Learning rules using genetic algorithms: application to a knowledge base in an expert system generator

RAHARIMANANA Tsilavina Antonio, ROBINSON Matio, ANDRIAMANOHISOA Hery ZO

student, Cognitive sciences and applications, University of Antananarivo in STII, Madagascar
Doctor, Cognitive sciences and applications, University of Antananarivo in ESPA, Madagascar
Professor, Cognitive sciences and applications, University of Antananarivo in ESPA, Madagascar

ABSTRACT

Nowadays, progress in machine learning is phenomenal, thanks to the computing power of current computers. The application of this field is very large, ranging from mobile phone to industrial process control. In this article we used genetic algorithm for learning rules in order to incorporate into the knowledge base of an expert system. Our method is a supervised machine learning that uses VL_2 language proposed by [1] and [2]. In order to reduce the length of chromosome, we transformed the bit string into integer according to [3] method which is detailed below. an expert system is software composed of a rule base, a fact base, an inference engine and a user interface. in order to automatically learn rules based on collected data we will incorporate a learning module into our expert system.

Keyword: - *genetic algorithms, expert system, Knowledge based, learning rules*

---

1. Introduction

Several method of supervised machine learning with genetic algorithm have been proposed. [5] uses an adaptative search technique based on a genetic algorithm which learns to classify rules which it calls GABIL. [6] uses another method called SIAO1 to learn rules in first order logic. [3] proposes a new method for encoding knowledge based on genetic algorithms in order to find decisions rules in a supervised learning context with genetic operators. [7] uses another form of representation for genetic algorithms. He proposes the PGA algorithm in order to learn a predicate and whose form of the predicate is: (P, x, y) where x and y are the arguments and P the predicate. [8] uses genetic algorithm to classify fuzzy rules to diagnose heart disease. For this, he proposes the AGAFL algorithm. [9] presents ECL as a learning system for first order logic. An extension of ECL was proposed by [10], MOECL and ECL have the same representation method. [11] proposes a self-learning algorithm that he calls SLGA.

1.1 Proposed method

we propose to use a representation like REGAL or DOGMA but transformed into natural coding with the method of [1]. We use Michigan approach i.e. one chromosome represents one rule like the figure:

If attrib a	And attrib b	…	Then class A

Fig-1: rule representation

Where attrib a, attrib b etc. are the conditions and class A is the conclusion.
For coding an example in binary, we stored the example in a dictionary with the form (key, value). We made an extraction of each value for an attribute.
For example, consider a ruler having five attributes: weight, color, shape, far. this rule can be represented according to the figure below:

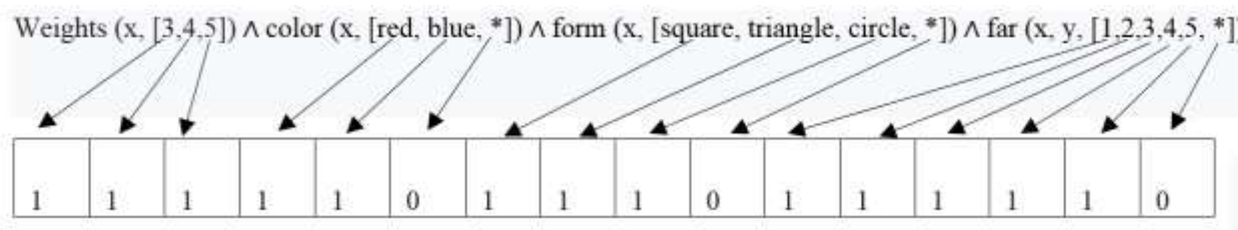

Fig-2: coding rule to bit string

Where x is the value for attribute weights, color, form and far

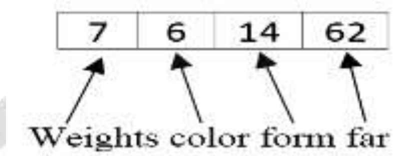

Fig-3: conversion bit string into natural coding

To transform the binary strings into natural coding for each attribute we use the formula:

$$nat(n_2) = \sum_{i=0}^{n-1} 2^i b_i \quad (1)$$

Where n_2 is the representation of the rule in binary coding for an attribute, b_i is the value of the ith bit, either 0 or 1 from right to left.
For the mutation of natural coding we use the formula used by [3]:

$$mut_k(n) = \left(n + 2^{k-1}\right)\%2^k + 2^k \left\lfloor \frac{n}{2^k} \right\rfloor \quad (2)$$

