EVALUATION OF LEARNING SYSTEMS : AN ARTIFICIAL DATA-BASED APPROACH[1]

H. LOUNIS G. BISSON
LRI, Equipe Inférence et Apprentissage
Université Paris-Sud, Bâtiment 490
91405 Orsay Cedex France
email : lounis@lri.lri.fr, bisson@lri.lri.fr, phone : (33) 69-41-64-09

Abstract

Experimentation has an important role in determining the capacities and restrictions of machine learning (ML) systems. In this paper we present the definition of some sensitivity and evaluation criteria which can be used to perform an evaluation of learning systems. Moreover, in order to overcome some of the limitations of real data sets, we introduce the specification of a parametrable generator of artificial learning sets which allows us to make easily complete experiments to discover some empirical rules of behavior for ML algorithms. Finally, we give some results obtained with different algorithms, showing that artificial data bases approach is an interesting direction to explore.

Keywords : Evaluation of Machine Learning Algorithms, Sensitivity Criteria, Evaluation Criteria, Artificial Data Base, Parametrable Generator, Modelization of Learning Domains.

1 Introduction

In Machine Learning and more generally in Artificial Intelligence domains, it is very difficult to completely evaluate a system through previous theoretical reasoning (Kibler, Langley 1988 ; Rendell 1989 ; Bisson, Laublet 1989&1990). This drawback comes mainly from the large utilization of heuristics in AI, the behavior of which is difficult to predict. Thus, information such as the actual complexity of an algorithm, is rarely provided by the authors and when it is, it often concerns the best and worst cases only. Nevertheless, the knowledge of its complexity is fundamental for those who

[1] This work is partially supported by CEC through the ESPRIT-2 contract MLT 2154 ("Machine Learning Toolbox") and also by MRT through PRC-IA.

want to use a system. On the other hand, some concepts such as the quality of learning, are subjective and very hard to define. In this context, the main goal of experimentation is to point out some relations between behaviors of learning systems and the conditions under which they occur.

In order to experiment we must set the characteristics of the output data (evaluation criteria) that we would like to measure and what are the relevant variables for these measures (sensitivity criteria). In this way, we determine a measure space that is characterized by the set of pairs [criterion (i), variable (j)]. However, for a given learning system, only a subset of these measures are feasible : for instance, the tests evaluating the bias influence or the degree of incrementality (Martin 1989), are feasible only if the system has these features. Moreover, according to the learning techniques and the goals of the system (diagnostic, problem solving, ...), the criteria change : in this way, for planning problems, the problem solving time is an important criterion that allows one to quantify the efficiency of the learned knowledge, however, this criterion is not very significant for classification problems.

In practice, it is useless to work out all the theoretically feasible tests, because this would cost too much time or would not be relevant when considering the purpose of the evaluation. So, the experimenter must decide the most useful tests in respect to this evaluation purpose. This paper is organized in three parts. In the first one, we detail how to execute an experimental study of learning systems by relating some of the sensitivity and evaluation criteria. In the second one, in order to establish correlations between the two types of criteria and to bring some answers to the limitations of real data sets, an artificial data base generator is proposed and its specifications are given. Finally, we show some results obtained with five different systems belonging to the supervised learning approach : CN2 (Boswell 1990 a), NewID (Boswell 1990 b), NewBOOLE (Bonelli 1990), LVQ (Mc-Dermott 1989) and MLP (Rumelhart 1986). We want to emphasize that the current work was partially done as part of the EEC ESPRIT contract "Machine Learning Toolbox". It has been integrated in the work package 7, whose goal is to develop an evaluation methodology of ML algorithms. The reader could find more details in the Deliverable 7.2 (MLT 1990) of this project.

2 Sensitivity criteria

The most important problem in inducing suitable knowledge is the problem of defining a "good" set of attributes and a "good" set of examples to represent a problem to be learned. The ML algorithms are sensitive to various characteristics of the information provided as input and they do not have the same behavior ; for example, some are more or less noise resistant, or more or less sensitive to the input order of the examples. The description and the choice of examples for a domain problem is of very great importance for the user of a ML algorithm. The sensitivity criteria correspond to the parameters on which depend the quality of the learned knowledge.

Information quantity : One of the main experimentations is concerned with the study of observable relations between the performance measures and the information quantity. On the one hand, they are relatively easy to achieve, if we limit ourselves to syntactic criteria. On the other hand, they are particularly informative about the main properties and limitations of the learning system. The size of the training set can be defined by several parameters such as the number of examples or descriptors, the number of concepts to learn or the quantity of available background knowledge ...

Problem complexity : In addition to the quantitative variables just described, we would like to define more qualitative criteria such as the difficulty of the problem that the learned knowledge must help to solve. These criteria are more domain dependent. For instance, in the case of concept recognition, the complexity can be measured by the number of conjunctions and disjunctions in the recognition functions (Rendell 1989). The presence of noise is another complexifying factor. Moreover, if the studied system types the descriptors or the data, it will be relevant to evaluate the learning results according to the types manipulated in the training data (nominal, ordered, ...).

Relevance of descriptors : In complex domains, the determination of the relevant descriptors is a very difficult problem (Cannat 1988). Thereby, the expression of knowledge can not be immediately suitable and there are often a lot of irrelevant descriptors in the initial training set. The system's capacity to learn in such an environment must be evaluated. We distinguish this variable from those about noise : here, all the needed information to learn are present and undamaged.

