A COMPARATIVE ANALYSIS OF PRISM AND MDTF ALGORITHMS

Madara Gasparovica and Ludmila Aleksejeva

Department of Modelling and Simulation
Institute of Information Technology
Riga Technical University
1 Kalku Str., LV-1658 Riga
Latvia
madara.gasparovica@rtu.lv, ludmila.aleksejeva@cs.rtu.lv

Abstract: *This research compares two algorithms that work with fuzzy data – the fuzzy PRISM algorithm and an algorithm for finding relevant attributes and membership functions – MDTF (merging – decision – table – first). Suitable data sets were searched for in the UCI Repository to examine the performance of both algorithms on real data. The most suitable data sets were Iris data set and Miles Per Gallon (MPG) data set. A series of experiments were carried out to compare the possibilities and working principles of each algorithm, depending on the size of the data set and the number of attributes used. As a result, recommendations for use of the algorithms are given, depending on the structure of the data set.*

Keywords: *IF-THEN fuzzy classification rules; Machine learning; Membership functions; Modular rules; Decision table; Iris data; MPG data.*

1 Introduction

Fuzzy systems that are able to extract IF-THEN rules from numerical data have been developed relatively lately. Nevertheless they have gained popularity in the recent years because of their property to describe the task issues with uncertainty that are typical of the real world. This research examines two different methods - fuzzy PRISM and an algorithm for finding relevant attributes and membership functions, MDTF. Both algorithms extract fuzzy IF-THEN rules but each algorithm uses different method. This research studies requirements of each algorithm towards real data and their capabilities by comparing both algorithms.

This paper is organized as follows. Section 2 describes in detail the methods used in this investigation. Section 3 presents experimental results as well as provides their analysis. Section 4 concludes the paper and outlines directions for further research.

2 Methods used

This section provides a description of the both studied algorithms that work with fuzzy data. The fuzzy PRISM algorithm extracts fuzzy IF-THEN rules as modular rules, that is, using value combinations of a specific attribute. Whereas MDTF algorithm searches for one or more most important attributes and extracts rules from them that can help classify new records.

2.1 The fuzzy PRISM algorithm

The fuzzy PRISM algorithm [1] differs from the classic PRISM algorithm [2] in its approach to measure the benefit of the fuzzy information for each attribute-value combination. The initial data also differ– for the fuzzy algorithm the data contain the membership function of each record to each individual attribute value (see Table 1.). It is also obvious that the training set holds more values of each attribute.

Table 1. Data set differences between PRISM and fuzzy PRISM algorithms

No.	**PRISM algorithm**					**Fuzzy PRISM algorithm**										
	A_1	A_2	A_3	A_4	*Class*	*Attribute* A_1				...	*Attribute* A_4			*Class*		
						1	2	3	4	…	1	2	3	O_1	O_2	O_3
1	2	2	1	1	O_1	0	1	0	0	…	0.8	0.2	0	1	0	0

The fuzzy inductive algorithm for learning modular rules consists of eight steps. Next we will describe the algorithm

step by step. If the training set contains instances of more than one class, then for each classification δ_k:

1. Initiate a complex $\tilde{C}$ or initial rule set, which is filled with zero values that are replaced with the established rules in the course of the algorithm execution.
2. Measure the fuzzy information gain, $I(\delta_k \mid s_i)$, of the classification δ_k for each possible selector s_i from the training set:

$$I(\delta_k \mid S_i) = \log_2\left(\frac{H(\delta_k \mid S_i)}{H(\delta_k)}\right) = \log_2\left(H(\delta_k \mid S_i)\right) - \log_2\left(H(\delta_k)\right), \tag{1}$$

where
$H(\delta_k)$ antecedent fuzzy information;
$H(\delta_k \mid S_i)$ subsequent fuzzy information defined as follows:

$$H(\delta_k \mid S_i) = \frac{\sum_{j=1}^{n} u_{\delta k}(e_j) \tau u_{si}(e_j)}{\sum_{j=1}^{n} u_{si}(e_j)}, \quad H(\delta_k) = \frac{1}{n}\sum_{j=1}^{n} u_{\delta k}(e_j), \tag{2}$$

where
n is the size of the training set;
e_j is the j-th instance in the training set;
$u_{\delta k}(e_j)$ is a class membership value specifying the degree to which instance e_j belongs to event δ_k.