Where % is the remainder of the division and $\lfloor \quad \rfloor$ is the integer part.
For the crossover [3] uses the following equation:

$$recomb(n_i, n_j) = \{z \in Q^t \ tel\ que\ \forall s \geq 0, s < t, Q^t \neq \emptyset, Q^s \neq \emptyset\} \quad (3)$$

With $t \geq 0$ and $Q^t = [mut(n_i)]^t \cap [mut(n_j)]^t$

This calculation reduces the speed of the method, in order to improve the performance instead of using the equation to calculate the mutation of order j we use the “and logical” operator and the “or logical” operator thus we improve the performance of the algorithm.

For example:

for 11 = 01011 the mutation of order 5 are:
10 = 01010
9 = 01001
15 = 01111
3 = 00011
27 = 11011
And for 19 = 10011 the mutation of order 5 are:
18 = 10010
17 = 10001
23 = 10111
27 = 11011

3 = 00011

Recomb (11,19) = {mut (11) ∪ {11}} ∩ {mut (19) ∪ {19}} = {10,9,15,3,27} ∩ {18,17,23,27,3,19} = {3,27}
This result is obtained by doing a logical "and" and a logical "or":

$$\text{recomb}\ (11,19) = \{11 \wedge 19\} \cup \{11 \vee 19\} = \{3,27\} \quad (4)$$

Recomb selection between "or" or "and" is guided by fitness function.

1.2 fitness function

for the performance function, we apply method uses by [4] for evaluating rule. This method uses four variable gives by confusion matrix below:

Table-1: confusion matrix for a class C

	Class		
		C	Not C
	C	True positif	False positif
Prediction class	Not C	False negatif	True negatif

Where TP: number of examples which covert the conditions ($cond_m$) and class C
FP: number of examples which covert the conditions ($cond_m$) but not class C
FN: number of examples don't covert the conditions ($cond_m$) but covert class C
TN: number of examples don't covert the conditions ($cond_m$) nor class C

The confidence factor of a rule is given by:

$$CF = \frac{TP}{(TP + FP)}$$

To calculate that a rule covering attributes and having class c, we use the following equation:

$$comp = \frac{TP}{(TP + FN)}$$

And the fitness function is given by:

$$Fitness = CF * comp$$

1.3 creating next generation

for the creation of the first generation, we take an example and we cover this example. we evaluate each individual then the loop for selection, recombination and the mutation start until the stopping criterion is met. To create the next generation, a selection is made by roulette wheel and then the selected individuals are inserted into the population. The best of this population will insert to the next generation. We carry out crossover and mutation in order to populate next generation.

2. Genetic algorithm and expert system flow chart

in the figure below is the architecture of our expert system. the rule base and the fact base are the knowledge bases, the inference engine works according to modus ponens. The inference engine works in backward chaining and forward chaining.