Information order : The use of pruning heuristics can modify the result of learning according to the input order of information within the training set. The study of its influence provides some information about the *learning stability* . These tests are different from those about the incrementality : the measure of stability is defined as soon as a learning step uses several set of data *simultaneously*. A low stability is awkward : the reliability of learned knowledge becomes questionable. Moreover in this case, it is probable that the learning process is very noise sensitive because a lack of stability reveals that the system does not use the data globally but incrementally when it builds its hypothesis.

Noise influence : These tests aim at measuring the noise sensitivity of the studied learning system. Indeed, in the real world domains, the noise is very difficult to erase because it is often linked to the acquisition process : for instance, the uncertainty on the measure instruments. There can be multiple sources for noisy data in a learning set (Manago, Kodratoff 1987) such as the uncertainty or imprecision of the valuation of the descriptors or a miss-classification of some examples.

Incrementality degree : These tests are very close to those effected to evaluate the stability of learning ; however, the conclusions are different. With an incremental system, we can accept that the

order of information in the training set has some consequences for the quality of learning. During these measures it is also interesting to quantify the problem of *forgetting*. With this aim in view, we shall verify if the problems solved by the learned knowledge at time (t) are always solved at (t+1).

Learning bias and heuristics : When the system provides the possibility of changing the learning strategies (heuristics or bias), the experimenter must test their influences on the results of the learning process. During these measures, some empirical relations between the domain characteristics and the parameter values could be profitably established.

3 Evaluation criteria

Evaluation criteria were elaborated in order to exploit the notion of having a "good" result for a ML algorithm. In fact, "good" means that a result succeeds in a selection of tests of performance. Generally, the criteria which are most frequently used for evaluating ML algorithms involve the study of learning efficiency, since the main interest of a learning system is to build up a knowledge base able to effect accurate prediction. Nevertheless, the other criteria are also important : for instance, the understandability to a human user of "what is learned" is often crucial in order to improve the learning set ; similarly, the constraints of time can restrain the application domains. We can classify the learning evaluation criteria into five major categories which are :

Cost of the learning set : Learning can be an expensive process, particularly to constitute the training set. Therefore, it is very important to evaluate the cost of this preliminary phase. Independently of specific domain costs (medical tests for instance), this cost is obtained by measuring the amount of data required for learning (number of examples, ...), as well as by evaluating the required quality of these data (no noise, domain theory needed, ...). In the case of learning apprentice systems, the cost also takes into account the number and the relevance of questions asked.

Time and memory constraints : The measures of learning rapidity allow us to evaluate empirically the actual complexity of algorithms as a function of different parameters : the amount of data, the used heuristics, ... In practice, with this information, users will avoid making use of systems that are O(number of examples^3) on very large training sets. In the case of systems that learn by successive refinements, connectionist systems for example, this criterion and the previous will also depend on the time and the number of observations required to obtain a specified accuracy level of the learned knowledge. In EBL we will measure the time to generalize a specific problem's solution (Shavlik 1989). The memory measure is less important. Nevertheless, the quantity of memory used during the learning phase indicates the usable computers for these learning techniques. Moreover, this criterion is related to the efficiency of the learning.

Effectiveness of the learned knowledge : When the purpose of learning is to obtain knowledge usable for classification or diagnosis, the main feasible measure concerns the predictive accuracy. Most empirical research on induction has focused on improving this criterion that measures the capability of the system to match a new situation with the learned expertise (Quinlan 1986). For planning problems, the quality of a solution will be evaluated by the number and the cost of operations used in the solutions of the problems. In the case of EBL, as the quality of learning is very time gain dependant, we have to take into account the memory used and the processing time. As a matter of fact, the learned knowledge can let the system resolve some problems that were too "expensive" to solve before learning. During the use of the test set, the predictive systems can provide three kinds of answers : right prediction, wrong prediction or no answer. Generally, the authors are just interested by the correct predictions, nevertheless, the study of the two other kinds of answers is also relevant and they must not be mixed. For example, in the case of wrong predictions, it would be useful for the experimenter to know the "distance" existing between the different concepts to identify in order to evaluate the "severity" of an erroneous answer.

Efficiency of the learned knowledge : The time required to solve a problem, corresponds to the indexation quality of learned knowledge (rules ...). In others terms, it reflects the structure of the knowledge base. Most analytical work has focused on efficiency. In fact, for planning systems and especially for EBL, the time is the most important measure because the purpose of learning is to accelerate problem solving. Another measure will be the number of searched nodes (Minton 1985). In decision tree systems, the indexing quality can be roughly linked to some syntactical criteria such as the height, the breadth and the number of nodes in the produced trees.

Intelligibility of results : One of the main advantages of symbolic learning systems consists in that the learnt knowledge is in a readable form. This property helps the system users to understand the results, and also to find and to rapidly correct mistaken or missing information in the training set. The intelligibility of the learned knowledge may be appreciated syntactically and semantically. The number of disjunctions and conjunctions may be a measure of the intelligibility. For decision trees, the smaller and more well balanced the tree produced, the more it will be readable and understandable. Another measure, attempting to consider the semantics of the results, is based on the type of analysis a human expert does when he looks at the induced knowledge base. The more an expert can identify prototypical situations, the more the knowledge is understandable.

