Simulated annealing-based fuzzy classifier induction

Luciano Sánchez
Dep. Informática
Univ. Oviedo
luciano@lsi.uniovi.es

Inés Couso
Dep. Estadística
Univ. Oviedo
couso@pinon.ccu.uniovi.es

J.A. Corrales
Dep. Informática
Univ. Oviedo
ja@lsi.uniovi.es

Abstract

The use of genetic algorithms, genetic programming and hybrid methods of both to induce fuzzy classifiers is quite common. In this work we propose to use some elements of genetic programming to implement a version of the simulated annealing algorithm that can be used to search for the minimum of a function defined over the chains of a context-free grammar, and we apply this method to derive a fuzzy rule-based, descriptive classification system with good numerical properties. Simulated annealing and genetic programming are compared over some datasets and the results are discussed.

1 Introduction

Let us suppose we wish to describe a classification system by means of a descriptive fuzzy rule base and let us admit by now that (1) we know the equivalence between every linguistic term and its corresponding fuzzy set and (2) we have chosen a suitable fuzzy reasoning method. With these two hypotheses, a rule-based fuzzy classifier is equivalent to a fuzzy relation R defined over the Cartesian product of the linguistic terms set and the class marks set, as in the following example:

Example 1.1 *Let us suppose that we need to decide whether a piece of fruit is a banana or a pear by examining its weight and color. Color can be yellow or green, weight high or low, and the rules that describe the decision process are:*

```
if weight is high and colour is yellow
 then truth(class is banana) = 0.8
if weight is high and colour is yellow
 then truth(class is pear) = 0.1
[etc.]
```

which, alternatively, can be encoded in the fuzzy relation

$$\begin{aligned} R(\texttt{high},\texttt{yellow},\texttt{banana}) &= 0.8 \\ R(\texttt{high},\texttt{yellow},\texttt{pear}) &= 0.1 \\ &\dots \end{aligned}$$

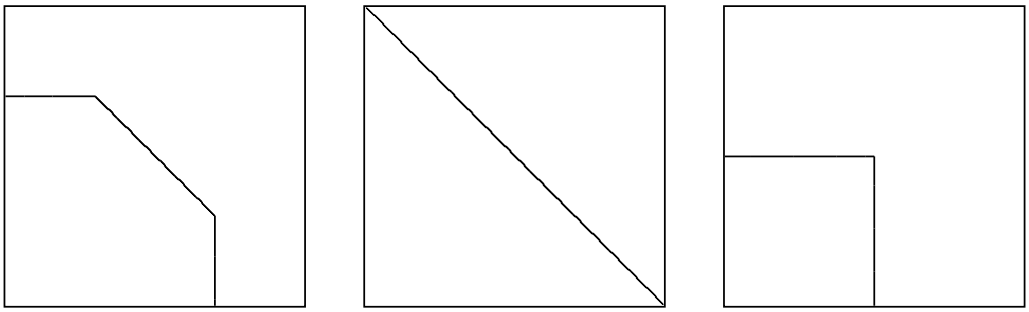
so called "surface structure" of the rule bank [8].

Given a definition for the fuzzy partitions of the variables in the universe of discourse (also called "deep structure", after Zadeh) and a formula to perform the fuzzy reasoning, every fuzzy relation R defines a crisp partition of the characteristics space. It is possible that different rule banks define the same fuzzy relation, and it is also known that different fuzzy relations can define the same crisp partition. For instance:

Example 1.2 *We have a set of objects belonging to classes C_0 or C_1, and we perform two different measurements over every object. Measurements can take real values ranking from 0 to 1, and we define for every measurement the labels "low", with $low(x) = 1 - x$ and "high", with $high(x) = x$. Let us define now the following rule bank:*

```
X is low and Y is low ⇒ (C0,C1)=(0,1)
X is low and Y is high ⇒ (C0,C1)=(α1,α2)
X is high and Y is low ⇒ (C0,C1)=(α1,α2)
X is high and Y is high ⇒ (C0,C1)=(1,0)
```

with α_1, $\alpha_2 \in [0,1]$. Observe that, when using the standard fuzzy reasoning method (see figure 1), for every α_1 all banks for which $\alpha_2 \leq \alpha_1$ will produce the same partition, and vice versa.



