Exploiting Example Selection and Ordering to Speed-Up Learning

Filippo Neri and Lorenza Saitta

Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 TORINO (Italy)
neri@di.unito.it saitta@di.unito.it

Abstract

In this paper the results of a preliminary study on the effects of selection and ordering of examples in incremental machine learning are presented. We consider learning as a two agents communication process, in which the teacher supplies information to the learner according to a pre–defined protocol, in order to transfer to him the knowledge sufficient to solve a given problem. The aim of this work is twofold: on one hand, to find out conditions under which, for a given protocol, the knowledge transfer can be successful and, on the other, to investigate how to exploit the example selection and ordering to speed up learning. A simulated robotic task has been chosen for the experiments, which have been performed with the availability of the system WHY. The results of the experiments show that selection and ordering of examples can speed up learning, in terms of number of training examples considered.

1. Introduction

Incremental learning is receiving increasing attention in the last years [1-6]. In fact, one-step learning cannot be the only answer to automatic knowledge acquisition for more than one reason. First of all, the examples needed to train the learner may not be all available at the same time; on the contrary, they can be supplied by the environment one by one. Second, even if a set of training examples is available, their number may not be sufficient to guarantee that the acquired knowledge shall not need to be modified in the future; this is also true when knowledge-intensive learning is performed, because knowledge base stability would be assured only by a perfect domain theory. Finally, the environment may gradually change in time and one would not be obliged to learn again from scratch every time some change is detected [7].

A basic problem, in every learning system, is the dependency of the content and quality of the acquired knowledge from the training examples. However, in most cases, only the number of the learning examples has been taken into account (as for instance, in the computational learning approach [8]), not their identity. Examples may vary largely with respect to the useful information they carry to the learner and an accurate choice of them may strongly influence the resulting knowledge base [9,10]. In incremental learning there may also be a dependency of the acquired knowledge on the *order* in which the examples are presented to the learner [1,11, 12].

If an incremental learning procedure should or should not depend upon the presentation order of the examples is a matter of controversy. On one hand, order independence is desirable, because training examples can be chosen more freely, there is no need of backtracking and there is a smaller danger of examples overfitting. On the other hand, we experience, in human learning, that a suitable presentation order of selected examples can help the learner to quickly focus on the important aspects of the matter, generating thus a robust kernel of knowledge, to which border cases and exceptions can be easily added later. The example presentation order can then be exploited to speed-up learning and to obtain a more robust knowledge base. We will consider, here, only knowledge bases consisting of sets of production rules.

A theoretical model of incremental learning, the "identification in the limit" paradigm, has been proposed by Gold [13] and further elaborated by others (see, for instance, [14]). In this paper, we basically adopt that paradigm, and extend it in order to accommodate considerations on the example presentation order, which were not present in the original formulation.

The class of tasks, we are dealing with, is the class of "concept learning" tasks, in which objects from a given universe $\mathcal{Z}$ are to be recognised as instances of one or more "concepts" belonging to a known set $\Omega = \{\omega_j \mid 1 \le j \le J\}$. Concept learning can be reduced to a function identification problem [14] as follows. Given $\mathcal{Z}$ and Ω, let $\mathcal{F}$ be the set of all the total (multiple) classification functions over $\mathcal{Z}$:

$$\mathcal{F} = \{f \mid f : \mathcal{Z} \rightarrow 2^{\Omega}\}.$$

The set $\mathcal{F}$ contains the correct classification function f_0, which has to be identified. This

function is known by the teacher (or an oracle) at least in tabular form. The learning goal is that of inferring a knowledge base K such that a specified inference engine can effectively use K for computing f_0, i.e., for assigning the correct value $f_0(\zeta)$ to every element ζ of $\mathcal{Z}$; $f_0(\zeta)$ is the subset of concepts, belonging to Ω, which ζ is an instance of. Let $K(\zeta)$ denote the output of the inference engine when it uses K. K can be interpreted as a *hypothesis* [14] or a *name* [13] for f_0. Let, moreover, $ID(f_0)$ be the set of knowledge bases allowing the correct function f_0 to be computed.

A knowledge base can be specified by means of a description language. Given a logical language $\mathcal{L}$, let $\varphi(\zeta, \bar{\mathbf{y}})$ be any well formed formula belonging to $\mathcal{L}$ and let

$$\forall \zeta \, | \exists \bar{\mathbf{y}} \, | \varphi(\zeta, \bar{\mathbf{y}}) \Rightarrow \Omega' | \qquad (\Omega' \subseteq \Omega)$$

be a production rule assigning the classification Ω' to an example ζ. A knowledge base is any set of production rules and the hypothesis space contains all the possible K's.

Before introducing the learning set up, some basic definitions are needed. Given a sequence σ_n of (possibly replicated) examples belonging to $\mathcal{Z}$, the elements of σ_n can be rearranged into n! different orders. Let I(n) be the set of integers $\{1, 2, ..., n\}$ and let

$$S_n = \{s_n : I(n) \rightarrow I(n) \, | s_n \text{ is a one-to-one mapping}\}$$

be the set of all permutations of the sequence 1, 2, ... , n. Hence, the notation $\zeta_{(i)}$ means that the example $\zeta_{s(i)}$ is presented in position i-th, according to the selected permutation s_n[1]. For instance, let m = 3 and s(1) = 2, s(2) = 3 and s(3) = 1. Then, $\zeta_{(1)} = \zeta_{s(1)} = \zeta_2$, $\zeta_{(2)} = \zeta_{(2)} = \zeta_3$ and $\zeta_{(3)} = \zeta_{s(3)} = \zeta_1$. In other words, the three examples $\{\zeta_1, \zeta_2, \zeta_3\}$ are arranged into the sequence $\zeta_2, \zeta_3, \zeta_1$.

Starting from S_n, the set S_{n+1} can be defined by:

$$S_{n+1} = \{s_n(1) \ldots s_n(i-1) \; n+1 \; s_n(i) \ldots s_n(n) \, | \; (1 \le i \le n+1) \text{ and } s_n(1) \ldots s_n(i-1) \; s_n(i) \ldots s_n(n) \in S_n\} \qquad (1.1)$$

By using the constructive rule (1.1), we can define:

$$S_\infty = \lim_{n \rightarrow \infty} S_n \qquad (1.2)$$

[1] This notation is used in Order Statistics

There are, in general, two set ups for a learning problem :

- The set of training examples is given (for instances available from the environment) and the teacher has no control over the identity of the examples belonging to it. Then, the teacher's task is to arrange the examples into an "optimal" presentation order σ_n, such that the learner can reach its learning goal, by using an incremental learning algorithm $\mathcal{A}$, as quickly as possible. The available examples may or may not be representative of the probability distribution p over $\mathcal{Z}$.
- The teacher selects from $\mathcal{Z}$, one at a time, the examples to be presented to the learner. The teacher is free of selecting the examples either according to p or according to any other criterion he believes to be more suited.

Actually, there may also be a third situation, when no teacher is available or the teacher has no control at all over the selection and ordering of the examples. In this case, the problem of an "optimal" ordering does not arise and order independence may be desirable.

Identification in the limit of a classification function $f_0 \in \mathcal{F}$ can be formulated as follows. Let $\mathcal{A}$ be an incremental learning procedure and $\mathcal{KB}$ the set of knowledge bases (the *names* of the f's) described by $\mathcal{L}$. A unit of information "i" is a pair $< \zeta, f_0(\zeta) >$, with $\zeta \in \mathcal{Z}$. Let moreover Σ_∞ be the set of infinite *admissible* example presentation sequences. After each i_n, $\mathcal{A}$ outputs a knowledge base K_n. The procedure $\mathcal{A}$ *identifies in the limit* f_0 iff, for any sequence $\sigma \in \Sigma_\infty$, there exists a finite t such that $\forall n \ge t$: $K_n(\sigma) = K_t(\sigma)$ and $K_t(\sigma) \in ID(f_0)$. The set Σ_∞ may by restricted in various ways, in order to eliminate anomalous sequences, hindering identification to be reached (for instance, sequences made up of infinite repetitions of the same example). However, we assume that, if a sequence σ belongs to Σ_∞, then any permutation of σ belongs to Σ_∞ , as well.

A possible problem with this learning paradigm is the difficulty of deciding when identification of f_0 has been achieved. This problem can be solved if the teacher knows at least one knowledge base, say K^*, belonging to $ID(f_0)$. In this case, learning progresses can

be evaluated by computing a suitably defined syntactic distance between the current guess K_n and the target knowledge base $\mathcal{K}^*$.

