Reasoning Methods in Fuzzy Rule-based Classification Systems for Big Data Problems

Antonio González, Raúl Pérez and Rocio Romero-Zaliz
Dpto. Ciencias de la Computación e IA, Universidad de Granada, 18071-Granada, Spain

Keywords: Approximate Reasoning, Fuzzy Rules, Classifications Problems, Big Data.

Abstract: The analysis with a very high number of examples is a subject of growing interest that needs new algorithms and procedures. In this case, we study how the massive use of data affects the reasoning processes for classification problems that make use of fuzzy rule-based systems. First, we describe the standard reasoning model and the operations associated with its use, and once it is verified that these calculations may be inefficient in some cases we propose a new model to perform such calculations. Basically, the proposal eliminates the need to review all the rules in every inference process, generating the rule that best adapts to the particular example, which does not have to be part of the set of rules, and from it explore only the rules that have some effect on the example. We make an experimental study that shows the interest of the proposal presented.

1 INTRODUCTION

Currently, there are large amounts of data that need to be analyzed and this fact is causing it to be necessary to review many of the algorithms that were used to date. It is the so-called Big Data problem. This has happened for example in classification problems using fuzzy rule-based systems.

There are two types of proposals to address fuzzy rule-based classification systems for big data problems, one that makes use of a decomposition strategy (del Río et al., 2015; Elkano et al., 2018) and uses the MapReduce model (Dean and Ghemawat, 2008; Dean and Ghemawat, 2010), and another that makes use of a sequential model and incremental learning algorithms (Gámez et al., 2016; Romero-Zalíz et al., 2017).

In the first case, given that in the final phase a process of aggregation of rules is required, the basic model of a fuzzy rule with weights has been used as a rule model. In the second case, more elaborate fuzzy rule models can be used. The models that make use of the basic model of a fuzzy rule with weights are models that provide a simple solution to the problem, but usually generate a large number of fuzzy rules, which generates an added problem when you have a large number of examples.

There are several papers in the specialized literature that compare different methods of fuzzy reasoning from the perspective of selecting different parameters within the general model of fuzzy reasoning (Cordón et al., 1999; Mizumoto and Zimmermann, 1982). In our case, we start from the standard model used in classification problems of the winning rule and we want to analyze its functioning in problems with a massive number of examples, particularly when we start from a very high number of fuzzy rules and examples.

Thus, we want to analyze the influence of using a massive data set in the process of inference of basic fuzzy rules, and for this, we will take into account the complexity of the problem (number of variables involved, number of fuzzy rules and number of examples to infer).

The objective of this work is to highlight the great difficulty of carrying out inference processes on sets with many examples, when it is also carried out on a set with many fuzzy rules, as well as to propose a more efficient calculation of the reasoning model that can provide answers in a reasonable time to classification problems.

The proposed model generates the fuzzy rule that best adapts to the example, and from it generates the neighboring fuzzy rules that could cover this example. The result of the inference process is obtained only from the analysis of that set of rules, and it is not necessary to process the rest of the fuzzy rules. In the experimental part, we will analyze that when we use a knowledge base composed of a high number of rules this proposal is better than the standard

González, A., Pérez, R. and Romero-Zaliz, R.
Reasoning Methods in Fuzzy Rule-based Classification Systems for Big Data Problems.
DOI: 10.5220/0007709002550261
In *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS 2019)*, pages 255-261
ISBN: 978-989-758-369-8

inference model.

In the following section, we analyze the standard reasoning model with fuzzy rules and show a detailed algorithm that allows the calculation giving a set of fuzzy rules and an example. Section 3 shows an alternative method of calculation of the reasoning process that is analyzed in Section 4. Finally, Section 5 shows some conclusions.

2 REASONING METHOD

The basic model of the fuzzy rule with weight is

$$R_k : \textbf{IF } X_1 \text{ is } A_1^k \text{ and } X_2 \text{ is } A_2^k \text{ and } \ldots \text{and } X_n \text{ is } A_n^k$$

$$\textbf{THEN } Y \text{ is } B \text{ with } \textbf{weight } w^k$$

where $X_1, \ldots, X_n$ are the attributes, $A_1^k, \ldots, A_n^k$ are the linguistic terms taken for each attribute, and represented by fuzzy sets, Y is the consequent variable, B is the fuzzy value of the consequent variable and w^k is a measure of the weight associated with the rule.

