FuzzyCN2: An Algorithm for Extracting Fuzzy Classification Rule Lists

Pablo Martín-Muñoz and Francisco J. Moreno-Velo

***Abstract*—Most of the algorithms for extracting fuzzy classification rules generate conjunctive antecedents that use all the attributes of the system. Using this kind of antecedents, the number of rules grows exponentially in terms of the number of attributes of the system. This paper presents a new algorithm, FuzzyCN2, for extracting conjunctive fuzzy classification rules. This algorithm is a fuzzy version of the well known CN2 algorithm and produces an ordered list of fuzzy rules. FuzzyCN2 generates antecedents that may not include all the attributes of the system. These antecedents may cover a wide number of instances and, so, the number of extracted rules is smaller. The algorithm introduces the use of linguistic hedges as part of the selectors, thus producing more compact rules and reducing the number of generated rules.**

I. Introduction

SYSTEMS based on fuzzy rules have proved to be an appropriate tool on classification problems. On the one hand, the use of fuzzy rules facilitates the induction of interpretable models from the sets of examples of classification. On the other hand, the output of a fuzzy classifier is not limited to the selected class, but indicates the degree of activation of each class which may be related to the degree of confidence given to the election of each class. Fuzzy classifiers are particularly suitable to problems in which the attributes are continuous. In these cases, classic rule-based systems are built in term of consults of the form *attribute* $\leq$ *value* [1]. The inference on these classifiers presents abrupt changes around the values of consultation. By contrast, the behavior of fuzzy classifiers reflects an assessment of the activation degrees of each class much smoother thus easing to detect situations in which decision making is more risky.

The best known algorithm for extracting fuzzy rules was proposed by Wang and Mendel [2]. This algorithm is based on first define linguistic labels associated with each attribute, and then generate the most active fuzzy conjunctive rule for each training example. The rules generated by the Wang and Mendel algorithm contain queries on the values of all the attributes of the system. This causes an exponential growth in the number of rules in terms of the number of attributes of the problem. Many of the proposed algorithms for extracting fuzzy rules try to reduce the number of rules generated by the Wang and Mendel algorithm but keep the form of the rules, that is, conjunctive rules containing all of the attributes (for instance, [3], [4]).

This contrasts with the approach generally followed in the algorithms for extracting classic rules. The goal in these algorithms is to minimize the number of rules by inducing more general rules in which not all the attributes are present. Among these algorithms the best known is ID3, proposed by Quinlan [5], which generates decision trees, and the AQ family of algorithms proposed by Michalsky [6], which generate sets of classification rules. Based on these algorithms, Clark and Niblett proposed the CN2 algorithm in 1989 [7], which induces ordered lists of classification rules. Later, Clark and Boswell suggested some improvements to this algorithm [8].

Some proposals for extracting fuzzy classification rules are based on fuzzy versions of the algorithms above. There are many adaptations of the ID3 algorithm to generate fuzzy decision trees [9]-[12]. The CN2 algorithm has also been adapted to generate fuzzy classification rules [13] but in this case the proposed algorithm is based on the CN2 version which generates unordered list of rules.

This paper presents a different adaptation of the CN2 algorithm for extracting ordered lists of fuzzy rules. The proposal contains not only a way to adapt the different procedures and metrics of CN2 to the fuzzy case, but also adds the capability to use linguistic hedges as part of the queries on the attributes, thus inducing more compact and expressive rules than those used usually in fuzzy classifiers.

II. The CN2 algorithm

The CN2 algorithm was proposed based on the AQ family of algorithms, trying to introduce the ability of TDIDT algorithms (Top Down Induction of Decision Trees) to treat with noisy data. From the AQ algorithms, CN2 takes the idea of finding the best set of rules through several parallel beam searches (what is known as a star search). From the TDIDT algorithms, CN2 takes the idea of ending the search when they found the rules which do not exceed a certain threshold for statistical significance (pruning techniques).

