İ.Ü. İşletme Fakültesi Dergisi, C: 25, S: 2/Kasım 1996, s.171–186.

MİSAFİR MAKALE

A REVIEW OF INDUCTIVE LEARNING ALGORITHMS

M. Sabih AKSOY
Sakarya University
Engineering Faculty Ind. Eng. Dept.
Esentepe, Adapazarı

ABSTRACT

In recent years, there has been a growing amount of research on inductive learning. Out of this research a number of promising algorithms have surfaced. In the paper knowledge acquisition, induction, inductive learning and the categories of inductive algorithms are discussed, CLS and its family, ID3 and its derivatives, AQ and its family, and recently developed RULES family of inductive learning algorithms, their strengths as well as weaknesses are explained and discussed respectively. Finally the applications of inductive learning are overviewed.

1. Knowledge Acquisition

Knowledge-based expert systems consist of two main components: a knowledge base and an inference mechanism. Collecting knowledge to form the knowledge base is the main task in the process of building an expert system[1,2,3].

The process of acquiring knowledge through interaction with an expert consists of a prolonged series of intense, systematic interviews, usually extending over a long period [4]. Human experts are capable of using their knowledge in their daily work, but they usually cannot summarise and generalise their knowledge explicitly in a form which is sufficiently systematic, correct and complete for machine representation and application [1]. Expert systems require large amounts of knowledge to achieve high levels of performance, yet the acquisition of knowledge is slow and expensive [5]. The shortage of trained knowledge engineers to interview experts and capture their knowledge is another problem of knowledge acquisition [6].

The aforementioned problems are not just difficulties of the early days of the technology, but are stili acknowledged today as paramount problems. Knowledge acquisition (and in particular machine learning) has become a major area of concern for expert systems research [5,7].

An alternative method of knowledge acquisition exists in which knowledge is learned, or induced, from examples. While it is very difficult for an expert to articulate his knowledge, it is relatively easy, to document case studies of the expert's skills at work [5]. Instead of asking an expert to summarise and articulate his knowledge, the main idea of automatic induction is to have him provide a basic structure of his discipline. The knowledge itself will be induced from examples expressed in this structure. Recent developments have proved that this method of knowledge acquisition is entirely possible. Indeed, the main feature of the second generation expert systems is that the knowledge acquisition process is highly automated [8].

2. Induction and Inductive Learning

In recent years, there has been a growing amount of research on inductive learning [9]. In its broadest sense, induction (or inductive inference) *is a method of moving from the particular to the general – from spesific examples to general rules* [5, 10,11]. Induction can be considered the process of generalising a procedural description from presented or observed examples [12,13,14].

The purpose of inductive learning is to perform a synthesis of new knowledge, and this is independent of the form given to the input information [15]. In order to form a knowledge base using inductive learning, the first task is to collect a set of represantative examples of expert decisions. Each example belongs to a known class and is described in terms of a number of attributes. These examples may be specified by an expert as a good tutorial set, or may come from some neutral source such as an archive. The induction process will attempt to find a method of classifying an example, again expressed as a function of the attributes, that explains the training examples and that may also be used to classify previously unseen cases [5]. The outcome of an induction algorithm is either a decision tree or a set of rules. Production rules can easily be extracted from decision trees [16, 17]. Each path of a decision tree can be regarded as an IF–THEN production rule.

3. Categories of inductive Learning Methods

Quinlan [18] states that there are two categories of induction methods: (1) Divide-and-conquer methods and (2) Covering methods. This classification depends on the strategy that the algorithms employ during the search for generalised descriptions.

Divide-and-conquer methods have received considerable attention among researches in the area of applied Al [19]. With a divide-and conquer method, the outcome is a decision tree. CLS [20] and ID3 [21] are two divide-and-conquer algorithms. In general, a divide-and-conquer algorithm constructs decision trees from a training set of objects according to the following simplified procedure:

* If ali training objects belong to a single class, the tree is a leaf labelled with that class.

* Otherwise,

– apply a test based on one attribute

– divide the training set into subsets, each corresponding to one of the possible (mutually exclusive) outcomes of the test, and

– repeat the procedure for each subset.

