# Improving Fuzzy Rule Based Classification Systems in Big Data via Support-based Filtering

Luis Íñiguez
*Dept. of Computer Science and A.I.*
*University of Granada, Spain*
Email: luisiniguez@correo.ugr.es

Mikel Galar
*Department of Automatic and Computation*
*Public University of Navarre, Spain*
Email: mikel.galar@unavarra.es

Alberto Fernández
*Dept. of Computer Science and A.I.*
*University of Granada, Spain*
Email: alberto@decsai.ugr.es

***Abstract*—Fuzzy Rule Based Classification Systems have the benefit of making possible to understand the decision of the classifier. Additionally, they have shown to be robust to solve complex problems. When these capabilities are applied to the context of Big Data, the benefits get multiplied.**

**Therefore, to achieve the highest advantages of Fuzzy Rule Based Classification Systems, the output model must be both interpretable and accurate. The former is achieved by using fuzzy linguistic labels, that are related to human understanding. The latter is achieved by means of robust fuzzy rules, which are identified by means of a component known as *"fuzzy rule weight"*. However, obtaining these rule weights is computationally expensive, resulting on a bottle-neck when applied in Big Data problems.**

**In this work, we propose Chi-BD-SF, which stands for Chi Big Data Support Filtering. It comprises a scalable yet accurate fuzzy rule learning algorithm. It is based on the well-known Chi et al., exchanging the rule weight computation by a support metric in order to solve the conflicts between different consequent rules. In order to show the goodness of this proposal, we analyze several performance metrics, such as the quality of classification, the robustness of the rule base generated and the runtimes of the usage of traditional weights and the support of the rule. The results of our novel Chi-BD-SF approach, in contrast to related Big Data fuzzy classifiers, show that this proposal is able to outspeed the usage of rule weights also obtaining more accurate results.**

## I. Introduction

Machine Learning and Data Mining are commonly used to solve difficult tasks and to extract knowledge from complex data. For this reason, most companies and practitioners take advantage of these algorithms [1], [2] and develop new methods and alternatives to improve the understanding of data. Such is the trend of these techniques that nowadays, we live surrounded by data and artificial intelligences. Our society is generating more and more information everyday reaching a point were it is nearly impossible to handle all the information created [3]. This situation is defined by the term Big Data [4].

Therefore, many techniques have been developed to address those problems generated by this continuous flow of data. Big Data techniques are mainly oriented to solve three common issues when handling large amounts of data. First, Big Data needs to be prepared to work with huge quantities of data in an easy and scalable way. Second, it also require methods to deal with the big stream of information generated every second from multiple sources. Lastly, Big Data algorithms are required to be fast and scalable to be able to work with the continuous flow of data and solve big and complex problems in a reasonable amount of time [5].

In order to accomplish all these requirements, several paradigms have been proposed for adapting standard Machine Learning and Data Mining techniques. Among them, the MapReduce [6] framework must be excelled as one of the most successful options.

There is a special interest on developing rule base methods due to their interpretability [7]. A crucial aspect of classification systems is to know the reasons under the decision of labeling a certain example. Linguistic Fuzzy Rule Based Classification Systems (FRBCSs) generate an interpretable model that can be used to classify an example and to understand the reasons behind this decision [8]. FRBCS [9] works using a rule base composed by fuzzy antecedents with their consequents being one of the classes of the problem. The rules are usually created using a combination of pattern matching techniques and fuzzy logic methods.

The rule weight (RW) computation is a core part of the rule base generation process. It helps to extract the most relevant fuzzy rules and consequently, it points useless rules which can lead to misclassification. When extracting rules from a dataset it is common to have multiple rules with the same antecedent but different consequent (known as *conflicts*). In these cases, RWs help at determining which one of those rules should be kept in the final rule base. [10].

Many weight computation methods have been developed to improve the rule base generation process. Heuristics like Certainty Factor (CF) and Penalized Certainty Factor (PCF) are widely used and their reliability came from multiple studies [10]. These methods test every fuzzy rule of the rule base with every instance of the dataset in order to calculate the weight of the fuzzy rule. The number of required operations is high and it grows exponentially with the size and complexity of the problem

To develop a new algorithm for Big Data problems one needs to focus on making it robust and scalable. Typically there are two options in order to implement a certain algorithm into MapReduce [11]. The first one, is a local divide and conquer implementation where the data is distributed among the cluster and every machine gets a solution of the problem and the final output is the combination of all of them. This

 978-1-5090-6020-7/18/$31.00 

approach allows one to fastly implement any existing method in MapReduce, but the results obtained are approximations that depends on the degree of parallelism since the data is not considered as a whole. The other is a distributed implementation of the algorithm, where the result is an exact solution. However, these approaches are much more complex to be designed because the data is distributed among a cluster of computing nodes. Both alternatives have been applied in the design of FRBCS algorithms for Big Data, e.g., Chi-FRBCS-BigData [12] with a divide and conquer strategy and Chi-BD [13] with an exact solution.

