Learning*. Langley, P (ed.), 66–78. Morgan-Kaufmann.
Ringland, G and Duce, D, 1988. *Approaches to Knowledge Representation – An Introduction*. Wiley.
Rivest, R, 1987. "Learning decision lists". *Machine Learning Journal* **2** 229–246.
Rumelhart, D, McClelland, J and the PDP Research Group, 1986. *Parallel Distributed Processing – Explorations in the Microstructure of Cognition – vol 1: Foundations*. MIT Press.
Rymon, R, 1993. "An SE-tree based characterisation of the induction problem". *Proceedings Machine Learning Conference* Amherst, MA.
Safavian, S and Landgrebe, D, 1991. "A survey of decision tree classifier methodology". *IEEE Transactions on Systems, Man and Cybernetics* **21**(3) May/June, 660–674.
Salzberg, S, 1990. *Learning with Nested Generalised Exemplars*. Kluwer Academic.
Sankar, A and Mammone, R, 1991. "Combining neural networks and decision trees". *Applications of Artificial Neural Networks 2* SPIE, vol 1469, 374–383.
Schaffer, C, 1991. "When does overfitting decrease prediction accuracy in induced decision trees and rule sets".

In: *Proceedings EWSL91 – European Working Session on Learning* Kodratoff, Y (ed.), 192–205. Springer-Verlag.

Schlimmer, JC and Granger, RH, 1986a. "Beyond incremental processing: tracking concept drift". *Proceedings 5th National Conference on AI* Philadelphia, PA, 496–501. Morgan-Kaufmann.

Schlimmer, JC and Granger, RH, 1986b. "Incremental learning from noisy data". *Machine Learning Journal* **1** 317–354.

Schlimmer, JC and Fisher, D, 1986. "A case study of incremental concept learning". *Proceedings 5th National Conference on AI* 496–501. Morgan-Kaufmann.

Schlimmer, JC, 1987a. "Incremental adjustment of representations for learning". *Proceedings 4th International Workshop on Machine Learning* 79–90.

Schlimmer, JC, 1987b. "Learning and representation change". *Proceedings 6th National Conference on AI* Seattle, WA, 511–515.

Schoenauer, M and Sebag, M, 1990. "Incremental learning of rules and meta-rules". In: *Proceedings 7th International Conference on Machine Learning* Porter, B and Mooney, R (eds.), 49–57.

Sebag, M and Schoenauer, M, 1991a. "Learning by successive approximations". *Proceedings* Symbolic-Numeric Data Analysis and Learning Diday, E and Lechevallier, Y (eds.), 215–230. Nova Science.

Sebag, M and Schoenauer, M, 1991b. "Discovering meta-rules from rules and examples". *Proceedings 11th International Conference on Expert Systems and their Applications* vol 1, 205–216.

Seshu, R, 1989. "Solving the parity problem". In: *Proceedings EWSL89, 4th European Working Session on Learning* Marik, K (ed.), 263–271. Pitman.

Sethi, I and Sarvarayudu, 1982. "Hierarchical classifier design using mutual information". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **4**(4), July, 441–445.

Sethi, I, 1990. "Entropy nets: from decision trees to neural networks". *Proceedings of the IEEE* **78**(10) 1605–1613.

Sethi, I, 1991. "Decision tree performance enhancement using an artificial neural network implementation". In: *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections* Sethi, I and Jain, A (eds.), 71–88. Elsevier.

Shapiro, AD, 1987. *Structured Induction in Expert Systems*. Addison-Wesley.

Shavlik, J and Dietterich, T, 1990. "General aspects of machine learning". In: *Readings in Machine Learning* Shavlik, J and Dietterich, T (eds.), 1–10. Morgan-Kaufmann.

Shen, W, 1992. "Complementary discrimination learning with decision lists". *Proceedings 10th National Conference on AI* July, 153–158. AAAI/MIT Press.

Shifu, C, Bin, C and Jingui, P, 1992. "ICAS: an incremental concept acquisition system using attribute-based description". *Journal of Computer Science and Technology* **7**(3) July, 284–288.

Shlien, S, 1992. "Nonparametric classification using matched binary decision trees". *Pattern Recognition Letters* **13**(2) February, 83–87.

Shrager, J and Langley, P, 1990. *Computational Models of Scientific Discovery and Theory Formation*. Morgan-Kaufmann.

Sleeman, D, 1994. "Towards a technology and a science of machine learning". *AI Communications – European Journal on AI* **7**(1) March, 29–38.

Spears, W and De Jong, K, 1990. "Using genetic algorithms for supervised concept learning". *Proceedings 2nd International IEEE Conference on Tools for AI* 335–341. IEEE Press.

Sutton, R and Whitehead, S, 1993. "On-line learning with random representations". *Proceedings 10th International Conference on Machine Learning*. Morgan-Kaufmann.

Swain, P and Hauska, H, 1977. "The decision tree classifier: design and potential". *IEEE Transactions on Geoscience and Electronics* GE-15(3) July, 142–147.

Tadepalli, P, 1989. "Lazy explanation-based learning: a solution to the intractable theory problem". *Proceedings 11th International Joint Conference on AI* vol 1, 694–700. Morgan-Kaufmann.