Algorithms of the covering type, in particular the AQ family of algorithms [22], represent classification knowledge as a disjunctive logical expression defining each class. These algorithms search for a set of possible generalisations in an attempt to find "good" hypotheses that satisfy certain requirements. The search proceeds by choosing as the initial working hypotheses some elements from the partially-ordered set of all possible descriptions. If the working hypotheses satisfy certain criteria then the search ends. Otherwise, the current hypotheses are modified by slightly generalising or specialising them. These new hypotheses are then checked to see if they satisfy the termination criteria [23]. The process of modifying and checking continues until the assumed criteria are met. The core of the covering method can be summarised as follows [18]:

* Find a conjunction of conditions that is satisfied by some objects in the target class, but not by objects from other classes.

* Append this conjuction as one disjunct of the logical expression being developed.

* Remove ali objects that satisfy this conjunction and, if there are still some remaining objects of the target class, repeat the procedure.

4. Divide–and–Conquer Induction Algorithms

4.1. CLS: Concept Learning System

One of the pioneers of concept learning using computer programs was Hunt who developed a series of learning algorithms called Concept Learning Systems, CLS–1 to CLS–9 [20]. All CLS versions except CLS-9 can deal only with binary classification problems. These algorithms are heuristic in the sense that they contain "rules of thumb" which uslly produce a concept in the form of a structurally simple tree with a minimum of computation. The CLS–1 algorithm, which is the basis of all later versions, is described in [20] as follows:

1. A sample of objects to be classified is randomly chosen from the given training set. It is assumed that these objects can belong to one of two classes, the Positive and Negative classes.

2. A search is made for some characteristics (values of attributes) or a set of characteristics which appear in objects in the Positive class and not in objects in the Negative class. If such characteristics are found, they are made the test associated with the root of the concept tree and the problem is solved (The information is stored in matrix form; ali that CLS-1 needs to do is compare successive rows representing positive or negative objects, column by column).

3. If step (2) cannot be carrid out, it is reversed. A search is made for a set of characteristics which appear in objects in the Negative class and not in any object in the Positive class. If such a set can be found, then the problem is solved as before.

4. If steps (2) and (3) fail, the relative frequehcies of characteistics that appear in the objects in the Positive class are counted. The characteristic with the highest frequency of occurrence is chosen as the test at the current node. The training set is split into two subsets, one containing all objects having the characteristic just chosen, and the other containing ali objects that do not.

5. The same procedure is reapplied to the subsests until no further divisions are required (which means ali objects in training set are correctly classified).

This procedure assumes that there are both positive and negative instances in the original set of examples. If, at any point in solving either the original problem or a subproblem CLS-1 encounters a sample containing only positive (negative) instances, the current node is immediately made a positive (negative) endpoint and the next sub-problem is attacked. In steps 2 and 3, if more than one characteristic qualify for selection as a test a random choice is made between them.

As mentioned above, Hunt et al, have developed nine versions of CLS. CLS-2 and CLS-3 are similar to CLS-1 except that an additional control parameter is inserted to indicate the maximum memory size of the computer (CLS–2 and CLS-3 differ from each other in the way that the set of examples is stored). CLS-4 and CLS-5 differ from CLS-1 only in the manner of choosing a sample. They were developed to improve the performance of CLS-1 by introducing some control over the order in which objects are added to the working set, following the orwrence of classification errors. CLS-6 is akin to CLS-1 except that sub samples are selected using relative freqnency. CLS-7 differs from CLS-1 only by redefining the Positive set. In this algorithm, the set of positive instances is defmed at each node as the largest of the two sets. CLS-8 is similar to CLS-7 except that the smaller, rather than the larger set, is considered as the Positive set. CLS-9 has the advantages of many of the earlier algorithms and allows multi-branch trees to be generated. More details of these algorithms can be found in [20].