4 The artificial data base generator

4.1 Introduction

The problem to solve for the experimenter is to choose the data sets that will be used for the experiments. A great number of different data sets are currently available in the published literature such as : soja diseases, iris features, thyroid cancer symptoms, lymphography ... These bases are useful for the designers to compare homogeneously the learning systems, however, this kind of approach presents four drawbacks that we are going to explain below.

Availability of the data base for experiment : In order to experiment, we must find data sets simultaneously available, usable by the system and, above all, carefully acquired. They must, for instance, contain enough examples or a domain theory. However, at the beginning of an experimental phase, it can be difficult to know exactly the type and amount of information needed. Moreover, in order to be significant, an experimentation must use several data sets taken from different domains.

Translation of the data into the studied system representation : Generally, the syntax of the found data is different from that used by the studied system, so a translation phase is needed. However, there are two problems : firstly, the translation is time consuming ; secondly when the translation is done by a non-expert of the domain, some information of the initial base can be damaged because the syntactic level and the semantic one are often linked.

Difficulties in the modification of the data in the course of experimentation : Even if an expert is available, it is not so easy to modify real data sets. Therefore, during the experimentation, it is not always possible to study a specific aspect of the behavior of the system unless a more adequate data set is available. On the other hand, the characteristics of the examples are not often well known. For instance, the noise quantity and its localization are generally ignored.

Difficulties in the correct evaluation of the learned knowledge : Usually, for evaluating the results of learning, as in data analysis, the researcher splits the experiment set in two parts : a training set for learning and a test set. However, by using the data sets from real domains, the evaluation can lack in precision without an expert (Ganascia, Helft 1988). For instance, in medical domain, an expert system under-estimating the gravity of a disease in 20% of cases, is far more dangerous that another one over-estimating the disease in the same proportion !

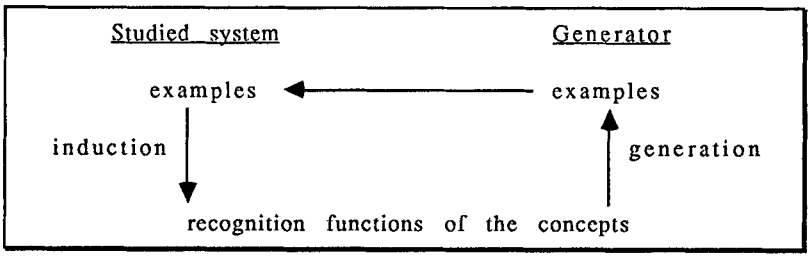
To take into account these problems that restrict strongly the benefits of real data sets, we propose the use of artificial data sets. This approach is not completely new (Quinlan 1988 ; Martin 1989), but its use was limited mainly to randomly introducing noise into data. Our approach is more

general since we would like to simulate the many different learning domains in which a ML algorithm might be applied, by using a generator (Bisson, Laublet 1989&1990) that the user could tune with parameters. This method allows to bring some answers to the limitations of real data sets. Firstly, the availability of data is no longer a problem : the data sets are generated in case of need and these data sets are homogeneous and give reliable criteria of comparison. Secondly, the translation stage becomes very simplified : One translator is enough to transform the generator outputs into inputs of the learning system, instead of a specific translator for each real data set. Finally, it is easier to answer the question : "what happens if the application domain was different ?", by changing the values of parameters and therefore the modeling of the domain performed by the generator. Moreover in this approach, the interpretation of the results can be easily done by the experimenter himself.

However, two questions become apparent. The first one concerns the validity of the evaluations obtained by this bias. The answer depends completely on the degree of realism of artificial data sets in comparison with the real data sets. In other terms, it depends on the fitness of the model of application domain. The second question obviously relates to the generator feasibility.

4.2 Generation algorithm

In this study, we have limited ourselves to the supervised learning approach. Algorithms within this learning paradigm cover a large number of current realizations. Broadly speaking, in this field, the learning problem is always the same : from a set of classified examples, the system must learn characteristic or discriminant functions that will be used to associate a class (or expertise), with each situation (or context) of the domain. All the recognition functions that were acquired in this way are elements of a knowledge base usable as a part of an expert-system. For generating the artificial data sets, the starting idea is that this inductive process could be inverted :



In the first stage, recognition functions of concepts are fixed by the experimenter himself or randomly produced by the generator whose parameters are tuned by the user. Then, this tool generates a set of examples from these recognition functions. The control parameters of the generator allow us to simulate an application domain and correspond logically to the previously seen sensitivity

parameters such as : the number and types of the descriptors, the number of conjunctions and disjunctions in the class descriptions, the noise intensity and its localization , ...

Currently, we have implemented two versions of the generator, the first one is working in attribute value logic and the second one is a predicate logic version. For both systems, the generation process is divided into four steps. We are going to detail these steps :

1) *Creation of a vocabulary* : This stage consists of creating the descriptors used to describe the learning set (concepts, examples and domain theory) and for each of them to associate a type and a set of possible values. This process can be performed automatically by a random generator or manually ; when it is manual, the users chooses the name, the type and the domain of each descriptor.

