SIA: a Supervised Inductive Algorithm with Genetic Search for Learning Attributes based Concepts

Gilles Venturini

Equipe Inférence et Apprentissage
Laboratoire de Recherche en Informatique, bat. 490
Université de Paris-Sud
91405 Orsay Cedex, FRANCE.
email: venturi@lri.lri.fr

Abstract. This paper describes a genetic learning system called SIA, which learns attributes based rules from a set of preclassified examples. Examples may be described with a variable number of attributes, which can be numeric or symbolic, and examples may belong to several classes. SIA algorithm is somewhat similar to the AQ algorithm because it takes an example as a seed and generalizes it, using a genetic process, to find a rule maximizing a noise tolerant rule evaluation criterion. The SIA approach to supervised rule learning reduces greatly the possible rule search space when compared to the genetic Michigan and Pitt approaches. SIA is comparable to AQ and decision trees algorithms on two learning tasks. Furthermore, it has been designed for a data analysis task in a large and complex justice domain.

1 Introduction

Learning rules in propositional logic from a set of preclassified examples described in an attribute/value based language is a problem well known and studied in Machine Learning (Gams and Lavrac 1987). One reason for that is the fact that in many domains, events or experiences can be easily described using a set of variables or attributes.

Among the many existing algorithms that solve this problem, one can point out, on one hand, some methods that use heuristics to search the rule space. For instance, the ID3-based algorithms learn decision trees involving attributes which are relevant from an information theory point of view (Quinlan 1986). Rules can then be extracted from decision trees, a process which usually increases the system classification accuracy (Quinlan 1987). Another example is the AQ algorithms, which learn rules using the heuristic star algorithm (Michalski et al 1986) (Wnek and Michalski 1991).

On the other hand, some methods use stochastic algorithms (Kononenko and Kovacic 1992) or genetic algorithms (Holland 1975) to find optimal rules.

Examples of such algorithms are the classifier systems, which usually learn rules (classifiers) from examples that are not preclassified (Goldberg 1989) (Wilson 1987) (Venturini 1992), but which can also learn from preclassified examples (McCallum and Spackman 1990) (Bonelli and Parodi 1991). Other algorithms, for instance, are the GABIL system (De Jong and Spears 1991) which learns rules incrementally and the SAMUEL system (Grefenstette 1989)

Finally, some algorithms use a multistrategic search, combining heuristic and probabilistic algorithms. For instance, genetic algorithms, denoted GAs in the following, can discover important attributes in cooperation with an AQ based algorithm (Vafaie and DeJong 1991), or, GAs can improve and refine rules learned by an AQ algorithm, using a subpart of the set of examples (Bala, DeJong and Pachowicz 1991).

The genetic based rule searching algorithms mentioned above have a longer execution time than the heuristics methods. They also have fewer learning abilities: for instance, they do not handle easily numeric attributes because the classifier system rule description language is too simple, or because the GABIL method of encoding all possible values would lead to very long rules in the case of real-valued attributes. Furthermore, it has been recently shown (Wnek and Michalski 1991) that a genetic rule learning system, namely the classifier system CFS of Riolo, obtains the lowest performances among other learning methods, on a simple, noise free learning task.

Thus, one motivation for this work is the building of a GA based learning method that would be globally equivalent, with respect to performances and learning abilities, to the heuristic based methods mentioned above. Furthermore, this work is also motivated by a real world data analysis task in a complex domain which would not be easily handled by the methods mentioned above. The main reasons for this are that attributes are numeric or symbolic, examples are described using a variable number of attributes (mainly because some attributes may be undefined for some examples), and may belong to several classes.

In the following, section 2 describes SIA example and rule representations. Section 3 describes the main learning algorithm, the rule filtering algorithm, the classification procedure chosen and their main properties. Section 4 shows two evaluations of SIA on common learning tasks and section 5 describes the real task for which SIA has been designed. Section 6 concludes on this and future work.

2 Example and Rule Representation

2.1 Examples with Undefined Attributes, Multiple Classes and Weights

SIA learns production rules from a set Ex of examples of events in a given domain. An example ex is described using n attributes $A_1, ..., A_n$, which can be either numeric (real-valued for instance) or symbolic (with discrete values).

Firstly, it is considered that k attributes are defined for all examples and that the remaining $n - k$ attributes may be *undefined* for some examples. For instance, suppose that cars are being described using the attributes *number* –

of − accidents and *date − of − last − accident*. The attribute *date − of − last − accident* is undefined for cars that were never crashed. This notion of undefined attributes includes not only logical cases of undefined values (as in the car example), but also the noisy cases of missing or unknown attributes values, and systematic missing values (Weinberg, Biswas and Koller 1992).

Secondly, an example ex may belong to several classes among a set $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_k\}$ of possible classes. For instance, the description of an animal can belong to the class "*dog*" and to the class "*mammal*". SIA learns a separated definition of each class.

