Machine Learning and Statistics

The Interface

Edited by

G. NAKHAEIZADEH
C. C. TAYLOR

A Wiley-Interscience Publication
JOHN WILEY & SONS, INC.
New York - Chichester - Brisbane - Toronto - Singapore - Weinheim

CHAPTER 4

A Multistrategy Approach to Learning Multiple Dependent Concepts

D. MALERBA
G. SEMERARO
F. ESPOSITO

1. INTRODUCTION

In concept learning from examples, it is instructive to distinguish the situation in which only one concept has to be induced (*single-concept learning*) from that in which there are several concepts to be learned (*multiple-concept learning*). In the former case, the learner is provided with examples (positive instances) and optionally counterexamples (negative instances) of the single concept and produces a generalization that is complete (all positive instances are explained by means of the generalization) and possibly consistent (negative instances, if any, cannot be explained). Many learning systems presented in the literature can actually learn only one concept at a time. Some of the most renowned are Vere's Thoth (Vere, 1975), Mitchell's candidate-elimination algorithm (Mitchell, 1977), Stepp's AQ7UNI (Stepp, 1979), and the recent FOIL system developed by Quinlan (1990a).

Few researchers have taken an interest in the problem of learning many concepts. Indeed, it is commonly believed that the task of learning m distinct concepts can simply be cast as m single concept learning tasks. No attention is paid to the sequential order in which concepts have to be considered, and sometimes it is claimed that all concepts could be learned in parallel by assigning each single-concept learning task to a distinct processor. Actually, this view of the matter is adequate only if we assume that concepts are mutually independent (*independence assumption*). Concept independence entails the independence of the m single-concept learning problems, in which case

Machine Learning and Statistics, The Interface, Edited by G. Nakhaeizadeh and C. C. Taylor.
ISBN 0-471-14890-3

there would be no reason to worry about the order in which concepts are learned.

Even though in many applications such an assumption is reasonable, it cannot be accepted as a general rule. In many other cases, it is possible to facilitate the learning task by exploiting information about the mutual relationships between concepts. In fact, when the independence assumption is dropped, the learner is required to hypothesize the description of each single concept, as well as the dependencies between concepts.

In this chapter we present problems that occur when the independence assumption is made, such as a decrease in predictive accuracy and an increase in the computational complexity of the learning task. We also point out some new issues that the consideration of concept dependencies can raise, namely, the definition of the correct order in which concepts have to be learned, the adoption of the notion of extensional coverage, and the control of mutual recursion. For the first of them, we propose a multistrategy approach in which statistical techniques are profitably exploited in order to infer concept dependencies before starting the learning process of each single concept. Knowledge on dependencies can subsequently be used by the learner that can perform the opportune shift of language before trying to learn each single concept. Since the appropriateness of the statistical techniques depends on the power of the representation language adopted, we have organized our presentation into two main sections: the next is devoted to attribute-based domains, and the third section is dedicated to structural domains. Some positive results obtained in a real-world problem with a first-order learning system, called INDUBI/CSL, are also illustrated.

2. MULTIPLE-CONCEPT LEARNING IN ATTRIBUTE-BASED DOMAINS

Traces of the independence assumption can already be found in some of the early studies that represented milestones for work on learning from examples in attribute-based domains. For instance, given a set of hypotheses, $V = \{V_i\}$, $i = 1, \ldots, m$, and a family of facts $F = \{F_i\}$, which are only partially described by the hypotheses, Michalski and Larson (1978, p. 2) define the learning problem as the production of a new set of hypotheses, $V^1 = \{V_i^1\}$, where each V_i^1 describes all the facts in the set F_i, and does *not* describe facts in any other set F_j, $j \neq i$. In this statement of the problem, there is an implicit assumption that each example must belong to only one class, that is, that classes are mutually exclusive. Since concepts are intensional descriptions of classes, this means that the possibility of dependence among concepts is excluded a priori. Quinlan (1986, p. 86) is more explicit when he establishes that each object in the universe belongs to one of a set of mutually exclusive classes. For Rendell (1986, p. 181), a hypothesis could be an assertion involving any number of classes, but it is always possible to consider one dichotomy at a time. For example, a universe of visual grids could be categorized into 26 letters plus a

nonsense class, but instead each letter can be learned *separately* by performing induction 26 times. Once again, there is no space for expressing dependencies among concepts.

Generally, the learning problem for attribute-based domains is stated as follows:

Given a set of examples (or observations) described by means of a feature vector:

$$\langle a_1, a_2, \ldots, a_n, c \rangle$$

Find a hypothesis for predicting the value of c from the values of $a_1, a_2, \ldots, a_n$.

The attributes a_i are *known* attributes, while c is the only attribute to be predicted (*target* attribute) and represents the class of each example. In a more general case, however, there are several target attributes and each observation is described as a feature vector:

$$\langle a_1, a_2, \ldots, a_n, c_1, c_2, \ldots, c_m \rangle,$$

where $c_1, c_2, \ldots, c_m$ are attributes to be predicted. This is the case in which we need to learn several concepts of the same domain (Datta and Kibler, 1993). For instance, the horse-colic database, which is available in the machine learning repository at the University of California, Irvine, collects data concerning horses hospitalized at Guelph (Canada) that suffer from "colicky" symptoms (McLeish and Cecile, 1990). For each case, there are at least four variables to be predicted, namely:

1. $c_1 \equiv V23$: outcome (lived, died, was euthanized);
2. $c_2 \equiv V24$: surgical lesion? (yes, no);
3. $c_3 \equiv V25$: site of lesion (gastric, sm intestine, lg colon, ...); and
4. $c_4 \equiv V26$: type of lesion (simple, strangulation, inflammation, other).