The algorithm is based on an external loop in which, given a set of classification examples, the best rule for this examples is found and, then, the examples covered by that

This work was supported in part by the Spanish CICYT Projects TEC2008-04920/TEC, and TIN2008-06681-C06-06, and by the Projects P07-TIC-03179 and P08-TIC-03674 from the Andalusian Regional Government.

P. Martín-Muñoz is with the Unidad para la Dirección Estratégica, Universidad de Huelva, Huelva, SPAIN (e-mail: pablo.martin@sc.uhu.es).

F. J. Moreno-Velo is with the Departamento de Tecnologías de la Información, Universidad de Huelva, Huelva, SPAIN (e-mail: francisco.moreno@dti.uhu.es).

rule are removed from the set. The loop ends when the set of examples is empty or when no rule is found with the required level of significance. Since the rules are generated from a set of examples where several examples may have been eliminated, the result must be interpreted as an ordered list of rules, so that a rule must be considered only in cases where the precedents rules are not active.

In order to find the best rule related to an instance set, a series of parallel beam searches are made. On each iteration, rules candidates are specialized by adding a new selector (a query term relating one attribute and one of its values). In the AQ algorithms, all the selectors used in the search was obtained from a positive example (seed). To avoid reliance on this example, CN2 consider all possible selectors in its search process.

In order to guide the process of finding the best rule, CN2 takes some ideas of the TDIDT algorithms. The heuristics for quality, using in guiding the search beam, is the entropy of information (such as in the ID3 algorithm). To avoid the generation of poorly significant rules, the algorithm introduces a second heuristic (the likelihood ratio) which is used as a pruning in the search process.

A later work, presented in 1991, proposes the use of the laplacian error estimate as the heuristic for quality, instead of the entropy of information. The study of this new heuristic shows that the effect of statistical significance in this case is just a stopping criterion because the laplacian error drives the search to the most significant rules by itself. The work contains also a version of the algorithm for generating unordered sets of rules.

The FuzzyCN2 algorithm, proposed in this paper, is based on a fuzzy version of the CN2 algorithm generating ordered lists of rules. The measure used as heuristic for quality is a fuzzy version of the laplacian error estimate and the measure used for the significance test is based on a minimum value for the coverage of the rules.

III. Ordered Lists of Fuzzy Rules

FuzzyCN2 produces an ordered list of fuzzy rules rather than an unordered set of fuzzy rules. Both representations have their respective advantages and disadvantages for comprehensibility. In an ordered list of fuzzy rules, the interpretation of a single rule depends on which other rules preceded it on the list. This can lead to a lost in the comprehensibility of the rule list. On the other hand, an unordered set of fuzzy rules may contain rules covering the same input space. In this case, the rule set needs some additional mechanism to decide which rule governs the classification when these conflicts appear. So, the interpretation of these overlapping rules cannot be considered as independent.

Table 1 shows these two kinds of fuzzy rule sets. The inference mechanism on an unordered list consists on taking the activation degree of the consequent of a rule ($\mu(Ci)$) as the activation degree of the antecedent ($\mu(Ai)$). The inference mechanism on an ordered list is different. In an ordered list there is and implicit else condition in each rule with respect to the previous ones. So the activation degree of a consequent ($\mu(Ci)$) depends on the activation degree of its antecedent ($\mu(Ai)$) and also on the activation degrees of the previous ones ($\Pi(1-\mu(Aj))$).

TABLE I
DIFFERENT KINDS OF FUZZY RULE SETS

Unordered list	Ordered list
if(A1) then C1	if(A1) then C1
if(A2) then C2	elseif(A2) then C2
if(A3) then C3	elseif(A3) then C3
$\mu(C1) = \mu(A1)$	$\mu(C1) = \mu(A1)$
$\mu(C2) = \mu(A2)$	$\mu(C2) = (1-\mu(A1))\bullet \mu(A2)$
$\mu(C3) = \mu(A3)$	$\mu(C3) = (1-\mu(A1))\bullet(1-\mu(A2))\bullet \mu(A3)$

