Machine-learning techniques and their applications in manufacturing

D T Pham* and **A A Afify**
Manufacturing Engineering Centre, Cardiff University, Cardiff, UK

The manuscript was received on 24 August 2004 and was accepted after revision for publication on 9 February 2005.

DOI: 10.1243/095440505X32274

Abstract: Machine learning is concerned with enabling computer programs automatically to improve their performance at some tasks through experience. Manufacturing is an area where the application of machine learning can be very fruitful. However, little has been published about the use of machine-learning techniques in the manufacturing domain. This paper evaluates several machine-learning techniques and examines applications in which they have been successfully deployed. Special attention is given to inductive learning, which is among the most mature of the machine-learning approaches currently available. Current trends and recent developments in machine-learning research are also discussed. The paper concludes with a summary of some of the key research issues in machine learning.

Keywords: machine learning, inductive learning, intelligent manufacturing, classification, scaling up, multiple models

1 INTRODUCTION

Many knowledge-based systems have been employed to automate different operations in manufacturing. Examples are expert systems for decision support, intelligent-scheduling systems for concurrent production, and fuzzy controllers. A problem is the gathering of the required expert knowledge to implement these knowledge-based systems. Machine-learning techniques can help in automating the time-consuming process of knowledge acquisition that is essential to the development of a knowledge-based system. Automation would increase the speed and reduce the cost of development by decreasing the amount of time needed from experts and knowledge engineers. Automation could also uncover knowledge that might otherwise be overlooked by those involved in the knowledge-acquisition process.

The field of machine learning is concerned with enabling computer programs automatically to improve their performance at some tasks through experience. The field is closely related to pattern recognition and statistical inference. A great deal of research in machine learning has focused on classification, the task of developing a model, from a set of previously classified examples, that can correctly categorize new examples from the same population. Classification has a wide range of applications, including manufacturing, telecommunications, marketing, and scientific analysis. Many manufacturing problems fall under the category of classification, where industrial domain experts are asked to assign a class label to an object or a situation based on the specific values of a set of parameters.

Machine-learning approaches commonly used for classification include inductive-learning algorithms such as decision-tree induction [**1**] and rule induction [**2**], instance-based learning [**3**, **4**], neural networks [**5**], genetic algorithms [**6**], and Bayesian-learning algorithms [**7**]. Among the various machine-learning approaches developed for classification, inductive learning from instances may be the most commonly used in real-world application domains. Inductive-learning techniques are fast compared to other techniques. Another advantage is that inductive-learning techniques are simple and their generated models are easy to understand. Finally, inductive-learning classifiers obtain similar and sometimes better accuracies compared with other classification techniques.

*Corresponding author: Manufacturing Engineering Centre, Cardiff University, Cardiff CF24 3AA, UK. email: phamdt@cf.ac.uk

The paper is organized as follows: section 2 formally defines the classification-learning problem and presents a framework for viewing approaches to it; section 3 describes in some detail the different techniques for inductive learning that have been or could be applied in the manufacturing domain; and section 4 briefly reviews other major machine-learning approaches. Current trends and recent developments in machine-learning research are presented in section 5. Successful manufacturing applications of machine-learning techniques are reviewed in section 6. Section 7 concludes the paper with a summary of some of the key research issues in machine learning.

2 THE SUPERVISED CLASSIFICATION-LEARNING PROBLEM

In classification learning, a learning algorithm is given a sample of preclassified examples from the problem domain called the training set. Each example is described by a vector of attributes. An attribute is either nominal or continuous. The algorithm learns a model that is used to predict the classification of future examples.

Learning methods can be divided into supervised and unsupervised schemes based on whether or not a dedicated target function for prediction has been defined. In unsupervised methods, such a function is not available and the goal is grouping or clustering instances based on some similarity or distance measure. In supervised learning, there is either a nominal or continuous-valued target function to be predicted. The former case is referred to as classification and the latter as regression or continuous prediction. In this paper, only methods for supervised classification learning will be addressed.

Ideally, given a complete description of an example (i.e. the values of all its attributes), its class should be unambiguously determined. In practical tasks, however, the available attributes will often not contain all the information necessary to do this. The training set may contain examples with the same attribute values but in different classes. Also, examples may appear with erroneous class values, or with erroneous attribute values, or both. These errors may stem from a diversity of sources, including limitations of measuring instruments and human error while typing examples into a computer. All these phenomena are referred to collectively as noise and limit the achievable accuracy in an induction problem. The degree of robustness of a learning system with respect to noise is one of its most important characteristics. It also occurs often in practice that the values of certain attributes for certain examples are simply not available. These are called missing values and again a practical induction system must be able to handle them.

3 DESCRIPTION OF INDUCTIVE-LEARNING TECHNIQUES

A classification-learning algorithm can be viewed as having three components: representation, search, and evaluation [**8**]. The representation component is the formal language in which concepts are described; the output of the learning algorithm is a statement in this language. The search procedure is the process by which the learning algorithm finds the concept description in the space of possible descriptions defined by the representation language. The evaluation component takes a candidate concept description and returns a measure of its quality. This is used to guide the search, and possibly to decide when to terminate it. Often, different evaluation procedures are used for these two purposes.

Inductive-learning techniques can be divided into two main categories, namely, decision-tree induction and rule induction. Each of these categories will be analysed in view of the above three components.

3.1 Decision-tree induction

There are a variety of algorithms for building decision trees. The most popular are: CART [**9**], ID3, and its descendants C4.5 and C5.0 [**10–13**]. These learning systems are categorized as 'divide-and-conquer' inductive systems. The knowledge induced by these systems is represented as decision trees. A decision tree consists of internal nodes and leaf nodes. Each internal node represents a test on an attribute and each outgoing branch corresponds to a possible result of this test. For a nominal attribute A_i with n_{A_i} possible values $v_{i1}, v_{i2}, \ldots, v_{ij}, \ldots, v_{in_{A_i}}$, there are n_{A_i} different branches originating from an internal node. For a continuous attribute A_i, a binary test is carried out and a corresponding branch $A_i \leqslant t_{ij}$ is created, with a second branch corresponding to $A_i > t_{ij}$, where t_{ij} is a threshold in the domain of A_i. Each leaf node represents a classification to be assigned to an example. Table 1 shows an example data set and Fig. 1 displays a decision tree constructed from these data.

To classify a new example, a path from the root of the decision tree to a leaf node is identified based

Table 1 An example of a data set

Vibration	Pressure	Temperature	Fault type
Present	30	65	A
Absent	23	15	B
Absent	40	75	B
Present	55	40	A
Absent	55	100	B
Present	45	60	A
Present	25	55	A
Absent	24	20	B

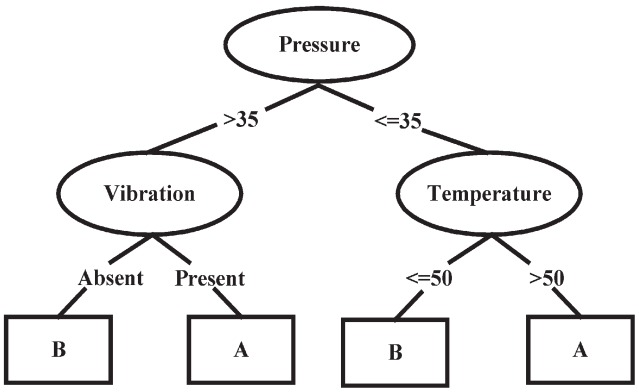

Fig. 1 A decision tree constructed from the data in Table 1

on values of the attributes of the example. The class at the leaf node represents the predicted class for that example.

Decision trees are generated from training data in a top–down, general-to-specific direction. The initial state of a decision tree is the root node that is assigned all the examples from the training set. If it is the case that all examples belong to the same class, then no further decisions need to be made to partition the examples and the solution is complete. If examples at this node belong to two or more classes, then a test is made at the node that will result in a split. The process is recursively repeated for each of the new intermediate nodes until a completely discriminating tree is obtained.

CART is a binary decision-tree algorithm that is extensively used. The evaluation function used for splitting in CART is the Gini index. Given a labelled data set S with k classes, let k classes be $C_1, C_2, \ldots, C_k$ and let $P(C_j, S)$ be the proportion of instances in S which are in class C_j. Then the index is defined as

$$\text{Gini}(S) = 1 - \sum_{j=1}^{k} P(C_j, S)^2 \tag{1}$$

For each candidate split, the 'impurity' (as defined by the Gini index) of all the subpartitions is summed and the split that causes the maximum reduction in impurity is chosen.

ID3 is a well-known decision-tree system. It utilizes the information gain criterion for splitting nodes. The information gain is computed from the entropy measure that characterizes the impurity in a collection of training instances as explained below. For a given data set S, the entropy is defined as

$$\text{Entropy}(S) = -\sum_{j=1}^{k} P(C_j, S) \log_2 P(C_j, S) \tag{2}$$

Let a test T with b outcomes partition the data set S into $S_1, S_2, \ldots, S_b$. Then, the total entropy of the partitioned data set is defined as the weighted sum of the entropy of the subsets as described below

$$\text{Entropy}(S, T) = \sum_{i=1}^{b} \frac{|S_i|}{|S|} \text{Entropy}(S_i) \tag{3}$$

where $|S_i|$ and $|S|$ are the numbers of instances in S_i and S respectively.

The information gained by partitioning in accordance with the test T is measured by

$$\text{Gain}(S, T) = \text{Entropy}(S) - \text{Entropy}(S, T) \tag{4}$$

$\text{Gain}(S, T)$ is therefore the expected reduction in entropy as a result of partitioning the data set into mutually exclusive subsets based on test T. The gain criterion selects a test to maximize this information.

C4.5, a variant and extension of ID3, is another popular decision-tree algorithm. It employs the gain-ratio criterion [**1**], because the information-gain criterion has a strong bias in favour of attribute tests with many values. To reduce the bias of the information-gain criterion, the split-information measure as defined by the following equation is employed

$$\text{Split\_Information}(S, T) = -\sum_{i=1}^{b} \frac{|S_i|}{|S|} \cdot \log_2 \left(\frac{|S_i|}{|S|} \right) \tag{5}$$

The split-information measure can be regarded as the cost of selecting a given attribute as a test. Notice that it discourages the selection of attributes with many values.

The gain ratio is then given by

$$\text{Gain\_Ratio}(T) = \frac{\text{Gain}(S, T)}{\text{Split\_Information}(S, T)} \tag{6}$$

The gain-ratio computation for a nominal attribute test is relatively straightforward. For continuous attributes, the d possible values appearing in the subset associated with an internal node are sorted. Then, all $d - 1$ possible splits on this continuous attribute are examined. The one that maximizes the gain-ratio criterion is selected as a threshold.

