LEARNING TWO-TIERED DESCRIPTIONS OF FLEXIBLE CONCEPTS: THE POSEIDON SYSTEM

by

F. Bergadano
S. Matwin
R. S. Michalski
J. Zhang

Machine Learning, Vol. 8, No.1, pp. 5-43, Jan. 1992.

Machine Learning, 8, 5–43 (1992)

Learning Two-Tiered Descriptions of Flexible Concepts: The POSEIDON System

F. BERGADANO,[1]
S. MATWIN,[2] (STAN@CSI.UOTTAWA.CA)
R.S. MICHALSKI MICHALSKI@AIC.GMU.EDU
J. ZHANG
Artificial Intelligence Center, George Mason University, Fairfax, VA 22030

Editor: Jaime Carbonell

Abstract. This paper describes a method for learning *flexible* concepts, by which are meant concepts that lack precise definition and are context-dependent. To describe such concepts, the method employs a *two-tiered* representation, in which the first tier captures explicitly basic concept properties, and the second tier characterizes allowable concept's modifications and context dependency. In the proposed method, the first tier, called *Base Concept Representation* (BCR), is created in two phases. In phase 1, the AQ-15 rule learning program is applied to induce a complete and consistent concept description from supplied examples. In phase 2, this description is optimized according to a domain-dependent quality criterion. The second tier, called the *inferential concept interpretation* (ICI), consists of a procedure for *flexible matching*, and a set of inference rules. The proposed method has been implemented in the POSEIDON system, and experimentally tested on two real-world problems: learning the concept of an acceptable union contract, and learning voting patterns of Republicans and Democrats in the U.S. Congress. For comparison, a few other learning methods were also applied to the same problems. These methods included simple variants of exemplar-based learning, and an ID-3-type decision tree learning, implemented in the ASSISTANT program. In the experiments, POSEIDON generated concept descriptions that were both, more accurate and also substantially simpler than those produced by the other methods.

Keywords. Concept learning, learning imprecise concepts, inductive learning, learning flexible concepts, two-tiered concept representation, flexible matching

1. Introduction

Most current methods of machine learning, both empirical and analytic, assume that concepts are precise and context-independent, and representable by a single symbolic description. An important consequence of this assumption is that the recognition of such concepts, which we call *crisp*, is very simple: if an instance satisfies a concept description, then it belongs to the concept, otherwise it does not. Another common assumption is that concept instances are equally representative, that is, there is no distinction in typicality among instances.

In some methods, these assumptions are partially relaxed by assigning to a concept a fuzzy set membership function (e.g., Zadeh, 1974) or a probability distribution (e.g., Cheeseman, et al., 1988; Fisher, 1987). However, once such a measure is defined explicitly for a given concept, the concept again has a fixed, well-defined meaning. Moreover, these

[1]On leave from the University of Torino, Italy.
[2]On leave from the University of Ottawa, Canada.

Early ideas, experiments and the first method for learning two-tiered concept representations were presented in (Michalski, et al., 1986; Michalski, 1988; and Michalski, 1990). The method proposed there employed a simple form of description simplification, called TRUNC, which used only specialization as a description reduction operator. An intriguing result of that research was that the description's complexity was substantially reduced without affecting its performance on new examples. The effect was obtained by removing those parts of the complete and consistent description that covered only a small fraction of examples (the so called *light disjuncts*, or *light rules*), and by applying a *flexible matching* procedure for concept recognition.

This paper extends and continues these early ideas. One important advance is the development of a heuristic double-level search procedure, called TRUNC-SG, which explores the space of two-tiered descriptions to derive a globally optimized description. The search employs both generalization and specialization operators, and is guided by a new criterion, the *general description quality* measure (*GDQ*). This measure considers the accuracy of the description, the computational cost of both tiers—Base Concept Representation and Inferential Concept Interpretation, and its cognitive comprehensibility (Bergadano, et al., 1988). By introducing such a general description quality measure one can view any form of concept learning as a process of modifying the input concept description in order to maximize a given description quality measure. In this characterization, the initial concept description may be in the form of positive examples, positive and negative examples, a complete and/or consistent concept description, a tentative description (e.g., supplied by a teacher), an abstract concept definition (as in the explanation-based learning), or a combination of these forms.

Another difference between the present approach and previous research is that flexible matching is used not only in the recognition process, as in (Michalski, et al., 1986), but also in the learning process, i.e., in searching for high "quality" concept descriptions. This feature also distinguishes the method from the related work described in (Bergadano & Giordana, 1989), which does not involve deductive reasoning in the learning phase, and evaluates the performance of generated descriptions solely on the basis of the coverage of examples. These earlier approaches may be compared to using hands in learning how to row a boat, and then using oars in the performance phase.

The idea that learning is more effective if one uses the same instruments for learning and for the performance phases was also present in some incremental learning systems (e.g., Fisher 1987). The work described here represents also an important advance over tree-pruning techniques (e.g., Quinlan, 1987) which apply much more restrictive description reduction operators and do not use any form of deductive matching or flexible interpretation of the learned descriptions. Other advances include the ability to take into consideration the typicality of training instances (when it is known), and the introduction of a rule base in the Inferential Concept Interpretation.

The paper presents also a body of experimental results comparing the performance of the proposed method with several other methods, such as variants of exemplar-based learning, decision tree learning, learning complete and consistent descriptions, and the earlier method using two-tiered representation based on the TRUNC procedure.

The method proposed has been implemented in the POSEIDON[1] system, and experimentally applied to two different real-world problems: learning the concept of an acceptable

a probabilistic inference based on some measure of similarity, e.g., the *flexible matching* method (Michalski et al., 1986). Advanced matching may involve any kind of inference—deductive, analogical or inductive.

Let us illustrate the idea of two-tiered representation using the concept of "chair." A simple two-tiered representation of that concept is given below (for an example of a two-tiered chair description actually learned, see Bergadano, et al., 1988a).

BCR: *Superclass*: A piece of furniture.

Function: To seat one person.

Structure: A seat supported by legs and a backrest attached from the side.

Physical properties: The number of legs is usually four. Often made of wood. The height of the seat is usually about 14–18 inches from the end of the legs, etc.

