Ducange *et al. J Big Data* (2020) 7:19
https://doi.org/10.1186/s40537-020-00298-6

Journal of Big Data

SURVEY PAPER Open Access

An overview of recent distributed algorithms for learning fuzzy models in Big Data classification

Pietro Ducange[1*], Michela Fazzolari[2] and Francesco Marcelloni[1]

*Correspondence:
pietro.ducange@unipi.it
[1] Dipartimento di Ingegneria dell'Informazione, Largo Lucio Lazzarino, 1, 56122 Pisa, Italy
Full list of author information is available at the end of the article

Abstract

Nowadays, a huge amount of data are generated, often in very short time intervals and in various formats, by a number of different heterogeneous sources such as social networks and media, mobile devices, internet transactions, networked devices and sensors. These data, identified as Big Data in the literature, are characterized by the popular *Vs* features, such as Value, Veracity, Variety, Velocity and Volume. In particular, Value focuses on the useful knowledge that may be mined from data. Thus, in the last years, a number of data mining and machine learning algorithms have been proposed to extract knowledge from Big Data. These algorithms have been generally implemented by using ad-hoc programming paradigms, such as MapReduce, on specific distributed computing frameworks, such as Apache Hadoop and Apache Spark. In the context of Big Data, fuzzy models are currently playing a significant role, thanks to their capability of handling vague and imprecise data and their innate characteristic to be interpretable. In this work, we give an overview of the most recent distributed learning algorithms for generating fuzzy classification models for Big Data. In particular, we first show some design and implementation details of these learning algorithms. Thereafter, we compare them in terms of accuracy and interpretability. Finally, we argue about their scalability.

Keywords: Big Data, Fuzzy models, Data mining, Classification algorithms, Distributed computing

Introduction

In the *Big Data Era* [1], a huge *Volume* of information is generated at very high speed. In most cases, such data are collected from different sources, may have different formats (*Variety*) and need to be elaborated in almost real time (*Velocity*) [2]. This is the so-called three-V's model of Big Data and it has been used for the first time by Douglas Laney in 2001 [3], to describe the data management in three-dimensions. This original three-V paradigm is still valid, but it has been recently enriched by additional Vs. In fact, Big Data may be poorly accurate or truthful (*Veracity*). Moreover, the added-*Value* that the analysis of Big Data may offer is already exploited in several contexts such as industrial applications [4], marketing strategies [5], Cloud Computing and Internet of Things [6, 7], and health care [8].

Dealing with Big Data to extract useful knowledge and value is not a trivial task. This is mainly due to volume, diversity, noisiness, redundancy and complexity features, which characterize this kind of data. In particular, due to their huge volume, it is impossible to load all the data into the memory of a single machine. This prevents the execution of classical sequential algorithms, including *data mining* and *machine learning* procedures [9].

To this aim, novel distributed implementations of data mining and machine learning algorithms for Big Data have been proposed, mainly based on the *MapReduce* paradigm [10]. For example, Ludwig in [11] and Kim et al. in [12] discuss the design, the implementation and the experimentation of distributed clustering algorithms. As regards classification algorithms, very interesting results, in terms of accuracy and scalability, have been discussed by Bechini et al. in [13] and by Maillo et al. in [14]. Highlights on the recent advances, challenges and objectives in designing, developing and using data mining and machine learning algorithms for Big Data can be found in the work of Zhou et al. discussed in [15].

As stated before, the classical data storage and elaboration paradigms are not suitable for handling Big Data. Thus, in the last years practitioners and researchers have experimented new distributed frameworks, specifically developed for large-scale data storage and processing over a large number of computers, called nodes, which communicate over a network. Nodes interact in order to achieve a common goal, i.e. to solve a problem or give insights on a set of data. Each node is an independent unit, with its own CPU cores, memory and network interface. In distributed computing the components are located on these networked computers and communicate and coordinate their actions through messages. Some important characteristics of distributed systems are concurrency of components, lack of a global clock and independent failure of components [16]. The most popular distributed frameworks to manage Big Data are *Apache Hadoop* [17] and *Apache Spark* [18], which are described in detail in "The MapReduce paradigm and distributed computing frameworks" section.

The strategies presented so far are useful to address the Big Data issues connected with Volume, Velocity and Value. On the other hand, other issues arise due to data Variety and Veracity, as already mentioned above. To deal with these additional problems, Fernandez et al. in [19] and Hariri et al. in [20] highlighted that fuzzy models are particularly suitable for handling the variety and veracity of Big Data. Indeed, fuzzy models are based on the concept of *fuzzy logic*, which is a many-valued logic in which the truth value of a variable may assume any real number between 0 and 1. Fuzzy logic is often exploited to express the concept of partial truth, in contrast with the Boolean logic, which assumes that the truth values of a variable may only be 0 and 1. Thus, fuzzy models have the ability to deal with data and information that are vague, uncertain and imprecise. These characteristics are often typical of Big Data.

Since 2014, a number of contributions on fuzzy models for Big Data have been proposed in the literature, focusing on different application fields. Most of them regard the design, implementation and experimentation of fuzzy models for classification [21–30]. There also exist some contributions regarding regression tasks [31–33] and descriptive models, such as fuzzy clustering [34–36], subgroup discovery [37] and the generation of fuzzy association rules [38].

In this paper, we aim to give an overview of distributed algorithms for learning fuzzy models from Big Data, focusing in particular on classification applications. We describe and discuss the most relevant algorithms designed and implemented for generating fuzzy classification models, specifically Fuzzy Rule-Based Classifiers and Fuzzy Decision Trees. We focus on the works in which actual big datasets were used in the experiments. As expected, only distributed versions of algorithms for generating fuzzy models were actually able to deal efficiently with dataset sizes larger than at least 0.5 GB. Thus, in our analysis we considered the distributed versions of the following algorithms:

- The Chi et al. algorithm for generating Fuzzy Rule-Based Classifiers (FRBCs) [21–24].
- Fuzzy Associative Classifiers (FACs) [26].
- Evolutionary Fuzzy Classifiers (EFCs) [27–29].
- Fuzzy Decision Trees (FDTs) [25, 30].

In order to evaluate the performance of the aforementioned algorithms, we selected four popular Big Data classification datasets, whose sizes span up to 8 GB, and ran the algorithms on these datasets in a computer cluster located at the University of Pisa. We compared the achieved results considering not only the accuracy of the models, but also their interpretability. Indeed, although most of the discussed fuzzy classifiers are very accurate, their complexity, in terms of number of rules or number of nodes of the fuzzy trees, is very high. As discussed by Gacto et al. in [39], the greater the complexity, the lower the *interpretability*. Interpretability is a very important feature that characterizes fuzzy models, and assumes a special significance in the context of Big Data, as stated by Fernandez et al. in [19] and by Wang et al. in [40]. Finally, we discuss some scalability properties of the different distributed learning algorithms.

In conclusion, the main objectives and motivations which support this overview are:

- To describe and discuss the most relevant algorithms designed and implemented for generating fuzzy classification models, specifically Fuzzy Rule-Based Classifiers and Fuzzy Decision Trees;
- To highlight both the strengths and the weaknesses of the discussed algorithms;
- To allow the reader to appreciate the fast evolution of theses algorithms, in terms of both their design schemes and frameworks used for implementing and experimenting them in classification tasks for big data;
- To compare these algorithms in terms of the performance obtained on a set of selected real big datasets;
- To give the opportunity to future works of exploiting the discussed approaches in specific engineering and technological applications, by focusing on the most promising ones;
- To suggest possible improvements of the discussed algorithms.

The paper is organized as follows. First, we present the MapReduce paradigm and some recent distributed computing frameworks for handling Big Data. Then, we

describe preliminary concepts about fuzzy systems and we introduce the different fuzzy classification models considered in this paper. Moreover, we briefly analyze the most relevant state-of-the-art sequential algorithms for learning fuzzy classification models. The central sections of the manuscript describe the details of the distributed fuzzy versions of these classification models. The last part of the manuscript includes three sections: in the first section, we discuss the experimental results in terms of accuracy and interpretability. In the second section we point out an in-depth analytic discussion and give some envisions on future research directions. Finally, in the last section, we draw some conclusions.

The MapReduce paradigm and distributed computing frameworks

In this section, we provide a snapshot of the MapReduce paradigm and the main frameworks employed in distributed data mining.

The issues related to Big Data require the adoption of new strategies for elaborating such data. To this aim, the MapReduce programming paradigm [10, 41] was introduced and adopted by Google in 2004, becoming the *de facto* standard for dealing with Big Data analysis. The MapReduce paradigm has been proposed to process huge volumes of data in a scalable way, according to a *divide and conquer* strategy, that consists in breaking down a problem into simpler sub-problems of the same type. The solutions to the sub-problems are then combined in some way to provide a solution to the original problem.

At high level, the paradigm divides the computational flow into two main phases, namely Map and Reduce, organized around $\langle key, value \rangle$ pairs.

When the MapReduce execution environment runs a user program, it automatically partitions the data into a set of independent *chunks* that can be processed in parallel by different nodes. In the Map phase, each task is fed by one chunk of data and, for each $\langle key, value \rangle$ pair as input, it generates a list of intermediate $\langle key, value \rangle$ pairs as output. In the Reduce phase, all the intermediate results are grouped together according to a key-partitioning scheme, so that each Reduce task processes a list of values associated with a specific key as input for generating a new list of values as output. Developers are able to implement parallel algorithms by simply defining Map and Reduce functions. The result of the whole MapReduce process is a set of $\langle key, value \rangle$ pairs produced by all the executed Reduce tasks.