Suppose we have a set of rules R=$\{R_1, R_2, \ldots, R_M\}$ and given an example e=$(e_1, e_2, \ldots, e_n)$ the fuzzy reasoning allows us to obtain the class associated with that example given the set of rules R. The procedure for obtaining the class associated with the example is very simple using the well-known method of the winning rule. Following the notation used in (Cordón et al., 1999) this method has the following steps:

- First we calculate the adaptation between the example and the antecedent of the rule by applying a t-norm T

  $$R^k(e) = T(\mu_{A_1^k}(e_1), \ldots, \mu_{A_n^k}(e_n)),$$

  where $\mu_{A_i^k}$ is the membership function of the fuzzy sets A_i^k.

- Next we incorporate the weight of the rule into this adaptation using a certain operator Op

  $$h(R^k(e), w^k) = Op(R^k(e), w^k).$$

- Finally we select the rule that maximizes this value, that is to say,

  $$max_k h(R^k(e), w^k) \qquad (1)$$

  and the class associated with that rule is assigned to the example.

Frequently, the minimum or the product is used as T-norms, and the product is used as the operator Op.

This fuzzy rule model has been used to obtain classifiers on big data problems (del Río et al., 2015; Elkano et al., 2018). These methods using the MapReduce model (Dean and Ghemawat, 2008; Dean and Ghemawat, 2010), and the Chi algorithm (Chi et al., 1996) learn a base of fuzzy rules in a relatively fast way but obtaining a very high number of rules. When it is necessary to perform the inference with massive databases and on a problem with a very high number of rules, the problem is that the inference can be very slow.

The possible slow running is due to the process necessary for the calculation of the equation 1. In calculating for that maximum you need to have the complete set of rules (which can be very large), evaluate the adaptation of each rule with the example R_k, and for all those that have some type of adaptation you need to calculate the maximum. This process is described step by step in Algorithm 1.

Algorithm 1: Standard inference process.

```
 1: BestCurrentMatching = 0
 2: for $i = 1$ to $M$ do
 3:     CurrrentMatching = 1
 4:     for $j = 1$ to $n$ do
 5:         CurrentMatching = T(CurrentMatching, $\mu_{A_j^i}(e_j)$)
 6:     end for
 7:     CurrentMatching = Op(CurrentMatching, $w^i$)
 8:     if CurrentMatching > BestCurrentMatching then
 9:         BestCurrentMatching = CurrentMatching
10:         Class = ClassOfRule(i)
11:     end if
12: end for
13: return Class
```

In this algorithm there are two nested FOR, the first one goes through all the rules, and the second one for each rule calculates the value of $R^k(e)$ using a t-norm and the membership function of each component of the example with each component of the antecedent of the rule. Once we have that value, the weight is included using the Op operator, that is, $h(R^k(e), w^k)$ is calculated and stored in the $CurrentMatching$ variable. Now begins the calculation of the maximum of the equation 1 to determine the rule with the maximum value of function h and return as output the class of that rule.

The calculation associated with inference when we have M rules and n antecedent variables per rule is of the order $O(nM)$ for each example. This order of complexity is reasonable when we work with a not very high number of variables and rules in the knowledge base and there are not many examples to classify. However, in problems where all these values are high, this inference algorithm may not be efficient. Let's suppose a problem similar to the one we will see in the experimental part that contains 28 variables and the knowledge base has 5 million rules and we want to classify 10 million examples. If we assume that the computational time needed to calculate each membership function $\mu_{A^i_j}(e_j)$ is 10^{-9} seconds, the time required to classify all the examples will be more than 16 days.

Therefore when we have a high number of examples and rules it is necessary to look for an alternative algorithm that reduces the response time of the inference process and in the next section, we will propose one.