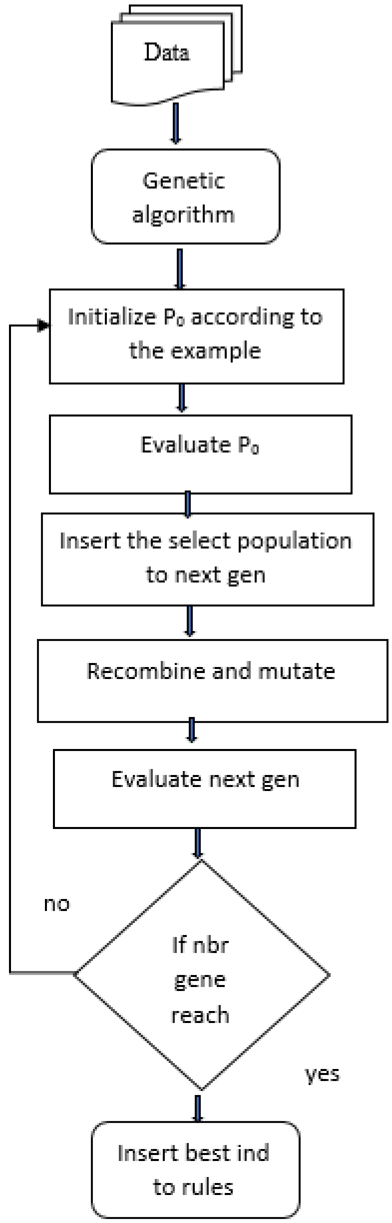

Fig-4: flow-chart of our algorithm

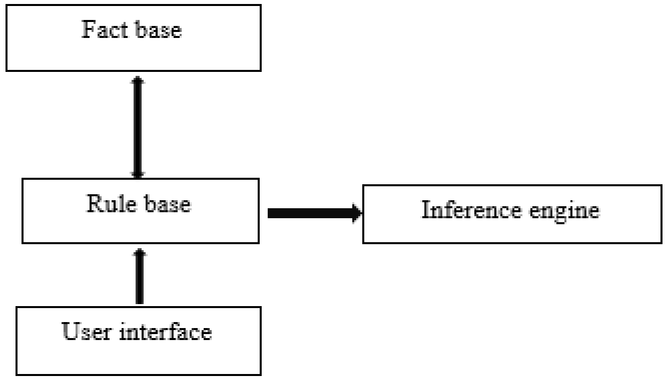

Fig-5: flow-chart of our expert system

3. Algorithm

the algorithm below only stops when there are no more examples to process.

Input: examples
    Number of generations
    Population size
    Mutation probability
    Crossover probability
output: one rule
while there are examples
    Initialize population
    i ⟵ Number of generations
    For i
        Evaluate population with equation (7)
        Parent ⟵ select with roulette wheel
        Next_generation ⟵ good fitness function of antecedent generation
        offspring ⟵ recombine and mutate parent
        Next_generation ⟵ Next_generation ∪ offspring
    End for
Select the best of last generation
Return rule
End while

4. CONCLUSIONS

In order to avoid the calculation of the mutation of order j, we simply have the logical operator "∧" and the logical operator "∨". In this learning method the genes for the conditions attributes participate in recombination and mutation during the evolutionary phase. This method applies for discrete and nominal attributes. In our future work we will look for a method to discretize continuous attributes in order to use the natural coding representation. We will use this method on data provided by UCI repository such as data on mushrooms, car and etc.

5. ACKNOWLEDGEMENT

I would like to thank the cognitive sciences and applications doctoral reception team. I would also like to thank the STII doctoral school.

6. REFERENCES

[1]. A. Giordana et al., learning disjunctive concepts with distributed genetic algorithms, IEEE, 1994
[2]. J. Hekanaho, DOGMA: A GA-Based relational learner, TUCS Technical Report, 1997
[3]. J. S. Aguilar et al., Natural encoding for evolutionary supervised learning, IEEE, 2006
[4]. A. A. Freitas, A review of evolutionary algorithms for data mining, Springer, 2010
[5]. K. A. De Jong et William M. Spears, learning concept classification rules using genetic algorithms,1995
[6]. S. Augier et al., Learning first order logic rules with a genetic algorithm, 2000
[7]. Ya-Wen Chang Chien et Yen-Liang Chen, A phenotypic genetic algorithm for inductive logic programming, Elsevier, 2008
[8]. G. Thippa Reddy et al., Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis, Springer Verlag, 2019
[9]. F. Divina, Evolutionary concept learning in first order logic, IOS Press
[10]. C. Dandois et al, A multi-objective evolutionary concept learner, IEEE, 2010
[11]. Ronghua Chen et al., A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem, Elsevier, 2020