In the last years, several projects have been developed to deal with distributed data storage and elaboration. The most popular is Apache Hadoop [17],[1] which is an open source project created in 2005 and currently maintained by a global community of contributors. The Hadoop framework consists of several modules, among which the most important ones are:

- A distributed storage system, called Hadoop Distributed File System (HDFS), which supports the storage of large datasets on distributed nodes in a cluster;

[1] Apache Hadoop, https://hadoop.apache.org/, accessed: December 2019.

- A parallel processing engine, called Hadoop MapReduce, which was originally intended to implement the MapReduce paradigm, whereas currently it supports multiple processing schemes;
- A set of libraries and utilities, included in Hadoop Common and used by other Hadoop modules;
- A resource-management platform, called Hadoop Yet Another Resource Negotiator (YARN), developed to manage computer resources in clusters and to use them for scheduling users' applications.

As regards data mining tools for Big Data, the library Apache Mahout [42] can be considered as the main contribution based on Apache Hadoop. Mahaout has been the first attempt of producing free implementations of distributed and/or scalable machine learning algorithms for Big Data. It includes many algorithms for machine learning tasks, such as classification, clustering and pattern discovery.

Although Hadoop is widely spread, it suffers from various limitations that make it not suitable for certain types of applications. In particular, it supports batch processing only and employs a two-stage disk-based MapReduce computation engine, which makes it very inefficient in managing real-time data processing and iterative algorithms. In the recent years, to overcome these limitations, Apache Spark and Apache Flink have been proposed.

Apache Spark has been developed to overcome the limitations of Hadoop. It is an open-source distributed general-purpose cluster computing framework with mostly in-memory data processing engine. This key feature allows running most of the computations in memory. Thus, Zaharia et al. in [18, 43] proved that Apache Spark performs faster than Hadoop at least in some specific applications, where the use of iterative algorithms or interactive data mining is required. The core of Spark is the Resilient Distributed Dataset (RDD). RDD is an immutable distributed collection of objects, partitioned across nodes in the cluster, that can be operated in parallel.

Spark is based on a master/slave architecture. A Spark application runs a set of independent processes that use as input the RDD: it consists of a coordinator, the *Driver*, that communicates with multiple distributed workers, the *Executors*. The Driver is in charge of processing the user's main function and activates the *tasks*, which are distributed to the Executors. Executors run the individual tasks in parallel and send the results back to the Driver.

The most popular machine learning library running on Spark is the MLlib library [44],[2] which implements several machine learning and data mining algorithms for clustering, classification, regression, recommendation systems, pattern mining, etc.

Spark is not real-time, but near real-time. This is one of its main limitations, together with problems in dealing with small files, the lack of a dedicated file management system and high memory consumption to run in-memory.

Apache Flink [45][3] tries to overcome these issues. It reduces the complexity, which has been faced by other distributed data-driven frameworks, by integrating query optimization, concepts from database systems and efficient parallel in-memory and out-of-core algorithms, with the MapReduce paradigm.

[2] Apache MLlib, http://spark.apache.org/mllib/, (accessed: December 2019).

[3] Apache Flink, https://flink.apache.org/, accessed: September 2019.

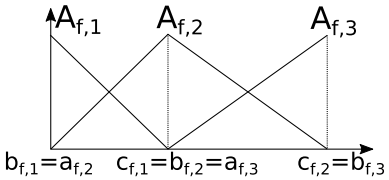

Fig. 1 An example of a strong fuzzy partition

Fuzzy classification models: preliminary concepts, architectures and classical learning algorithms

In this section, we briefly introduce some preliminary concepts regarding fuzzy partitions and the classification model structure of Fuzzy Rule-Based Classifiers (FRBCs), Fuzzy Associative Classifiers (FACs) and Fuzzy Decision Trees (FDTs). Moreover, we also provide an overview of the classical sequential model learning methods. Let $X = \{X_1, \ldots, X_F\}$ be the set of attributes and X_{F+1} be the output of a fuzzy classification model. The output X_{F+1} is a categorical variable assuming values in the set $\Gamma = \{C_1, \ldots, C_k, \ldots, C_K\}$ of K possible classes C_k. Let U_f, with $f = 1, ..., F$, be the universe of the *fth* attribute X_f. In the following, we assume to have available a training set $TS = \{(\mathbf{x}_1, x_{F+1,1}), \ldots, (\mathbf{x}_N, x_{F+1,N})\}$ composed of N input-output pairs..

Fuzzy partitions

Most of the approaches discussed in the next sections employ fuzzy partitions. Let $P_f = \{A_{f,1}, \ldots, A_{f,j}, \ldots, A_{f,T_f}\}$ be a partition of X_f consisting of T_f fuzzy sets $A_{f,j}$. Although there exist several types of fuzzy partitions, *strong* fuzzy partitions [46] are widely used, because they require few parameters for their definition, thus simplifying the modeling process. A strong fuzzy partition is an ordered collection of fuzzy sets, such that:

$$\forall x \in X_f : \sum_{j=1}^{T_f} A_{f,j}(x) = 1 \tag{1}$$

In Fig. 1, we show an example of a strong fuzzy partition composed of three triangular fuzzy sets $A_{f,j}$, whose membership function is defined by the tuples $(a_{f,j}, b_{f,j}, c_{f,j})$, where $a_{f,j}$ and $c_{f,j}$ correspond to the left and right extremes of the support of $A_{f,j}$, and $b_{f,j}$ to the core.

The approaches described in the following adopt strong fuzzy partitions of the attributes.

Fuzzy Rule-Based Classifiers

An FRBC includes a Rule Base (RB), a Data Base (DB) containing the definition of the fuzzy sets used in the RB, and a reasoning method. An RB is composed of M rules expressed as:

$$\begin{aligned} R_m\text{:}&\textbf{IF } X_1 \textbf{ is } A_{1,j_{m,1}} \textbf{ AND} \ldots \textbf{AND } X_F \textbf{ is } A_{F,j_{m,F}} \\ &\textbf{THEN } X_{F+1} \textbf{ is } C_{j_m} \textit{ with } RW_m \end{aligned} \tag{2}$$

where C_{j_m} is the class label associated with the *mth* rule, and RW_m is the rule weight, i.e., a certainty degree of the classification in the class C_{j_m} for a pattern belonging to the subspace delimited by the antecedent of rule R_m.

Given an input pattern $\hat{\mathbf{x}} \in \mathbb{R}^F$, being $\mathbb{R}^F$ an F-dimensional real space, the strength of activation (*matching degree* of the rule with the input) of the rule R_m is usually computed as:

$$w_m(\hat{\mathbf{x}}) = \prod_{f=1}^{F} A_{f,j_{m,f}}(\hat{x}_f), \tag{3}$$

where $A_{f,j_{m,f}}(\hat{x}_f)$ is the membership value of $\hat{x}_f$ associated with the fuzzy set $A_{f,j_{m,f}}$. In this case, we have considered the product as t-norm for implementing the logical conjunction in the antecedent of the rule expression in (2).

Two different definitions of rule weight RW_m are commonly found in the literature [47]:

1. The certainty factor:

$$CF_m = \frac{\sum_{\mathbf{x}_t \in C_{j_m}} w_m(\mathbf{x}_t)}{\sum_{t=1}^{N} w_m(\mathbf{x}_t)}. \tag{4}$$

2. The penalized certainty factor:

$$PCF_m = CF_m - \frac{\sum_{\mathbf{x}_t \notin C_{j_m}} w_m(\mathbf{x}_t)}{\sum_{t=1}^{N} w_m(\mathbf{x}_t)}. \tag{5}$$

As regards the algorithms discussed in the following, we highlight that the different versions of the distributed Chi et al. algorithm adopt the penalized certainty factor, while the Distributed Fuzzy Associative Classifiers adopt the certainty factor. Finally, the distributed multi-objective evolutionary classifiers adopt no weights. A specific reasoning method uses the information from the RB and DB to determine the class label for a given input pattern. Details on different types of rule weights and reasoning methods used in the literature can be found in the contribution of Cordon et al. in [47].

The RB and the DB of an FRBC can be generated adopting different algorithms, such as the Chi et al. algorithm [48] and the Antonelli et al. evolutionary-based algorithms [49].

Specifically, the RB design process aims to determine the optimal set of rules for managing the classification problem. The DB design process consists of finding the appropriate number of fuzzy sets for each attribute and their parameters. The objective of the design process is to concurrently maximize the classification accuracy and, possibly, the model interpretability.

As regards the Chi et al. algorithm [48], it is one of the first heuristics adopted for generating the RB of an FRBC: given a pre-defined DB describing the fuzzy partitions of each attribute, the algorithm generates a rule for each training pattern. The antecedent of a rule is generated considering the list of fuzzy sets, which have been activated by a certain training input pattern with the highest membership degree. The consequent is

directly specified by the class of the training pattern. Duplicated rules are removed and appropriate strategies have been defined for handling rules with the same antecedent and different consequents and for associating a weight with each rule.