Tan, M and Schlimmer, J, 1990. "Two case studies in cost-sensitive concept acquisition". *Proceedings 8th National Conference on AI* vol 2. AAAI/MIT Press.

Thrun, S, Bala, J, Bloedorn, E, Bratko, I, Cestnik, B, Cheng, J, De Jong, K, Džeroski, S, Fahlman, S, Fisher, D, Hamann, R, Kaufman, K, Keller, S, Kononenko, I, Kreuziger, J, Michalski, R, Mitchell, T, Pachowicz, P, Reich, Y, Vafaie, H, Van de Velde, W, Wenzel, W, Wnek, J and Zhang, J, 1991. "The Monk's problems: a performance comparison of different learning algorithms". *Technical Report*, Carnegie Mellon University, CMU-CS-91-197.

Towell, G and Shavlik, J, 1992. "Using symbolic learning to improve knowledge-based neural networks". *Proceedings 10th National Conference on AI* 177–182. MIT/AAAI Press.

Towell, G and Shavlik, J, 1994. "Refining symbolic knowledge using neural networks". In: *Machine Learning: A Multistrategy Approach* Michalski, R and Tecuci, G (eds.), vol 4. Morgan-Kaufmann.

Utgoff, P, 1986. *Machine Learning of Inductive Bias*. Kluwer Academic.

Utgoff, P, 1988a. "ID5: an incremental ID3". *Proceedings of the 5th International Conference on Machine Learning* Laird, J (ed.), 107–120. Morgan-Kaufmann.

Utgoff, P, 1988b. "Perceptron trees: a case study in hybrid representations". *Proceedings 7th National Conference on AI*. 601–606. Morgan-Kaufmann.

Utgoff, P, 1989a. "Incremental induction of decision trees". *Machine Learning Journal* **4** 161–186.

Utgoff, P, 1989b. "Improved training via incremental learning". In: *Proceedings 6th International Workshop on Machine Learning* Segre, A (ed.), 362–365. Morgan-Kaufmann.

Utgoff, P and Brodley, C, 1990. "An Incremental method for finding multivariate splits for decision trees". In: *Proceedings 7th International Conference on Machine Learning* Porter, B and Mooney, R (eds.), 58–65.

Utgoff, P, 1995. "Decision tree induction based on efficient tree restructuring". *Technical Report 95-18*, Department Computer Science, University of Massachusetts, Amherst, MA.

Vafaie, H and De Jong, K, 1994. "Improving a rule induction system using genetic algorithms". In: *Machine Learning: A Multistrategy Approach* vol 4, Michalski, R and Tecuci, G (eds.), 453–469, Morgan-Kaufmann.

Valdes-Perez, R, 1995. "Some recent human-computer discoveries in science and what accounts for them". *AI Magazine* **16**(3) 37–44.

Valdes-Perez, R, 1996. "Computer-science research on scientific discovery". *The Knowledge Engineering Review* **11**(1) 57–66.

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

Van de Velde, W, 1989. "IDL, or taming the multiplexor". In: *EWSL89 – Proceedings 4th European Working Session on Learning* Morik, K (ed.), 211–225. Pitman.

Van de Velde, W, 1990. "Incremental induction of topologically minimal trees". *Proceedings 7th International Conference on Machine Learning* 66–74. Morgan-Kaufmann.

Vrain, C and Lu, C, 1988. "An analogical method to do incremental learning of concepts". In: *Proceedings, EWSL88 – 3rd European Working Session on Learning* Sleeman, D (ed.). Pitman.

Wang, W and Chen, J, 1991. "Learning by discovering problem solving heuristics through experience". *IEEE Transactions on Knowledge and Data Engineering* **3**(4) December, 415–419.

Watanabe, L and Elio, R, 1987. "Guiding constructive induction for incremental learning from examples". In: *Proceedings 10th International Joint Conference on AI* McDermott, J (ed.). Morgan-Kaufmann.

Watson, I and Marir, F, 1994. "Case-based reasoning: a review". *The Knowledge Engineering Review* **9**(4).

Webb, G, 1995. "OPUS: an efficient admissible algorithm for unordered search". Unpublished manuscript.

Weiss, S and Kulikowski, C, 1991. *Computer Systems That Learn – Classification and Prediction Methods From Statistics, Neural Nets, Machine Learning and Expert Systems* Morgan-Kaufmann.

White, A and Liu, W, 1994. "Bias in information-based measures in decision tree induction". *Machine Learning Journal* **15** 321–329.

Winkelbauer, L and Fedra, K, 1991. "ALEX: Automatic Learning in Expert Systems". *Proceedings 7th Conference on Artificial Intelligence Applications* 59–62. IEEE Press.

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

Wnek, J and Michalski, R, 1994. "Comparing symbolic and subsymbolic learning: three studies". In: *Machine Learning: A Multistrategy Approach* Michalski, R and Tecuci, G (eds.), vol 4. Morgan-Kaufmann.

Zadeh, L, 1994. "Fuzzy logic, neural networks, and soft computing". *Communications of the ACM* **37**(3) 77–84.

Zytkow, J and Baker, J, 1991. "Interactive mining of regularities in databases". In: *Knowledge Discovery in Databases* Piatetsky-Shapiro, G and Frawley, W (eds.). AAAI Press.