3 ALTERNATIVE CALCULATION METHODS

In this section, we present a proposal to improve the time required to perform inference. First, we present a simple improvement of the previous algorithm in which it allows to stop calculating the function $R^k(e)$ when the current calculation already indicates that it is not better than the one obtained for another rule.

Algorithm 2: Improved inference process.

```
1:  BestCurrentMatching = 0
2:  for $i = 1$ to $M$ do
3:      CurrrentMatching = 1
4:      while $j \leq n$ and CurrentMatching > BestCurrentMatching do
5:          CurrentMatching = T(CurrentMatching, $\mu_{A^i_j}(e_j)$)
6:          $j = j+1$
7:      end while
8:      if CurrentMatching > BestCurrentMatching then
9:          CurrentMatching = Op(CurrentMatching, $w^i$)
10:         if CurrentMatching > BestCurrentMatching then
11:             BestCurrentMatching = CurrentMatching
12:             Class = ClassOfRule(i)
13:         end if
14:     end if
15: end for
16: return Class
```

The Algorithm 2 is just an improvement of the Algorithm 1 so if the partial calculation of the $R^k(e)$ function already shows that the current rule cannot be better than the current best rule, the calculation is stopped. Therefore it is an improved version that returns exactly the same output but reducing some calculations.

In any case, the problem that makes the inference process complex is to detect the set of rules whose antecedents cover the example. Both in the standard model and in the previous proposal, these rules are detected by checking one by one the complete set of rules, which is very inefficient when the number of rules is very high.

The alternative calculation proposal, that we are going to describe, uses the basic idea of the Algorithm 2 but also changes the way of detecting the set of rules that affect an example. Thus, from the concrete example is constructed the antecedent of the rule that best adapts to that example, in a way similar to how Chi's algorithm does. This antecedent does not have to belong to any rule of the set R on which we make the inference, but it will be the starting point to detect the rules that could affect the example.

If the antecedent corresponds to a rule of the set R, we calculate the value of the function h associated to that rule and the obtained value corresponds to the degree in which such rule assigns the particular consequent to that example. Then, as described above, we modify this antecedent in such a way that we generate a new antecedent close to the previous one and that still has possibilities of being applied to the example, and we repeat the process. The change from one consequence to another will be carried out following a branch and bound algorithm.

The following is an example of how the second proposal works. Let's suppose three variables X_1, X_2, X_3 with the same fuzzy domain (shown in Figure 1) composed of five labels $\{A_1, \ldots, A_5\}$, and a consequent variable Y that can take two values B_1 or B_2. Let us suppose that R is composed of the following two fuzzy rules:

R_1 : **IF** X_1 is A_2 and X_2 is A_3 and X_3 is A_3

THEN Y is B_1 with **weight 0.9**

R_2 : **IF** X_1 is A_1 and X_2 is A_3 and X_3 is A_4

THEN Y is B_2 with **weight 1.0**

Thus given an example e=(0.7,2.1,2.8), let us suppose the following values of the membership function

$$\mu_{A_1}(0,7) = 0.3\ \mu_{A_3}(2.1) = 0.9\ \mu_{A_3}(2.8) = 0.2$$

$$\mu_{A_2}(0,7) = 0.7\ \mu_{A_4}(2.1) = 0.1\ \mu_{A_4}(2.8) = 0.8.$$

The membership function for the values of the example for the rest of the labels is zero. Let's suppose we use the minimum T-norm and the *Op* operator product.

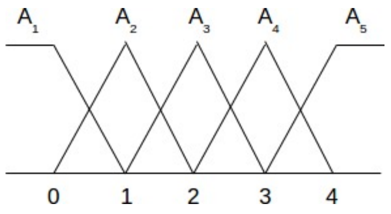

Figure 1: Fuzzy domain of variables X_1,X_2 and X_3.