As discussed by Fernandez et al. in [50], Evolutionary Fuzzy Systems (EFSs) are well known *hybrid models*, which exploit evolutionary algorithms (EAs) for learning the parameters of fuzzy models. EAs are able to solve optimization tasks by imitating some aspects of natural evolution [51]. Learning the RB and the DB of an FRBC can be considered as an optimization process, where the accuracy of the final model is usually the fitness function to be optimized. However, in the last decades, *interpretability* of the fuzzy models has been often taken into account concurrently with the accuracy, leading to the so-called Multi-Objective Evolutionary Fuzzy Systems (MOEFSs): more details on MOEFSs can be found in the works of Ducange et al. and of Fazzolari et al. [52, 53]. MOEFSs adopt multi-objective evolutionary algorithms [54], which aim to concurrently maximize both the accuracy and the interpretability during the evolutionary learning process of fuzzy models. In the last decade, several MOEFSs have been successfully experimented for selecting (Ishibuchi et al. in [55]) or learning (Cococcioni et al. in [56]) the set of rules, for optimizing the fuzzy partitions (Botta et al. in [57]) and for concurrently learning the RB and the DB of FRBCs (Antonelli et al. in [49] and Fazzolari et al. in [58]). We recall that MOEFSs return a set of solutions characterized by different trade-offs between accuracy and interpretability.

Interpretability regards the capability of explaining how decisions have been taken, using terms understandable to humans. Thus, the simplicity of the fuzzy reasoning method, adopted to deduce conclusions from facts and rules, assumes a special importance. Moreover, the intepretability is strictly related to the *transparency* of the model, namely to the capability of understanding the structure of the model itself. Fuzzy models, especially FRBCs, can be characterized by a high transparency level, whenever the linguistic RB is composed of a reduced number of rules and conditions and the fuzzy partitions have a good *integrity*. The integrity of fuzzy partitions depends on some properties, such as coverage, distinguishability and normality [59]. Several measures have been proposed in the specialized literature for evaluating the interpretability of an FRBC, taking into consideration semantic and complexity aspects of both the RB and the DB (check the contribution of Gacto et al. in [39]).

Fuzzy Associative Classifiers

As discussed by Baralis et al. in [60] and by Abdelhamid et al. in [61], Associative Classifiers (ACs) integrate a *frequent pattern mining* algorithm and a *rule-based classifier* into a single system. Specifically, first, frequent patterns are extracted from the dataset using an appropriate mining algorithm. In the classification context, a pattern consists of set of items, where one item is a class. Thereafter, classification rules are generated from the frequent patterns, and pruned according to their support, confidence, and redundancy. In the fuzzy context, for each fuzzy partition P_f of an attribute X_f, the single item is defined as the couple $(X_f, A_{f,j})$, where $A_{f,j}$ is the j-th fuzzy set defined in P_f.

The *mth* Fuzzy Classification Association Rule ($FCAR_m$) out of the Rule Base $RB = \{FCAR_1, \ldots, FCAR_M\}$ is expressed as

$$FCAR_m{:}FAnt_m \rightarrow C_{j_m} \text{ with } RW_m \tag{6}$$

where the consequent C_{j_m} is the class label selected for the rule, the antecedent $FAnt_m$ is equal to the antecedent of the rule expression in (2) and RW_m is the weight of the rule. Thus, the FCARs that can be generated by using a fuzzy association rule mining approach are the same as the ones of classical FRBCs. Examples of classification models based on FCARs can be found in [62, 63]. In [62], Alcala et al. discuss the use of a fuzzy version of the Apriori algorithm [64] for generating an initial set of *FCARs*. Then, a single-objective genetic algorithm is adopted for selecting the most relevant rules along with the optimization of the fuzzy set parameters. In [63], Segatori et al. introduce a fuzzy extension of the well known FP-Growth [65], which allows them to quickly mine a set of *FCARs* characterized by a high confidence level. In order to reduce the number of rules in the final RB, an *FCARs* pruning process, based on redundancy and training set coverage, is also applied after the first mining step.

Fuzzy Decision Trees

A decision tree is a directed acyclic graph, where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of the test, and each leaf (or terminal) node holds one or more class labels. The topmost node is the root node. In general, each leaf node is labeled with one or more classes $C_k \in \Gamma$ with an associated weight w_k: weight w_k determines the strength of class C_k in the leaf node [66]. Given a training set *TS*, the structure of a decision tree, in terms of nodes, branches and leaves, is usually generated using a recursive scheme. First of all, one of the attributes is selected in the decision node corresponding to the root, taking the overall *TS* into consideration. The attribute selection algorithm returns also a set of branches and corresponding nodes. For each node, a new attribute is selected from the set of the attributes, considering only the instances of the *TS*, which satisfy the test associated with the branch. When no attribute can be selected, the node is denoted as a leaf node. The attribute selection algorithm is usually based on a specific metric, such as Gini Index and Information Gain. More details regarding decision trees can be found in the contribution of Quinlan in [66].

In [67], Altay et al. discuss some fuzzy extensions of the classical ID3 and SLIQ algorithms [68]. Recently, an incremental algorithm for learning FDT, based on the concept of Fuzzy Hoeffding Bound and Fuzzy Information Gain, was presented by Pecori et al. in [69]. Some preliminary experiments have been discussed by the authors: very promising results in the context of data streams classification have been achieved.

In this paper, we adopt the distributed implementation proposed by Segatori et al. in [25] of an algorithm for learning a decision tree based on fuzzy information gain: each attribute is preliminarily partitioned by using strong fuzzy partitions. Figure 2 shows an example of multi-way FDT, in which we consider two attributes. Each attribute is partitioned with three fuzzy sets. A test branch is always generated for each fuzzy set of the input variable involved in a test node.

Once the tree has been generated, a given unlabeled instance $\hat{\mathbf{x}}$ is assigned to a class $C_k \in \Gamma$ by following the activation of nodes from the root to one or more leaves. In classical decision trees, each node represents a crisp set and each leaf is labeled with a

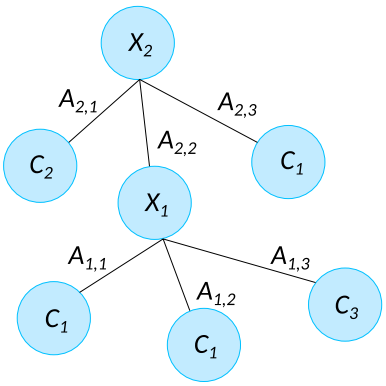

Fig. 2 An example of multi-way Fuzzy Decision Tree

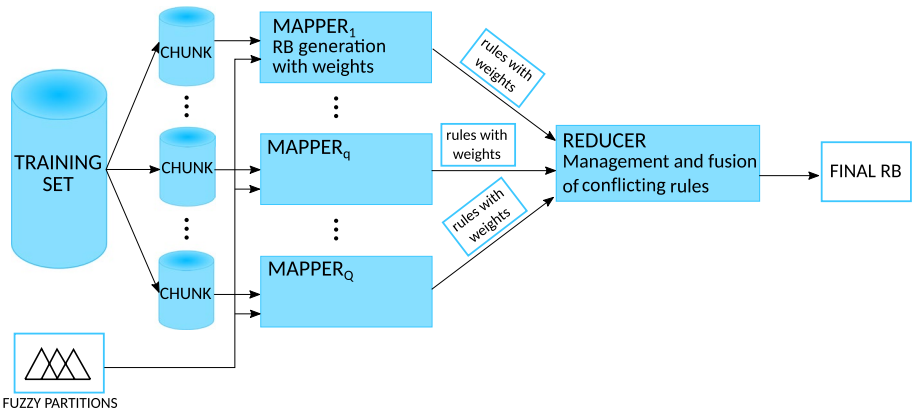

Fig. 3 MapReduce scheme of the Chi-FRBCS-BigData algorithm

unique class label. It follows that $\hat{\mathbf{x}}$ activates a unique path and is assigned to a unique class. In FDT, each node represents a fuzzy set. Thus, $\hat{\mathbf{x}}$ can activate multiple paths in the tree, reaching more than one leaf with different strengths of activation, called *matching degrees*. Details on how determine the output class label of a given unlabeled instance, using an FDT, can be found in the work of Segatori et al. [25].

Local and global implementations of the distributed Chi et al. algorithm

Lopez et al. discussed in [22] the first attempt of extending an algorithm for generating FRBCs for Big Data to a distributed computing venue. Here, the authors discuss the design and the implementation of a distributed version of the well-known Chi et al. algorithm [48]. This algorithm has been developed adopting the MapReduce programming paradigm under the Hadoop framework. Figure 3 gives a snapshot of the implementation scheme: according to the MapReduce paradigm, the training dataset is split into chunks which feed the mappers. Each mapper generates an RB from the specific chunk of the training data, using the classical procedure of the Chi et al. algorithm. Then, a single reducer combines these RBs for generating the final RB. Two strategies have been proposed for solving the problem of conflicting rules: both strategies search for rules with the same antecedent. For each set of rules with the same antecedent, the first

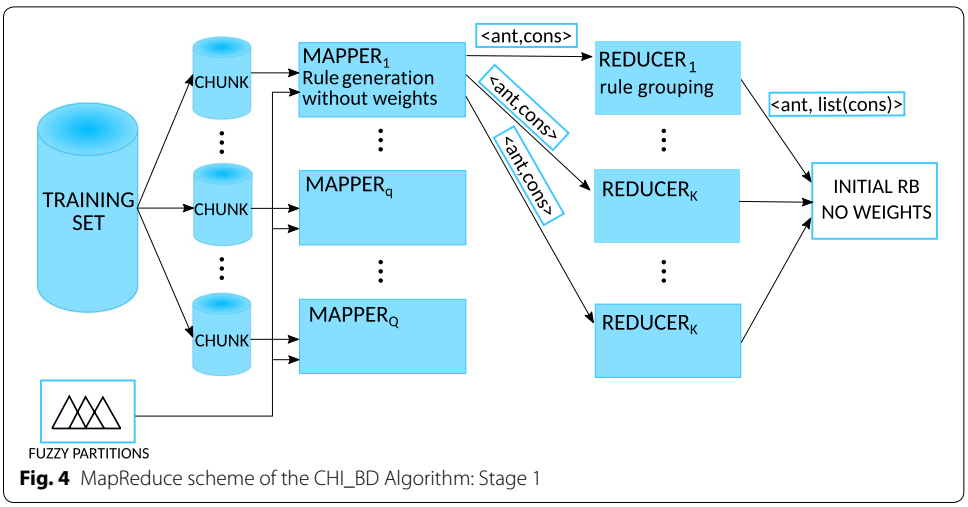

Fig. 4 MapReduce scheme of the CHI_BD Algorithm: Stage 1

approach retains the rule with the highest weight. The second one calculates the average weight of the rules that have the same consequent. Finally, the rule with the highest average weight is kept in the final RB. The algorithm discussed above is labeled as Chi-FRBCS-BigData and represents a *local* distributed implementation of the Chi et al. algorithm. Indeed, each mapper generates an RB using only the subset of instances processed by that specific mapper. Thus, the rule weights widely depend on the proportion and distribution of the classes in the specific subset of training instances. Different RBs can be generated considering different training dataset partitions and different number of mappers. Lopez et al. and Fernandez et al., also experimented the Chi-FRBCS-BigData considering imbalanced classification datasets and analyzing the effects of different granularities of the fuzzy partitions, respectively, in [23, 70].