4.2. ID3 Inductive Learning Algorithm

ID3, a descendant of Hunt's CLS, is a simple algorithm for discovering a set of classification rules from a collection of objects belonging to different classes [21]. ID3 uses a multi-branch rather than a binary decision tree. It differs from CLS-9 in the way it chooses the test node (attribute) to divide the original set of examples or subsets. The attribute selection part of ID3 is based on the assumption that the complexity of the decision tree is strongly related to the amount of information contained in the tree. ID3 uses an information-theoretic approach aimed at minimising the expected number of tests to classify an object [21]. The method can be summarised as follows:

1. Select at random a subset (or window) of the given instances.

2. Form a rule to explain the current window.

3. Find the exceptions to this rule in the remaining instances.

4. Form a new window from the current window and the exceptions to the rule generated from it.

5. If there are no exceptions to the rule then stop. Otherwise, repeat steps 2 to 5 until no exceptions remain.

The core procedure of the algorithm is the recursive process to form a decision tree from the current window. Let S be a set of objects. If all objects in S belong to the same class, then the decision tree is a leaf bearing that class name. Otherwise, S contains representatives of more than one class. An attribute is selected to partition S into subsets S_1, S_2, ..., S_n where S_i contains those members of S that have the i^{th} value of the selected attribute. An object is classified by starting at the root of the decision tree. To find the root of the tree, the information gain of each attribute is computed. Let A be a set of m attributes *{A_1, A_2, ..., A_m}* and C a set of p classes *{C_1, C_2, ..., C_p}*. The set of possible values for an attribute Aj is referred to as Range (Aj). Each example in *S* is an m+1 tuple of the form: *(V_1, V_2, ..., V_m, C_k)* where V_j Range *(A_j), j=1, ... m* and C_k *C* is the class of the example.

Define the probability of occurrence of examples of class C_k in a set S_l, to be the proportion of examples in S_l that S_l in class C_k. As a measure of the randomness of example distribution in S_l over the possible classen in C, the information measure is defined thus [24, 25, 26].

$$I(S_l) = -\sum_{k=1}^{p} P_{S_l,C_k} \mathrm{Log}_2 P_{S_l,C_k}$$

ID3 aims to partition S to produce subsets S_l in which the examples are distributed less randomly over the possible classes. To choose the attribute that would best achieve this, for each attribute Aj having more than one value in the examples in S, ID3 partitions S into subsets S_l consisting of ali examples in S having value V_l for attribute Aj. If the number of examples containing value V_l in S is n(S) and the number of examples containing value V_l in subset S_l is n(S_l), the information entropy of the resulting partition is given by:

$$E(A_j, S) = \sum_{V_l \in \mathrm{Range}(A_j)} I(S_l)\frac{n(S_l)}{n(S)}$$

ID3 chooses as a node the attribute Aj which maximises the following quantity:

Gain (Aj, S) = I(S) –E(Aj, S)

If Gain (Aj, S) is the same for more than one attribute Aj, then one of them is randomly chosen.

ID3 is a non-incremental algorithm and is a good choice for building a classification rule set for problems in which a database of instances is available and is not likely to change. However, for problems in which new instances are expected to become available on a regular basis, it is preferable to accept instances incremantally, without needing to build a new decision tree from the beginning each time.

Since ID3 was developed, it has been improved several times by Quinlan and other researchers. Schlimmer and Fisher have proposed

ID4 [27], which incremantally builds a decision tree similar to that which ID3 would build. Instead of forming a decision tree from a window of instances, ID4 updates a tree based on each individually observed instance.

Utgoff [28] has described ID5 which builds on ID4 but differs from ID4 in its method of replacing the test attribute. When ID4 replaces a test attribute, it discards the subtrees below the old test attribute. ID5 reshapes the tree by pulling the best attribute up from below. An experimental comparison of ID3, ID4 and ID5 is given in [28].

A problem with ID3 is that the decision tree produced might not be as general as it could be. This is caused by the fact that decision rules can sometimes involve unnecessary or irrelevant conditions. When the algorithm chooses an attribute for branching out of a node, it produces a branch for each value of the attribute. However, some of those values may be irrelevant to the classification. Those irrelevant values will result in overspecialized classification rules. Cheng et al. [25] have developed GID3 (Genaral version of ID3) to overcome this problem. Essentially, the algorithm is similar to ID3 except that not every value of the attribute is chosen to produce a branch. In order to avoid branching on irrelevant values of the attribute, only values that appear to be relevant, according to their information measure (a quantity similar to the information measure used in ID3), may potentially be branched on. A user-defined tolerance level is adopted to specify the degree of tolerance for the deviation of the entropy measure of an attribute-value pair from the minimal entropy measure over all pairs. Cheng et al. have not decribed how to set the tolerance level systematically.