Finally, examples can be weighted in order to create artificial examples distributions in Ex. For example, let us suppose that the learning task is to learn rules about cars. These rules should conclude that a car is *safe* or *unsafe*. Let us suppose that $\frac{9}{10}$ of the cars are safe. In order to learn reliable rules, many car examples must be recorded. Thus, a hundred of safe cars and a hundred of unsafe cars examples are recorded in Ex. The probability, in Ex, of a car to be safe is now $\frac{1}{2}$, instead of $\frac{9}{10}$ in the real domain. To recreate the examples original distribution in Ex, a weight $w = 9$ should be assigned to safe cars examples, and a weight $w = 1$ to unsafe cars examples. These weights introduce biases in the learning process by making some examples more important than others.

Thus, an example ex is represented as

$$(e, \mathcal{C}_L, w)$$

where $e = \{e_1, ..., e_n\}$ is a vector of attributes values, among which some values may be undefined, $\mathcal{C}_L$ is the list of classes ex belongs to, and w is the example weight.

2.2 Attributes based Rule Representation

SIA learns a representation of each class in $\{\mathcal{C}_1, ..., \mathcal{C}_k\}$. For a class $\mathcal{C}_i$, the representation learned is a set of rules R of the form

$$R : IF \underbrace{cond_1 \wedge ... \wedge cond_n}_{Condition\ part} THEN \underbrace{Class = \mathcal{C}_i}_{Conclusion\ part}, \underbrace{Strength}_{Strength\ part}$$

In R condition part, $cond_i$ involves attribute A_i and equals either:

- "$\star$", meaning that A_i is not taken into account (the condition is always true), or
- "$A_i = value$", where A_i is a symbolic attribute and *value* is an observed value of A_i, or
- "$B \leq A_i \leq B'$", where A_i is a numeric attribute and where the lower and upper bounds B and B' are such that $B' \geq B$ (B and B' computation is detailed in the following).

An example of such a rule is

$$R1 : IF \star \wedge date - of - last - accident = yesterday \ THEN \ Class = unsafe$$

Using this rule condition format, a matching operator between a rule R and an example ex can be defined: R matches the example $ex = (e, \mathcal{C}_L, w)$ if all R conditions are true for vector e. If ex has an undefined or missing value for an attribute A_i, it can be matched by R only if A_i is not taken into account by R conditions ($cond_i = \star$). For instance, the event "$number - of - accidents = 0 \wedge date - of - last - accident = undefined$" can not be matched by $R1$.

The conclusion part concludes that an example $ex = (e, \mathcal{C}_L, w)$, matched by R, belongs to class $\mathcal{C}_i$. If ex really belongs to class $\mathcal{C}_i$ (i.e. $\mathcal{C}_i \in \mathcal{C}_L$), then ex is said to be correctly classified by R, else ex is misclassified.

The strength part of R is a set of coefficients that are used to measure the quality of R by computing a quality criterion $C_q(R)$ (see section 3). The higher $C_q(R)$ is, the more interesting R is. The value of this criterion represents the genetic strength of the rule that the genetic search will try to maximize.

This rule description language is thus slightly less powerful than the one use in AQ, because, for instance, a disjunction of conditions for the same class, is coded by several rules rather than by just one rule.

Possible Bounds for Numeric Conditions. When a numeric attribute A_i is involved in the condition $cond_i$ of the condition part of a rule, one must define what possible values the bounds B and B' described above can take. Let us suppose that A_i has m distinct ordered values $v_1, ..., v_m$ observed in Ex. One solution, similar to one used in AQ, is to define the possible set $\mathcal{B}_i$ of bounds for $cond_i$ as $\mathcal{B}_i = \{v_1, ..., v_m\}$.

Another solution, which has been used in the following, is to define $\mathcal{B}_i$ as

$$\mathcal{B}_i = \{-\infty, \frac{v_1 + v_2}{2}, ..., \frac{v_{m-1} + v_m}{2}, +\infty\}$$

which is similar to one approach used for finding thresholds in decision trees algorithms, but here, no statistical techniques are used: the learning process will select itself the proper bounds in $\mathcal{B}_i$.