Recently, an optimized version of the distributed Chi et al. algorithm, denoted as CHI_BD, has been proposed by Elkano et al. in [24]. The optimization regards both the generation of the rules and the architecture of the distributed execution scheme. Figures 4 and 5 resume the MapReduce stages adopted for the improved implementation of the distributed Chi et al. algorithm.

In Stage 1, an initial RB, which can contain rules with the same antecedent and different consequents, is generated without rule weights. Each mapper generates a pair $< antecedent, consequent >$ ($< ant, cons >$ in Fig. 4) for each pattern included in its own *TS* chunk. The pairs generated by all the mappers feed the reducers, which group together all the pairs for generating the initial RB without weights.

Stage 2 generates the final RB, composed of weighted classification rules. To this aim, each mapper loads its training data chunk and also the initial RB generated in Stage 1. Each mapper calculates the matching degree of each training pattern of the chunk. In the reduce phase, for each rule, a reducer sums up the matching degrees generated for the specific rule by the different mappers. Thereafter, the weights for each consequent are calculated, ensuring that the overall *TS* contributes to their values. Only the consequent associated with the highest weight is retained in the final RB.

CHI_BD represents the *global* counterpart of the local implementation discussed by Lopez et al. in [22]. Indeed, as stated before, the CHI_BD algorithm ensures that the rule

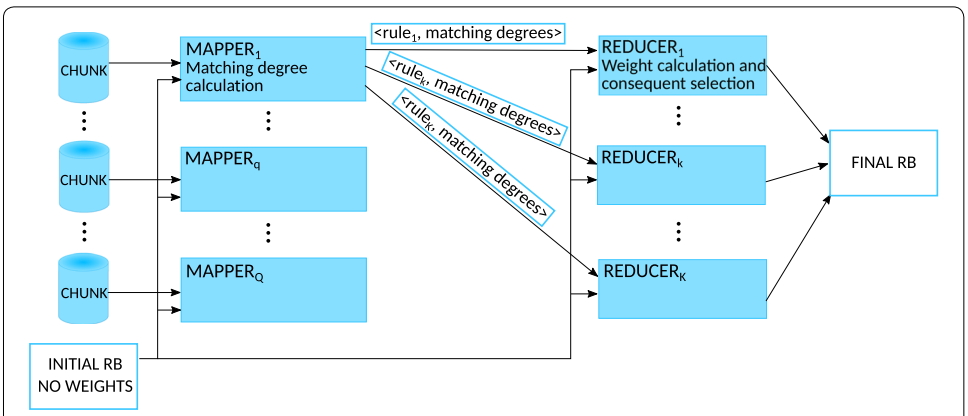

Fig. 5 MapReduce scheme of the CHI_BD Algorithm: Stage 2

weights are calculated considering the overall *TS* and that their values do not depend on the data partitions and on the number of adopted mappers. Also the CHI_BD algorithm has been developed under the Hadoop framework.

Distributed Fuzzy Associative Classifiers

The recent contribution discussed by Segatori et al. in [26] introduces a novel distributed algorithm for generating FRBCs based on ACs. The algorithm discussed in the following, labeled as Distributed Fuzzy Associative Classifier based on Fuzzy Frequent Pattern (DFAC-FFP) mining, adopts the fuzzy definition of *support* and *confidence* to determine the strength of a classification rule. For a generic $FCAR_m$, fuzzy support and confidence are defined as:

$$\text{fuzzySupp}\,(FAnt_m \rightarrow C_{j_m}) = \frac{\sum_{\mathbf{x}_n} \in TS_{j_m} w_m(\mathbf{x}_n)}{N} \tag{7}$$

$$\text{fuzzyConf}\,(FAnt_m \rightarrow C_{j_m}) = \frac{\sum_{\mathbf{x_n}} \in TS_{j_m} w_m(\mathbf{x}_n)}{\sum_{\mathbf{x}_n} \in TS\; w_{FAnt_m}(\mathbf{x}_n)} \tag{8}$$

where $TS_{j_m} = \{\mathbf{x}_n \mid (\mathbf{x}_n, y_n) \in TS, y_n = C_{j_m}\}$ is the set of *TS* instances labelled with class C_{j_m}, $w_m(\mathbf{x}_n)$ is the matching degree, as defined in formula (3), of rule $FCAR_m$, and $w_{FAnt_m}(\mathbf{x}_n)$ is the matching degree of all the rules whose antecedent is equal to $FAnt_m$.

In order to generate a set of FCARs, the following procedures are sequentially executed during the DFAC-FFP learning process:

- Distributed fuzzy partitioning: A strong fuzzy partition is directly generated on each continuous attribute using a distributed approach based on fuzzy entropy;
- Distributed Fuzzy Classification Association Rule (FCAR) Mining: A distributed fuzzy frequent pattern mining algorithm extracts frequent FCARs with confidence and support higher than a given threshold;
- Distributed FCAR pruning: The mined FCARs are pruned by means of two distributed rule pruning phases based on redundancy and training set coverage.

For the sake of brevity, in this work we omit the description of the implementation of the distributed fuzzy partitioning algorithm. Details on this algorithm can be found in [26]. However, the distributed FCAR mining and pruning procedures may be applied whenever an initial partition P_f for each attribute X_f has been previously defined.

Each stage of the DFAC-FFP has been implemented using the MapReduce paradigm on the Apache Spark framework.

Figure 6 shows the three MapReduce stages of distributed FCAR mining, namely *distributed fuzzy counting*, *distributed fuzzy FP-growth* and *distributed rule selection* stages.

The first MapReduce stage takes as inputs the fuzzy partitions and the *TS* chunks, and outputs a list of fuzzy sets $A_{f,j}$ whose fuzzy support is larger than threshold *minSupp*. In detail, each mapper produces, for each fuzzy set $A_{f,j}$ of each fuzzy partition P_f, the list of membership degrees $\mu_{f,j}(x_{r,f})$ calculated for each r^{th} input pattern of the specific *TS* chunk. Each reducer receives in input a fuzzy set $A_{f,j}$ and the corresponding list of membership degrees, calculated by each mapper, and calculates the fuzzy support as follows:

$$\text{fuzzySupp}\,(A_{f,j}) = \frac{\sum_{t=1}^{N} \mu_{f,j}(x_{t,f})}{N} \tag{9}$$

where N is the total number of instances of the *TS*. Only the fuzzy sets whose fuzzy support is higher than *minSupp* are retained and included in the list of frequent fuzzy items.

The second MapReduce stage is based on a distributed version of the Fuzzy FP-growth algorithm. FP-growth is a well known frequent pattern mining algorithm, introduced by Han et al. in [65], which allows handling high dimensional datasets. Indeed, it first extracts the frequent items, and sorts them by descending frequencies. Thereafter, such a dataset of frequent items is compressed into a frequent pattern tree, called *FP-tree*. Finally, frequent patterns are recursively mined by extracting from the FP-tree a set of projected datasets, each one associated with a frequent item or a pattern fragment. In the work discussed in [26], Segatori et al. proposed a distributed implementation of the Fuzzy FP-growth algorithm introduced by the same authors in [63]. Each mapper takes in input a chunk of the training set and the list of frequent fuzzy items $A_{f,j}$ and outputs *item-projected objects*. These objects feed reducers, which first build the *item-projected datasets*, and then generate local conditional FP-trees. From the local conditional FP-trees, FCARs are mined, retaining only the ones whose support, confidence and χ^2 values exceed the relative thresholds.

The last MapReduce stage is in charge of selecting, from the set of very specialized FCARs mined by the Fuzzy FP-growth algorithm, the top H non-redundant FCARs per class. To this aim, each mapper is fed by a block of FCARs previously generated and outputs pairs containing the rule and the consequent class. Each reducer processes all the rules with the same class label, outputting the most relevant ones. Details on item-projected objects and datasets and on the FCAR relevance measures can be found in the work of Segatori et al. in [63].

Figure 7 shows the two MapReduce stages of the distributed FCAR pruning.