A decision tree generated as described above is potentially an overfitted solution, i.e. it may have components that are too specific to noise and outliers that may be present in the training data. To relax this overfitting, C4.5 uses a tree-pruning method that tries to simplify the tree by eliminating subtrees that seem too specific. Pruning is done by examining each subtree and replacing it with one of its branches or leaf nodes, if such a replacement does not degrade the accuracy of the subtree.

The C4.5 inductive-learning system can also transform the generated decision tree to a set of IF–THEN rules. For the transformation to a rule set, every path from the root of the unpruned tree to a leaf gives one initial rule, in which the left-hand side is the conjunction of all attribute-based tests established by the path and the right-hand side specifies the class predicted at the leaf. If the path to each leaf node is transformed into a production rule, the resulting collection of rules would classify examples exactly as the tree and, as a consequence of their tree origin, the rules would be mutually exclusive and hence their order would not matter. After producing a rule set from an unpruned tree, C4.5 implements a very complicated multiphase rule-pruning procedure. First, each rule is simplified by deleting some conditions based on the pessimistic-error estimate as adopted in tree pruning. Second, the set of rules is partitioned into several groups according to the rule consequent, with one group corresponding to one class. All possible subsets of rules from each group are then examined and the best subset based on the minimum description length (MDL) principle is selected. In the third stage, all the rule subsets are ordered, based on their classification error on the training data set. A default rule is then chosen whose consequent is the class that contains the largest number of training instances not covered by any rule. The pruning procedure then attempts to reduce the size of the rule set further by eliminating rules, the removal of which does not cause a deterioration in the accuracy of training-data classification.

3.2 Rule induction

As with decision-tree learning, there are many rule-induction algorithms. AQ [**14–16**], CN2 [**2**, **17**], RIPPER [**18**], SLIPPER [**19**], and RULES [**20–22**] are examples. These learning systems are categorized as 'separate-and-conquer' inductive systems.

In contrast to decision-tree learning, rule induction directly generates IF–THEN rules. Each rule can be represented in the following form: $Cond_1 \wedge \ldots \wedge Cond_i \wedge \ldots \wedge Cond_{n_c} \rightarrow C_j$, where the antecedent consists of a conjunction of conditions $Cond_i$. Each condition takes the form $[A_i = v_{ij}]$ or $[t_{i1} < A_i \leqslant t_{i2}]$ depending on the property of the attribute A_i. If A_i is a nominal attribute, v_{ij} is a valid nominal value that A_i can take. If A_i is a continuous attribute, t_{i1} and t_{i2} are two thresholds in the domain of attribute A_i. The consequent is the class to which instances satisfying the antecedent can be assigned. Figure 2 displays a rule set generated from the data set given in Table 1.

Rule-induction systems produce either an unordered set of IF–THEN rules or an ordered set of IF–THEN rules, also known as decision lists [**23**], both including a default rule. To classify an instance in the case of ordered rules, the ordered list of rules is examined to find the first whose antecedent is satisfied by the instance. The predicted class is then the one nominated by this rule. If no rule antecedent is satisfied, the instance is predicted to belong to the default class. In the case of unordered rules, it is possible for some instances to be covered by more than one rule. To classify a new instance in this case, some conflict resolution approach must be employed.

If [Pressure > 35] [Vibration = Absent] → B

If [Pressure > 35] [Vibration = Present] → A

If [Pressure ⩽ 35] [Temperature ⩽ 50] → B

If [Pressure ⩽ 35] [Temperature > 50] → A

Fig. 2 A set of rules derived from the data in Table 1

The general operation of rule-induction algorithms is the same. They induce the rule 'set one rule at a time'. After a rule is generated, the instances covered by it are removed from the training data set and the same induction procedure is applied to the remaining data set until all the instances are covered by at least one rule in the rule set.

AQ15 is a well-known inductive-learning system. It is based on the AQ algorithm as originally described in [**14**] and implements the STAR method of inductive learning [**24**]. In AQ15, decision rules are represented as expressions in the variable-valued logic system 1 (VL1). VL1 is a multiple-valued extension to propositional logic. In VL1, a *selector* relates an attribute to an attribute value or disjunct of values using one of the relational operators $<$, $\leqslant$, $=$, $!=$, $\geqslant$, or $>$. A selector or a conjunction of selectors forms a *complex*. A *cover* is a disjunction of complexes describing all positive instances and none of the negative instances of the concept. A cover defines the condition part of a corresponding decision rule. AQ15 is able to implement a form of constructive induction as well. An example of a decision rule with an internal disjunct is

$$[\text{Outlook} = \text{sunny} \vee \text{cloudy}] \wedge [\text{Temperature} > 60] \vee [\text{Wind} = \text{true}] \wedge [\text{Temperature} > 70] \rightarrow \text{class}[\text{Nice}]$$

When building a complex, AQ15 performs the general-to-specific beam-search technique to find the best complex. The algorithm considers specializations that exclude some particular covered negative instances from the complex, while ensuring some particular 'seed' positive instances remain covered,

iterating until all negative instances are excluded. As a result, AQ searches only the space of complexes that are completely consistent with the data. Seeds are selected at random and negative examples are chosen according to their distance from the seed (the nearest ones are picked first, where distance is the number of attributes with different values in the seed and negative instances).

The AQ15 system can generate unordered and ordered rules. In the case of unordered rules, a new instance is classified by finding which of the induced rules have their complexes satisfied by the instance. If the instance satisfies only one rule, then the class predicted by that rule is assigned to the instance. If the instance satisfies more than one rule, a heuristic called estimate of probability (EP) is used to predict its class. With this method, each rule is weighted by the proportion of learning instances covered by it. The weights of rules of the same class are probabilistically combined to form a weight for the entire class and the class with the highest weight is taken as the predicted class of the test example. If the instance is not covered by any rule, a heuristic called measure of fit (MF) is used. In this case the instance belongs to a part of the decision space that is not covered by any decision rule. The measure of best fit of a class can be interpreted as a combination of 'closeness' of the instances to a class and an estimate of the prior probability of the class.

The AQ15 algorithm uses a post-pruning technique to remove redundant conditions from the body of a rule and to remove unnecessary rules from the rule set. Simplification generally leads to smaller, more accurate rule sets. This framework was later generalized in the POSEIDON system [**25**]. POSEIDON can simplify a complete and consistent concept description, which has been induced by the AQ15 algorithm, by removing conditions and rules, and by contracting and extending intervals and internal disjunctions. POSEIDON successively applies the operator that results in the highest coverage gain as long as the resulting rule set increases some quality criteria.

CN2 is a rule-induction algorithm that incorporates ideas from both ID3 and AQ. The representation of decision rules in CN2 is very similar to that of AQ15 and can be viewed as a subset of VL1. The inductive-learning system CN2 was developed by Clark and Niblett [**2**, **26**] and later modified by Clark and Boswell [**17**]. The objective behind the design of CN2 was to modify the AQ algorithm by retaining its beam search through the space of complexes, but removing its dependency on specific training instances during search. While the AQ algorithm searches only the space of complexes that are completely consistent with the training data, CN2 extends its search space to rules that do not perform perfectly on the training data by broadening the specialization process to examine all specializations of a complex, in much the same way as ID3 considers all attribute tests when growing a node in a tree. A cut-off method similar to decision-tree pruning is applied to halt specialization when no further specializations are statistically significant. The modified version of CN2 produces either an ordered set of IF–THEN rules like the original CN2 version or an unordered set of IF–THEN rules.

The CN2 algorithm consists of two main procedures: a search algorithm performing a beam search for a good rule and a control algorithm for repeatedly executing the search. The control procedure of the CN2 algorithm for ordered rules iteratively calls the beam search procedure to find the best complex, until no better complexes are found. It then appends a rule to the rule set with this best complex as the condition and the most common class among the instances covered by this complex as the prediction. The instances covered by a rule are removed from the training set. The last rule in the rule list is a default rule predicting the most common class among the training examples not covered. The beam search procedure to find the best complex corresponds to the STAR procedure of the AQ algorithm. The pruned general-to-specific search retains a size-limited set or star of 'best complexes found so far'. The system examines only specializations of this set, carrying out a beam search for the space of complexes. A complex is specialized by either adding a new selector to the conjunction or by removing a disjunctive element in one of its selectors.

The CN2 algorithm can be easily modified to generate an unordered rule set by changing only the control procedure, leaving the beam search procedure unaltered (apart from the evaluation function, described below). The main modification to the algorithm is to iterate the search for each class in turn, removing only covered instances of the current class where a rule has been found. Unlike the case for ordered rules, the negative instances remain because now each rule must independently stand against all negatives. The covered positives must be removed to stop CN2 from repeatedly finding the same rule.

The CN2 algorithm employs two types of heuristics in the search for the best complexes, goodness and significance. Goodness is a measure of the quality of the complex that is used for ordering complexes that are candidates for inclusion in the final cover. Like ID3, the original CN2 version used the information-theoretic entropy measure to evaluate the quality of the complex (the lower the entropy, the better the complex). This function prefers complexes covering a large number of instances of a single class

and few examples of other classes, but it tends to select very specific rules covering only a few training instances. The modified version of CN2 employs the Laplacian error estimate instead. The expected accuracy, one minus the expected Laplacian error estimate, is given by

$$\text{Laplace\_Accuracy}(n_{\text{class}}, n_{\text{covered}}, k) = \frac{n_{\text{class}} + 1}{n_{\text{covered}} + k} \tag{7}$$

where k is the number of classes, n_{class} is the number of positive instances covered by the rule, and n_{covered} is the number of instances covered by the rule. This formula is a special case of the m-probability-estimate developed in [**27**]. This estimate avoids the downward bias of the entropy measure of favouring very specific complexes in the general-to-specific search operation.

The second evaluation function tests whether a complex is statistically significant, i.e. whether it locates a regularity that is unlikely to have occurred by chance and thus reflects a genuine correlation between attribute values and classes in the training data. To test significance, CN2 uses the likelihood ratio statistic [**28**]. This is given by

$$\text{Likelihood\_Ratio}(F, E) = 2 \cdot \sum_{i=1}^{n} f_i \cdot \log \frac{f_i}{e_i} \tag{8}$$

where the distribution $F = (f_1, f_2, \ldots, f_n)$ is the observed frequency distribution of instances among classes satisfying a given complex and $E = (e_1, e_2, \ldots, e_n)$ is the expected frequency distribution of the same number of instances under the assumption that the complex selects instances randomly from the training set. Thus the two functions, the Laplacian error estimate and statistical significance serve to determine whether complexes found during the search are both 'good' (have high accuracy when predicting the majority class covered) and 'reliable' (the high accuracy on the training data is not just due to chance).

CN2 performs another check that can be viewed as a form of pre-pruning. It checks whether the Laplace estimate of the best complex is greater than that of the default rule predicting the class for all instances. If this is not the case, then the new complex does not contribute any new information and the generation of complexes for the current class is terminated.