3 SIA Main Algorithms

3.1 Learning Algorithm Overview

The SIA basic learning algorithm is somewhat similar to the AQ algorithm because it uses a seed example ex as a start point, and tries to find the most optimal rule that covers this example using generalization. One important difference between the two methods is that SIA uses a genetic based search:

1. Let $\mathcal{R}$ be an empty set of rules,
2. Label "uncovered" all classes in the class lists of all examples in Ex,

3. Let $ex = (e, C_L, w)$ be an example of Ex such that there exists a class $C_i \in C_L$ labelled "uncovered".

4. Let R_{init} be the most specific rule that matches ex and concludes "$Class = C_i$",

5. Using a GA, generalize the condition part of R_{init} to find the optimal rule(s) R^* that match(es) ex (rules that maximize the rule evaluation criterion $C_q(R)$),

6. Label "covered" all classes C_i in the class list of examples matched by R^* rule(s),

7. Add R^* to $\mathcal{R}$,

8. If some examples remain such that a class in their class list is labelled "uncovered", then go to 3,

9. Possibly, eliminate rules in $\mathcal{R}$ using the rule filtering algorithm,

10. Ouput $\mathcal{R}$

In step 4, the R_{init} rule is computed as follows: the symbolic conditions of R_{init} are of the form "$A_i = e_i$". The numeric conditions are of the form "$B \leq A_i \leq B'$" where B and B' are the closest lower and upper bounds to e_i in $\mathcal{B}_i$. However, if the value of A_i is undefined for ex, the corresponding condition $cond_i$ in R_{init} is set to "$\star$": the algorithm must learn a rule that classifies ex without using the missing attribute A_i.

The behavior of the algorithm is illustrated in a simple case on figure 1: two attributes A_1 and A_2 define an example space where the examples can belong to the classes "+" or "-".

This algorithm ensures the completeness of $\mathcal{R}$ over Ex, if the rule filtering step 9 is ommitted. No mechanism is used to choose the seed example in step 3: SIA is sensitive to the order of the examples, unless every example is selected as a seed in steps 3 and 8.

3.2 Genetic based Rule Discovery Process: SIA approach

The genetic search process of step 5 in the SIA main algorithm tries to find rules that maximize $C_q(R)$ by generalizing the condition part of the starting rule R_{init}. According to the GA principles (Holland 1975), this process uses a population P of rules to perform a probabilistic parallel search in the rule space. The search process generates rules using genetic operators, which here are based on generalization. The population P is initially empty and has a maximum size of 50 rules. The search process is the following one:

1. Let $P = \emptyset$,

2. Generate one or two rules by choosing an operator to apply among:

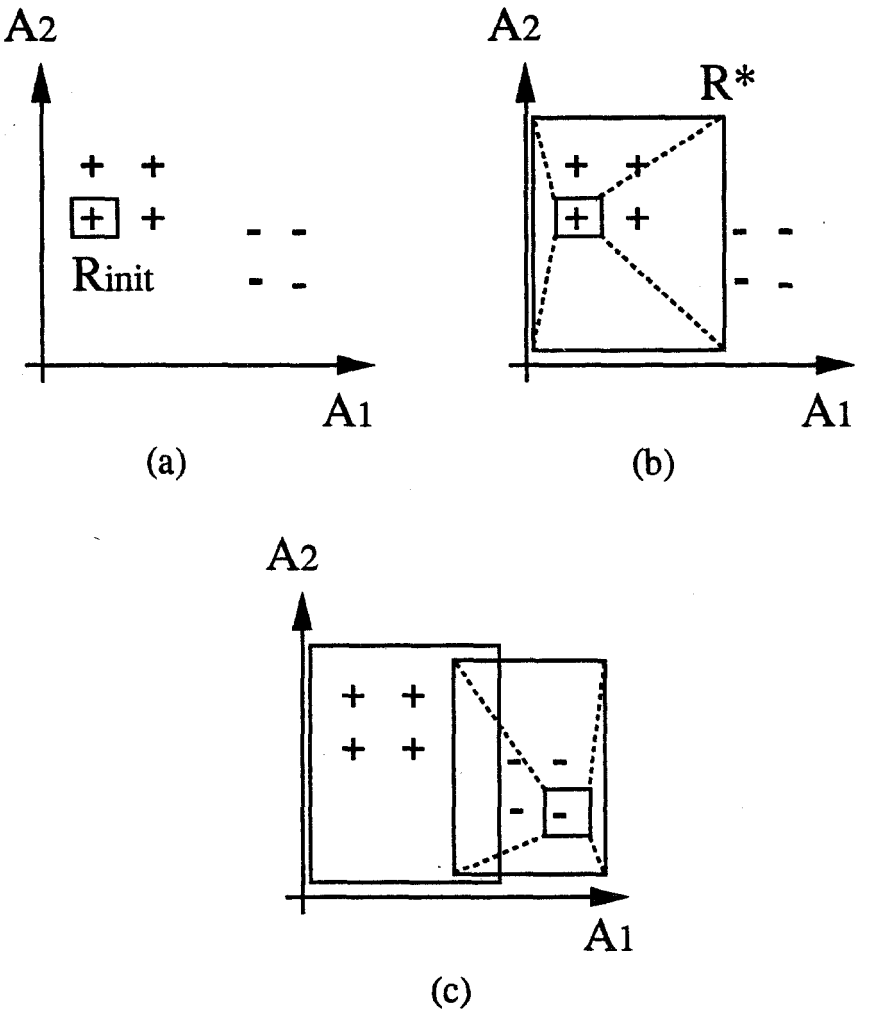

Figure 1: (a) SIA chooses an initial example of class "+", and (b) generalizes R_{init} to find R^*. Then (c), it starts on again with another uncovered example of class "-".