IV. The FuzzyCN2 algorithm

There are several features that need to be redefined in order to propose a fuzzy version of the CN2 algorithm. The section above describe how does an ordered list of fuzzy rules work, but this is not the only aspect to be considered. When creating a fuzzy version of the CN2 algorithm there are some problems to solve:

1) How to remove the examples covered by a fuzzy rule? As a fuzzy rule covers an input vector only in a certain degree (from 0 to 1), the example should be only partially removed.
2) How to represent a set of classification examples containing instances that are partially removed?
3) How to compute the coverage of a fuzzy rule over an example that is partially removed?
4) How to represent a selector (a query relating an attribute and one of its values) of a fuzzy rule?
5) How to traduce the Laplacian error estimate to the fuzzy case?
6) How to compute a significance test over a fuzzy rule and a set of examples partially removed?

The following definitions give an answer to the previous questions.

Definition 1. A *fuzzy selector* is a fuzzy term relating an attribute and one of its linguistic labels. This relation can refer to the usual equality relation (i.e., *attr is equal to label*) but also to relations based on linguistic hedges (as, *attr is greater than label*). Fig. 1 shows the different fuzzy selectors used in FuzzyCN2.

Definition 2. A *complex* refers to a conjunction of some fuzzy selectors. They are used as antecedents of the fuzzy rules generated by FuzzyCN2.

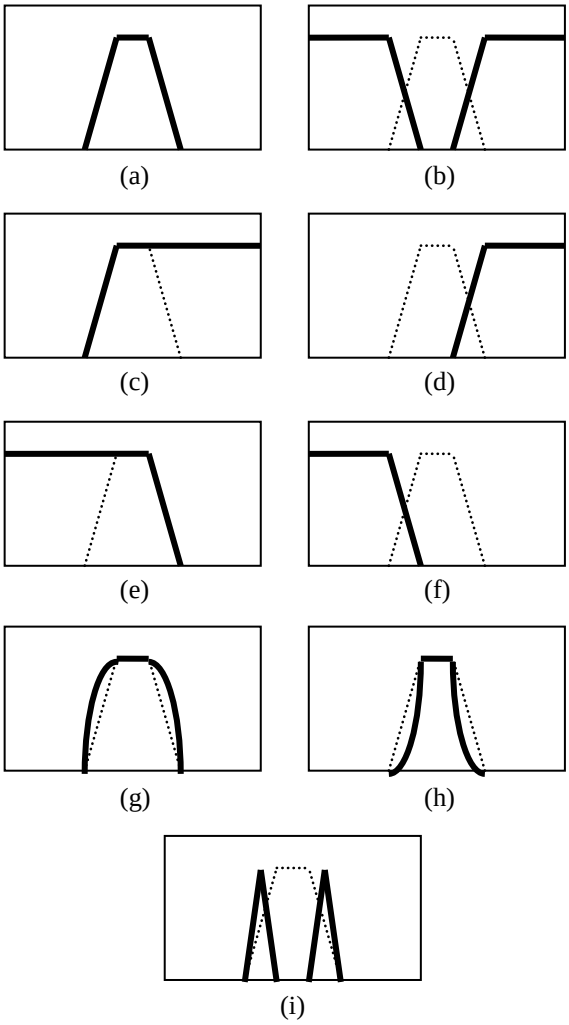

Fig. 1. Fuzzy selectors used in FuzzyCN2 for each attribute (*attr*) and label (*lb*). (a) *attr is equal to lb*; (b) *attr is not equal to lb*; (c) *attr is greater than or equal to lb*; (d) *attr is greater than lb*; (e) *attr is less than or equal to lb*; (f) *attr is less than lb*; (g) *attr is approximately equal to lb*; (h) *attr is strongly equal to lb*; (i) *attr is slightly equal to lb*.

Definition 3. A *fuzzy instance set* is a set of classification examples in which every example has an associated activation degree. A *crisp instance set* can be seen as fuzzy instance set in which every activation degree is equal to the unit. A fuzzy instance set can be *truncated* by deleting every instance which activation degree is less than a certain *cut value*.

