Fuzzy classifier induction with GA-P algorithms

Luciano Sánchez Ramos
Universidad de Oviedo
Departamento de Informática
luciano@lsi.uniovi.es

Santiago García Carbajal
Universidad de Oviedo
Departamento de Informática
carbajal@lsi.uniovi.es

Summary

In this work we will use GA-P algorithms, a hybrid method between genetic algorithms and genetic programming, to induce fuzzy rule based classifiers with a more general structure than the attainable when applying GA's.

We will show that the numerical results this method gets are comparable to those produced by black-box classifiers.

Keywords: Genetic Programming, Genetic Fuzzy Systems, Fuzzy Classifiers.

1 Genetic Programming and GA-P Algorithms

Genetic programming has been recently regarded as the application of a genetic algorithm to optimize a function defined over the chains of a context-free language [2].

Crossover and mutation operations are specific to this problem. One or two point crossover of chains is not directly applicable, because the offspring would not always consist in valid chains for the language. The same thing can be said about mutation. Both cases are solved by codifying the chains with their parse trees or, as we will do here, by means of their syntax trees, annotated with the names of the production rules. If crossover is defined by the exchange of subtrees whose root is labeled with the same name, descendants are always parse or syntax trees of valid chains. It is remarkable that, under this point of view, GP crossover as defined by Koza [3] is a particular case of two-point GA crossover.

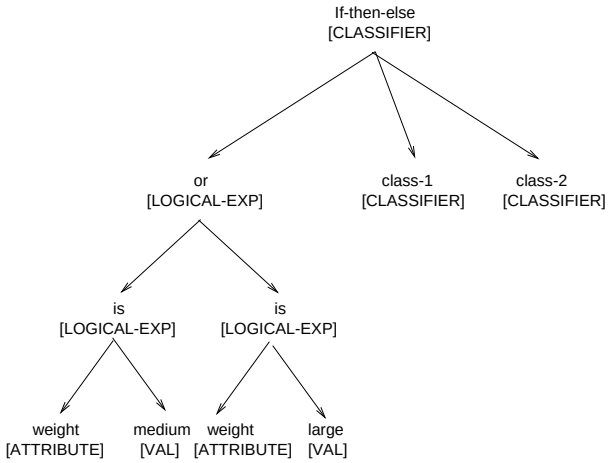

Figure 1: Annotated syntax tree of the chain "If weight is medium or weight is large then class-1 else class-2". The names of the non terminal symbols are shown in brackets.

1.1 Numerical coefficients evolution

If the grammar of the language contains numerical terminals, the values of these terminals are randomly assigned while generating the initial population and they will only evolve by mutation. As a practical consequence, final chains can be too complex (for instance, the number '2' could be represented by an expression like '(2.45+2.45)/2.45'). This is a problem with canonical GP.

Better results can be obtained if numbers are not terminal symbols, so mutation and crossover can operate on them and also when "named constants" are used. These are identifiers that are evaluated to a value and can be in more than one point of the chain.

An individual must contain two parts in order to use named constants: (1) the codification of their values and (2) the program that uses them. For instance, a polynomial that depends on a maximum of two different coefficients can be defined by means of the follow-

ing grammar:

S → C P

C → VALUE-K1 VALUE-K2

VALUE-K1 → D-CHAIN

VALUE-K2 → D-CHAIN

D-CHAIN → DIGIT | D-CHAIN DIGIT

DIGIT → 0|1|2|3|4|5|6|7|8|9

P → x_1 | x_2 | k_1 | k_2 | + P P | * P P

and the chain "123 11 * k_1 + k_1 * x_1 k_2" codifies the expression $k_1 * (k_1 + k_2 x_1) = 123 * (123 + 11\, x_1)$ in this language.