3. Choose a selector s_i for which $I(\delta_k \mid s_i)$ is maximum.
4. Add selector s_i to $\tilde{C}$ and calculate $B(\delta_k \mid \tilde{C})$ which is defined as follows:

$$B(\delta_k \mid \tilde{C}) = \frac{\sum_{j=1}^{n} u_{\tilde{C}}(e_j) \tau u_{\delta k i}(e_j)}{\sum_{j=1}^{n} u_{\tilde{C}}(e_j)}, \tag{3}$$

where
$\tilde{C}$ - condition of the induced rule;
δ_k - conclusion of the induced rule.

5. If $B(\delta_k \mid \tilde{C})$ is above the predefined truth level β, then execute step 6; otherwise, create a new training set in which each instance is α - covered by the selector s_i, and go to step 2.
6. Form the rule „IF $\tilde{C}$ THEN δ_k".
7. Remove all instances α – covered by the rule „IF $\tilde{C}$ THEN δ_k" from the original training set.
8. Repeat step 1 to step 7 until all instances α-belonging to class δ_k in the original training set have been removed.

When the rules for one classification have been induced, the training set is restored to its initial state and the algorithm is applied again to induce a set of rules covering the next classification [1].

2.2 Algorithm for finding relevant attributes and membership functions

The algorithm for finding relevant attributes and membership function – MDTF consists of three main stages – find relevant attributes, then build initial membership functions, and at the end derive decision rules [3]. Further the algorithm is described in more detail.

Stage I. Find relevant attributes – in this stage it is necessary to find information about attributes to understand which of them are relevant.

1. Sort values of each attribute A_i which appear in training instances in ascending order.
2. For each attribute value, A_{ij}, count how many instances belong to the same classes.

3. Sum how many instances of each attribute values A_{ij} belong to only one class.
4. Calculate the fitness degree f_i of each attribute. Several approaches can be used for that purpose, for example, the first method is based on average fitness (Equation 4):

$$f_i = \frac{t_i}{n}, \tag{4}$$

where

t_i – the number of instances of attribute A_i that belong to only one class,

n – the total number of training instances.

The second method uses the concept of entropy:

$$f_i = \left\{ -\frac{1}{q_i} \sum_{j=1}^{q_i} \sum_{k=1}^{p} \left[\left(\frac{D_{ijk}}{\sum_{k=1}^{p} D_{ijk}} \right) \log_p \left(\frac{D_{ijk}}{\sum_{k=1}^{p} D_{ijk}} \right) \right] \right\}, \tag{5}$$

where

D_{ijk} – number of instances that fall within attribute value A_{ij};

p – total number of output classes;

q_i – number of values of A_i.

5. Sort the attributes in ascending order of fitness degrees;
6. Select relevant attributes using procedure (6) and threshold β_1. An attribute (or attributes) with higher fitness degree can be taken as relevant.

PROCEDURE Select_Relevant_Attributes:
$l = 1,\ Total\_E = 1$
DO WHILE $l \le m$
$Total\_E = Total\_E \cdot \left(1 - f_l{}'\right)$
IF $Total\_E \le \beta_1$ THEN exit
ENDIF
$l = l + 1$
ENDDO

(6)

Stage II. Build initial membership functions – in this stage a triangular membership function needs to be constructed using information about attributes.

7. Find the initial default group number G of each relevant attribute as $G = 1 + 3.3 \log n$;
8. Find the range of each attribute $R_i = \max(A_i') - \min(A_i')$;
9. Find the group interval of each attribute $H_i = R_i / (G - 1)$;
10. Extend the possible minimum attribute value as $V_i = \min(A_i') - H_i / 2$;
11. Divide the possible range of each attribute into G groups;
12. Find the typical points of triangular membership function (see Fig.1) a_{ij}, b_{ij} and c_{ij} for each initial membership function (Equation 7):

$$b_{ij} = \sum_{s=1}^{r_{ij}} \frac{A_{ij}'(I_{ijs})}{r_{ij}}, \quad a_{ij} = b_{i(j-1)} \quad c_{ij} = b_{i(j+1)}, \tag{7}$$

where

$A_{ij}'(I_{ijs})$ represents the attribute value of instance I_{ijs} in A_{ij}';

r_{ij} is the number of instances that fall into attribute value range A_{ij}'.