In the first stage, the pruning of the set of FCARs is carried out on the basis of fuzzy support and fuzzy confidence thresholds. Each mapper is fed by a *TS* chunk and by the list of FCARs generated during the distributed FCAR mining approach. For each $FCAR_m$

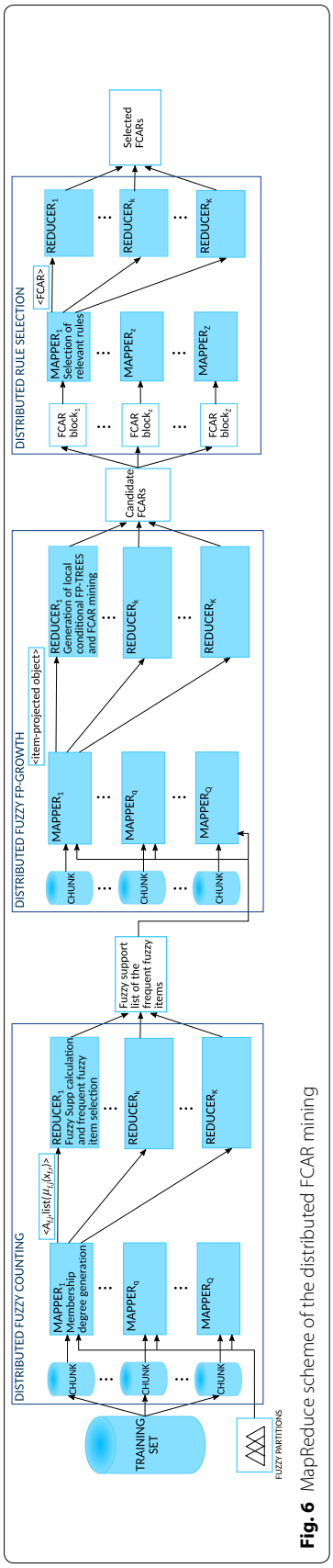

Fig. 6 MapReduce scheme of the distributed FCAR mining

and for each input pattern x_r in the TS chunk, the matching degree $w_m(x_r)$ is calculated and produced as output of each mapper. Each reducer calculates the actual fuzzy support and confidence of a specific $FCAR_m$, given the list of the $w_m(x_r)$ computed by the different mappers. Only the rules, whose values of fuzzy support and confidence are higher than specific thresholds, are retained and taken into consideration for the next stage. In the second MapReduce stage, only the rules characterized by a training coverage higher than a threshold are inserted into the final RB. Each mapper is fed by a training data chunk and by the set of rules previously generated: it calculates the training set coverage and returns the most covered rules (see [26] for more details). The reducers generate the final RB considering only the rules, identified by the mappers, which satisfy the coverage criteria.

Distributed Evolutionary Fuzzy Systems

To the best of our knowledge, the first distributed version of an EFS can be found in [28], where Fernandez et al. discuss a distributed evolutionary rule selection approach. Specifically, this approach is carried out inside each mapper of the Chi-FRBCS-BigData, previously introduced by the authors and discussed in "Local and global implementations of the distributed Chi et al. algorithm" section. Indeed, in each mapper, after the generation of rules by means of the classical Chi et al. algorithm, a single objective evolutionary algorithm selects the most relevant rules. The optimized fitness function is a linear combination of the accuracy and the measure of complexity of the RB. The algorithm is implemented under the Apache Spark framework and experimented considering imbalanced classification datasets.

As discussed by Fernandez et al. in [19] and by Wang et al. in [40], in the context of Big Data, the interpretability of fuzzy models assumes a special relevance. If an interpretable and transparent model can be derived from a big dataset, the model itself may be considered as a sort of "visualization tool", which may allow us to understand the phenomena hidden behind the data. Thus, in 2017, the first distributed MOEFS for Big Data classification was discussed by Ferranti et al. in [27]. The algorithm, denoted as Distributed Pareto Archived Evolution Strategy with Rule and Condition Selection (DPAES-RCS) is a distributed implementation on the Apache Spark environment of the Pareto Archived Evolution Strategy with Rule and Condition Selection (PAES-RCS), introduced by Antonelli et al. in [49]. PAES-RCS learns the RB of a set of FRBCs through a Rule and Condition Selection (RCS) strategy: an initial set of rules is generated by means of heuristics, such as the Chi et al. algorithm or a decision tree, and the most relevant rules and conditions are selected during the evolutionary learning process. In DPAES-RCS the initial set of rules is generated exploiting a distributed version of the C4.5 algorithm [66], available in the MLib of Spark, which is a well known algorithm for the generation of classical decision trees. The parameters of the fuzzy sets are learnt concurrently with the RB. Recently, a novel version of DPAES-RCS, called DPAES-FDT-GL has been presented by Barsacchi et al. in [29]. Here, the initial rule set is generated adopting the distributed FDT discussed in "Distributed Fuzzy Decision Trees" section and introduced by Segatori et al. in [26]. Moreover, during the evolutionary learning process also the granularity of the fuzzy partitions is concurrently learnt with the RB and the parameters of the fuzzy sets.

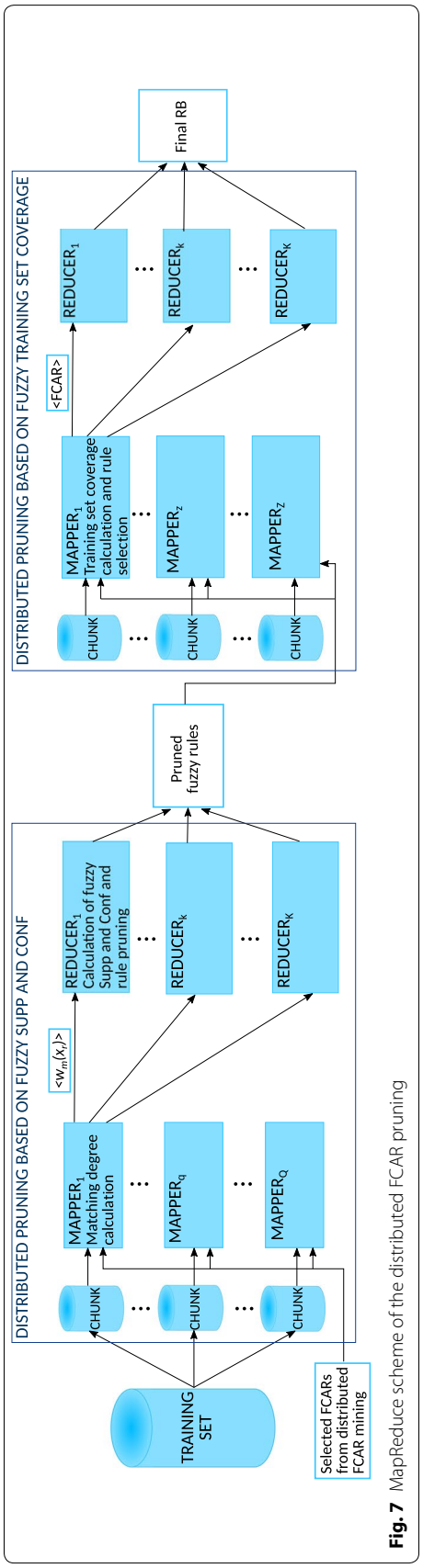

Fig. 7 MapReduce scheme of the distributed FCAR pruning

Figure 8 shows the scheme of the distributed learning process by means of PAES-RCS. Once the initial set of rules has been learned, for instance using a distributed FDT learning algorithm as discussed in "Distributed Fuzzy Decision Trees" section, two solutions are randomly generated and inserted into an archive of non-dominated solutions. The RBs of the two initial solutions contain a random number of rules. Moreover, for each rule, a random number of conditions is selected. The fuzzy partitions are strong fuzzy partitions composed of a random number of fuzzy sets (between a minimum and a maximum value) and a random distribution of the cores along the universe of definition. At each iteration, two solutions are randomly selected from the archive and mating operators, namely crossover and mutation operators, are applied for generating two offspring solutions. A two dimensional fitness function is calculated for each offspring, considering the interpretability, in terms of total number of conditions in the RB, and the accuracy of the FRBCs associated with each solution. All the previously discussed steps of the algorithm and the calculation of the interpretability can be easily executed by a sequential driver program. On the other hand, the accuracy needs to be calculated by using a distributed approach, when the amount of data is very huge. Indeed, the overall *TS* must be scanned for computing the accuracy. Thus, the *TS* is divided into chunks and a number of Computing Units (CUs) are in charge of calculating the output of the classifier, associated with each offspring, and return the number of patterns correctly classified. A driver program collects the results provided by each CU and calculates the accuracy. Finally, the driver program updates the archive of non-dominated solutions considering the two offspring solutions. The multi-objective evolutionary learning scheme terminates when a stopping condition is reached (usually, a maximum number of fitness evaluations is fixed and adopted as stopping condition). The final archive contains a set of non-dominated FRBC characterized by different trade-offs between accuracy and interpretability.

More details regarding the chromosome coding, the mating operators and the PAES-RCS learning scheme can be found in the contributions of Ferranti et al. in [27] and of Barsacchi et al. in [29].

Distributed Fuzzy Decision Trees

As discussed in "Fuzzy classification models: preliminary concepts, architectures and classical learning algorithms" section , an FDT can be recursively generated considering an initial partition P_f for each input variable X_f. In the work of Segatori et al. in [25], authors discuss a distributed implementation of an FDT learning scheme, considering the MapReduce paradigm under the Apache Spark framework. Similar to DFAC-FFP, the proposed algorithm includes an initial distributed fuzzy discretization step for each input variable, based on fuzzy entropy. Also in this case, we skip the description of the initial fuzzy discretization. On the other hand, the distributed FDT learning scheme discussed in the following can be applied whenever an initial strong fuzzy partition is available for each input variable.

