Machine learning research at the Laboratoire de Recherche en Informatique at Orsay, France

ANTOINE P. CORNUEJOLS

Inference and Learning Group, Laboratoire de Recherche en informatique, Unité associée au Centre national de la recherche scientifique, n° 410, bâtiment 490, Université de Paris-Sud, 91405 Orsay Cédex, France

Received August 2, 1987

Revision accepted April 15, 1988

This article provides a brief account with sketchy technical details of the major directions in machine learning research done at the Laboratoire de recherche en informatique (LRI) at Orsay University in France. References contain publications giving details on the projects described in this paper and on closely related works.

Our research has several objectives: looking for a sound basis of the process of *generalization* from examples, using this to study conceptual clustering with automatic synthesis of descriptors; studying the nature and goodness of an *explanation* in the context of apprentice systems; and developing experimental learning systems based on these principles applied to various practical domains. The approach taken by our research group has evolved with time but is still mainly based on learning of concepts from examples using logic representations and techniques. It corresponds to a major goal of our group: to give a clear and rigorous picture, if not a theory, of the topics under investigation. Several aspects are persued at the same time: concept learning by generalization, developments of explanation-based learning techniques, analogy reasoning, and automatic tuning of the description language. These different directions are related to or stimulated by different domains of tasks: learning of rule bases, games, computer-aided teaching, learning in noisy environments, and so on. They are described in this article in the light of the main directions. The goal of a complete universal integrated system is still a far cry ahead, but as states a famous Chinese proverb: "The end lies in the way".

Key words: automatic learning, generalization, concept learning, explanation-based learning.

Cet article a pour but de fournir un bref panorama, accompagné de détails techniques concis, des principales orientations des recherches en Apprentissage automatique effectuées au Laboratoire de recherche en informatique à l'Université d'Orsay, en France. On trouvera, dans la liste des références, des publications détaillant les projets décrits dans cet article ainsi que certains travaux connexes.

Notre démarche comporte plusieurs objectifs : recherche d'une base solide rendant compte du processus de généralisation à partir d'exemples, utilisation de celle-ci pour l'étude de l'apprentissage de concepts avec synthèse automatique de descripteurs, étude de ce qui rend une explication pertinente dans le contexte des systèmes apprentis, et développement de systèmes d'apprentissage expérimentaux basés sur ces principes appliqués à divers domaines d'application. L'approche adoptée par notre groupe de recherche a évolué avec le temps, elle continue cependant à être fortement axée sur l'apprentissage de concepts à partir d'exemples utilisant un langage de représentation et des techniques d'inférence basées sur la logique. Cela correspond à l'une des motivations principales de notre équipe qui est de fournir une image aussi claire et rigoureuse que possible, si ce n'est une théorie, des sujets considérés. Plusieurs aspects sont explorés à la fois : apprentissage de concepts par généralisation, développement des techniques d'apprentissage par explications, raisonnement par analogie et raffinement automatique du langage de description. Ces multiples directions de recherche sont stimulées par différents domaines de tâches : apprentissage de bases de règles, jeux, enseignement assisté par ordinateur, apprentissage dans des environnements bruités, etc. Ces tâches sont décrites dans cet article à la lumière des principaux problèmes théoriques. L'objectif ultime d'un système intégré complètement universel est encore au-delà de l'horizon, mais comme le dit aussi un proverbe chinois fameux : « La fin est dans le chemin pour l'atteindre ».

Mots clés : apprentissage automatique, généralisation, apprentissage de concepts, apprentissage par explications.

Comput. Intell. 4, 212–221 (1988)

1. Research on the generalization process and its bias

Historically, at the Laboratoire de recherche en informatique (LRI), the primary focus of research has been on the theoretical analysis and development of general learning algorithms for the task of inductive learning from examples. In this type of learning, the system must induce a *general description* of a class of objects or patterns given independent instances or examples of it. This process is actually central to most learning activities, however different they may seem in their principles. For instance, in using analogy, you must first recognize that two different problems belong in fact to the same class, which requires being able to generalize enough of their characteristics to draw the analogy, and then you must further generalize when applying the method of one to solve the other, even though the data are different. Likewise with the so-called *explanation-based learning* (Dejong 1981; Mitchell *et al.* 1986), which is described as an alternative to learning a concept through many examples by relying instead on a large background knowledge to exploit the relevant features of a single example, generalization is very much in need not only in the exploration of the hierarchy as a version space, which is a trivial task, but far more importantly, both in the *construction* of the hierarchies (a "useful" generalization of $x = 2$, $x = 4$, and $x = 8$ to put in one hierarchy might be (Even x)) and in the elaboration of an explanation of the example at hand. This shows that *generalization* is a central piece in the learning game. It is then of the utmost importance to master its theory, and that has been a central goal at LRI for several years.

If we adopt the proposition of Dietterich and Michalski (1983) to characterize inductive learning systems along several dimensions: representation, type of description sought, rules of generalization, and control strategy, then the early efforts at LRI have dealt with the study of generalization as the search

Printed in Canada / Imprimé au Canada

for plausible general descriptions of concepts presented under the form of examples described *structurally* (various components and characteristics of the object) using *first order logic* representation (e.g., (Red A) & (Square A)). What is looked for in this paradigm is a simple general conjunctive generalization which characterizes the set of examples presented and discriminates it against counter examples. This is known as "concept learning from examples." Furthermore, the rules of generalization that were adopted at first involved only selective process (no construction of new descriptors), with *three basic technics: the dropping condition rule, turning constants into variables, and climbing the generalization tree.* The control strategy chosen was *bottom-up*, that is, from the examples to the concept without any presupposition about the likely concepts to be learned, and the method was to be general.

1.1. A generalization engine: AGAPE

Research within the framework defined above led to the definition of a generalization algorithm which culminated in the implementation of a "generalization engine" called AGAPE (Kodratoff 1983; Ganascia 1985; Bollinger 1986; Kodratoff *et al.* 1986); some essential features of which are now described.

In order to work, AGAPE must be given

- a list of the examples to be generalized, e.g.,
  E_1 : (SQUARE a) & (STRIPED a) & (TRIANGLE b)
  E_2 : (SQUARE c) & (STRIPED d) & (TRIANGLE d)
- a hierarchy of the descriptors (predicates) used, e.g., SQUARE is-a POLYGON
- a set of axioms about these descriptors, e.g., (NEAR xy) $\Leftrightarrow$ (NEAR yx)
- theorems defining the semantics of the representation language used, e.g.,
  Idempotency: $A \Leftrightarrow A \,\&\, A$
- control heuristics.