Stage III. Derive decision rules – in this stage decision tables are constructed, simplified and rebuilt to derive decision rules.

13. Construct an initial decision table using relevant attributes ranges;
14. Simplify the initial decision table by eliminating redundant and unnecessary table cells;
15. Rebuild membership functions using initial decision table cell merging operations;
16. Derive decision rules from the modified decision table.

3 Experimental part

This section provides the experiments carried out in the research and the results obtained, as well as conclusions about the potential of both algorithms. The data used in the experiments were difficult to find and required a lot of work to sort out the fittest real life data sets.

3.1 Data sets descriptions

While studying the algorithms, the MDTF algorithm has shown the highest expectations towards parameters of data sets used in the experiments, therefore data sets that could be used in the experiments should comply with the requirements of this algorithm.

The MDTF algorithm needs numeric data. To compensate the amount of work put into the experiments with information gain, the maximum number of attributes should be ten. The data set cannot hold the predefined small numeric values. From 198 data sets contained in UCI Repository [4], 10 were chosen (see Table 2.) for tests with the MDTF algorithm; and from these only few most perspective were chosen for further research.

Table 2. Tested data sets

Name from UCI	Instances	Attributes in data set	Used attributes (real)	1st relevant	2nd relevant	Fitness degree *<0.1*	Perspective
Blood Transfusion	748	5	4	0.08	0.033	0.89	No
Pima Indians Diabetes	768	8	8	0.29	0.23	0.55	No
Ecoli	336	8	7	0.28	0.18	0.59	No
Gamma Telescope	19020	11	10	1	1	0	Yes/No
Horse Colic	368	27	5	0.34	0.28	0.48	No
Forest Fires	517	13	3	0.09	0.008	0.903	No
Iris	150	4	4	0.78	0.7	0.066	Yes
Ionosphere	351	34	7	0.63	0.24	0.28	Yes/No
Haberman's Survival	306	6	6	0.062	0.045	0.89	No
Auto MPG	392	8	5	0.97	0.62	0.03	Yes

As can be seen from Table 2., only two data sets fully satisfy the requirements, and two other sets show results that would be worth testing. Gamma Telescope data set acquires high quality result and it is shown that one attribute should accurately classify all of the records.