To apply unordered rules to classify a new instance, all rules are tried and those whose conditions are all satisfied are collected. If a clash occurs, i.e. more than one class is predicted by the collected rules, a probabilistic method is employed to resolve the clash. Each rule is tagged with the distribution of covered instances among classes and these distributions are summed to find the most probable class.

RULES (RULe Extraction System) is a set of inductive-learning algorithms that follow a similar approach to the AQ family. The first three algorithms in the RULES family of algorithms (RULES-1, 2, and 3) were developed by Pham and Aksoy [**20**, **29**, **30**]. Later, Pham and Dimov [**21**] introduced a new algorithm called RULES-3 Plus. Compared to its immediate predecessor RULES-3, RULES-3 Plus has two new strong features. First, it employs a more efficient search procedure instead of the exhaustive search conducted in RULES-3. Second, it incorporates a simple metric for selecting and sorting candidate rules according to their generality and accuracy. RULES-3 does not employ any measure for assessing the information content of rules. The first incremental-learning algorithm in the RULES family was RULES-4 [**31**]. It allows the stored knowledge to be updated and refined rapidly when new examples are available. RULES-4 employs a short term memory (STM) to store training examples when they become available. The STM has a user-specified size. When the STM is full, the RULES-3 Plus algorithm is used to generate rules. In order to increase the efficiency of the RULES family of algorithms, Pham *et al.* [**32**] used a simple clustering technique to select a good set of training examples that are representative of the overall data set. The method was tested on different problems. The results showed that when the algorithm was applied to clustered data sets, the execution time was reduced, as well as the size of the generated rule sets. Pham *et al.* [**22**] described a new algorithm, called RULES-5, which overcomes some of the deficiencies of the RULES family. In particular, RULES-5 employs a new method for handling continuous attributes and uses a simple and more efficient search method. The test results obtained with RULES-5 showed that the rule sets extracted were more accurate and compact than those obtained using its immediate predecessor RULES-3 Plus. One of the main weaknesses of the RULES-5 algorithm is its inability to handle noisy data. Pham *et al.* [**33**] proposed a new pruning technique that improved significantly the performance of the RULES-5 algorithm on data sets containing noisy examples.

4 OTHER MACHINE-LEARNING APPROACHES TO CLASSIFICATION LEARNING

Besides decision trees and rule induction, several other approaches to classification learning exist, as shown in Fig. 3. This section will briefly review those alternative approaches: instance-based learning, neural networks, genetic algorithms, and Bayesian methods.

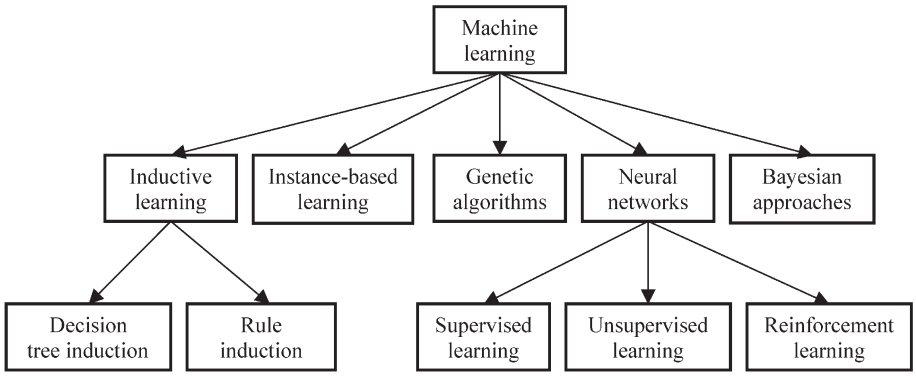

Fig. 3 A classification of the main machine-learning techniques

4.1 Instance-based learning

Instance-based learning is based upon the idea of letting the examples themselves form an implicit representation of the target concept [**3**, **4**]. In contrast to learning methods that construct a general, explicit description of the target concept when training instances are provided, instance-based learning methods, such as those using nearest-neighbour methods, simply store the training instances. Generalizing beyond these instances is postponed until a new instance must be classified. Because of this, instance-based methods are sometimes referred to as 'lazy' learning methods. A test instance is classified by finding the nearest stored instance according to some similarity function and assigning the class of the latter to the former. Advantages of instance-based methods include the ability to model complex target concepts and the fact that information present in the training instances is never lost (because the instances themselves are stored explicitly). One disadvantage of the instance-based approaches is that the cost of classifying new instances can be high. This is because nearly all the computation takes place at classification time rather than when the training instances are first encountered. Therefore, techniques for efficiently indexing training instances are a significant practical issue in reducing the computation required at classification time. A second disadvantage of many instance-based approaches, especially nearest-neighbour methods, is that they typically consider all attributes of the instances when attempting to retrieve similar training instances from the memory. If the target concept depends on only a few of the many available attributes, then the instances that are really most 'similar' may be a long distance apart.

Domingos [**34**, **35**] presented a new approach to inductive learning that combines the best features of instance-based learning and rule induction. The approach is implemented in the RISE system, which stands for 'Rule Induction from a Set of Exemplars'.

4.2 Neural networks

Neural networks provide a general practical method for learning real-valued and discrete-valued target concepts in a way that is robust to noise in the training data [**36–38**]. Neural-network learning is well-suited to problems in which the training data correspond to noisy, complex sensor data, such as inputs from cameras and microphones. Neural networks can be classified according to their mode of learning, namely, supervised, unsupervised, and reinforcement learning. The latter can be regarded as a special form of supervised learning. Instead of using a teacher to provide target outputs, a reinforcement learning algorithm employs a critic only to evaluate the goodness of the neural network output corresponding to a given input. The backpropagation algorithm is a common supervised learning method adopted for multilayer perceptrons, the most popular type of neural networks. The algorithm has been successfully applied to a variety of learning tasks, such as interpreting visual scenes, speech recognition, handwriting recognition, and robot control. One of the chief advantages of neural networks is their wide applicability; however, they also have two particular drawbacks. The first is the difficulty in understanding the models they produce. The second is the often time-consuming training. Recent years have seen much research in developing new neural-network methods that effectively address these comprehensibility and speed issues [**39–42**].

4.3 Genetic algorithms

Genetic algorithms are stochastic search techniques, which have been inspired by the process of biological evolution [**43–45**]. Several genetic algorithm-based systems for learning classification rules have been developed [**46–55**]. In these systems, rules are represented by bit strings whose particular interpretation depends on the application. The search for an appropriate rule begins with a population, or collection, of

initial rules. Members of the current population give rise to the next-generation population by means of operations such as random mutation and crossover. At each step, the rules in the current population are evaluated relative to a given measure of fitness, with the fittest rules selected probabilistically as seeds for producing the next generation. The process performs generate-and-test beam search of the rules, in which variants of the best current rules are most likely to be considered next. Genetic algorithms have been applied successfully to a variety of learning tasks and to other optimization problems. For example, they have been used to learn collections of rules for robot control, to optimize the topology and learning parameters for neural networks, and to solve job-shop scheduling problems. Genetic algorithms have a potentially greater ability to avoid local minima than is possible with the simple greedy search employed by most learning techniques. On the other hand, they have a high computational cost.

4.4 Bayesian approaches

Bayesian approaches employ probabilistic concept representations and range from a simple Bayesian classifier [**56**] to Bayesian networks, which learn the full joint probability distributions of the attributes and class, as opposed to just the class description [**57**]. Bayesian networks provide a natural platform for combining domain knowledge and empirical learning. However, inference in Bayesian networks can have a high time complexity and as tools for classification learning they are not yet as mature or well tested as other approaches. More generally, as Buntine [**58**] notes, the Bayesian paradigm extends beyond any single representation and forms a framework in which many learning tasks can be usefully studied.

5 CURRENT TRENDS IN MACHINE-LEARNING RESEARCH

Machine-learning research has been making significant progress in many directions. This section examines two of the most important directions and discusses some current problems. The two directions are scaling up of machine-learning algorithms and learning multiple models.

5.1 Scaling up machine-learning algorithms

In modern manufacturing environments vast amounts of data are being collected in database management systems and data warehouses from areas such as product and process design, production planning and control, assembly, scheduling, and maintenance. The first major research area concerns techniques for scaling up machine-learning algorithms so that they can process very large data sets efficiently, while building from them the best possible models. The recent emergence of data mining as a major application of machine-learning algorithms has underlined the need for algorithms to be able to handle large data sets that are currently beyond their scope. In data-mining applications, data sets with millions of training examples, thousands of attributes and hundreds of classes are common. Fayyad *et al.* [**8**] cited several representative examples of databases containing many gigabytes (even terabytes) of data. Designing learning algorithms appropriate for such applications has thus become an important research problem.

Many approaches have been proposed and implemented for scaling up machine-learning algorithms [**59–65**]. The most straightforward approach is to produce more efficient algorithms or increase the efficiency of the existing algorithms. This approach includes a wide variety of algorithm design techniques for optimizing the search and representation, for finding approximate instead of exact solutions, or for taking advantage of the inherent parallelism of the task. A second approach is to partition the data, avoiding the need to run algorithms on very large data sets. This approach involves breaking the data set up into subsets, learning from one or more of the subsets, and possibly combining the results. Data partitioning is useful to avoid memory management problems that occur when algorithms try to process huge data sets in the main memory. An approach orthogonal to the selection of example subsets is to select subsets of relevant features on which to focus attention.

In order to provide focus and specific details, the application of inductive learning techniques to very large data sets is now reviewed; the issues and techniques discussed generalize to other types of machine learning.

Decision-tree algorithms have been improved to handle large data sets efficiently and several new algorithms have been proposed. Catlett [**66**, **67**] proposed two methods for improving the time taken to develop a classifier. The first method used data sampling at each node of the decision tree and the second method discretized continuous attributes. These methods decrease classification time significantly but also reduce the classification accuracy. Moreover, Catlett only considered data sets that could fit in main computer memory.

Methods for partitioning the data set such that each subset fits in main memory are considered in [**68–70**]. Although these methods enable classification of large data sets, their studies show that the quality of the resulting decision tree is worse than that of a classifier that was constructed by using the

complete data set at once. Incremental-learning methods, where the data are classified in batches, have also been studied [**71**]. However, the cumulative cost of classifying data incrementally can sometimes exceed the cost of classifying the entire training set once.