Definition 4. Given a complex C and an example e with an activation degree of $\mu(e)$, the *support of C over e* is defined as the product of the activation degree of that complex when evaluating the example and the activation degree of the example, that is,

$$\text{support}(C,e) = \mu(C,e) \cdot \mu(e) \tag{1}$$

Definition 5. The *support of a complex C over a fuzzy instance set S* is defined as the sum of the support of C over every example e of the set.

$$\text{support}(C,S) = \sum_{e \in S} \text{support}(C,e) \tag{2}$$

Definition 6. The *support of a complex C over a class c and a fuzzy instance set S* is defined as the sum of the support of C over every example e of the set which class is c.

$$\text{support}(C,S,c) = \sum_{e \in S, class(e)=c} \text{support}(C,e) \tag{3}$$

Definition 7. The *relative support of a complex C over a fuzzy instance set S* is defined as

$$\text{relative\_support}(C,S) = \text{support}(C,S) \Big/ \sum_{e \in S} \mu(e) \tag{4}$$

Definition 8. The *deactivation of the fuzzy instance set S in terms of the complex C* consist in updating the activation degree of each example e in the instance set in the following way:

$$\mu_{deactivate}(e) = \mu(e) \cdot (1 - \mu(C,e)) \tag{5}$$

Definition 9. A complex C is said to be *significant* with respect to a fuzzy instance set S if the relative support of C over S is greater than a certain given value (*minimum relative support*) and the support of C over S is greater than a certain given value (*minimum absolute support*).

Definition 10. The *fuzzy Laplacian error estimate* of a complex C is a measure of the quality of that complex and is defined as

$$Laplacian(C,S) = \frac{\text{support}(C,S,c)+1}{\text{support}(C,S)+k} \tag{6}$$

where c is the class with the highest support and k is the number of classes.

The definitions above let us to present a fuzzy version of the CN2 algorithm. Table 2 shows the pseudocode of the FuzzyCN2 algorithm. The algorithm takes a set of classification examples and generates an ordered list of fuzzy rules.

TABLE II
THE FUZZYCN2 ALGORITHM

```
procedure FuzzyCN2(InstanceSet)
returns RuleList
  RuleList ← ∅
  FuzzyInstSet ← AddActivationDegree(InstanceSet)
  Complex ← FindBestComplex(FuzzyInstSet)
  while IsNotNil(Complex) and IsNotEmpty(FuzzyInstSet)
    Conseq ← MostCoveredClass(FuzzyInstSet,Complex)
    NewRule ← "if( Complex ) then class = Conseq"
    RuleList ← Add(RuleList,NewRule)
    FuzzyInstSet ← Deactivate(FuzzyInstSet,Complex)
    FuzzyInstSet ← Truncate(FuzzyInstSet)
    Complex ← FindBestComplex(FuzzyInstSet)
  endwhile
  return RuleList
end
```

```
procedure FindBestComplex(FuzzyInstSet)
returns BestComplex
  BestComplex ← nil
  Selectors ← CreateAllSelectors()
  Star ← InitializeStar()
  while IsNotEmpty(Star)
    NewStar ← SpecializeStar(Star, Selectors)
    NewStar ← SignificanceTest(NewStar,FuzzInstSet)
    NewStar ← QualityTest(NewStar,FuzzyInstSet)
    BestComplex ← BetterQuality(NewStar,BestComplex)
    Star ← NewStar
  endwhile
  return BestComplex
end
```

As the CN2 algorithm, the FuzzyCN2 algorithm is divided into two parts. In the external loop, a fuzzy instance set is considered by adding the activation degree to the set of classification examples (this is the goal of the *AddActivationDegree* method). Each iteration on that loop consist in obtaining a new fuzzy rule (the *FindBestComplex* method gives the antecedent of the new rule and the *MostCoveredClass* method give the consequent of the rule), adding this rule to the ordered list (the *Add* method), deleting partially those examples covered by the new rule (the *Deactivate* method) and truncating the fuzzy instance set to eliminate those examples poorly activated (the *Truncate* method). The loop ends when the fuzzy instance set is empty or when no complex is found. The last step is to add the default rule to the ordered list. The consequent of this default rule is the most populated class of the instance set.