Rule weights are an important part of a fuzzy rule, they are not only used to add more discriminant power to fuzzy rules, but they also serve as a conflict resolver in case of overlapping areas, i.e. rules with same antecedent but different consequent. However, as stated previously the cost in time required to compute the weights is excessive, not scalable and unrealizable when the dataset is too large, even for a cluster.

Our hypothesis is that the weight of a fuzzy rule is an issue when working with Big Data problems and that we could use other more efficient metrics that can be introduced into MapReduce without additional cost. The new metric must be able to solve conflicts between rules and not require an excessive amount of operations at the same time.

We suggest the standard support of a rule to deal with conflicts. Unlike the weight, obtaining the standard support require less operations and does not need to use the whole dataset iteratively. When a conflict between rules arises, the rule with the highest support is chosen as to be the most representative among them. In this work we propose Chi-BD-SF, a robust alternative approach for FRBCS in Big Data that allows the rule base generation process to be more efficient using the standard support as filtering method.

In order to show the true benefits of this novel approach based on the support, we will contrast the behavior of Chi-BD-SF versus several related FRBCS models for Big Data, namely Chi-FRBCS [14] (the sequential approach), Chi-FRBCS-BigData [12] (the approximate MapReduce implementation) and Chi-BD [13] (the exact Big Data implementation). This evaluation will be carried out in terms of several performance metrics. First, we will consider the classification accuracy and geometric mean. Second, we will analyze the interpretability of the system by means of the rule base size. Finally, the scalability will be contrasted using the elapsed training times.

This work is organized as follows. First, Section II introduce the theoretical framework of Big Data. Then, Section III explain the basis of FRBCS, Chi-FRBCS algorithm and two Big Data implementations, Chi-FRBCS-Big Data and Chi-BD. Followed by Section IV which describes this work proposal, Chi-BD-SF algorithm. Section V shows the experimental framework, the algorithms used and their configuration, the datasets, the measures considered and the computational infrastructure. Section VI presents the experimental results and the analysis, being divided into two subsections; on the one hand, an experimentation with standard datasets to test in a single machine; on the other hand, the experimentation with big datasets in a cluster of computing nodes. Last Section VII presents the final conclusions and future work.

## II. Big Data environment

Big Data is a term referring to problems where one should deal with large amounts of information. Getting an overwhelming input data involve the processes of storing and working with it too. The complexity of these processes is beyond the capabilities of a single machine. A cluster with multiple interconnected machines is required. Big Data technologies are oriented to work with a cluster of computers, distribute the information and process the information in parallel. This section is divided in two parts. First, Section II-A explains Hadoop and MapReduce programming paradigm, technologies used in Big Data. Then, Apache Spark is presented in Section II-B, a computation engine which improves it and solves some of its deficiencies.

### A. Hadoop and MapReduce

Hadoop is an open source Big Data framework for distributed programming [15]. It is oriented to be used from one machine to a whole cluster of computers, being able to store large amounts of data and obtain a great computation performance. Hadoop offers as main features accessibility to the data no matter if it is in a single machine or stored in the cloud, robustness and scalability to add new machines to the infrastructure and in a simple and fault tolerant way. To work with large amounts of data, Hadoop needs to store all the information in an effective and simple way to be used by programs. The information is stored using the Hadoop Distributed File System (HDFS) [16], a file system specifically designed to distribute all the information across a cluster of computing nodes and keep it secure and accessible.

The programming paradigm used by Hadoop is MapReduce [6], which enables to execute computational expensive programs in parallel to reduce the time needed. The origin of MapReduce came from two famous functional operators, map and reduce. Consequently all the action of a MapReduce algorithm need to have a map phase and a reduce phase. Map is a functional operator that take data and transform it in order to create new information, structures, etc.. The reduce phase is when all the transformed data from the previous map phase is taken and used to obtain the desired result from the MapReduce operation by aggregating all the previous information. MapReduce is a powerful tool to create robust parallel algorithms but not all methods can be implemented in MapReduce. To properly implement an algorithm in MapReduce, it should be possible to adapt the original algorithms to the properties of this framework. All the iterative method have issues with MapReduce paradigm, but some alternatives to Hadoop has been implemented to solve this problem such as Spark.