The process starts by assigning to X_1 the label that gives it the greatest value of membership on the example, in this case A_2, continues with variable X_2 and repeats the same criterion, assigns A_3, continues with X_3 and assigns A_4. But the antecedent

$$X_1 \text{ is } A_2 \text{ and } X_2 \text{ is } A_3 \text{ and } X_3 \text{ is } A_4$$

is not in R. Therefore it goes back and assigns X_3 the second best value for the example, in this case A_3. Now the assignment coincides with the rule R_1, the assigned class would be B_1 and the value h=0.1134.

The process continues going through all the nearby rules that could apply to the example. On the previous consequent, it goes backward looking for new consequents, since there are no more possible labels for the variable X_3, so go back to the variable X_2 and try the second best label A_4. But in this case the partial calculation of function h is equal to 0.07 which is less than the value of function h of the antecedent previously found, therefore it is not possible that an antecedent with the current assignment is better than the one previously found, we do not continue assigning, and we do a return backward.

Now the backward step assigns the A_1 label for the X_1 variable, the A_3 label for X_2 and the A_4 label for X_3, this assignment corresponds to the R_2 rule, and we assign the consequent B_2 with h=0.216. As the value of the function h is greater than the value of the rule found before, we discard the previous assignment and we keep this new assignment. The process continues and we assign A_3 to X_3, but the partial value is 0.054 and the assignment is discarded. We go back, and the last possible assignment is A_4 to X_2, again the partial assignment is 0.03 and again it is discarded. Thus, the process ends up assigning the class B_2 with value h=0.216.

It is important to note that this process only looks at the rules that could be applied to the example, and does not have to review the rest of the rules. This new procedure, which we will call Algorithm 3, is an improved calculation version of the standard inference algorithm, which again returns the same output but reduces the necessary calculations.

A basic element for this alternative way to calculate the maximum to be efficient, in comparison with the previous one, is that the operation to determine whether the antecedent of the rule exists in the rule base must be of order O(1). This is achieved by using a method of coding the rules that are used as a key to store them in a hash table.

In the worst case, this proposal explores in a recursive way all the rules neighboring the rule. In the case of problems where the domains are discretized by means of fuzzy labels distributed uniformly and with the cut in 0.5, the number of rules to evaluate is 2^d, being d the number of continuous variables discretized by means of fuzzy labels, being $d \leq n$.

The order of complexity in the worst case of the algorithms 1 and 2 is $O(nM)$, and is better than $O(2^d)$ which is the order of complexity of the proposed algorithm. However, this second one is independent of the number of rules and this allows that in problems with a great number of rules and a low number of variables (discretized as fuzzy variables), in the practical application, this algorithm presents better response time, as we will check later. For example, in the census database (see Table 1), the number of continuous variables is 8 so Algorithm 3 would have to study 256 rules at most, but instead, this number for Algorithms 1 and 2 would be 80791 rules (see Table 2).

However, with a low number of rules and many continuous variables, the algorithm 2 presents better behavior. Therefore, we could use an inference algorithm that works with two processes (each implements an inference algorithm) that run in parallel, so that once one of them finds the solution, it eliminates the other thread and returns the output found. Thus, the response time of the algorithm will always be limited in the worst case to $O(nM)$. In any case, in this work, we want to know the behavior of the recursive algorithm by itself when it is used on some of the most used databases to test fuzzy models in big data problems. In the next section, we show this experimental study.

4 EXPERIMENTAL STUDY

The purpose of this experimental part is to show the behavior of the new calculation models for reasoning (Algorithms 2 and 3) in terms of response time, on some of the databases that have been used in recent proposals to address the problem of classification through algorithms based on the use of fuzzy rules.

In this experimentation, we will use eight databases from the UCI dataset repository (Bache and Lichman, 2013), which are described in Table 1. In this table, *Nemo* refers to the short name for each dataset, *#Ex* represents the number of examples and *#Atts* shows the number of attributes and *Cont* reflects the number of continuous attributes (all of them discretized as fuzzy domains). Some of the databases (those marked with a *) have been converted to a binary classification problem in the same sense that they were used in (Gámez et al., 2016) and (del Río et al., 2015). Furthermore, in the same way of the previous papers, in the Poker database, 5 attributes originally considered as an integer in the range $\{1,13\}$, have been considered as continuous variables in the range $[1,13]$.