They are the main objects of the following.

One finds therefore that an important set of formal knowledge is necessary in addition to the simple description of the examples. Particularly the generalizations found by AGAPE are very dependent on the *hierarchies* given to the system. But the choice of the *heuristics* provided to guide the generalization process has also proven crucial. For instance AGAPE favors systematically the use of idempotency over other theorems, which biases the concept obtained. The clean way in which AGAPE has been designed helps to make explicit the relationship between a given generalization and the aspect it favors.

The generalization proceeds in four steps in AGAPE: *structural matching*, adjunction of *bindings* between variables, *elimination* of everything that differ from one expression to another, and *simplification*. One of the worries when designing AGAPE was to clearly separate the deductive procedures (consisting of successive equivalences) from the inductive ones (consisting of generalizations). In this way, it appeared that, contrary to what is commonly thought, a generalization algorithm can be mainly a deductive process.

The main idea behind AGAPE and the other systems, OGUST and MAGGY, that stemmed from it is that *concept discovery is first of all the discovery of relevant variable bindings.* The role of generalization is the discovery of the links between the components of each example that are significant and those that are not, and therefore during generalization, information should be dropped from a formula only with extreme care.

This entails a whole lot of consequences. First, if all links are to be detected, then the dropping condition rule must be used very cautiously, otherwise important information about the concept may be lost. Second, a side effect of avoiding the dropping condition rule ($A \,\&\, B \Rightarrow A$) is that no disjunctive generalization can be obtained from the usually conjunctive form of the examples because a formula containing a disjunction describes two concepts rather than one, unless strong links between the variables of the two disjuncts are found. Third, the discovery of the links can be made using deductive procedures and a very restricted definition of generalization.

At this point it might be useful to delve a little bit into the technical details of the algorithm, even though the limited space does not allow us to give a full account of the system and its ramifications. See Kodratoff and Ganascia (1986), Vrain *et al.* (1986), and Vrain (1987) for a more detailed treatment.

Structural matching is one of the essential point that makes AGAPE's originality. Contrary to other programs that learn from similarities and make an intensive use of the dropping condition rule (Michalski 1983), AGAPE attempts to "fill the gaps" (when they exist) between the expressions to generalize.

For instance in the following example:

E_1 : (SQUARE a) & (RED a)
E_2 : (SQUARE b) & (SQUARE c)

we notice that there is a gap between E_1 and E_2 (one SQUARE is missing in E_1 and no element of E_2 can match RED). By using the dropping condition rule, the system would produce the generalization (SQUARE x). AGAPE will try to make a predicate appear in E_2 that can match RED and another one in E_1 that can match SQUARE. This will be achieved by applying the idempotency of & and the axioms. In this case the idempotency of & allows AGAPE to transform E_1 into

E_1': (SQUARE a) & (SQUARE a) & (RED a)

Then, assuming that we have the axiom:

(SQUARE x) $\Leftrightarrow$ (SQUARE x) & (RED x)

E_2 will be turned into

E_2' : (SQUARE b) & (SQUARE c) & (RED b)

(According to the heuristic chosen we could alternatively have produced (RED c) in E_2.)

Both expressions will now match structurally and will eventually be generalized to: (SQUARE x) & (SQUARE y) & (RED x) with the binding (x may be the same as y). This expression is better than the one resulting from the dropping condition rule because it preserves more information.

When the use of the idempotency and of the axioms does not enable AGAPE to fill the gap (failure of the structural matching), the system resorts to the dropping condition rule during the final step (elimination of differences).

The above algorithm uses implicitly a new definition of "generalization," where the dropping condition rule is no longer the only valid rule of generalization. But that definition might be insufficient, even though it allows to generalize in the first-order logic framework, because it does not prescribe how to get rid of the useless terms to reach the minimum meaningful generalization. This is the reason why a careful study of the

role of counter-examples and the definition of an operation of *overgeneralization* has been undertaken (Ganascia 1985; Kodratoff and Ganascia 1986; Kodratoff *et al.* 1986).

There are two cases: either the counter examples are very different from the examples (*far-miss*) or they are very similar (*near-miss*). In the first case, the predicates are sufficient to distinguish between the examples and the counter examples, and the overgeneralization can be built from the predicates only (Vrain 1987). On the other hand, in the case of near-miss (Ganascia 1985), a finer discrimination is needed using the occurrences of the variables in the descriptions of the examples. This is done using a structural matching of the examples and the counter examples. One gets a formula F common to the examples and the counter examples. The distinction bears on the links. For each example E_i has a set of links, lin(E_i), between the constants and the generalization variables introduced, and the same, lin(CEj), for each counter example CEj. A complete description (i.e., covering all the examples) G of the concept to be learned is composed of the formula F together with the links that are common to all the examples. In order to obtain also a generalization that rejects the counter examples, one has to study the discriminating links between the generalization G and the links of the counter examples. Once the discriminating links (corresponding to the relevant variables) have been singled out, overgeneralization can be done by eliminating all the components of the description where no variable belonging to a discriminating link appears. In this way, "near misses" (Winston 1975) can help also to find explanations for the validity of the generalization.

This use of the counter examples differs from the more traditional one where they are only used for particularization. In addition, this approach points out that another role of the counter examples may lie in their guiding of the heuristics for the choice of the constants that will be taken as generalization variables. The study of some of these heuristics is one of the central themes of the system that we turn to now.

1.2. Generalization with stronger theory domain: OGUST

OGUST (outil de généralisation utilisant systématiquement les théorèmes/generalization engine using theorems systematically) improves on AGAPE on several aspects and illustrates also the impact of the use of different heuristics to guide the generalization process (Vrain 1987).

As AGAPE, OGUST relies on the principle of "structural matching" to compare and store elements of the descriptions of the examples and the counter examples. But it brings new sources of power to do it.