### B. Spark

Spark is a cluster computing platform designed to be fast and general purpose. It extends the popular MapReduce

model to efficiently support more types of computations. At its core, Spark is a *computational engine* that is responsible for scheduling, distributing, and monitoring applications consisting of many computational tasks across many worker machines, or a computing cluster [17]. Due to Spark engine, it supports multiple packages such as SQL, machine learning or stream processing.

The key of Spark came from his unique data structure and the lazy evaluation strategy. The Resilient Distributed Dataset (RDD) is a data structure which divide and distribute the data into different partitions. The RDDs are immutable, persistent and can cast MapReduce operations (among others), ideal to develop machine learning and data mining algorithms. Spark works in a lazy way. When a transform operation, like a map function, is called, it does not execute it, Spark will save it and wait. When an action operation is called, like the reduce function, then spark will execute it and all the saved transformation operations before the action. This strategy gives Spark the big picture of what the application desires to do and optimize the process. As a result of combining this two features, the incompatibilities between MapReduce paradigm and iterative method are solved when working with Spark [17].

## III. Fuzzy Rule Base Classification Systems

In this section we introduce the fundamentals of FRBCS (Section III-A), then we include the description of the Chi fuzzy rule learning algorithm (Section III-B). Finally, we focus on the Big Data implementations of the fuzzy rule learning algorithm, namely Chi-FRBCS-Big Data and Chi-BD (Section III-C).

### A. Basic components of FRBCS

An FRBCS uses fuzzy sets and rules to represent the knowledge of a specific area and model the relationship between variables in order to use them to classify future instances. They are used as classifier because they contribute with a fuzzy rule base, which is interpretable by humans [18], making them desirable for problem related experts. The main components of every FRBCS are the Knowledge Base (KB) and the Fuzzy Reasoning Method (FRM).

The KB is compounded of the rule base, a database and the membership functions, used to model linguistic labels and store them. To generate the rule base, a fuzzy rule learning algorithm is required. Taking a training dataset where every instance has the form $x_p = (x_{p1}, ..., x_{pn})$ $p = \{1, ..., P\}$ where $x_{pi}$ is the value of the i-th attribute and $i = \{1, 2, ..., n\}$ is the p-th instance of the training dataset. Every instance belongs to a specific class $y_p \in C = \{C_1, C_2, ..., C_m\}$ where $m$ is the total number of classes. Every instance is transformed into a fuzzy rule using fuzzy variables created by fuzzy sets like Fig. 1 and a membership function with the following structure:

$$\begin{aligned} &Rule\ R_j : If\ x_1\ is\ A_{j1} and \dots and\ x_n\ is\ A_{jn} \\ &Then\ Class\ C_j\ with\ RW_j \end{aligned} \tag{1}$$

The RW is the weight of the rule, it is calculated using heuristic methods like CF (2) or PCF (3), where the membership function (4) is used [10].

$$RW_j = CF = \frac{\sum\limits_{x_p \in ClassC_j} \mu_{A_j}(x_p)}{\sum\limits_{p=1}^{P} \mu_{A_j}(x_p)} \tag{2}$$

$$RW_j = PCF = \frac{\sum\limits_{x_p \in ClassC_j} \mu_{A_j}(x_p) - \sum\limits_{x_p \notin ClassC_j} \mu_{A_j}(x_p)}{\sum\limits_{p=1}^{P} \mu_{A_j}(x_p)} \tag{3}$$

$$\mu A_j = \prod_{i=i}^{n} \mu_{A_{ji}}(x_{pi}) \tag{4}$$

Once the KB of the FRBCS has been built, one can classify new examples following its FRM. Given an unclassified example, the FRM uses the KB to compare it with every fuzzy rule and takes the class of the winning rule [19], using the coverage degree (5) to decide the class of the example by selecting the class of the rule achieving the largest degree.

$$b_j(x_p) = \mu_{A_j}(x_p) RW_j \tag{5}$$

### B. Chi-FRBCS

One of the most widely known FRBCS algorithms is Chi-FRBCS [14]. To generate the rule base the algorithm makes the following steps:

- First of all it transforms the attributes of the problem into fuzzy variables. All the variables are equally transformed using triangular fuzzy sets (Fig. 1).
- Then, it takes the instances of the training datatset and generates a set of fuzzy rules using the fuzzy variables.
- In order to solve conflicts between rules with same antecedent but different consequent the RWs are computed using methods like the CF (2) or the PCF (3) and the one with the largest weight is kept.