Ex : *descriptors randomly produced by the generator :*

$att_1, att_2, att_3, ...$ *(attribute value logic)*

$P(?x1, ?x2), Q(?x1), ...$ *(predicate logic)*

descriptors fixed by the experimenter :

height, hair, ...

height (?x1, ?x2), hair (?x1, ?x2), ...

2) *Creation of the recognition functions* : When the descriptors are generated, with respect to the given parameters, the generator builds up the logical functions which characterize the set of generated concepts. The logical operators used are AND (Λ) and OR (V) and the recognition functions RFi are written under normal disjunctive form. In the current release, for a given attribute (or argument of predicate) the values are selected randomly following a uniform distribution law.

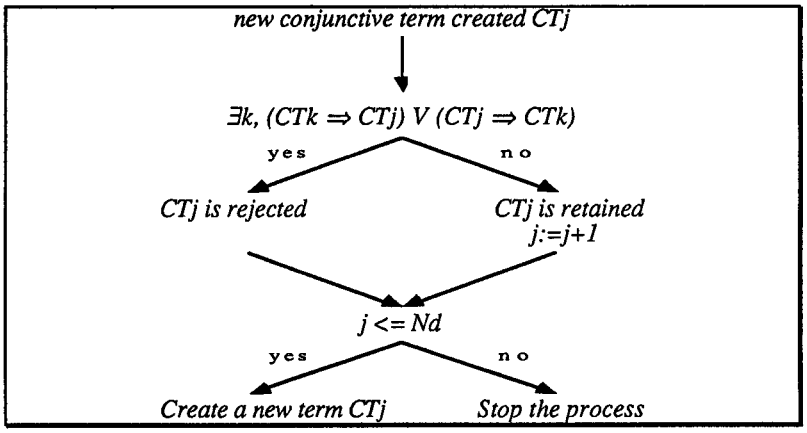
$RF_i = V_{j=1..Nd}\ CT_j$

with $CT_j = \Lambda_{k=1..Nc}\ (att_k\ sel_k\ val_k)$ *(attribute value logic)*

or $CT_j = \Lambda_{k=1..Nc}\ [not]\ P_k\ (v_1, v_2, ..., v_n)$. *(predicate logic)*

v_i *: variable or constant, Nc = number of conjunctions, Nd = number of disjunctions*

However, the main problem to solve during this stage, is that the generator must assume that two different functions $RF_1=V_jCT_j$ and $RF_2=V_kCT_k$ could not be verified at the same time by confirming that : $\forall j, \forall k$, not $(CT_j \Rightarrow CT_k)\ \Lambda$ not $(CT_k \Rightarrow CT_j)$. Like in the previous stage, the experimenter can define himself his own recognition functions. The algorithm used for creating automatically a recognition function is divided into the following steps.



Examples : RFi : ($att_1 = val_{11} \Lambda att_4 = val_{42}$) V ($att_5 > val_{52}$) *(attribute value logic)*

RFj : height (?x1, tall) Λ eyes (?x1, brown) *(predicate logic)*

3) *Generation of the examples* : For constructing an example, firstly, the generator randomly chooses one concept Ci, then it selects one conjunctive term CT in the recognition function of Ci. The example is built up around CT by adding some new terms to the conjunction. As during the elaboration of the recognition function, the system must take care that the new example belongs to one concept only ; the criteria to verify are the same as previous.

Examples (the recognition functions RFi and RFj are the same as previously) :

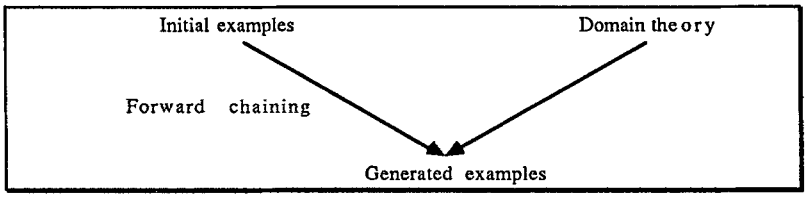
ei ∈RFi : $< class = Ci >$

$att_1 = val_{11} \Lambda att_2 = val_{22} \Lambda att_3 = val_{31} \Lambda att_4 = val_{42} \Lambda att_5 = val_{51}$

ej ∈RFj : $< class = Cj >$

height (Paul, tall) Λ eyes (Paul, brown) Λ hair (John, dark)

4) *Generation of a domain theory* : Even if the construction of a domain theory is not needed in all cases, it will be interesting to perform for some other systems such as CHARADE (Ganascia 1987) or KBG (Bisson 1991). In the current release, the generator is able to build up a domain theory in the form of a set of rules. The method used is very simple : we assume that the previously produced examples result from the virtual application (by forward chaining) of a domain theory on the initial examples ; this corresponds to the following scheme :



Then the idea is to "reverse" this process ; the method for creating one rule is as follows. Firstly, the generator chooses randomly a term T appearing in one or several examples ; this one will constitute the conclusion of the rule. Secondly, the system randomly builds the premises T_1, T_2, ... of this rule, taking notice of the two following constraints:

1) The premises of the rule do not verify an existing recognition function.
2) The premises of the rule do not subsume and are not subsumed by the premises of the previously generated rules.

Next, when the rule is created, the examples containing the term T are rewritten by applying the rule in backward chaining. Finally, these rewritten examples and the rules constitute the initial set of examples and the domain theory.