OGUST's first concern is about the choice of the constants that must be matched to get a generalization. In most cases the comparison of all possible combinations would be prohibitive in terms of the computations involved. Besides, we also need a basis on which to establish such a comparison. In OGUST, this basis is provided by all the knowledge available on the domain: taxonomic hierarchies such as in AGAPE, by theorems between the predicates used to describe the domain, and, when possible, by the types of constants and the order defined on the predicates or classes of predicates.

OGUST's originality rests also on the integration principles and techniques from both of the two main approaches to machine learning: *similarity-based learning* exemplified by the research of R. Michalski and others on INDUCE (Dietterich and Michalski 1983) and *explanation-based learning* (Dejong 1981) pioneered in particular by T. Mitchell and coworkers with the system LEX (Mitchell *et al.* 1983).

From the first approach, OGUST retains the goal of learning concepts from examples (bottom-up) by discovering discriminating functions of classes of objects presented. From the second one, OGUST uses operationally the concept of explanation to guide the process of generalizing. More to the point, the system OGUST is able to discover recognition functions of a world of objects in conjunctive form in domains where, contrary to previous SBL systems, an important part of the knowledge is expressed in forms of theorems between the predicates rather than by hierarchies on them. That means that the system can evolve in domains with stronger theory background and use effectively this knowledge in its generalization. To achieve this without being overcome with the exponential explosion of all the combinations of theorems applicable to the situation that would beset a classical theorem prover, the system generates first "loose" explanations about the generalization that can be obtained from the available knowledge of the domain: kinds of properties that look promising to describe the world and the theorems likely to be useful in this task. And then, OGUST, based on these guidelines, realizes the actual proof of the generalization that will allow substantiation of the former "explanation."

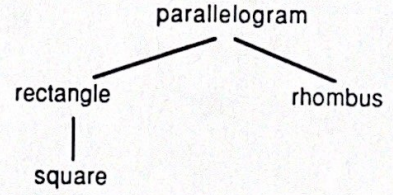
In this short presentation of the system, we don't insist on the mechanism by which theorems are used to remove all discriminating occurrences of the variables during generalization, rather we prefer to give an overview of the heuristics on which OGUST bases its choice of the constants to match while generalizing.

This choice is indeed very influential on the generalization eventually obtained, even though the links between the constants and the generalization variables generated keep most of the information gained from the examples. Moreover, an exhaustive exploration of all the possible combinations is rarely realizable. This is why OGUST attempts to attach a merit measure to each matching. This measure results from the consideration of the number of common characteristics between corresponding objects (called identity number, or ID) and a number representing the total similarity of the other descriptors of the constants (called potential similarity, or SIM). The computation of these numbers is affected by the known relations between the descriptors, like taxonomies and theorems. This gives a similarity measure denoted as (ID, SIM).

For instance, consider the following two examples:

E_1 : (RECTANGLE a) & (RED a)
E_2 : (RECTANGLE b) & (ON c b) & (RHOMBUS c) & (RED c)

given with the following taxonomy:



In E_1, there is no choice but a, whereas in E_2 there is the choice between b and c. A comparison must then be done to measure if the similarity between a and b is greater than the similarity between a and c or not. Here we get (2,0) for a and b. Indeed, both appear in the predicate "rectangle," while there is no relation between "on" and "red." We get (2,1/2) for a and c since a and c have a common descriptor "red," and

there is a relation through the taxonomy between "rhombus" and "rectangle."

The details of the computation of (ID,SIM) depends on a lot more intricate aspects than can be described here. It offers a useful tool to evaluate at each step, but specially during the first ones when the choice is most open, the most promising constant matching.

Another source of powerful heuristics is the use of counter examples, either after a generalization of the examples has been obtained or during the generalization process, and not only when afterwards as is commonly the case.

The special treatment of the counter examples studied by Christel Vrain goes beyond the method developed by Jean-Gabriel Ganascia in that it covers also the case where the counter examples are very general—a situation where an undifferentiated matching is inappropriate—and allows to choose relevant counter examples during generalization to constrain the choice of constant matching.

Two methods have been considered. In the first one, counter examples are compared with the generalization G obtained from the examples in order to determine rapidly if one counter example is covered or not by G and then to modify G correspondingly if necessary. Finding which concept has been too generalized in the definition of G is not a trivial task and requires again knowledge about the similarities and dissimilarities between G and the counter examples. In the second approach, a natural following of the first one, for each combination of constant matching, an estimation of the resulting generalization is generated and is then compared against counter examples. The most promising combination is then selected for the next step in the generalization process. Several heuristics have been developed to determine which set of counter examples to consider and which constants should be chosen. We refer the interested reader to Vrain (1987) for the full study of these problems.

Two last remarks deserve to be made. First, by examining numerous possible combinations of constants at each step and measuring their relative merits, OGUST is able to generate explanations of its reasoning. They will be used both in the generalization process as we have seen and in the interaction with the user, a noticeable advantage of the method. Second, the integrated treatment of the counter examples during generalization opens ways to realize incremental learning, especially thanks to the explanations generated hereby. New works on this, on the use and learning of contexts (to limit the extent of the knowledge to be analyzed in each situation), and on the use of analogical processes are under way to further extend OGUST's capabilities.

1.3. MAGGY

One thing that might be baffling at first when observing AGAPE or OGUST generalizing a set of examples is the apparent multiplication of "fictitious objects" that stem from the use of structural matching with the subsequent storage of the links between variables and constants. For instance (see Ganascia 1985 and Vrain 1986), the generalization of Fig. 1 could be

G_1: There is a small cube on a big cube

or

G_2: There is a black cube and a cube with stripes

The data-driven algorithm AGAPE introduces fictitious objects in order to avoid having to choose between the two generalizations, thus losing informations. Therefore it finds

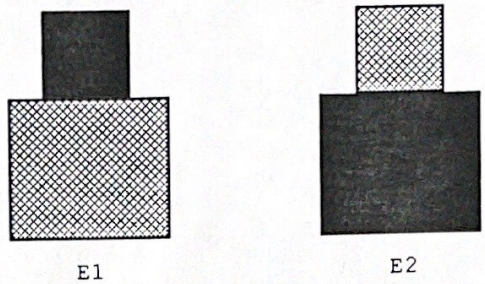


FIG. 1. Two examples from the world of blocks.