5. Induction Algorithms Based on Covering:

5.1. AQ and its family

Unlike decision-tree based algorithms, AQ produces IF-THEN rules directly. AQ is an algorithm for developing Variable-Valued Logic (VL) rules from a given set of examples (training set) [29, 30]. VL rules are equivalent to decision rules. For example, the following VL rule:

$[x\neq 6]\,[x=3,5] \vee [x>7] \Rightarrow [decision = B]$

can be interpreted as:

IF x IS NOT 6 AND x IS 3 or 5, OR x IS LARGER THAN 7 THEN decision is B.

Before describing the AQ family of algorithms, the terminology used by Michalski will be introduced.

Selectors are predicates over attribute subranges. For example, the selector *[hair = dark, red]* is a predicate over the attribute hair. It indicates that *dark* and *red* are the allowed values of hair.

Conjunctive Concepts are groups of selectors where no two selectors refer to the same atrribute.

Rules are of the form

Conjunctive concepts $\Rightarrow$ concept class.

A complete description is a concept description (a group of attribute-value pairs) that correctly describes ali the positive examples (The class which an example belongs to is considered the Positive class. Ali other classes form the Negative class).

A consistent description is a concept description that fits none of the negative examples.

A star is a set of ali possible alternative, nonredundant descriptions of a chosen unclassified example (the "seed") which cover that example and do not cover any of the examples in the Negative class.

The **"extend against"** process is a process used in **AQ** to extend the value of an attribute that appears in the seed, against the set of values for the same attribute for examples in the negative set. The algorithm determines the most general consistent value which does not intersect the negative set on that attribute. The results of extending the seed against each negative event are then combined by intersecting the values for each of the previous tests for each attribute. The system then selects these combined values to give a conjunctive description that covers the seed, but none of the negative events.

The algorithm is provided with two preclassified sets of events. Events are observations from the problem domain, where each event is a list of attribute-value pairs. The basic algorithm is performed once for each class, generating a single VL rule per class [31]. **AQ** begins with an unclassified example (a seed) from the Positive class and generates a star using the "extend against" process. In that star, a description is chosen according to a consistency measure and a completeness measure. If the description is consistent (covers no negative events) it is saved in the concept set C (by disjoining it with the previous conjunctive concepts in C). If ali the positive examples and no negative examples are covered by the obtained description then a new positive class cënstructed and the same procedure is repated. If there are some examples that are not is covered by the description, then a new seed is selected from the remaining uncovered examples in the Positive class and the procedure is repeated until ali ëxamples in the Positive class (are covered. The Process continues until all examples in the training set are correctly classified. A general and simplified version of AQ can be summarised as follows:

1. Select an unclassified example. Consider the class of the example as the Positive class, and ali other classes as the Negative class.

2. Apply the extend against process to generate a star for the example.

3. From the star obtained in step (2), select a description D according to the consistency and completeness values.

4. If description D covers ali positive examples then go to step (6).

5. Otherwise, reduce the set of positive examples to contain only events not covered by D, and repeat the process from step (1).

6. Convert ali generated descriptions to the form of a VL rule which can cover ali examples in the Positive class, and remove those examples from the set of unclassified examples. If there are no more unclassified examples then stop, else go to step (1).

Since Michalski first developed AQ in 1969, it has been improved several times. Michalski and Larson have described AQ11 [32] which uses the same search strategy as AQ [33] but can work incremantally. That is, when new examples become available, AQ11 can modify a previously obtained set of rules to make them consistent with those examples. Hong, Mozetic and Michalski have proposed AQ15 [34] which is similar to AQ11. An important feature of AQ15 is the ability to deal with noisy or overlapping examples. Bloedorn and Michalski [35] have reported the AQ17–DCI (Data-driven contructive induction) algorithm. AQ17–DCI is an algorithm for constructive induction (an induction process that produces new attributes not present in the training events). The algorithm generates a number of new attributes and selects as test attributes those with a "quality" value exceeding a given threshold. Thrun et al have described the AQ17–FLCS (flexible concept learning) algorithm that combines both symbolic and numeric representations in generating a concept description [36]. Wnek and Michalski have proposed AQ17–HCI (Hypothesis driven constructive induction) which implements an iterative constructive induction procedure in which the generation of new attributes is based on the analysis of the hypotheses produced in the previous iteration [37]. Pachowicz and Bala have reported AQ14–NT (Noise tolerant). The algorithm is specially designed for learning from noisy engineering data [38, 39]. Thrun et al. have described AQ15–GA which combines AQ15 and a genetic algorithm (GA) for attribute selection. The GA is used to explore the space of all subsets of a given attribute set. Each of the selected attribute subsets is evaluated (its fitness measured) by invoking AQ15 and determining the recognition rate of the rules produced [36].