$$(x_1, x_2) \in \begin{cases} C_0 & \text{if } \max\limits_{X,Y \in \{\text{low, high}\}} \min(R(X,Y,C_0), X(x_1), Y(x_2)) \geq \\ & \quad \max\limits_{X,Y \in \{\text{low, high}\}} \min(R(X,Y,C_1), X(x_1), Y(x_2)) \\ C_1 & \text{otherwise} \end{cases}$$

Figure 1: Given a fuzzy partition of the universe of discourse and a fuzzy reasoning method, different fuzzy relations can produce the same classification system. In this figure we show the decision surfaces arising from the relation $R(\text{low}, \text{low}, C_0) = 0$, $R(\text{low}, \text{low}, C_1) = 1$, $R(\text{low}, \text{high}, C_0) = \alpha_1$, $R(\text{low}, \text{high}, C_1) = \alpha_2$, $R(\text{high}, \text{low}, C_0) = \alpha_1$, $R(\text{high}, \text{low}, C_1) = \alpha_2$, $R(\text{high}, \text{high}, C_0) = 1$, $R(\text{high}, \text{high}, C_1) = 0$; the reasoning method shown above, and the values $\alpha_1 = 0.3, \alpha_2 \in [0, 0.3]$, (left) $\alpha_1 = 0, \alpha_2 = 0$ (center) and $\alpha_1 = 0.5, \alpha_2 \in [0, 0.5]$ (right) for the rules defined in example 1.2

This last example is to recall that it is not only possible to define different rule banks that share the same surface structure, but it is also possible to obtain the same solution to a classification problem with different surface structures. The practical relevance of this fact (the rule-based representation of a fuzzy classifier being non unique) has to see with the concept of linguistic interpretation: we usually need to search for the shortest linguistic description, so just finding any rule bank that solves the numerical problem is not enough. For instance, in the last example, if $\alpha_1 = 0$ and $\alpha_2 = 0$ then rules 2 and 3 are redundant.

Because of this property, most of modern fuzzy rule induction methods (see [3] and cites contained therein) consist in two phases: in the first stage, the values of the fuzzy relation are determined (and the fuzzy partitions are tuned) to build an initial bank in which a fuzzy rule is tied to every not null term in the relation. In a second stage, the initial bank is simplified to obtain an easier classifier than can be described with a relation equal or very similar to the former one.

We follow a different approach (see [7]) and search simultaneously for both fuzzy partitions and the most simple linguistic expression by means of an optimization of a function defined over the chains of a language. We first define a fuzzy classifier as a valid chain in certain context free language, with an associated semantic that relates every rule bank with a fuzzy relation. The grammar we use permits defining classifiers in which not all input variables are present in every rule, thus combining the classification induction and feature selection phases in one (see appendix). Our work follows the line started in [4], where fuzzy rule induction is presented as a genetic search over the chains of a language that we have extended to allow tuning the fuzzy partitions of the input variables.

2 Genetic programming and simulated annealing

Up to our knowledge, the only optimization method that has been applied to optimice a function defined over the chains of a language is genetic programming based. We intend to show that the same results can be reproduced with simulated annealing, which is a much simpler and less memory consuming algorithm. In this section we will extend the Metropolis algorithm, defined as follows:

```
algorithm sa
```

Parmeter	Meaning	Value
Cooling rate	Temperature decrease / iteration	0.9999
T_0	Initial temperature	5
Iterations	Maximum number of iterations	50000
K_1	Inverse of maximum (percentual) jump in constant part	5
K_2	Inverse of maximum jump in expressional part	0
p	Probability of GA part being modified	0.5