(a) creation (probability of 10 %): generalize R_{init} into an offspring rule R'. Evaluate $C_q(R')$. Apply the insertion operator to R'.

(b) generalization (80 %): select randomly one rule R in P and generalize its condition part to generate R'. Evaluate $C_q(R')$. If $C_q(R') > C_q(R)$ then R' replaces R in P, else apply the insertion operator to R'.

(c) crossover (10 %): select randomly two rules R_1 and R_2 in P. Apply a uniform crossover operator to obtain two offsprings R'_1 and R'_2. Evaluate $C_q(R'_1)$ and $C_q(R'_2)$, and apply to them the insertion operator.

3. Termination criterion: if step 2 has been repeated for more than Nb_{max} times without generating a better rule R' than the best rule in $P \cup R_{init}$, then Stop and output the rule(s) R^* of $P \cup R_{init}$ that maximize(s) $C_q(R)$, else go to 2.

where Nb_{max} is given by the domain expert or user.

The creation operator (point 2a) introduces new start points in the search space and is applied with a probability greater than 10% at the beginning of the search.

The generalization operator (point 2b and partially in 2a) generalizes randomly some conditions of a selected rule R. For instance, a condition "$A_i = value$", where A_i is a symbolic attribute, is generalized to "$\star$". A condition "$B_k \leq A_i \leq B_l$", $k < l$, where A_i is a numerical attribute is generalized to "$B_{k-k'} \leq A_i \leq B_{l+l'}$" where $B_{k-k'}, B_{l+l'} \in \mathcal{B}_i$, or can also be generalized to "$\star$". R' may replace R in order to avoid following too many times the same path in the search space.

The uniform crossover operator (point 2c) exchanges conditions between two selected rule R_1 and R_2, with a probability $p_c = 0.5$, which generates two offsprings. The aim of the crossover is to exchange building blocks between rules.

The insertion operator is used to insert an offspring rule R' in P: if $R' \in P$ then R' is not inserted. If $|P| < 50$ then R' is added to P, else, R' replaces the lowest strength rule R_{low} in P if $C_q(R') > C_q(R_{low})$.

This optimization process stops when no better rules where generated during the last Nb_{max} rule generations. It may find multiple (and different) optimal rules because SIA has no way to choose between several optimal rules (unless the expert gives a more precise criterion). The search may be intensive if Nb_{max} is high. Generated rules always match the seed example ex and have the same conclusion part as R_{init}.

Rule Evaluation Criterion C_q. Each rule R is assigned a quality or strength value $C_q(R)$ which evaluates R quality in the following way:

$$\begin{cases} C_q(R) = \frac{c - \alpha nc + \beta g}{csize} \\ C_q(R) \geq 0 \end{cases}$$

where $\alpha \geq 0$, $\beta = 0\, or\, -0.001\, or\, +0.001$ and where

- c is the total weight of the examples that R classifies correctly,

- nc is the total weight of the examples that R misclassifies,
- g is an abstract measure of R generality, which takes values between 0 and 1: 0 means that R is very specific, 1 that R is very general. Intermediate values of g are computed by measuring the proportion of attributes not taken into account in R condition part,
- $csize$ is the total weight of the examples in Ex that belong to class C_i (the concept total weight or size).

The strength of a rule is high if this rule classifies correctly many examples and misclassifies as few examples as possible.

This evaluation criterion has several interesting properties:

1. it ensures the expert that learned rules accuracies ($\frac{c}{c+nc}$), if there are no missing values introducing irreductible errors, is above $\frac{\alpha}{1+\alpha}$. A short proof of this is the following one: the rule R_{init} has a strength above 0 because it classifies (correctly) one example only. Thus, succeeding optimal rules will have a strength above $C_q(R_{init})$, and also greater than 0. This implies, as β is chosen small enough so that βg is negligible compared to $c - \alpha\, nc$, that these rules will verify $c - \alpha\, nc \geq 0$, which can be rewritten as follows:

$$Accuracy(R) = \frac{c}{c+nc} \geq \frac{\alpha}{1+\alpha}$$

   The domain expert can thus ask SIA for consistent rules ($\alpha > |Ex|$), or relax this constraint by asking for rules with, for instance, a minimum of 98% accuracy (with $\alpha = 50$). To deal efficiently with noise and find a good value for α, the expert should have a rough idea of the noise percentage in its data.

2. it can guide the search process either towards specific or general rules expressions with $\beta = -0.001$ or $\beta = +0.001$ respectively (see figure 2). If the expert wants to favor the generality of the learned rules instead of their consistency, β can be increased, but the property described above may not hold any more.