Ex : *Example e_i : $att_1 = val_{11} \Lambda\, att_2 = val_{22} \Lambda\, att_3 = val_{31} \Lambda\, att_4 = val_{42} \Lambda\, att_5 = val_{51}$*
Rule k : $att_7 <= val_{74} \Lambda\, att_6 = val_{62} \rightarrow att_2 = val_{22}$
The rewritten example e_i :
$att_1 = val_{11} \Lambda\, att_7 = val_{72} \Lambda\, att_6 = val_{62} \Lambda\, att_3 = val_{31} \Lambda\, att_4 = val_{42} \Lambda\, att_5 = val_{51}$

Example ej : height (Paul, tall) Λ eyes (Paul, brown) Λ hair (Paul, dark)
Rule l : eyes (?x1, brown) --> hair (?x1, dark)
The rewritten example ej :
height (Paul, tall) Λ eyes (Paul, brown)

4.3 Parameters of the generator

For this first version of the generator, we have chosen a set of parameters mainly based on syntactic criteria described during the presentation of the sensibility criteria in a previous section. However, in the future, it would be possible to increase their number and to introduce some more semantic criteria. The parameters of the attribute value logic version are the following :

Number and type of descriptors : the generator provides five standard types to the users which are : nominal, ordered, taxonomy, integer and real. Here are the description, for the attribute value

generator, of the semantics of each one and the list of authorized selectors. The predicative logic version works similarly, indeed, each argument of each predicate is typed :

Nominal	: set of different values	: =, <>
Ordered	: set of ordered values	: =, <>, <, <=, >, >=
Taxonomy	: tree of values	: =, <>, <, <=, >, >=
Integer	: interval of number	: =, <>, <, <=, >, >=
Real	: interval of number	: =, <>, <, <=, >, >=

So, the user can specify the employed descriptors as following :

att1 nominal (red blue green)
att2 ordered (small medium large)
att3 taxonomy (shape (rectangle (square)) circle)
att4 integer (0 100)

Number of classes : The user gives the number of possible classes for the examples ; for each class, the generator creates a symbol and its recognition function. In the current version of the generator, it is also possible to choose "by hand" the recognition function : in that case the system will just check there are no subsuming links between the different recognition functions.

Number of conjunctions and disjunctions : these two parameters guide the generator to elaborate the recognition function of the different concepts. Both numbers are described as an interval setting the lowest and highest possible values. So, the user can say for example that the conjunctive (or disjunctive) parts of the recognition functions must have between two and four terms.

Constraints on the variables : for the predicative logic version of the generator, there are two additional parameters, which control the generation of the variables into the predicates during the elaboration of the recognition functions (RF). The first parameter gives the repartition percentage between the variables and the constants in the RF. The second parameter controls the number of occurrencse of each variable in the RF. It corresponds to the probability of creation of new variables.

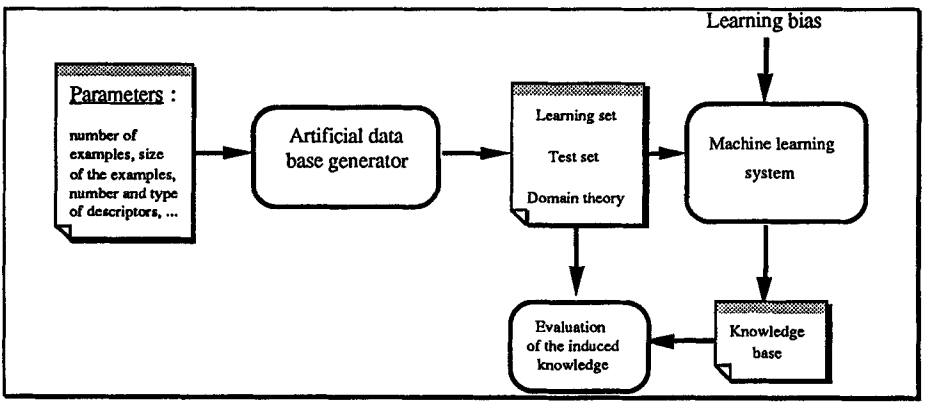
Number of examples : the number of classified examples that the generator must build up. Generally, the set of examples is split in two parts, corresponding to the training set and the test set.

Size of the examples : this parameter corresponds to the number of descriptors in the examples. This parameter is directly linked to the sensitivity criterion concerning the irrelevant information : indeed, when you increase the number of terms, you add some irrelevant parameters too.

Noise intensity : the user can introduce some noise into the examples. For instance, if the user sets this parameter to the value 10, he assumes that 10% of the generated examples are noisy. Here, the word "noise" has a very precise meaning : we say that an example is noisy when it can simultaneously belong to two different concepts. Several other types of noise remain to be implemented such as the attribute errors and the value errors.

Unknown and Don't care values : the generator handles the "unknown" and "dont'care" values. They correspond to an existential (respectively universal) quantification of the concerned attribute (Boswell 1990 a&b)). The user can choose the name of the attributes which have these kind of values and also the percentage of unknown and dont'care values for each one.

Number of rules and size of premises : the user fixes the number of rules that he wants in the domain theory and the size of the premises in the rules (number of terms).