Figure 2: Execution parameters of SA. Expressional part jumps are not limited, $K_2 = 0$, constant part jumps are limited to a 20%, $K_1 = 5$.

```
needs: cooling factor, Tinitial, Tfinal, p
produces: Cbest

T=Tinitial
C=Cbest=initial, random chain
while T>Tfinal do
      C1=adjacent(C,p,T,K_1,K_2)
      delta=error(C1)-error(C)
      v=random value U(0,1)
      if delta<0 or v<exp(-delta/T) then
         C=C1
         if C<Cbest then Cbest=C end if
      end if
      T=T*cooling factor
end while
```

where output the function "adjacent" is selected from the offspring of a GA-P subtree crossover of the chain C and other, randomly generated, chain of the language. The parameter p stands for the balance between GA and GP crossover probability and the temperature T will be related to the maximum edition distance allowed between the output of the function and the chain C:

```
algorithm adjacent
needs: C, p, T, K_1, K_2
produces: C1

      if U(0,1)<p then
         A=random real vector
         param(C)=
          param(C)*(T/K_1)+(1-(T/K_1))*A
      else
         repeat
           E=expr(C)
           select a subtree of E
           replace it by a randomly
                      generated subtree
         until edition_distance
                 (E,expr(C))<T/K_2
         expr(C)=E
      end if
```

`param(C)` and `expr(C)` stand for "chain of parameters" and "expressional part" of the chain C (see appendix). Parameters K_1 and K_2 adjust the amplitude of the jumps in the chain of parameters and expressional part, respectively.

3 Numerical results

The values of SA algorithm execution parameters are displayed in table 2. Every experiment has been repeated 10 times starting from different, random candidates. SA was rather more sensible to initial temperature and cooling pattern than GA-P is to its own execution parameters.

Genetic programming experiments used 10 subpopulations of 100 individuals, 10 niches in each one, one per cent migration. Steady state, selection with 3-size tournament and tournament losers were replaced by offspring. One per cent of the offspring is mutated. GA crossover is applied when both parents belong to the same niche and GP crossover otherwise. The number of inter-niche and intra-niche crossover is balanced (50% intra-niche crossover). Experiments were repeated 10 times from random populations. Evolution finishes after 50000 fitness evaluations in both SA and GA-P algorithms.

In table 3 we can see that SA is not different from genetic programming in all datasets. In table 4 the results of applying K-NN, linear classifiers and multilayer perceptrons to the same

Dataset	GA-P Mean Dev	Rules	Variables Uses/Tot	SA	Rules	Variables Uses/Tot
Cancer-1	2.58 0.78	5	5.1/9	2.93 1.07	4.9	4.9/9
Cancer-2	5.91 1.20	5.1	4.7/9	5.74 1.28	5.4	5.2/9
Cancer-3	5.34 1.33	5.4	4.3/9	5.28 0.56	5	5.3/9
Mean	4.62		4.7/9	4.65		5.1/9
Thyroid-1	5.18 0.67	5.3	4.8/21	4.51 1.08	5.5	5/21
Thyroid-2	4.51 0.80	5.9	3.9/21	4.16 1.34	5.2	5.2/21
Thyroid-3	4.90 0.70	5.6	4.3/21	4.38 1.66	4.8	5.1/21
Mean	4.87		4.3/21	4.35		
Pima-1	25.57 1.12	4.7	4.5/8	25.41 1.86	3.9	4.1/8
Pima-2	28.07 2.10	4.7	4.2/8	27.29 1.66	3.7	4.1/8
Pima-3	24.06 1.66	4.7	4/8	22.86 0.94	3.4	4.2/8
Mean	25.90		4.2/8	25.19		4.1/8
Glass-1	43.96 7.91	9.1	6.2/9	41.88 4.97	7.8	5.5/9
Glass-2	41.13 2.03	8.7	5.9/9	42.07 5.34	9.1	6.6/9
Glass-3	44.52 7.55	10.2	7.2/9	42.45 7.41	8.6	5.8/9
Mean	43.20		6.4/9	42.13		6/9

Figure 3: Comparison of results between genetic programming and simulated annealing when inducting fuzzy rule banks in classification problems. In this table, means and standard deviation of the test results obtained after repeating 10 times every experiment are shown. The equivalent "only and" number of rules and the mean number of variables used are also displayed.