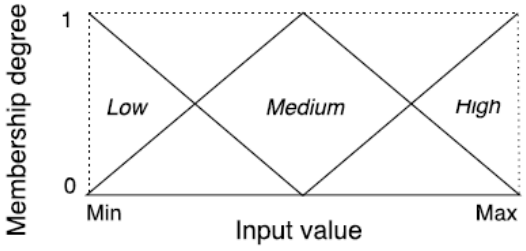


Fig. 1. Fuzzy variable with triangular fuzzy sets

### C. Chi-FRBCS-Big Data and Chi-BD

In order to use Chi-FRBCS algorithm on a cluster with large datasets, MapReduce implementations of Chi algorithm must be considered. The development of Chi-FRBCS on MapReduce has two alternatives, the first one is the local version, where the data is distributed and several local Chi classifiers are learned. As a result a local approximation is obtained, which depends on the degree of parallelism. The second one is a global version, which recovers the original Chi algorithms and obtains exactly the same result, but being designed to deal with big datasets. Both approaches make use of MapReduce paradigm in order to develop the distributed algorithms [11].

Chi-FRBCS-Big Data [12] uses the distribution of the training dataset into its benefit creating multiple rule bases, one per mapper, and merges them into a final rule base. A Big Data implementation that can improve its execution time as more mappers are added. However, the quality of the final classifier gets degraded because each data chunk of data becomes smaller and less representative.

Chi-BD [13] is a more recent Big Data implementation of Chi-FRBCS, unlike Chi-FRBCS-Big Data, this algorithm shares a candidate rule base, a rule base with fuzzy rules with same antecedent and different consequent, with all the mappers in order to get the same results as using the original Chi-FRBCS algorithm. It also speed-ups the process splitting the fuzzy rules and storing the common antecedent parts to pre-compute the majority of RW calculus operations.

## IV. Chi-BD-SF: An alternative to the usage of rule weights in FRBCSs for Big Data

Current FRBCS implementations of Chi for Big Data present several issues. In Chi-FRBCS-BigData, the problem is that, in order to alleviate the RWs computation cost, several rule bases are created with different chunks of data, but each rule base will not have a general view of the problem and hence, some RWs may not properly represent the problem. On the other hand, Chi-BD avoids this point by learning the same RWs as Chi-FRBCS in a distributed way, as stated in Section III-C. However, this leads to a not totally scalable method due to the bottle-neck being in the RW computation, which is $O(n * m)$, $n$ being the number of examples and $m$ being the number of rules.

An ideal implementation needs to be fast and accurate but we have previously observed, the calculus of the weight is expensive and the main issue in the MapReduce implementation.

In order to make the learning of Chi algorithm faster, we propose an alternative approach, named as Chi Big Data Support Filtering (Chi-BD-SF). This method omits the weight heuristic methods like CF (2) or PCF (3) which are computationally expensive. However, we must take into account that although discarding RWs solves the computational complexity of the algorithm, there is no way of knowing which fuzzy rules are reliable and which not. And even more important, there will be conflicts between rules with same antecedent but different consequent that cannot be determined.

To solve this crucial problem, this algorithm calculates the standard support of every fuzzy rule as decision making method. The rule with the highest support is chosen since it is the most probable class of an example in its region. We must point out that the fuzzy support can not be considered, as it suffers from the same computational complexity problems as the RW computation.

Applying this heuristic, the rule base generation stage becomes much faster due to the lineal computational complexity of the support. Fuzzy rules are selected using their support but this metric is not used for classification (that is, all rules have weight equal to one). Hence, the FRM way to labeling an example to a certain class is using the membership degree (4) and the winning rule method [19].

## V. Experimental framework

In this section we present the experimental framework used. Therefore, we enumeate the methods and their configuration, metrics, datasets and infrastructure.

As presented in Section III, we will use Chi-FRBCS Big Data implementations and the alternative algorithm purposed. All of them were developed in Scala using Apache Spark, Section II-B. In summary, the methods considered in our experimental study are the following.

- Chi-FRBCS: Original Chi-FRBCS algorithm, it does not use any parallelism.
- Chi-NE: Chi-FRBCS-Big Data algorithm, its name is an abbreviation from "Chi-No Exact" due to the approximated results it obtains compared to the original Chi-FRBCS algorithm.
- Chi-BD: Exact Chi-FRBCS optimized Big Data implementation. It is only used with Big Data datasets to speed-up the experimentation process.
- Chi-BD-SF: Our proposed algorithm as a parallel version, i.e. it provides a global version where the standard support is computed considering all the examples.

In order to study the FRBCS algorithms it is necessary to set the corresponding configuration. The parameters for each classification algorithm are presented in Table I. Notice that all of them are common for every algorithm except for Chi-BD which needs two extra parameters to accelerate the rule computation process.