The decision-tree classifier in [**72**], called SLIQ, utilized the novel techniques of pre-sorting, breadth-first growth, and MDL-based pruning to improve learning time for the classifier without loss of accuracy. At the same time, these techniques allowed classification to be performed on large amounts of disk-resident training data. However, due to the use of a memory-resident data structure that scales with the size of the training set, SLIQ has an upper limit on the number of examples it can process. Shafer *et al.* [**73**] presented a classification algorithm called SPRINT that removes all memory restrictions that limit existing decision-tree algorithms and yet exhibits the same excellent behaviour as SLIQ. SPRINT efficiently allows classification of virtually any sized data set; also, the algorithm can be easily and efficiently parallelized. However, SPRINT has been criticized for several reasons. For example, it utilizes data structures called attribute lists that can be costly to maintain, including a potential tripling of the size of the data set and an associated significant increase in scan cost [**74**]. Like C4.5, both SLIQ and SPRINT are two-stage algorithms that include building and pruning phases. Generating the decision tree in two distinct phases could result in a substantial amount of wasted effort since an entire subtree constructed in the first phase may later be pruned in the next phase. PUBLIC [**75**] is a decision-tree classifier that tightly integrates the pruning phase into the building phase instead of performing them one after the other. Its tree-growing phase is the same as that of SPRINT except that it uses entropy instead of the Gini index. However, when a leaf node is generated, PUBLIC can immediately decide whether there is a need to split it further by estimating a lower bound on the cost of coding the subtree rooted at this leaf node. The integrated approach of PUBLIC can result in substantial performance improvements compared to traditional classifiers such as SPRINT.

In recent work, Gehrke *et al.* [**76**] proposed Rainforest, a framework for developing fast and effective algorithms for constructing decision trees that gracefully adapt to the amount of main memory available. Finally, Morimoto *et al.* [**77**] developed algorithms for decision-tree construction for categorical attributes with large domains. The emphasis of this work is to improve the quality of the resulting tree.

As with decision-tree learning, there are a number of rule-induction algorithms that can scale up to large data sets. IREP [**78**] is a rule-learning algorithm that can efficiently handle large sets of noisy data. The main reason for its efficiency is the use of a technique called incremental reduced error pruning, which prunes each rule immediately after it has been induced, rather than after all rules have been generated. This speeds up the induction process as the pruned rules allow larger subsets of the remaining positive instances to be removed at each iteration compared to the non-pruned rules. Unfortunately, the accuracy of the class descriptions learned by IREP is often lower than the accuracy of those learned with the C4.5 rules algorithm [**1**], a rule-inducing variant of C4.5. Cohen [**18**] detailed several modifications to improve the accuracy of IREP, including a different rule-evaluation criterion, a different stopping criterion, and a post-processing optimization operation, producing an algorithm called RIPPER. He showed that RIPPER is competitive with C4.5 rules in terms of error rates and that it maintains the efficiency of IREP. RIPPER supports missing attributes, continuous variables, and multiple classes. This makes it applicable to a wider range of benchmark problems.

5.2 Learning multiple models

The second active research area concerns a particular method for improving accuracy in supervised learning. The term multiple models or ensemble of classifiers is used to identify a set of classifiers whose individual decisions are combined in some way (typically by voting) to classify new examples [**79**]. Ensembles have been found to be more accurate than the individual classifiers that make them up [**80–83**], and also have substantial theoretical foundations [**84–86**]. An ensemble can be more accurate than any of its component classifiers only if the individual classifiers are 'accurate' and 'diverse' [**87**]. An accurate classifier is one that performs at least better than random guessing. Two classifiers are diverse if they make different errors on new data points. The reason why ensembles improve performance can be intuitively explained in that taking a majority vote of several hypotheses reduces the random variability of individual hypotheses.

Several methods have been proposed for generating multiple classifiers using the same learning algorithm. Most of them manipulate the training set to generate multiple hypotheses. The learning algorithm runs several times, each time using a different distribution of the training instances. This technique works especially well for unstable learning algorithms – algorithms whose output classifier undergoes major changes in response to small changes in the training data.

Breiman [**88**] described a technique called *bagging* that manipulates the training data to generate

different classifiers. Bagging produces a replication of the training set by sampling with replacement from the training instances. Each replication of the training set has the same size as the original data. Some examples do not appear at all while others may appear more than once. Such a training set is called a *bootstrap replicate* of the original training set and the technique is called *bootstrap aggregating* (from which the term bagging is derived). From each replication of the training set a classifier is generated. All classifiers are used to classify each instance in the test set, usually using a uniform voting scheme where each component classifier has the same vote. Bagging methods require that the learning system should be unstable, so that small changes to the training set should lead to different classifiers. Although Breiman also notes that poor predictors can be transformed into worse ones by bagging, it is a simple and easy way to improve an existing learning method. All that is required is the addition of a preprocessor that selects the bootstrap sample and sends it to the learning algorithm and a postprocessor that does the aggregation of votes. What is lost, in comparison with decision trees and rule sets, is a simple and interpretable structure. What is gained is increased accuracy.

Freund and Schapire [**80**, **89**] presented another method for manipulating the training set called *boosting*. Instead of drawing a succession of independent bootstrap samples from the original instances, boosting maintains a weight for each instance in the training set that reflects its importance – the higher the weight the more the instance influences the learned classifier. During each iteration, the weights are adjusted in accordance with the performance of the corresponding classifier, with the result that the weight of misclassified instances is increased. Adjusting the weights causes the learner to focus on different instances, leading to different classifiers. The final classifier is constructed from the learned classifiers by a weighted voting scheme where each component classifier contributes to the final classification with a different strength based on its accuracy on the training instances that it was trained with. Like bagging, boosting depends on instability of the boosted learning system. However, it does not preclude poor predictors, provided that their error on the given distribution of instances can be kept below 50 per cent.

A third technique for constructing a good ensemble of classifiers is to manipulate the set of classes that are given to the learning algorithm. Dietterich and Bakiri [**90**] described a technique called Error-Correcting Output Coding (ECOC). This method was originally designed to handle multiclass problems by solving multiple two-class problems. ECOC represents classes with a set of output bits, where each bit encodes a binary classification function corresponding to a unique partition of the classes. Algorithms that use ECOC learn a function corresponding to each bit. All functions are then combined to generate class predictions.

Bagging, boosting, and ECOC are general combining algorithms that significantly improve classifiers such as decision trees, rule learners, or neural networks. Quinlan [**82**] conducted experiments with boosting and bagging over a diverse collection of data sets. His experiments confirmed that boosted and bagged versions of C4.5 produced noticeably more accurate classifiers than the standard version. The results also showed that boosting seemed to be more effective than bagging when applied to C4.5, although the performance of the bagged C4.5 was less variable than that of its boosted counterpart. Freund and Schapire [**80**] also applied boosting and bagging to C4.5 on 27 data sets. Their results confirmed that the error rates of boosted and bagged classifiers were significantly lower than those of single classifiers. However, they found bagging much more competitive than boosting. Bauer and Kohavi [**83**] presented an empirical comparison between boosting and bagging, and argued that both techniques, when applied to decision trees, were able to reduce the error rate at the cost of increased tree size. They also observed that boosting was not robust when dealing with noise. This is expected, because noisy examples tend to be misclassified and their weight will consequently increase. Dietterich and Bakiri [**90**] reported that ECOC improved the performance of both the C4.5 and backpropagation algorithms on a variety of different classification problems. Schapire [**91**] showed how boosting can be combined with ECOC to yield an excellent ensemble-classification method that was superior to the ECOC method.

In addition to these methods for generating ensembles using a single-learning algorithm, there are other methods that produce an ensemble by combining classifiers constructed with different-learning algorithms. When classifiers from different-learning algorithms are combined, as in stacked generalization [**92**], diversity is implied. Therefore, they only need to be checked (e.g. by cross-validation) for accuracy, with some form of weighted combination employed. This approach of generating ensembles has been shown to be effective in some applications [**93–95**].

6 APPLICATIONS OF MACHINE-LEARNING TECHNIQUES IN MANUFACTURING

This section discusses some applications of machine-learning techniques in manufacturing with special emphasis on inductive learning. The focus is on

inductive learning because detailed surveys concerning the other techniques can be found elsewhere (see, for example [**96**, **97**]).

Machine-learning techniques can be useful tools for discovering valuable patterns in data. As there are no universally applicable methods, it is important to have a clear understanding of the requirements of a particular problem and to choose the technique that best fits those requirements. In general, to be useful in a manufacturing application, a machine-learning system should have the following abilities:

(a) dealing with different types of data (numerical, nominal, text, and images);
(b) handling noise, outliers, and fuzzy data;
(c) real-time processing;
(d) dealing with large data sets and data of high dimensions;
(e) producing results that are easy to understand;
(f) being simple to implement.

Machine-learning algorithms are domain independent. In principle, they can be a very useful tool for the construction of knowledge-based systems. Efforts to apply machine-learning methods follow a standard pattern. The main stages of the process are: formulating the problem, determining the representation, collecting the training data, evaluating the learned knowledge, and fielding the knowledge base [**98–100**].

There is a wide range of manufacturing domains to which machine learning has been successfully applied. Lu and Chen [**101**] proposed an inductive-learning approach and used it to build a qualitative knowledge base utilizing results of a simulation experiment. Given a class (defined by the values of class objective parameters), inductive learning was used to derive a generalized description over the values of the control parameters. By this means the generated knowledge base could be used for deductive reasoning for control of the process.

Stirling and Buntime [**102**] employed inductive learning to investigate process-planning decision-making problems. They adopted a combination of induction and interviews with experts to acquire knowledge about processing routes through a steel mill. Although the experts were reasonably articulate, considerable time and effort was saved by using rule induction as a tool to assist the experts in formalizing and structuring their knowledge.

Inductive-learning methods have helped engineers summarize large amounts of data to support decision making. An inductive-learning algorithm was used to analyse turning-process simulation data in support of machine operation planning in manufacturing [**103**].

Lu *et al.* [**104**] reported a knowledge-processing methodology that combines the power of simulation and optimization from engineering with inductive learning. The inductive-learning approach was integrated with multi-objective optimization to form a system that provides flexible support to engineers during both the model formation and utilization phases.

In production-operations management, it is increasingly important to develop automated scheduling systems as the production stages become more complicated. Learning-based scheduling, which involves automatic acquisition of dispatching rules, is one of the practical methods used to solve this problem. There have been various attempts to use learning in scheduling problems [**105–109**]. Proposed methods for acquisition of scheduling rules by using inductive-learning techniques were applied to flow-shop scheduling problems [**110–115**], job-shop scheduling problems [**116–121**], and flexible manufacturing systems scheduling problems [**122–129**]. Experimental results indicated that effective scheduling can be achieved by applying the proposed methods.

El Attar and Hamery [**130**] designed an inductive-learning system using the ID3 algorithm that enabled knowledge to be integrated automatically into an expert system to guide the repair of helicopter blades.