Table 1: Summary of Databases. #Ex is the total number of examples, #Atts is the total numbers of attributes and #Cont is the number of continuous attributes.

Dataset	Nemo	#Ex	#Atts	#Cont
Census	cens	141544	41	8
Covtype*	covt	495141	54	10
Fars*	fars	62123	29	5
Hepmass	hepm	10500000	28	27
Higgs	higg	11000000	28	28
Kddcup*	kddc	4856150	41	26
Poker*	poke	946799	10	5
Susy	susy	5000000	18	18

In this experimental study to obtain the fuzzy rule set, we have used the well-known Chi algorithm (Chi et al., 1996). The rule structure used by this classifier is the same that was described in section 2, being the weight of the rule computed by the Penalized Certainty Factor (Ishibuchi and Yamamoto, 2005). In the implementation of this algorithm, we have used the product both as t-norm and Op operator. Furthermore, for this experimental part, we are used domains of three fuzzy labels uniformly distributed for all the continuous variables and ten cross-validations.

An experiment has been carried out on the databases considered and on the sets of fuzzy rules obtained for each database. The experimentation has been carried out on a computer with Intel(R) Core(TM) i7-6700 CPU 3.40GHz with 8 cores and 16 Gb of memory, working under the Linux operating system (Fedora 28). The implementations of the Chi algorithm necessary to obtain the rules, as well as the three forms of inference calculation described previously in this work have been implemented in C++ (using the gcc compiler version 8). In the implementations, the data structures of the STL library have been used. In this sense, indicate that the implementation used for the definition of the hash table has been taken from this library.

The results obtained in the number of rules and accuracy on training and test sets using this algorithm on the previous database are shown in Table 2. Obviously, as the reasoning model does not change, any of the three calculation proposals (Algorithms 1, 2 and 3) considered on that reasoning model will give the same result in the number of rules and accuracy. On the other hand, the result may vary if we analyze the time required for each proposal. In some databases the number of rules obtained by Chi's algorithm is very high especially taking into account the number of examples processed, being able to reach 67.33 of the number of examples (this happens for example in the "fars" database).

Table 2: Results obtained on accuracy and number of rules, applying Chi algorithm using 10 cross validation.

Database	#R	%Train	%Test
cens	80791	98.72	49.82
covt	8294	77.29	76.73
fars	41825	100.00	49.88
hepm	5523370	87.16	70.47
higg	929050	62.40	56.84
kddc	772	99.95	99.95
poke	196403	76.77	56.63
susy	11470	64.44	68.02

Table 3 shows the behavior of the different inference algorithms that we have considered. The column labeled "Alg1" shows the number of inferences per second that Algorithm 1 performs on each database. The "Alg2" column shows both the number of inferences per second that Algorithm 2 performs on each database and the percentage of time needed in relation to that used by the "Alg1". These same two measures are presented for the Algorithm 3 under the column "Alg3". The last row shows the mean of the values obtained in each of the columns. The values corresponding to the databases "hepm" and "higg" for the algorithms "Alg1" and "Alg2" are marked with an "*". This asterisk indicates that the values shown here correspond to the estimation of those values after 12 hours of execution, since the complete execution of only the first of the ten executions of the cross-validation for the 'hepm" using Algorithm 1 would consume more than 32 days of computation, that is, almost a full year to complete the cross-validation.

A simple analysis of the results presented in Table 3 shows that the proposals presented significantly reduce the time of the inference process compared to Algorithm 1. In 5 of the 8 cases, the time is reduced by more than 99%, and in the case less favorable to the proposal, the "kddc" database, the reduction is greater than 90%. A deeper analysis of the "kddc" database, allows us to verify that it is the database that has the least number of rules, and has a high number of continuous attributes, 26, which are just the conditions in which the proposal presents an order of complexity in the worst case of 2^{26}.

Table 3: Number of inferences that each algorithm can make in a second and percentage of time reduction obtained to process the examples compared to Algorithm 1.