Problem	K-NN	Linear Mean Dev	MLP Mean Dev
Cancer-1	1.7	2.93 0.18	1.38 0.49
Cancer-2	4.0	5.00 0.61	2.38 0.35
Cancer-3	4.5	5.17 0.00	3.70 0.52
Media	3.4	4.36	3.28
Thyroid-1	5.95	6.56 0.00	2.38 0.35
Thyroid-2	6.00	6.56 0.00	1.91 0.24
Thyroid-3	6.50	7.23 0.02	2.27 0.32
Media	6.15	6.78	2.18
Pima-1	25.5	25.83 0.56	24.10 1.91
Pima-2	27.6	24.69 0.61	26.42 2.26
Pima-3	23.5	22.92 0.35	22.59 2.23
Media	25.5	24.48	24.37
Glass-1	35.8	46.04 2.21	32.70 5.34
Glass-2	33.9	55.28 1.27	55.57 3.70
Glass-3	35.0	60.57 3.82	58.40 7.82
Media	34.9	53.96	48.89

Figure 4: Results obtained with K-NN, linear and multilayer perceptron classifiers over the same datasets used in the fuzzy rule induction problem. MLP and linear classifier results are taken from [6].

K_2	Pima-1	Pima-2	Pima-3	Pima-1	Pima-2	Pima-3	μ TRA	μ TST
5	39.28 11.68	39.06 12.54	41.61 14.43	40.63 10.17	41.15 8.69	41.03 14.46	36.65	40.93
1	21.65 0.59	21.15 0.35	22.38 0.44	25.88 1.42	28.22 1.41	22.60 1.53	21.72	25.55
0.5	21.14 0.46	20.83 0.47	22.25 0.38	25.15 1.51	28.95 1.45	23.54 1.37	21.40	25.86
0.2	21.28 0.41	20.87 0.44	21.86 0.46	25.78 2.00	27.66 1.87	23.39 1.39	21.34	25.61
0	21.18 0.41	20.63 0.43	21.89 0.42	25.42 1.81	27.29 1.66	22.86 0.94	21.23	25.19

Figure 5: Comparative results: SA with expressional part mutation limited to different degrees. Surprisingly, best results are obtain when limits are not applied.

	Variables (IM)	Variables (GAP)	Variables (SA)
Cancer	0 1 2 5 6	0 1 2 5 7	0 1 2 5 7
Pima	0 1 5 7	1 5 6 7	1 5 6 7
Glass	1 2 3 4 5 6	2 3 4 5 6 7	1 2 3 5 6 7
Thyroid	2 16 17 18 20	0 2 16 19 20	2 16 17 18 19

Figure 6: There exists a relation between the order of the variables as determined by their mutual information and the order defined by their relative frequency of use when SA or GP induction is repeated a number of times from random starting values.

datasets are included. "Not different" means that differences are not statistically significant if 5x2cv test is applied (95%).

In figure 5 the results of applying SA to PIMA dataset are tabulated for train and test partitions for different limits in the jumps in the expressional part. Best results are obtained when no restrictions are imposed over these jumps.

By last, in table 6 the most relevant characteristics, as found with an statistical method (a greedy algorithm that selects characteristics by querying mutual information between the class and every variable) and the most used variables in GP and SA are displayed.

4 Concluding remarks

In this work we have shown that genetic programming was not better than older and simpler SA method when applied to the search of the minimum of a function defined over the chains of a language in fuzzy classification problems. It remains to study whether this technique is also useful for inducting rules in other problems. Preliminary results in fuzzy modeling go in this direction.

Since the most simple version of SA was used, perhaps more elaborate cooling programs or parallel SA can improve the results.

References

[1] Díaz, A. *et al. Optimización Heurística y Redes Neuronales*. Paraninfo, 1996.

[2] Banzhaf, W. y otros *Genetic Programming: An Introduction. On the automatic evolution of computer programs and its applications*. Morgan Kaufmann, 1998.

[3] Cordón O., del Jesus M. J., Herrera F. y Lozano M. (1999) Mogul: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. *Intenational Journal of Intelligent Systems* 14(9).