Machine-learning techniques have been applied to just-in-time (JIT) production systems. First, neural networks and decision trees were used to set the number of *kanbans* in a dynamic JIT factory [**131–134**]. The number of kanbans is important to the effective operation of a JIT production system. Results showed that neural networks and decision trees represent two practical approaches with special capabilities for dynamically adjusting the number of kanbans. Second, an approach based on inductive learning was used to predict JIT factory performance from past data that includes both good and poor factory performance [**135**]. In particular, the CART decision-tree classifier was employed to generate rules automatically in a JIT manufacturing environment. The rules obtained can accurately classify and predict factory performance based on shop factors and can identify the important relationships between the shop factors that determine factory performance. Results indicated that inductive learning is a feasible technique for predicting JIT factory performance from dynamic shop-floor data.

Reich [**136**, **137**] addressed the critical role of machine-learning modelling in engineering design. He showed that quality engineering designs can be achieved through the information modelling capabilities of machine-learning programs.

Chatterjee *et al.* [**138**] discussed the application of machine-learning techniques to the problem of metal etching. They used three different techniques to analyse, classify, and predict the quality of metal etchings. The techniques adopted were neural

networks, inductive learning, and case-based reasoning, a form of instance-based learning. The authors concluded that machine-learning techniques with their ability to analyse and automate the analysis of large amounts of complex data in various aspects of manufacturing help to reduce cycle time and scrap, and improve resource utilization.

Operating and maintaining modern sensor-equipped systems, such as aircraft, generate vast amounts of numeric and symbolic data. Létourneau *et al.* [**139**] developed an approach that uses these data to build models for predicting when to replace various aircraft components before they fail. Three different machine-learning techniques were used to infer the desired models: inductive learning, instance-based learning, and Bayesian learning.

Semiconductor manufacturing is a complex operation that involves monitoring a large number of parameters from the early stages of production to the packaging of the end product. Improving the quality of the manufacturing process requires a great deal of data analysis work and is still accomplished by human engineers. The transaction volume in a typical wafer fabrication plant is as many as one million wafers a day [**140**]. The amounts of data involved make the data analysis task extremely time-consuming and difficult. Several authors proposed procedures for using machine-learning techniques in semiconductor manufacturing [**141–147**]. Research results showed that machine-learning techniques can be powerful tools for continuous quality improvement in a large and complex process such as semiconductor manufacturing.

Zhou *et al.* [**148**] developed an intelligent data-mining system and applied it to drop-testing analysis of portable electronic products to discover useful design knowledge. The rule-induction method adopted is based on the C4.5 algorithm. Studies with the proposed system indicated that its approach is flexible and can be applied to a number of other design and manufacturing processes to reduce costs and improve productivity.

Aksoy *et al.* [**149**] presented an industrial visual inspection system that can be used to carry out quality control for mass production. The system employs the RULES-3 inductive-learning algorithm.

Peng [**150**] developed a fuzzy inductive-learning-based intelligent-monitoring system for improving the reliability of manufacturing processes. The method was successfully applied to diagnosing the conditions of tapping processes to ensure the quality of products.

Pham *et al.* [**151**] described an application of data mining and machine-learning techniques in a steel bar manufacturing company. The application covered five areas, namely, make-to-order (MTO) versus make-to-stock (MTS) analysis, product sale profile analysis, rogue order determination, overdue order analysis, and product combination analysis. The results demonstrated that the techniques employed can extract information to help intelligent decision making in industry.

Other applications include generating decision rules for the conceptual design of steel members under bending [**152**], diagnosing engine faults [**153**], detecting defects in disk drive manufacturing [**154**], diagnosing motor pumps [**155**], analysing non-destructive testing of spotweld quality [**156**], managing and controlling the procurement of raw materials [**157**], accelerating rotogravure printing [**158**, **159**], optimizing processes in electrochemical machining [**160**], selecting appropriate cutting tools in grinding [**161**], determining suitable cutting conditions in operation planning [**162**, **163**], re-formulating and generalizing the machining knowledge from a machining database [**164**], choosing sheet metal working conditions [**165**], discovering the laws governing metallic behaviour [**166**], modelling job complexity in clothing production systems [**167**], acquiring and refining operational knowledge in industrial processes [**168**, **169**], and identifying arbitrary geometric and manufacturing categories in CAD databases [**170**].

From the above-mentioned problems, features of successful applications of machine learning in manufacturing can be summarized as follows:

(a) the problem is of a sufficient degree of complexity;
(b) the problem can be formulated to match existing learning algorithms;
(c) ample training data are available in an appropriate format;
(d) the training data are representative;
(e) the data are free from noise or can be cleansed cost effectively;
(f) methods for learning and evaluating performance are suitable for the application under consideration.

7 CONCLUSIONS AND FUTURE WORK

Machine-learning algorithms are demonstrably of significant value in a variety of real-world manufacturing applications. Dozens of companies around the world now provide commercial implementations of these algorithms (see www.kdnuggets.com), along with efficient links to commercial databases and well-designed user interfaces [**171**]. However, these algorithms also have some limitations. With the techniques described here, data sets having tens of thousands of training instances can be mined in relatively short times. However, many important data sets are significantly larger. To provide efficient machine-learning methods for such large data sets

requires additional research. Another problem is that most machine-learning techniques to date handle only data with continuous and nominal values. A topic of considerable research interest is the development of algorithms that can learn regularities in other data types, such as text and images. Most of the existing machine-learning methods for generating multiple models can improve significantly on the accuracy of single models, but at the expense of comprehensibility. An important line of research, therefore, is to find ways to obtain ensemble classifiers that are easy to interpret. Examples of work in this area can be found in [**19**, **172**]. Finally, although machine-learning techniques can overcome a wide range of problems in manufacturing, they are still not applied on a large scale. Given these challenges, further developments in machine-learning research can be expected in the future.

ACKNOWLEDGEMENTS

This work was carried out within the ERDF (Objective One) projects 'Innovation in Manufacturing', 'Innovative Technologies for Effective Enterprises', and 'Supporting Innovative Product Engineering and Responsive Manufacturing' (SUPERMAN), and within the project 'Innovative Production Machines and Systems' (I*PROMS).

REFERENCES

1 **Quinlan, J. R.** *C4.5: Programs for Machine Learning*, 1993 (Morgan Kaufmann, San Mateo, CA).
2 **Clark, P.** and **Niblett, T.** The CN2 induction algorithm. *Mach. Learn.*, 1989, **3**(4), 261–284.
3 **Aha, D., Kibler, D.,** and **Albert, M.** Instance-based learning algorithms. *Mach. Learn.*, 1991, **6**(1), 37–66.
4 **Aha, D.** Special issue in lazy learning. *Artif. Intell. Rev.*, 1997, 7–10.
5 **Chauvin, Y.** and **Rumelhart, D.** *Backprobagation: Theory, Architecture, and Applications*, 1995 (Lawrence Erlbaum Associates, Hillsdale, N.J.).
6 **Freitas, A. A.** *Data mining and knowledge discovery with evolutionary algorithms*, 2002 (Springer-Verlag, Berlin and New York).
7 **Heckerman, D., Geiger, D.,** and **Chickering, D.** Learning Baysian networks: the combination of knowledge and statistical data. *Mach. Learn.*, 1995, **20**(3), 197–243.
8 **Fayyad, U. M., Piatetsky-Shapiro, G.,** and **Smyth, P.** From data mining to knowledge discovery: an overview. In *Advances in Knowledge Discovery and Data Mining* (Eds U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy) 1996, pp. 1–36 (AAAI Press, Menlo Park, CA).
9 **Breiman, L., Friedman, J. H., Olshen, R. A.,** and **Stone, C. J.** *Classification and Regression Trees*, 1984 (Wadsworth, Belmont).
10 **Quinlan, J. R.** Learning efficient classification procedures and their application to chess endgames. In *Machine Learning: An Artificial Intelligence Approach* (Eds R. S. Michalski, J. G. Carbonell, and T. M. Mitchell), vol. I, 1983, pp. 463–482 (Tioga Publishing Co., Palo Alto, CA).
11 **Quinlan, J. R.** Induction of decision trees. *Mach. Learn.*, 1986, **1**, 81–106.
12 **ISL.** *Clementine Data Mining Package*, 1998 (SPSS UK Ltd, 1st Floor, St Andrew's House, West Street, Woking, Surrey GU21 1EB, UK).
13 **RuleQuest.** *Data Mining Tools C5 0.* (Pty Ltd, 30 Athena Avenue, St Ives, NSW 2075, Australia). Available from: http://www.rulequest.com/see5-info.html [accessed: 1 February 2003].
14 **Michalski, R. S.** On the quasi-minimal solution of the general covering problem. In Proceedings of the 5th International Symposium on *Information Processing (FCIP 69)*, Bled, Yugoslavia, 1969, A3 (Switching Circuits), 125–128.
15 **Michalski, R. S., Mozetic, I., Hong, J.,** and **Lavrac, N.** The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In Proceedings of the 5th National Conference on *Artificial Intelligence*, Philadelphia, PA, 1986, pp. 1041–1045 (Morgan Kaufmann, San Fransisco, CA).
16 **Michalski, R. S.** and **Kaufman, K. A.** The AQ19 system for machine learning and pattern discovery: a general description and user guide. *Reports of the Machine Learning and Inference Laboratory*, MLI 01-2, George Mason University, Fairfax, VA, USA, 2001.
17 **Clark, P.** and **Boswell, R.** Rule induction with CN2: some recent improvements. In Proceedings of the 5th European Conference on *Artificial Intelligence*, Porto, Portugal, 1991, 151–163 (Springer-Verlag, New York).
18 **Cohen, W. W.** Fast effective rule induction. In Proceedings of the 12th International Conference on *Machine Learning*, Tahoe City, CA, 1995, 115–123 (Morgan Kaufmann, San Fransisco, CA).
19 **Cohen, W. W.** and **Singer, Y.** A simple, fast and effective rule learner. In Proceedings of the 16th National Conference on *Artificial Intelligence*, Orlando, Florida, 1999, 335–342 (AAAI/MIT Press, Menlo Park, CA).
20 **Pham, D. T.** and **Aksoy, M. S.** An algorithm for automatic rule induction. In *Artificial Intelligence in Engineering*, 1993, **8**(4), pp. 227–282 (Elsevier Science Limited).
21 **Pham, D. T.** and **Dimov, S. S.** An efficient algorithm for automatic knowledge acquisition. *Pattern Recognit.*, 1997, **30**(7), 1137–1143.
22 **Pham, D. T., Bigot, S.,** and **Dimov, S. S.** RULES-5: A rule induction algorithm for problems involving continuous attributes. *Proc. Inst. Mech. Engs, Part C: J. Mechanical Engineering Science*, 2003, **217**(12), 1273–1286.
23 **Rivest, R.** Learning decision lists. *Mach. Learn.*, 1987, **2**, 229–246.
24 **Michalski, R. S.** and **Larson, J. B.** Incremental generation of VL1 hypotheses: the underlying methodology and the descriptions of program AQ11. *ISG 83-5*, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 1983.