"There is a small cube on a big cube, a black cube and a cube with stripes" (with appropriate May-be-the-Same links). But giving a generalization under conjunctive form is not always appropriate. The user may prefer a generalization based on the size or on the color depending on the context of utilization. Therefore, providing disjunctive forms of a generalization might sometimes be better. It is possible to reduce the learning of a disjunction to the learning of a conjunction by modifying the language of description and thus make use of the same techniques as the one described earlier, but this often renders the resulting descriptions opaque to human experts. This is why Michel Manago developed the system MAGGY (see Manago 1986). With the object-oriented structural matching à la MAGGY, both generalizations are generated and the system relies on the model to decide whether the size is more important than the color or the contrary. Particularly, one crucial concern in the conception of MAGGY has been its integration in operational learning systems where it is important to take into account knowledge about the noise in the descriptions and the fuzziness of the concepts present in the domain at hand.

Indeed, as soon as one is designing expert systems for real-world applications, the problem of handling noisy data, uncertain features, and errors arises. The traditional approach consists in attaching numerical coefficients, such as certainty factors, to the rules. In our laboratory, we feel, however, that this methodology does not solve all the problems related to noise and generates further difficulties. We see noise as a critical problem that must be dealt with during the knowledge acquisition process.

Different kinds of noise have been particularized, such as *unreliable information* when the noise originates from the low-level concepts themselves (e.g., "light gray" and "gray"), *wrong information*, and *incomplete information* (see Manago and Kodratoff 1987). One of the techniques of interest, and the most widely used to cope with noisy data when numerous examples are available, is the one popularized by Quinlan (1983) with the system ID3. It consists in producing a decision tree that discriminates the classes by minimizing the entropy of the resulting partition. The problems with this otherwise powerful approach are (1) that it is unable to take into account background knowledge of the domain (other than the one implicitly contained in the examples), (2) that it cannot usefully manipulate and take profit of structured objects such as frames, (3) that it has no clear links with the version space algorithm (Mitchell *et al.* 1983) and other more symbolically oriented generalization systems like AGAPE or OGUST, and (4) that it relies on the assumption of feature independency, an hypothesis that might need qualifications in some applications. MAGGY has thus been designed to be integrated with such a

system and bring with it the capabilities that were absent from ID3-like systems.

MAGGY's main characteristics are the following. Given a set of objects, it outputs a most specific conjunctive generalization using first-order logic as internal as well as external representation. (This means, in particular, that incremental learning should be eased since inputs and outputs share the same formalization.) The domain knowledge can be expressed in the structures of the objects manipulated (expressed as frames) and in simple axioms and constraints. The generalization procedure is similar to AGAPE's, but, in addition, it provides a rough measure of how much information was lost during the process. This can then be used by the human expert to guide further generalizations. The project to integrate MAGGY to an ID3-like system has been decomposed into two phases. In the first one, MAGGY was installed on top of the NEDDIE system (a descendant of ID3 developed at the Marconi Research Center of the General Electric Company by Jim Blythe and Rob Coolett).

Given a set of examples that may belong to different classes, NEDDIE produces a decision tree that discriminates the classes. This tree is built using criteria such as maximizing the separation between concepts with the most discriminating features being toward the root of the tree, and minimizing, i.e., pushing as deep as possible in the tree, the features that are inherently noisy such as "white" and "yellow" or "big" and "very big," which are concepts that overlap and are difficult to determine with precision. A novel aspect of NEDDIE compared to ID3-like systems is that it also tries to maximize the homogeneousness of the clusters of examples thus obtained at each node of the tree, so as to facilitate their explicativeness in terms of user's semantic concepts. The influence of a given example is relatively low and does not deeply affect the resulting decision tree. As similar statistics-based systems, NEDDIE is thus inherently robust to noise and errors. It is, however, hardly able to produce understandable rules and to give a semantic meaning to the isolated concepts. MAGGY, on the other hand, can take these clusters, translate them in its own symbolic description language (provided by the user), and output significant generalizations of the data which maximize the explanation ability of the individual clusters.

Thanks to its object-oriented representation, its ability to deal with tangled hierarchies and inheritance, and default and procedural attachments, MAGGY can modify the decision tree output by NEDDIE accordingly to the semantics of the domain it holds.

These generalizations take the form of disjunctions built from the leaves of the tree generated by NEDDIE, where MAGGY tries to maximize the number of disjunctive terms inside each generalization and to further reduce the discriminations based on inherently noisy features.

Presently, however, there is no integrated interface between the two systems, NEDDIE and MAGGY; the former system cannot take advantage easily from the knowledge held by the latter during the tree construction. In the same way, whatever knowledge about the interdependencies between the features MAGGY might have does not affect NEDDIE's process.

This is the reason why a second phase is now under way to realize the full integration of the two systems. In a longer-term research perspective, it is planned to improve the capability of the system to handle noisy observations with an incremental procedure.

2. Automatic synthesis of symbolic descriptors

The works described above can be seen as belonging to a "learning description" approach where one is trying to find general descriptions of sets of patterns from syntactic manipulation (hopefully corresponding to the semantics of the domain). Another way of considering learning from examples emphasizes the notion of pattern recognition and the construction of decision trees to achieve this. There the goal is rather to determine good discriminating features and to build a taxonomy of the observed objects organized around tests on these features. In that way, performant recognition procedures can be realized and significant concepts be discovered that correspond to important nodes of taxonomy.

Presented this way this approach is not new to the pattern recognition field, it is in fact akin to the ID3 approach alluded to before, but its range differs when one introduces semantic considerations and measures instead of only dealing with numbers and syntax. The research described in this section exemplify this as they are striving to happily combine numeric with symbolic processing and data-driven with model-driven strategies. In contrast to the previous section, the stance here is on the discovery of new concepts or useful macro-descriptors belonging to a given universe rather than on the determination of good generalizations and partitions.

2.1. Learning to recognize real-world scenes

We are starting a project aimed at the recognition of objects in real-world images. As this difficult problem requires the integration of very low level processing up to high level learning and reasoning abilities, it is expected that the project will play a federative role for the other works done in the LRI-IA team. Nevertheless, the main thrust will be centred on the *automatic generation of optimum decision trees for recognition.* This will be performed *in an incremental manner* with each new example capable of triggering modification of the tree. Particularly, a promising technique (see Kodratoff and Lemerle-Loisel 1984) will be adapted to and tested on real-world images. It uses "differentiating resemblances" in order to get the "most promising partition" obtained from a suitable combination of near-misses. The recognition tree is obtained by considering each of these combinations of near-misses as a node of a tree, which partitions the learning set in a most useful way with, for instance, the left son of a node containing examples this near-miss belongs to, and the right son containing all the other examples. Some special cases must be handled with care, but the method has been proven to work well and to be resilient to some forms of errors. One important feature is that, in trying to determine discriminating near-misses, one may have to provide more details at specific places in the description, a process that amounts to a change of description language.