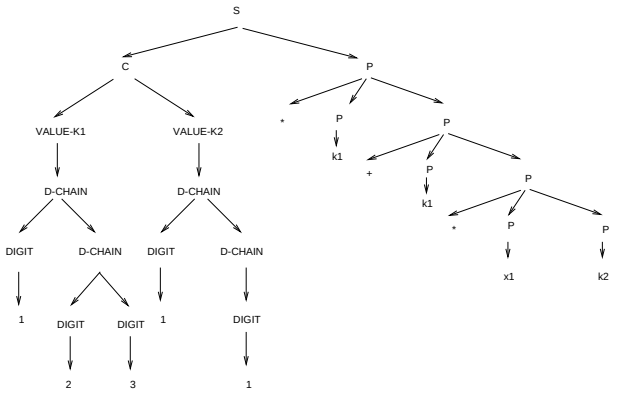

Figure 2: Parse tree of the chain "123 11 * k_1 + k_1 * x_1 k_2"

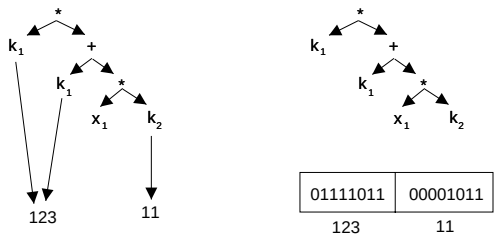

Figure 3: ADG of the chain "123 11 * k_1 + k_1 * x_1 k_2" and its GA-P representation

Representing a chain of the preceding language by means of its parse tree does not pose conceptual problems (see Figure 2) but it is very inefficient in time and space. We will use an acyclical graph best, that is analogous to the syntax tree of the chain (see figure 3).

A genetic algorithm that uses this last representation is named GA-P algorithm. In GA-P algorithms an individual comprises two parts: (a) the syntax tree of a chain of a language in which the identifiers of constants are terminal symbols and (b) the chain formed by the values of the constants, codified as in GA's (see Figure 3, right part). Mutation and crossover operations are defined to be equivalent to subtree exchange and random subtree replacement in the parse tree based representation.

2 Fuzzy rule based classifiers induction

A fuzzy rule based classifier can always be written as a chain in certain context-free language, so applying genetic programming to perform classifier induction is straightforward.

In this section we will use the names $C_1, C_2, \ldots, C_N$ to refer to the characteristics of an object (for example: C_1="length" and C_2="weight"). The name C_0 will be used to denote the variable "class", and this variable will take the values "class-1", "class-2", ..., "class-M". We will also use the names $V_{i1}, V_{i2}, \ldots, V_{in_i}$, $i = 0 \ldots N$, for denoting the symbolic labels of the n_i values of every variable. For instance, we can use V_{01} = "class-1", V_{02} = "class-2", V_{11} = "short", V_{12} = "medium", V_{13} = "long", and V_{21} = "light", V_{22} = "heavy".

2.1 Relation and rule based classifiers

We will define a fuzzy classifier by means of a fuzzy relation that maps every point in $C_0 \times C_1 \times \ldots \times C_N$ to a value between 0 and 1. Inference is made by intersecting the relation and the cylindrical extension of the fuzzy set that represents the measurement of the characteristics of the object being classified and projecting the intersection over C_0.

Every classifier defined by a relation can be written with rules, and the opposite is also true. Rule-based representation is useful to express linguistically the classifier's structure. We will use the following conventions:

2.1.1 Meaning of a rule

The following rule

$$\text{"if } C_i = V_{ij} \text{ then } \mathbf{t}(C_0 = V_{0k}) = \alpha\text{"}$$

where $\mathbf{t}(P)$ means "truth of assert P", is defined by the relation R_{ijk} that follows:

$$R_{ijk}(x_0, x_1, \ldots, x_N) = \begin{cases} \alpha & \text{If } x_i = V_{ij},\ x_0 = V_{0k} \\ 0 & \text{otherwise} \end{cases}$$

To shorten the notation, instead of writing

$$\text{"if } C_i = V_{ij} \text{ then } \mathbf{t}(C_0 = V_{0k}) = 1\text{"}.$$

we will write

$$\text{“}\texttt{if } C_i = V_{ij} \texttt{ then } C_0 = V_{0k}\text{”}$$

By last, the rule “$\texttt{if } \alpha \texttt{ then } C_0 = V_{0k}$” is defined by the relation $R_\alpha(V_{0k}, x_1, \cdots, x_N) = \alpha$, and $R_\alpha(x_0, x_1, \cdots, x_N) = 0$ for all $x_0 \neq V_{0k}$.