In this study we will use two different case studies to extract well-founded conclusions from our experiments: standard classification datasets (to be run on a single machine) and big datasets (to be run on a cluster of counting nodes) [20]. The former ones are presented in Table II, whereas the latter ones are presented in Table III. All the datasets are two-class problems in accordance to the previous experiments with Chi algorithms in Big Data problems [12], [13].

In order to compare the different algorithms we will consider the quality of the classifier, measured by the classification accuracy (8) and the geometric mean (9) (which helps in better analyzing the classification of both classes). In order to obtain the classification performance we will use a confusion matrix, Table IV.

TABLE I
FRBCS CONFIGURATION

| | |
|---|---|
| Number of labels in local: | 3 and 5 fuzzy sets |
| Number of labels in Big Data: | 3 fuzzy sets |
| T-norm: | Product |
| Use of PCF rule weight: | Local |
| Use of CF rule weight: | Local and Big Data |
| Use of weightless fuzzy rules: | Local and Big Data |
| Fuzzy reasoning method: | Winning rule |
| Chi-BD specific parameters | |
| Number of rule splits: | 4 |
| Minimum frequent occurrences: | 10 |

TABLE II
STANDARD TWO-CLASS DATASETS

| Dataset | Instances | Real | Integer | Nominal | Total |
|---|---|---|---|---|---|
| Banana | 4,300 | 2 | 0 | 0 | 2 |
| Magic | 19,020 | 10 | 0 | 0 | 10 |
| Phoneme | 5,404 | 5 | 0 | 0 | 5 |
| Pima | 768 | 8 | 0 | 0 | 8 |
| Ring | 7,400 | 20 | 0 | 0 | 20 |
| Skin | 245,157 | 0 | 3 | 0 | 3 |
| Spambase | 4,597 | 57 | 0 | 0 | 57 |
| Susy-1% | 50,103 | 18 | 0 | 0 | 18 |
| TwoNorm | 7,400 | 20 | 0 | 0 | 20 |
| Wdbc | 560 | 30 | 0 | 0 | 30 |

TABLE III
BIG DATA TWO-CLASS DATASETS

| Dataset | Instances | Real | Integer | Nominal | Total |
|---|---|---|---|---|---|
| Covtype_1 | 581,012 | 10 | 0 | 44 | 54 |
| ECBDL_14-0.6 | 600,000 | 60 | 0 | 30 | 90 |
| Poker_0 | 1,025,010 | 0 | 10 | 0 | 10 |
| Susy | 5,000,000 | 28 | 0 | 0 | 28 |

TABLE IV
CONFUSION MATRIX

| | Positive prediction | Negative prediction |
|---|---|---|
| Positive class | True positive (TP) | False negative (FN) |
| Negative class | False positive (FP) | True negative (TN) |

$$TP_{rate} = \frac{TP}{(TP + FN)} \tag{6}$$

$$TN_{rate} = \frac{TN}{(TN + FP)} \tag{7}$$

$$Accuracy = \frac{TP + TN}{TN + FP + TN + FN} \tag{8}$$

$$GM = \sqrt{TP_{rate} * TN_{rate}} \tag{9}$$

To compare the robustness of the rule base generated by the weight heuristic, we will use the rule base size which will be analyzed in combination with the quality as classifier.

Finally, we will also take into account the fuzzy rule base generation runtime of each algorithm. In this way we could analyze in some way the cost related to the computation of the RW in the learning stage.

In this study we have used two different infrastructures to perform the studies, a cluster for Big Data datasets and a local machine for the executions with standard datasets.

Local machine infrastructure:

- CPU: Intel(R) core(TM) i5-4210M CPU 2'6GHz (4 cores)
- Main memory: 4GB
- OS: Ubuntu 14.04 LTS

Cluster infrastructure:

- Number of nodes: 12
- CPU: Intel(R) Core(TM) i7-4930K CPU 3.40GHz (6 cores)
- Main memory: 64GB
- Storage: 1x2 TB (root + HDFS)
- Connection: Gigabit Ethernet
- OS: CentOS 6.8
- Hadoop: CDH 5.10.0

## VI. EXPERIMENTAL RESULTS

In this section we present the experimental results of the study. It is divided into two parts. First, the results obtained using standard datasets with a local machine (Section VI-A). Second, the results using a cluster of computers and Big Data datasets (Section VI-B). The analysis of the results is included in each part.

### A. Results over the standard classification datasets

In this subsection we present the results using standard datasets. Tables V and VI shows the summary of all the results using different algorithms and methods, separated by the number of fuzzy sets used.