3. it makes a difference between noise and concept boundaries : in the situation (a) of figure 3, the "-" example is considered like noise and SIA, with $\alpha = 1$ for instance, learns R_1 because $C_q(R_1) > C_q(R_2)$. In situation (b), the "-" example is not considered like noisy but like belonging to the concept boundary, and SIA learns R_1 and not R_2.

This criteria can also be customized.

SIA Versus Michigan and Pitt Approaches. Two approaches to genetic based rule learning exist, known as the Michigan and Pitt approaches. In the Pitt approach (Grefenstette 1989) (Janikow 1992), a genetic entity of the population is a rule set of N rules, which strength is a measure of the N rules performance.

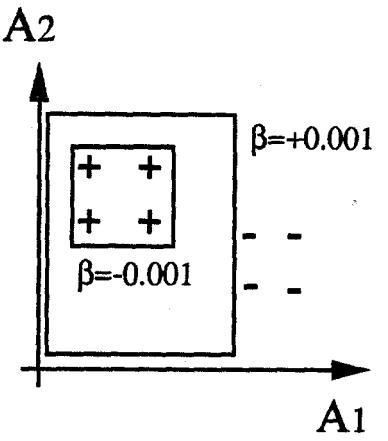

Figure 2: SIA can learn rules with most specific ($\beta = -0.001$) or most general ($\beta = +0.001$) expression.

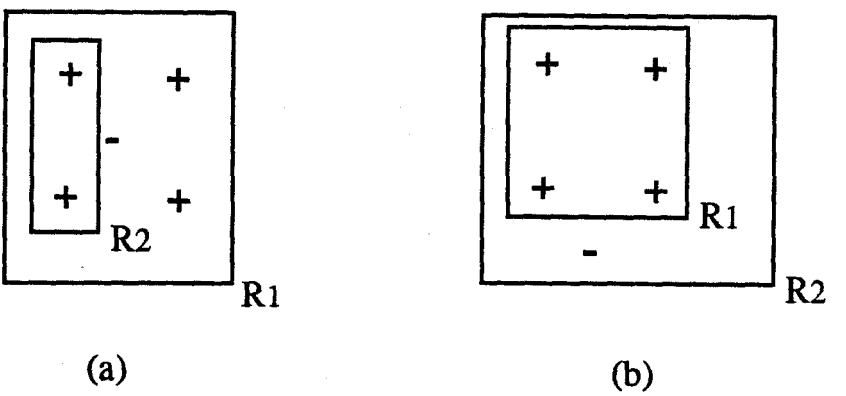

Figure 3: The rule evaluation criterion $C_q(R)$ makes a difference between noise (a) (see the position of the "-" example) and concept boundary (b). SIA learns R_1 and not R_2 (for $\alpha = 1$) in both cases.

Genetic approaches	Search space size
Pitt	$N^{k(val+1)^n}$
Michigan	$k(val+1)^n$
SIA	2^n

Table 1: Search space size for different genetic learning approaches, where N is the number of rules per entity in the Pitt approach, k the number of classes, n the number of attributes and val the number of values per attributes (for symbolic attributes and one class per example).

Thus, this approach performs a global optimization of a set of N rules, and learns a set of well co-adapted rules. However, the genetic search space is very large and the value of N must be known in advance.

In the Michigan approach, a genetic entity is one rule, and the GA searches for a subpopulation of efficient rules (Wilson 1987). The genetic search space is thus reduced.

In the SIA approach, a genetic entity is a rule, but the GA searches for one rule only among the possible generalizations of an example. Thus, in the case of supervised learning, the most positive aspect of the SIA approach, compared to the Michigan and Pitt approaches, is that it reduces drastically the genetic search space (see table 1), even if several searches must be performed if several rules are to be learned.

3.3 Rule Filtering Algorithm

This algorithm is a kind of rules postprocessing method which eliminates fastly some redundant rules in a set of rules $\mathcal{R}$. For every rule $R \in \mathcal{R}$, it computes R internal strength (Venturini 1992), denoted $strength_I(R)$, which measures how useful R is with respect to the other rules of $\mathcal{R}$. Then, rules with an internal strength below a given threshold T_{str} can be deleted:

1. Let $strength_I(R) = 0$ for all rules $R \in \mathcal{R}$,
2. For every example $ex = (e, C_L, w)$ of Ex do
   (a) Let M be the set of rules that matches ex and R^* the subset of rules in M that have the highest strength $C_q(R^*)$
   (b) Let $strength_I(R) = strength_I(R) + w$ for rules R of R^*.
3. Let $strength_I(R) = \frac{strength_I(R)}{c+nc}$ for all rules R, ($c + nc$ is the total weight of the examples that R matches)
4. Remove every rule R from $\mathcal{R}$ such that $strength_I(R) \leq T_{str}$, where T_{str} is given by the expert.