The internal loop of the algorithm is implemented by the *FindBestComplex* procedure. This method takes a fuzzy instance set and generates the better significant complex in term of the heuristic of quality. The procedure considers a set of selectors (the *CreateAllSelectors* method returns a set with the selectors for each attribute and label using those linguistic hedges chosen by the user). The star contains a set of candidate complexes. Initially, the star just contains a complex with no selectors (the *InitializeStar* method). The internal loop consists in creating a new star with all possible complexes generated by adding one of the selectors to one of the complexes of the star (the *SpecializeStar* method). Then those complexes of the new star which are not significant (as describe in Definition 9) are removed from the new star (the *SignificantTest* method). After that, the heuristic of quality is computed as described in (6) and the worst complexes are removed from the new star thus generating a star of a given size (the *QualityTest* method). Finally, if there is a complex in the new star that is better that the best complex found previously, the best complex is set to this new one (the *BetterQuality* method). The loop ends when the new star is empty, that is, when every specialized complex in the new star is found to be not significant.

In order to maintain the interpretability of the rules, there is a constraint in the *SpecializeStar* method when adding a selector to a complex. An attribute cannot appear twice into the complex, unless in the relation "*attr is greater than* $label_1$ *and attr is less than* $label_2$". So, complexes like "*attr is strongly equal to* $label_1$ *and attr is less than* $label_2$" are forbidden.

V. EXPERIMENTS

This section shows an experimental study on the behaviour of the FuzzyCN2 algorithm and some other fuzzy and non-fuzzy classification algorithms. In this study, we have used 11 well known classification datasets extracted from the UCI repository on machine learning [14].

The selected datasets are all formed by continuous attributes. In the case of the fuzzy algorithms, the attributes have been described by means of three linguistic labels as shown in Fig. 2b. Regarding the non-fuzzy algorithms, the attributes have been discretized into three intervals with the same size, as shown in Fig. 2a.

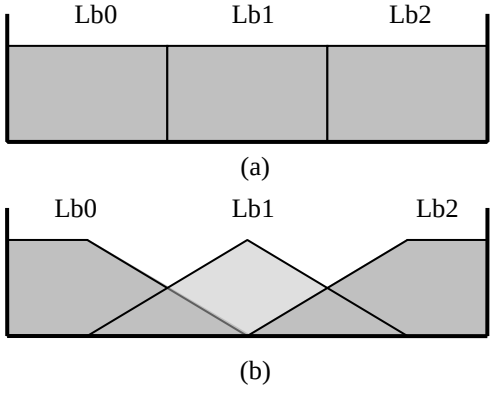

Fig. 2. Discretization on the continuous attributes used in the study (a) with the non-fuzzy algorithms; (b) with the fuzzy algorithms.

TABLE III
MEAN NUMBER OF RULES INDUCED BY DIFFERENT ALGORITHMS

	ID3	RIPPER	W-M	FUZZYID3	FUZZYCN2 (V1)	FUZZYCN2 (V2)	FUZZYCN2 (V3)
haberman	17.6	2.1	17.6	15.3	9.7	9.3	16.5
iris	8.9	3.3	20.4	5.8	5.0	5.1	6.0
bupa	40.4	2.1	52.8	16.9	13.9	14.4	19.2
ecoli	51.0	8.6	71.4	20.7	12.1	10.6	11.0
pima	115.7	3.3	152.6	30.0	24.6	24.5	32.1
glass	41.4	7.2	55.2	17.5	9.4	9.5	12.0
wbc	54.0	5.9	231.4	19.3	15.4	14.2	15.5
wine	25.7	6.0	152.1	18.4	8.7	8.3	8.2
cleveland	143.4	3.8	249.1	60.2	22.5	24.0	24.2
vehicle	283.7	12.5	462.9	38.2	28.6	31.3	34.5
ion	56.4	7.7	239.4	28.2	18.7	18.6	18.0
average	76.2	5.7	155.0	24.6	15.3	15.4	17.9