(BCR may also include a picture or a 3D model of typical chairs)

ICI: *Possible variations of the properties in BCR*: The number of legs can vary from one to four. The legs may be replaced by any support. The shape of the seat, the legs and the backrest, and the material of which they are made are irrelevant, as long as the function is preserved. The backrest may be very small or missing, etc.

Variations:

If legs are replaced by wheels → type(chair) is wheelchair

Chair without the backrest → type(chair) = stool

Chair with the armrests → type(chair) = armchair

Context dependency:

Context = museum exhibit → chair is not used for seating persons any more, but has all the physical characteristics of a chair.

Context = toys → the size can be much smaller than stated in BCR. The chair does not serve for seating persons, but correspondingly small dolls.

This simple example illustrates several important features of two-tiered representation. Commonly occurring cases of chairs directly satisfy the BCR, and the ICI is not involved. The recognition time can thus be reduced for common cases. The BCR is not the same as a description of a prototype (e.g., Rosch & Mervis, 1975), as it can be a generalization characterizing different typical cases or be a set of different prototypes. The ICI does not represent only distortions or corruptions of the prototype, but it can describe some radically different cases. When an entity does not satisfy the base representation of any relevant

A disjunction of VL_1 conjunctive statements is a *VL_1 DNF expression*. An expression in which one VL_1 conjunctive statement implies another is called a *VL_1 rule*. If a VL_1 conjunctive statement, S, is a description of examples of class C, then this fact is equivalent to a rule S → [class = C]. For simplicity, from now on, a VL_1 elementary condition is called a *condition*, a VL_1 conjunctive statement with an implied class is called a *rule*, and a VL_1 DNF description of a class is called a *ruleset* for that class. Both, the base representation, as well as the rules in the inferential interpretation of a concept are represented as (VL_1) rulesets.

2.2. *Inferential Concept Interpretation: Flexible matching function*

A part of the Inferential Concept Interpretation is *flexible matching function*, F, which is assumed to be given as the background knowledge of the learner. The function measures the degree of match between an event (example) and a concept description. In the method implemented, F maps events from the set E, and concept descriptions from the set D, into the degree of match from the interval [0..1]:

$$F: E \times D \rightarrow [0..1]$$

The value of F for an event e, and a concept description D, is defined as the probabilistic sum of F for its rules. Thus, if D consists of two rules, r_1 and r_2, we have:

$$F(e,D) = F(e, r_1) + F(e, r_2) - F(e, r_1) \times F(e, r_2)$$

A weakness of the probabilistic sum is that it is biased toward descriptions with many rules. If a concept description D has a large number of rules, the value of $F(e, D)$ may be close to 1, even if $F(e, r)$ for each rule r, is relatively small (see Table 3 in Section 6). To avoid this effect, if the value of $F(e, r)$ is below a certain threshold, then it is assumed to be 0. In POSEIDON, this problem does not occur, because concept descriptions are typically reduced to only few rules (see the TRUNC-SG procedure in Section 3).

The degree of match, $F(e, r)$ between an event e, and a rule r, is defined as the average of the degrees of fit for its constituent conditions, weighted by the proportion of positive examples to all examples covered by the rule:

$$F(e, r) = \left[\sum_i F(e, c_i)/n \right] \times \#rpos/(\#rpos + \#rneg)$$

where $F(e, c_i)$ is a degree of match between the event e and the condition c_i in the rule r, n is the number of conditions in r, and *#rpos* and *#rneg* are the number of positive examples and the number of negative examples covered by r, respectively.

The degree of match between an event and a condition depends on the type of the attribute in the condition. An attribute can be one of four types: nominal, structured-nominal, linear and structured-linear (Michalski & Stepp, 1983). Values of a structured-nominal

are S-covered are called *representative* examples; examples that are F-covered are called *nearly-representative* examples; and examples that are D-covered are called *exceptions*. As mentioned earlier, one of the major advances of the presented method over previous methods using two-tiered representation (e.g., Michalski , et al., 1986) is that the Inferential Concept Interpretation includes not only a flexible matching procedure, but also inference rules. Thus, using our newly introduced terminology, we can say that the method can handle not only representative or nearly representative examples, but also exceptions.

3. An overview of POSEIDON

The ideas presented above have been implemented in a system for learning two-tiered descriptions, called POSEIDON. Table 1 summarizes the two basic phases in which the system learns the Base Concept Representation. The first phase generates a general consistent and complete concept description, and the second phase optimizes the description according to a given measure of description quality.

3.1. Basic algorithm

The complete and consistent description is determined by the AQ15 inductive learning program (Michalski, et al., 1986). The second phase improves this description by conducting a "double level" best-first search. This search is implemented by the TRUNC-SG procedure (SG symbolizes that the method uses both specialization and generalization operators). In this "double level" search, the first level is guided by the *description quality measure*, which ranks candidate descriptions (see Section 4). The second level search is guided by heuristics controlling the search operators to be applied to a given description. The search operators simplify the description by removing some of its components, or by modifying the arguments or referents of some of its predicates (see sec. 3.2). A general structure of the system is presented in Figure 1.

Table 1. Basic phases in generating BCR in POSEIDON.

Phase 1
Given:
Concept examples obtained from some source
Relevant background knowledge
Determine:
Complete and consistent description of the concept
Phase 2
Given:
Complete and consistent description of the concept
A general description quality (GDQ) measure
Typicality of examples (if available)
Determine:
The Base Concept Representation that maximizes GDQ.

of the "adding an alternative" generalization rule (Michalski, 1983). Condition removal (from a rule) is a generalization operator, as it is an instance of the "dropping condition" generalization rule.

The referent modification operator changes the referent in a condition. Such changes can either generalize or specialize a description. Consequently, two more specific operators are defined: *condition extension*, which generalizes the description, and *condition contraction*, which specializes the description.

To illustrate these two types of referent modification, consider the condition

$$[\text{size} = 1..5 \vee 7]$$

A referent modification that produces a condition

$$[\text{size} = 1..7]$$

is a condition extension operator. If, however, the initial condition was

$$[\text{size} \neq 1..5 \vee 7]$$

then the same referent modification, i.e., the change to [size $\neq$ 1..7], would represent a condition contraction operator. A referent modification (in the original condition) that produces a condition

$$[\text{size} = 1..5]$$

is a condition contraction operator. Similarly, if the initial condition was [size $\neq$ 1..5 v 7], then the modification to [size $\neq$ 1..5] would represent a condition extension operator. A summary of the effect of different operators on a description is given in Table 2.