Some difficulties arise, however, in universes where a priori providing a significant description language is hard (this is a prerequisite for the algorithm discussed above) and where the notion of near-miss is severely blurred by the unavoidable presence of noise in the data. The last point in particular will deserve attention in the future.

2.2. Conceptual distance and classification

Augmenting the description language by learning new concepts can be studied as well from another point of view, one that is to classify a set of examples already expressed with a

high level description language, and to use the resulting clusters to produce new significant concepts and descriptors for the domain at hand. The problem then is to realize a classification that is coherent both with the examples set and with the already available high level description language. This is called *conceptual classification*. The method here consists in recovering and using most of the well-known techniques of data analysis, but replacing the traditional distances between examples by "conceptual" or semantic ones. Thus, measures of internal similarity (actually a vector defined over the descriptors) will be based, for instance, on the common descriptors of a set of examples and on a certain evaluation of the difficulty to generalize these examples (based on procedures closed to the ones of AGAPE); and measures of dissimilarity or of discriminating power will be based on idempotency and dropping coefficients. Near-misses between the examples will be used as well and is most important in the definition of the final generated concepts and in the discrimination between several points of view or micro-worlds.

The algorithm developed so far (Benamou 1986), written in Lisp, will generate, starting from a set of examples described with a certain description language, a set of classification trees, clusters of objects that are significant in conceptual terms, and possibly some rules or links between descriptors or a same cluster. At this stage the algorithm is to be taken as a tool to help conceptualization of a domain and must be used in conjunction with a practitioner of the field who may judge the validity of the proposed hierarchies.

2.3. Learning heuristics and rule systems: CHARADE

CHARADE is a complete self-contained operational system developed by Jean-Gabriel Ganascia (Ganascia 1987*a*, *b*). CHARADE learns a coherent *system* of classification rules from a description language, the axioms that express the semantics of this language, and a set of examples described with this language. The originality and interest of this research is twofold. First, it states most clearly that to be useful and operational *the rule set* of an expert system must be endowed with global properties such as coherency, nonredundancy, and good chaining characteristics and must therefore *be thought of as a whole* and not as only a collection of isolated rules. This increases the performances of the system and eases the process of increasing or modifying its rule base. Second, this work innovates by using a clever organization of both the descriptors and the examples at hand, by way of an *hypercube*, to guide and reduce the search for a minimal and coherent set of rules. The following elaborates on this point.

Most rule learning systems from examples are limited to the acquisition of descriptions of concepts from examples and counter examples. This leads to the exploration of the whole example space for each concept that is to be learned and, moreover, yields a collection of poorly related rules. CHARADE, on the other hand, considers the example space globally and explores the possible conjunctions of descriptors through a well-suited knowledge structure called an Hilbert cube (i.e., a boolean lattice). It is an n-dimensional cube where n is the number of examples, and where each vertice is a conjunction of descriptors. An order relation corresponding to the generality and inheritance links among examples and concepts can be immediately defined on this structure, and allows to use very efficient search techniques to find a coherent set of rules describing the example set. In addition to that, the Hilbert cube is a very sparse way of storing the data base.

But CHARADE provides the user with more powerful tools to both constrain the search procedure with respect to the task and the situation at hand, thus yielding different possible sets of rules, and act on and modify if necessary the very description language itself. In particular, several parameters such as the minimum number of examples that must be subsumed by a conjunction of descriptors to be considered for a rule, or the plausibility factors linked to inherent uncertainties in the examples sample and allowing the acquisition of approximate rules to limit the effects of noise in the data, make possible a fine tuning of the system to the context. Furthermore, indications from CHARADE on the examples hard or impossible to classify at any stage of the procedure may provide significant clues to the user over the characteristics of the language description that could profitably be changed.

To sum up, CHARADE may be considered as a first step toward the construction of a bridge between classical data analysis techniques and symbolic learning methods, a characteristics it shares with the work of Benamou described in Sect. 2.2. Above all, it contributes, through a systematic study of rule sets and of their adequacy to the context, to evaluate the description language proposed and possibly to transform it. This is the aim of current and future efforts. Presently CHARADE has been tested on several domains such as tomato diseases, first announce in the game of bridge, galaxies recognition, or classification of archeological objects.

3. A learning apprentice system in weak theory domains: DISCIPLE

Expert systems have proven useful in every domain where the knowledge can naturally be expressed with rules, but, up to now, their application has been strongly limited by their lack or poor ability to acquire knowledge and learn by themselves. One promising approach to this problem is the creation of learning apprentice systems (LAS). These systems are able, while helping the human expert with their available knowledge, to observe, analyze, and question the expert reasoning and expert's solution if provided.

A LAS deals with a given problem by using its own knowledge, although often incomplete, and presenting all the steps of its reasoning to the expert. In the case of a failure to find a correct solution, the user can provide one of his/her own. The system then try to "understand" why this is a solution to the problem, and eventually, when successful in its understanding, will generalize it or be able to apply it to new, similar, problems. That way, a LAS is able to help a user even with a still incomplete knowledge of the domain and to build its own knowledge base in the process of observing the user's behaviour.

Nevertheless, a LAS, relying on the principle of explanation-based learning to analyze and generalizing from single examples, needs a lot of formal knowledge on the domain, what is called by T. Mitchell and coworkers a strong theory domain. Four kinds of information must be specified:

- the *goal concept* which defines the concept to be acquired. For instance (Mitchell *et al.* 1986), "to recognize pairs of objects $\langle x,y \rangle$ such as that it is safe to stack x on top of y";
- the *training example* that is a positive example of the goal concept;
- the *domain theory* which includes a set of rules and facts

that allow explaining how the training examples are members of the goal concept;

- the *operationality criterion* defining in which terms the output concept must be expressed.

