Implementation of a Fuzzy Rule-Based Classifier using Genetic Algorithms

Rashed,David,Joseph

November 18, 2017

1 Introduction

This work is an attempt to implement the classifier presented by Xiong, N., and Lothar Litz. in *Generating Linguistic Fuzzy Rules for Pattern Classification with Genetic Algorithms.* (1999)
The Iris data is a well known classification data with four attributes and 3 classes. The attributes are Sepal Length, Sepal Width, Petal Length and Petal Width.

2 Classification model for fuzzy rules

For each of these attributes $x_i (i = 1 \cdots n)$ there is three fuzzy sets that are denoted A(i,S), A(i,M), A(i,L), respectively for Small, Medium and Large. Let's define **p** as the mapping function from $\{1 \cdots s(s \leq 4)\}$ to $\{1 \cdots 4\}$, with $\forall x \neq y, p(x) \neq p(y)$. We use that function to express that not all parameters are needed in a rule. Thus we can generalise rules as follow :

$$if \left([x_{p(1)} = \bigcup_{j \in D(1)} A(p(1), j)] \bigcap \cdots \bigcap [x_{p(s)} = \bigcup_{j \in D(s)} A(p(s), j)] \right) then \quad \mathbf{B}$$

With $D(i) \subseteq \{Small, Medium, Large\}$
$\bigcup$ corresponds to **or** and $\bigcap$ to **and**.
So for exemple, if:

$$D(1) = \{Small\}, D(2) = \{Medium, Big\}, D(3) = \{\}, D(4) = \{Small, Medium, Big\}$$

Then our rule would read as:

If Sepal Length is Small AND Sepal Width is Medium OR Big AND Petal Width is Small OR Medium OR Big, then C1 or C2 or C3 or C4.

There is 12 different membership functions that can be present or not in a rule. If all the functions are absent from the rule, we have an empty fuzzy set for the condition part, which is an invalid rule.
Thus there is $2^{12} - 1 = 4095$ possible rules. The number of rules is exponentially increasing with the number of parameters and linguistic variables. We could have think to encode the resulting class in our rule, but it would have increase the search space.

Xiong & Litz proposed a way to compute the classes using the antecedant and the data set:

A rule can be summarised as *If A then B, or* $A \Rightarrow B$.
$B \in \{Class_1, Class_2, Class_3, Class_4\}$

$$A = [x_{p(1)} = \bigcup_{j \in D(1)} A(p(1), j)] \bigcap \cdots \bigcap [x_{p(s)} = \bigcup_{j \in D(s)} A(p(s), j)]$$

$\mu_A(u_i)$ is the membership value of an item $u_i (i = 1 \cdots m)$ from the training set $U_t = \{u_1, \cdots, u_m\}$ to A. The consequent B is a crisp subset which defines as follow:

$$\mu_B(u_i) = \begin{cases} 1 & \text{if } class(u_i) = B \\ 0 & \text{otherwise} \end{cases}$$

$A \Rightarrow B$ is equivalent to $A \subseteq B$, so the subsethood of A in B will be used to compute the truth value of a rule for each class:

$$truth(A \Rightarrow B) = \frac{M(A \cap B)}{M(A)} = \frac{\sum\limits_{u_i \in U_T} (\mu_A(u_i) \wedge \mu_B(u_i))}{\sum\limits_{u_i \in U_T} \mu_A(u_i)} = \frac{\sum\limits_{class(u_i)=B} \mu_A(u_i)}{\sum\limits_{u_i \in U_T} \mu_A(u_i)}$$

3 Applying Genetic Algorithm

3.1 Modeling the algorithm

3.1.1 Individual

We define an individual as a list of rules. Rules are represented with a list of 12 bits, 3 bits for each parameters. These 3 bits express the presence or not of each linguistic variable for D(i), 1 if present, 0 otherwise. For exemple the rule presented in section 2 would be represented as [1,0,0,0,1,1,0,0,0,1,1,1]. We decided to set 10 rules for each individual.

3.1.2 Selection and Crossover of individuals

In genetic algorithms, it's important to avoid a **premature convergence**, that is when a population converge toward a local extremum rather than the global optimum.
The Tournament Selection is a way to chose a user-selected number of random individuals in a population, and then take the fittest.
It is possible to give a chance to weaker individuals by reducing the number of participing candidates, or in the opposite to prevents weaker ones to breed by increasing the number. Two selections gives two individuals that can mate with a crossover. The idea is to mix the chromosomes of the individuals to produce a child inheriting the genes of his parents.
Many techniques exist to cross two chromosomes. We chose to pick the uniform crossover.
Given two rules, for each bit there is a 0.5 probability to inherit the gene of one of the parent.
We repeat the process for each pair of rules.

3.1.3 Mutation

Mutation is a key point in genetic algorithm to explore the problem space by maintaining a problem diversity. Similar to biological mutation, this operator modifies a chromosome in a unpredictable way and can give a better solution.
In our algorithm, there is a mutation parameter that will give the chance of each gene to flip the bit. Each bit of each rule of an individual will have a probability p_{mut} to be flipped.
If p_{mut} is too high the information given by the chromosome might be lost because it changed too much, and thus the algorithm may never converge. Else if p_{mut} is too low, the exploration space may be restricted and thus the algorithm may converge to a local optimum.