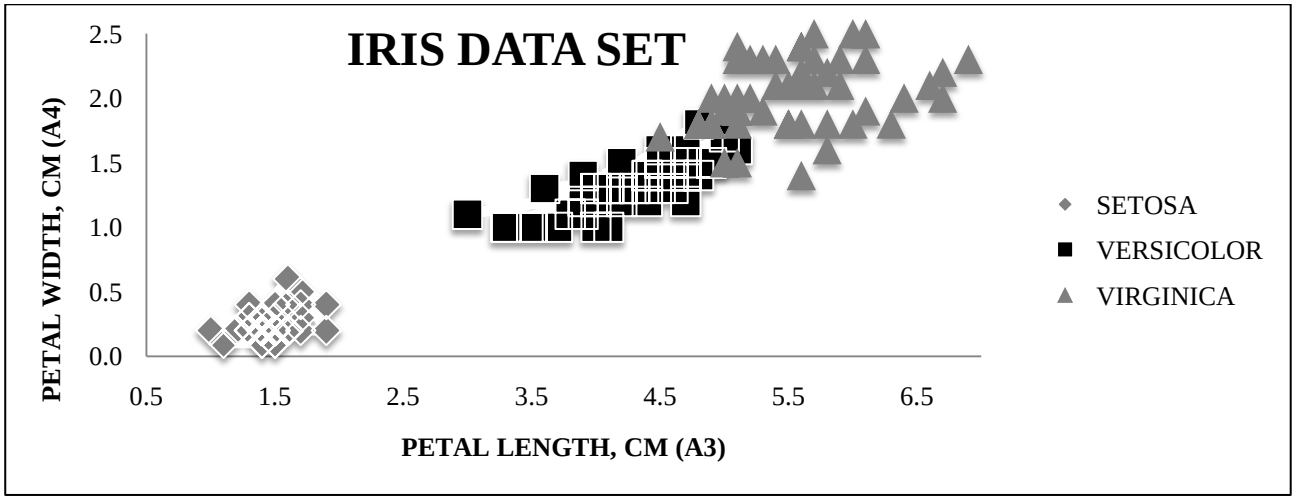

Fig. 1. Iris data set relevant attributes

It can also be seen from Table 2. that the Iris data set has two relevant attributes; Fig. 1. shows their representation in classes. It is obvious that one class – SETOSA is easily separable, whereas values of VERSICOLOUR and VIRGINICA overlap. Both algorithms studied in this paper are designed for problematic cases like that.

Another data set that satisfies the requirements is Miles Per Gallon (MPG) data set. In this data set one attribute is relevant. The representation of this data set according to the relevant attribute is shown in Fig. 2. It is obvious that the classes are more mixed together and overlap more but it is due to the specific character of the data set.

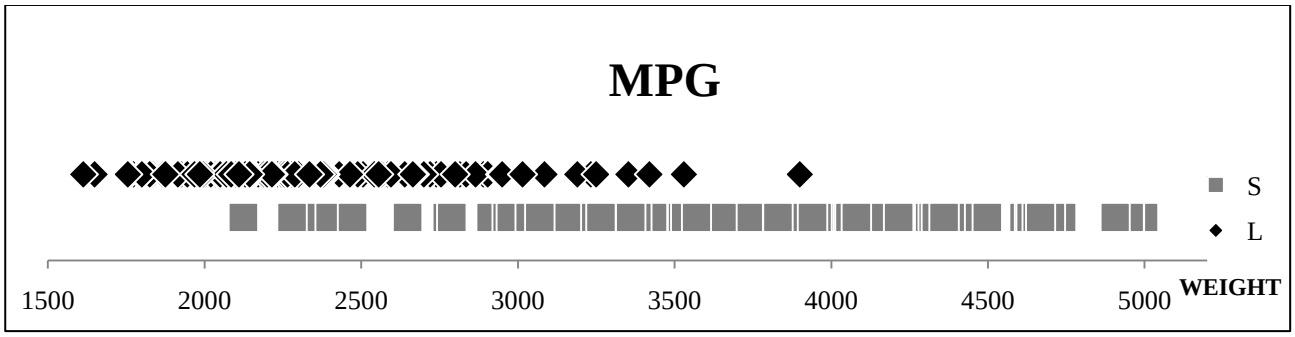

Fig. 2. Car MPG data set relevant attribute

For each record the 'Weight' attribute takes up values in a broad range of the scale but it shows that lighter cars belong to the class 'L' more, that means they are fuel-efficient, whereas heavier cars tend to belong to the 'S' class, which means they are not fuel-efficient.

3.2 Experiments with Iris data set

The results of the experiments with the Iris data set are summarized in Table 3. The experiments were carried out varying sizes of the test and training sets to establish the impact of the size on the result. As can be seen from Table 3., the first six experiments were conducted with different numbers of records. From these results it can be concluded that the classic division of 70% for training and 30% for testing is enough to obtain high class results. To see the effect of contents of the training set on the results and to examine possibilities to raise efficiency, it is useful to include three-fold cross-validation into experiment plan. In the first series of the experiments with the MDTF algorithm these experiments showed good results, therefore they will be included in further experiments [5]. As can be seen from Table 3., the best result (the smallest number of incorrectly classified records, 0) is achieved in one of the experiments that used cross-validation.