2.1.2 Logical expressions in antecedents

If the rules

$$\text{“}\texttt{if } A \texttt{ then } C_0 = V_{0k}\text{”}$$
$$\text{“}\texttt{if } B \texttt{ then } C_0 = V_{0k}\text{”}$$

are defined by means of the relations R_A and R_B, then the rules

$$\texttt{if } A \wedge B \texttt{ then } C_0 = V_{0k}$$
$$\texttt{if } A \vee B \texttt{ then } C_0 = V_{0k}$$

are defined by the relations

$$R_{A\wedge B}(x) = \min(R_A(x), R_B(x))$$
$$R_{A\vee B}(x) = \max(R_A(x), R_B(x))$$

respectively. With this definition, the rule

$$\text{“}\texttt{if } A \texttt{ then t(}C_0 = V_{0k}\texttt{)=}\alpha\text{”}$$

is identical to this one:

$$\text{“}\texttt{if } A \wedge \alpha \texttt{ then } C_0 = V_{0k}\text{”}.$$

2.1.3 Rule concatenation

Concatenation of N rules defined by the relations $R_1, R_2, \ldots, R_N$ is defined by the relation $R(x) = \max\{R_1(x), \ldots, R_N(x)\}$. As a corollary, two rules with the same antecedent can be combined in one:

$$\texttt{if } A \texttt{ then } C_0 = V_{0k}$$
$$\texttt{if } B \texttt{ then } C_0 = V_{0k}$$

is identical to

$$\texttt{if } A \vee B \texttt{ then } C_0 = V_{0k}$$

2.2 Induction of the deep structure of the bank

The “deep structure” of the bank of fuzzy rules comprises rule definitions and the semantic associated to every value, while the definition of the classifier without specifying the fuzzy sets linked to the semantics of the variables is named “surface structure” after Zadeh [5]. Surface structure of a classifier has been induced with genetic programming if the memberships of the linguistic variables are known [2]. We propose a method for simultaneously evolving fuzzy partitions of the variables and rule definitions, that is inducting the deep structure of a bank from examples.

The first step consists in parameterizing a Ruspini's partition for every characteristic. We use triangular membership functions and decided that no measures are compatible with more than two linguistic values, so a fuzzy partition depends on so many parameters as linguistic values. Then, the semantics of a fuzzy classifier can be codified with a set of $n_1+n_2+\ldots+n_N$ numbers.

The second step consists in choosing an adequate grammar. We will use a grammar like the one that follows, and GA-P representation, to codify a complete fuzzy classifier:

CLASSIFIER →
V-LOGICAL-CT V-PARTITION-CT
if COND then C$_0$=V_{01}
if COND then C$_0$=V_{02}
...
if COND then C$_0$=V_{0M}

COND → ASSERT-1 | ... | ASSERT-N |
(COND ∨ COND) | (COND ∧ COND) |
(COND ∧ LOGICAL-CT)

LOGICAL-CT → K$_1$ K$_2$ ... K$_{N_K}$

ASSERT-1 → trapezium-left(x$_1$,K$_{11}$,K$_{12}$) |
triangle(x$_1$,K$_{11}$,K$_{12}$,K$_{13}$) | ...
trapezium-right(x$_1$,K$_{1n_1-1}$,K$_{1n_1}$)

...