Thus, applying the above search operators can either specialize or generalize the given description. A generalized (specialized) description covers potentially a larger (smaller) number of training examples, which can be positive or negative. At any given search step, the algorithm chooses an operator on the basis of an evaluation of the changes in the coverage caused by applying the operator (Section 5.2).

3.3. *Learning the Inferential Concept Interpretation*

As indicated above, by applying a search operator (RR, CR, CE or CC) to the current Base Concept Representation, one can make it either more general or more specific. If

Table 2. Search operators and their effect on the description.

Search Operator	Type of Knowledge Modification
Rule removal (RR)	Specialization
Condition removal (CR)	Generalization
Condition extension (CE)	Generalization
Condition contraction (CC)	Specialization

examples, the system may produce overly complex and detailed descriptions. Such descriptions may strongly depend on the particular training set, and, consequently, may perform poorly in classifying future examples. For that reason, when learning from imperfect inputs, it is often better to produce descriptions that are only partially complete and/or consistent.

The comprehensibility of the acquired knowledge depends on subjective and domain-dependent criteria. If an intelligent system is supposed to give advice to humans, knowledge used by such a system should be comprehensible to human experts. A "black box" classifier is not satisfactory in such situations. Therefore, knowledge acquired by a learning system should be related to terms, relations and concepts used by experts, and should not be too complex syntactically. This requirement is called the *comprehensibility principle* (Michalski, 1983). There is no well established measure of description's comprehensibility, therefore we approximate it by using a measure of a *representational simplicity* of a description. Such a simplicity measure is determined by counting the number of operators of different kinds involved: disjunctions, conjunctions, and the relations embedded in individual conditions. In the case of two-tiered representations, the proposed approximation of the comprehensibility takes into account the operators occurring in both, the BCR and the ICI, and weighs the relative contribution of each part to the comprehensibility of the whole description.

The third criterion, the description cost, captures the properties of the description related to its storage and its use (the *computational complexity*). Other things being equal, descriptions which are easier to store and easier to use for recognizing new examples are preferred. When evaluating the description cost, two characteristics are of primary importance. The first is the cost of measuring values of variables occurring in the description. In some application domains, e.g., in medicine, this is a very important factor. The second characteristic is the computation cost (time and space) of evaluating the description. Again, in some real-time applications, e.g., in speech or image recognition, there may be stringent constraints on the evaluation time. The cost and the comprehensibility of a description are frequently mutually dependent, but generally these are different criteria.

The criteria described above need to be combined into a single evaluation measure that can be used to compare different concept descriptions. One solution is to have an algebraic formula that, given numeric evaluations for individual criteria, produces a number that represents their combined value. Such a formula may involve, e.g., a multiplication, weighted sum, maximum/minimum, or t-norm/t-conorm of the component criteria (e.g., Weber, 1983). Although such an approach is often appropriate, it also has significant disadvantages.

First, it combines a set of heterogeneous evaluations into a single number, and the meaning of this final number is hard to understand for a human expert. Second, it usually forces the system to evaluate all the criteria for each description, even if it is sufficient to compare descriptions on the basis of just one or two most important ones. The latter situation occurs when one description is so much better than the other according to some important criterion, that it is not worth to even consider the alternatives. To overcome these problems, we use a combination of two measures, a lexicographic evaluation and a linear function-based evaluation, as described in the next section.

of positive and negative examples covered by the description (training and/or testing). One can argue, however, that in evaluating accuracy one should take into consideration also the *typicality* of examples (Rosch & Mervis, 1975). If two descriptions cover the same number of positive and negative examples, the one that covers more typical positive examples and less typical negative examples can be considered more accurate.

For the above reason, we propose a measure of the degree of completeness and the degree of consistency of a description that takes into account the typicality of the examples. The typicality of examples can be approximated by the frequency of their occurrence. Alternatively, the teacher supplying the examples can be asked to assign the typicality to the examples. If the typicality is not provided, the system makes the standard assumption that the typicality is the same for all examples. The degree of completeness of a description is proportional to the typicality of the positive events covered. The consistency of the description is inversely proportional to the typicality of the negative events covered.[3]

In defining the completeness and consistency of a two-tiered description, other factors may be taken into account. One may postulate that a description should cover the typical examples explicitly, and non-typical ones implicitly. Thus, the typical examples are covered by the Base Concept Representation, and non-typical, or exceptional ones by the Inferential Concept Interpretation. In POSEIDON, the Base Concept Representation is inductively learned from examples provided by a teacher, and it is desirable that they are typical of the concept being learned. The Inferential Concept Interpretation rules are provided by a teacher. They are assumed to handle special or rare cases. An advantage of such an approach is that the system learns a description of typical examples by itself, and the teacher needs to explain only the special cases.

In view of the above, the explicitly-covered (strictly-covered, or S-COV) examples are assumed to contribute to the completeness of a description more than implicitly-covered, i.e., flexibly-covered (F-COV) or deductive inference rules-covered (D-COV) examples. These assumptions are reflected by weights w_s, w_f, and w_d used in the definition of completeness and consistency described in the next section.