Two types of FDTs are considered by the authors, namely multi-way and binary decision trees. As regards multi-way decision trees (see Fig. 2), the splitting points are generated considering a branch for each fuzzy set of the selected attribute. On the other hand, binary trees consider just a two-way splitting point for the specific

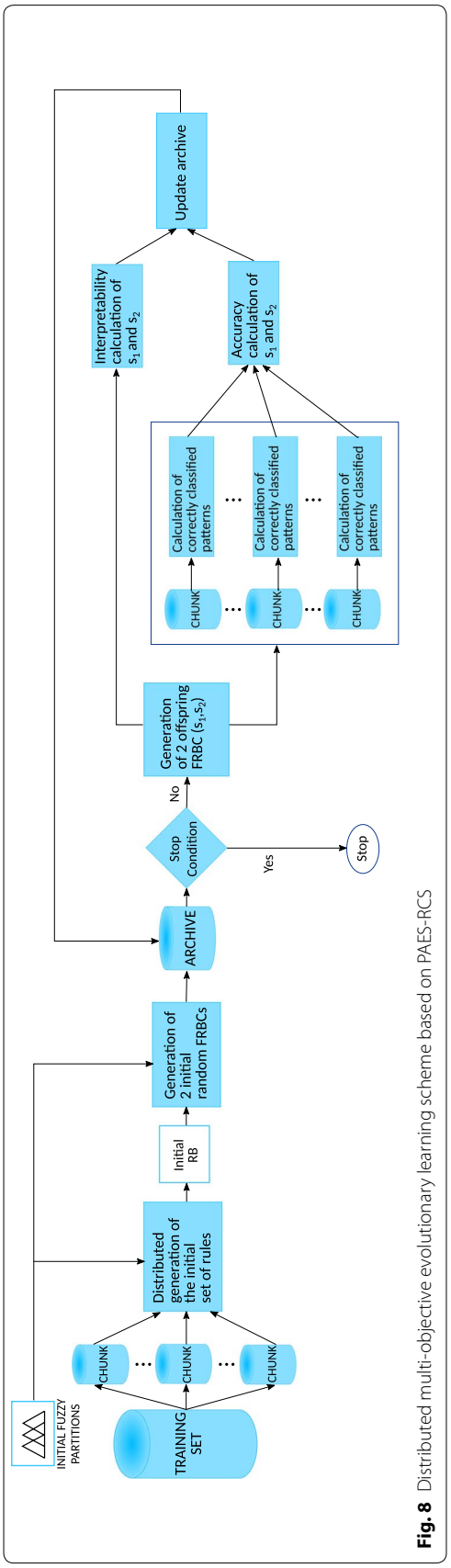

Fig. 8 Distributed multi-objective evolutionary learning scheme based on PAES-RCS

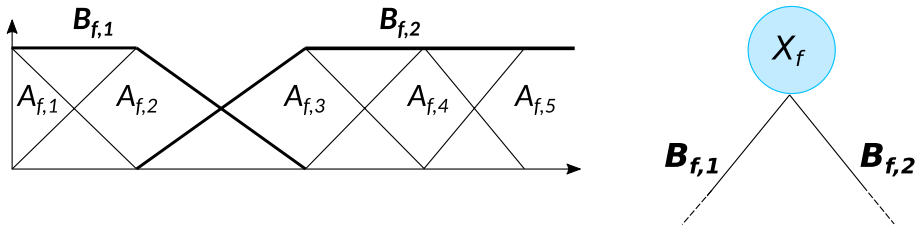

Fig. 9 An example of fuzzy binary splitting

selected variable. In this case, as shown in Fig. 9, for the selected attribute the splitting point is identified considering two fuzzy sets created by applying the union operator among the fuzzy sets of the specific partition. The best splitting point is directly generated by the attribute selection algorithm. Both for multi-way and binary decision trees, the attribute selection algorithm is based on the fuzzy information gain. More details on fuzzy entropy, fuzzy information gain and the distributed fuzzy discretization can be found in the paper of Segatori et al. [25].

Figure 10 shows the scheme of the MapReduce implementation of the FDT learning approach. The scheme is valid both for multi-way and binary decision trees. A MapReduce stage is re-iterated for the identification of the attributes to be selected and of their splitting points (in the case of binary splitting). At each iteration the set of nodes identified at the previous iteration is taken into consideration (set R of nodes) and, for each of them, the selected attribute and the splitting points are returned. Each mapper is fed by a TS chunk and by the list of nodes to be split. For each node, each mapper calculates a vector of statistics, considering the contribution of the handled TS chunk. The statistics calculated by the mappers are then used by the reducers for calculating the fuzzy information gain, selecting the new input variables and determining the splitting points. The algorithm stops when the set R of current nodes is composed only by leaves. A node is identified as a leaf if the following conditions are satisfied [25]: (i) the node contains only instances of the same class, (ii) the node contains a number of instances lower than a fixed threshold, (iii) the tree has reached a maximum fixed depth and (iv) the fuzzy information gain is lower than a fixed threshold. Recently, an improved fuzzy partitioning algorithm, which exploit the probability integral transform, has been introduced in [30]. This new partitioning algorithm allows reducing the complexity of the multi-way decision trees generated using the distributed FDT learning scheme discussed above. This approach has been labeled as FMDT_l, where l is the number of partitions considered for each attribute.

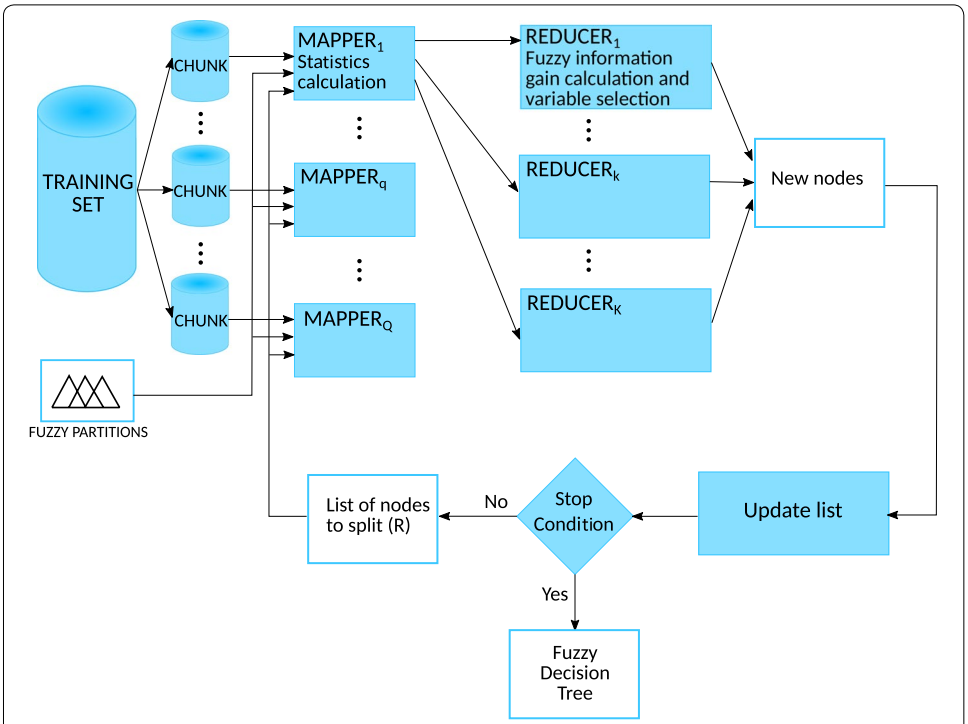

Fig. 10 MapReduce scheme of the distributed FDT learning approach

Experimental results: some discussions

The distributed implementations of the fuzzy classification models discussed in the previous sections have been experimented by their authors on a number of public benchmark datasets. Most of these datasets can be retrieved from the UCI[4] and the LIBSVM[5] repositories.

In order to compare the results achieved by the different distributed learning algorithms discussed so far, we performed a number of experiments, adopting the source codes publicly available and the ones developed at the University of Pisa. Table 1 summarizes the algorithms analyzed in this work, specifying their names, their acronyms and the reference papers. We skipped the single-objective EFS because it was designed for imbalanced binary datasets. As regards the values of the parameters of each specific algorithm, we adopted the best setup suggested in the paper in which each algorithm was introduced and experimented.

We executed all the algorithms on the same cluster located at the University of Pisa. The cluster consists of one master equipped with a 4-core CPU (Intel Core i5 CPU 750 × 2.67 GHz), 8 GB of RAM and a 500 GB Hard Drive, and four slave nodes equipped with a 4-core CPU with Hyperthreading (Intel Core i7-2600K CPU × 3.40 GHz, 8 threads), 16 GB of RAM and a 1 TB Hard Drive. All nodes are connected by a Gigabit Ethernet (1

[4] Available at https://archive.ics.uci.edu/ml/datasets.php.

[5] Available at www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Table 1 Algorithms used in the experimental comparison

Name	Acronym	References
Local distributed Chi for Big Data	Chi-FRBCS-BigData	[22, 23, 70]
Global distributed Chi for Big Data	CHI_BD	[24]
Distributed Fuzzy Associative Classifier based on Fuzzy Frequent Pattern	DFAC-FFP	[26]
Distributed PAES with Rule and Condition Selection	DPAES-RCS	[27]
Distributed PAES with Fuzzy Decision Tree and Granularity Learning	DPAES-FDT-GL	[29]
Multi-way Fuzzy Decison Tree	Multi-way FDT	[25]
Multi-way Fuzzy Decison Tree with Improved Fuzzy Partitioning	$FMDT_I$	[30]

Gbps) and run Ubuntu 12.04. The training sets were stored in the HDFS (Hadoop Distributed File System).

In Table 2 we show four datasets that have been considered in our experimental analyses and that have been also discussed in most of the papers where the models have been proposed. The chosen datasets represent three different categories, namely datasets with only real-valued attributes (Susy and Higgs), with both real-valued and categorical attributes (KDD) and with only categorical values (Poker Hand). Moreover, these datasets are characterized by different numbers of input/output instances (up to 11 millions) and input variables (from 10 to 41). Furthermore, for each dataset, the size in terms of memory occupancy (up to 8.04 GB) is also reported in Table 2.

Tables 3 and 4 show the results achieved by the classifiers considered in this paper on the four selected big datasets. We recall that both SUS and HIG datasets are balanced binary classification datasets. KDD is a slightly imbalanced classification dataset, where the minority class is distributed in more or less 20% of the instances, thus the percentage of correctly classified instances can be considered as a good accuracy measure. We also experimentally verified that the minority class instances in the training set folds are enough to induce the generation of classifiers able to accurately distinguish the two classes. As regards the interpretability measure, we adopt the number of rules for the FRBCs, generated by means of Chi-FRBCS-BigData, Chi_BD, DFAC-FFP, DPAES-RCS and DPAES-FDT-GL, and the number of leaves for the FDTs (we considered just multi-way FDTs).

In the tables, we show the average results of a fivefold stratified cross validation.

As shown in Table 3 and Figure 11, on the KDD dataset all the algorithms perform well in terms of accuracy. On the other datasets, the FDTs perform always better than the FRBCs. Regarding FRBCs, both the local and global versions of the distributed Chi et. al algorithm achieve the worst results. On Sus dataset, the accuracies achieved

Table 2 Datasets used in the experimental comparison

Datasets				
Name	**# Instances**	**# Attributes**	**# Classes**	**# Size**
Higgs (HIG)	11,000,000	28 (real: 28)	2	8.04 GB
Kddcup 2 (KDD_2)	4,856,151	41 (real: 26, cat: 15)	2	476 MB
Susy (SUS)	5,000,000	18 (real: 18)	2	2.4 GB
Poker Hand (POK)	10,000,000	10 (cat: 10)	10	24.6 MB

Table 3 Experimental comparison: average accuracies on the test set

Algorithm	Dataset			
	HIG	SUS	KDD	POK
Chi-FRBCS-BigData	55.89	55.75	99.93	51.78
CHI_BD	57.81	65.36	99.91	52.22
DFAC-FFP	66.00	78.26	99.99	76.65
DPAES-RCS	65.00	78.12	99.94	60.22
DPAES-FDT-GL	65.03	78.60	99.88	61.80
Multi-way FDT	71.25	79.63	99.98	77.17
$FMDT_5$	72.32	78.97	99.98	76.21

Table 4 Experimental comparison: average complexities

Algorithm	Dataset			
	HIG	SUS	KDD	POK
Chi-FRBCS-BigData	24,058	678	1020	813,193
CHI_BD	624,358	9355	5498	52,652
DFAC-FFP	9365	10,970	890	5712
DPAES-RCS	30.2	28	21.8	50
DPAES-FDT-GL	14	14.6	10.8	41.6
Multi-way FDT	920,942	758,064	630	28,561
$FMDT_5$	2987	2865	96	1873

by the remaining algorithms are similar. On HIG dataset, DFAC-FFP and DPAESs achieve accuracies up to 5% lower than the ones achieved by FDTs. On POK dataset, DFAC-FFP achieves accuracies similar to FDTs, while the accuracies of the DPAESs are considerably lower than the ones obtained by FDTs, although higher than the ones achieved by Chi-FRBCS-BigData and CHI_BD algorithms.

As regards the complexities, we have to consider that the high accuracies of the FDTs are supported by very complex models, constituted by a huge number of parameters (number of leaves). Especially for the multi-way FDT, the generated models are up to four orders of magnitude more complex than the one generated by DPAES-RCS and DPAES-FDT-GL. Even though the complexities associated with the models generated by $FMDT_5$ are lower than the ones generated by the multi-way FDT, DPAES-RCS and DPAES-FDT-GL result to be the most interpretable models for Big Data classification tasks. Moreover, DPAES-RCS and DPAES-FDT-GL achieve results similar to DFAC-FFP (except for the POK dataset), with a complexity smaller by two orders of magnitude. More details on the interpretability of DPAES-RCS and DPAES-FDT-GL can be found in the works of Ferranti et al. [27] and of Barsacchi et al. [29]. In these two papers, some examples of actual interpretable and transparent RBs and DBs, regarding real world classification problems, are shown and discussed in depth.

It is worth noticing that, in some of the papers in which the algorithms discussed in this paper have been introduced, some comparisons with non-fuzzy classification models have been carried out. For instance, Segatori et al. in [26] demonstrate that DFAC-FFP generates models characterized by accuracies similar to the ones

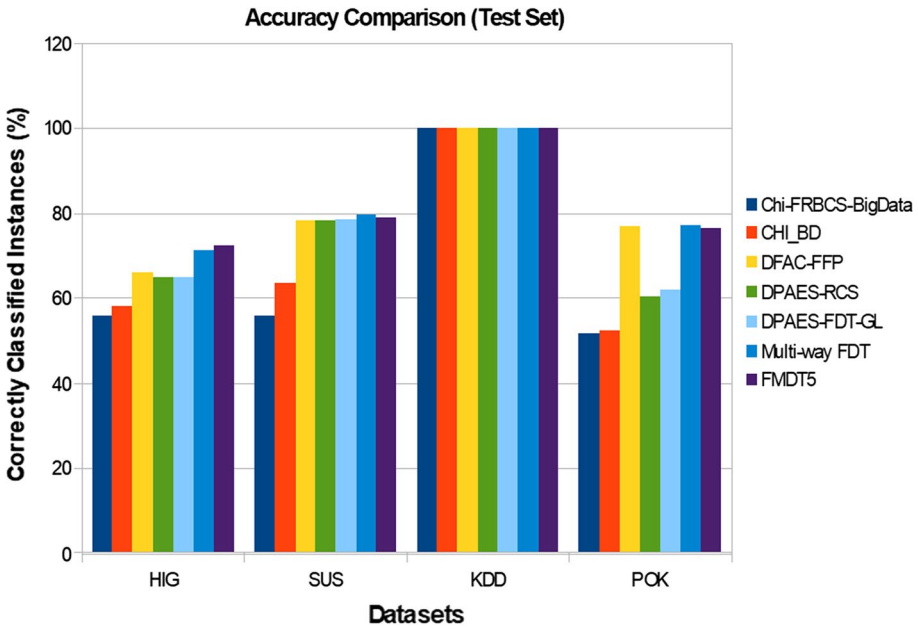

Fig. 11 Accuracy comparison among the different fuzzy classification models

generated by two distributed non fuzzy ACs, namely MRAC and MRAC+, introduced by Bechini et al. in [13]. On the other hand, the fuzzy AC models are more compact than MRAC and MRAC+. Segatori et al. in [25], compared the distributed FDTs with non-fuzzy Distributed Decision Trees (DDTs), available in the Spark Mlib. Results show that, in most datasets, the distributed FDTs achieve better accuracies than DDTs. As regards the complexities, the binary FDT are comparable with DDTs in terms of number of leaves and nodes, while multi-way FDTs are the most complex classification models. As regards DPAES-RCS, Ferranti et al., in [27], show that this algorithm achieves performances, in terms of accuracy, comparable with the ones achieved by DDTs. Obviously, the interpretabilty of the classification models generated by DPAES-RCS is much higher than the one of DDTs, both at complexity and semantic levels. Indeed, the rules that can be derived from DDTs are not linguistic rules, thus they are very hard to read and interpret.

Another important aspect to take into consideration when dealing with distributed algorithms is the scalability. To this aim, Chu et al. in [71] suggest to adopt the speedup σ as the main metrics for evaluating the scalability in parallel and distributed computing. According to the speedup definition, the efficiency of a program using multiple CUs is calculated comparing the execution time of the parallel implementation against the corresponding sequential version. For most of the distributed fuzzy classification models discussed in this work, authors carried out a scalability analysis. Specifically, for CHI_BD, DFAC-FFP, DPAES-RCS, and the Multi-way FDT, authors calculated different values of speedups, varying the number of CUs from 4/8 to 24/32. These algorithms have shown an almost linear behavior for the speedup of the classification model learning process. This means that, whenever needed, additional CUs can be used to effectively

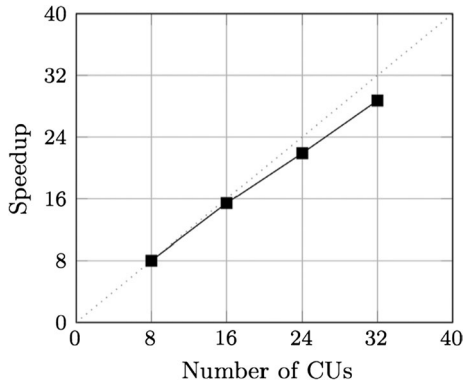

Fig. 12 The speedup trend of DPAES-RCS

reduce the runtimes. As an example, in Fig. 12, we show the speedup trend of the DPAES-RCS algorithm, extracted from the paper of Ferranti et al. [27].

Discussion and future directions

After the analysis that we have provided till now, we can state that a set of effective algorithms and tools are available for approaching the problem of generating fuzzy classification models from Big Data. In Table 5, for each discussed algorithm, we highlight its strengths and its weaknesses.