Table V shows an interesting and surprising behavior for Chi-BD-SF versus the remaining Chi approaches. In this case, the use of support filtering instead of the common RW provides better results in terms of performance. The accuracy is slightly better than the rest of methods but the results with the geometric mean shows that the usage of the RW tends to benefit a certain class.

Looking at the rule base size results it is clear that PCF method gets the smallest rule base, since it removes rules having negative weights. Comparing PCF and CF the only difference between both methods is the rule base size, there is not a significant improvement on the classification quality when using PCF despite of the reduced rule base.

Comparing the time employed to generate the rule base, it is clear the difference between the methods using weighted fuzzy rules and the weightless one. Chi-FRBCS algorithm obtains good classification results but the time needed to obtain them is high. Taking a look at Chi-NE results, the rule base generation time descend when using a higher level of parallelism but the lose of classification quality is excessive. Chi-BD-SF has the best results in rule base generation time and classification.

As expected, using the standard support instead of the weight makes the algorithm faster and solves the rule base generation time problem. Furthermore, and as pointed out previously, Chi-BD-SF obtains the best results in classifications, which is unexpected. We consider that the granularity of the FRBCS is responsible of this issue. Using a low granularity makes a fuzzy rule space representation wider, in consequence, a few well positioned instances of a class have more impact than the rest of the instances, even when they are more numerous. For this reason we also use 5 fuzzy sets to test this hypothesis due to make more specific fuzzy rules.

When using a higher number of fuzzy sets, the differences in rule base generation time between Chi-BD-SF and the rest of methods is greatly increased since more rules are generated. While weighted fuzzy rule methods increments their rule base generation time, Chi-BD-SF execution time is not altered. Chi-FRBCS algorithm obtains an improvement at accuracy remaining almost equal to Chi-BD-SF but not in geometric mean, where Chi-BD-SF proves to be the fairest algorithm among all when classifying.

TABLE V
AVERAGE RESULTS OF ACCURACY, GM, NUMBER OF RULES AND TIMES IN TEST USING 3 LABELS IN STANDARD CLASSIFICATION DATASETS

| Method | Acc. | GM | # Rules | Time (mm:ss) |
|---|---|---|---|---|
| CF | | | | |
| Chi-FRBCS | 0.74048 | 0.58571 | 580.46 | 00:33 |
| Chi-NE 2 Mappers | 0.74070 | 0.58762 | 580.46 | 00:19 |
| Chi-NE 16 Mappers | 0.71881 | 0.43986 | 580.38 | 00:10 |
| Chi-NE 64 Mappers | 0.57832 | 0.43614 | 580.38 | 00:06 |
| PCF | | | | |
| Chi-FRBCS | 0.74539 | 0.57563 | **498.16** | 00:33 |
| Chi-NE 2 Mappers | 0.74167 | 0.57962 | 502.46 | 00:20 |
| Chi-NE 16 Mappers | 0.71500 | 0.42419 | 545.9 | 00:10 |
| Chi-NE 64 Mappers | 0.57840 | 0.43632 | 575.6 | 00:07 |
| Chi-BD-SF | **0.76993** | **0.70342** | 580.46 | **00:02** |

TABLE VI
AVERAGE RESULTS OF ACCURACY, GM, NUMBER OF RULES AND TIMES IN TEST USING 5 LABELS IN STANDARD CLASSIFICATION DATASETS

| Method | Acc. | GM | # Rules | Time (mm:ss) |
|---|---|---|---|---|
| CF | | | | |
| Chi-FRBCS | 0.80033 | 0.75197 | 3,605.46 | 02:51 |
| Chi-NE 2 Mappers | 0.80041 | 0.75297 | 3,605.22 | 01:12 |
| Chi-NE 16 Mappers | 0.76441 | 0.60280 | 3,605.14 | 00:14 |
| Chi-NE 64 Mappers | 0.71077 | 0.67100 | 3,605.12 | 00:07 |
| PCF | | | | |
| Chi-FRBCS | **0.80792** | 0.75175 | **3,188.1** | 02:52 |
| Chi-NE 2 Mappers | 0.80815 | 0.75458 | 3,261.94 | 01:16 |
| Chi-NE 16 Mappers | 0.76445 | 0.59787 | 3,518.34 | 00:15 |
| Chi-NE 64 Mappers | 0.71090 | 0.67075 | 3,597.7 | 00:07 |
| Chi-BD-SF | 0.79907 | **0.77432** | 3,605.46 | **00:02** |

### *B. Results over the Big Data classification datasets*

In this subsection we present the results obtained with FRBCS Big Data implementations with large datasets in a cluster of computers. Due to the small differences in performance observed between CF and PCF, in this part of the study we have set CF by default. In this case we have used only experiments with 3 fuzzy partitions because of time constraints for the Chi-BD and Chi-NE implementations. Results are divided depending on the parallelism used in order to test the scalability, Table VII and VIII shows the results with 32 and 64 mappers respectively.