5.2. RULES and Its Family

Pham and Aksoy have developed RULES (RULe Extraction System) [40], a simple algorithm for extracting a set of classification rules for a collection of objects belonging to a given set of classes. An object must be described in terms of a fixed set of attributes, each with its own set of possible values. For example "Weather" and "Temperature" might be attributes with sets of possible values {rainy, sunny, snowy} and {low, average, high} respectively.

In RULES, an attribute-value pair constitutes a condition. If the number of attributes is n_a, a rule may contain between one and n_a, conditions, each of which must be a different arttribute-value pair. The conjunction of conditions only is permitted in a rule and therefore the attributes must ali be different if the rule comprises more than one condition. The attributes and the values associated with them in a collection of objects form an array of attiributes and values. The total number of elements of the array is the total number of all possible values. For example if there are four attributes with, 3, 4, 2 and a 5 values respectively, the total number of elements is 14.

The rule forming procedure may require at most n_a iterations. The first iteration produces rules with one condition and the second iteration results in rules with two conditions, etc. In the first iteration, each element of the array attributes and values is examined to decide whether it can forma rule with that element as the condition. For the whole set of examples if a given element applies to only one class, then it is a candidate for forming a rule. If it pertains to more than one class, it is passed over and the next element is examined. When ali elements of the array have been looked at, the whole set of examples is checked for any example that cannot be classified by the candidate rules. If there are no unclassified examples the procedure terminates. Otherwise, a new array is constructed which comprises attributes and values contained in ali the unclassified examples. In the second iteration elements of the array are examined in pairs to determine whether they apply to only one class in the whole set of examples. As before, for those pairs of elements that pertain to unique classes, candidate rules are obtained. If there are stili unclassified examples at the end of this iteration, a new array is formed and the next iteration is initiated. This procedure continues until all examples are correctly classified or the number of iterations (the number of conditions) is equal to n_a. In the latter case, ali remaining unclassified examples are taken as rules. For each iteration after the first, candidate rules extracted in the current iteration are checked against previously obtained rules. Candidate rules that do not contain irrelevant conditions are added to the rule set and the others are ignored. This check is not required for the first iteration as each rule can only have one condition. The procedure can be summarised as follows:

1. $n_c = 0$

2. If $n_c < n_a$ then $n_c = n_c+1$.

3. Find all values contained in unclassified examples.

4. Form objects which are combinations of n_c values taken from the values obtained in Step (3).

5. If at least one of the objects belongs to a unique class then form rules with those objects ELSE go to Step (2).

6. Check for irrelevant conditions.

7. Check all unclassified examples using the extracted rules.

8. Remove ali classified examples.

9. If ali examples are classified using extracted rules then STOP; ELSE go to Step (2).

Pham and Aksoy have improved RULES and produced RULES-2 [41]. The rule forming strategy of RULES–2 is almost the same as that of RULES. The difference is that instead of considering the values of all unclassified examples, in each iteration, only the values of one unclassified example are used to produce rules for classifying that example. Compared to RULES, RULES–2 is generally faster as it requires fewer rule searching operations in its induction process. Furthermore, it allows the user to specify the number of rules to be extracted, is able to deal with incomplete examples and can handle attributes with numerical as well as nominal values.

The latest version of RULES family of automatic rule extraction systems is RULES-3 [42]. RULES–3 inherits ali advantageous features of its predecessors. In addition it has two new features: it generates a compact set of more general rules and provides the user with the option of adjusting the precision of the extracted rules.