Note that $c_1, c_2, \ldots, c_m$ do not represent mutually exclusive classes in which the universe of observations is partitioned, but conversely, each of them defines a distinct partitioning. Therefore, it is important to take into account possible dependencies among target attributes, since they can help to produce simpler and more accurate and comprehensible hypotheses. In the example above, the prediction of the type and site of lesion can help to establish the outcome and whether the problem is surgical or not. Another instance is that given in Figure 4.1. In this case there are only two known attributes and three target attributes, each of which defines a distinct partitioning of the feature space. If the learner had no possibility of defining piecewise region boundaries in its language of

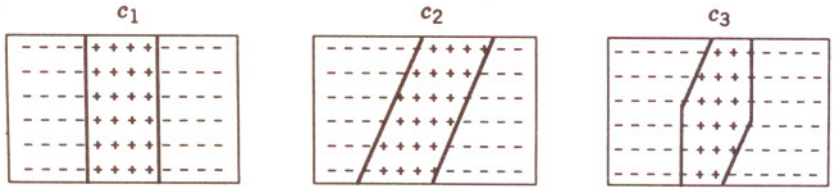


Figure 4.1. *Concept c_3 can easily be recognized when concepts c_1 and c_2 have already been learned.*

hypotheses, then it would not be able to find any generalization for c_3. On the contrary, by taking concept dependencies into consideration, the system could easily learn the concepts c_1 and c_2, and could then express c_3 as a logical *and* of two tests on c_1 and c_2: $c_3 = dc_1 \wedge c_2$.

2.1. The Problem of the Order

The main issue raised by the abolition of the independence assumption is the need to define the order in which concepts should be learned, since the definition of a concept may depend on the definition of other previously learned concepts. In particular, if a target attribute c_i is relevant for the target attribute c_j, whereas all known attributes are not, then it is extremely important to learn c_i first, and then to use it as a known attribute while learning c_j. Wrong ordering may also affect the learnability of some target concepts. Basically we can say that the definition of an order in which concepts are to be learned is the main aspect of learning several dependent concepts. But how is it possible to find the right order?

In some application domains, the user already knows which are the possible dependencies among concepts to be learned. For instance, in the case of the horse-colic database, it is not difficult to establish that c_1 and c_2 depend on c_3 and c_4. When concept dependencies are acyclic, it is possible to use directed acyclic graphs, called *dependency graphs*, to represent them (see Figure 4.2). In dependency graphs a node represents a concept to be learned, while an arrow from c_i to c_j indicates that c_j depends on c_i. For each concept c_i, it is possible to associate a level according to the following criterion: if c_i depends on $\{c_{i_1}, c_{i_2}, \ldots, c_{i_m}\}$, then $level(c_i) = 1 + max\{level(c_{i_1}), level(c_{i_2}), \ldots, level(c_{i_m})\}$. By definition, if c_i does not depend on any concept, then $level(c_i) = 0$. The order in which concepts should be learned is fully defined by the dependency graph. In particular, concepts at the zeroth level, called *minimally dependent concepts* (Malerba, 1993) or *golden point* (Baroglio et al., 1992), have to be learned first, since they do not depend on any other concept. Concepts at the i-th level ($i > 0$) will be learned only after all concepts at the j-th level, with $j < i$, have been learned, since concepts at a lower level must be considered as known attributes/predicates while learning concepts at the i-th level. In other words,

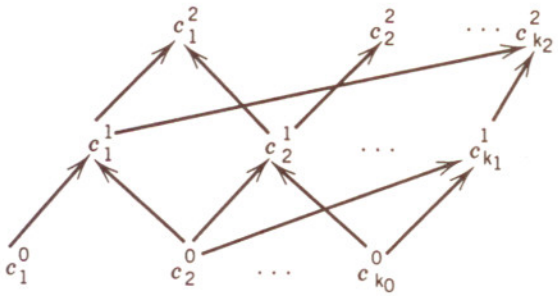


Figure 4.2. *An example of a dependency graph. The upper index denotes the level of each concept.*

the order in which concepts have to be learned defines the appropriate shift of language that is necessary while learning multiple concepts.

When the user does not know the dependencies among concepts, it is possible to infer the graph before learning the single concepts. The approach followed to discover the graph is inherently multistrategic. Statistics provide several tools for studying probabilistic (in)dependencies between variables, the best known of them being the χ^2 test for contingency tables. Given two variables, say c_i and c_j, then the independence hypothesis:

$$H_0: P(c_i \cap c_j) = P(c_i) \cdot P(c_j)$$

can be tested by comparing the expected cell frequencies to the actual values according to the following formula:

$$\chi^2 = \Sigma_k[(f_o^k - f_e^k)^2 / f_e^k]$$

where f_o^k equals the observed frequency in each cell, and f_e^k equals the expected frequency (Hogg and Tanis, 1977). By rejecting the null hypothesis, however, we can only conclude that there is a systematic relationship between c_i and c_j, without being able to indicate a *direction* of dependence. For this purpose, several asymmetric *measures of association* have been proposed in statistics.

Lambda (λ) is a measure of association for *nominal* variables and gives the percentage of improvement in our ability to predict the value of the dependent variable once we know the value of the independent variable (Mueller et al., 1970). This is based on the assumption that the best strategy for prediction is to select the value of the dependent variable that covers most cases, since this choice will minimize the number of wrong guesses. Given two concepts, say c_i and c_j, we have:

$$0 \leqslant \lambda(c_i, c_j) \neq \lambda(c_j, c_i) \leqslant 1$$

A value of zero means no improvement in predicting one concept given the other one, while the value 1.0 occurs when one concept can be deterministically predicted from the other one. Thus we can choose the dependency direction $c_i \rightarrow c_j$ if $\lambda(c_i, c_j) > \lambda(c_j, c_i)$.

Piatetsky-Shapiro (1991) proposed a different measure for the dependency-analysis of nominal values, called *probabilistic dependency* (*pdep*). If $c_i^{(k)}$ denotes the value taken by the target attribute c_i in the k-th observation, then $pdep(c_i, c_j)$ is the conditional probability that c_j will take on the same value in two observations in which c_i is the same, $P(c_j^{(k)} = c_j^{(l)} \mid c_i^{(k)} = c_i^{(l)})$. Normalization of *pdep* using a proportional reduction in variation results in Goodman and Kruskal's τ measure of association (1972), which is always between 0 and 1. Once again, we choose the dependence direction $c_i \rightarrow c_j$ if $\tau(c_i, c_j) > \tau(c_j, c_i)$. Finally, Somer's D is an asymmetric measure of association suitable for ordinal variables (Gibbons, 1993). It is based on the computation of the number of concordant and discordant pairs. Given two pairs of observations, $(c_i^{(k)}, c_i^{(l)})$ and $(c_j^{(k)}, c_j^{(l)})$, they are *concordant* if the relative ordering of $c_i^{(k)}$ and $c_i^{(l)}$ is the same as the relative ordering of $c_j^{(k)}$ and $c_j^{(l)}$; otherwise they are discordant. Somer's D assumes a value in the interval $[-1, 1]$, where the absolute value is proportional to the strength of the correlation between the two variables, while the sign indicates whether the correlation is positive (the dependent variable grows with the independent variable), or negative.