25 **Bergadano, F., Matwin, S., Michalski, R. S.,** and **Zhang, J.** Learning tow-tiered descriptions of flexible concepts: the POSEIDON system. *Mach. Learn.*, 1992, **8**, 5–43.

26 **Clark, P.** and **Niblett T.** Induction in noisy domains. In Proceedings of the 2nd European Working Session on *Learning*, Bled, Yugoslavia, 1987, 11–30 (Sigma, Wilmslow, UK).

27 **Cestnik, B.** Estimating probabilities: a crucial task in machine learning. In Proceedings of the 3rd European Conference on *Artificial Intelligence*, Stockholm, Sweden, 1990, 147–149, ECAI-90 (Pitman, London).

28 **Kalbfleish, J.** *Probability and Statistical Inference*, vol. 2, 1979 (Springer-Verlag, New York).

29 **Pham, D. T.** and **Aksoy, M. S.** A new algorithm for inductive learning. *J. Syst. Engng*, 1995, **5**, 115–122.

30 **Pham, D. T.** and **Aksoy, M. S.** RULES: a simple rule extraction system. *Expert Syst. Appl.*, 1995, **8**(1), 59–65.

31 **Pham, D. T.** and **Dimov, S. S.** An algorithm for incremental inductive learning. *Proc. Inst. Mech. Engrs, Part B: J. Engineering Manufacture*, 1997, **211**, 239–249.

32 **Pham, D. T., Dimov, S. S.,** and **Salem, Z.** Technique for selecting examples in inductive learning. In Esit 2000 European Symposium on *Intelligent Techniques*, Erudit, Aachen, Germany, 2000, 119–127.

33 **Pham, D. T., Bigot, S.,** and **Dimov, S. S.** RULES-5: A rule merging technique for handling noise in inductive learning. *Proc. Inst. Mech. Engrs, Part C: J. Mechanical Engineering Science*, 2004, **218**(C10), 1255–1268.

34 **Domingos, P.** Rule induction and instance-based learning: a unified approach. In Proceedings of the 14th International Joint Conference on *Artificial Intelligence*, Montréal, Canada, 1995, 1226–1232 (Morgan Kaufmann, San Fransisco, CA).

35 **Domingos, P.** *A Unified Approach to Concept Learning*, PhD Thesis, University of California, Irvine, CA, 1997.

36 **Haykin, S. S.** *Neural Networks: a Comprehensive Foundation*, 1994 (Maxwell Macmillan International, New York and Macmillan College Publishing, Oxford).

37 **Mitchell, T. M.** *Machine Learning*, 1997 (McGraw Hill, New York).

38 **Pham, D. T.** and **Liu, X.** *Neural Networks for Identification, Prediction and Control*, 1999 (Springer-Verlag, London).

39 **Towell, G. G.** and **Shavlik, J. W.** The extraction of refined rules from knowledge-based neural networks. *Mach. Learn.*, 1993, **13**(1), 71–101.

40 **Craven, M. W.** and **Shavlik, J. W.** Using neural networks for data mining. *Future Generation Comput. Syst.*, 1997, **13**, 211–229.

41 **Zhou, Z. H., Jiang, Y.,** and **Chen, S. F.** A general neural framework for classification rule mining. *Int. J. Comput., Syst. Signals*, 2000, **1**(2), 154–168.

42 **Jiang, Y., Zhou, Z. H.,** and **Chen, Z. Q.** Rule learning based on neural network ensemble. In Proceedings of the International Joint Conference on *Neural Networks*, Honolulu, HI, 2002, 1416–1420.

43 **Goldberg, D. E.** *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1989 (Addison-Wesley, Reading, MA).

44 **Michalewicz, Z.** *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd Edition, 1996 (Springer-Verlag, Berlin).

45 **Pham, D. T.** and **Karaboga, D.** *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, 2000 (Springer-Verlag, London and Heidelberg).

46 **De Jong, K. A., Spears, W. M.,** and **Gordon, F. D.** Using genetic algorithms for concept learning. *Mach. Learn.*, 1993, **13**, 161–188.

47 **Greene, D. P.** and **Smith, S. F.** Competition-based induction of decision models from examples. *Mach. Learn.*, 1993, **13**, 229–257.

48 **Janikow, C. Z.** A knowledge-intensive genetic algorithm for supervised learning. *Mach. Learn.*, 1993, **13**, 189–228.

49 **Venturini, G.** SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In Proceedings of the European Conference on *Machine Learning*, Vienna, Austria, 1993, 280–296 (Springer-Verlag, Berlin).

50 **Giordana, A.** and **Saitta, L.** Learning disjunctive concepts by means of genetic algorithms. In Proceedings of the 11th International Conference on *Machine Learning*, Catania, Italy, 1994, 96–104 (Morgan Kaufmann, San Fransisco, CA).

51 **Giordana, A.** and **Neri, F.** Search-intensive concept induction. *Evol. Comput.*, 1995, **3**(4), 375–416.

52 **Neri, F.** and **Saitta, L.** Exploring the power of genetic search in learning symbolic classifiers. *IEEE Trans. Pattern Analysis Mach. Intell.*, 1996, **18**, 1135–1142.

53 **Freitas, A. A.** A genetic algorithm for generalised rule induction. In *Advances in Soft Computing – Engineering Design and Manufacturing* (Eds R. Roy, T. Funhashi, and P. K. Chawdhry), 1999, pp. 340–353 (Springer-Verlag, London).

54 **Noda, E., Freitas, A. A.,** and **Lopes, H. S.** Discovering interesting prediction rules with a genetic algorithm. In Proceedings of the 1999 Congress on *Evolutionary Computation*, Washington D.C., USA, 1999, 1322–1329 (Piscataway, NJ).

55 **Liu, J.** and **Kwok, J. T.** An extended genetic rule induction algorithm. In Proceedings of the 2000 Congress on *Evolutionary Computation*, California, USA, 2000, 458–463 (Piscataway, NJ).

56 **Domingos, P.** and **Pazzani, M.** Beyond independence: conditions for the optimality of the simple bayesian classifier. In Proceedings of the 13th International Conference on *Machine Learning*, Bari, Italy, 1996, 105–112 (Morgan Kaufmann, San Fransisco, CA).

57 **Heckerman, D.** Bayesian networks for knowledge discovery. In *Advances in Knowledge Discovery and Data Mining* (Eds U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy), 1996, pp. 273–305 (AAAI Press, Menlo Park, CA).

58 **Buntine, W.** *A Theory of Learning Classification Rules*, PhD Thesis, School of Computer Science, University of Technology, Sydney, Australia, 1991.

59 **Dash, M.** and **Liu, H.** Feature selection for classification. *Intell. Data Analysis*, 1997, **1**, 131–156.

60 **Fürnkranz, J.** Integrative windowing. *J. Artific. Intell. Res.*, 1998, **8**, 129–164.

61 **Liu, H.** and **Setiono, R.** Some issues on scalable feature selection. *Expert Syst. Appl.*, 1998, **15**, 333–339.

62 **Moore, A.** and **Lee, M.** Cached sufficient statistics for efficient machine learning with large datasets. *J. Artific. Intell. Res.*, 1998, **8**, 67–91.

63 **Zaki, M.** *Scalable Data Mining for Rules*, PhD Thesis, Department of Computer Science, University of Rochester, Rochester, NY, 1998.

64 **Opitz, D. W.** Feature selection for ensembles. In Proceedings of the 16th National Conference on *Artificial Intelligence*, Orlando, FL, 1999, 379–384 (AAAI Press, Menlo Park, CA).

65 **Provost, F.** and **Kolluri, V.** A survey of methods for scaling up inductive algorithms. *Data Mining Knowl. Discov.*, 1999, **2**, 1–42.

66 **Catlett, J.** *Megainduction: Machine Learning on Very Large Databases*, PhD Thesis, School of Computer Science, University of Technology, Sydney, Australia, 1991.

67 **Catlett, J.** On changing continuous attributes into ordered discrete attributes. In Proceedings of the 1991 European Working Session on *Learning*, Porto, Portugal, 1991, 164–178 (Springer-Verlag, Berlin).

68 **Chan, P.** and **Stolfo, S.** Toward parallel and distributed learning by meta-learning. In *Working Notes of AAAI Workshop on Knowledge Discovery in Databases*, Washington, DC, 1993, pp. 227–240, (AAAI Press, Manlo Park, CA).

69 **Chan, P.** and **Stolfo, S.** On the accuracy of meta-learning for scalable data mining. *J. Intell. Inf. Syst.*, 1997, **8**, 5–28.

70 **Zhang, S.** and **Wu, X.** Large scale data mining based on data partitioning. *Appl. Artific. Intell.*, 2001, **15**, 129–139.

71 **Wu, X.** and **Lo, W. H.** Multi-layer incremental induction. In Proceedings of the 5th Pacific Rim International Conference on *Artificial Intelligence*, Singapore, 1998, 24–32 (Springer-Verlag).

72 **Mehta, M., Agrawal, R.,** and **Rissanen, J.** SLIQ: A fast scalable classifier for data mining. In Proceedings of the 5th International Conference on *Extending Database Technology*, Avignon, France, 1996, 18–32 (Springer-Verlag).

73 **Shafer, J., Agrawal, R.,** and **Mehta, M.** SPRINT: A scalable parallel classifier for data mining. In Proceedings of the 22nd International Conference on *Very Large Data Bases (VLDB)*, Mumbai (Bombay), India, 1996, 544–555.

74 **Graefe, G., Fayyad, U.,** and **Chaudhuri, S.** On the efficient gathering of sufficient statistics for classification of large SQL databases. In Proceedings of the 4th International Conference on *Knowledge Discovery and Data Mining*, New York, USA, 1998, 204–208 (AAAI Press, Menlo Park, CA).

75 **Rastogi, R.** and **Shim, K.** PUBLIC: A decision tree classifier that integrates building and pruning. In Proceedings of the 24th International Conference on *Very Large Data Bases (VLDB)*, New York, USA, 1998, 404–415 (Morgan Kaufmann, San Fransisco, CA).

76 **Gehrke, J., Ramakrishnan, R.,** and **Ganti, V.** Rainforest – A framework for fast decision tree construction of large datasets. In Proceedings of the 24th International Conference on *Very Large Data Bases (VLDB)*, New York, USA, 1998, 416–427 (Morgan Kaufmann, San Fransisco, CA).

77 **Morimoto, Y., Fukuda, T., Matsuzawa, H., Tokuyama, T.,** and **Yoda, K.** Algorithms for mining association rules for binary segmentations of huge categorical databases. In Proceedings of the 24th International Conference on *Very Large Data Bases (VLDB)*, New York, USA, 1998, 380–391 (Morgan Kaufmann, San Fransisco, CA).