One method is *to explain* through the domain theory how the training example, for instance a solution provided by the user, satisfies the goal concept definition, and then *to generalize* this example by regressing the goal concept through the explanation structure.

However, domains where the goal concept and the domain theory can be precisely defined are seldom. This is why one of the effort LRI has been to design an apprentice system able to learn in less well formalized domains. The chosen approach combines learning by analogy and learning by generalizing instances and relies on a careful organization and utilization of the domain knowledge, on special heuristics and on the user supervision.

DISCIPLE is equipped with two problem-solving operators: decomposition rule of problems into subproblems and specialization rule. Its knowledge of the domain is embedded in a semantic network with various types of links that can be augmented while functioning. In a normal session, DISCIPLE, confronted with a given problem, will try to solve it using its operators expressed in the form of rules. If it fails it waits for the user solution that it considers as an instance of the rule to be learned.

For instance, suppose that DISCIPLE confronted with the problem:

attach sectors on chassis-membrane-assembly

needs help from the user, and that the user provides a solution in the form of

apply mowicoll on sectors
press sectors on chassis-membrane-assembly

Then DISCIPLE will consider the new rule:

attach sectors on chassis-membrane-assembly
⊢apply mowicoll on sectors
  press sectors on chassis-membrane-assembly

as an instance of the rule to be learned:

If x,y, and z satisfy ⟨constraints⟩
then
  attach x on y
  ⊢apply z on x
    press x on y

That is, rule learning is then reduced to discovering the constraints on the x, y, and z variables.

In a first stage, DISCIPLE looks for plausible explanations of the validity of the user's solution. This is the *explanation-based mode*. Then DISCIPLE enters its *analogy-based mode*. It will explore its semantic network to determine the possible relevant links between the items specified in the user's supplied rule. If there are none, it will ask for more explanations from the user; if there are many, it will, by way of "clever" questions, try to isolate the significant ones in the given context. These questions are in fact statements of other similar candidate rules obtained from variations from the user's one through exploration of the semantic network around the specified items. In a way the system tries to establish the foundations for an analogy between the current situation and its rule base. For each of these rules the user has to characterize them as examples or counter examples of the general rule to be learned. In a subsequent stage, DISCIPLE will generalize, from the positive instances found, as much as possible under the constraints expressed by the negative instances. In our example, that would yield for instance (given a certain semantic network),

if (z is a adhesive) & (z glues x) & (z glues y)
then
  attach x on y
  ⊢apply z on x
    press x on y

It must be noted that the generalization undertaken in DISCIPLE is not of the careful but costly type of the AGAPE kind, but rather very elementary: turn constants into variables by giving the same variable name to all the occurrences of the same constraints. This is of course only an approximation and does not provide necessary conditions. Improvements are planned to refine this definition.

The idea behind the use of explanations for learning is that an explanation points to the relevant features of the objects, that is, to the features that are to be kept in the generalization. But DISCIPLE uses this method in a quite original way. It uses the explanations for generating possible positive instances of the rule to be learned, which allows also a much more easy exchange with the user who is not obliged to cast his/her expertise in the own terms of the system. DISCIPLE is then able to use a similarity-based learning technique to generate the final general rule.

This apprentice system has been tested in the design of industrial components, but it is still in a preliminary version (Kodratoff and Ganascia 1986; Kodratoff *et al.* 1986; Kodratoff and Tecuci 1986*a*, *b*, 1987), and much more powerful mechanisms are under study to improve its performances and remedy some weaknesses. For instance, in the present state, DISCIPLE generalizes an explanation by replacing the specified constants, linked to the particular problem at hand, by variables the domain of validity of which will then be explored. DISCIPLE does not realize any "structure generalization" in its semantic network, nor does it perform goal regression to abstract from the user's specific proof a sufficient general condition (getting a necessary one is still another problem). These have to be implemented in future versions of DISCIPLE. In this respect, DISCIPLE is still close to an instruction-based learning system (Michalski 1983). Furthermore, it would be quite interesting to have DISCIPLE able to draw intelligent analogies from the given example and explanation to generate instances of positive rules rather than just generalizing and specifying from the particular explanation provided. In order to do that, the nature of analogy between objects and actions and, above all, between the explanations themselves have to be more fully understood. And this is the focus of current research to improve on DISCIPLE.

4. Explanation-based learning in two-opponent games

Learning plans in two-person games introduces *new issues in the use of explanation-based learning (EBL) methods* because the presence of an opponent precludes the direct use of classical goal regression techniques. It is therefore quite interesting to delve into the specific problems raised in this particular domain. This is the research topic of Jean-Francois Puget (1987).

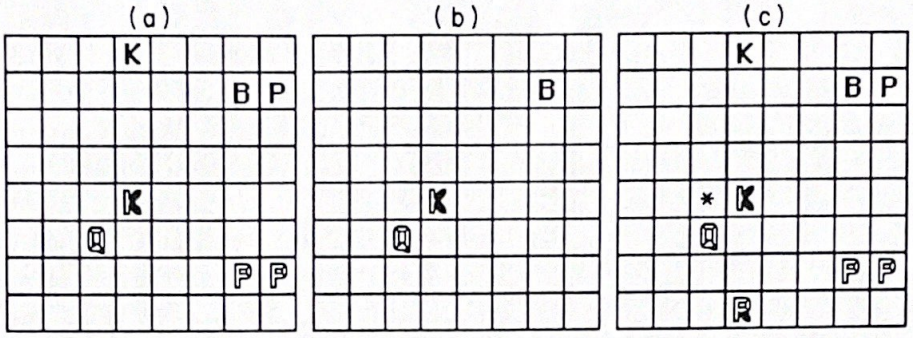


FIG. 2. Examples of chess positions: (*a*) a position; (*b*) the regressed pattern; (*c*) a counter example.

As was seen in the previous section, in traditional EBL, when an example of a solution is provided to the system, the latter tries to find a proof or explanation that the example actually solves the problem, and then generalizes from this particular proof using a method called *goal regression* in order to find a sufficient condition for the goal to be achieved. However, when there is an opponent, the problem of proving that a goal can be achieved becomes somewhat different because it must be proved that the goal will be achieved *against every possible defenses of the adversary*. And this makes the search for a sufficient condition difficult if not impracticable. This is best seen with the following example taken from a chess game. Here, in the position shown in Fig. 2*a*, the goal of black is to take the white queen with the bishop. White is to move and must take its king out of check. The position obtained through goal regression is that of Fig. 2*b*, i.e., the king has to be between the bishop and the queen. This seems to be indeed a sufficient (and necessary) condition for black to win the white queen. But this pattern is not a sufficient condition in general as shown in Fig. 2*c*. This position verifies the pattern, but this time if white king moves to the square marked *, the combination fails as the black king is in check.