It is worthwhile noting that probabilistic measures of association discover simple (in)dependencies between *pairs* of variables. Nevertheless, the dependency graph can be simplified when *conditional* independencies between variables are considered. For instance, if $c_1 \rightarrow c_2 \rightarrow c_3$ is the true underlying dependency graph of three concepts, then non-conditional tests of independence for some probabilistic measure of association might also find a direct dependency between c_1 and c_3. Several studies have been performed on the problem of inferring probabilistic dependency graphs from data. In particular, Cooper and Herskovits (1992) proposed a Bayesian method for constructing *Bayesian belief-networks* (Neapolitan, 1990; Pearl, 1988) from a database. A Bayesian belief-network is a directed acyclic graph in which nodes represent domain variables (i.e., concepts, in our case) and arcs between nodes represent probabilistic dependencies.

When concept dependencies express causal influence, it is possible to exploit other statistical methods for discovering causal models from empirical data, such as Pearl and Verma's Inductive-Causation Algorithm (Malerba et al., 1994; Pearl and Verma, 1991). Basically, a *causal model* is a directed acyclic graph whose nodes are variables and whose edges denote direct causal reationships between pairs of variables. Causal relationships can be *potential* or *genuine*, according to whether we have sufficient (control) knowledge to be able to reject the hypothesis of a *spurious* association. In the latter case, correlation between two variables can only be explained by postulating the existence of a common latent cause. Finally, it is possible to have *indefinite* associations, for which it is not even possible to hypothesize the existence of a

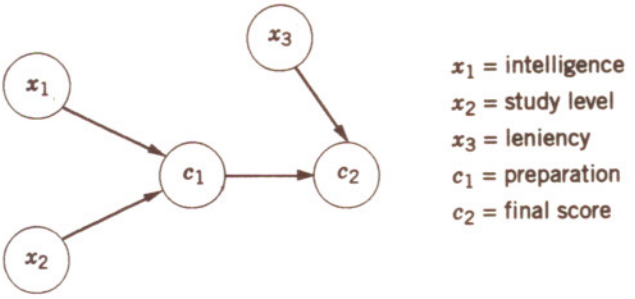


Figure 4.3. *An example of causal model.*

common latent cause. Pearl and Verma provide a theory of causal inference in which they show how it is possible to discriminate among these kinds of relationships on the basis of the conditional independencies between variables. Henceforth, the independence of two variables, say x_1 and x_2, given a set S of variables, will be denoted as $I(x_1, x_2 \mid S)$. The set S is said to be the *context* in which the independence holds. To show how causal models discovered by means of the Inductive-Causation Algorithm can be exploited in multiple-concept learning, we generated a sample of 1,000 observations concerning the model in Figure 4.3.

Each attribute has an ordinal domain with three possible values: low (L), medium (M), and high (H). Two of them are target attributes, while the other three are known. We built a contingency table for each pair of variables to test the independence in the empty context, and we used a χ^2 test on each table. For testing conditional independence with non-empty contexts we had to generate as many subtables as the number of possible values that variables in the context could take. At the significance level $\alpha = 0.05$ we detected the following independencies:

$$I(x_1, x_2 \mid \varnothing) \qquad I(x_1, x_3 \mid \varnothing) \qquad I(x_1, c_2 \mid \{c_1\})$$
$$I(x_2, x_3 \mid \varnothing) \qquad I(x_2, c_2 \mid \{c_1\}) \qquad I(x_3, c_1 \mid \varnothing)$$

The Inductive-Causation Algorithm builds the original model but with the difference that all the causal relationships except $c_1 \rightarrow c_2$ are potential and not genuine.

Having discovered the causal dependencies between the variables, we used a learning system to induce the causal rules from the data. In particular, two learning problems were defined: the first for learning the causal law that relates the study level and the intelligence of a student with his/her preparation, and the second for learning how the preparation and leniency of the examiner affect the final result of the examination. In order to solve both problems we used

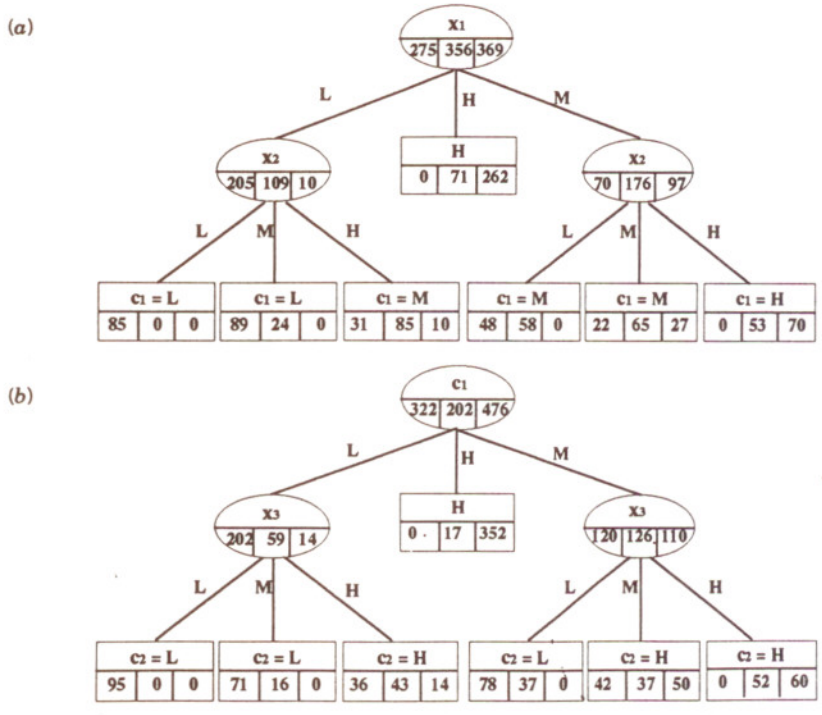


Figure 4.4. *The two decision trees generated by C4.5 for the causal model in Figure 4.3. Here, leaves are represented by boxes. Triplets of numbers in each node represent the distribution of examples reaching the node with respect to the values taken by the target attribute, namely L, M and H.*

C4.5 (Quinlan, 1993), a learning system that induces decision trees from examples. In particular, in the first problem, examples are described by means of x_1, x_2, and the class (target) attribute c_1, while, in the second problem, examples are described by means of c_1, x_3, and the class (target) attribute c_2. The results are shown in Figure 4.4. The composition of the training cases at a leaf F gives a probability $P(K \mid F)$ that a case at F belongs to class K, where the probability can be estimated as a relative frequency (Quinlan, 1990b). For instance, in the case of a moderately lenient examiner and an insufficiently prepared student, the probability of having a low score is $71/(71 + 16) = 0.82$, while the probability of having a medium score is $16/(71 + 16) = 0.18$. From tree b) in Figure 4.4, we can also draw the conclusion that even students with an insufficient or moderately good preparation can get a high score if the examiner is particularly lenient. These and other rules can be directly derived in an explicit form by transforming each decision tree into a set of production rules. Classification rules whose condition parts contain target concepts are called *contextual rules* (Malerba, 1993).