ASSERT-N → trapezium-left(x$_N$,K$_{N1}$,K$_{N2}$) |
triangle(x$_N$,K$_{N1}$,K$_{N2}$,K$_{N3}$) | ...
trapezium-right(x$_N$,K$_{Nn_N-1}$,K$_{Nn_N}$)

Non terminal symbol V-LOGICAL-CT is defined by concatenating the representation of the values of K$_1$ K$_2$ ... K$_{N_K}$ and V-PARTITION-CT contains the values of K$_{11}$ ... K$_{1n_1}$ ... K$_{N1}$ ... K$_{Nn_N}$

3 Numerical results

We have chosen the benchmark defined in [4] to contrast GA-P algorithms when inducting fuzzy classifiers. Every experiment has been repeated 30 times over three different permutations of the dataset. 75% of samples were used to design the classifier, 25% to

Dataset	K-NN	Linear Mean Dev	MLP Mean Dev	GA-P Mean Dev	Complexity Nodes	Variables Used Total
Cancer-1	1.7	2.93 0.18	1.38 0.49	3.06 1.00	37.7	4.5/9
Cancer-2	4.0	5.00 0.61	4.77 0.94	6.30 1.99	36.5	4.4/9
Cancer-3	4.5	5.17 0.00	3.70 0.52	4.50 1.79	35	4.4/9
Thyroid-1	5.95	6.56 0.00	2.38 0.35	4.92 0.94	34.5	4.1/21
Thyroid-2	6.00	6.56 0.00	1.91 0.24	4.80 0.82	37.7	4.5/21
Thyroid-3	6.50	7.23 0.02	2.27 0.32	4.88 0.88	36.6	4.3/21
Pima-1	25.5	25.83 0.56	24.10 1.91	24.92 1.89	34.1	3.7/8
Pima-2	27.6	24.69 0.61	26.42 2.26	29.68 2.05	24.3	3/8
Pima-3	23.5	22.92 0.35	22.59 2.23	24.29 2.17	25.4	3/8
Glass-1	35.8	46.04 2.21	32.70 5.34	39.62 4.73	73.9	6.7/9
Glass-2	33.9	55.28 1.27	55.57 3.70	42.11 2.78	81	5.8/9
Glass-3	35.0	60.57 3.82	58.40 7.82	41.15 4.72	68.5	5.3/9

Figure 4: Compared results between fuzzy classifiers obtained with GA-P algorithms and statistical and neural classifiers over the benchmark defined in [4].

test it by cross-validation. In Figure 4 the mean values obtained are shown for every permutation along with the standard deviations of the experiments. In all cases the algorithm was stopped after evaluating 50000 times the fitness function. Populations comprise 10 niches of 100 individuals each. The probability of crossover is 0.95, steady state was used and selection is performed with a tournament of size 4. Offspring replaces losers in the tournament.

Data relative to linear and neural classifiers is taken from [4]. The parameter of the k-neighbors classifier was chosen to minimize the classification error evaluated with jackknife. Observe that the dispersion of the results obtained when training a neural network from different starting points can be higher than the dispersion of GA-P results.

Since a classifier that decides between M classes can always be written with M rules, counting the number of rules makes no sense here. We have measured the complexity with the number of nodes in the classifier's syntax tree instead. Last column displays the mean number of characteristics a classifiers use.

4 Concluding Remarks

When designing fuzzy rule based classifiers, one of the main concerns is how much predictive accuracy has to be sacrificed for the sake of intelligibility. We have shown that GA-P algorithms can be used to induce the deep structure of fuzzy classifiers from data with good numerical behavior and obtaining compact solutions.

References

[1] Banzhaf, W. et al *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.

[2] Geyer-Schulz, A. *Fuzzy Rule-Based Expert Systems and Genetic Machine Learning*. Second edition. Physica-Verlag, 1997.

[3] Koza, J., *Genetic Programming*. MIT Press, Cambridge, MA. 1992

[4] Prechelt, Lutz. "PROBEN1 – A set of benchmarks and benchmarking rules for neural network training algorithms". Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.

[5] Zadeh, L. "Fuzzy Logic, Neural Networks and Soft Computing". Communications of the ACM, **37**(3), pp. 77-84. 1994.