Definition 1.1 – Let K and H be two knowledge bases containing sets of production rules. If $K \cap H$ denotes the set of rules occurring in both K and H (after suitable unifications), then:

$$\rho(K,H) = 1 - \frac{|K \cap H|}{\text{Max}\{|K|,|H|\}}$$

is a non-negative, symmetric syntactic distance, such that $\rho(K, H) = 0$ iff $K \equiv H$. ❑

By using $\rho(K,H)$, the criterion for identification in the limit can be stated as follows: if, for any $\sigma \in \Sigma_\infty$, there exists a finite t such that $\rho(K_t, \mathcal{K}^*) = 0$, then $\mathcal{A}$ has *identified in the limit* f_0 (through $\mathcal{K}^*$). In this case, we know that $\mathcal{K}^*$ is a correct hypothesis. Let moreover[2]:

$$t_{\mathcal{A}}(\sigma) = \begin{cases} \min\ n\ [\rho(K_n, \mathcal{K}^*) = 0] \\ \quad \text{if such an n exists} \\ \infty \qquad \text{otherwise} \end{cases} \tag{1.3}$$

The integer $t_{\mathcal{A}}(\sigma)$ is the "learning time" [13] or the "convergence point" [14] with respect to σ.

In general, however, $\mathcal{K}^*$ is not known. Then, we can use the weaker *behavioural correct identification* paradigm [14], in which a semantic evaluation of K_n is used. A "natural" evaluation for the task at hand is the error rate of the generated hypothesis.

Definition 1.2 – Given a knowledge base K, let P(K) be the true error rate of K on $\mathcal{Z}$. Then:

$$\tau^*(K, H) = |\, P(K) - P(H)\,|$$

is a non-negative, symmetric semantic distance. ❑

The value P(K) is usually unknown, but its value can be estimated, for instance, by using an independent test set; the error rate $\nu_M(K)$, measured on a test set of M examples, is an unbiased estimate of P(K) .

Definition 1.3 – Given two knowledge bases K and H, let $\nu_M(K)$ and $\nu_M(H)$ be the measured error rates of K and H, respectively, on a test set of M examples. Then:

$$\tau_M(K, H) = |\, \nu_M(K) - \nu_M(H)\,|$$

is a non-negative, symmetric semantic distance. ❑

Several statistical methods exist to test the null hypothesis P(K) = P(H), given the value of $\tau_M(K, H)$. Notice that $\rho(K, H) = 0$ implies both $\tau^*(K, H) = 0$ and $\tau_M(K, H) = 0$ but not vice–versa. It is easy to prove, using Bernoulli's theorem, that:

$$\lim_{M \to \infty} \Pr\{\tau_M(K, H) \to \tau^*(K, H)\} = 1$$

By using $\tau_M(K, H)$ and noticing that P(H) = 0, $\forall H \in ID(f_0)$, the following identification criterion can be used: given any $\eta \in (0,1)$, if, $\forall \sigma \in \Sigma_\infty$, there exists a finite t such that, $\forall\ n \geq t$:

$$\Pr\{P(K_n(\sigma)) = 0 \mid \nu_M(K_t(\sigma)) = 0\} \geq 1 - \eta$$

then $\mathcal{A}$ has *probably identified in the limit* f_0 (through K_t) with a confidence degree $(1-\eta)$. A definition of the convergence point $t_{\mathcal{A}}(\sigma)$, analogous to (1.3), can also be given:

$$t_{\mathcal{A}}(\sigma) = \begin{cases} \min\ n\ [\Pr\{P(K_n(\sigma)) = 0 \mid \nu_M(K_n(\sigma)) \\ \quad = 0\} \geq 1-\eta] \\ \quad \text{if such an n exists} \\ \infty \qquad \text{otherwise} \end{cases} \tag{1.4}$$

Let us consider now the problem of example ordering. Given an example presentation sequence $\sigma \in \Sigma_\infty$ and the set S_∞ of all infinite permutations defined in (1.1), the application of any $s \in S_\infty$ to σ generates, by definition, a new sequence σ' belonging to Σ_∞, too. Then, given the initial finite subsequence σ_n of σ, the application of any permutation $s_n \in S_n$ to σ_n generates an initial subsequence of a sequence belonging to Σ_∞, too. Let $\Sigma_n(\sigma)$ be the set of all permutations of the first n elements of σ: for what precedes, these are all initial subsequences of admissible sequences. Then, we can introduce a first definition of order independence.

Definition 1.4 – Given a learning procedure $\mathcal{A}$ and an admissible example sequence σ, let $K(\sigma_n)$ be the knowledge base output by $\mathcal{A}$

[2] The function min x [P(x)] returns the least value for which the predicate P is true.

after processing σ_n. $\mathcal{A}$ will be said *strongly order independent* iff:

$$\forall \sigma \in \Sigma_\infty, \forall n \geq 1, \forall \sigma_n \in \Sigma_n(\sigma):$$

$$\exists K_n(\sigma) \mid K(\sigma_n) = K_n(\sigma) \quad \square$$

In other words, given any admissible example presentation, the knowledge base inferred by $\mathcal{A}$ does not depend at any time upon the example presentation order but only upon the example identity. A strongly order independent learning procedure is, for instance, the Candidate Elimination algorithm, used by Mitchell [15] to reduce the Version Space of consistent hypotheses. The notion of strong order independence coincides with that of order independence occurring in the literature [1, 11, 12, 15].

Strong independence seems to be too a demanding property, both conceptually and computationally; furthermore, it does not allow for the learning speed-up effects, noticed in humans, obtained by exploiting the knowledge of a good teacher. Then, we will introduce, in the following, a less constraining definition.

Definition 1.5 – A learning procedure $\mathcal{A}$ will be said *weakly order independent* iff:

$\forall \sigma \in \Sigma_\infty, \forall s \in S_\infty: \exists K(\sigma) \mid K(\sigma, s) = K(\sigma)$ $\square$

In other words, $\mathcal{A}$ is weakly order independent iff the sequence of knowledge bases $K(\sigma_n)$ converges, as $n \rightarrow \infty$, to the same knowledge base $K(\sigma)$ independently of the order with which the examples are added to the sequence. However, different σ's can generate different knowledge bases.

Notice that strong order independence implies weak order independence. Furthermore, identification in the limit and order independence (both strong and weak) are incomparable. In fact, if $\mathcal{A}$ identifies f_0 in the limit, it is nevertheless possible that $K_t(\sigma) \neq K_t(\sigma')$, or $K(\sigma) \neq K(\sigma')$, for two admissible sequences such that σ' is a permutation of σ. On the other hand, if $\mathcal{A}$ is order independent, there may be no n such that $K_n(\sigma) \in ID(f_0)$.

Finally, we can link order independence (strong or weak) and identifiability in the following:

Definition 1.6 – If a learning procedure $\mathcal{A}$ identifies a classification function f_0 in the limit and, moreover, $\mathcal{A}$ is order independent, then it will be said *unbiased* with respect to f_0. Otherwise, $\mathcal{A}$ will be said *biased.* $\square$

In the considered learning problem we are mainly interested in two aspects: ability of $\mathcal{A}$ to find a knowledge base $K \in ID(f_0)$ and learning speed-up. If $\mathcal{A}$ identifies f_0, then $\mathcal{A}$ will succeed in finding a $K \in ID(f_0)$ for any choice of an admissible example presentation. However, if $\mathcal{A}$ does not identify f_0, then a $K \in ID(f_0)$ either cannot be inferred for any admissible presentation (in this case the learning problem has no solution) or it can be inferred only from a presentation sequence σ belonging to a subset Σ_∞^{succ} of Σ_∞. In the latter case, a teacher could exploit his/her knowledge in order to select a suitable $\sigma \in \Sigma_\infty^{succ}$.

Concerning the learning time, we can restrict ourselves to consider only Σ_∞^{succ}. If $\mathcal{A}$ is strongly order independent, then the set Σ_∞^{succ} can be partitioned into equivalence classes, each one containing permutations of the same sequence. Learning speed-up, i.e., decreasing of the learning time, can be forced by the teacher by selecting one among the equivalence classes, i.e., by selecting which examples are to be presented to $\mathcal{A}$. If $\mathcal{A}$ is weakly order independent, then learning speed-up can be obtained by the teacher by selecting both the identity of the examples and their presentation order.

In this paper learning is viewed as a two agents communication process, in which the teacher supplies information to the learner according to a pre-specified protocol, with the aim of letting the learner build up a knowledge base sufficient to solve a given problem. We describe a preliminary set of experiments in order to test the effectiveness of the selection and ordering of training examples, based on the suggestions supplied by a deep model of the application domain, available to the teacher and, partially, to the learner. The chosen application is an artificial robotic domain, described in Section 2; the experiments have been performed using the system WHY [16-18], an overview of which is given in Section 3. In Section 4 the protocol of interaction between the teacher and the learner is described, in Section 5 the experimental results are discussed and some conclusions are drawn in Section 6.

2. Description of the Application

The application chosen to perform the experiments consists in a simulated robotic task: a robot has to move objects from their current location without damaging the objects themselves or the environment. The robot is presented with scenes, each containing the object to be moved (the "focus" of attention) and other objects in spatial or structural relations with the main one.

The robot has four different mechanisms to move an object. The applicability of a specific transportation mechanism to an object characterises the class to which the object belongs. In the following, the considered classes are briefly described.