Table 3. Experiments with Iris data set

	Training data set	Testing data set	Incorrectly classified examples	Accuracy	Classification error	Number of rules	Comments
MDTF algorithm	105	45	2	0.94	0.06	11	Classical (70:30)
	102(96)	48(54)	0	0.96	0.04	8	Three-fold cross validation 150 inst.
	75	75	4	0.96	0.04	8	Training: testing (50:50)
	135	15	1	0.93	0.07	10	Larger training, smaller testing
	105	15	1	0.93	0.07	8	Smaller training data set
	78(81)	42(39)	1	0.96	0.04	8	Three-fold cross validation 120 inst.
PRISM	105	45	3+3[1]	0.87	0.13	6	Classical (70:30)
	96(102)	54(48)	3	0.89	0.11	18	Three-fold cross validation (6 intervals)
Fuzzy PRISM	105	45	0	1.00	0.0	6	Classical (6 intervals)
	105	45	3	0.93	0.07	8	Classical (3 intervals)
	96(102)	54(48)	4	0.88	0.12	6	Three-fold cross validation (3 interv)

[1] not classified at all

Experiments with the classical PRISM algorithm show the expected order, that is, the three-fold cross-validation shows better results than the classic division (70:30) but the number of the obtained rules has risen as well. It is a good factor on the one hand and a bad factor on the other. The positive point is that the more rules are obtained, the more accurate the classification of new records is, but the negative aspect is that it considerably increases the amount of work required for the classification of new records.

The third block of the experiments shows the results of the fuzzy PRISM algorithm. As it was expected, the fuzzy prism outperformed the classical algorithm, because this algorithm is more suitable for cases of fuzzy classes. During the experiment another interesting discovery was made – the larger the number of intervals the original values are split into, the more accurate the result which can be obtained using the fuzzy PRISM algorithm. It can be observed when the

results of experiments, where the initial scale was split into three or six intervals, are compared. Overall, both algorithms that use membership functions perform better, but comparison of the number of the obtained rules gives preference to the fuzzy PRISM algorithm.

3.3 Experiments with MPG data set

When using the MPG data set for experiments it is important to emphasize that the data set didn't have the predefined classes, they were derived from the attribute that contained information about the number of miles that can be driven using one gallon of petrol. First, the attribute was split into three classes but the division into two classes showed considerable improvement, therefore the division into two classes was used for further experiments. The results of the experiments are shown in Table 4. Comparing the results of the classic division (70:30) and the three-fold cross-validation, it shows, that cross-validation slightly improves the results. It should be stressed that this data set had only one relevant attribute, due to that, the number of the IF-THEN classification rules obtained by algorithms is so small.

More than two proposed experiments were carried out using classic PRISM. At first, using MDTF algorithm the first attribute showed contribution to the classification of so small extent that it was decided to exclude it from further research, which led to negative results, therefore another experiment was carried out using the whole data set with all attributes. It showed improvement in the number of incorrectly classified records reducing it, as well as in accuracy. However it leads to a higher number of rules that have to be examined while classifying new records.

Table 4. Experiments with MPG data set

	Training DS	Testing DS	Incorrectly classified examples	Accuracy	Classification error	Number of rules	Comments
MDTF algorithm	275	117	27	0.761	0.239	3	5 attributes, 3 classes 70:30
	275	117	15	0.872	0.128	2	5 attributes, 2 classes 70:30
	262(260)	130(132)	12	0.880	0.120	2	Three-fold cross validation (5 attributes, 2 classes
PRISM	275	117	25+5[1]	0.744	0.256	28	4 attributes, 2 classes 70:30
	275	117	19+3[1]	0.811	0.189	36	5 attributes, 2 classes 70:30
	275	117	72(11)[2]	0.385	0.615	36	5 attributes, 2 classes 70:30
	275	117	12	0.897	0.103	36 (27 used)	5 attributes, 2 classes 70:30, using rules importance
	262(260)	130(132)	31+3[1]	0.740	0.260	36	Three-fold cross validation 5 attributes, 2 classes
Fuzzy PRISM	275	117	11	0.906	0.094	15	5 attributes, 2 classes 70:30
	262(260)	130(132)	8	0.918	0.082	16	Three-fold cross validation

[1] classified incorrectly +not classified at all; [2]72 instances classified in both classes

In the data set of such specific character that holds mixed values (see Fig. 2.), the algorithms obtain rules that overlap, which means that two different rules map the same record to different classes. To avoid this problem, it is valuable to evaluate rules while constructing them because if the created rule correctly classifies two out of ten records (if these ten records had the same values, but different classes) classifying the rest incorrectly, this rule is not worth including into the set of newly created rules, since it is obvious that the same rule will be created for the other class and it will correctly classify eight records out of ten. Of course, this approach means classification error but, as the results show, it is prospective; besides, it can be seen that three-fold cross-validation does not improve the results. A look at the number of classification rules indicates that it is large and exceeds the number of rules obtained by using MDTF algorithm almost 20 times.