If $T_{str} = 0$, $\mathcal{R}$ completeness over Ex is kept. If $T_{str} > 0$, this completeness constraint may be relaxed.

One interesting property of this algorithm is that its complexity is linear with the number of rules and examples. The experimental results of section 4 show that it reduces significantly the number of rules, and most of the time, increases the rules classification accuracy on unseen cases.

Several other methods could be used as well, like the rule elimination algorithm Quinlan uses when extracting rules from decision trees (Quinlan 1987).

3.4 Classification Procedure

The aim of the classification procedure is to decide, with a set of rules $\mathcal{R}$, to which classes a new unseen example $ex = (e, ?, w)$ belongs to.

Firstly, a rule-example distance $d(R, ex)$, similar to the one used in (Salzberg 1991), is defined in the following way

$$d(R, ex) = \frac{1}{n(R)} \sqrt{\sum_{i=1}^{n} d_i^2}$$

where:

- for a symbolic attribute A_i, $d_i = 0$ if the condition $cond_i$ of R is true for e, else $d_i = 1$,
- for a numeric attribute A_i, $d_i = 0$ if the condition $cond_i$ of R is true else:
  - if $e_i > B'$ then $d_i = \frac{e_i - B'}{max_i - min_i}$
  - else if $e_i < B$ then $d_i = \frac{B - e_i}{max_i - min_i}$

  where $cond_i = "B \leq A_i \leq B'"$, min_i and max_i are the minimum and maximum values of A_i in Ex (if $d_i > 1$ then $d_i = 1$)
- $n(R)$ is the number of conditions in R which are different than "$\star$"

If $d(R, ex) = 0$ then R matches ex (as explained in section 2.2.1), else $d(R, ex)$ computes a partial match score between R and ex.

The decision procedure computes this distance for every rules of $\mathcal{R}$. Let d_{min} be the minimal distance measured. Let R^* be the set of rules such that $d(R^*, ex) = d_{min}$ and which have the highest value of $C_q(R)$: ex belongs to the classes on which R^* rules conclude. This measure separates the example space with boundaries made of straight lines and parabols (see Salzberg work). Here, several classes can be given to an example, like "dog" and "mammal" for instance.

3.5 Complexity

Giving an interesting and useful bound for the time complexity of SIA genetic search process is difficult. The worst case analysis (for symbolic attributes and example belonging to several classes) supposes that the search process, starting from the rule R_{init}, generates the 2^n possible rules. Further more, it supposes that better rules appear only at the cycle before the deadline of Nb_{max} cycles, which give a maximum of $2^n Nb_{max}$ rule evaluations. It then supposes that a new search process starts for every example of Ex and for every classes it belongs to, which gives a worst case complexity of $2^n Nb_{max}|Ex|k$ rule evaluations, or $2^n Nb_{max}|Ex|^2 nk$ tests of the form "$A_i = value$" (where k is the number of classes, n the number of attributes).

However, for learning the $F20$ problem described in the next section with 800 examples, the worst number of rule evaluations would be 10^{12}, and in reality, SIA evaluates $1.3\,10^5$ rules which is about $7.7\,10^6$ times less.

4 Evaluations

Two evaluations have been performed with $\alpha > |Ex|$, $T_{str} = 0$, $\beta = +0.001$ (consistent, complete and most general rules), and $Nb_{max} = 600$ cycles.

4.1 Robots Domain

This learning task comes from (Wnek and Michalski 1991). It consists of learning independently five different concepts (robot descriptions) from 6 attributes taking less than 4 values each. The number of all possible robot descriptions is 432 and the concepts to be learned are described in a logic way: for instance, concept $C1$ is "head is round and jacket is red or head is square and is holding a balloon", where "head", "jacket" and "holding" are attributes. Thus, the learning task is easier for systems that learn rules described in the same language as the concept description language like AQ15, AQ17-HCI or SIA, than for systems like CFS (a classifier system), neural networks or decision trees (C4.5), which use a different representation. The experiment starts with a training set containing 6% of the whole set of positive examples and 3% of the whole set of negative examples of concept C1, and goes up to (100%,10%). This process is repeated from concept C2 to C5. SIA learns rules for the positive class only, which is an ability common to AQ15 and AQ17-HCI. The evalution procedure evaluates the learned rules with an exact error rate on the whole set of possible descriptions. Results obtained without the rule filtering algorithm are given in table 2.

SIA performances, which were averaged over five runs, are comparable to or even higher than AQ17-HCI performances for this learning task.