Class NOT-MOVE (= ω_1)

An object belongs to the class NOT-MOVE (which is exclusive with respect to the other classes) if its displacement from its current location is either impossible for the robot or unsuitable. For instance, if the object is screwed to a wall or lies inside a closed drawer, then it cannot be moved.

However, it is not always so clear that an object cannot be moved. In some cases, a displacement mechanism could in principle be applied, but it fails to produce the desired effect because of contingent reasons: for instance, the object is too heavy for the robot strength.

If all the applicable mechanisms fail, then the object is classified in class ω_1. This kind of classification "by default" has to be treated carefully by the learner.

Class PUSH (= ω_2)

The robot can push an object x by applying to it a horizontal force $\bar{F}_p$.

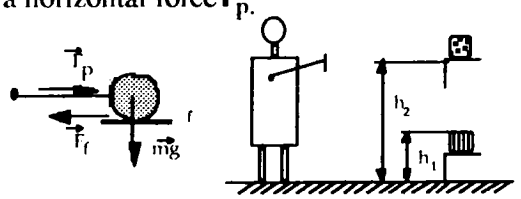

Fig 2.1 – Scheme of the PUSH mechanism. The values of the constants are: Fp = 100 Kg, h_1 = 30 cm, h_2 = 160 cm. The object can be displaced only if F_p > fmg, where m is the mass of x and f is the friction coefficient between x and its support surface.

The objects must lye inside a height range $[h_1,h_2]$, as represented in Fig. 2.1. Moreover, we have prohibit an object with handles to be a member of this class, in order to avoid too much overlapping with other classes. Finally, as the robot can only push an object away from itself and not draw it towards itself, the object must not be located in the corner made by two vertical touching obstacles (for instance, in the corner between two walls).

Class HAND-TAKE (= ω_3)

The robot can grasp a not fragile object x with one hand by applying to x a horizontal force $\bar{F}_h$; the resulting vertical force $\bar{F}$ on the grasped object depends on F_h and on the friction coefficient f between x's surface and the robot's metal hand. It is required that $2F_hf > mg$, being m the mass of the object x.

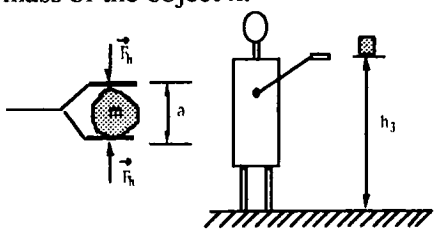

Fig 2.2 – Scheme of the HAND-TAKE mechanism. The values of the constants are: F_h= 1 Kg, a = 15 cm, h_3 = 120 cm. Only objects with a maximum width "a" and with a height above the floor less than h_3 can be grasped.

This mechanism is applicable only to objects whose height above the floor is less than a value h_3. Finally, objects must not be fragile, because the robot's grip would break them.

Class ARMS-TAKE (= ω_4)

The robot can grasp a not fragile object x with its arms by applying to x a horizontal force $\bar{F}_a$; the resulting vertical force $\bar{F}$ on the grasped object depends on F_a and on the friction coefficient f between x's surface and the robot's metal arms. It is then required that $2F_af > mg$, being m the mass of the object x.

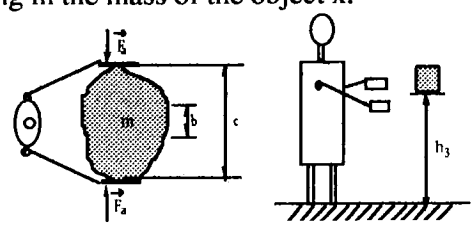

Fig 2.3 – Scheme of the ARMS-TAKE mechanism. The values of the constants are: F_a= 1 Kg, b = 30 cm, c = 70 cm, h_3 = 120 cm. Only objects whose width is in the range [b,c] and whose height above the floor is less than h_3 can be grasped.

Class RAISE (= ω_5)

The robot has also one or two hooks to lift objects. The global vertical force $\bar{F}_k$ can be

concentrated into one hook or distributed over the two. As before, only objects whose height above the floor is less than a value h_4 can be lifted. Finally, liftable objects must have at least one holed handle. In case the object has exactly one holed handle, it has to be empty or closed, in order not to spill the possible content during transportation.

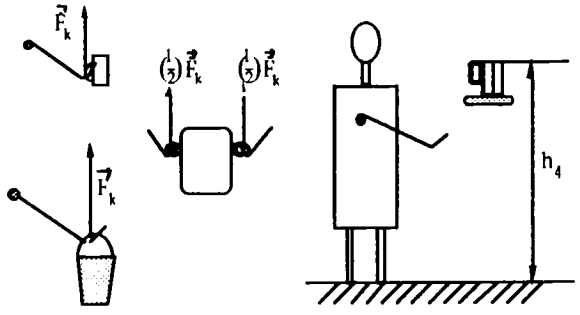

Fig 2.4 – Scheme of the RAISE mechanism. The values of the constants are F_k = 30 kg, h_4 = 100 cm.

Notice that the same example can be instance of more than one class (excluding NOT-MOVE).

3. WHY's Overview

In order to make this paper as much as possible self-consistent, a brief overview of the system WHY, used to perform the experiments, will be given in this section. Extensive descriptions of this system can be found elsewhere [16-18].

WHY is a system that learns and refines a knowledge base using a *causal model* of the domain, $\mathbf{C}$, a set of examples and a *phenomenological* theory, $\mathbf{P}$, describing concrete manifestations of abstract concepts occurring in $\mathbf{C}$. The system can also be used in a semi-automated way, allowing a direct interaction with an expert. This aspect of the system proved to be particularly useful in the work presented here.

3.1. Knowledge Representation

The representation of all kinds of knowledge used by WHY is based on a first order logic language $\mathbf{L}$. As concerns the notation, operational predicates and non-operational predicates are written in lower-case and upper-case letters, respectively. As usual, variables occurring in the head of an implication are universally quantified, whereas variables occurring only in the body are existentially quantified.

The causal model $\mathbf{C}$ is represented, at the logical level, as a network. In Fig. 3.1 the part of $\mathbf{C}$ referring to the class HAND-TAKE is reported. Nodes in the network are either *primary* (ellipses) or *accessory* (rectangles and clouds). Primary nodes correspond to processes or system states and a subset of these nodes contains the *first causes* (shaded nodes). Effects of the same cause are AND-ed, causes of the same effect are OR-ed. Accessory nodes represent either *constraints* on causal links (rectangles) or *contexts* (clouds), i.e., conditions to be satisfied by the environment. Primary nodes of the network may or may not be observables, i.e., they may or may not have manifestations associated with them. On the contrary, constraints and contexts must be observable.

The phenomenological theory $\mathbf{P}$ contains structural information, definitions of ontologies, general knowledge and a set of rules describing manifestations associated to abstract concepts, which can, thus, be "operationalized". The theory $\mathbf{P}$ contains also the links between the causal network $\mathbf{C}$ and the classes. In fact, rules of the following types occur in $\mathbf{P}$:

$$\underline{C}(x) \wedge \alpha(x, \bar{\mathbf{y}}) \Rightarrow \omega_j(x) \qquad (3.1)$$

where $\underline{C}$ is the name of a primary causal node[3], α is a (possibly empty) conjunction of additional predicates and ω_j is the name of a class.

The target knowledge base $\mathbf{K}$ consists of a set of decision rules of the following type:

$$r \equiv \varphi(x, \bar{\mathbf{y}}) \Rightarrow \omega_j(x) \qquad (3.2)$$

In (3.2) $\bar{\mathbf{y}}$ denotes a set of variables, x an object to be classified, ω_j a class and φ is a formula of the language $\mathbf{L}$, containing both operational and non-operational predicates.

3.2. Basic Reasoning Mechanisms

Four basic reasoning mechanisms are integrated into the system WHY: induction, deduction, abduction and prediction. *Induction* is used when theory incompleteness occurs and is performed by invoking the inductive module of the ML-SMART system [19]. *Deduction* is also performed using ML-SMART, as described in [20].

Deduction is used to build up a data structure called the *justification forest*, containing operationalizations of abstract predicates; this forest is the basis for both the knowledge justification process and the classification procedure. The causal model can be searched either from effects to causes (search for an explanation via *abduction*) or from causes to effects (*prediction* of conse-

[3] Underlined predicates denote primary causal nodes.

quences via deduction), in order to assert the truth status of each node in the network.