Using fuzzy PRISM has slightly improved the results as it was expected; besides, the number of rules is only 15, which is better result than with PRISM – 28 to 36. If we compare the number of rules in MDTF and fuzzy Prism, then MDTF has only two rules, which is six times smaller than fuzzy PRISM – 15. But if the accuracy is compared, the fuzzy PRISM algorithm outperforms MDTF because the results are better than those obtained with MDTF (seeTable 4.), accordingly 0.88 for MDTF and 0.918 for fuzzy PRISM. The question remains whether a 0.038 increase in accuracy is adequate for 12 more rules.

4 Conclusion

The authors have thoroughly examined the algorithms during the research and writing of this paper. The main difficulty was to find appropriate data sets that suited both algorithms. A lot of work was put in using a shell module for the first

steps of the algorithms created in MS EXCEL. However, as a result, two different data sets that best satisfied the requirements of the MDTF algorithm were obtained.
The following conclusions were drawn:

1. Even if an attribute shows a very small relevance, it cannot be excluded from the following experiments (if the algorithm doesn't state the contrary) because each algorithm uses its own classification methods.
2. It is necessary to evaluate whether in data sets, where one attribute is crucial, the work invested in calculations to obtain more rules is adequate for the increase in accuracy.
3. Recommendations for use of the explored algorithms depending on data set structure and parameters are listed in Table 5. Given that the fuzzy PRISM algorithm is more universal, it is not surprising that it gives often better results than MDTF.

The MDTF algorithm is more accurate but it is not suited for all data sets: if the number of relevant attributes is more than two, the calculations become more complicated. The fuzzy PRISM algorithm is more universal since it is suited to continuous as well as categorical data, but the process of calculating membership functions is complicated.

Table 5. Recommendations for practical use of the algorithms

Precondition	Algorithm	Comments
If – only one relevant attribute	MDTF	Good results within the **shortest time**, small number of rules
If – only one relevant attribute	Fuzzy PRISM	**The most accurate result**, large number of rules
If there are two or more relevant attributes	Fuzzy PRISM	The overall number of attributes is small
If the initial data is both numeric and continuous	MDTF	Good results within the **shortest time**
If the initial data is categorical	Fuzzy PRISM	Good results within the **shortest time**

If in the future the MDTF algorithm could be used on categorical data, it would definitely become a universally usable algorithm. It can be the theme of future research. The research should be also continued using other data sets and for examined algorithms with the modified version of MDTF – MMFF (*merging – membership – functions – first)* [6] to explore its benefits, as well as investigating other fuzzy algorithms.

Acknowledgement: Thanks to Professor Arkady Borisov, Dr.habil.sc.comp.(Riga Technical University) for help and support.

References:

[1] Wang C. H., Liu J. F., Hong T. P., Tseng S.S., A fuzzy inductive learning strategy for modular rules, Fuzzy sets and Systems, Vol. 103, 1999, pp. 91–105.
[2] Cendrowska J., PRISM: an algorithm for inducing modular rules, Internat. J. Man – Machine Studies, Vol. 27, 1987, pp. 349-370
[3] Hong T. P., Chen J. B., Finding relevant attributes and membership functions, Fuzzy Sets and Systems, Vol. 103, No. 3, 1999, pp. 389-404.
[4] Asuncion A., Newman, D.J., UCI Machine Learning Repository Irvine, CA:University of California, School of Information and Computer Science (2007). Link http://www.ics.uci.edu/~mlearn/MLRepository.html
[5] Gasparovica M., Aleksejeva L., A study on the behaviour of the algorithm for finding relevant attributes and membership functions, Scientific Proceedings of Riga Technical University, Vol.40, 2009, pp.75-80.
[6] Hong T. P., Chen J. B., Processing individual fuzzy attributes for fuzzy rule induction, Fuzzy Sets and Systems, Vol. 112, 2000, pp. 127-140.