4.4. General Description Quality measure

This section defines the General Description Quality (GDQ) measure implemented in POSEIDON. To this end, we first define the *typicality-based completeness*, T__COMPLETENESS, and the *typicality-based consistency*, T__CONSISTENCY, of a two-tiered concept description:

$$\text{T\_\_COMPLETENESS} = \frac{\underset{e^+ \in \text{S-cov}}{\Sigma w_s * \text{Typicality}(e^+)} + \underset{3^+ \in \text{F-cov}}{\Sigma w_f * \text{Typicality}(e^+)} + \underset{e^+ \in \text{D-cov}}{\Sigma w_d * \text{Typicality}(e^+)}}{\underset{e \in \text{POS}}{\Sigma\ \text{Typicality}(e)}}$$

$$\text{T\_\_CONSISTENCY} = 1 - \frac{\underset{e^- \in \text{S-cov}}{\Sigma w_s * \text{Typicality}(e^-)} + \underset{3^- \in \text{F-cov}}{\Sigma w_f * \text{Typicality}(e^-)} + \underset{e^- \in \text{D-cov}}{\Sigma w_d * \text{Typicality}(e^-)}}{\underset{e \in \text{NEG}}{\Sigma\ \text{Typicality}(e)}}$$

Measuring-Cost (MC)—the cost of measuring variables used in the concept description.
Evaluation-Cost (EC)—the cost of evaluating the concept description.

$$\text{MC(description)} = \sum_{e \in Pos+Neg} \sum_{v \in vars(e)} mc(v)/(/Pos/ + /Neg/)$$

$$\text{ED(description)} = \sum_{e \in Pos+Neg} ec(e)/(/Pos/ + /Neg/)$$

where *vars(e)* is the set of all occurrences of variables used to evaluate a concept description to classify the event *e*, *mc(v)* is the cost of measuring the values of the variable *v*, and *ec(e)* is the computational cost of evaluating the concept description to classify the event e. The latter depends on the computing time and/or on the number of operators involved in the evaluation.

We now define the cost of a description:

$$\text{Cost(description)} = u_1 * \text{MC(description)} + u_2 * \text{EC(description)}$$

where u_1 and u_2 are weights defining the relative importance of the measuring-cost and the evaluation-cost for a given problem.

The definitions of the above measures together with the specification of the way they can be combined (sec. 4.2) define the general description quality. Various weights used in the measures are specified by the program's user to reflect the requirements of the problem, or determined experimentally. For more details about the description quality measure see (Bergadano, et al., 1988).

5. Learning by maximizing the concept description quality

As mentioned before, learning a base representation of a concept is performed in two phases. In the first phase, a complete and consistent concept description is generated by inductive learning from examples. In the second phase, the obtained complete and consistent description is optimized according to the general description quality criterion. In our approach, the first phase is done using the AQ15 learning program (Michalski, et al., 1986a). This section describes the second phase.

5.1. Search heuristics for optimizing Base Concept Representation

Optimizing BCR by a direct application of the General Description Quality measure is computationally expensive, because every newly generated description has to be matched flexibly against the complete set of training examples. To make the process more efficient, we have introduced a *double-level* search method. The first level uses a simple heuristic to determine which operator (RR, CR, CE or CC) is likely to improve the description, and the second level actually applies the operator, and evaluates the description using General Description Quality.

5.2. Search algorithm

The search process is conducted according to the algorithm in Table 3:

Table 3. The algorithm for optimizing a concept description.

1. Identify in the search tree the best candidate description D

   (Initially, D is the complete and consistent description obtained by the AQ15 program in Phase 1, and then it is the highest rank description according to the General Description Quality criterion).

2. Apply to D the operator, from among the operators:

   RR_i: Remove the i-th rule from D.
   CR_{ij}: Remove the j-th condition from the i-th rule in D.
   CE_{ij}: Extend the referent of the j-th condition in the i-th rule in D.
   CC_{ij}: Contract the referent of the j-th condition in the i-th rule in D,

   that maximizes the Potential Accuracy Improvement measure.

3. Compute the General Description Quality (GDQ) of the description obtained in step 2. If the GDQ is smaller than the GDQ of original D, then proceed to step 1. (When computing the description accuracy for GDQ, flexible matching is used).

4. Ask an expert for an explanation of the exceptional examples, which are

   (a) the positive examples that cease to be covered, and
   (b) the negative examples that become covered.

   If an explanation is given, add the rules that make up the explanation to the Inferential Concept Interpretation, otherwise add to it the exceptional example(s).

5. Update the GDQ value of the new node by taking into account the added Inferential Concept Interpretation.

6. If the *stopping criterion* is satisfied, then STOP, otherwise proceed to step 1.

Let us explain the motivation and individual steps of the algorithm. Step 1 chooses the node (description) for expansion on the best-first basis, that is, chooses the node with the highest General Description Quality. (This is not always an optimal choice, because "worse" nodes can sometimes lead to better descriptions after a number of removals. Whether the search will behave in this manner will depend on the adequacy of the General Description Quality as the measure of concept quality).

Step 2 chooses the "best" search operator according to the Potential Accuracy Improvement heuristic, and applies it to the current description. Step 3 computes the General Description Quality of the new node. It should be noted that, in the General Description Quality measure, the typical examples covered directly by the base representation can weigh more than those covered through flexible matching. The examples covered by Inferential Concept Interpretation rules weigh more than the ones covered through flexible matching, but less than the ones covered by the Base Concept Representation.

Step 4 determines exceptional examples, and asks an expert for an explanation of them. If the explanation is provided, appropriate rules are added to the Inferential Concept Interpretation. These rules may extend or contract the Base Concept Representation. For example, the rule removal operator might uncover some positive examples, that were previously covered. In this case, new rules added to the Inferential Concept Interpretation would allow

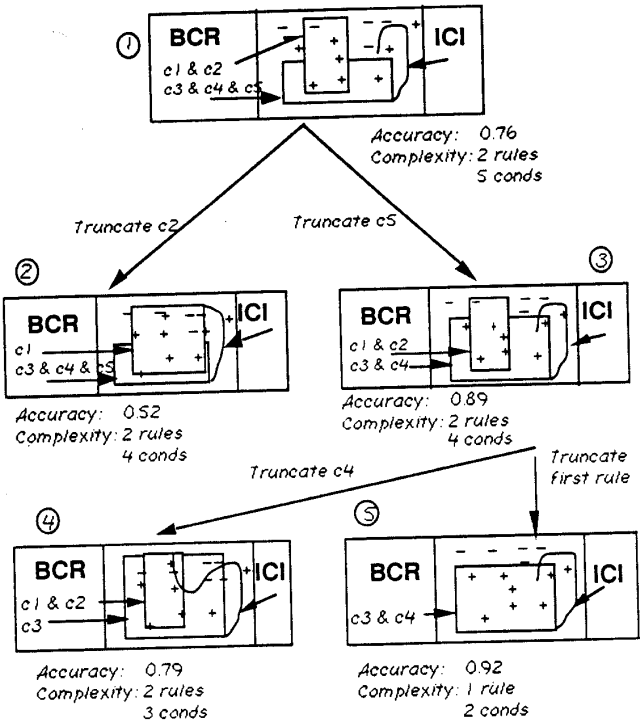

Figure 2. An illustration of the search process.