3. MULTIPLE-CONCEPT LEARNING IN STRUCTURAL DOMAINS

Issues raised by the independence assumption are even more evident when we consider a more powerful representation language, such as first-order predicate logic, which allows us to represent structural descriptions. In this case, each concept can be represented by means of a single predicate, hence the name *mutiple predicate learning* (DeRaedt et al., 1993). In addition to the problem of the order observed for attribute-based representations, there are other difficulties, caused by the adoption of a more powerful language, that concern the particular model of generalization employed by the learning system and the generation of mutually recursive rules.

3.1. The Issues of Extensional Coverage and Mutual Recursion

Several systems that learn first-order logic theories expressed as sets of Horn clauses resort to the *syntactic* notion of *extensional coverage*, in order to define the properties of completeness and consistency for a hypothesis H. Let E be the set of positive (E^+) and negative (E^-) instances of one or more target predicates $p_1, p_2, \ldots, p_n$, that is, $E = E^+ \cup E^-$. Then H *extensionally covers* an example $e \in E$, if and only if there exist a clause $c \in H$ and a substitution θ, such that the head of c matches the head of e ($head(c)\theta = head(e)$) and the body of c θ-subsumes the body of e ($body(c)\theta \subseteq$ body(e)) (Plotkin, 1970). Such a syntactic notion of coverage is adopted in many learning systems, since it is more mechanizable than the *semantic* notion of intensional coverage: Given a background knowledge B, which possibly includes previously learned hypotheses, we say that H and B intensionally cover an example $e \in E$, if and only if $B \cup H \cup body(e)$ logically entails $head(e)$ ($B \cup H \cup body(e) \models head(e)$). In fact, the implementation of the notion of intensional coverage requires a theorem prover, while a pattern matcher is enough for the extensional coverage.

The adoption of the notion of extensional coverage to check whether a hypothesis H is complete and consistent can lead to a number of problems, especially when E does not include all positive and negative instances of p_1, $p_2, \ldots, p_n$, for a given set of constant symbols. In fact, we can have hypotheses that are extensionally but not intensionally consistent, as well as hypotheses that are intensionally but not extensionally complete. Note that the notion of extensional coverage causes troubles in the case of single predicates to be learned and there is therefore all the more reason for believing that it could be a source of trouble in the case of learning many predicates. Indeed, when there are several concepts to be learned, it is also possible to have hypotheses that are extensionally but not intensionally complete. This problem occurs when the generated hypotheses are mutually recursive and causes a non-terminating computation.

Mutual recursion is another issue, which is independent from the adopted notion of coverage. In fact, it requires a change in the solutions for the ordering problem, since the definition of a concept c_i depends on the definition of

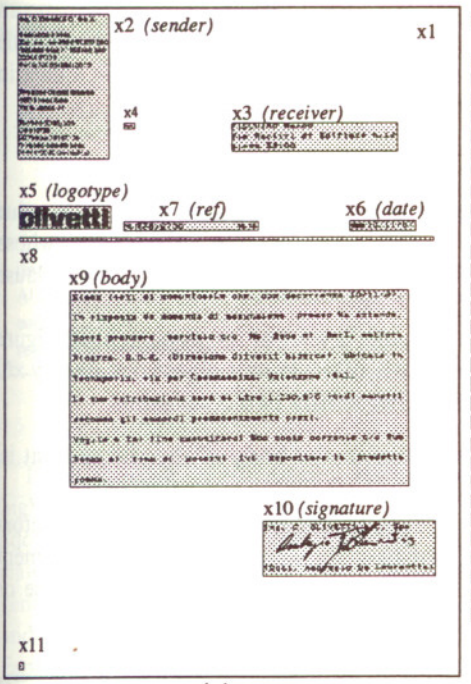
another concept c_j, which, in its turn, depends on the very concept c_i that cannot be completely defined. In this case, the dependency graph is cyclic, and we have to consider the possibility of learning single clauses instead of complete predicate definitions before moving on to a dependent concept. Obviously, the order in which single clauses are learned markedly affects the final result.

3.2. Inferring Concept Dependencies in Structural Domains

Finding the right order in which concepts have to be learned is a more difficult task when we move on to structural domains. This is due to several reasons. First, statistical methods for testing independence are suitable for attribute-based representations and not for structural representations. Second, first-order learning systems adopt stronger language/search biases than zeroth-order learners, thus the right choice of the order in which concepts should be learned is much more critical in the former systems than in the latter. Third, it is possible to represent mutually recursive definitions, which makes dependency graphs no longer acyclic. Here, we do not present any general solution to the problem of inferring the right order while learning multiple concepts in structural domains, but we illustrate an *ad hoc* solution that seems to work well on those images analysis and interpretation problems in which both symbolic and numeric descriptions are available. We have successfully tried this approach in the domains of cartographic map interpretation (Barbanente et al., 1992) and document understanding. The latter is described below.

3.2.1. The Problem of Document Understanding. According to international standards (Horak, 1985), any document is characterized by two different structures representing its spatial organization and its content: the *layout* and the *logical* structure. The former concerns the *organization* of the document on the presentation medium. Each page layout can be decomposed into a set of layout objects that are rectangular areas, each of which is associated with a portion of content. Ideally, each layout component should contain only text or graphics, but sometimes it can contain both, owing to the difficulties of automatically separating text from graphics in a document bitmap image. Conversely, the logical structure is concerned with "meaning" of some layout objects, such as sender or receiver in a business letter (see Figure 4.5a). The set of possible "meanings" associated with the layout objects of a document is named *logical classes*, while the layout objects associated with a logical class are called *logical objects*. The term *document understanding* denotes the process of identification of logical objects in a document image.