Working of the generator with a ML system

5 Application to some learning algorithms

The goal of experimenting learning systems is to vary sensitivity parameters in order to determine the algorithm's behavior over a range of situations. We have tested our generator with five learning systems. All of them use the attribute value representation :

Algorithms	Authors	Learning method
CN2	Boswell 1990 a	Rule induction program based on the AQ algorithm
NewID	Boswell 1990 b	ID3 program which can generate a pruning tree
NewBOOLE	Bonelli 1990	Genetic based learning system
LVQ	Mc Dermott 89	Classifier based on the Euclidian distance
MLP	Rumelhart 1986	Multi layer perceptron using back propagation

For all the experimentations, we have fixed the different parameters of the generator in order to compare the five algorithms to the following values :

- Number of attributes : 10 nominal attributes
- Number of classes : 3
- Number of conjunctions : [2 .. 4]
- Number of disjunctions : [1 .. 2]
- Size of the examples : 10
- Number of examples : 200 (100 for learning and 100 for testing)
- No domain theory

Number of possible values (which the distribution is uniform) for each attribute:

a1	a2	a3	a4	a5	a6	a7	a8	a9	a10
2	5	5	4	2	4	7	4	6	3

The recognition functions randomly set for each concept (class) are :

Class C1 : $att_2 = val_{22} \Lambda\ att_6 = val_{62}$
Class C2 : $att_1 = val_{11} \Lambda\ att_4 = val_{41} \Lambda\ att_5 = val_{52}$
Class C3 : $(att_{10} = val_{102} \Lambda\ att_4 = val_{43} \Lambda\ att_6 = val_{63}) V\ (att_9 = val_{92})$

The measures that are consigned in the following table are the following :

NR	= number of induced rules	%RP	= % Right Prediction
N	= number of nodes	%WP	= % Wrong Prediction
L	= number of leaves	%NP	= % No Prediction
H	= mean of the way heights		

5.1 Learned knowledge as a function of the size of the learning set

These experiments aim at defining an empirical relation between the size of the learning set (for a given set of recognition functions) and the effectiveness and the size of the learned knowledge. Two tests have been done, the first one is performed with noise free examples, and the second one in an environment in which 10% of the attributes coding the class of the examples are noisy.

1) Results with 0% noise :

	25 learning examples					50 learning examples					75 learning examples					100 learning examples				
	CN2	NewID	NBOOL	LVQ	MLP	CN2	NewID	NBOOL	LVQ	MLP	CN2	NewID	NBOOL	LVQ	MLP	CN2	NewID	NBOOL	LVQ	MLP
NR	6					5					5					4				
N		17					33					38					44			
L		13					27					28					32			
H		2.58					3.22					3.32					3.32			
% RP	100	74	97	72	71	100	94	100	76	86	100	95	100	78	93	100	95	100	78	97
% WP	0	13	3	28	29	0	0	0	24	14	0	0	0	22	7	0	0	0	22	3
% NP	0	13	0	0	0	0	6	0	0	0	0	5	0	0	0	0	5	0	0	0

The obtained results seem reasonable. They show that the correct prediction depends logically on the size of the learning set. We can remark that CN2 and NewBOOLE obtain comparable good results and that the results obtained by the back propagation based neural net MLP are better than those obtained by the classifier LVQ.

2) Results with 10% noise :

	25 learning examples					50 learning examples					75 learning examples					100 learning examples				
	CN2	NewID	NBOOL	LVQ	MLP	CN2	NewID	NBOOL	LVQ	MLP	CN2	NewID	NBOOL	LVQ	MLP	CN2	NewID	NBOOL	LVQ	MLP
NR	8					9					11					12				
N		24					47					64					62			
L		17					36					48					47			
H		2.75					3.31					3.38					3.86			
% RP	21	20	50	44	67	82	66	55	47	69	91	69	83	47	69	95	82	88	59	78
% WP	79	58	50	56	33	18	21	45	53	31	9	16	17	53	31	5	13	12	41	22
% NP	0	22	0	0	0	0	13	0	0	0	0	15	0	0	0	0	5	0	0	0

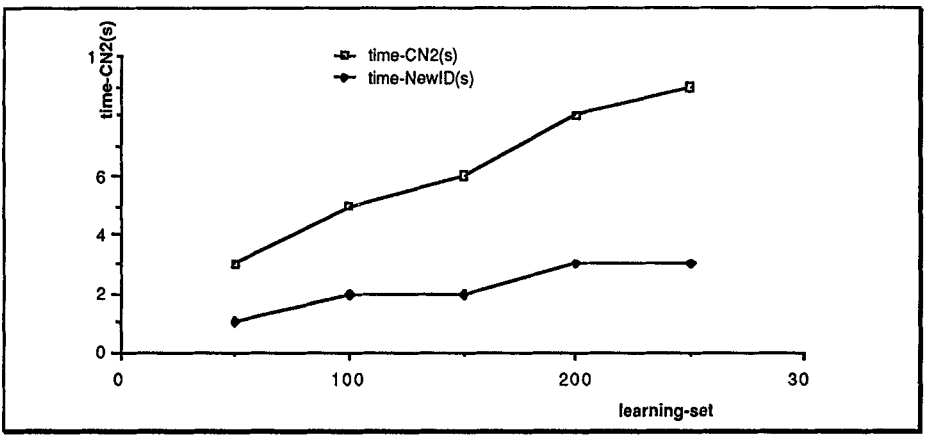
This table displays that learning in a noisy environment (error on the class information) makes the correct prediction of the different systems worse. In such an environment, CN2 and NewBOOLE also obtain the best results and we confirm that the correct prediction of MLP (based on back propagation) is better than LVQ one. In other respects, in the case of NewID we can improve the prediction capacity of the system by the use of a pruning algorithm. We obtain these results :

1) Without pruning = an average of 57% of correct prediction.
2) With pruning = an average of 62.7% of correct prediction.