Concept Interpretation extends this coverage by recognizing one more positive example. Next nodes correspond to descriptions obtained by an application of operators marking the branches of the search tree. For example, node 3 is obtained by eliminating condition C5 in the second rule of the initial description. The new description is more accurate because all positive examples are now covered, without changing the coverage of the negative examples.

By truncating the first rule in node 3, node 5 is generated. The description no longer covers negative examples, and is simpler. This node is then accepted as the optimized description resulting from the search. The other nodes lead to inferior concept representations with respect to General Description Quality, and are discarded. The quality has been computed with $w_1 = w_2 = 0.5$. For simplicity, the cost is omitted, and the complexity of the Inferential Concept Interpretation is ignored. The complexity of the Base Concept Representation is indicated by the number of rules and the number of conditions.

6. Experiments

Experiments tested the POSEIDON program, and compared its performance with that of three other methods: variants of exemplar-based learning, the method for learning consistent

Shift differential = second shift is paid 25% more than first shift
Educational allowance is offered
Holidays/per year = 11 days
Vacation offer = better than average in the industry
Long term disability insurance = offered by the employer
50% dental insurance cost = covered by the employer
Bereavement leave = available
Employer-sponsored health plan = not mentioned

The above description was represented as the following VL_1 rule:

[Dur = 2][wagel = 1.5][Wage2 = 3.5][cola = unknown][Work-hours = 38]
[Pension = none][Stby-pay = 12][Shift-diff = 25][Educ-allw = yes]
[Holidays = 11][Vacation = better][lngtrm-disabil = true][Dntl-ins = half]
[Bereavement = yes][Empl-hlth-plan = unknown]

::> [contract = acceptable]

(In the above rule, for simplicity, the conjunction is represented by concatenation).

U.S. Congress voting record

The data regarding the U.S. Congress voting record were the same as the ones used by Lebowitz (1987) in his experiments on conceptual clustering. The data represent the 1981 voting records of 100 selected representatives (50 in the training set and 50 in the testing set). The problem was to learn descriptions discriminating between the voting record of Democrats and Republicans. Below is an example of the voting record of a Democrat in the U.S. Congress:

Draft registration = no
Ban aid to Nicaragua = no
Cut expenditure on MX missiles = yes
Federal subsidy to nuclear power stations = yes
Subsidy to national parks in Alaska = yes
Fair housing bill = yes
Limit on PAC contributions = yes
Limit on food stamp program = no
Federal help to education = no
State = north east
Population = large
Occupation = unknown
Cut in Social Security spending = no
Federal help to Chrysler Corp. = vote not registered

```
[contract-duration >1]&[wage_incr_yr2 >3.0%]&[#holidays >10]:      (t = 11, u = 11)
 or
[wage_incr_yr1 > 4.5%]:                                           (t = 4, u = 4)
 or
[wage_incr_yr1 > 4%] & [wage_incr_yr2 > 4.0%]:                    (t = 1, u = 1)
 or
[wage_incr_yr1 > 4.5%] & [#holidays > 9]:                         (t = 1, u = 1)
 or
[wage_incr_yr1 > 2%] & [vacation > average]:                       (t = 1, u = 1)

     ::>  [Contract =  acceptable]

[wage_incr_yr1 = 2..4%]&[#holidays < 10]&[vacation ≤ average]:     (t = 3, u = 3)
 or
[wage_incr_yr1 ≤ 4.5%] & [wage_incr_yr2 ≤ 4.0%] &
[holidays = 10] & [vacation = below_average v average]:           (t = 2, u = 2)
 or
[duration = 1] & [wage_incr_yr1 < 4.0%] & [#holidays < 10]&
[vacation = below_average v average]:                             (t = 1, u = 1)
[wage_incr_yr1 ≤ 4.0%] & [wage_incr_yr2 ≤ 3.0%] &
[vacation = below_average v average]:                             (t = 1, u = 1)
 or
[duration = 1] & [wage_incr_yr1 ≤ 4.0%]&
[vacation = below_average v average]:                             (t = 1, u = 1)
 or
[wage_incr_yr1 = 2.0%] & [wage_incr_yr2 ≤ 3.0%]:                  (t = 1, u = 1)

     ::>  [Contract =  unacceptable]
```

Figure 3. The complete and consistent descriptions generated by AQ15.

BCR:

```
[contract-duration >1]&[wage_incr_yr2 >3%]&[#holidays >10]:        (t = 11, u = 11)
     ::>  [Contract =  acceptable]

[wage_incr_yr1 = 2..4%]&[#holidays < 10]&[vacation ≤ average]:     (t = 3, u = 3)
     ::>  [Contract =  unacceptable]
```

ICI: Flexible matching

Figure 4. The *Top rule descriptions* generated by the TRUNC method.

As one can see, the optimized BCR descriptions are significantly simpler than the complete and consistent descriptions generated by AQ15. They also seem to represent the most important characteristics of the labor management contracts. Specifically, a contract is unacceptable when it offers a significant wage increase (the first two rules in Fig. 5), or it offers many holiday days, or the vacation time is above average.

6.3. Results from testing POSEIDON and other methods

As mentioned earlier, experiments tested POSEIDON and three other methods, specifically, variants of exemplar-based learning, the method for learning consistent and complete descriptions, a method for generating *top rule* descriptions, and a method for generating pruned decision trees. All of these methods were employed to learn a concept description from

Table 4. Results of experiment 1.