6. Applications of inductive Learning

Inductive learning algorithms are domain independent. In principle, they can be used in any task involving classification or pattern recognition [1]. There have been several succesful applications of inductive learning systems. Medical applications such as lymphography, prognosis of breast cancer recurrence, location of primary tumour and thyroid problem diagnosis have been reported [5, 43, 44]. Other applications include investment appraisal [45], forensic classification of glass fragment evidence [46], extraction of decision rules for analysis of test data for the space shuttle main engine [47], experimental generation of decision rules for the conceptual design of steel members under bending [10], soil classification [48], stock control [49], software resource analysis [50], assessing credit card applications [51], military decision making [52], dynamic system identification [53, 54], engine fault diagnosis [55] and identification of the mass-spectra of complex materials [56, 57, 58].

7. Conclusion

The induction of decision trees and rules from empirical data is a useful technique for automatic knowledge acquisition. It offers a modularised, clearly explained format for decision making which is compatible with human reasoning procedures. Also, the resulting rules are suitable for use in expert systems. Two of the largest expert systems developed prior to 1987 (BMT and GASOIL) were built using automatic induction [1]. Also it is reported that the decision tree based algorithms have been incorporated into a number of commercial systems including Expert-Ease, RuleMaster and ACLS [59]. In recent years, more task-oriented inductive learning systems have been developed that have demonstrated, impressive performance in their specific domain of application. However, some problems still remain. Most systems lack generality and extensibility. The theoretical principles upon which they are built are often not well explained. Lack of common terminology and an adequate formal theory makes it difficult to compare diferent learning methods [60].

REFERENCES

[1] Liu W.Z. and White A.P. (1991) "A review of inductive learning", in proc. *Research and Development in Expert Systems VIII*, Cambridge, pp. 112-126.

[2] Mrozek, A. (1992) "A new method for discovering rules from examples in expert systems", *Int. J. Man-Machine Studies*, 36. pp. 127-143.

[3] Hart A. (1989) "Knowledge acquisition for expert systems", Chapman and Hail, London.

[4] Waterman D.A. (1986) "A guide to expert systems", Addison-Wesley, California.

[5] Quinlan J.R. (1988) "Induction, knowledge and expert systems",in *Artificial Intelligence Developments and Applications,* Eds J.S. Gero and R. Stanton, Amsterdam, North-Holland, pp. 253-271.

[6] Weiss S.M. and Kulikowski C.A. (1991) "Computer systems that learn", Morgan Kaufmann, San Mateo, California.

[7] Williams G.J. (1988) "Combining decision trees, initial results from MIL algorithm", in *Artificial Intelligence Developments and Applications,* Eds: J.S. Gero and R. Stanton, pp. 273-289.

[8] Devedzic, V. and Velasevic D. (1990) "Features of second generation expert systems, an extended overview", *Eng. Appl. of AI*, Vol. 3, December, pp. 255-270.

[9] Nakakuki Y., Koseki Y. and Tanaka M. (1990) "Inductive learning in probabilistic domain", in proc. Eighth National Conf. on Al, Boston, July 29, August 3, pp. 809-814.

[10] Forsyth R. (1989) "Machine Learning principles and tecniques", Ed: R. Forsyth, Chapman and Hall, London.

[11] Hancox P.J., Mills W.J. and Reid B.J. (1990) "Artificial intelligence / expert systems", Ergosyst Associates, Lawrence, Kansas.

[12] Charniak, E. and McDermott, D. (1985) "Introduction to artifical intelligence", Addison-Wesley, California.

[13] Tanimoto S.L. (1987), "The elements of artifical intelligence", Computer Science Press, Maryland, USA.

[14] Rubin S.H. (1991) "Expert systems for knowledge acquisition" in proc. *First World Congress on Expert Systems,* Vol. 3 Orlando, Florida, December 16-19, pp. 1793-1799.

[15] Kodratoff Y. (1988) "Introduction to machine learning", Pitman Publishing, London.

[16] Al-Attar A. (1991) "Rule induction from mythology to methodology", *Research and Developments in Expert Systems VIII,* London, September, pp. 85-103.

[17] Quinlan J.R. (1987a) "Generating production rules from decision trees", in proc. *Tenth IJCAI–87,* Milan, Italy, pp. 304-307.

[18] Quinlan J.R (1990) "Learning logical definitions from relations", in *Machine Learning,* 5, Kluwer Publishers, Boston, pp. 239-266.

[19] Shapiro S.C. et. al. (1991) "Encyclopedia of artifical inteligence", Second edition.