The behavior of the algorithms does not change from standard to large datasets. Chi-BD obtains better results than the approximate (local) counterpart, that is, Chi-NE. However, in terms of runtimes it becomes slower due to the bottleneck coming from the computation of the RWs. The Chi-BD algorithm tested was developed in Spark. In spite of the benefits of Spark, it does not reach the runtimes of original Hadoop Chi-BD [13]. Its efficiency, like Chi-NE, grows with the capacity of the cluster but the weight time calculus grows near exponentially with the complexity of the problem ($O(n * m)$ $n$ number of examples and $m$ number of fuzzy rules of the rule base), making them not scalable. Chi-BD-SF obtains the best results of all the algorithms. Increasing the level of parallelism affect positively its runtime results and does not affect its quality. Avoiding the weight and using the standard support makes the algorithm linear and scalable, even improving the quality as classifier.

## VII. CONCLUDING REMARKS

In this work we have proposed Chi-BD-SF as a robust novel FRBCS algorithm for Big Data problems. In order to obtain these results, it employs the standard support to make a quick decision between conflict fuzzy rules, i.e. those with same antecedent but different consequent.

TABLE VII
RESULTS OF ACCURACY, GM, NUMBER OF RULES AND TIME IN TEST USING 3 LABELS, CF AND 32 MAPPERS IN BIG DATA CLASSIFICATION DATASETS

| 32 mappers | | | | |
|---|---|---|---|---|
| Method | Acc. | GM | # Rules | Time (hh:mm:ss) |
| Covtype_1 | | | | |
| Chi-BD | 0.75365 | 0.67290 | 8,689.6 | 00:00:55 |
| Chi-NE | 0.68937 | 0.49344 | 8,689.6 | 00:00:50 |
| Chi-BD-SF | **0.78871** | **0.74322** | 8,689.6 | **00:00:03** |
| ECBDL_14-0.6 | | | | |
| Chi-BD | **0.97986** | 0.03506 | 479,340.0 | 01:55:18 |
| Chi-NE | 0.97985 | 0.03506 | **479,025.6** | 00:04:41 |
| Chi-BD-SF | 0.97983 | 0.03506 | 479,340.0 | **00:00:09** |
| Poker_0 | | | | |
| Chi-BD | **0.64103** | **0.63722** | 56,177.6 | 00:12:22 |
| Chi-NE | 0.54834 | 0.54494 | 56,177.6 | 00:02:46 |
| Chi-BD-SF | 0.60982 | 0.60805 | 56,177.6 | **00:00:03** |
| Susy | | | | |
| Chi-BD | 0.61378 | 0.45313 | 10,516.8 | 00:09:40 |
| Chi-NE | 0.59888 | 0.38954 | 10,516.8 | 00:25:58 |
| Chi-BD-SF | **0.65524** | **0.63631** | 10,516.8 | **00:00:19** |
| Average | | | | |
| Chi-BD | 0.74708 | 0.44958 | 138,681.0 | 00:34:34 |
| Chi-NE | 0.70411 | 0.36574 | **138,602.4** | 00:08:34 |
| Chi-BD-SF | **0.75840** | **0.50566** | 138,681.0 | **00:00:09** |

TABLE VIII
RESULTS OF ACCURACY, GM, NUMBER OF RULES AND TIMES IN TEST USING 3 LABELS, CF AND 64 MAPPERS IN BIG DATA CLASSIFICATION DATASETS