Simple Exemplar-Based Description Labor-mgmt problem (Labor): 27 rules and 432 conditions Congress problem (Congress): 51 rules and 969 conditions						
	Correct		Incorrect		No_Match	
	Labor	Congress	Labor	Congress	Labor	Congress
Strict Match						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	0%	4%	0%	0%	100%	96%
1-Nearest Neighbor						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	77%	**86%**	23%	14%	0%	0%
3-Nearest Neighbors						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	**83%**	84%	17%	16%	0%	0%
5-Nearest Neighbors						
Training Set	100%	100%	0%	0%	0%	0%
Testing Set	80%	84%	20%	16%	0%	0%

Subsequent parts of Experiment 1 tested the factual description using the *k-nearest neighbor* method with different k. The method involved determining k closest (best "fitting") learning examples to the one being classified, and assigning to the testing example the class of the majority of the closest examples. Such a method is equivalent to simple forms of exemplar-based learning. The *1-Nearest Neighbor* row in the table lists results from applying the factual description with a matching method somewhat similar to the one described in (Kibler & Aha, 1987). The only difference was that Kibler and Aha's method uses the *maximum* function for evaluating a ruleset (disjunction), while our flexible matching uses the *probabilistic sum* (Section 2.2). We also tested the method with k = 3 and 5.

The second experiment (*Table 5*) used concept descriptions generated by AQ15 without truncation. Such descriptions are consistent and complete with regard to the training examples, i.e., they classify all training examples 100% correct when using the strict matching method. The flexible matching method did not change this result. For the testing set, the number of correct classifications was relatively high (80–86%), the same for the strict and flexible matching methods. Flexible matching made no difference, probably due to two factors. Firstly, the complete and consistent descriptions include many specific rules, leaving little space for the "no match" cases (3%), in which flexible matching could help. Secondly, the descriptions consisted only of disjoint rules, as the program was run using the "disjoint cover" parameter. In such a situation, the "multiple match" cases do not occur, and flexible matching cannot help.

The above results are similar to those obtained in the previous experiment, which used an exemplar-based approach (*Table 4*). The main difference is that the AQ descriptions are much simpler in terms of the number of rules and the number of conditions involved (11 vs. 27 rules in the labor management problem, and 10 vs. 51 rules in the congress voting problem). The simpler descriptions allow the system to be more efficient in the recognition mode.

Table 7. Results of experiment 4.

	Optimized (POSEIDON) Labor-mgmt problem (Labor): 9 rules and 12 conditions Congress problem (Congress): 10 rules and 21 conditions					
	Correct		Incorrect		No_Match	
	Labor	Congress	Labor	Congress	Labor	Congress
Strict Match						
Training Set	63%	84%	0%	0%	37%	16%
Testing Set	43%	73%	3%	4%	54%	23%
Flexible Match						
Training Set	85%	100%	0%	0%	15%	0%
Testing Set	83%	92%	13%	8%	4%	0%
Deductive Match						
Training Set	96%	96%	0%	4%	4%	0%
Testing Set	**90%**	**92%**	10%	8%	0%	0%

For comparison, the performance of these descriptions was also tested using strict match. The latter is rather an impractical combination. As expected, these descriptions used with strict matching gave relatively poor performance.

The optimized descriptions (BCR) combined with deductive matching (ICI) gave the best performance (90–92% correct). When used with only flexible matching, the performance was slightly lower. The descriptions are simpler than complete and consistent descriptions, although they include the Inferential Concept Interpretation rules. They are, of course, more complex than the top rule descriptions, which do not use any interpretation rules.

For the Labor data, descriptions applied with deductive matching produced higher performance than when used with flexible matching only (90 vs. 83%).[4] For the Congress data problem, the performance was the same for the two matching methods. This is because deductive rules were acquired on the training set; in the specific testing set, the D-covered events were the same as F-covered ones.

Table 8 summarizes the results of experiments, specifically, it compares the performance and complexity of descriptions generated by simple exemplar-based methods, the two-tiered descriptions generated by POSEIDON, and pruned decision trees generated by ASSISTANT (a descendant of the Quinlan's ID3 program; Cestnik, et al., 1987). ASSISTANT was applied to the same learning and training data, which were used in the previous experiments (whose results were presented in Tables 4–7). The decision trees obtained by ASSISTANT were optimized using a tree-pruning mechanism (Cestnik, et al., 1987). This mechanism is compared with the TRUNC-SG method in the next section.

The factual description was applied with the flexible matching function. The complexity of a rule-based description was measured by stating the number of rules (#Rules) and the number of conditions (#Conds). The complexity of a decision tree was measured by the number of leaves (#Leaves) and the number of nodes (#Nodes). The number of rules in a rule-based description can be taken as comparable with the number of leaves in a decision tree, because for each leaf of the tree one can generate one rule by tracing the nodes from the root to the leaf.

6.4. The role of parameters and related issues

POSEIDON has many parameters which can be controlled by a user. On the surface, this might be considered as a disadvantage. In our view, a learning system that allows the user to explicitly modify parameters that affect learning processes (but which are not just method-dependent), is to be preferred over a system that does not explicitly define such parameters. The point is that in the latter systems these parameters are defined only *implicitly*, by the assumptions and the structure of the method. For example, many systems do not take into consideration the typicality of examples. In POSEIDON, this is equivalent to an assumption that the typicality of all examples is equal to the default value 1. As another example, consider the cost of measuring the value of attributes. If a learning program does not have parameters representing such costs, then this is equivalent to an assumption that all costs are the same (which in reality is often not true). By being able to control such learning parameters, the user can produce results that better fit the task at hand. For example, for some tasks, the accuracy of descriptions may be a decisive criterion, while for others the description simplicity may be of equal concern.

An important problem to be investigated is the sensitivity of POSEIDON to its various parameters. While a comprehensive answer to this problem goes beyond the scope of this paper, we report below a preliminary sensitivity analysis regarding the parameters controlling the trade-off between the description accuracy and simplicity. Such parameters are considered to have the most important effect on the performance of learned descriptions. Specifically, they are the *tolerances* in the lexicographic evaluation functional measuring the description quality (Sec. 4.2). To explain their role, let us briefly review the description quality measure. This measure combines several criteria, such as the accuracy, the simplicity, and the cost. Each criterion is associated with a tolerance interval such that differences within this interval are not considered unimportant. Thus, if the tolerance interval of accuracy is very narrow, then the accuracy becomes the prevailing criterion in quality evaluation. On the other hand, if this tolerance interval is wide, the remaining criteria become more significant.