4.2 $F20$ Multipexor Learning Task

This task is a boolean function learning task. The first n bits of a boolean input vector are used to select one of the 2^n remaining bits of the vector. If the selected

	percentage of pos. and neg. training examples				
	(6%,3%)	(10%,10%)	(15%,10%)	(25%,10%)	(100%,10%)
CFS	21.3%	20.3%	22.5%	19.7 %	16.3%
NNets	9.7%	6.3 %	4.7 %	7.8%	4.8 %
C4.5	9.7%	8.3 %	1.3 %	2.5%	1.6 %
AQ15	22.8%	5.0 %	4.8 %	1.2%	0.0%
AQ17-HCI	4.8 %	1.2 %	0.0%	0.0%	0.0%
SIA	7.8 %	0.4 %	0.0%	0.0%	0.0%

Table 2: Robots domain: average error rate for different learning methods from (Wnek and Michalski 1991) and SIA error rate

bit equals 0 then, the vector class is 0, else it is 1. The problem studied here is $F20$ where 4 bits select one of the 16 remaining bits, for a total of 20 boolean attributes. The learning and evaluation methodologies used are the same as those used by Quinlan for C4.5 (Quinlan 1988): a training set of examples is randomly generated with a test set of 1000 unseen cases. The evaluation of the learned rules is performed with these unseen cases and all runs are repeated five times. The averaged results are reported on table 3.

Above 400 examples, SIA outperforms the decision tree algorithm, but not the rule extraction procedure associated to it. SIA learns much more rules than C4.5, firstly because, unlike C4.5, SIA learns classes 0 and 1 (which doubles at least the number of rules), and secondly because the multiplexor learning task has the characteristic that several rules with equal generality can represent the same portion of the examples space. As mentioned before, SIA discovers all possible (among the most general and consistent) concept descriptions, and the postprocessing algorithm does not eliminate all redundancy in rules, even if it may improve globally their performances. For 800 examples, SIA learned rules are comparable to the rules extracted from C4.5 decision trees. When searching for one rule (for 800 examples), SIA explores about 0.13 % of the possible 2^{20} generalizations.

However, the execution time is one hour on a Sun Sparc Elc (for 800 examples), which is certainly much longer than what C4.5 and the rule extraction procedure would need.

5 Application

SIA has been initially designed for a data analysis task in the french justice domain. A domain expert, Bruno Aubusson de Cavarlay[1], has described 1250 french justice files using 100 attributes. These files have a variable length (some

[1] Bruno Aubusson de Cavarlay works at the Centre de Recherches Sociologiques sur le Droit et les Institutions Pénales (CESDIP), URA 313 du CNRS, Ministère de la Justice, 4 rue de Mondonvi, 75001 Paris FRANCE

	C4.5			
	D. Trees		Rules	
# ex.	# nodes	% accu.	# rules	% accu.
200	49.0	68.8 %	8.6	69.2 %
400	95.8	82.1 %	14.4	88.0 %
600	121.0	87.4 %	16.2	97.4 %
800	171.4	92.4 %	18.4	98.3 %

	SIA				
		before filt.		After filt.	
# ex.	# eval.	# rules	% accu.	# rules	% accu.
200	90000	62	60.7 %	53.6	60.4 %
400	131000	93.8	78.8 %	78.4	79.2 %
600	160000	95.2	90.8 %	73.8	91 %
800	130000	89.4	97.1 %	66.2	98.3 %

Table 3: Results for $F20$: "# ex." is the number of training examples, "% accu." is the learned rule accuracy (percentage of correctly classified unseen examples), "# eval" is the mean number of rules SIA has evaluated

attributes are undefined), because, for instance, no culprit may be found for a given file, and thus the attributes about culprits are undefined for these files. The aim of the expert is, for instance, to analyse how the french law is applied in reality (Aubusson de Cavarlay 1987a) (Aubusson de Cavarlay 1987b). Each attribute has roughly 20 values. The rule space is thus very large (at least 21^{100}). Examples are weighted to recreate the real domain probabilites. This task is currently under study and several problems appear, like for instance: some concepts are described with a very small number of examples (2 or 3 for instance) in a huge description space, or, some attributes values are redundant (they code differently the same information), which leads SIA to find rules that underline these dependancies instead of some more interesting ones.

6 Conclusion

This study has tried to show that genetic algorithms can be useful tools for supervised inductive learning. The resulting algorithm, SIA, reduces the rule search space by searching rules one at a time. It can learned rules from examples that may be described with a variable number of attributes and that may have multiple classes. SIA learning abilities are comparable to those of other heuristic based algorithms, as well as the experimental results obtained on two learning tasks. However, SIA learning times are still important compared to the decision trees or AQ based methods. SIA is currently applied to the analysis of the complex french justice domain.

The work presented here is a beginning. Many theoretical points must be

studied such as incrementality, introduction of more background knowledge and learnability results. Also, further evaluations and comparisons are needed in order to evaluate all properties of SIA, like dealing with noisy data and missing values.