However, in a noisy environment, the size of the learned knowledge (number of rules in the case of CN2 and tree size for NewID) increases. Nevertheless, the use of the pruning algorithm of NewID reduces considerably the size of this tree without damaging the correct prediction criterion.

5.2 Learning time as a function of the size of the learning set

We have also studied, the learning time of CN2 and NewID as a function of the size of the learning set (the three other systems are too slow to be compared to the two previous ones and the stopping criteria are too different). The obtained results are the following :



We remark that NewID takes less time to learn then CN2. The curve shows also that both are usable with large learning sets. These results confirm that a induction algorithm based on producing rules is inherently more time consuming than a tree-induction algorithm (Elomaa 1989).

5.3 Effect of the descriptor type on learning process

CN2 and NewID both are able to treat real attributes, so we will now examine their effect on the learning process by varying the number of real descriptors from 2 to 8 (let us recall that the total number of descriptors is 10). We obtain the following results :

	2 real descriptors		4 real descriptors		6 real descriptors		8 real descriptors	
	CN2	NewID	CN2	NewID	CN2	NewID	CN2	NewID
number of rules	5		7		7		8	
N = number of nodes		17		36		44		80
L = number of leaves		13		25		30		60
H = mean of the way heights		2.33		2.92		3.18		3.34
% right prediction	96	97	88	79	87	68	73	64
% wrong prediction	4	3	12	17	13	32	27	25
% no prediction	0	0	0	4	0	0	0	11

It appears that the two learning systems are sensitive to the proportion of numeric attributes present in the learning set. The number of correct predictions of the two algorithms decreases and the

size of the induced knowledge (number of rules and number of nodes) increases when the number of numeric descriptors increases. It is interesting to note that such an experiment would have been very difficult to make using real data sets.

5.4 Effect of Unknown and Don't care values on learning process

CN2 and NewID provide the possibility of treating Unknown values which correspond to an existantially quantified variable, or Don't care values which correspond to a universally quantified variable. Then, we study their effect on the learning process by fixing the proportion of Unknown values (respectively Don't care values) to 25 %. After several runs, we obtain the following results :

% RP	CN2	NewID
Without Unknown or Don't care values	98	83
The corrupted descriptor with don't care value do not belong to the recognition function.	92.25	76.5
The corrupted descriptor with unknown value do not belong to the recognition function.	96	81
The corrupted descriptor with don't care value belongs to the recognition function.	85	62.5
The corrupted descriptor with unknown value belongs to the recognition function.	94	74.25

First, we remark that the effect of Don't care and Unknown values is logically more marked in the case of a corrupted descriptor belonging to the recognition function. Secondly, we notice the difference between the results obtained with Don't care values and those obtained with Unknown values. The explanation is that, in the case of an Unknown value, the system splits the concerned example into a set of examples, with weights distributed according to the value distribution at that node ; in the case of a Don't care value, it replaces the concerned example by as many examples as there are possible values, but the weight of each new example is the same as that of the original (the concerned descriptor is transformed into an irrelevant one). So, in this way, we can consider that by introducing Don't care and Unknown values in the learning base, we have also introduced noise in this learning set, but the effect is less marked in the case of Unknown values.

5.5 Comparison with real world data bases

Finally, in order to show that our approach is realistic, we have compared the results obtained with two medical data sets (lymphography and breast cancer) to the results obtained with two artificial data sets having the same syntactic characteristics than the real ones :

1) *Lymphography* : 18 descriptors (all of them are nominal), 4 classes and 109 examples.

	Medical data set					Artificial data set				
	CN2	NewID	NBOOLE	LVQ	MLP	CN2	NewID	NBOOLE	LVQ	MLP
Time (s)	17	2				15	2			
NR	14					17				
number of premises	3					3				
N		78					72			
L		61					53			
H		4.27					3.54			
%RP	87	67	82	83	70	87	57	80	47	42
%WP	13	26	18	17	30	13	28	20	53	58
%NP		7					15			

The main difference between the two learning sets is that the distribution of the examples is different. This distribution is better in the artificial data set : both classes 1 (normal findings) and 4 (malignant lymphoma) in the medical data set contain respectively only one pattern.This feature has an effect on the results obtained with the two neural nets MLP and LVQ which are not significant. For the other systems it is interesting to notice that both the structure of the learned knowledge (size of tree, number of rules) and the experimentation results are very similar between these two bases.

2) *Breast cancer* : 10 descriptors, 2 classes and 269 examples.