A solution to this problem would be to add futher conditions to the regressed pattern, such as "white can be out of check only by moving his king *and* no other white piece can prevent the combination." But this requires a powerful description language and associated deductive ability, and weakens the original aim which was to reduce search as much as possible in game playing and learning.

The goal is thus to reduce the search needed to chose a move by *transferring the need for search*, which is the usual way in automated game players, *to pattern recognition* used by human experts. This is done using goal regression as stated above together with a *weak sufficiency* condition which amounts to considering only the moves that were present in the example provided to the system during goal regression. The point is indeed precisely that only by relaxing the sufficiency constraint, useful goal regression can be performed. Of course, as seen in Fig. 2, the plan thus obtained may fail sometimes. This is why an augmented responsibility is assigned to the problem solver, which must check for goal interactions during the game playing before applying plans it has *recognized* as suited in the current situation.

This research therefore underlines problems of weak sufficiency for certain goal regression processes that seem central to numerous problem-solving contexts and gives new operationality criterion for EBL. It suggests also some kind of *structural generalization* in that the structure of the proof itself is generalized during goal regression. Applying EBL in game playing allows to enlarge previous works on plan learning (Pitrat 1976) and regression of winning moves by Minton (1984). The refinement of already known plans and the use of the symmetry between the situations of the two players are under study.

5. The project INFLUENCE: dynamic data base and explicativeness

INFLUENCE is a system that stems from early efforts to study joke understanding and ambiguity resolution. To this end it uses mechanisms similar to the ones found in nature when, for instance, a molecule spontaneously rearranges its configuration when sollicited by new constraints. The focus here is not on constructing and modifying the building blocks of the semantic part of the memory (available concepts and types of links), but on the process by which can occur the *modification and revision of an "interpretation,"* or model, in the *episodic memory* represented as a semantic network. This implies nonmonotonic understanding and the judicious retraction and modification of erroneous inferences.

Understanding and inferencing, in a frame-based system, result in attachments or links between slots in some frames and some other frames. Retracting inferences amounts then to be able to remove corresponding links and to build new ones. This is what realizes the system INFLUENCE (Cornuejols 1987). In that approach, the memory is seen as a kind of self-organizing structure where at each moment links can retract themselves and connect to other targets depending on informations local to its situation in the semantic net. These informations reflect the goodness or the "fitness" of the current interpretation.

In this model, possible interpretations of a given set of inputs are seen as local equilibrium states. Reinterpretation occurs when pertubations, either external such as new incoming data or internal akin to day-dreaming, push the system to a new minimum of the potential function.

Although this project is removed from other ones at LRI, it can provide some interesting glimpses on several problems common to all. First, INFLUENCE is inherently concerned with the *adaptation* of a current theory or concept to new incoming informations. Therefore it might be potentially illuminating to confront its methodology to the incremental learning problems encountered in the other projects. First attempts and analyses show that the approaches are very foreign to each

other. Particularly, since INFLUENCE is prone to forgetting and uses only local informations, there is no guaranty of coherence or optimality of the resulting theory. On the other hand, the algorithm is much simpler and is real-time oriented. More work is in order to compare the two methods.

Second, INFLUENCE can be considered in a way as a compulsive hypotheses tester, since it tends to constantly put to test the weakest assumptions (i.e., links) of the knowledge base. And this in turn could be translated into questions from the system to the human user. It thus offers some possible interrogation mechanism for DISCIPLE-like systems, or for computer-assisted teaching devices. It also requires a better understanding of what constitutes a good explanation (to the system).

Those are the reasons why the system INFLUENCE, while inspired by different concerns than the other projects described in this paper, finds its place among them. More symbiosis should emerge as the project evolves toward its full maturity.

6. Some hands must be dirty or the confrontation with the real-world

Let us confess it, all the works cited above are, in more than one way, the easy and pleasant part and are rather clean exercises of style giving birth handily from time to time to gratifying papers. But a lot of the actual job must be done under the deck, in darkness, with the hands in the grease grappling with real-world tasks and trying to cast them into a framework that makes them amenable to the techniques developed in theory. Everyone at LRI is more or less spending some time away from the light, but some, charged with the more applied projects, have paler faces and dirtier hands.

As we have seen, a lot can be learned from difficulties arising in real-world applications. For instance, the system MAGGY and its integration with other learning methods stemmed for a large part from such a confrontation to the application world. Another such big project, which Jean-Jaques Cannat is in charge of, is learning and automating air traffic control systems. For several months now, the problem has been the discovery of an adequate set of descriptors and possible knowledge representation schemes. But a solution is not yet at hand because two difficulties, at least, compound each other. One is that the importance and semantics of the descriptors (such as speed, height, and so on) is very context-dependent. The second is that the events themselves, for instance planes in the sky, must be described with respect to each other, which makes that some aspects important in some situations will be quite irrelevant in others and vice versa. The discovery of the interdependencies among factors is quite difficult and, in the present stage, cannot yet be assisted by an automatic system, or an apprentice! This project, however, should bring a lot of new important issues to the attention of the learning theorists, the "above deckers," and is the object of their solicitude.

7. Conclusions

After an initial stage in which the goal was to understand and clarify the theoretical foundations of learning concepts by examples, the scope of our research at LRI has now broaden to encompass a great deal of the learning tasks that require at some point a clean generalization procedure, such as explanation-based learning and analogy making. All through these works a key notion has emerged, that of "explicativeness." It is indeed our belief that the ability of a system to give a conceptually understandable explanation of its behaviour is central both to its *AI status* and to its *efficiency* as a genuine partner to human users, but more fundamentally to its very *learning capabilities*. The *concept of explanation* itself is therefore the focus to a great extent of our current research and is hoped to bring new power to bear on such problems as learning in noisy environments or in weak theory domains, discovering analogies and producing multi-agents learning, each agent exchanging explanations of its behaviour and domain of competence with its counterparts.