TABLE IV
TRAINING ERROR

	ID3	RIPPER	W-M	FUZZYID3	FUZZYCN2 (V1)	FUZZYCN2 (V2)	FUZZYCN2 (V3)
haberman	24.1%	25.4%	24.8%	24.7%	24.8%	25.1%	22.2%
iris	2.1%	2.1%	2.7%	3.2%	4.4%	4.8%	2.7%
bupa	33.7%	39.0%	36.9%	41.3%	36.0%	36.0%	31.3%
ecoli	12.1%	17.5%	15.1%	19.5%	20.7%	14.4%	12.6%
pima	18.6%	25.4%	22.2%	27.7%	21.5%	20.9%	17.2%
glass	22.9%	31.2%	26.0%	48.2%	37.7%	34.1%	26.9%
wbc	0.4%	3.0%	1.4%	2.6%	2.6%	1.9%	1.2%
wine	0.1%	6.5%	0.0%	4.6%	3.5%	2.1%	1.1%
cleveland	1.4%	40.9%	4.1%	29.9%	27.1%	25.4%	19.6%
vehicle	13.2%	38.4%	23.9%	40.1%	30.3%	26.7%	20.7%
ion	0.6%	7.0%	1.4%	6.9%	3.2%	1.6%	1.1%
average	11.7%	21.5%	14.4%	22.6%	19.2%	17.5%	14.2%

TABLE V
TEST ERROR

	ID3	RIPPER	W-M	FUZZYID3	FUZZYCN2 (V1)	FUZZYCN2 (V2)	FUZZYCN2 (V3)
haberman	28.4%	26.5%	26.1%	29.4%	28.4%	27.1%	27.2%
iris	13.3%	12.0%	5.3%	3.3%	4.7%	5.3%	3.3%
bupa	52.4%	39.8%	42.6%	44.6%	41.8%	42.6%	43.5%
ecoli	32.1%	29.7%	19.3%	24.5%	23.8%	16.1%	14.9%
pima	43.2%	28.6%	27.2%	29.5%	27.6%	27.1%	25.8%
glass	56.8%	50.3%	34.5%	61.1%	41.4%	40.5%	34.9%
wbc	5.0%	4.4%	5.4%	3.5%	4.5%	4.1%	2.9%
wine	10.2%	13.5%	10.7%	9.5%	6.7%	5.0%	3.9%
cleveland	57.5%	46.1%	60.0%	46.4%	46.7%	47.4%	48.5%
vehicle	46.4%	50.6%	38.2%	42.3%	35.4%	33.1%	31.2%
ion	12.9%	13.7%	7.1%	12.2%	12.5%	10.5%	8.2%
average	32.6%	28.7%	25.1%	27.8%	24.9%	23.5%	22.2%

The non-fuzzy algorithms used in the experimental study have been ID3 and RIPPER. These algorithms were analyzed using the software Weka [15] with the default configuration of each algorithm. The fuzzy algorithms under the experiments were the Wang and Mendel algorithm and FuzzyID3. In this case, the algorithms were analyzed with the Xfuzzy 3 environment [16].

In order to evaluate each algorithm, a 10 fold cross validation test has been made. Tables III, IV and V shows the results of this evaluation in terms of the mean number of rules induced by each algorithm, the mean error on the training sets and the mean error on test.