An experiment was performed using the same Congress voting data, as used in experiments reported in Tables 4–7. The training set had 51 examples, while the testing set had 49 examples. The concept to be learned was the voting record of Republicans in the U.S. Congress. The description tested in Table 7, had 10 rules and 21 conditions, and yielded the accuracy of 100% on the training set, and 92% on the testing set. The description was obtained using the accuracy tolerance (τ_1) value equal .05. To determine the method's sensitivity to this parameter, the accuracy tolerance τ_1 was set to values .55, .35, .02, .005, and for each value the description accuracy was measured. For the above accuracy tolerances, the system's performance on the testing set was 88%, 88%, 90%, and 92%, respectively. Thus, this experiment seems to indicate that the accuracy of the descriptions slowly grows with the narrowing of the tolerance interval on the accuracy in the description quality measure, which completely confirms an intuitive expectation.

In general, when the accuracy tolerance interval is wide, the simplicity of the description assumes an important role, yielding performances close to the performance of the *top rule* in the two-tiered description. Intermediate values, such as the one used in the experiments presented in Table 8 ($\tau_1 = 0.05$) produced the best results, e.g., the performance of 92%

the search, such a procedure should, however, be significantly faster than the one implemented in POSEIDON. If the speed of learning and the simplicity of descriptions are of central importance, then the TRUNC method (that determines the top rule descriptions without search) should be applied rather than TRUNC-SG. In the same paper (Quinlan, 1987), other methods for pruning decision trees are also described. Some of these methods require a separate testing set for the simplification phase, and others use the same training set that was used in creating the tree. The simplification phase in POSEIDON can also be done either using the original training set, or using a separate set of examples.

The experiments by Fisher and Schlimmer (1988) on pruning decision trees use a statistical measure to determine the attributes to be pruned. Such measures require a rather large data sample, and thus do not apply well to small training sets. In the two-tiered approach, training events are analyzed logically, rather than statistically, both in the phase creating a complete and consistent description, and in the optimization phase. Consequently, the two-tiered approach seems to be more suited for learning from a relatively small number of examples. An interesting possibility for future research is to integrate a statistical measure, such as used by Fisher and Schlimmer, or other, in the process of rule learning and truncating with large data sets.

The system developed by (Iba, et al., 1988) uses a trade-off measure that is somewhat similar to the general description quality (GDQ) measure proposed in this paper. Our GDQ measure considers more factors. Besides taking into account the typicality of the instances covered by the description, it considers different types of matching between an instance and a description. Moreover, the simplicity measured by GDQ depends not only on the number of rules in the description as in (Iba, et al., 1988), but also on the different syntactic features in the description.

The CN2 inductive algorithm (Clark & Niblett, 1989) uses a heuristic function to terminate search during rule construction. The heuristic is based on an estimate of the noise present in the data. Such pruning of the search space of inductive hypotheses results in rules that may not classify all the training examples correctly, but that perform well on testing data. CN2 can be viewed as an induction algorithm that includes pre-truncation, while the algorithm reported here is based on post-truncation. CN2 applies truncation during rule generation, and POSEIDON applies truncation after rule generation. The advantage of pre-truncation is efficiency of the learning process. On the other hand, such an approach has difficulty with identifying irrelevant conditions and redundant rules.

The two-tiered method described here can also be viewed as a kind of constructive induction in the sense of (Michalski, 1983). In fact, the whole learned description may include new terms, absent from the examples used for learning. This behavior is also encountered in several other systems (e.g. Sammut & Banerji, 1986; Drastal, Czako, & Raatz, 1989). However, constructive learning in POSEIDON is due to the second tier based on domain knowledge characterizing non-typical examples. This is different from using domain knowledge to rewrite or augment the whole training set (e.g., Rouveirol, 1991), or to generate new attributes by a data-driven approach (Bloedorn & Michalski, 1991), or a hypothesis-driven approach (Wnek & Michalski, 1991).

The exemplar-based learning system PROTOS (Bareiss, 1989) is similar to POSEIDON in the sense that both systems use a sophisticated matching procedure—a knowledge-based matching of an event with a concept description and acquiring the matching knowledge

monolithical structure. In this representation, the first tier, the base concept representation (BCR), captures the explicit and common concept meaning, and the second tier, the inferential concept interpretation (ICI) defines allowable modifications of the base meaning and exceptions. Thus, typical concept instances match the BCR, and thus can be recognized efficiently. Such a two-tiered representation is particularly suitable for learning flexible concepts, i.e., concepts that lack precise definition and are context-dependent.

In the POSEIDON system that implements the method, the BCR is learned in two steps. First, a complete and consistent description is learned by a conventional learning program (AQ15). Next, the description is optimized according to a general description quality measure. This is done by a double-level search process that uses both generalization and specialization operators. The General Description Quality takes into account not only properties of BCR, but also of ICI (by measuring the complexity and accuracy of the total description). The ICI has two components: one specifies a flexible matching function, and the second specifies inference rules for handling exceptions and context-dependency. The ICI rules can be of two types. The rules of the first type extend the meaning of the concept, while the rules of the second type contract this meaning. The first type rules are employed when an instance is neither covered by the BCR (not S-covered), nor by the flexible matching function (not F-covered). The second type of rules are used when an unknown instance covers a base representation of more than one concept, or when concept membership has to be confirmed. In both cases, the rules are used deductively. An advantage of using rules for matching over other matching methods is that they can serve as an explanation why a given instance does or does not belong to the concept.

The experimental results have strongly supported the hypothesis that two-tiered concept descriptions can be simpler and easier to understand than "single-tier" descriptions. Two-tiered descriptions also perform better. For example, the two-tiered descriptions obtained for the acceptable labor managements contracts gave a performance of over 90% correct using only about 9 rules. In contrast, the best performance of a simple exemplar based method gave the 80% correct predictions on new examples and used 27 rules, and the corresponding pruned decision tree performed at 86%, and had 29 leaves (each of which may be viewed as corresponding to one rule). The system also performed better than the previous method based on the TRUNC procedure in terms of the performance (80%), but at the cost of a more complex concept description. In addition, two-tiered descriptions are relatively easy to understand, and can easily represent an explicit domain knowledge.