[20] Hunt E.B, Marin J., and Stone P.J. (1966) "Experiments in iduction", Academic Press, New York.

[21] Quinlan J.R. (1983) "Learning efficient classification procedures and their applications to chess end games" in *Machine Learning, An Artificial Intelligence Approach,* Eds: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, Morgan Kaufmann, Tiago, Palo Alto, CA, pp. 463-482.

[22] Michalski R.S. (1990) "A theory and methodology of inductive learning", in *Readings in Machine Learning,* Eds: J.W. Shavlik and T.G. Dietterich, Morgan Kaufmann, San Mateo, California, pp. 70-95.

[23] Dietterich, T.G. and Michalski R.S. (1983) "A comparative review of selected methods for learning from examples", in *Machine Learning, an*

Artificial Intelligence Approach, Vol 1, Eds: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, Morgan Kaufmann, pp. 41-81.

[24] Evangelos, I.P., et. al (1992) "A minimum entropy approach to rule learning from examples", *IEEE Trans. Systems Man and Cybernetics,* 22(4), July/August, pp. 621-635.

[25] Cheng J. et. al. (1988) "Improved decision trees: a generalised version of ID3", in proc. Fifth Int. Conf. on Machine Learning, The University of Michigan, Ann Arbor, M1 June, 12-14, pp. 100-106.

[26] Mace R. (1974) "Management information and the computer", Haymarket, London.

[27] Schlimmer J.C and Fisher D. (1986) "A case study of incremental concept induction" in proc. *Fifth National Conf. on AI,* Morgan Kaufmann, San Mateo, CA, pp. 496-501.

[28] Utgoff P.E., (1988) "ID5; an incremental ID3"In proc. *Fifth Int. Conference on Machine Learning,* The University of Michigan Ann Arbor, MI, June 12-14, pp. 107-120.

[29] Michalski R.S. (1973) "Discovering classification rules using variable–valued logic system VL1", in *Artificial Intelligence Proceedings of the Third Int. Joint Conf.* Stanford, pp. 162-172.

[30] Michalski R.S. (1975) "Synhesis of optimal and quasi-optimal variable-valued logic formulas", in proc. 1975 *Int. Symposium on Multiple-Valued Logic,* Bloomington, May, pp. 76-87.

[31] Chan K.C.C., Ching J.Y. and Wong A.K.C. (1992) "Learning fault diagnostic rules: a probabilistic inductive inference approach", in *Applications of AI in Engineering VII,* Eds: D.E. Grierson, G. Rzevski and R.A. Adey, Elsevier Applied Science, New York, pp. 125-142.

[32] Michalski R.S. and Larson J.B. (1978) "Selection of most represantative training examples and incremental generation of VL1 hypothesis: the underlying methodology and the descriptions of programs ESEL and AQ11", Report No. 867, Department of Computer Science, University of Illinois, Urbana, Illinois.

[33]Cohen P.R. and Feigenbaum E.A. (1982) "The handbook of artificial intelligence", Vol. 3, William Kaufmann, California.

[34]Hong J., Mozetic I. and Michalski R.S. (1986) "AQ15: incremental learning of attribute-based descriptions from examples, the method and user's guide", Report ISG 86-5, UIUCDCS-F-86-949, Dept. of Computer Science, Univ of Illinois, Urbana, Illinois.

[35] Bloedorn E. and Michalski R.S. (1991) "Data-driven constructive induction in AQ17–DCI: a method and experiments", *Reports of Machine Learning and Inference Laboratory,* Center for Artificial Intelligence, George Mason University.

[36] Thrun S.B. et al. (1991) "The MONK's problems- a performance comparison of different learning algorithms", School of Computer Science, Carnegie Mellon University, Research Report, CMU-CS-91-197, December, Pittsburg, Pennsylvania.

[37] Wnek J. and Michalski R.S. (1991) "Hypothesis-driven constructive induction in AQ17: a method and experiments", in proc. *Twelfth Int. Joint Conf. on AI*, August, Sydney Australia.

[38] Pachowicz P.W. and Bala J. (1991a) "Improving recognition effectiveness of noisy texture concepts through optimization of their descriptions", in proc. *8 th Int. Workshop on Machine Learning,* Evanston, pp. 625-629.