In our previous works (Esposito et al., 1993a, b), we investigated the possibility of using machine learning techniques for document understanding. More precisely, given a set of page layouts of single-page documents belonging to the same class (e.g., business letters, scientific papers, or magazine indexes) and assuming that the user-trainer has already labeled some layout objects




```
[logic_type(x6)=date] ←
[part_of(x1,x2)]
[part_of(x1,x3)]
[part_of(x1,x4)]
[part_of(x1,x5)]
[part_of(x1,x6)]
[part_of(x1,x7)]
[part_of(x1,x8)]
[part_of(x1,x9)]
[part_of(x1,x10)]
[part_of(x1,x11)]
[position(x2)=north_west]
[position(x3)=north]
[position(x4)=north_west]
[position(x5)=north_west]
[position(x6)=north_east]
[position(x7)=north]
[position(x8)=center]
[position(x9)=center]
[position(x10)=south_east]
[position(x11)=south_west]
[width(x2)=medium]
[width(x3)=medium_large]
[width(x4)=smallest]
[width(x5)=medium]
[width(x6)=medium_small]
[width(x7)=medium_large]
[width(x8)=very_very_large]
[width(x9)=very_very_large]
[width(x10)=medium_large]
[width(x11)=smallest]
[height(x2)=medium_large]
[height(x3)=small]
[height(x4)=smallest]
[height(x5)=very_small]
[height(x6)=very_very_small]
[height(x7)=very_very_small]
[height(x8)=smallest]
[height(x9)=large]
[height(x10)=medium_small]
[height(x11)=smallest]
[type(x2)=text]
[type(x3)=text]
[type(x4)=text]
[type(x5)=picture]
[type(x6)=text]
[type(x7)=text]
[type(x8)=text]
[type(x9)=text]
[type(x10)=mixture]
[type(x11)=text]
[on_top(x5,x8)]
[on_top(x6,x8)]
[on_top(x7,x8)]
[on_top(x9,x10)]
[to_right(x2,x4)]
[to_right(x5,x7)]
[aligned(x2,x5)=both_columns]
[aligned(x5,x7)=ending_row]
[aligned(x4,x7)=starting_col]
[aligned(x7,x6)=both_rows]
[aligned(x8,x9)=ending_col]
[aligned(x4,x3)=starting_row]
[aligned(x8,x11)=starting_col]
```

(b)

Figure 4.5. *a) Page layout of an Olivetti business letter. b) A VL_{21} definite clause describing an instance of the logical class* date.

according to their meanings, the problem is that of learning rules that allow logical components to be correctly identified on the basis only of layout information. The training instances are symbolic descriptions of the page layout of each document, together with information on the logical class of some layout objects. Note that concepts to be learned may be related to each other in some way (e.g., the receiver is reported above the date), thus we should learn contextual rules in order to benefit from concept dependencies. For this purpose, we adopted a first-order learning system, called INDUBI/CSL (Contextual Supervised Learner) developed at the University of Bari, Italy.

3.2.2. INDUBI/CSL. INDUBI/CSL, an extension of INDUBI and INDUBI/H (Esposito et al., 1994a), can learn contextual rules expressed as VL_{21} linked clauses, which have the same expressive power as Horn clauses. In VL_{21} (Michalski, 1980), the equivalent of an atom in predicate logic is the notion of *selector* or *relational statement*, which is written as:

$$[L = R]$$

where L, named *referee*, is a function symbol with its *arguments*, and R, named *reference*, is a set of values in the range of the function. A range can be an unordered (nominal), totally ordered (linear), or partially-ordered set. The semantics of a selector is the following: it is true if L takes one of the values in R. A VL_{21} *definite clause* is an expression of the kind:

$$[f(t_1, t_2, \ldots, t_n) = \lambda] \leftarrow \phi(t_1, t_2, \ldots, t_m)$$

where f is a function symbol, t_i's may be constants or variables, λ is a value in the range of f, and $\phi(t_1, t_2, \ldots, t_m)$ is a conjunction of selectors whose arguments contain the terms $t_1, t_2, \ldots, t_m$. An example of VL_{21} definite clause is given in Figure 4.5b.

The body of the clause describes the page layout of the document in Figure 4.5a, while the head defines the logical class of the layout object denoted by $x6$.

Definite clauses generated by INDUBI/CSL satisfy two constraints:

1. All variables in the head must occur in the body of the clause (that is, the clause is *range-restricted*).
2. Clauses must be linked. A VL_{21} definite clause is *linked* if all its selectors are. A selector is linked if at least one of its arguments is. An argument of a selector is linked if either the selector is the head of the clause or another argument of the same selector is linked.

Both training/testing examples and hypotheses are expressed as VL_{21} clauses, but there are some differences:

1. Each training/testing example is represented by a single ground VL_{21} clause.
2. Each hypothesis is expressed as a set of constant-free VL_{21} clauses having the same head.
3. The reference of each selector in any training/testing example has one value only.
4. In the examples, false predicates are omitted, while all the attributes are specified.
5. In the hypotheses, each omitted selector is assumed to be a function, which takes on any value of its range.
6. Variables with different names must be bound to distinct constants (Esposito et al., 1994b), since they are distinctly existentially quantified (Michalski, 1980).

Note that all these constraints on the languge of hypotheses are also valid for the languge of background knowledge. In addition to the training examples and the background knowledge. INDUBI/CSL requires a dependency graph in order to perform the appropriate shift of languge (procedure update_example_description) after all concepts at the same level have been learned (see Algorithm 1).

Algorithm 1. *Learning Contextual Rules and Shift of Language*

```
procedure learn_multiple_concepts(Examples, BK, {{C_1^0, ..., C_n1^0}, ..., {C_1^l, ..., C_nl^l}})
  update_example_description(Examples, BK)
  AllLearnedRules: = ∅
  for i: = 0 to l do
    LearnedRulesSameLevel: = ∅
    foreach C_j^i in {C_1^i, C_2^i, ..., C_ni^i} do
      LearnedRules: = Separate_and_Conquer(Examples, C_j^i)
      LearnedRulesSameLevel: = LearnedRulesSameLevel∪LearnedRules
    endforeach
  update_example_description(Examples, LearnedRulesSameLevel)
  AllLearnedRules: = AllLearnedRules∪LearnedRulesSameLevel
  endfor
  return AllLearnedRules
```