The presented method is different in several significant ways from the earlier method of learning two-tiered representations (Michalski, et al., 1986). The flexible matching procedure is used not only in the testing phase, but also in the learning phase. In addition to a flexible matching function, the method employs rules for extending or contracting the concept meaning. The earlier TRUNC method used only one specialization operator (rule removal), while TRUNC-SG employed in POSEIDON uses two generalization and two specialization operators. The price for that is that the new method is significantly more complex.

There are many interesting problems for future research. Some of them were indicated previously, in particular, in section 6.4. Among especially interesting and important problems is how to integrate the description optimization phase with the initial description generation phase (done by AQ). The first step in this direction is currently being investigated.

This research was done in the Artificial Intelligence Center of George Mason University. Research activities of the Center are supported in part by the Defense Advanced Research Projects Agency under grant No. N00014-87-K-0874 and N00014-91-J-1854 administered by the Office of Naval Research, and in part by the Office of Naval Research under grants No. N00014-88-K-0226, No. N00014-88-K-0397 and N00014-91-J-1351. The first author was supported in part by the Italian Ministry of Education (ASSI), and the second author was supported in part by the Natural Sciences and Engineering Research Council of Canada.

Notes

1. The system is named after POSEIDON, the Greek god of the sea, which represent fluidity and changing aspects of nature.
2. The term "exceptions" is used here in its colloquial meaning. In section 3.4, the term is given a precise meaning.
3. When negative examples are instances of another concept, as is often the case, their typicality is understood as the typicality of being members of that other concept.
4. This difference, for the labor data, is not χ_2 significant. Nevertheless, we think that there are other reasons to prefer deductive matching over flexible matching. Deductive classification is based on rules and knowledge-based inference, and is therefore easier to understand by humans. The rules may be modified locally, while changing the flexible matching function is difficult and produces uncontrolled, global consequences. In other words, examples that are correctly recognized through ICI deductive rules are also explained *ipso facto* in terms of domain knowledge. The same cannot be said of examples correctly recognized by flexible matching, which is a knowledge-independent distance measure. To reflect this, the GDQ measure assigns a higher score to a description with deductive matching than with flexible matching.
5. In our experiment, for small values of $\tau_1 = 0.02$ and .005, which emphasize the role of accuracy in the measured quality of a description, the performance on the testing set was close or equal to the performance obtained for $\tau_1 = 0.05$, and higher than performance of 86% for AQ15 in Table 8. The reason is that in the last two experiments, as well as in the original experiment in Table 8, it was always possible to find a description that was simpler than the one produced by AQ15, but still 100% correct on the training data. Therefore, by giving more importance to accuracy, the simpler description was preferred, and better performance on the test set was obtained.
6. This three-way classification of the examples is, in fact, a simple method of learning typicality. A similar feature is available in Cobweb (Fisher, 1987). On the other hand, if the typicality information is available, it is used by POSEIDON to improve the quality of the learned description.

References

Bareiss, R. (1989). *An exemplar-based knowledge acquisition.* Academic Press.

Bergadano, F., & Giordana, A. (1989). Pattern classification: An approximate reasoning framework. *International Journal of Intelligent Systems.*

Bergadano, F., Matwin, S., Michalski, R.S., & Zhang, J. (1988a). *Learning flexible concept descriptions using a two-tiered knowledge representation: Part 1—ideas and a method* Reports of Machine Learning and Inference Laboratory. MLI-88-4. Center for Artificial Intelligence, George Mason University.

Bergadano, F., Matwin, S., Michalski, R.S., & Zhang, J. (1988b). Measuring quality of concept descriptions. *Proceedings of the Third European Working Sessions on Learning* (pp. 1-14). Glasgow.

Bloedorn, E. & Michalski, R.S. (1991). Data-driven constructive induction in AQ17: A method and experiments (*Reports of Machine Learning and Inference Laboratory*). Center for Artificial Intelligence, George Mason University.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AutoClass: A Bayesian classification system. *Proceedings of the Fifth International Conference On Machine Learning* (pp. 54–64). Ann Arbor, MI.

Mooney, R. & Ourston, D. (1989). Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. *Proceedings of Sixth International Workshop on Machine Learning*. (pp. 5-7). Ithaca, NY.

Plante, B., & Matwin, S. (1990). *Learning second tier rules by chunking of multiple explanations* Research Report. Department of Computer Science, University of Ottawa.

Prieditis, A.E. & Mostow, J. (1987). PROLEARN: Towards a prolog interpreter that learns. *Proceedings of IJCAI 87* (pp. 494–498). Milan.

Quinlan, J.R. (1987). Simplifying decision trees. *Int. Journal of Man-Machine Studies, 27*, 221-234.

Robinson J.A. & Sibert E.E. (1982). LOGLISP: An alternative to prolog. In J.E. Hayes & D. Michie, (Eds.), *Machine intelligence, vol. 10.*

Rosch, E. & Mervis, C.B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology, 7*, 573–605.

Rouveirol, C. (1991). Deduction and semantic bias for inverse resolution. *Proceedings of IJCAI 91*. Sydney. (to appear).

Sammut, C. & Banerji, R.B. (1986). Learning concepts by asking questions. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. Palo Alto, CA: Tioga Pub. Co.

Smith, E.E. & Medin, D.L. (1981). *Categories and concepts*. Harvard University Press.

Sowa, J.F. (1984). *Conceptual structures*. Addison Wesley.

Sturt, E. (1981). Computerized construction in Fortran of a discriminant function for categorical data. *Applied Statistics, 30*, 213–222.

Watanabe, S. (1969). *Knowing and guessing, a formal and quantitative study*. Wiley Pub. Co.

Weber, S. (1983). A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. *Fuzzy Sets and Systems, 11*, 115–134.

Winston, P.H. (1975). Learning structural descriptions from examples. In P. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.

Wnek, J. & Michalski, R.S. (1991). *Hypothesis-driven constructive induction in AQ17: A method and experiments* (Reports of Machine Learning and Inference Laboratory). Center for Artificial Intelligence, George Mason University.

Zadeh, L.A. (1974). Fuzzy logic and its applications to approximate reasoning. *Information processing. North Holland.*

Zhang, J. & Michalski, R.S. (1989). Rule optimization via SG-Trunc method. *Proceedings of the Fourth European Working Sessions on Learning*. Glasgow, December.