78 **Fürnkranz, J.** and **Widmer, G.** Incremental reduced error pruning. In Proceedings of the 11th International Conference on *Machine Learning*, New Brunswick, NJ, 1994, 70–77 (Morgan Kaufmann, San Fransisco, CA).

79 **Dietterich, T. G.** Machine learning research: four current directions. *Artific. Intell. Mag.*, 1997, **18**(4), 97–136.

80 **Freund, Y.** and **Schapire, R. E.** Experiments with a new boosting algorithm. In Proceedings of the 13th International Conference on *Machine Learning*, Bari, Italy, 1996, 148–156 (Morgan Kaufmann, San Fransisco, CA).

81 **Pham, D. T.** and **Oztemel, E.** *Intelligent Quality Systems*, 1996 (Springer-Verlag, London, Berlin, and Heidelberg).

82 **Quinlan, J. R.** Bagging, boosting and C4.5. In Proceedings of the 13th National Conference on *Artificial Intelligence*, Portland Oregon, 1996, pp. 725–730 (AAAI Press and MIT Press, Menlo Park, CA).

83 **Bauer, E.** and **Kohavi, R.** An empirical comparison of voting classification algorithms: bagging, boosting and variants. *Mach. Learn.*, 1999, **36**, 105–139.

84 **Friedman, J. H.** On bias, variance, 0/1-loss, and the curse-of-dimensionality. Technical report, Department of Statistics and Stanford Linear Accelerator Center, Stanford University, Stanford, CA, 1996.

85 **Madigan, D., Raftery, A. E., Volinsky, C. T.,** and **Hoeting, J. A.** Bayesian model averaging. In Proceedings of the AAAI-96 Workshop on *Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, Portland, OR, 1996, 77–83 (AAAI Press, Menlo Park, CA).

86 **Schapire, R., Freund, Y., Bartlett, P.,** and **Lee, W. S.** Boosting the margin: A new explanation for the effectiveness of voting methods. In Proceedings of the 14th International Conference on *Machine Learning*, Nashville, Tennessee, 1997, 322–330 (Morgan Kaufmann, San Fransisco, CA).

87 **Hansen, L. K.** and **Salamon, P.** Neural networks ensembles. *IEEE Trans. Pattern Analysis Mach. Intell.*, 1990, **12**(10), 993–1001.

88 **Breiman, L.** Bagging predictors. *Mach. Learn.*, 1996, **24**(2), 123–140.

89 **Freund, Y.** and **Schapire, R. E.** A decision-theoretic generalisation of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 1997, **55**(1), 119–139.

90 **Dietterich, T. G.** and **Bakiri, G.** Solving multi-class problems via error-correcting output codes. *J. Artific. Intell. Res.*, 1995, **2**, 263–286.

91 **Schapire, R.** Using output codes to boost multiclass learning problems. In Proceedings of the 14th International Conference on *Machine Learning*, Nashville, Tennessee, 1997, 313–321 (Morgan Kaufmann, San Fransisco, CA).

92 **Wolpert, D.** Stacked generalization. *Neural Networks*, 1992, **5**, 241–259.

93 **Zhang, X., Mesirov, J.,** and **Waltz, D.** A hybrid system for protein secondary structure prediction. *J. Molecular Biology*, 1992, **225**, 1049–1063.

94 **Breiman, L.** Stacked regressions. *Mach. Learn.*, 1996, **24**(1), 49–64.

95 **Ting, K. M.** and **Witten, I. H.** Stacked generalization: when does it work? In Proceedings of the 15th International Joint Conference on *Artificial Intelligence*, Nagoya, Japan, 1996, 250–265 (Morgan Kaufmann, San Fransisco, CA).

96 **Monostori, L.** AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing. *Engng Appl. Artific. Intell.*, 2003, **16**(3), 227–291.

97 **Monostori, L., Márkus, A., Van Brussel, H.,** and **Westkämper, E.** Machine learning approaches to manufacturing. *CIRP Annals*, 1996, **45**(2), 675–712.

98 **Langley, P.** and **Simon, H. A.** Applications of machine learning and rule induction. *Commun. ACM*, 1995, **38**(11), 55–64.

99 **Braha, D.** *Data Mining for Design and Manufacturing: Methods and Applications*, 2001 (Kluwer Academic Publishers, Boston, MA).

100 **Lavrač, N., Motoda, H., Fawcett, T., Holte, R., Langley, P.,** and **Adriaans, P.** Introduction: lessons learned from data mining applications and collaborative problem solving. *Mach. Learn.*, 2004, **57**, 13–34.

101 **Lu, S. C.-Y.** and **Chen, K.** A machine learning approach to the automatic synthesis of mechanistic knowledge for engineering decision making. *Artif. Intell. Engng Des. Anal. and Mfg* 1987, **1**(2), 109–118.

102 **Stirling, D.** and **Buntine, W.** Process routing in a steel mill: A challenging induction problem. In *Artificial Intelligence Developments and Applications* (Eds J. S. Gero and R. Stanton), 1988, pp. 301–313 (Elsevier Science Publisher B.V., North-Holland).

103 **Whitehall, B. L., Lu, S. C.-Y.,** and **Stepp, R. E.** CAQ: a machine learning tool for engineering. *Artif. Intell. Engng*, 1990, **5**(4), 189–198.

104 **Lu, S. C.-Y., Tcheng, D. K.,** and **Yerramareddy, S.** Integration of simulation, learning and optimisation to support engineering design. *Annals CIRP*, 1993, **38**(1), 143–146.

105 **Aytug, H., Bhattacharyya, S., Koehler, G. J.,** and **Snowdon, J. L.** A review of machine learning in scheduling. *IEEE Trans. Engng Mgmt*, 1994, **41**(2), 165–171.

106 **Aytug, H., Koehler, G. J.,** and **Snowdon, J. L.** Genetic learning of dynamic scheduling within a simulation environment. *Comput. Op. Res.*, 1994, **21**, 909–925.

107 **Jain, A. S.** and **Meeran, S.** Job-shop scheduling using neural networks. *Int. J. Prod. Res.*, 1998, **36**(5), 1249–1272.

108 **Priore, P., Fuente, D., Gomez, A.,** and **Puente, J.** Dynamic scheduling of manufacturing systems with machine learning. *Int. J. Found. Comput. Sci.*, 2001, **12**(6), 751–762.

109 **Akyol, D. E.** Application of neural networks to heuristic scheduling algorithms. *Comp. Indust. Engng*, 2004, **46**, 679–696.

110 **Pierreval, H.** and **Ralambondrainy, H.** A simulation and learning technique for generating knowledge about manufacturing systems behaviour. *J. Op. Res. Soc.*, 1990, **41**(6), 461–474.

111 **Nakasuka, S.** and **Yoshida, T.** Dynamic scheduling system utilising machine learning as a knowledge acquisition tool. *Int. J. Prod. Res.*, 1992, **30**(2), 411–431.

112 **Suwa, H., Fujii, S.,** and **Morita, H.** Heuristic rule acquisition for flow-shop scheduling. In Proceedings of the 1996 Japan–USA Symposium on *Flexible Automation*, Kobe, Japan, 1996, 1345–1351.

113 **Suwa, H., Fujii, S.,** and **Morita, H.** An acquisition of scheduling rules for flow-shop problems. In Proceedings of the 6th IFIP TCS/WG5.7 International Conference on *Advances in Production Management System*, Japan, 1996, 631–636.

114 **Koonce, D. A., Fang, C.,** and **Tsai, S.** A data mining tool for learning from manufacturing systems. *Comput. Indust. Engng*, 1997, **33**(1–2), 27–30.

115 **Murata, T., Sugimoto, T., Tsujimura, T.,** and **Gen, M.** Rule conversion in knowledge acquisition for flow shop scheduling problems. In Proceedings of the Joint 9th International Fuzzy Systems Association (IFSA) World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference, Vancouver, Canada, 2001, **4**, 2417–2421.

116 **Quiroga, L. A.** and **Rabelo, L. C.** Learning from examples: a review of machine learning, neural networks and fuzzy logic paradigms. *Comput. Indust. Engng*, 1995, **29**(1–4), 561–565.

117 **Lee, C.-Y., Piramuthu, S.,** and **Tsai, Y.-K.** Job shop scheduling with a genetic algorithm and machine learning. *Int. J. Prod. Res.*, 1997, **35**(4), 1171–1191.

118 **Suwa, H., Fujii, S.,** and **Morita, H.** Acquisition of scheduling rules for job-shop problems. IFAC/IFIP Conference on *Management and Control of Production and Logistics*, Campinas, Brazil, 1997, 353–358.

119 **Suwa, H., Fujii, S.,** and **Morita, H.** Acquisition and refinement of scheduling rules for job-shop problems. In Proceedings of the IEEE International Symposium on *Industrial Electronics*, 1998, **2**, 720–725.

120 **Ben-Arieh, D., Chopra, M.,** and **Bleyberg, M. Z.** Data mining applications for real-time distributed shop floor control. In Proceedings of the IEEE International Conference on *Systems, Man, and Cybernetics*, San Diego, CA, 1998, 2738–2743.

121 **Osisek, V.** and **Aytug, H.** Discovering subproblem prioritization rules for shifting bottleneck algorithms. *J. Intell. Mfg*, 2004, **15**, 55–67.

122 **Shaw, M. J., Park, S.,** and **Raman, N.** Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge. *IEE Trans.*, 1992, **24**(2), 156–168.

123 **Piramuthu, S., Raman, N., Shaw, M. J.,** and **Park, S.** Integration of simulation modeling and inductive learning in and adaptive decision support system. *Decis. Supp. Syst.*, 1993, **9**, 127–142.

124 **Li, D. C.** and **She, I. S.** Using unsupervised learning technologies to induce scheduling knowledge for FMSs. *Int. J. Prod. Res.*, 1994, **32**(9), 2187–2199.

125 **Piramuthu, S., Raman, N.,** and **Shaw, M. J.** Learning-based scheduling in a flexible manufacturing flow line. *IEEE Trans. Engng Mgmt*, 1994, **41**(2), 172–182.

126 **Park, S. C., Raman, N.,** and **Shaw, M. J.** Adaptive scheduling in dynamic flexible manufacturing systems: a dynamic rule selection approach. *IEEE Trans. Robot. Automat.*, 1997, **13**(4), 486–502.

127 **Kim, C. O., Min, Y. D.,** and **Yih, Y.** Integration of inductive learning and neural networks for multi-objective FMS scheduling. *Int. J. Prod. Res.*, 1998, **36**(9), 2497–2509.

128 **Morello, B. C., Michaut, D.,** and **Baptiste, P.** A knowledge discovery process for a flexible manufacturing system. In Proceedings of the 8th IEEE International Conference on *Emerging Technologies and Factory Automation*, Nice, France, 2001, **1**, 651–658.

129 **Priore, P., Fuente, D., Pino, R.,** and **Puente, J.** Dynamic scheduling of manufacturing systems using neural networks and inductive learning. *Integr. Mfg Syst.*, 2003, **14**(2), 160–168.

130 **El Attar, M.** and **Hamery, X.** Industrial expert system acquired by machine learning. *Applic. Artif. Intell.*, 1994, **8**(4), 497–542.

131 **Rakes, T. R., Rees, L., Siochi, F. C.,** and **Wray, B. A.** Estimating the number of kanban using neural networks. *Adv. Artif. Intell. Econom. Finance Mgmt*, 1994, **1**, 125–139.

132 **Wray, B. A., Rakes, T. R.,** and **Rees, L.** Neural network identification of critical factors in dynamic just-in-time kanban environment. *J. Intell. Mfg*, 1997, **8**, 83–96.

133 **Markham, I. S., Mathieu, R. G.,** and **Wray, B. A.** A rule induction approach for determining the number of kanbans in a just-in-time production system. *Comput. Indust. Engng*, 1998, **34**(4), 717–727.

134 **Markham, I. S., Mathieu, R. G.,** and **Wray, B. A.** Kanban setting through artificial intelligence: a comparative study of artificial neural networks and decision trees. *Integ. Mfg Syst.*, 2000, **11**(4), 239–246.

135 **Mathieu, R. G., Wray, B. A.,** and **Markham, I. S.** An approach to learning from both good and poor factory performance in a kanban-based just-in-time production system. *Prod. Plan. Control*, 2002, **13**(8), 715–724.

136 **Reich, Y.** Measuring the value of knowledge. *Int. J. Human-Comp. Stud.*, 1995, **42**(1), 3–30.

137 **Reich, Y.** Modeling engineering information with machine learning. *Artif. Intell. Engng Des., Analysis Mfg*, 1996, **10**(2), 171–174.

138 **Chatterjee, A., Croley, D., Ramamurti, V.,** and **Chang, K.-Y.** Application of machine learning to manufacturing: results from metal etch. In Proceedings of the 19th IEEE/CPMT International Electronics Manufacturing Technology Symposium, Austin, Texas, 1996, pp. 372–377.

139 **Létourneau, S., Famili, F.,** and **Matwin, S.** Data mining to predict aircraft component replacement. *IEEE Intell. Syst.*, 1999, **14**, 59–66.

140 **Shin, C. K.** and **Park, S. C.** A machine learning approach to yield management in semiconductor manufacturing. *Int. J. Prod. Res.*, 2000, **38**(17), 4261–4271.

141 **Perez, R. A., Hall, L. O., Romaniuk, S.,** and **Lilkendey, J. T.** Inductive learning for expert systems in manufacturing. In Proceedings of the 25th Hawaii International Conference on *System Sciences*, Hawaii, 1992, **3**, 14–25.

142 **Irani, K. B., Jie, C., Fayyad, U. M.,** and **Zhaogang, Q.** Applying machine learning to semiconductor manufacturing. *IEEE Expert*, 1993, **8**(1), 41–47.

143 **Saxena, S.** Fault isolation during semiconductor manufacturing using automated discovery from wafer tracking databases. In Proceedings of the Knowledge Discovery in Databases Workshop, 1993, 81–88.

144 **Khera, D., Cresswell, M. W., Linholm, L. W., Ramanathan, G., Buzzeo, J.,** and **Nagarajan, A.** Increasing profitability and improving semiconductor manufacturing throughput using expert systems. *IEEE Trans. Engng Mgmt*, 1994, **41**(2), 143–151.

145 **Turney, P.** Data engineering for the analysis of semiconductor manufacturing data. In Proceedings of the International Joint Conference on *Artificial Intelligence (IJCAI-95) Workshop on Data Engineering for Inductive Learning*, Montreal, Quebec, 1995, 50–59.

146 **Kang, B. S., Lee, J. H., Shin, C. K., Yu, S. J.,** and **Park, S. C.** Hybrid machine learning system for integrated yield management in semiconductor manufacturing. *Expert Syst. Applic.*, 1998, **15**, 123–132.

147 **Gardner, M.** and **Bieker, J.** Data mining solves tough semiconductor manufacturing problems. In Proceedings of the 6th ACM SIGKDD International Conference on *Knowledge Discovery and Data Mining*, Boston, MA, 2000, 376–383 (ACM Press, New York).

148 **Zhou, C., Nelson, P. C., Xiao, W., Tirpak, T. M.,** and **Lane, S. A.** An intelligent data mining system for drop test analysis of electronic products. *IEEE Trans. Electr. Packag. Mfg*, 2001, **24**(3), 222–231.

149 **Aksoy, M. S., Torkul, O.,** and **Cedimoglu, I. H.** An industrial visual inspection system that uses inductive learning. *J. Intell. Mfg*, 2004, **15**, 569–574.

150 **Peng, Y.** Intelligent condition monitoring using fuzzy inductive learning. *J. Intell. Mfg*, 2004, **15**, 373–380.

151 **Pham, D. T., Packianather, M. S., Dimov, S., Soroka, A. J., Girard, T., Bigot, S.,** and **Salem, Z.** An application of data mining and machine learning techniques in the metal industry. In Proceedings of the 4th CIRP International Seminar on *Intelligent Computation in Manufacturing Engineering (ICME-04)*, Sorrento (Naples), Italy, 2004.

152 **Forsyth, R.** *Machine Learning Principles and Techniques*, 1989 (Chapman and Hall, London).

153 **Ke, M.** and **Ali, M.** A learning representation and diagnostic methodology for engine fault diagnosis. In Proceedings of the 2nd International Conference on *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Tullahoma, Tennessee, TN, 1989, **2**, 824–830 (ACM Press, New York).

154 **Apté, C., Weiss, S.,** and **Grout, G.** Predicting defects in disk drive manufacturing: a case study in high-dimensional classification. In Proceedings of the 9th IEEE Conference on *Artificial Intelligence Applications*, New York, 1993, 212–218, CAIA-93.

155 **Giordana, A., Neri, F.,** and **Saitta, L.** Automated learning for industrial diagnosis. In Proceedings of the 3rd International Workshop on *Multistrategy Learning*, Harpers Ferry, WV, 1996, 125–134 (AAAI Press, Menlo Park, CA).

156 **Ercil, A.** Classification trees prove useful in nondestructive testing of spotweld quality. *Welding J.*, 1993, **72**(9), 59.

157 **Das, S. K.** and **Bhambri, S.** A decision tree approach for selecting between demand based, reorder and JIT/Kanban methods for material procurement. *Prod. Plan. Control*, 1994, **5**(4), 342–352.

158 **Evans, B.** and **Fisher, D.** Overcoming process delays with decision tree induction. *IEEE Expert*, 1994, **9**(1), 60–66.

159 **Evans, B.** and **Fisher, D.** Using decision tree induction to minimize process delays in printing industry. In *Handbook of Data Mining and Knowledge Discovery* (Eds W. Klösgen and J. M. Zytkow), 2002 (Oxford University Press).

160 **Famili, A.** Use of decision tree induction for process optimization and knowledge refinement of an industrial process. *Artif. Intell. Engng Des. Analysis Mfg (AIEDAM)*, 1994, **8**(1), 63–75.

161 **Filipič, B.** and **Junkar, M.** Inductive learning approach to the selection of tools in grinding process. In Proceedings of the 2nd International Conference on *Artificial Intelligence Applications*, 1994, 513–520.

162 **Park, M. W., Rho, H. M.,** and **Park, B. T.** Generation of modified cutting conditions using neural networks for an operation planning system. *Annals CIRP*, 1996, **45**(1), 475–478.

163 **Schultz, G., Fichtner, D., Nestler, A.,** and **Hoffmann, J.** An intelligent tool for determination of cutting values based on neural networks. In Proceedings of the 2nd World Congress on *Intelligent Manufacturing Processes and Systems,* Budapest, Hungary, 1997, 66–71.

164 **Sluga, A., Jermol, M., Zupanič, D.,** and **Mladenić, D.** Machine learning approach to machinability analysis. *Comp. Indust.*, 1998, **37**, 185–196.

165 **Lin, Z.-C.** and **Chang, D.-Y.** Application of a neural network machine learning model in the selection system of sheet metal bending tooling. *Artif. Intell. Engng*, 1996, **10**, 21–37.

166 **Mitchell, F., Sleeman, D., Duffy, J. A., Ingram, M. D.,** and **Young, R. W.** Optical basicity of metallurgical slags: a new computer-based system for data visualisation and analysis. *Ironmaking and Steelmaking*, 1997, **24**, 306–320.

167 **Hui, P. C. L., Chan, K. C. K.,** and **Yeung, K. W.** Modelling job complexity in garment manufacturing by inductive learning. *Int. J. Cloth. Sci. Technol.*, 1997, **9**(1), 34–44.

168 **Shigaki, I.** and **Narazaki, H.** A machine-learning approach for a sintering process using a neural network. *Prod. Plan. Control*, 1999, **10**(8), 727–734.

169 **Shigaki, I.** and **Narazaki, H.** An approximate summarization method of process data for acquiring knowledge to improve product quality. *Prod. Plan. Control*, 2001, **12**(4), 379–387.

170 **Ip, C. Y., Regli, W. C., Sieger, L.,** and **Shokoufandeh, A.** Automated learning of model classification. In Proceedings of the 8th ACM Symposium on *Solid Modeling and Applications,* Seattle, WA, 2003, 322–327 (ACM Press, New York).

171 **Mitchell, T. M.** Machine learning and data mining. *Commun. ACM*, 1999, **42**(11), 31–36.

172 **Domingos, P.** Knowledge discovery via multiple models. *Intell. Data Analysis*, 1998, **2**(1–4), 187–202.

APPENDIX

Notation

A_i	the ith attribute in an example
b	the number of partitions resulting from the test T
C_j	the jth class
$Cond_i$	the ith condition in a given rule
d	the number of distinct values for a continuous attribute A_i
E	the expected frequency distribution of instances
ECOC	Error-Correcting Output Coding
EP	Estimate of Probability
F	the observed frequency distribution of instances among classes satisfying a given complex
JIT	just-in-time
k	the number of classes in a data set
MDL	Minimum Description Length
MF	Measure of Fit
MTO	make-to-order
MTS	make-to-stock
n_{A_i}	the number of possible values for a nominal attribute A_i
n_{c}	the number of conditions in a rule
n_{class}	the number of positive instances covered by a given rule
n_{covered}	the total number of instances covered by a given rule
$P(C_j, S)$	the proportion of instances in S which are in class C_j
S	a set of instances
$\|S\|$	the number of instances in S
S_i	the ith partitions of the data set S
STM	Short Term Memory
t_{ij}	the jth threshold value (cutting point) in the domain of a continuous attribute A_i
T	a test for partitioning S at a decision-tree node
v_{ij}	the jth value for a nominal attribute A_i