INDUBI/CSL implements a separate-and-conquer search strategy at the high level, in order to construct a set of clauses (see Algorithm 2), while it adopts a beam search at the low level, that is, for the construction of a single clause. More precisely, INDUBI/CSL starts with a positive example e^+ and generates a set *Cons* of at least m distinct generalizations, which are consistent and cover e^+. Such generalizations are extended-against and stored in MQ [see Esposito et al. (1994a) for a detailed description of the extension-against generalization rule]. Then, the best generalization is selected from MQ according to a preference criterion, which takes into acount the number of positive examples covered and the complexity of the generalization expressed by the number of its selectors, as well as the total cost of each generalization. Then, positive examples covered by the best generalization are removed from the set

Algorithm 2. *High-level Separate-and-conquer Strategy for Learning a Set of VL_{21} Definite Clauses*

```
procedure separate_and_conquer(Examples, Class)
  E^+: = set of positive Examples belonging to Class
  E^-: = set of negative Examples belonging to Class
  LearnedRules: = ∅
  while E^+ ≠ ∅ do
    MQ: = ∅
    randomly select e^+ from E^+
    Cons: = Beam_Search_for_consistent_hypotheses(e^+, E^+, E^-, m)
    foreach generalization G in Cons do
      MQ: = MQ∪extension_against(G, E^+, E^-)
    endforeach
    Best: = FindBest(MQ)
    LearnedRules: = LearnedRules∪{Best}
    E^+: = E^+ − Covers(Best, E^+)
  endwhile
  return LearnedRules
```

of positive examples and a new clause is generated, if the set of remaining positive examples is not empty.

At the low level, INDUBI/CSL proceeds top-down, specializing the unit clause:

$$[f(t_1, t_2, \ldots, t_n) = \lambda] \leftarrow$$

by adding one of the selectors in the positive example e^+ and turning constants into variables. Thanks to the constraint on the binding of variables, it is possible to turn distinct constants in the example description into distinct variables. Only a subset of n selectors among all selectors in the example are considered: they are chosen according to the cost associated to each function symbol in the referee and according to the arity of the function symbol, so that the greater the arity, the better. The former criterion offers the user a way to express any preference for some literals, while the latter criterion guarantees that relations are not treated unfairly. Obviously, selectors that cause the partial clause to become unlinked are not considered at all. All specialized VL_{21} definite clauses, which cover e^+ and possibly other positive examples, are ranked according to another preference criterion, which takes into account the number of positive and negative examples covered and the complexity of the clause, as well as the total cost. The first p generalizations are selected and stored in a set PS. Consistent generalizations are removed from PS and stored in *Cons* (see Algorithm 3).

To sum up, INDUBI/CSL adopts two generalization rules during the inductive learning process, the turning-constants-into-variables rule and the extension-against rule, and one specialization rule called adding-a-selector. Moreover, when background knowledge expressed in the form of VL_{21} definite clauses is available, INDUBI/CSL applies the *elementary saturation* operator

Algorithm 3. *Low-level Beam Search Strategy for Learning One VL_{21} Definite Clause*

```
procedure Beam_Search_for_consistent_hypotheses(e+, E+, E−, m)
  PS := {[f(t1,t2,...,tp)=λ]←}
  OldPS := ∅
  Cons := ∅
  while PS ≠ OldPS and |Cons| ≤ m do
  OldPS := PS
  PS := ∅
  foreach VL21 generalization G in OldPS do
    S := choose_best_linked_selectors(e+, G, n)
    S := turn_constants_into_variables(S)
    PS := PS ∪ specialize_G_by_adding_a_selector_in_S(G,S)
  endforeach
  Cons := Cons ∪ (consistent(PS) ∩ range-restricted(PS))
  PS := select_best_p_generalizations(PS)
  foreach VL21 generalization G in CONS do
    MQ := MQ ∪ extension_against(G, E+, E−)
  endforeach
endwhile
return Cons
```

TABLE 4.1. Experimental Results with INDUBI/CSL: Independence Assumption

Rule/Run	1	2	3	4	5	6	Av. Error
logo	0/0	0/0	0/0	0/0	0/0	0/0	0.0%
sender	0/0	0/1	0/1	0/1	0/9	0/2	16.9%
ref	1/0	1/3	1/0	1/1	1/4	0/2	14.3%
date	0/4	1/4	0/2	0/4	0/2	0/4	28.8%
receiver	2/1	0/0	0/0	0/3	0/5	0/2	17.9%
signature	0/1	0/2	0/2	0/0	0/0	0/0	8.9%
body	1/0	2/0	2/1	0/2	0/1	1/2	20%
Total	4/6	4/10	3/6	1/11	1/21	1/12	11%

described by Rouveirol (1992), before starting the learning process of all concepts. In particular, all clauses in the background knowledge are repeatedly matched against the training examples and all their conclusions are added to the examples in a forward-chaining way. The process ends when no new conclusion expressed as a selector can be added to any training instance. The added selectors are then considered by the procedure choose_best_linked_ selectors in the low-level beam-search strategy, so that they can be selected only if they appear useful.

It is worthwhile noting the INDUBI/CSL adopts the extensional notion of coverage, but provides an answer to some of the related problems we discussed in Section 3.1. Indeed, the problem of the generation of extensionally, but not intensionally, consistent hypotheses is faced by updating the example descriptions every time all concepts at the same level have been learned. Moreover, there is no possibility of generating extensionally, but not intensionally, complete hypotheses, since this could occur only with recursive definitions, which cannot be generated by INDUBI/CSL. Finally, the combination of this latter limit with the updating of the example descriptions prevents the system from generating intensionally, but not extensionally, complete hypotheses.

3.2.3. Experimental Results and Inference of Dependencies. INDUBI/CSL has been applied to the problem of document understanding. Several

TABLE 4.2. Experimental Results with INDUBI/CSL: Contextual Rules with Predefined Order

Rule/Run	1	2	3	4	5	6	Av. Error
logo	0/0	0/0	0/0	0/0	0/0	0/0	0.0%
sender	0/0	0/1	0/1	0/1	0/9	0/1	15.6%
ref	2/0	0/1	0/1	0/1	1/1	0/2	10.6%
date	0/3	0/0	1/0	0/5	0/6	1/2	23.5%
receiver	0/1	0/0	0/0	0/2	0/5	0/2	13.5%
signature	1/0	0/2	0/2	0/0	0/0	0/0	8.9%
body	0/1	2/0	2/1	0/2	0/1	1/1	18.3%
Total	3/5	2/4	3/5	0/11	1/22	2/8	9.0%

experiments have been organized to verify whether learning contextual rules leads to better results than learning under the independence assumption. For this purpose, we considered a set of 30 business letters,[1] with 354 layout objects. In each experiment, six different runs were made by randomly splitting the set of documents into two subsets according to the following criteria:

- 20 documents for the training set; and
- 10 documents for the test set.