| 64 mappers | | | | |
|---|---|---|---|---|
| Method | Acc. | GM | # Rules | Time (hh:mm:ss) |
| Covtype_1 | | | | |
| Chi-BD | 0.75365 | 0.67290 | 8,689.6 | 00:01:09 |
| Chi-NE | 0.68409 | 0.61706 | 8,689.6 | 00:00:23 |
| Chi-BD-SF | **0.78871** | **0.74322** | 8,689.6 | **00:00:03** |
| ECBDL_14-0.6 | | | | |
| Chi-BD | **0.97986** | 0.03506 | 479,340.0 | 01:13:44 |
| Chi-NE | 0.97984 | 0.03506 | **479,024.8** | 00:01:15 |
| Chi-BD-SF | 0.97983 | 0.03506 | 479,340.0 | **00:00:08** |
| Poker_0 | | | | |
| Chi-BD | **0.64103** | **0.63722** | 56,177.6 | 00:08:09 |
| Chi-NE | 0.55469 | 0.55239 | 56,177.6 | 00:01:04 |
| Chi-BD-SF | 0.60982 | 0.60805 | 56,177.6 | **00:00:03** |
| Susy | | | | |
| Chi-BD | 0.61378 | 0.45313 | 10,516.8 | 00:06:09 |
| Chi-NE | 0.53592 | 0.29762 | 10,516.8 | 00:12:06 |
| Chi-BD-SF | **0.65524** | **0.63631** | 10,516.8 | **00:00:13** |
| Average | | | | |
| Chi-BD | 0.74708 | 0.44958 | 138,681.0 | 00:22:18 |
| Chi-NE | 0.68864 | 0.37553 | **138,602.2** | 00:03:42 |
| Chi-BD-SF | **0.75840** | **0.50566** | 138,681.0 | **00:00:07** |

We have tested Chi-BD-SF with different levels of granularity using a local machine showing its capacity to have low runtimes and good classification performance. When dealing with large datasests using 3 fuzzy sets, the differences at runtime compared with the rest of the algorithms increases greatly while the classification quality remains equal. We still have to determine how higher levels of granularity affects its classification results with Big Data datasets.

The lessons learned through this work contribution lead us to new possibilites for FRBCS in Big Data. Showing the weight as an excessive computational calculus for large datasets and at the same time, exposing an alternative method more suitable for Big Data algorithms. This study brings new future works. Firstly, test the impact of granularity in FRBCS in Big Data problems. Secondly, research and design new scalable FRBCS algorithms using the presented ideas. Thirdly, test the possibility of use the support of the rules to calculate the coverage degree. Lastly, adapt fuzzy support and confidence heuristics to Big Data problems.

## ACKNOWLEDGMENT

This work was supported in part by the Spanish Ministry of Science and Technology under Projects TIN2015-68454-R, TIN2016-77356-P and TIN2017-89517-P (AEI/FEDER, UE); and the Project BigDaP-TOOLS - Ayudas Fundación BBVA a Equipos de Investigación Científica 2016.

## REFERENCES

[1] I. Witten, E. Frank, M. Hall, and C. Pal, *Data Mining Practical Machine learning tools and techniques*. Elsevier, 2016.
[2] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan 2003.
[3] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314 – 347, 2014.
[4] A. Fernández, S. del Río, V. López, A. Bawakid, M. J. del Jesus, J. M. Benítez, and F. Herrera, "Big data with cloud computing: an insight on the computing environment, mapreduce, and programming frameworks," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 5, pp. 380–409, 2014.
[5] X. Wu, X. Zhu, G. Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, Jan 2014.
[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[7] G. Michael, "A framework for considering comprehensibility in modeling," *Big Data*, vol. 4, no. 2, pp. 75–88, 2016.
[8] M. Gacto, R. Alcalá, and F. Herrera, "Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures," *Information Sciences*, vol. 181, no. 20, pp. 4340 – 4360, 2011, special Issue on Interpretable Fuzzy Systems.
[9] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*, ser. Advanced Information Processing. Springer Berlin Heidelberg, 2006.

[10] H. Ishibuchi and T. Yamamoto, "Rule weight specification in fuzzy rule-based classification systems," *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 4, pp. 428–435, Aug 2005.
[11] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, and F. Herrera, "Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce," *Information Fusion*, vol. 42, pp. 51 – 61, 2018.
[12] S. del Río, V. López, J. M. Benítez, and F. Herrera, "A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules," *International Journal of Computational Intelligence Systems*, vol. 8, no. 3, pp. 422–437, 2015.
[13] M. Elkano, M. Galar, J. Sanz, and H. Bustince, "Chi-bd: A fuzzy rule-based classification system for big data classification problems," *Fuzzy Sets and Systems*, 2017, in press, doi:10.1016/j.fss.2017.07.003.
[14] Z. Chi, H. Yan, and T. Pham, *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1996.
[15] C. Lam, *Hadoop in Action*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2010.
[16] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, I. C. Society, Ed., 2010, pp. 1–10.
[17] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media, 2015.
[18] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*, ser. Advanced Information Processing. Springer Berlin Heidelberg, 2006.
[19] O. Cordón, M. J. del Jesus, and F. Herrera, "A proposal on reasoning methods in fuzzy rule-based classification systems," *International Journal of Approximate Reasoning*, vol. 20, no. 1, pp. 21–45, 1999.
[20] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml