2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

On the use of MapReduce to build Linguistic Fuzzy Rule Based Classification Systems for Big Data

Victoria López, Sara del Río, José Manuel Benítez and Francisco Herrera

***Abstract*— Big data has become one of the emergent topics when learning from data is involved. The notorious increment in the data generation has directed the attention towards the obtaining of effective models that are able to analyze and extract knowledge from these colossal data sources. However, the vast amount of data, the variety of the sources and the need for an immediate intelligent response pose a critical challenge to traditional learning algorithms.**

To be able to deal with big data, we propose the usage of a linguistic fuzzy rule based classification system, which we have called Chi-FRBCS-BigData. As a fuzzy method, it is able deal with the uncertainty that is inherent to the variety and veracity of big data and because of the usage of linguistic fuzzy rules it is able to provide an interpretable and effective classification model. This method is based on the MapReduce framework, one of the most popular approaches for big data nowadays, and has been developed in two different versions: Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave.

The good performance of the Chi-FRBCS-BigData approach is supported by means of an experimental study over six big data problems. The results show that the proposal is able to provide competitive results, obtaining more precise but slower models in the Chi-FRBCS-BigData-Ave alternative and faster but less accurate classification results for Chi-FRBCS-BigData-Max.

I. Introduction

ONE of the most highlighted trends in the recent years by the information technology industry is what is known as Big Data. The term "Big Data" symbolizes the analysis and treatment of data repositories of a colossal size, which traditional data management systems and analytics are unable to deal with [1]. This trend can be observed in multiple environments like webpages, multimedia data, social networks, mobile devices, sensor networks and so on [2].

With more data available, the analysis and knowledge extraction process should be benefited, and more accurate and precise information should be obtained. However, the standard techniques and approaches that are commonly used in data mining are not able to manage datasets this size [3]. Therefore, the standard learning methods need to be modified following the guidelines of the existing solutions that are able to effectively deal with big data while maintaining their predictive capacity.

Victoria López, Sara del Río, José Manuel Benítez and Francisco Herrera are with the Department of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, Granada, Spain (email: {vlopez, srio, J.M.Benitez, herrera}@decsai.ugr.es).

This work was partially supported by the Spanish Ministry of Science and Technology under project TIN2011-28488 and the Andalusian Research Plans P10-TIC-6858, P11-TIC-7765 and P12-TIC-2958. Victoria López holds a FPU scholarship from Spanish Ministry of Education.

Fuzzy Rule Based Classification Systems (FRBCSs) [4] are potent and popular tools for pattern recognition and classification. They are able to provide good precision results while they are able to supply an interpretable model for the end user by the usage of some linguistic labels. One of the complications that difficult the extraction of potential useful information in big data is the uncertainty that is associated to the variety and veracity inherent to big data. FRBCSs are able to effectively deal with uncertainty, ambiguity or vagueness making them a very interesting approach to deal with big data as they are able to manage its inherent incertitude.

In a scenario with big data, usually a high number of instances and/or attributes is provided. FRBCSs decrement their performance in these cases as the search space grows exponentially. This growth difficults the learning process leading to scalability or complexity problems that may end up with non-interpretable models [5]. To overcome this situation, several approaches that try to build parallel fuzzy systems have been presented [6][7]; however, they are focused on reducing the processing time while preserving the accuracy and they are not able to manage colossal collections of data.

The frameworks that are typically used to handle big data somehow involve some kind of parallelization so that they can easily process and analyze the data that is ready to be used. One of the most popular platforms nowadays, MapReduce [8], suggests a computational scheme where all the processing is distributed along two key operations: a map function that will act over a subset of the data, and a reduce function that will integrate the results obtained in the map function.

In this work, we present a FRBCS that is able to provide an interpretable model while maintaining a competitive predictive accuracy in the big data scenario, which has been denoted as Chi-FRBCS-BigData. This method is based on the Chi et al.'s approach [9], a classical FRBCS learning method, which has been modified to deal with big data following a MapReduce procedure. The Chi-FRBCS-BigData proposal has been developed under two different versions, Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave, which precisely differ in the "Reduce" operation and which are compared to analyze how they deal with big data.

Moreover, the Chi et al's method is especially suitable to be used in a parallel approach instead of using a more complex FRBCS method as it provides fuzzy rules that have the same structure and which can be independently created from a subset of examples. Furthermore, the usage of a FRBCS in big data is also quite interesting as it provides

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

a mechanism to manage the uncertainty that is inherent in this scenario because of the variety and veracity of data.

To support the suitability of the Chi-FRBCS-BigData approach we have selected six big data problems for our experimental study that will help to understand how the proposal works, which are its strong points and its limitations. This experimental study will measure the performance of the classifiers according to the accuracy obtained and the runtime spent by the models. In addition, and to detect the differences between the versions of the proposal, we study the significance of the results by means of statistical tests [10][11].

The rest of this paper is organized as follows. Section II briefly introduces the problem of big data. Next, Section III contains the approaches developed in this work, the versions of the Chi-FRBCS-BigData method, together with some previous concepts about FRBCSs. Then, the experimental study is shown along Section IV. Finally, Section V summarizes and concludes the work.

II. Big data in classification

In the late years, the term "Big Data" has emerged as one of the most hot topics related to the information technology industry. This concept is related to the impressive growth in data generation that has taken place recently and that has highlighted the interest in obtaining useful information from these immense data sources. Specifically, the concept of "Big Data" is is applied to all the information that cannot be processed or analyzed using traditional techniques or tools [12]. One of the early and well-known definitions of big data [13], describes the concept as a 3Vs model (volume, velocity and variety):

- **Volume:** This feature is related to the enormous size of the data that needs to be treated to extract useful information.
- **Velocity:** When analyzing big data it is of the utmost importance to provide an informed response within a reasonable time limit.
- **Variety:** This characteristic refers to the diverse type of data that will compose the data corpus. For instance, in big data it is typical to merge structured and unstructured data like tabular data from databases, hierarchical data, documents, graph data and so on.

Later on, additional Vs have been proposed by some organizations to describe the big data model [14]: validity, volatility, value, variability and veracity.

Big data can be seen in numerous real-world environments, it can be presented in any format and can be attained from diverse origins. For instance, financial data like the tradings of the day in the New York Stock Exchange can add up to one new TB per day. Multimedia data, leads to occupy one PB in Facebook servers with approximately 10 billion photos. Even more simple data like the one stored by the Internet Archive can accumulate 2 PB of data per day [12].

To effectively deal with big data, Google presented a parallel programming model, MapReduce, which is a framework for processing large volumes of data over a cluster of machines [8][15] and which has become one of the most popular approaches nowadays. The MapReduce paradigm revolves around two key operations: a map function and a reduce function. In a first phase, the input data is processed by the map function which produces some intermediate results; these intermediate results will be then fed in a second phase to a reduce function, which somehow combines the intermediate results to present a final output.

The MapReduce model is based on a essential data structure that is traditionally known as a key-value pair. All the data processed, the intermediate results and the final output are expressed in this key-value form. In this manner, the map and reduce functions that can be seen in a MapReduce procedure are:

- **Map function:** In the map function the master node performs an automatic division of the data into independent data blocks which are then distributed and transferred to the worker nodes. Each worker node processes independently its data chunk and produces a result that is transmitted back to the master node. In terms of the key-value pairs, it is said that the map function receives a key-value pair as input and produces a list of intermediate key-value pairs. These intermediate key-value pairs are then automatically shuffled and ordered according to the intermediate key to speed up the reduce step.
- **Reduce function:** In the reduce function, the master node collects the output results produced in the previous phase and then, uses them in some manner to conform the final result of the algoritm. Again, in terms of the key-value pairs, the reduce function obtains the intermediate key-value pairs computed previously aggregated by the key values and creates an output value that will be the output of the method.

Figure 1 depicts a standard MapReduce program with its map step and its reduce step. The terms k and v refer to the original key and value pair respectively; k' and v' are the intermediate key-value pair that is generated after the use of the map function; and v'' is the final value given as a result of the algorithm.

Hadoop is the most popular implementation of the MapReduce programming model [12]. It is an open-source project written in Java and supported by the Apache software foundation that tries to facilitate the processing and management of large datasets in a distributed manner. It provides auxiliary services analogous to the ones available in Google's MapReduce implementation.

Machine learning methods have also started to be integrated using the MapReduce paradigm to deal with big data. The Mahout project [16], also supported by the Apache software foundation, is a machine learning library that features scalable machine learning applications over Hadoop or other scalable systems.

Nevertheless, a MapReduce design is not always suitable for all kind of computations [17]. Some examples of that are iterative algorithms or graph-based algorithms. In order

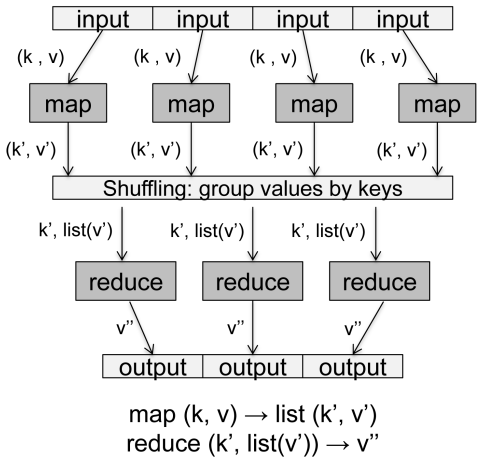

Fig. 1. The MapReduce programming model

to overcome these problems, several approaches have been proposed to deal with big data as substitutes for MapReduce and Hadoop. These approaches include projects like Spark, Apache Drill, Twister or Impala, just to mention some of them.

III. Chi-FRBCS-BigData: A Linguistic Fuzzy Rule Based Classification System for Big Data

In this section, we will introduce two versions of a linguistic FRBCS that manage big data. To do so, first, we present some definitions related to FRBCSs and the fuzzy learning algorithm that has been adapted in this work, Chi-FRBCS. Then, we will describe how this method is adapted for big data using a MapReduce scheme that is modified to produce two variants that will provide different classification results.

A. Fuzzy Rule Based Classification Systems

A FRBCS is composed by two elements: the Inference System and the Knowledge Base (KB). In a linguistic FRBCS, the KB is formed from the Data Base (DB), which contains the membership functions of the fuzzy partitions associated to the input attributes, and the Rule Base (RB), which comprises the fuzzy rules that describe the problem. Traditionally, expert information to build the KB is not available and therefore, a machine learning procedure is needed to construct the KB from the available examples.

A classification problem is usually defined by m training samples $x_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, 2, \ldots, m$ from M classes where x_{pi} is the value of attribute i ($i = 1, 2, \ldots, n$) of the p-th training sample. In this work, we use fuzzy rules of the following form to build our FRBCS:

$$\begin{aligned} &\text{Rule } R_j: \text{ If } x_1 \text{ is } A_j^1 \text{ and } \ldots \text{ and } x_n \text{ is } A_j^n \\ &\text{then Class = } C_j \text{ with } RW_j \end{aligned} \tag{1}$$

where R_j is the label of the j-th rule, $\mathbf{x} = (x_1, \ldots, x_n)$ is a n-dimensional pattern vector, A_j^i is an antecedent fuzzy set, C_j is a class label, and RW_j is the rule weight [18]. We use triangular membership functions as linguistic labels.

There are many alternatives that have been proposed to compute the rule weight [18]. Among them, a good choice is to use the heuristic method known as the Penalized Certainty Factor (PCF) [19]:

$$\text{RW}_j = \text{PCF}_j = \frac{\sum_{x_p \in C_j} \mu_{A_j}(x_p) - \sum_{x_p \notin C_j} \mu_{A_j}(x_p)}{\sum_{p=1}^{m} \mu_{A_j}(x_p)} \tag{2}$$

where $\mu_{A_j}(x_p)$ is the membership degree of the x_p p-th example of the training set with the antecedents of the rule and C_j is the consequent class of rule j. We use the fuzzy reasoning method of the wining rule [20] when predicting a class using the built KB for a given example.

B. The Chi et al.'s algorithm for Classification

To build the KB of a linguistic FRBCS, we need to use a learning procedure that specifies how the DB and RB are created. In this work, we use the method proposed in [9], an extension of the well-known Wang and Mendel method for classification [21], which we have called the Chi et al's method, Chi-FRBCS.

To generate the fuzzy KB, this generation method tries to find the relationship between the input attributes and the classes space following the next steps:

1) *Building the linguistic fuzzy partitions*: This step builds the fuzzy DB from the domain associated to each attribute A_i using equally distributed triangular membership functions.
2) *Generating a new fuzzy rule associated to each example* $\mathbf{x}_p = (x_{p1}, \ldots, x_{pn}, C_p)$:
   a) Compute the matching degree $\mu(\mathbf{x}_p)$ of the example with respect to the fuzzy labels of each attribute using a conjunction operator.
   b) Select the fuzzy region that obtains the maximum membership degree in relation with the example.
   c) Build a new fuzzy rule whose antecedent is calculated according to the previous fuzzy region and whose consequent is the class label of the example C_p.
   d) Compute the rule weight.

When following the previous procedure, several rules with the same antecedent can be built. If they have the same class in the consequent, then, duplicated rules are deleted. However, if the class in the consequent is different, only the rule with the highest weight is maintained in the RB.

C. The Chi-FRBCS-BigData algorithm: A MapReduce Design

At this point, we present the Chi-FRBCS-BigData algorithm which is a FRBCS that is able to effectively classify big data. To do so, this method uses two different MapReduce processes to deal with two different parts of the algorithm: one MapReduce process is devoted to the building of the fuzzy KB from a big data training set and the other MapReduce process is used to estimate the class of samples belonging to big data sample sets. Both processes follow the

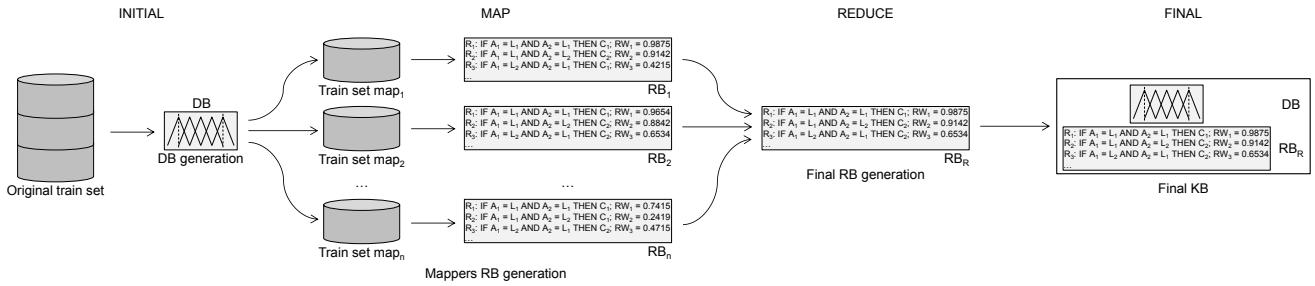

Fig. 2. A flowchart of how the building of the KB is organized in Chi-FRBCS-BigData

MapReduce structure distributing all the computations along several processing units that manage different chunks of information, aggregating the results obtained in an appropriate manner.

Furthermore, we have produced two versions of the Chi-FRBCS-BigData algorithm, which we have named Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave. These versions share most of their operations, however, they behave differently in the "Reduce" step of the approach, when the different rule bases generated by each mapper are combined. These versions obtain different rule bases and thus, different KBs, providing different results when estimating the class of new examples.

The procedure to build the fuzzy KB following a MapReduce scheme in Chi-FRBCS-BigData is depicted in Figure 2. This procedure is divided into the following phases:

1) *Initial:* In this first phase, the method computes the domain associated to each attribute A_i using the whole training set. With that information, the fuzzy DB is created using equally distributed triangular membership functions as in Chi-FRBCS. Then, the system automatically segments the original training dataset into independent data blocks which are automatically transferred to the different processing units together with the created fuzzy DB.
2) *Map:* In this second phase, each processing unit works independently over its available data to build its associated fuzzy RB (called RB_i in Figure 2) following the original Chi-FRBCS method.
   Specifically, for each example in the data partition, an associated fuzzy rule is created: first, the membership degree of the fuzzy labels is computed according to the example values; then, the fuzzy region that obtains the greatest value is selected to become the antecedent of the rule; next, the class of the example is assigned to the rule as consequent; and finally, the rule weight is computed *using the set of examples that belong to the current map process*.
   After the rules have been created and before finishing the map step, each map process searches for rules with the same antecedent. If the rules share the same consequent, only one rule is preserved; if the rules have different consequents, only the rule with the highest weight is kept in the mappers RB.
3) *Reduce:* In this third phase, a processing unit receives the results obtained by each map process (RB_i) and combines them to form the final RB (called RB_R in Figure 2). The combination of the rules is straight-forward: the rules created by each mapper $RB_1, RB_2, \ldots, RB_n$ are all integrated in one RB, RB_R. However, contradictory rules (rules with the same antecedent, with or without the same consequent and with different rule weight) may be created. Therefore, specific procedures to deal with these contradictory rules are needed. Precisely, these procedures define the two variants of the Chi-FRBCS-BigData algorithm:
   a) **Chi-FRBCS-BigData-Max:** In this approach, the method searches for the rules with the same antecedent. Among these rules, only the rule with the highest weight is maintained in the final RB, RB_R. In this case it is not necessary to check if the consequent is the same or not, as we are only maintaining the most powerful rules. Equivalent rules (rules with the same antecedent and consequent) can present different weights as they are computed in different mapper processes over different training sets.
      For instance, if we have five rules with the same antecedent and the following consequents and rule weights: R_1: Class 1, $RW_1 = 0.8743$; R_2: Class 2, $RW_2 = 0.9254$; R_3: Class 1, $RW_3 = 0.7142$; R_4: Class 2, $RW_4 = 0.2143$ and R_5: Class 1, $RW_5 = 0.8215$, then, Chi-FRBCS-BigData-Max will keep in RB_R the rule R_2: Class 2, $RW_2 = 0.9254$ because it is the rule with the maximum weight.
   b) **Chi-FRBCS-BigData-Ave:** In this approach, the method also searches for the rules with the same antecedent. Then, the average weight of the rules that have the same consequent is computed (this step is needed because rules with the same antecedent and consequent may have different weights as they are built over different training sets). Finally, the rule with the greatest average weight is kept in the final RB, RB_R.
      For instance, if we have five rules with the same antecedent and the following consequents

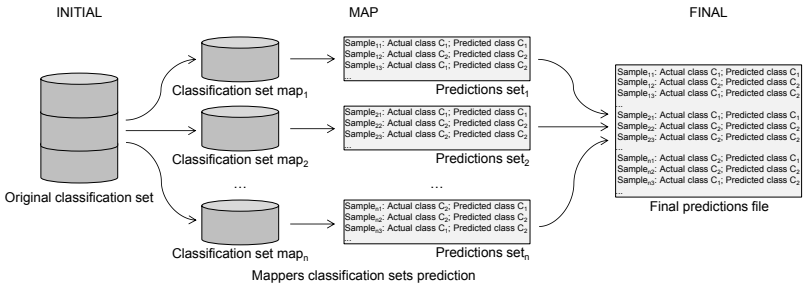

Fig. 3. A flowchart of how the classification of a big data classification set is organized in Chi-FRBCS-BigData

and rule weights: R_1: Class 1, $RW_1 = 0.8743$; R_2: Class 2, $RW_2 = 0.9254$; R_3: Class 1, $RW_3 = 0.7142$; R_4: Class 2, $RW_4 = 0.2143$ and R_5: Class 1, $RW_5 = 0.8215$, then, Chi-FRBCS-BigData-Ave will first compute the average weight for the rules with the same consequent, namely, R_{C1}: Class 1, $RW_{C1} = 0.8033$ and R_{C2}: Class 2, $RW_{C2} = 0.5699$, and it will keep in RB_R the rule R_{C1}: Class 1, $RW_{C1} = 0.8033$ because it is the rule with the maximum average weight.

Please note that it is not needed for any Chi-FRBCS-BigData version to recompute the rule weights in the "Reduce" stage, as we are calculating the new rule weights from the previously rule weights provided by each mapper.

4) *Final:* In this last phase, the results computed in the previous phases are provided as the output of the computation process. Precisely, the generated fuzzy KB is composed by the fuzzy DB built in the "Initial" phase and the fuzzy RB, RB_R, is finally obtained in the "Reduce" phase. This KB will be the model that will be used to predict the class for new examples.

As it was previously said, Chi-FRBCS-BigData uses another MapReduce mechanism to estimate the class of examples that belong to big data classification sets using the fuzzy KB built within the previous step. This approach follows a similar scheme to the previous step where the initial dataset is distributed along several processing units that provide a result that will be part of the final result. Specifically, this class estimation process is depicted in Figure 3 and follows the coming phases:

1) *Initial:* In this first phase, the method does not need to perform a specific operation. The system automatically segments the original big data dataset that needs to be classified into independent data blocks which are automatically transferred to the different processing units together with the previously created fuzzy KB.
2) *Map:* In this second phase, each map task estimates the class for the examples that are included in its data partition. To do so, each processing unit goes through all the examples in its data chunk and predicts its output class according to the given fuzzy KB and using the fuzzy reasoning method of the wining rule. Please note that Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave will produce different classification estimations because the input fuzzy RBs are also different, however, the class estimation process followed is exactly the same for both approaches.
3) *Final:* In this last phase, the results computed in the previous phase are provided as the output of the computation process. Precisely, the estimated classes for the different examples of the big data classification set are aggregated just concatenating the results provided by each map task.

It is important to note that this mechanism does not include a "Reduce" step as it is not necessary to perform a computation to combine the results obtained in the "Map" phase.

IV. Experimental Study

In this section, we first provide some details of the problems selected for the experiments, the configuration parameters for the methods analyzed and the statistical tests applied to compare the results (Section IV-A). Then, we provide in Section IV-B the accuracy performance of the approaches tested in the study with respect to the number of mappers considered. Finally, the runtime spent by the algorithms over the selected data is shown in Section IV-C.

A. Experimental Framework

In this study, our aim is to analyze the behavior of the Chi-FRBCS-BigData algorithm in the scenario of big data. To do so, we will consider six problems from the UCI dataset repository [22], shown in Table I, where we denote the number of examples (#Ex.), number of attributes (#Atts.), selected classes and the number of examples per class. This table is in descending order according to the number of examples.

TABLE I
Summary of datasets

Datasets	#Ex.	#Atts.	Selected classes	#Samples per class
RLCP	5749132	2	(FALSE; TRUE)	(5728201; 20931)
Kddcup_DOS_vs_normal	4856151	41	(DOS; normal)	(3883370; 972781)
Poker_0_vs_1	946799	10	(0; 1)	(513702; 433097)
Covtype_2_vs_1	495141	54	(2; 1)	(283301; 211840)
Census	141544	41	(-_50000.; 50000+.)	(133430; 8114)
Fars_Fatal_Inj_vs_No_Inj	62123	29	(Fatal_Inj; No_Inj)	(42116; 20007)

The selected datasets only feature two classes even when some of them are multi-class problem. In this work, we have decided to limit the number of classes despite of the ability of the Chi-FRBCS-BigData algorithm to deal with multiple classes to avoid the imbalance in the data that arises in many real-world problems [23], as the division approach of the MapReduce scheme presented aggravates the small sample size problem, which decrements the performance in the imbalanced scenario.

In order to develop our study we use a 10-fold stratified cross validation partitioning scheme, that is, nine random partitions of data with a 10% of the samples where the combination of 9 of them (90%) is considered as training set and the remaining one is treated as test set. For each dataset we consider the average results of the ten partitions.

To verify the performance of the proposed model, we compare the results obtained by Chi-FRBCS-BigData-Max with Chi-FRBCS-BigData-Ave so that we can understand how they behave over the selected big data problems.

The configuration parameters used for these algorithms are the following: three fuzzy labels for each attribute, the product T-norm is used to compute the matching degree of the antecedent of the rule with the example, the PCF is used to compute the rule weight and the winning rule is used as fuzzy reasoning method. Additionally, another parameter is used in the MapReduce procedure, which is the number of mappers associated to the computation. This value has been set to 16, 32 and 64 mappers.

To perform the experiments we have used the Atlas research group's cluster with 16 nodes, connected with a 1Gb/s ethernet. Each node is composed by two Intel E5-2620 microprocessors (at 2 GHz, 15MB cache) and 64GB of memory running under Linux CentOS 6.3. Furthermore, the cluster works with Hadoop 2.0.0 (Cloudera CDH4.5.0), where one node is configured as name-node and job-tracker, and the rest are data-nodes and task-trackers.

Moreover, when an experimental study is carried out, it is highly advised that the extracted conclusions are validated through the use of statistical tests [10][11]. Standard parametric tests, like the t–test, need to meet some initial conditions in data that are not always met in classification experiments and, therefore, non-parametric tests need to be used in their place.

In this work, we compare the performance of the approaches using a Wilcoxon signed-rank test [24], a non-parametric statistical set suitable for pairwise comparisons. This test calculates the differences between two classifiers and then, ranks them in ascending order with respect to their absolute value. With these ranks, we compute the R^+ and R^- values: R^+ is the addition of the ranks where the first algorithm outperforms the second, and R^- sums the contrary case. With this information, the p–value associated to the statistical distribution is calculated and if that value is below a pre-specified significance level α, then the null hypothesis of equality of means can be rejected.

B. Analysis of the Chi-FRBCS-BigData precision

In this section, we will try to identify the feasible differences between the two versions of the Chi-FRBCS-BigData proposal: Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave (for the sake of space, these algorithms are called Chi-BigData-Max and Chi-BigData-Ave in the Tables).

With this aim, Table II shows the average accuracy classification values obtained by the Chi-FRBCS-BigData versions. This table shows the average training and test results of each approach and is divided in three horizontal parts that correspond to the performance results achieved with the different number of mappers. Moreover, the bold values indicate which algorithm is more effective to classify the test examples for a given number of mappers, and the underlined values highlight which is the best performing method in test in all the experiments considered.

TABLE II
AVERAGE ACCURACY RESULTS FOR THE CHI-FRBCS-BIGDATA VERSIONS USING 16, 32 AND 64 MAPPERS

Datasets	**16 mappers**			
	Chi-BigData-Max		Chi-BigData-Ave	
	Acc_{tr}	Acc_{tst}	Acc_{tr}	Acc_{tst}
RLCP	99.63	**99.63**	99.63	**99.63**
Kddcup_DOS_vs_normal	99.93	**99.93**	99.93	**99.93**
Poker_0_vs_1	62.18	59.88	62.58	**60.35**
Covtype_2_vs_1	74.77	**74.72**	74.77	74.69
Census	97.14	**93.75**	97.15	93.52
Fars_Fatal_Inj_vs_No_Inj	96.69	94.75	97.06	**95.01**
Average	88.39	87.11	88.52	**87.19**
	32 mappers			
	Chi-BigData-Max		Chi-BigData-Ave	
	Acc_{tr}	Acc_{tst}	Acc_{tr}	Acc_{tst}
RLCP	99.63	**99.63**	99.63	**99.63**
Kddcup_DOS_vs_normal	99.92	**99.92**	99.92	**99.92**
Poker_0_vs_1	61.27	58.93	61.82	**59.30**
Covtype_2_vs_1	74.69	74.62	74.88	**74.85**
Census	97.11	**93.48**	97.12	93.32
Fars_Fatal_Inj_vs_No_Inj	96.49	94.26	96.87	**94.63**
Average	88.19	86.81	88.37	**86.94**
	64 mappers			
	Chi-BigData-Max		Chi-BigData-Ave	
	Acc_{tr}	Acc_{tst}	Acc_{tr}	Acc_{tst}
RLCP	99.63	**99.63**	99.63	**99.63**
Kddcup_DOS_vs_normal	99.92	99.92	99.93	**99.93**
Poker_0_vs_1	60.45	57.95	60.88	**58.12**
Covtype_2_vs_1	74.67	74.52	75.05	**74.96**
Census	97.07	**93.30**	97.13	93.11
Fars_Fatal_Inj_vs_No_Inj	96.27	93.98	96.76	**94.56**
Average	88.00	86.55	88.23	**86.72**

From this table we can observe that, in average, the Chi-FRBCS-BigData-Ave method is able to provide better classification results both in training and test than Chi-FRBCS-BigData-Max for any number of mappers considered. Therefore, obtaining the average rule weight of all the partial rule bases obtained show a positive influence in classification as we are trying to make the rules as general as possible. The

only clear exception to this tendency can be observed in the "Census" dataset that obtains slightly better results for the Chi-FRBCS-BigData-Max variant. This behavior may be explained in relation with the training results, as it seems that this specific dataset is the one that presents a greater gap between the training and testing results.

Moreover, the performance results improve when a smaller number of mappers is used for both Chi-FRBCS-BigData versions and for both training and test sets. This behavior is also expected from the MapReduce design followed, as the rule weights are originally estimated from smaller data subsets if the number of mappers is high, and therefore, the estimation performed is even further in theses cases from the rule weight value what would be computed if the whole dataset was available. However, there are also cases like the "Covtype_2_vs_1" dataset where this trend is not observed.

In order to give statistical support to the findings previously extracted, in Table III we carry out a Wilcoxon test to compare how both Chi-FRBCS-BigData variants behave when different number of mappers are used. From this test, we may conclude that there are no clear differences between the approaches as the obtained p–values are not lower than a given significance level $\alpha = 0.05$ or 0.1.

TABLE III

WILCOXON TEST TO COMPARE THE ACCURACY ON THE CHI-FRBCS-BIGDATA VERSIONS. R^+ CORRESPONDS TO THE SUM OF THE RANKS FOR CHI-BIGDATA-MAX AND R^- TO CHI-BIGDATA-AVE

Comparison	#Mappers	R^+	R^-	p-Value
Chi-BigData-Max vs Chi-BigData-Ave	16	5.0	10.0	0.4185
Chi-BigData-Max vs Chi-BigData-Ave	32	3.0	12.0	0.1775
Chi-BigData-Max vs Chi-BigData-Ave	64	3.0	12.0	0.1775

Even when we cannot find statistical differences, we can observe that there is a tendency to consider the Chi-FRBCS-BigData-Ave approach as the best performing one, as the sum of ranks is always directed to its side. Furthermore, we can also see that the difference between the approaches is smaller (higher p–value) when the number of mappers is also smaller, which is precisely when both approaches obtain a better classification performance.

C. Analysis of the Chi-FRBCS-BigData runtime

In this section, we will focus on understanding the different behavior of the two versions of the Chi-FRBCS-BigData proposal with respect to the runtime of the model.

Table IV shows the runtime in seconds spent by the Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave methods. As in the previous case, this table is divided in three parts, which show the results for each dataset with respect to the different number of mappers. There are two types of highlighting in the table: the bold values correspond to the fastest method within the same number of mappers while the underlined values refer to the quickest execution for a dataset.

In average, we can see that the runtime results show a better behavior for the Chi-FRBCS-BigData-Max algorithm

TABLE IV

AVERAGE RUNTIME ELAPSED IN SECONDS FOR THE CHI-FRBCS-BIGDATA VERSIONS USING 16, 32 AND 64 MAPPERS

Datasets	Chi-BigData-Max	Chi-BigData-Ave
	16 mappers – Runtime (s)	
RLCP	9023.82	**8868.84**
Kddcup_DOS_vs_normal	30120.03	**29820.01**
Poker_0_vs_1	**3075.50**	6582.32
Covtype_2_vs_1	1477.67	**924.65**
Census	939.32	**884.30**
Fars_Fatal_Inj_vs_No_Inj	363.05	**236.40**
Average	**7499.90**	7886.09
	32 mappers – Runtime (s)	
RLCP	2460.89	**2303.02**
Kddcup_DOS_vs_normal	7890.87	**7708.96**
Poker_0_vs_1	**2210.13**	6331.09
Covtype_2_vs_1	**391.40**	493.00
Census	**388.64**	771.04
Fars_Fatal_Inj_vs_No_Inj	**141.92**	228.96
Average	**2247.31**	2972.68
	64 mappers – Runtime (s)	
RLCP	**701.31**	714.41
Kddcup_DOS_vs_normal	**2079.93**	2096.34
Poker_0_vs_1	**1635.98**	8373.40
Covtype_2_vs_1	**252.19**	348.86
Census	**325.24**	764.94
Fars_Fatal_Inj_vs_No_Inj	**136.24**	241.75
Average	**855.15**	2089.95

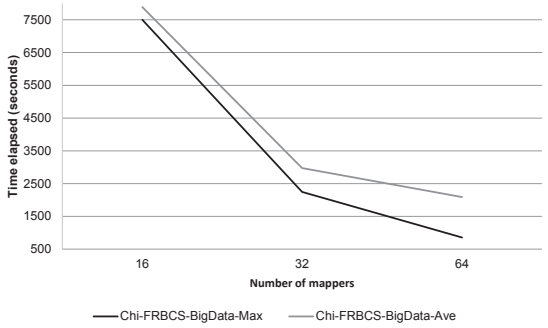

Fig. 4. Average runtimes for the Chi-FRBCS-BigData versions

for all the values of the number of mappers considered. This behavior is expected as this version of the algorithm performs less operations than the alternative and the operations performed are simpler. For the smallest number of mappers considered, it seems that there is a greater number of cases that benefit the Chi-FRBCS-BigData-Ave alternative, however, this improvement per dataset is not very high and it is not able to compensate how much slower this alternative is in the "Poker_0_vs_1" dataset. In Figure 4, we can see the difference between the runtime of the Chi-FRBCS-BigData alternatives in average, where the Chi-FRBCS-BigData-Ave version consumes more of time.

Furthermore, both versions also notably decrement their runtimes when larger values of mappers are used. This diminution in the runtime does not follow a lineal proportion

(as it can be seen from Figure 4). For instance, the speed gain when we double the number of processing units is much greater than reducing the processing time by half.

We can also see that this runtime improvement is not proportional over the different datasets: the biggest datasets are the ones that are able to further improve their performance while the smaller datasets are not able to do so in the same proportion. Moreover, the Chi-FRBCS-BigData-Max algorithm is able to scale up better than the Chi-FRBCS-BigData-Ave alternative, as the second approach seems to halt its progression when 64 mappers are used.

To sum up, our experimental study shows that the Chi-FRBCS-BigData-Ave alternative allows us to obtain better classification results for the Chi-FRBCS-BigData algorithm. We have also encountered that greater values for the number of mappers decrement the accuracy of the model as the model is less general when it is built over the smaller data subsets.

As a counterpart, the Chi-FRBCS-BigData-Max version does not have a significant drop in the accuracy performance with respect to the Chi-FRBCS-BigData-Ave alternative and it provides better response times than it. Furthermore, its speed gain is notable when higher number of mappers are used. In this manner, it is necessary to establish a trade-off in each occasion so that the most suitable Chi-FRBCS-BigData approach is selected according to our needs.

V. Concluding Remarks

In this work, we have introduced a linguistic fuzzy rule-based classification method for big data named Chi-FRBCS-BigData. This model obtains an interpretable model that manages colossal collections of data without damaging the classification accuracy and with fast response times.

Moreover, this approach has been designed using one of the most popular approaches for big data nowadays: the MapReduce framework. In this manner, this algorithm distributes its computing using a map function and combines the output via a reduce function. Specifically, the Chi-FRBCS-BigData proposal has been developed under two versions which have been called Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave. Although these alternatives follow the same structure and share numerous operations, its differences in the reduce function finally produce diverse classification models with divergent classification results.

The performance of the Chi-FRBCS-BigData alternatives has been contrasted in an experimental study including six different big data problems. These results corroborate the goodness of the approaches; however, it is not possible to identify a clear winner and it is needed to select one of them according to our needs. If we aim to obtain the best precision results, then, using the Chi-FRBCS-BigData-Ave method with a lower value for the number of mappers seems to be choice in spite of worse runtime results. On the contrary case, if we are interested in obtaining the fastest results without greatly damaging the performance, then, using the Chi-FRBCS-BigData-Max with a high number of mappers seems to be the sensible choice.

References

[1] P. Zikopoulos, C. Eaton, D. DeRoos, T. Deutsch and George Lapis, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data.* McGraw-Hill, 2011.

[2] S. Madden, "From Databases to Big Data," *IEEE Internet Computing*, vol. 16, no. 3, pp. 4–6, 2012.

[3] A. Sathi, *Big Data Analytics: Disruptive Technologies for Changing the Game.* MC Press, 2012.

[4] H. Ishibuchi, T. Nakashima and M. Nii, *Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining.* Springer–Verlag, 2004.

[5] Y. Jin, "Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 2, pp. 212–221, 2000.

[6] T.P. Hong, Y.C. Lee and M.T. Wu, "An effective parallel approach for genetic-fuzzy data mining," *Expert Systems with Applications*, vol. 41, no. 2, pp. 655–662, 2014.

[7] H. Ishibuchi, S. Mihara and Y. Nojima, "Parallel distributed hybrid fuzzy GBML models with rule set migration and training data rotation," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 2, pp. 355–368, 2013.

[8] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[9] Z. Chi, H. Yan and T. Pham, *Fuzzy algorithms with applications to image processing and pattern recognition.* World Scientific, 1996.

[10] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[11] S. García and F. Herrera, "An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2607–2624, 2008.

[12] T. White, *Hadoop, The Definitive Guide.* O'Reilly Media, Inc., 2012.

[13] D. Laney, "3D Data Management: Controlling Data Volume, Velocity, and Variety," *[Online; accessed January 2014]* (`http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf`), 2001.

[14] IBM, "What is big data? Bringing big data to the enterprise," *[Online; accessed January 2014]* (`http://www-01.ibm.com/software/data/bigdata/`), 2012.

[15] J. Dean and S. Ghemawat, "MapReduce: A flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[16] S. Owen, R. Anil, T. Dunning and E. Friedman, *Mahout in Action.* Manning Publications Co., 2011.

[17] J. Lin, "MapReduce is Good Enough? If All You Have is a Hammer, Throw Away Everything That's Not a Nail!," *Big Data*, vol. 1, no. 1, pp. 28–37, 2013.

[18] H. Ishibuchi and T. Nakashima, "Effect of Rule Weights in Fuzzy Rule-Based Classification Systems," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 506–515, 2001.

[19] 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, 2005.

[20] 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.

[21] L.X. Wang and J.M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992.

[22] K. Bache and M. Lichman, "UCI Machine Learning Repository," *[Online; accessed January 2014]* (`http://archive.ics.uci.edu/ml`), 2014.

[23] V. López, A. Fernández, S. García, V. Palade and Francisco Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013.

[24] D. Sheskin, *Handbook of parametric and nonparametric statistical procedures.* Chapman & Hall/CRC, 2006.