Seven concepts have to be learned, namely, *sender* of the letter, *receiver*, *logotype*, *reference number (ref)*, *date*, *body*, and *signature*. Obviously, not all layout objects are instances of one of these concepts: some layout components have no logical class associated with them, while in some cases more than one layout object in a document is associated with the same logical class.

INDUBI/CSL has been used for learning both contextual and non-contextual rules. In the former case, we considered a linear dependency order, namely:

$$log \rightarrow signature \rightarrow body \rightarrow sender \rightarrow receiver \rightarrow ref \rightarrow date$$

Such an order is user-defined and can be partly explained in terms of spatial reasoning. Indeed, when the *logotype* and the *signature* have been recognized, the understanding of the geometrically contiguous logical objects, namely, *sender* for the *logotype* and *body* for the *signature*, should be easier. Following the same line of reasoning, we expect recognition of the *sender* to help to identify the *receiver*, which, together with the identification of *logo*, should help to locate the *ref* and, finally, the *date*. Comparison of Tables 4.1 and 4.2 shows that there is a uniform decrease in error rate for all the logical classes when the independence assumption is not made. Globally, the total average error is 2.0% lower for contextual rules than for noncontextual rules. The learning time was the same in both experiments.

Since the dependency order is linear, we have also tried to discover the order by means of statistics. The idea is the following: the accuracy of a classifier that takes into consideration only the characteristics of each layout object can provide information on how easily a logical object can be recognized without taking into account other logical objects in the context. We are aware that the dependency order defined in this way may be just a rule of thumb, but, as we will show later, it can help to identify at least the minimally dependent concepts. The classifiers that we adopted for this third experimentation are Fisher's linear discriminant functions (Hand, 1981), which take the following form:

$$q_i(x) = \mathbf{v} \cdot \mathbf{x} - v_0$$

where:

- $\mathbf{x}$ is an M-dimensional feature vector representing an observation;
- $\mathbf{v}$ is a vector of M coefficients; and
- v_0 is a constant term.

[1]Data are available via anonymous ftp at the UCI machine learning repository.

TABLE 4.3. Experimental Results with INDUBI/CSL: Contextual Rules with Order Defined by the Discriminant Analysis

Class/Run	1	2	3	4	5	6	Av. Error
logo	0/0	0/0	0/0	0/0	0/0	0/0	0.00%
sender	0/0	0/1	0/1	0/1	0/9	0/2	16.9%
ref	1/0	0/1	0/1	0/2	1/0	0/2	9.2%
date	0/2	0/0	0/1	0/5	0/2	0/4	16.4%
receiver	0/1	0/0	0/0	0/3	0/4	0/2	13.5%
signature	1/0	0/2	0/2	0/0	0/0	0/0	8.9%
body	0/1	2/0	2/1	1/2	0/1	0/2	20.0%
Total	2/4	2/4	2/6	1/13	1/16	0/12	8.7%

In order to find those logical components that can more easily be recognized by means of layout information, the discriminant functions have been applied directly on a subset of numerical features describing each single layout object. Such features result from previous phases of document processing, and concern the coordinates of the centroid of an object, the height, width, and eccentricity of a layout component, the number of black pixels in a layout object, and so on. For each logical class c_i we computed the following coefficient:

$$\kappa_i = (\text{number of positive instances covered} + \text{number of negative instances not covered})$$

and we ordered the logical classes according to the decreasing value of κ_i. Ties are broken by preferring the class with the greater number of positive instances. In the following, we list the different orderings obtained in the six experiments:

1. $logo \rightarrow ref \rightarrow date \rightarrow body \rightarrow signature \rightarrow sender \rightarrow receiver$
2. $logo \rightarrow ref \rightarrow signature \rightarrow date \rightarrow sender \rightarrow body \rightarrow receiver$
3. $logo \rightarrow signature \rightarrow ref \rightarrow body \rightarrow date \rightarrow sender \rightarrow receiver$
4. $logo \rightarrow ref \rightarrow date \rightarrow signature \rightarrow sender \rightarrow body \rightarrow receiver$
5. $logo \rightarrow sender \rightarrow date \rightarrow signature \rightarrow ref \rightarrow body \rightarrow receiver$
6. $logo \rightarrow sender \rightarrow ref \rightarrow date \rightarrow signature \rightarrow body \rightarrow receiver$

Table 4.3 shows the experimental results obtained by exploiting the ordering defined by the discriminant analysis. The idea of using statistical methods in order to define the dependency graph seems to work well, since the average error rate was slighty lower even than that obtained with the predefined order. Only in two experiments were the results worse, while globally, the number of commission errors decreased.

Finally, it is worthwhile noting that the results of the first two experiments agree with those obtained with a different first-order learning system, called

FOCL (Pazzani and Kibler, 1992). Even in that case we have observed an improvement in accuracy when the concept dependencies are considerd (Esposito et al., 1993a; Malerba, 1993): this result corroborates our opinion that taking concept dependencies into account is important.

4. CONCLUSIONS

In this chapter we have criticized the assumption of concept independence made by most of the well-known learning systems. Problems caused by the independence assumption are particularly evident in at least three situations: learning multiple attributes in attribute-based domains, learning multiple predicates in inductive logic programming, and learning classification rules for labeling problems. However, dropping the independence assumption raises several issues in its turn, the most important of which is the definition of an order in which concepts have to be learned.

When a dependency graph is given, it is possible to learn contextual rules by performing a shift of language according to the level assigned to each single concept in the dependency graph. On the contrary, when the user does not know which concepts depend on which, statistical methods can help to discover concept dependencies. In particular, statistical tests for independence, such as χ^2, can help to find significant correlations between concepts, while asymmetric measures of association, such as λ, Goodman and Kruskal's τ, and Somer's D can help to define a direction of dependence between two variables. Such a direction, however, simply establishes which variable can be more easily predicted given the other, but does not say anything about the possible causal relationships between them. In order to infer causal information from data, different methods have to be used, such as the one implemented in Pearl and Verma's Inductive-Causation Algorithm.