The last columns show the results for three different configurations of the FuzzyCN2 algorithm. In the first configuration we have used only selectors based on the equality, that is, rules do not use linguistic hedges. The second configuration (v2) considers some linguistic hedges as selectors: *greater than*, *smaller than*, *greater than or equal to*, and *smaller than or equal to*. In the third configuration (v3) all the linguistic hedges shown in Fig. 1

TABLE VI
RESULTS FOR THE WINE DATA SET

FUZZYCN2 (V1)

```
if(at12 == Lb0 & at11 == Lb0 & at7 == Lb0) -> class = c3;
elseif(at13 == Lb0 & at1 == Lb0) -> class = c2;
elseif(at1 == Lb2 & at9 == Lb1) -> class = c1;
elseif(at13 == Lb0 & at5 == Lb0 & at2 == Lb0) -> class = c2;
elseif(at7 == Lb1 & at1 == Lb2) -> class = c1;
elseif(at10 == Lb1 & at2 == Lb0) -> class = c1;
elseif(at7 == Lb0 & at2 == Lb1 & at9 == Lb0) -> class = c3;
elseif(at7 == Lb1 & at13 == Lb1 & at6 == Lb1) -> class = c1;
elseif(at10 == Lb0 & at2 == Lb0) -> class = c2;
```

FUZZYCN2 (V2)

```
if(at1 == Lb0 & at12 > Lb0 & at10 == Lb0 & at13 <= Lb1) -> class = c2;
elseif(at12 < Lb1 & at2 > Lb0 & at7 <= Lb0 & at3 != Lb0 & at11 <= Lb1 &
       at13 <= Lb1) -> class = c3;
elseif(at13 >= Lb1 & at1 >= Lb2 & at7 != Lb0 & at4 < Lb2 & at12 != Lb0)
       -> class = c1;
elseif(at13 == Lb0 & at7 != Lb0 & at5 <= Lb0 & at10 <= Lb1) -> class = c2;
elseif(at7 > Lb0 & at13 != Lb0 & at5 < Lb2 & at1 != Lb0 & at4 <= Lb1 &
       at6 > Lb0 & at12 != Lb0) -> class = c1;
elseif(at11 != Lb0 & at2 == Lb0 & at3 < Lb2 & at10 <= Lb1) -> class = c2;
elseif(at7 == Lb0 & at10 >= Lb1 & at12 < Lb2 & at11 <= Lb1) -> class = c3;
elseif(at1 != Lb2 & at8 != Lb0) -> class = c2;
```

FUZZYCN2 (V3)

```
if(at1 <= Lb0 & at12 > Lb0 & at10 ~= Lb0 & at13 <= Lb1) -> class = c2;
elseif(at13 %= Lb2) -> class = c1;
elseif(at12 += Lb0 & at11 += Lb0) -> class = c3;
elseif(at10 += Lb0 & at1 %= Lb0 & at13 <= Lb1) -> class = c2;
elseif(at13 != Lb0 & at7 >= Lb1 & at4 <= Lb1 & at1 != Lb0 & at12 >= Lb1 &
       at6 != Lb0) -> class = c1;
elseif(at7 += Lb0 & at4 != Lb0 & at2 != Lb0 & at3 != Lb0 & at12 < Lb2 &
       at11 < Lb2 & at5 <= Lb1) -> class = c3;
elseif(at13 += Lb0 & at5 ~= Lb0 & at10 <= Lb1) -> class = c2;
elseif(at7 != Lb0 & at5 ~= Lb1 & at11 < Lb2 & at12 >= Lb1 & at6 != Lb0 &
       at13 < Lb2 & at10 <= Lb1) -> class = c1;
```

were used. In all these configurations, the value of the alpha-cut was 0.5, the star size was fixed to 5, the value of the minimum absolute support was fixed to 5 and the minimum relative support was fixed to 0.05.

Regarding the mean number of rules (Table III), the three configurations of FuzzyCN2 show similar results, although the configuration v3 seems to induce a number of rules a bit higher. Comparing these results with respect to the other algorithms, it can be seen that FuzzyCN2 generates a smaller number of rules. Only the RIPPER algorithm induces a lower number of rules. Generally, RIPPER induces a small number of rules, but the test error is higher than that generated by the rest of the algorithms.

Table V shows the test error for the different algorithms. In this study, the ID3 algorithm shows the pourer results and fuzzy algorithms obtain better results than the non-fuzzy ones. It is also worth to notice that the Wang and Mendel algorithm have quite good results on test error, but generates a very high number of rules. The proposed FuzzyCN2 algorithm obtains the better results among the algorithms under study. This is particularly true regarding the third configuration (v3). In this last case, despite of the low number of datasets in the study, the Wilcoxon´s Signed Rank Test let us to affirm with a confidence level of 95% that the FuzzyCN2 algorithm is the best of the algorithms included in this study in terms of the test error.

Table VI shows an example of the rule list induced by FuzzyCN2. The example uses the wine data set. In this case, all the instances of the data set have been considered. The description of the rule lists is written in XFL3, the specification language used in Xfuzzy. In this language, the term (*attr ~= lb*) means *attr is approximately equal to lb*; the term (*attr += lb*) means *attr is strongly equal to lb*; and the term (*attr %= lb*) means *attr is slightly equal to lb*.

VI. Conclusion

Classifiers based on rule sets use to contain conflicting rules, that is, rules covering a common space and with different conclusions. Ordered lists of rules are a good choice to avoid these conflicts. This paper proposes an algorithm to extract ordered lists of fuzzy classification rules. The algorithm is an adaptation of the well-known CN2 algorithm. The paper also proposes the use of linguistic hedges to obtain more precise and compact rules. These hedges are easily included in the algorithm by adding them into the selector list in the beam search. The experimental results show that the algorithm produces a smaller number of rules and a lower classification error than other well-known fuzzy classification algorithms, thus giving a better balance interpretability-accuracy.

References

[1] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *Journal of Artificial Intelligence Research*, vol. 4, pp. 77-90, 1996.

[2] L. Wang and J. M. Mendel, "Generation of rules by learning from examples," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, pp. 1414-1427, 1992.

[3] D. Nauck and R. Kruse, "NEFCLASS – A neuro-fuzzy approach for the classification of data," in *Proc. 1995 ACM Symposium on Applied Computing*, pp. 461-465.

[4] R. Senhadji, S. Sánchez-Solano, A. Barriga, I. Baturone, and F. J. Moreno-Velo, "NORFREA: An algorithm for non-redundant fuzzy rule extraction," in *Proc. IEEE International Conference on Systems, Man and Cybernetics*, 2002, pp. 604-608.

[5] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

[6] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains," in *Proc. 5th National Conference on Artificial Intelligence (AAAI86)*, 1986, pp. 1041-1047.

[7] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, pp. 261-283, 1989.

[8] P. Clark and R. Boswell, "Rule induction with CN2: some recent improvements," in *Proc. 5th European Working Session on Learning (EWSL-91)*, 1991, pp. 151-163.

[9] R. Weber, "Fuzzy ID3: a class of methods for automatic knowledge acquisition," in *Proc. 2nd Int. Conf. on Fuzzy Logic and Neural Networks*, 1992, pp. 265-268.

[10] T. Tani, M. Sakoda, and K. Tanaka, "Fuzzy modeling by ID3 algorithm and its application to prediction," in *Proc. IEEE International Conference on Fuzzy Systems*, 1992, pp. 923-930.

[11] M. Umano, H. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umedzu, and J. Kinoshita, "Fuzzy decision trees by Fuzzy ID3 algorithm and its application to diagnosis systems," in *Proc. IEEE International Conference on Fuzzy Systems*, 1994, pp. 2113-2118.

[12] K. J. Cios and L. M. Sztandera, "Continuous ID3 algorithm with fuzzy entropy measures," in *Proc. IEEE Int. Conf. on Fuzzy Systems*, 1992, pp. 469-476.

[13] J. van Zyl and I. Cloete, "An inductive algorithm for learning conjunctive fuzzy rules," in *Proc. 3rd IEEE International Conference on Machine Learning and Cybernetics*, 2004, pp. 4181-4187.

[14] A. Asunción and D. J. Newman,. UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, School of Information and Computer Science.

[15] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[16] F. J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, and A. Barriga, "Nuevos algoritmos de clasificación integrados en Xfuzzy." in *Actas del XIV Congreso Español de Tecnologías y Lógica Fuzzy (ESTYLF 2008)*, pp. 277-284, 2008 (in spanish).