Acknowledgements

I would like to thank Janusz Wnek for providing robots domain data and Bruno Aubusson de Cavarlay, the french justice expert. I would also like to thank Yves Kodratoff and the Inference and Learning group for providing useful comments on this work.

References

Aubusson de Cavarlay B. (1987a), La diversité du traitement pénal, *Données sociales 19*, 589-593.

Aubusson de Cavarlay B. (1987b), *Les filières pénales*, CESDIP, Déviance et Contrôle Social 43.

Bala J., De Jong K.A. and Pachowicz P. (1991), Learning noise tolerant classification procedures by integrating inductive learning and genetic algorithms, Proceedings of the First International Workshop on Multistrategy Learning 1991, R.S. Michalski and G. Tecuci (Eds), 316-323.

Bonelli P. and Parodi A. (1991), An efficient classifier system and its experimental comparison with two representative learning methods on three medical domains, Proceedings of the Fourth International Conference on Genetic Algorithms, R.K. Belew and L.B. Booker (Eds), 288-295, Morgan Kaufmann.

De Jong K. (1988). Learning with Genetic Algorithms: An overview. Machine Learning 3, 121-138: Kluwer Academic.

De Jong K. and Spears W.M. (1991), Learning concept classification rules using genetic algorithms, Proceedings of the 12^{th} International Joint Conference on Artificial Intelligence 1991, J. Mylopoulos and R. Reiter (Eds), 651-656, Morgan Kaufmann.

Gams M. and Lavrac N. (1987), Review of five empirical learning systems within a proposed schemata, Progress in Machine Learning, I. Bratko and N. Lavrac (Eds), 46-66, Sigma Press.

Goldberg D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*: Addison Wesley.

Grefenstette J.J. (1989), A system for learning control strategies with genetic algorithms. In Proceedings of the third International Conference on Genetic Algorithms, J.D. Schaffer (Ed), 183-190, Morgan Kaufmann.

Holland J.H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

Janikow C.Z. (1992), Combining competition and cooperation in supervised inductive learning, Proceedings of the Ninth International Workshop on Machine Learning 1992, D. Sleeman and P. Edwards (Eds), 241-248 , Morgan Kaufmann.

Kononenko I. and Kovacic M. (1992), Learning as optimization: stochastic generation of multiple knowledge, Proceedings of the Ninth International Workshop on Machine Learning 1992, D. Sleeman and P. Edwards (Eds), 257-262 , Morgan Kaufmann.

McCallum J.H. and Spackman K.A. (1990), Using genetic algorithms to learn disjunctive rules from examples, Proceedings of the Seventh International Conference on Machine Learning 1990, B.W. Porter and R.J. Mooney (Eds.), 153-159, Morgan Kaufmann.

Michalski R.S., Mozetic I., Hong J. and Lavrac N. (1986), The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, Proceedings of AAAI-86 Fifth National Conference on Artificial Intelligence, 1041-1045, Morgan Kaufmann.

Quinlan J.R. (1986), Induction of decision trees, Machine Learning 1,1.

Quinlan J.R. (1987), Generating production rules from decision trees, Proceedings of the Tenth International Joint Conference on Artificial Intelligence 1987, J. McDermott (Ed), 304-307, Morgan Kaufmann.

Quinlan J.R. (1988), An empirical comparison of genetic and decision trees classifiers, Proceedings of the Fifth International Conference on Machine Learning 1988, J. Laird (Eds), 135-141, Morgan Kaufmann.

Salzberg S. (1991), A nearest hyperrectangle learning method, Machine Learning 6, 251-276.

Vafaie H. and De Jong K. (1991), Improving the performance of a rule induction system using genetic algorithms, Proceedings of the First International Workshop on Multistrategy Learning 1991, R.S. Michalski and G. Tecuci (Eds), 305-315.

Venturini G. (1992), AGIL: solving the exploration versus exploitation dilemma in a simple classifier system applied to simulated robotics, Proceedings of the Ninth International Workshop on Machine Learning 1992, D. Sleeman and P. Edwards (Eds), 458-463 , Morgan Kaufmann.

Weinberg J.B., Biswas G. and Koller G.R. (1992), Conceptual clustering with systematic missing values, Proceedings of the Ninth International Workshop on Machine Learning 1992, D. Sleeman and P. Edwards (Eds), 464-469, Morgan Kaufmann.

Wilson S.W. (1987), Quasi-Darwinian Learning in a Classifier System, Proceeding of the Fourth International Workshop on Machine Learning 1987, P. Langley (Ed), 59-65, Morgan Kaufmann.

Wnek J. and Michalski R.S. (1991), An experimental comparison of symbolic and subsymbolic learning paradigms: phase I - learning logic-style concepts, Proceedings of the First International Workshop on Multistrategy Learning 1991, R.S. Michalski and G. Tecuci (Eds), 324-339.