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

[5] Howard, L., D' Angelo, D. "The GA-P: a genetic algorithm and genetic programming hybrid". IEEE Expert, pp 11-15. 1995.

[6] 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.

[7] Sánchez, L., García Carbajal, S. "Fuzzy classifier induction with GA-P algorithms".

ESTYLF-EUSFLAT joint conference, Palma de Mallorca, 1999.

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

A Grammar of a fuzzy rule based classifier and its associated semantic

A.1 Meaning of a fuzzy rule

The rule

"`if` x_i `if` $\widetilde{l}_{ij}$ `then class is` c_k `with conf.` α"

defines the relation R_{ijk} that follows:

$$R_{ijk}(x_1,\ldots,x_m,c) = \begin{cases} \alpha & \text{if } x_i = \widetilde{l}_{ij} \text{ and } c = c_k \\ 0 & \text{otherwise} \end{cases}$$

and the rule

"`if` α `then class is` c_k"

defines the relation

$$R_{\alpha k}(x_1,\ldots,x_m,c) = \begin{cases} \alpha & \text{if } c = c_k \\ 0 & \text{otherwise} \end{cases}$$

A.1.1 Antecedents with logical expressions

If the rules

"`if` A `then class is` c_k"

"`if` B `then class is` c_k"

define the relations R_A and R_B, then the rules

`if` $A \wedge B$ `then class is` c_k

`if` $A \vee B$ `then class is` c_k

define 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. As a consequence of this, the rule

"`if` $A \wedge \alpha$ `then class is` c_k"

describes the same relation as the rule

"`if` A `then class is` c_k `with confidence` α".

an that means that by introducing logical constants in the antecedent all relations can be described by rules with confidence 1.

A.1.2 Rule concatenating

The concatenation of N rules defined by relations $R_1, R_2, \ldots, R_N$ define the relation $R(x) = \max\{R_1(x), \ldots, R_N(x)\}$. Two rules with the same consequent can be combined in one. The construction

`if` A `then class is` c_k

`if` B `then class is` c_k

is identical to

`if` $A \vee B$ `then class is` c_k

A.1.3 Grammar of a fuzzy rule-based classifier

We will define a fuzzy rule based classifier as a chain of the following language:

```
CLASSIFIER → if CONDITION then class is c_1
             if CONDITION then class is c_2
             ...
             if CONDITION then class is c_{N_c}
             LOGICAL-CONS
             PARTITION-CONS
CONDITION → ASSERT_1 | ASSERT_2 | ... | ASSERT_m
             | (CONDITION ∨ CONDITION)
             | (CONDITION ∧ CONDITION)
             | (CONDITION ∧ logical–cons)
ASSERT_1 → left-trapezium(x_1,K_11,K_12) |
             triangle(x_1,K_11,K_12,K_13) |
             ...
             right-trapezium(x_1,K_{1n_1-1},K_{1n_1})
...
ASSERT_m → left-trapezium(x_m,K_m1,K_m2) |
             triangle(x_m,K_m1,K_m2,K_m3) |
             ...
           right-trapezium(x_m,K_{mn_m-1},K_{mn_m})
PARTITION-CONS → K_11 ... K_1n1 ... K_m1 ... K_mnm
LOGICAL-CONS → K_1 | K_2 | ... | K_Nc
```

where `left-trapezium`, `triangle` and `right-trapezium` are the fuzzy memberships usually used in triangular fuzzy partitions [7]. There are two semantic restrictions over the values of the constants:

1. Constants $\mathrm{K}_1\ \mathrm{K}_2\ \ldots\ \mathrm{K}_{N_c}$ take values between 0 and 1.
2. The lists $[\mathrm{K}_{11}\ \ldots\ \mathrm{K}_{1n_1}]\ \ldots\ [\mathrm{K}_{m1}\ \ldots\ \mathrm{K}_{mn_m}]$, are ordered and their values must be inside the range defined for every one of the variables.

The "GA part" of a chain comprises the terminal simbols under `LOGICAL-CONS` and `PARTITION-CONS`. "GP part" comprises all other symbols.