Data	Alg1	Alg2	%	Alg3	%
cens	32.5	34.9	6.76	9502.0	99.63
covt	251.3	456.7	49.97	17810.4	97.44
fars	69.8	71.0	1.68	24852.0	99.71
hepm	0.3*	0.4*	6.31	611.2	99.93
higg	2.3*	2.4*	2.48	991.2	99.76
kddc	6316.4	6396.1	1.25	67363.0	90.51
poke	7.0	21.7	67.80	242051.3	99.99
susy	242.6	365.8	33.68	25706.9	98.58
mean	865.3	918.6	20.62	48611.0	98.19

It can also be observed that while Algorithm 2 produces an improvement in time of 20.62% over the Algorithm 1, the Algorithm 3 produces a reduction of 98.19%, going from being able to make 865 inferences per second to more than 48000. Algorithm 3, as expected, presents its best results in those cases where a high number of rules are used, such as "hepm", "higg" and "poke" databases. It also shows good results when the number of continuous attributes is small in relation to the total number of attributes.

These results show that the process of evaluation of membership functions and pruning used in the proposal makes it more efficient than the other two algorithms.

In summary, the results presented show that the proposed algorithm provides a substantial improvement in the process of inference in problems where the description of knowledge contains a high number of rules and/or the number of continuous attributes (discretized as fuzzy domains) is small in relation to the total number of attributes.

5 CONCLUSIONS

The use of a large number of examples generates problems in obtaining knowledge from these examples, but also in using them in a reasoning model. Some of the algorithms proposed in the field of fuzzy logic to deal with big data problems have the disadvantage of generating a very high number of rules. The standard inference algorithm of the winning rule used in fuzzy logic has problems when confronted with knowledge bases with many rules.

The modified version of the algorithm that optimizes the calculation through adaptation thresholds is not sufficient to significantly improve the response time.

We have analyzed the problem and have proposed a model for reasoning that is more efficient than the standard one in cases where there are a large number of fuzzy rules. Thus, we have presented a calculation of the inference algorithm that uses a different strategy to obtain the winning rule. Although according to its order of complexity is a worse approximation than the original algorithm, in the experimental part we show that it presents a significantly better behavior applied to databases than the field of classifiers based on fuzzy logic are being applied.

In future work, it seems interesting to combine both algorithms using a parallel model to ensure a better time response to the inference process.

ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish MEC Projects TIN2015-71618-R, DPI2015-69585-R and co-financed by FEDER funds (European Union).

REFERENCES

Bache, K. and Lichman, M. (2013). Uci machine learning repository.

Chi, Z., Yan, H., and Pham, T. (1996). *Fuzzy algorithms: with applications to image processing and pattern recognition*, volume 10. World Scientific.

Cordón, O., del Jesus, M. J., and Herrera, F. (1999). A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximate Reasoning*, 20:21–45.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Commun ACM 51(1), 107-113*.

Dean, J. and Ghemawat, S. (2010). Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77.

del Río, S., López, V., Benítez, J. M., and Herrera, F. (2015). A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. *International Journal of Computational Intelligence Systems*, 8(3):422–437.

Elkano, M., Galar, M., Sanz, J., and Bustince, H. (2018). Chi-bd: A fuzzy rule-based classification system for big data classification problems. *Fuzzy Sets and Systems*, 348(1):75–101.

Gámez, J. C., Garcia, D., González, A., and Pérez, R. (2016). On the use of an incremental approach to learn fuzzy classification rules for big data problems. In *2016 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 1413–1420.

Ishibuchi, H. and Yamamoto, T. (2005). Rule weight specification in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 13(4):428–435.

Mizumoto, M. and Zimmermann, H.-J. (1982). Comparison of fuzzy reasoning methods. *Fuzzy Sets Syst.*, 8(3):253–283.

Romero-Zalíz, R., González, A., and Pérez, R. (2017). Incremental fuzzy learning algorithms in big data problems: A study on the size of learning subsets. In *Proceedings of the 2017 IEEE International Conference on Fuzzy Systems*, pages 1–6.