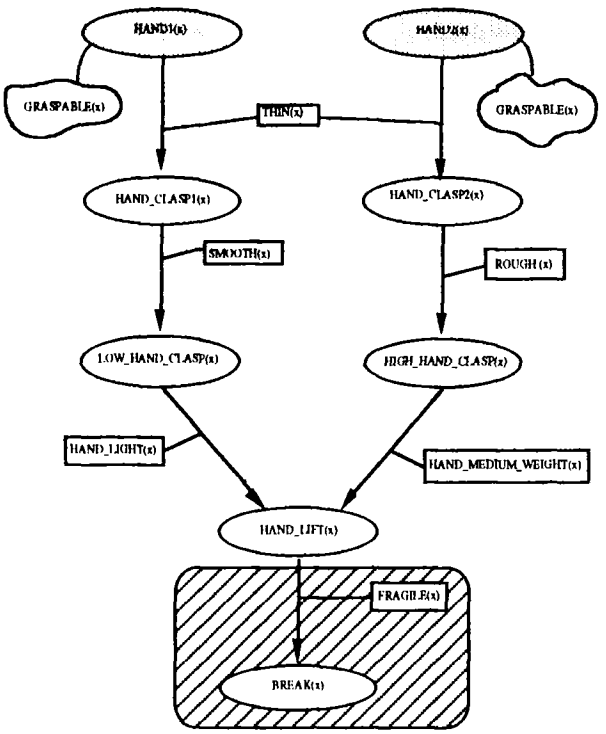

Fig. 3.1 – Part of the causal model referring to the class HAND-TAKE. The predicates GRASPABLE(x), ROUGH(x), THIN(x), HAND-MEDIUM-WEIGHT(x) state conditions on the location of the object x, on the roughness of the constituent material and on the values of its width and weight, respectively. The following rule, occurring in $\mathcal{P}$, establishes a link between a node of the network and the corresponding class:
HAND-LIFT(x) ⇒ HAND-TAKE(x)

3.3. Learning Mechanisms

The system WHY can learn both in one step from a set of examples, or incrementally, by analysing one example at a time and updating correspondingly the target knowledge, if necessary. It can also perform a semi-automated theory revision, by interacting with a human expert [18].

3.3.1. One-step Learning

The one-step learning consists of two phases. First, the system tries to explain why the training examples are correct instances of their class(es), by building up the *justification forest* $\mathcal{G}$. This forest is a generalization of the EBL explanation tree to the case in which many predicates are operationalized on many examples at the same time. In each node of $\mathcal{G}$ the corresponding set of examples, satisfying the predicate associated with that node, is stored. Obviously, the forest $\mathcal{G}$ contains only those parts of the theory $\mathcal{P}$ which have been stimulated by the given training examples. When new examples will be considered later, the forest will possibly expand. In Fig. 3.2 part of the justification forest, obtained for an instance of the class HAND-TAKE, is reported.

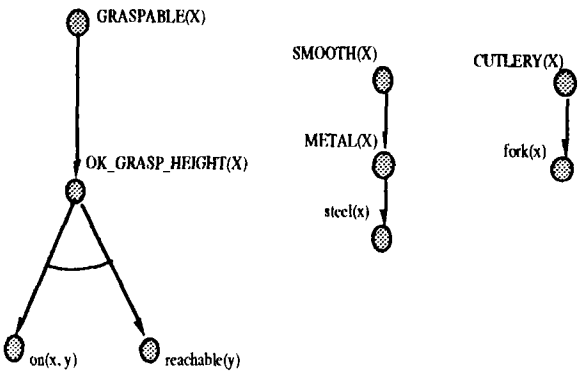

Fig. 3.2 – Part of the justification forest obtained for an instance of the class HAND-TAKE.

When the construction of the justification forest halts, some nodes of the causal network have been activated (at least the ones occurring in the rules (3.1)). Starting from the activated nodes, the system tries to follow paths in the causal network, in search for first causes. For instance, for the example referred to in Fig. 3.2, the system tries to reach one of the first causes HAND-1(x) or HAND-2(x), starting from HAND-LIFT(x). This process envisages the verification of all the constraints and contexts encountered along the path[4]. If the search for first causes is successful, the conjunction $\psi(x,\bar{y})$ of all the predicates verified along the path will imply the entry point node, under the assumption that the first cause holds true. For example, from the right–most path in Fig. 3.1 and from the rule

HAND-LIFT (x) ⇒ HAND-TAKE (x)

we obtain:

GRASPABLE(x) ∧ THIN(x) ∧ ROUGH(x) ∧ HAND-MEDIUM-WEIGHT(x) ⇒ HAND-TAKE(x) (3.3)

Rule (3.3) is added to the target knowledge base $\mathcal{K}$. Two important aspects are to be pointed out about this rule:

- The rule contains non-operational predicates. The examples, from whose justification the rule has been extracted, possibly verify alternative operationaliza-

[4] Actually, the process is more complex, because of the possibility of making assumptions about the truth of some predicates during this process.

alizations of these predicates. The operationalizations instantiated in the observed examples are stored in the justification forest. As a consequence, the $\mathcal{K}$ is very compact, because a single rule in the $\mathcal{K}$ may summarise many operational rules, each one corresponding to a different combination of operationalizations of the non-operational predicates.

- The rule is correct, as long as the causal model is correct. In fact, the rule has been causally justified by the model[5].

If the causal model is complete, then the acquired knowledge base is "perfect", in the sense that it contains all and only the abstract rule allowing any possible example ζ to be correctly classified. This knowledge base will be denoted by $\mathcal{K}^*$.

3.3.2. Incremental Learning

The system WHY can refine a knowledge base acquired either as described above or in any other way, for instance, given by an expert. Incremental refinement of a knowledge base consists of three steps:

(1) A new example is supplied to the system for classification. The classification generated by WHY is compared with the one given by the teacher. If no error occurred, then the example is only added to the extensions of the verified predicates in the justification forest and in the causal net.

(2) If an error occurs, then an accurate analysis of the reasons underlying the error is done.

(3) According to the results of the previous analysis, the system has a number of options, including generalizing or specialising the knowledge base, running the inductive module, deferring learning or asking the expert for more information.

3.4. Classification Procedure

The possibility of multiple classifications and the presence of a "default" class are not trivial issues for the classifier. Some of the adopted solutions are not relevant for the present work; hence, we will give a simplified version of the classification procedure:

Let ζ be the example to be classified.

(1) As the class NOT-MOVE is incompatible with respect to any other class, and there are rules explicitly concluding that class, these rules are evaluated first. If at least one of them is verified by ζ, then ζ is labelled as a member of the class NOT-MOVE and the procedure halts.

(2) If ζ does not verify any of the above rules, the rules related to the other classes are tried. For each rule verified by ζ, the label corresponding to the implied class is assigned to it, possibly obtaining a multiple classification.

(3) If ζ does not verify any of the rules in (1) and (2), WHY backs up to the causal reasoning. If a new justification, leading to a new rule or to a new operationalization of an existing rule is found for a class different from NOT-MOVE, then the sample is classified as a member of that class. Otherwise, WHY has proved that there are justified reasons for ζ not to be a member of any of the classes different from NOT-MOVE and, then, ζ is assigned to the class NOT-MOVE.

4. Selection and Ordering of the Training Examples

The learning protocol, used in the experiments reported later, involves an artificial learning system $\mathcal{A}$ and a human teacher $\mathcal{T}$. The learner knows the phenomenological theory $\mathcal{P}$, an incomplete version, $\mathcal{C}$, of the causal theory and the rules, such as $\underline{C}(x) \Rightarrow \omega_j(x)$, linking the causal node $\underline{C}(x)$ to the class ω_j; if these last rules have additional predicates, such as in $\underline{C}(x) \wedge \alpha(x, \bar{y}) \Rightarrow \omega_j(x)$, the learner does not know $\alpha(x, \bar{y})$. The teacher knows a perfect causal model $\mathcal{C}^*$, the phenomenological theory $\mathcal{P}$ and the complete rules $\underline{C}(x) \wedge \alpha(x, \bar{y}) \Rightarrow \omega_j(x)$ for each class ω_j. In Fig. 3.1, the two nodes included in the shaded rectangle are hidden from the learner.

Moreover, the teacher knows what the learner's knowledge is and also its learning and classification procedure. It would have been possible to allow the teacher to have the same partial causal model as the learner has and to modify it, guided by the learner answers (as WHY actually does). However, we decided to use, for this preliminary study, the simpler setting in which the teacher does not need himself to learn. Finally, the learner knows that the teacher is co-operative.

Finally, the teacher knows the classification function $f(\zeta)$ and the perfect abstract knowledge base $\mathcal{K}^*$.

[5] The description of the justification process is not relevant for the present paper.

The exploitation, by the part of the learner, of an only slightly incomplete version of the causal model and of the complete phenomenological theory eases the learning task. This choice has been done in order to give more evidence to the order effects, which are thus less masked by the difficulties of learning per se. However, even sharing with the teacher a large part of the background knowledge, the learner may find hard to acquire the target knowledge base.

The communication protocol between the teacher and the learner is as follows: the teacher iteratively presents to the learner a classified example $\zeta_{(n)}$ and the learner updates its knowledge base, showing the results to $\mathcal{T}$, who evaluates it and chooses accordingly the next example. The examples cannot be generated in a totally arbitrary way, because they must be reasonable objects of the world. In order to meet this requirement, the teacher has a set of scene (example) prototypes, specifying what kind of scenes can be built up. These prototypes are unknown to the learner.

The teacher's knowledge allows him to build up the target knowledge base he wants the learner to acquire by using the method sketched in section 3.3.1. As mentioned there, his knowledge base, $\mathcal{K}^*$, can be expressed, at an abstract level, using non operational predicates and, at a more concrete one, using only operational predicates. Let us denote by $\mathcal{K}^*_{op}$ the operational knowledge base. Let, moreover, $|\mathcal{K}^*| = R$ and $|\mathcal{K}^*_{op}| = R_{op}$. Each abstract rule in $\mathcal{K}^*$ corresponds to several operational ones. Let us consider, for instance, the rule:

$$r \equiv \varphi(x, \bar{y}) \Rightarrow \omega_j(x) \qquad j \in |1, J|.$$

$OP(\varphi)$ or, equivalently, OP(r) will denote the set of operational formulas (rules) corresponding to $\varphi(x, \bar{y})$ (or to r), whereas

$$t(\varphi) \underset{Def}{=} t(r) = |OP(r)|$$

will denote the number of operational formulas (rules) corresponding to $\varphi(x, \bar{y})$ (or to r). As a special case, t(Q) denotes the number of alternative operationalizations corresponding to a non–operational predicate Q. Then, a rule

$$r \equiv Q_1(x, \bar{y}) \wedge \ldots \wedge Q_m(x, \bar{y}) \wedge \beta(x, \bar{y}) \Rightarrow \omega_j(x),$$

where β is a conjunction of operational predicates, corresponds to a number t(r) of operational rules, such that:

$$t(r) \leq \prod_{i=1}^{m} t(Q_i) \qquad (4.1)$$

The value given in (4.1) is only an upper bound for t(r), because it may happen that some of the operationalizations of two predicates Q_i and Q_j ($i \neq j$) are mutually incompatible, reducing thus the number of possible combinations.

Let us now consider the target operational knowledge base:

$$\mathcal{K}^*_{op} = \bigcup_{r \in \mathcal{K}^*} OP(r) \qquad (4.2)$$

then:

$$R_{op} = \sum_{r \in \mathcal{K}^*} t(r) \qquad (4.3)$$

In our learning set up, the teacher wants the learner to acquire, first, $\mathcal{K}^*$. When this goal is achieved, convincing the teacher that the learner has understood the fundamental aspects of the domain, $\mathcal{T}$ can concentrate himself in the task of showing to the learner as many alternative operationalizations as possible of the non-operational predicates occurring in $\mathcal{K}^*$, using a number of examples as small as possible. In other words, the learner has now to transform $\mathcal{K}^*$ into $\mathcal{K}^*_{op}$ by associating to each $r \in \mathcal{K}^*$ the corresponding set OP(r). Actually, the two goal of teaching $\mathcal{K}^*$ and $\mathcal{K}^*_{op}$ need not to be fulfilled in a strict temporal sequence: the teacher starts to teach $\mathcal{K}^*_{op}$ since the beginning, but the choice of the examples is primarily influence by the goal of reaching $\mathcal{K}^*$ first.

Notice that the separation between $\mathcal{K}^*$ and $\mathcal{K}^*_{op}$ partially solves the problem of handling changes in the environment: in fact, it is rare that the basic cause-effect relations holding in a domain undergo changes, whereas operationalizations are easily prone to updates and novelties. On the other hand, the knowledge base $\mathcal{K}^*$ would be unuseful without $\mathcal{K}^*_{op}$, because the predicates occurring in the rules of $\mathcal{K}^*$ could not be evaluated in the world. This is the main reason why the learner needs examples and cannot just compile $\mathcal{K}^*$ directly from $\mathcal{C}$, even in the case it could have access to the perfect $\mathcal{C}^*$. For what concerns $\mathcal{P}$, the role of the examples is that of selecting those combinations of operationalizations, which actually occur in some example, among the large number of possible ones allowed by $\mathcal{P}$.

As long as the learner does not know completely $\mathcal{K}^*_{op}$, its knowledge K, even if al-

ready equal to $\mathcal{K}^*$, would still produce errors. Suppose, for instance, that:

$$r_1 \equiv Q(x, \bar{y}) \wedge \alpha(x, \bar{y}) \Rightarrow \omega_j(x) \quad (4.4)$$

is a rule belonging to $\mathcal{K}^*$ and that

$$\beta(x, \bar{y}) \Rightarrow Q(x, \bar{y}) \quad (4.5)$$

$$\gamma(x, \bar{y}) \Rightarrow Q(x, \bar{y}) \quad (4.6)$$

are two operationalizations of Q. Suppose, furthermore, that the learner currently knows only rule (4.5). Then, rule (4.4) will turn out to be not verified on an example ζ such that $\gamma(\zeta, \bar{b})$ is true, but $\beta(\zeta, \bar{b})$ is false, generating thus an omission error. The omission error could produce, in turn, a commission error, if ζ is assigned by default to class ω_1.

According to the above considerations, the teacher $\mathcal{T}$ has to evaluate the current knowledge base of the learner, both for assessing its performance and to guide his own selection of the next example. Let $\zeta_{(1)}, \zeta_{(2)}, \ldots, \zeta_{(n)}$ be the sequence of examples already presented to the learner. Let K_n be the abstract knowledge base acquired after incorporating the information carried by $\zeta_{(n)}$, and let $OP(K_n)$ be the corresponding operational one.

Using the distance measure introduced in Definition 1.1, $\mathcal{T}$ evaluates $\rho(\mathcal{K}^*, K_n) \equiv \rho_n$; his goal is to let ρ_n tend to zero as quickly as possible. The set $H_n = K_n \cap \mathcal{K}^*$ contains the rules that have reached their final abstract formulation.

As concerns the comparison of $\mathcal{K}^*_{op}$ and $OP(K_n)$, the distance ρ is not suited, firstly because of computational reasons, due to the potential complexity of $\mathcal{P}$, and, secondly, because $\mathcal{K}^*_{op}$ is usually not known. For the above reasons, the teacher uses the distance $\tau(\mathcal{K}^*_{op}, OP(K_n))$, introduced in the Definition 1.3, to evaluate the current status of $OP(K_n)$. Actually, the error rate $\nu_M(OP(K_n))$ will be used directly as a measure of the performance level of $OP(K_n)$; in fact, in a static environment, $\nu_M(\mathcal{K}^*_{op})$ is a constant (ideally, equal to zero).

In order to better explain how learning proceeds and how the example selection works, we will describe in more details how the rules are generated from examples. Given a path ξ^* in the causal net $\mathcal{C}^*$, let $\psi^*(x, \bar{y}) \Rightarrow \omega_j(x)$ be the decision rule associated to it (see (3.3) as an example). Given a conjunctive formula $\varphi \in \mathcal{L}$, let $\{\varphi\}$ denote the set of predicates occurring in it. Let ζ be an example presented to $\mathcal{A}$. Even though ζ may be instance of more than one class, we consider one class ω at a time. For ζ the ground predicate $\omega(\zeta)$ is true. Let ξ^* be the complete causal path followed by ζ in $\mathcal{C}^*$ and corresponding to ω (each example satisfies only one causal path for each class). When ζ is presented to $\mathcal{A}$, several events may happen:

(A) The causal path ξ^* occurs also in the uncomplete net $\mathcal{C}$, known by $\mathcal{A}$. Then, $\mathcal{A}$ learns, from the causal justification of ζ, the rule:

$$r \equiv \psi^*(x, \bar{y}) \Rightarrow \omega(x)$$

which belongs to $\mathcal{K}^*$ and does not need to be further modified. $\mathcal{A}$ does not know that $r \in \mathcal{K}^*$, but $\mathcal{T}$ will tell it, after examination of the generated knowledge base.

(B) The causal path ξ^* does not completely occur in $\mathcal{C}$, because some constraints or contexts are missing. Then $\mathcal{A}$ acquires a rule:

$$r \equiv \psi(x, \bar{y}) \Rightarrow \omega(x)$$

where ψ corresponds to the partial path ξ, occurring in $\mathcal{C}$, which, for $\mathcal{A}$, is the causal justification. Then, $\{\psi\} \subset \{\psi^*\}$, and r has to be specialised further in order to reach its final form. More precisely, the predicates in $\{\varphi\} = \{\psi^*\} - \{\psi\}$ are to be added to r's left-hand side. Notice that all the predicates in $\{\varphi\}$ are true of ζ.

(C) Because of $\mathcal{C}$'s incompleteness, it may happen that ζ satisfies also another causal path $\xi' \neq \xi$, corresponding to the same class ω ; then the rule:

$$r \equiv \psi'(x, \bar{y}) \Rightarrow \omega(x)$$

is generated. As the complete path ξ'^* is not verified by ζ, if and when rule r will be completed, it will not cover ζ anymore, because the first cause explaining why ζ is of class ω belongs to the path ξ^* and not to ξ'^*. However, the learner is not aware of its current mistake, because r covers ζ and gives the current classification ω.

Rule r has to be completed by adding to its left-hand side the predicates in $\{\varphi\} = \{\psi'^*\} - \{\psi'\}$, some of which are certainly false of ζ.

(D) $\mathcal{C}$'s incompleteness can produce still another situation. The example ζ can satisfy an incomplete causal path ξ' corresponding to a class $\omega' \neq \omega$. in this case the rule:

$$r \equiv \psi'(x, \bar{y}) \Rightarrow \omega'(x)$$

is generated, but the learner is aware of the mistake, because it knows that $\omega'(\zeta)$ is false. Rule r has to be completed by adding the predicates in $\{\varphi\} = \{\psi'^*\} - \{\psi'\}$, some of which are necessarily false of ζ.

The most difficult rules to complete are the rules of type (B). In order to eventually acquire $\mathcal{K}^*$, the learner uses only specialisation, i.e., it adds predicates to the condition part of a rule or descends the justification forest. The learning procedure has been so simplified, in order to allow an easier analysis of its behaviour.

If a rule $r \equiv \psi(x, \bar{y}) \Rightarrow \omega(x)$ is generated by an example $\zeta_{(k)}$, no example $\zeta_{(j)}$, with $1 \leq j \leq k-1$, verifies it (otherwise it would have been generated before). A set $SPEC_k(r)$ is then associated to a r; this set contains all the roots of the justification forest $\mathcal{G}$ true of $\zeta_{(k)}$, less the predicates occurring in $\{\psi\}$. If $r \equiv \psi^*(x, \bar{y}) \Rightarrow \omega(x)$ is the corresponding complete rule in $\mathcal{K}^*$, then all the predicates occurring in $\{\varphi\} = \{\psi^*\} - \{\psi\}$ belong to $SPEC_k(r)$ for rules of type (B), whereas some predicates must be missing for rules of type (C) and (D). For rules of type (A), $\{\psi^*\} = \{\psi\}$. The set $SPEC_k(r)$ contains all the predicates that can be used to specialise rule r without excluding $\zeta_{(k)}$ from its cover. Obviously, $SPEC_k(r)$ may contain many irrelevant predicates, because the ones that will make r equal to r* are only those in $\{\varphi\}$. The set $SPEC_k(r)$ is not created when the r is of type (D).

The interaction between $\mathcal{T}$ and $\mathcal{A}$, while $\mathcal{T}$ is teaching $\mathcal{K}^*$, is described by the algorithm TEACH–ABS, reported in Fig. 4.1, whereas their interaction, while $\mathcal{T}$ is teaching $\mathcal{K}^*_{OP}$, is described by the algorithm TEACH–OP, reported in Fig. 4.4.

The basic loop in algorithm TEACH-ABS is repeated until $\mathcal{A}$ reaches the final abstract knowledge base $\mathcal{K}^*$ or until $\mathcal{T}$ discovers that $\mathcal{A}$ cannot possibly acquire $\mathcal{K}^*$. In fact, it may happen that some of the predicates to be added to a rule have been lost, using the simple learning strategy adopted in this paper. This fact appears clearly from Fig. 4.2 and 4.3, in which a more detailed description of Steps (1) and (2) of the algorithm TEACH-ABS is reported. In Step (1) $\mathcal{T}$ executes the algorithm SELECT, whereas in Step (2) $\mathcal{A}$ executes the algorithm LEARN.

TEACH–ABS

Let K_n be the knowledge base learned by $\mathcal{A}$, after processing $\zeta_{(n)}$ and let $H_n = K_n \cap \mathcal{K}^*$

Let n = 0. By definition $K_0 = \emptyset$; $H_0 = \emptyset$;

while $\rho(\mathcal{K}^*, K_n) \neq 0$ do

(1) $\mathcal{T}$ generates a new example $\zeta_{(n+1)}$ and sends it to $\mathcal{A}$

(2) $\mathcal{A}$ uses $\zeta_{(n+1)}$ to build up the updated versions K_{n+1} and $\{SPEC_{n+1}(r), \forall r \in K_{n+1}\}$ and sends them to $\mathcal{T}$

(3) $\mathcal{T}$ evaluates $\rho(\mathcal{K}^*, K_{n+1})$ and send H_{n+1} back to $\mathcal{A}$

(4) n = n+1

end

Fig. 4.1 – Algorithm TEACH-ABS, describing the interaction between the teacher $\mathcal{T}$ and the learner $\mathcal{A}$ while $\mathcal{T}$ teaches $\mathcal{K}^*$ to $\mathcal{A}$.

The basic loop in algorithm TEACH-ABS is repeated until $\mathcal{A}$ reaches the final abstract knowledge base $\mathcal{K}^*$ or until $\mathcal{T}$ discovers that $\mathcal{A}$ cannot possibly acquire $\mathcal{K}^*$. In fact, it may happen that some of the predicates to be added to a rule have been lost, using the simple learning strategy adopted in this paper. This

fact appears clearly from Fig. 4.2 and 4.3, in which a more detailed description of Steps (1) and (2) of the algorithm TEACH-ABS is reported. In Step (1) $\mathcal{T}$ executes the algorithm SELECT, whereas in Step (2) $\mathcal{A}$ executes the algorithm LEARN.

In SELECT, the teacher $\mathcal{T}$ evaluates the progress made by $\mathcal{A}$ in acquiring $\mathcal{K}^*$. First of all, it looks for completed rules, which are recognised from the fact that their specialisation set $SPEC_n(r)$ contains only the predicates in $\{\psi^*\}-\{\psi\}$, i.e., those that are missing from the complete causal path. Then, it searches for rules to be refined. If in the current knowledge base K_n there are no more rules to be refined ($K_n = H_n$) but still $K_n \neq \mathcal{K}^*$, then a new rule has to be added to K_n. To this aim, $\mathcal{T}$ sends to $\mathcal{A}$ a positive example of a causal path not yet activated, on the basis of which $\mathcal{A}$ will generate a new rule. If, on the contrary, there are in K_n still rules to be refined, the teacher selects one at random (notice that the learner does not know what rule has been selected) and selects a negative example of the class implied by the rule according to the criterion described in the following. Let $r \equiv \psi(x,\bar{y}) \Rightarrow \omega(x)$ be the selected rule. Then, its $SPEC_n(r)$ set shall contain all the predicates in $\{\psi^*\}-\{\psi\}$ plus a number of irrelevant ones. The negative example of ω should be such that all the predicates in $\{\psi^*\}-\{\psi\}$ be false of it and at least one of the irrelevant predicates be true.

SELECT

$\mathcal{T}$ has received K_n and $\{SPEC_n(r), \forall r \in K_n\}$

for all $r \equiv [\psi(x,\bar{y}) \Rightarrow \omega(x)] \in K_n - H_n$ do

  if $SPEC_n(r) = \{\psi^*\} - \{\psi\}$

    then $H_n = H_n \cup \{r\}$

  endif

end

$\mathcal{T}$ evaluates $\sigma_n = 1 - |H_n|/R$

  if $K_n \neq H_n$

    then

      $\mathcal{T}$ selects $r \equiv [\psi(x,\bar{y}) \Rightarrow \omega(x)] \in K_n - H_n$ to be specialised, but does not tell $\mathcal{A}$ which rule has been selected.

      if $SPEC_n(r) \cap \{\psi^*\} - \{\psi\} = \emptyset$

        then 'ERROR'

        else $\mathcal{T}$ generates a negative example $\zeta_{(n+1)}$ of ω, such that every predicates in $SPEC_n(r) \cap \{\psi^*\} - \{\psi\}$ is false of $\zeta_{(n+1)}$ and at least one of the remaining predicates in $SPEC_n(r)$ is true of it.

      endif

    else $\mathcal{T}$ generates a positive example $\zeta_{(n+1)}$ of a causal path ξ^* (relative to a class ω), not yet generated.

  endif

$\mathcal{T}$ sends to $\mathcal{A}$: H_n and $< \zeta_{(n+1)}, \omega_1, ..., \omega_q >$

($\zeta_{(n+1)}$ may have a multiple classification)

Fig. 4.2 – Algorithm SELECT, describing the process performed by $\mathcal{T}$ for selecting a rule to be refined and a new example to be sent to $\mathcal{A}$.

This last will be cancelled from $SPEC_n(r)$, as described in the algorithm LEARN. Then, the greater the number of irrelevant predicates satisfied by the example, the better, because the set $SPEC_n(r)$ will take less examples to be reduced. It may happen that no negative example can be constructed in such a way to be false for all the predicates in $\{\psi^*\}-\{\psi\}$. In this case, the predicates in $\{\psi^*\}-\{\psi\}$, satisfied by the example, will be cancelled from $SPEC_n(r)$ and will be lost. When the teacher notice that a rule r has still to be specialised, but no relevant predicate is

anymore present in $SPEC_n(r)$, he signals an error condition. Currently, upon occurrence of this condition, the teacher moves the rule to H_n, marking it as incomplete. This is done in order to let the cycle of algorithm TEACH-ABS halt in any case. A possible strategy, that will be investigated in the future, is the possibility of generalising the rules, so that lost predicates could be considered again.

Notice that the example generated by the teacher can be an instance of more than one class $\omega_1, \ldots, \omega_q$, so that it will be used by $\mathcal{A}$ to update more rules than just the one the teacher had in mind.

LEARN

$\mathcal{A}$ has received H_n and $< \zeta_{(n+1)}, \omega_1, \ldots, \omega_q >$

$K_{n+1} = K_n$

for all $r \equiv [\psi(x, \bar{y}) \Rightarrow \omega(x)] \in K_n - H_n$ such that $\zeta_{(n+1)}$ verifies r but $\omega(\zeta_{(n+1)})$ is false do

  for all $Q \in SPEC_n(r) \mid Q(\zeta_{(n+1)})$ do

    $SPEC_{n+1}(r) = SPEC_n(r) - \{Q\}$

    $SPEC_{n+1}(r) = SPEC_n(r) \cup SON(Q)$

    where SON(Q) is the set of predicates which are son of Q in the current justification forest $\mathcal{G}$ and are false of $\zeta_{(n+1)}$.

  end

end

if $\zeta_{(n+1)}$ verifies a new causal path $\xi \in \Sigma$, relative to a class ω occurring in $\{\omega_1, \ldots, \omega_q\}$ do

  $K_{n+1} = K_{n+1} \cup \{r' \equiv \psi'(x, \bar{y}) \Rightarrow \omega_i(x) \mid i \in \{1, \ldots, q\}\}$

  Generates $SPEC_{n+1}(r') = \{$Roots in $\mathcal{G}$ true of $\zeta_{(n+1)}\} - \{\psi'\}$

endif

Fig. 4.3 – Algorithm LEARN, describing the process performed by $\mathcal{A}$ when it receives a new example $\zeta_{(n+1)}$.

Notice that the set $SPEC_k(r)$, corresponding to a rule r, is generated when $\mathcal{A}$ is presented with the first positive example of the class implied in r, and that it is reduced any time r is satisfied by a negative example of the same class.

In the algorithm LEARN, $\mathcal{A}$ refines the sets $SPEC_n(r)$ corresponding to those rules for which $\zeta_{(n+1)}$ is a negative example. Moreover, it possibly adds some new rules. The updating of $SPEC_n(r)$ consists in cancelling those predicates which are true of $\zeta_{(n+1)}$. If a cancelled predicate Q is an internal node in $\mathcal{G}$, then the son of Q in $\mathcal{G}$ are substituted to Q in $SPEC_n(r)$ and the process is recursively repeated for each one of them.

When the acquisition of $\mathcal{K}^*$ ends (either with success or failure) the teacher starts to complete, as much as possible, $\mathcal{K}^*_{op}$, according to the algorithm TEACH-OP, reported in Fig. 4.4.

TEACH-OP

Let n be the number of examples already seen, K_n the knowledge base learned by $\mathcal{A}$ after processing $\zeta_{(n)}$ and $\nu_M(OP(K_n))$ the error rate of $OP(K_n)$ on the test set TEST of cardinality M.

$\mathcal{A}$ evaluates $\nu_M(OP(K_n))$ and sends the obtained value to $\mathcal{T}$.

repeat

(1) $\mathcal{T}$ generates a new example $\zeta_{(n+1)}$ and sends it to $\mathcal{A}$.

(2) $\mathcal{A}$ uses $\zeta_{(n+1)}$ to build up the updated version $OP(K_{n+1})$ and evaluates $\nu_M(OP(K_{n+1}))$

(3) $n = n+1$

until $\nu_M(OP(K_n)) \leq \varepsilon$

Fig. 4.4 – Algorithm TEACH-OP, describing the interaction between the teacher $\mathcal{T}$ and the learner $\mathcal{A}$ while $\mathcal{T}$ teaches $\mathcal{K}^*_{op}$ to $\mathcal{A}$.

In TEACH-OP, $\mathcal{T}$ chooses examples that contain as many operationalizations of non–operational predicates still unknown to $\mathcal{A}$ as possible. The progresses made by $\mathcal{A}$ are measured by the error rate on the TEST set.

A rough estimate of the number of examples sufficient to learn $\mathcal{K}^*$ and OP($\mathcal{K}^*$) can be done as follows. Let $r \equiv \psi(x, \bar{y}) \Rightarrow \omega(x)) \in \mathcal{K}^*$. In order to learn r, at least one positive example of ω has to be shown to the learner. If $SPEC_k(r)$ is the first set created for r, then, $SPEC_k(r)$ has to be reduced from cardinality $t = |SPEC_k(r)|$ to $t = 1$ (in the worst case). Notice that the cardinality of $SPEC_k(r)$ can grow, when predicates are substituted by their sons in $\mathcal{G}$. In the best case, a single negative example of class ω will suffice to reduce $SPEC_k(r)$, whereas, in the worst one, the predicates have to be eliminated from $SPEC_k(r)$ one at a time. The maximum size of $SPEC_k(r)$ is upper bounded by the number of predicates occurring in $\mathcal{P}$, denoted by $|\mathcal{P}|$. As this process is repeated for each $r \in \mathcal{K}^*$, in order to learn $\mathcal{K}^*$ the teacher has to use, at most, a number of examples $\eta(\mathcal{K}^*)$ such that:

$$2R \leq \eta(\mathcal{K}^*) \leq R|\mathcal{P}|$$

In the performed experiments, $\eta(\mathcal{K}^*)$ has been close to 3R (1 positive and 2 negative examples, in average).

As concerns $\mathcal{K}^*_{op}$, the upper bound of the number of operationalizations that have to be shown is:

$$\sum_{Q \in \mathcal{C}^* \cup \mathcal{P}} t(Q)$$

which is also the maximum number of examples that could be necessary to show to the learner.

5. Experiments

In this section the results of a set of explorative experiments are reported. In the application described in Section 2, we have:

$$\sum_{Q \in \mathcal{P} \cup \mathcal{C}^*} t(Q) = 848 \qquad (5.1)$$

The teacher randomly generates an independent test set TEST containing M = 400 examples.

By applying the algorithm SELECT, the teacher $\mathcal{T}$ presents to $\mathcal{A}$ a first example $\zeta_{(1)}$[6], which is a positive example of the class HAND–TAKE. This example correctly follows the right–most causal path in Fig. 3.1, generating the rule:

$r_1 \equiv$ GRASPABLE(x) $\wedge$ THIN(x) $\wedge$ ROUGH(x) $\wedge$ HAND-MEDIUM-WEIGHT(x) $\Rightarrow$
$\Rightarrow$ HAND-TAKE(x) (5.2)

Moreover, this example follows another causal path, generating the rule:

$r_2 \equiv$ HOOK–GRASP(x) $\wedge$ HOOK–LIGHT(x) $\Rightarrow$
$\Rightarrow$ RAISE(x) (5.3)

Rule (5.3) is incomplete because of the absence of the predicate 1-HOLED-HANDLE(x) in $\mathcal{C}$.

At the same time, $\mathcal{A}$ builds up the justification forest for $\zeta_{(1)}$, a part of which is reported in Fig. 3.2. After $\zeta_{(1)}$, the correct classification rate on the test set is 44%. Both generated rules are over general: r_1 because $\mathcal{T}$ knows that the object to be taken must not be fragile (prediction of the effect of the grasp by means of the part of $\mathcal{C}^*$ below the node HAND–LIFT(x)), and r_2 because of the absence of the predicate 1-HOLED-HANDLE(x). Only the set $SPEC(r_1)$ is generated:

$SPEC(r_1)$ = {ROUGH-CONTACT, GRASPABLE, HOOK-GRASP, NOT-FRAGILE, NOT-ON-FLOOR, NOT-IN-CORNER, ..., METAL, THIN, HAND-LIGHT, HAND-MEDIUM-WEIGHT, HIGH-WEIGHT, LIGHT-WEIGHT}

At this point, $\mathcal{A}$ shows $K_2 = \{r_1, r_2\}$ to $\mathcal{T}$, who knows that $H_1 = \emptyset$ and decides to modify r_2, by showing to $\mathcal{A}$ a positive example, $\zeta_{(2)}$[7], of the class RAISE. Rule r_1 is not modified, whereas rule r_2 is augmented with the set $SPEC(r_2)$ containing all the predicates true of $\zeta_{(2)}$, and false of $\zeta_{(1)}$ (less the predicates occurring in the antecedent of rule r_2):

$SPEC(r_2)$ = {S-R-CONTACT, CLEAR, 1-HOLED-HANDLE, FRAGILE, SEAT-FURN, TWO-CURVE, THREE-CURVE, COOKWARE, ...}

As $SPEC(r_2)$ still contains unuseful predicates, $\mathcal{T}$ presents a near-miss (to speed up the detection of unuseful predicates in $SPEC(r_2)$)

[6] A fork with a not holed handle, located on a small table.
[7] A cup with a holed handle, located on a stool.

of the class RAISE, i.e. an example $\zeta_{(3)}$ of the class NOT–MOVE. After seen this example, the set SPEC(r_2) is reduced to the single predicate{1-HOLED-HANDLE(x)} and the teacher moves r_2 to H_3. Then, $\mathcal{T}$ tries to let $\mathcal{A}$ complete r_1, and so on. After seen 9 examples (1 of class HAND–TAKE, 4 of class NOT–MOVE, 1 of class HAND–TAKE $\wedge$ ARMS–TAKE and 1 of class PUSH $\wedge$ HAND–TAKE) the learner has acquired a knowledge base K_9, containing 8 out of the 17 rules belonging to $\mathcal{K}^*$, with a correct classification rate of 65.5% on the test set.

At this point, the problem that the teacher has to face is to teach rules containing a disjunction of predicates, as it would be necessary for the class PUSH (see Fig. 5.1). For instance, if the teacher wants to supply the learner with the information contained below the node CRAWL in the network of Fig. 5.1, he should tell the learner that the rules found from the paths starting in the first causes and ending in CRAWL have to be completed with the following disjunction:

ON-FLOOR$\vee\neg$FRAGILE(x)$\vee$[(FLOOR(y)$\wedge$SOFT(y)].

In other words, each of the three rules implying PUSH should be transformed into three others, each one with one of the above disjunct added. By proceeding as before, $\mathcal{A}$ arrives at the final $\mathcal{K}^*$ with n = 21 examples. When $\mathcal{K}^*$ is reached, OP(K_{21}) is still largely incomplete.

The teacher starts now to provide examples with new operationalizations, handling one rule at a time. Let us consider, for instance, the rule:

$$\text{GRASPABLE}(x) \wedge \text{THIN}(x) \wedge \text{SMOOTH}(x) \wedge \text{HAND-LIGHT}(x) \wedge \neg\text{FRAGILE}(x) \Rightarrow \Rightarrow \text{HAND-TAKE}(x) \qquad (5.4)$$

We have, from $\mathcal{P}$:

t(GRASPABLE) = 3 t(SMOOTH) = 9
t(HAND-LIGHT) = 1 t(THIN) = 1
t($\neg$FRAGILE) = 23

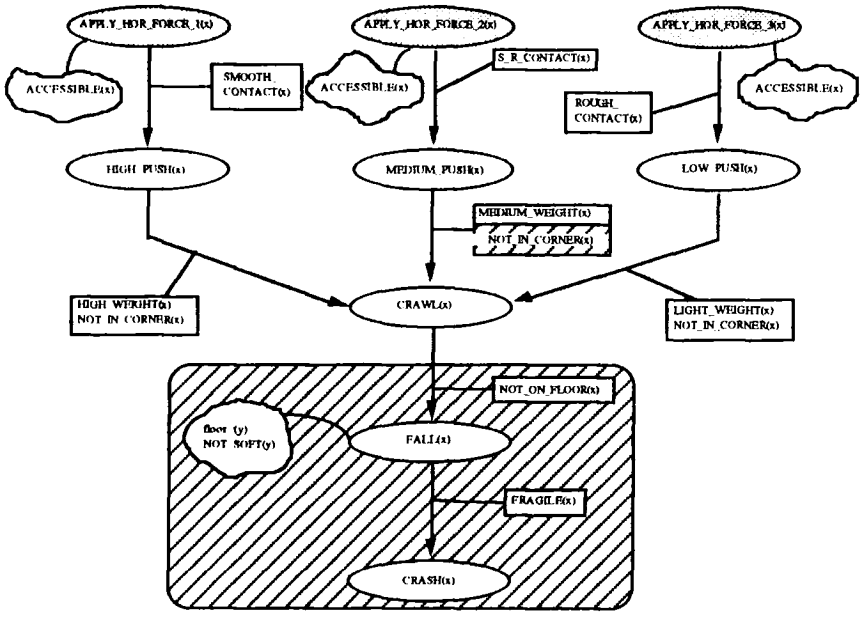

Fig. 5.1 – Part of $\mathcal{C}^*$ referring to the class PUSH.

Even if the number of possible operationalizations of (5.4) is:

$$\prod_{j=1}^{5} t(Q_j) = 3 \cdot 9 \cdot 1 \cdot 23 \cdot 1 = 621$$

the teacher does not need to show to $\mathcal{A}$ the number of examples. In fact, the maximum number he has to present is:

$$\sum_{j=1}^{5} t(Q_j) = 3+9+1+23+1 = 37$$

provided that each example satisfies (5.4).

Actually, the number of examples to be generated is even less, because only 16 out of the 23 not fragile materials are also smooth.

In the presented application we obtained a number of examples equal to 212 for learning the complete $\mathcal{K}^*_{op}$.

6. Conclusions

This paper presents the preliminary results of a set of experiments in training example ordering. These results suggest that the presence of a deep model of the domain can greatly help

a teacher in suitably selecting and ordering training examples in such a way to speed up the convergence of the acquired knowledge base towards a maximum of performance.

A main conclusion already emerging is that the use of deep models of the domain, even limited and qualitative, drastically reduces the number of training examples needed to learn. In the application chosen, a purely inductive system cannot find anything useful unless it is presented with thousands of training examples. Experiments done confirm this intuition.

A set of interesting issues emerged during this preliminary study, deserves to be further investigated:

- To find out how an increasing reduction of the domain theory, known by the learner, affects the number of examples needed to obtain a given classification error ν_M.
- To compare different teacher's strategies for selecting and ordering examples.
- To try another learning setting, in which both the teacher and the learner are artificial.
- To deal with the case in which both the teacher and the learner have imperfect knowledge and, hence, both learn, because of their interaction.

References

[1] Cornuejols, A. (1989). "An Exploration into Incremental Learning: The INFLUENCE System". Proc. *6th Int. Machine Learning Workshop* (Ithaca, NY), pp. 383-386.

[2] Craw, S. & Sleeman, D. (1990). "Automating the Refinement of Knowledge-Based Systems". Proc. *ECAI*, (Stockholm, Sweden), pp. 167-172.

[3] Fisher, D. H. (1987). "Knowledge Acquisition via Incremental Conceptual Clustering". *Machine Learning*, 2 , 139-172.

[4] Lebowitz, M. (1987). "Experiments with Incremental Concept Formation: UNIMEM". *Machine Learning*, 2, 103-138.

[5] Utgoff, P. (1989). "Incremental Induction of Decision Trees". *Machine Learning*, 4, 161-186.

[6] Giordana A., Saitta L., Bergadano F., Brancadori F. & De Marchi D. (1993). "ENIGMA: A System that Learns Diagnostic Knowledge". *IEEE Trans. on Knowledge and Data Engineering*. In press.

[7] Widmer, G. & Kubat, M. (1992). "Learning Flexible Concepts from Streams of Examples: FLORA2". Proc. *ECAI-92* (Vienna, Austria), pp. 463-467.

[8] Valiant, L. (1984). "A theory of the learnable". *Communications of ACM*, 27, 1134–1142.

[9] Weiss, S. M., Ginsberg, A. & Politakis, P. (1988). "Automatic Knowledge Base Refinement for Classification Systems". *Artificial Intelligence*, 35, 197-226.

[10] Winston, P. H. (1975). "Learning Structural Descriptions from Examples". In P. Winston (Ed.), *The Psychology of Computer Vision*, McGraw Hill (New York, NY).

[11] Fisher, D. H., Xu, L. & Zard, N. (1992). "Ordering Effects in Clustering". Proc. *9th Int. Conf. on Machine Learning* (Aberdeen, UK), pp. 163-168.

[12] MacGregor J. (1988). "The Effects of Order on Learning Classifications by Example: Heuristics for Finding the Optimal Order", *Artificial Intelligence*, 34, 361-370.

[13] Gold M. (1967). "Language Identification in the Limit", *Information & Control*, 10, 447-474.

[14] Angluin D. & Smith C. (1983). "Inductive Inference: Theory and Methods", *Computing Surveys*, 15, 237-269.

[15] Mitchell T. (1982). "Generalization as Search", *Artificial Intelligence*, 18 , 203-226.

[16] Baroglio, C., Botta, M. & Saitta L. (1992). "WHY: A System that Learns from a Causal Model and a Set of Examples". In R. Michalski & G. Tecuci (Eds.), *Machine Learning: A Multistrategy Approach, Vol IV*, Morgan Kaufmann (Los Altos, CA).

[17] Saitta, L., Botta, M., Ravotto, S. & Sperotto, S. (1991). "Improving Learning by Using Deep Models". Proc. *First International Workshop on Multistrategy Learning* (Harpers Ferry, WV), pp. 131-143.

[18] Saitta, L., Botta, M. & Neri, F. (1993). "Multistrategy Learning and Theory Revision". *Machine Learning*. 11. In press.

[19] Bergadano, F., Giordana, A. & Saitta, L. (1988). "Automated Concept Acquisition in Noisy Environments". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-10 , 555-578.

[20] Bergadano, F. & Giordana, A. (1990). "Guiding Induction with Domain Theories". In R. S. Michalski & Y. Kodratoff (Eds.) *Machine Learning: An Artificial Intelligence Approach, Vol. III*, Morgan Kaufmann (Los Altos, CA), pp. 474-492.