	Medical data set					Artificial data set				
	CN2	NewID	NBOOLE	LVQ	MLP	CN2	NewID	NBOOLE	LVQ	MLP
Time (s)	19	3				19	3			
NR	32					31				
number of premises	5					4				
N		177					152			
L		132					108			
H		6.03					4.08			
%RP	72.2	69.4	77	65.12	75	75.5	70	82.5	75.5	63.95
%WP	27.8	25.1	23	34.88	25	24.5	22.5	17.5	24.5	36.05
%NP		5.5					7.5			

The distribution of the examples in this second medical base is more homogeneous. This is why we notice that the obtained results (both structure of the induced knowledge and predictive accuracy) are similar between the real world data set and the artificial one.

6 Conclusion

The aim of this paper has been to present an approach based on the use of a parametrable generator of learning sets to discover the effect of the learning environment on system performance. This generator resolves also some limitations of real data sets for such experiments and it represents a

useful tool for exploring capacities of ML algorithms using different learning methods (symbolic algorithms, genetic based algorithms and neural networks in this first contribution).

In our study of the five previous learning algorithms, we have verified with our approach some known results like the effect of noise on concept learning. We have also pointed out some new more specific informations. For example, CN2 and NewID are sensitive to the proportion of numeric attributes and that CN2 is more resistant to the unknown values than to the don't care ones. It is important to notice that the discovery of these results by using real data sets would have been *more difficult* . Indeed, the types of the descriptors are fixed in a real data base and even if an expert is available, it is not so easy to modify them. So, the experimenter can not study the effect of the descriptor type on the behavior of the system unless more adequate data sets are available.

At present time, we have restricted our study to inductive approaches such as classification and diagnosis. However, this generation method of learning sets is probably transposable to other approaches of machine learning, like learning by discovery for instance. We think that it will be desirable that a future version of our generator builds up "maquettes" of real domains in order to test ML systems with more parameters than in the present version. The next step of our work will consist in introducing some semantic parameters, such as vocabulary constraints to inhibit or authorize the creation of some conjunctive terms in the recognition functions. These semantic parameters will allow us to simulate real domains with more accuracy.

Acknowledgments

We would like to thank Yves Kodratoff, our thesis supervisor, for the support he gave to this work and all the members of the Inference and Learning group at LRI, particularly, Karine Causse, Adrian Gordon, P.Bonnelli, and also J.J.Cannat, N.Chaourar, P.Laublet, A.Mellouk.

References

Bisson, Laublet 1989, Méthode d'évaluation des systèmes d'apprentissage, Rapport de fin d'étude, ONERA-GIA, August 1989.

Bisson, Laublet 1990, Theoritical and empirical study of learning systems, Cognitiva 1990 Madrid, 1990.

Bisson 1991, Learning of Rule Systems in a First Order Representation, LRI Internal report 1991

Bonelli 1990, A.PARODI, S.SEN, S.WILSON, NewBOOLE : A fast GBML system, 7th ICML, Austin, 1990.

Boswell 1990 a, Manual for CN2, The Turing Institute, January 1990.

Boswell 1990 b, Manual for NewID version 2.1, The Turing Institute, January 1990.

Cannat 1988, Machine Learning Applied to Air Trafic Control, Proc. Human-Machine Interaction Artificial Intelligence Aeronautics & Space, pp.265-274, Toulouse, 1988.

Elomaa 1989, N.HOLSTI, An experimental comparison of inducing decision trees and decision lists in noisy domains, EWSL Montpellier, 1989.

Ganascia 1987, AGAPE et CHARADE: deux techniques d'apprentissage symbolique appliquées à la construction de bases de connaissances, Thèse d'état soutenue 4/87,Université Paris Sud.

Ganascia, Helft 1988, Evaluation des Systèmes d'Apprentissage, 3éme Journées Francaises de l'Apprentissage 1988 Cassis, p 3-20.

Kibler, Langley 1988, Machine Learning as an Experimental Science, EWSL 1988, Glasgow, pp 81-92.

Manago, Kodratoff 1987, Noise and Knowledge Acquisition, Proceeding of 10th IJCAI, Los Altos, 1987

Martin 1989, Reducing Redundant Learning, in Proceedings of the 6° International Workshop on Machine Learning , Ithaca 1989, p 396-99.

Minton 1985, Selectively Generalizing Plans for Problem Solving, IJCAI 85 p596-599.

MLT 1990, Laboratoire de Marcoussis, British Aerospace, Université de Paris-Sud, Evaluation Methods for Attribute-Based Algorithms, Deliverable 7.2, Projet ESPRIT 2154, Aout 1990.

Quinlan 1986, The effect of noise on concept learning, Machine learning 2 an artificial intelligence approach, TIOGA 1986.

Quinlan 1988, Symplifying decision trees, in B.Gaines, J.Boose Knowledge Acquisition for Knowledge-Based Systems, Academic Press 1988.

Rendell 1989, H.CHO, R. SESHU Improving the Design of Similarity-Based Rule-Learning Systems, International Journal of Expert System, pp 97-133, volume 2, Number 1, 1989.

Rumelhart 1986, G.E.HINTON, R.J.WILLIAMS, Learning internal representation by error propagation. Parallel distributed processing : Explorations in the microstructures of cognition, MIT Press, vol 1, pp 318-362, 1986.

Shavlik 1989, An Empirical Analysis of EBL Approaches for Learning Plan Schemata, in Proceedings of the 6° International Workshop on Machine Learning , Ithaca 1989, p183-87.