All these approaches are appropriate for attribute-based domains, but no easy solution is yet available for the case of structural domains. In this chapter we have presented an ad hoc solution for a real-world problem: document understanding. Experimental results in this domain confirm that by taking into account concept dependencies, it is possible to improve the classification accuracy. These encouraging results were obtained both when the order was defined by the user and when it was found by using linear dicriminant functions. In the latter case, the classification accuracy of a classifier that takes into consideration only the numerical characteristics of each layout object was exploited in order to understand how easily a logical component can be recognized without taking into account other logical components in the context. This simple rule proved effective not only for identifying the minimally dependent concepts, but also for suggesting the whole sequence in which concepts should be learned.

REFERENCES

Barbanente, A., Borri, D., Esposito, F., Leo, P., Maciocco, G., and Selicato, F. (1992). Automatically acquiring knowledge by digital maps in artificial intelligence planning. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*. Lecture Notes in Computer Science (A. Frank, I. Campari and U. Formentini, eds.), **639**, pp. 379–401. Springer-Verlag, Berlin.

Baroglio, C., Giordana, A., and Saitta, L. (1992). Learning mutually dependent relations. *J. Intell. Inf. Syst.* **1**, 159–176.

Cooper, G., and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.* **9**, 309–347.

Datta, P., and Kibler, D. (1993). Concept sharing: A means to improve multi-concept learning. In *Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA*, pp. 89–96. Morgan Kaufmann, San Mateo, CA.

DeRaedt, L. D., Lavrač, N., and Dzeroski, S. (1993). Multiple predicate learning. In *Proceedings of the Third International Workshop on Inductive Logic Programming*, pp. 221–240.

Esposito, F., Malerba, D., and Semeraro, G. (1993). Automated acquisition of rules for document understanding. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pp. 650–654. IEEE Computer Society Press, Los Alamitos, CA.

Esposito, F., Malerba, D., Semeraro, G., and Pazzani, M. (1993b). A machine learning approach to document understanding. In *Proceedings of the Second International Workshop on Multistrategy Learning*, pp. 276–292. George Mason University, Harpers Ferry, WV.

Esposito, F., Malerba, D., and Semeraro, G. (1994a). Multistrategy learning for document recognition. *Appl. Artif. Intell.* **8**, 33–84.

Esposito, F., Malerba, D., Semeraro, G., Brunk, C., and Pazzani, M. (1994b). Traps and pitfalls when learning logical definitions from relations. In *Methodologies for Intelligent Systems, Lecture Notes in Computer Science* (Z. W. Ras and M. Zemankova, eds.), **869**, pp. 376–385. Springer-Verlag, Berlin.

Gibbons, J. (1993). *Nonparametric Measures of Association, Sage University Paper, Series on Quantitative Applications in Social Sciences*. Sage, Newbury Park, CA.

Goodman, L., and Kruskal, W. (1972). Measures of association for cross classification IV: Simplification of asymptotic variances. *J. Am. Stat. Assoc.* **67**, 415–421.

Hand, D. (1981). *Discrimination and Classification*. Wiley, London.

Hogg, R., and Tanis, E. (1977). *Probability and Statistical Inference*. Macmillan, New York.

Horak, W. (1985) Office document architecture and office document interchange formats: Current status of international standardization. *IEEE Comput.* **18**(10), pp. 50–60.

Malerba, D. (1993). *Document Understanding: A Machine Learning Approch*, Tech. Rep., Esprit Proj. 5203 INTREPID.

Malerba, D., Semeraro, G., and Esposito, F. (1994). An analytic and empirical comparison of two methods for discovering probabilistic causal relationships. In *Machine Learning: ECML-94, Lecture Notes in Artificial Intelligence*,, (F. Bergadano and L. DeRaedt, eds.), **784**, 198–216. Springer-Verlag, Berlin.

McLeish, M. M., and Cecile, M. (1990). Enhancing medical expert systems with knowledge obtained from statistical data. *Ann. Math. Artif. Intelli.* **2**, 261–276.

Michalski, R. (1980). Pattern recognition as rule-guided inductive inference. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-2**, 349–361.

Michalski, R., and Larson, J. (1978). *Selection of the Most Representative Training Examples and Incremental Generation of VL_1 Hypotheses: The Underlying Methodology and the Description of Programs Esel and Aq11*, Tech. Rep. UIUCDCS-R-78-867. Dept. of Computer Science, University of Illinois, Urbana.

Mitchell, T. (1977). Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA*, pp. 305–310.

Mueller, J., Schessler, K., and Costner, H. (1970). *Statistical Reasoning in Sociology.* Houghton Mifflin, Boston.

Neapolitan, R. (1990). *Probabilistic Reasoning in Expert Systems.* Wiley, New York.

Pazzani, M., and Kibler, D. (1992). The utility of knowledge in inductive learning. *Mach. Learn.* **9**, 57–94.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Mateo, CA.

Pearl, J., and Verma, T. (1991). A theory of inferred causation. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, p. 441–452. Morgan Kaufmann, San Mateo, CA.

Piatetsky-Shapiro, G. (1991). Probabilistic data dependencies. In *Proceedings of the ML-92 Workshop on Machine Discovery*, pp. 441–451, Morgan Kaufmann, San Mateo, CA.

Plotkin, G. (1970). A note on inductive generalization. In *Machine Intelligence 5* (B. Meltzer and D. Michie, eds.), pp. 153–163. Edinburgh Univ. Press, Edinburgh.

Quinlan, J. (1986). Induction of decision trees. *Mach. Learn.* **1**, 81–106.

Quinlan, J. (1990a). Learning logical definitions from relations. *Mach. Learn.* **1**, 177–226.

Quinlan, J. (1990b). Probabilistic decision trees. In *Machine Learning: An Artificial Intelligence Approach* (Y. Kodratoff and R. Michalski, eds.), Vol. 3, pp. 140–152. Morgan Kaufmann, San Mateo, CA.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

Rendell, L. (1986). A general framework for induction and a study of selective induction. *Mach. Learn.* **1**, 177–226.

Rouveirol, C. (1992). Extensions of inversion of resolution applied to theory completion. In *Inductive Logic Programming* (S. Muggleton, ed.), pp. 63–90. Academic Press, London.

Stepp, R. (1979). *Learning without Negative Examples via Variable Valued Logic*, Tech. Rep. UIUCDCS-R-79-982. Dept. of Computer Science, University of Illinois, Urbana.

Vere, S. (1975). Induction of concepts in the predicate calculus. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tblisi, USSR*, pp. 281–287.