[39] Pachowicz P.W. and Bala J. (1991b) "Advancing texture recognition through machine learning and concept optimisation", *Reports of Machine Learning and Inference Laboratory,* MLI-6, Artificial Intelligence Center, George Mason University.

[40] Pham, D.T. and Aksoy M.S. (1995a) "RULES: A simple rule extraction system", *Expert Systems with Applications,* Vol. 8, No.1, pp. 59-65, USA.

[41] Pham, D.T. and Aksoy M.S. (1993) "An algorithm for automatic rule induction", *Artificial Intelligence In Engineering,* No: 8, pp. 277-282, U.K.

[42] Pham D. T. And Aksoy M.S. (1995b), "A new algorithm for inductive learning", *Journal of Systems Eng.,* No: 5, pp. 115-122, U.K.

[43] Michalski R.S. et al (1986) "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains", in proc. *National Conf. on AI*, Philadelphia, PA., August, pp. 1041-1044.

[44] Quinlan J.R. (1987b), "Inductive knowledge acquisition: a case study", in *Applications of Expert Systems*, Ed: J.R. Quinlan, Turing Institute Press, pp. 157-173.

[45] Race P.R and Thomas R.C. (1988), "Rule induction in investment appraisal", *Journal of the Operational Research Society,* 38, pp. 1113-1123.

[46] Spiehler E.J. (1987), "Application of machine learning to classification of glass fragment evidence in forensic science", Seminar Materials, Machine Learning Seminar, Learned Information Ltd., Oxford, and Machine Learning Research Ltd., Nottingham, London.

[47] Modesitt K.L. (1987) "Space shuttle main engine anomaly data and inductive knowledge-based systems: automated corporate expertise", in proc. *Conf Artificial Intelligence for Space Applications,* Huntsville, Alabama, November, pp. 1-8.

[48] Dale M.B. McBratney A.B. and Russell J.S. (1989), "On the role of expert systems and numerical taxonomy in soil classification", *Journal of Soil Science,* 40 pp. 223-234.

[49] Thorpe J.C., Marr A and Slack R.S. (1989), "Using an expert system to monitor an automatic stock control system", *Journal of the Operational Research Society,* 40, pp. 945-952.

[50] Selby R.W. and Porter A.A. (1988) "Learning from examples: generation and evaluation of decision trees for software resource analysis", *IEEE Transactions on Software Engineering,* 14, pp. 1743-1756.

[51] Carter C and Catlett J. (1987) "Assesing credit card applications using machine learning", *IEEE Expert Intelligent Systems and Their Applications,* FALL, pp. 71-79.

[52] Lirov Y., Rodin E.Y. and Ghosh B.K. (1989) "Automated learning by tactical decision systems in air combat", *Computer and Mathematics with Application,* 18, pp. 151-160.

[53] Batur C., Srinivasan A. and Chan C.C. (1991) "Automated rule-based model generation for uncertain complex dynamic systems", *Eng. Appl. of AI,* 4(5), pp. 359-366.

[54] Pham D.T. and Aksoy M.S. (1993b), 'Dynamic system modelling using a new induction algorithm", (to be published).

[55] Ke M. and Ali M. (1989), "A learning representation and diagnostic methodology for engine fault diagnosis", in proc. *Second Int. Conf. on Industrial and Engineering Applications of AI and Expert Systems,* Vol. 2, June, 6-9, pp. 824-830.

[56] Harrington P.D. Street T.E. and Voorhees K.J. (1989),"Rule building expert systems for classification of mass spectra", *Analytical Chemistry,* 61, pp. 715-719.

[57] Harrington P.D. and Voorhees K.J. (1990) "Multivariate rule-building expert system", *Analytical Chemistry,* 62, pp. 729-734.

[58] Scott D.R. (1989) "Classification and identification of mass spectra of toxic compounds with an inductive rule building expert system and information theory", *Analytical Chimica Acta,* 223, pp. 105-121.

[59] Wu X. (1993) "Inductive learning: Algorithms and Frontiers", *Artificial Intelligence Review,* 7, pp. 93-108.

[60] Michalski, R.S. and Stepp R.E. (1983), "Learning from observation: conceptual clustering", in *Machine Learning and Artificial Intelligence Approach*, Vol 1, Eds: R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, Morgan Kaufman, pp. 331-363.