Acknowledgements

It is a pleasure to thank all the members of the Learning Group in Yves Kodratoff's team at LRI. Their help was essential, and there would not be anything to report on without them. These proud "apprentis" are Serge Belhassen, Norbert Benamou, Jean-Jacques Cannat, Sylvie Duchenoy, Béatrice Duval, Jean-Gabriel Ganascia, Nicolas Graner, Michel Manago, Stéphane Moscatelli, René Natovitcz, Jean-François Puget, Christian Roy, Ba-Phuong Tran, and Christel Vrain. A special mention is due to Yves Kodratoff, who is the heart and inspirer of our research group at LRI. Finally, the careful review of one of the reviewers was very useful in polishing an earlier version of this paper. All remaining imperfections are mine, though.

BENAMOU, N. 1986. Conceptual hierarchical ascending classification. Internal Report, Laboratoire de Recherche en Informatique, Orsay, France.

BOLLINGER, T. 1986. Généralisation en apprentissage à partir d'exemples. Thesis, Paris-Sud Orsay University, Orsay, France.

CORNUEJOLS, A. 1987. Revision of "interpretation" in episodic memory by using chemistry instead of reason maintenance systems. Proceedings, MARI-87, Paris, France.

DEJONG, G. 1981. Generalizations based on explanations. Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, B.C., pp. 67–69.

DEJONG, G., and MOONEY, R. 1986. Explanation-based learning: an alternative view. Machine Learning, **1**: 145–175.

DIETTERICH, T. G., and MICHALSKI, R. S. 1983. A comparative review of selected methods for learning from examples. *In* Machine learning: an artificial intelligence approach, vol. I. *Edited by* R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. Morgan Kaufmann, Los Altos, CA, pp. 41–81.

GANASCIA, J.-G. 1985. Comment oublier à l'aide de contre-exemples. Congrès AFCET-INRIA 1985, Grenoble, France, pp. 145–159.

———1987*a*. AGAPE et CHARADE: deux techniques d'apprentissage symbolique appliquées à la construction de bases de connaissances. Thése d'État, Paris-Sud Orsay University, Orsay, France.

———1987*b*. CHARADE: a rule system learning system. Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy.

KODRATOFF, Y. 1983. Generalizing and particularizing as the techniques of learning. Computers and Artificial Intelligence, **2**: 417–441.

KODRATOFF, Y., and GANASCIA, J.-G. 1986*a*. Improving the generalization step in learning. *In* Machine learning: an artificial intelligence approach, vol. II. *Edited by* R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. Morgan Kaufmann, Los Altos, CA, pp. 215–244.

KODRATOFF, Y., and LEMERLE-LOISEL, R. 1984. Learning complex structural descriptions from examples. Computer Vision, Graphics and Image Processing, **27**: 266–290.

KODRATOFF, Y., and TECUCI, G. 1986*a*. Rule learning in DISCIPLE. Proceedings of the First European Working Session on Learning, Orsay, France.

———1986*b*. DISCIPLE: an interactive approach to learning apprentice systems. Internal Publication No. 293, Laboratoire de Recherche en Informatique, Orsay, France.

———1987. DISCIPLE-1: interactive apprentice system in weak theory fields. Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy.

KODRATOFF, Y., GANASCIA, J.-G., and VRAIN, C. 1986*b*. Apprentissage "déductif" et explications à partir de contre-exemples. Journées Françaises d'Apprentissage, Orsay, France.

MANAGO, M. 1986. Object oriented generalization: a tool for improving knowledge based systems. Proceedings of the International Meeting on Advances in Learning, Les Arcs, France, pp. 93–104.

MANAGO, M., and KODRATOFF, Y. 1987. Noise and knowledge acquisition. Proceedings of the 10th International Joint Conference on Artificial Intelligence, Milan, Italy.

MICHALSKI, R. S., 1983. A theory and methodology of inductive learning. *In* Machine learning: an artificial intelligence approach, vol. I. *Edited by* R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. Morgan Kaufman, Los Altos, CA, pp. 83–134.

MINTON, S. 1984. Constraint based generalization: learning game playing plans from single examples. Proceedings of the National Conference on Artificial Intelligence, Austin, TX.

MITCHELL, T. M., UTGOFF, P. E., and BANERJI, R. 1983. Learning by experimentation: acquiring and refining problem-solving heuristics. *In* Machine Learning: an artificial intelligence approach, vol. I. *Edited by* R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. Morgan Kaufman, Los Altos, CA, pp. 163–189.

MITCHELL, T. M., KELLER, R. M., and KEDAR-CABELLI, S. T. 1986. Explanation-based generalizations: a unifying view. Machine Learning, **1**: 47–80.

PITRAT, J. 1976. A program to learn to play chess. *In* Pattern recognition and artificial intelligence. *Edited by* Chen. Academic Press, New York, NY, pp. 399–419.

PUGET, J.-F. 1987. Goal regression with opponent. Proceedings of the Second European Working Sesssion on Learning, Yougoslavia.

QUINLAN, J. R. 1983. Learning efficient classification procedures and their application to chess end games. *In* Machine learning: an artificial intelligence approach, vol. I. *Edited by* R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. Morgan Kaufmann, Los Altos, CA, pp. 463–481.

VRAIN, C. 1985. Contre-exemple: explications déduites de l'étude des prédicats. Congrès AFCET-INRIA 85, Grenoble, France, pp. 145–159.

———1986. The use of domain properties expressed as theorems in machine learning. Proceedings of the International Meeting on Advances in Learning, Les Arcs, France pp. 78–92.

———1987. Un outil de généralisation utilisant systématiquement les théorèmes: le système OGUST. Thesis, Paris-Sud Orsay University, Orsay, France.

VRAIN, C., MANAGO, M., GANASCIA, J. G., and KODRATOFF, Y. 1986. AGAPE: an algorithm that learns from dissimilarities. Proceedings of the First European Working Session on Learning, Orsay, France; Internal Publication No. 258, Laboratoire de Recherche en Informatique, Orsay, France.

WINSTON, P. H. 1975. Learning structural descriptions from examples. *In* The psychology of computer vision. *Edited by* Patrick H. Winston. McGraw-Hill Book Company, New York, NY.