The analysis of the results has highlighted that FDTs are the most accurate classification models. However, these models are characterized by a high complexity level. On the other hand, the FRBCs generated by a distributed multi-objective learning scheme, based on the DPAES-RCS algorithm, are characterized by an optimal trade-off between their interpretability and their accuracy. As counterpart, these interpretable models are generated by means of EAs, which are, in general, characterized by a quite long execution time. Finally, the fuzzy classification models discussed in this work are not able to deal with streaming data. Indeed, once a specific fuzzy classification model has been generated, it cannot be adapted with new training data, which may reflect some changes of the domain context.

We envision that the future directions in the context of fuzzy classification models for Big Data will regard: (i) *enhancing the interpretability* of the rules and of the fuzzy partitions, both at semantic and complexity levels, (ii) *handling data streams* [72] moving towards a more general granular computing framework [73, 74]; and (iii) *reducing the computation efforts* for generating compact and accurate solutions. The three aforementioned challenges should be conducted in parallel as much as possible. Indeed, interpretable models, able to extract knowledge in almost real-time from huge amount of streaming and heterogeneous data, will be the actual added values for future research activities on classification tasks for Big Data.

Additional efforts can also be done with respect to the fields of application of fuzzy classification models. In fact, recent developments in several fields such as, cyber-physical systems [75, 76], cyber-security [77], and learning analytics [78], have increased the amount of collected data to an enormous scale. These data are inherently uncertain due to noise, incompleteness, and inconsistency, thus they require the adoption of

Table 5 Algorithms used in the experimental comparison, strengths and weaknesses

Algorithm	Strengths	Weaknesses
Chi-FRBCS-BigData	The first distributed algorithm proposed in the literature for learning a fuzzy model in big data classification	Employs a local search, thus the structure of the final model depends on how data chunks are generated
		Adopts a single reducer for fusing the rules generated by a distributed mapping stage
		Generates a large number of rules
		Generally achieves accuracies lower than the comparison algorithms
CHI_BD	Global search: unlike Chi-FRBCS-BigData, employs a global search, thus the structure of the final model does not depend on how data chunks are generated	Generates a large number of rules
		Generally achieves accuracies lower than the comparison algorithms
DFAC-FFP	Includes a fuzzy discretization algorithm	Generates a large number of rules
	The generated models are very accurate	The input variables may be partitioned with a large number of fuzzy sets, thus the interpretability of the fuzzy partitions may be low
DPAES-RCS	Optimizes concurrently the rule bases and the parameters of the fuzzy sets	Adopts a pre-fixed number of fuzzy set for each input variable
	Generates solutions characterized by good trade-off between accuracy and interpretability	Is very slow with respect to the other algorithms (it is based on evolutionary optimization)
	Even the most accurate solutions are characterized by a reduced number of rules	
DPAES-FDT-GL	Adds to the strengths of the PAES-RCS algorithm the capability of optimizing also the number of fuzzy sets for each attribute	Is very slow with respect to the other algorithms (it is based on evolutionary optimization)
Multi-way FDT	Includes a fuzzy discretization algorithm	Is characterised by a low interpretability of the final models because of the large number of rules generated
	Is very fast for generating the models	
	The fuzzy classification models are very accurate	
$FMDT_I$	Adds to the strengths of the Multi-way FDT algorithm the capability of reducing the model complexity	The final models are still characterised by a low interpretability because of the large number of rules

appropriate techniques to manage them. With the increase of the amount, variety, and speed of data, also the inherent uncertainty increases consequently. Moreover, the interpretability of fuzzy models may accomplish with one of the most recent and relevant requirements of Artificial Intelligence (AI)-based applications, namely the explainability. Indeed, eXplainable AI (XAI) [79, 80] refers to all methods and techniques in the application of AI that allow users to understand how, given specific inputs, AI systems produce the corresponding outputs. Several application domains consider model interpretability to be fundamental and require appropriate trade-offs between accuracy and interpretability.

Conclusions

In this work, we have briefly discussed the main design and implementation issues regarding the most recent fuzzy models for handling classification tasks on Big Data. Specifically, we have analyzed different distributed implementations of learning

algorithms for generating the model structure of FRBCs and FDTs. Most of the discussed learning algorithms, specifically the ones regarding FRBCs, are extensions to the parallel and distributed environment of well-known sequential approaches for generating the RB and the fuzzy set parameters from data. In particular, we have discussed the distributed versions of the classical Chi algorithm, of an FAC and of some EFCs. As regards FDT, we have briefly resumed the steps of a novel distributed learning process, which exploits an attribute selection and splitting algorithm based on fuzzy information gain. We have drawn a comparison among the discussed distributed fuzzy classification algorithms, by considering the results obtained on four popular classification datasets for Big Data, in terms of accuracy and scalability. Moreover, for each algorithm, we identified its benefits and limitations.

In conclusion, through this work we have provided a clear description of the current background in the field of fuzzy models for big data. Moreover, we have carried out an accurate analysis on research challenges and gaps. Finally, we have suggested areas for further investigation for supporting researchers in positioning their works.

Abbreviations

AC: Associative Classifier; CU: Computing Unit; DB: Data Base; DDT: Distributed Decision Tree; DFAC-FFP: Distributed Fuzzy Associative Classifier based on Fuzzy Frequent Pattern; DPAES-RCS: Distributed Pareto Archived Evolution Strategy with Rule and Condition Selection; EA: Evolutionary Algorithm; EFC: Evolutionary Fuzzy Classifier; EFS: Evolutionary Fuzzy System; FAC: Fuzzy Associative Classifier; FCAR: Fuzzy Classification Association Rule; FDT: Fuzzy Decision Tree; FRBC: Fuzzy Rule-Based Classifier; HDFS: Hadoop Distributed File System; MOEFS: Multi-Objective Evolutionary Fuzzy System; PAES-RCS: Pareto Archived Evolution Strategy with Rule and Condition Selection; RB: Rule Base; RDD: Resilient Distributed Dataset; YARN: Yet Another Resource Negotiator.

Acknowledgements

Not applicable.

Authors' contributions

PD, MF and FM contributed equally to the analysis and the selection of the materials for the survey and to writing of the manuscript. All authors read and approved the final manuscript.

Funding

This work was partially supported by Tuscany Region, in the context of the projects Talent and Sibilla in the framework of regional program "FESR 2014-2020", and by the Italian Ministry of Education and Research (MIUR), in the framework of the CrossLab project (Departments of Excellence).

Availability of data and materials

The datasets adopted in this study are available in the UCI repository, at https://archive.ics.uci.edu/ml/datasets.php.

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Dipartimento di Ingegneria dell'Informazione, Largo Lucio Lazzarino, 1, 56122 Pisa, Italy. [2] Istituto di Informatica e Telematica - Consiglio Nazionale delle Ricerche (IIT-CNR), Via Giuseppe Moruzzi, 1, 56124 Pisa, Italy.

Received: 17 September 2019 Accepted: 21 February 2020
Published online: 10 March 2020

References

1. John Walker S. Big data: a revolution that will transform how we live, work, and think. London: Taylor & Francis; 2014.
2. Anuradha J, et al. A brief introduction on big data 5vs characteristics and hadoop technology. Procedia Comput Sci. 2015;48:319–24.
3. Laney D. 3-d data management: controlling data volume, velocity, and variety. META Group Res Note. 2001;6:6.
4. Wan J, Tang S, Li D, Wang S, Liu C, Abbas H, Vasilakos AV. A manufacturing big data solution for active preventive maintenance. IEEE Trans Ind Inform. 2017;13(4):2039–47.
5. Ducange P, Pecori R, Mezzina P. A glimpse on big data analytics in the framework of marketing strategies. Soft Comput. 2018;22(1):325–42.
6. Al-Ali A, Zualkernan IA, Rashid M, Gupta R, Alikarar M. A smart home energy management system using iot and big data analytics approach. IEEE Trans Consum Electron. 2017;63(4):426–34.

7. Stergiou C, Psannis KE. Recent advances delivered by mobile cloud computing and internet of things for big data applications: a survey. Int J Netw Manage. 2017;27(3):1930.
8. Wang Y, Kung L, Wang WYC, Cegielski CG. An integrated big data analytics-enabled transformation model: application to health care. Inf Manage. 2018;55(1):64–79.
9. Han J, Kamber JPM. Data Mining. Concepts and techniques. In: Data management systems, 3rd edn. Burlington: Morgan Kaufmann; 2012.
10. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Comm ACM. 2008;51(1):107–13.
11. Ludwig SA. Mapreduce-based fuzzy c-means clustering algorithm: implementation and scalability. Int J Mach Learn Cybern. 2015;6(6):923–34.
12. Kim Y, Shim K, Kim M-S, Lee JS. DBCURE-MR: an efficient density-based clustering algorithm for large data using mapreduce. Inf Syst. 2014;42:15–35.
13. Bechini A, Marcelloni F, Segatori A. A MapReduce solution for associative classification of big data. Inf Sci. 2016;332:33–55.
14. Maillo J, Ramírez S, Triguero I, Herrera F. KNN-IS: an iterative spark-based design of the k-nearest neighbors classifier for big data. Knowl Based Syst. 2017;117:3–15.
15. Zhou L, Pan S, Wang J, Vasilakos AV. Machine learning on big data: opportunities and challenges. Neurocomputing. 2017;237:350–61.
16. Coulouris G, Jean Dollimore TK. Distributed systems: concepts and design. London: Pearson Education; 2009.
17. Apache Hadoop. https://hadoop.apache.org/. Accessed Jan 2016.
18. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX conference on hot topics in cloud computing, vol. 10. 2010. p. 10.
19. Fernández A, Carmona CJ, del Jesus MJ, Herrera F. A view on fuzzy systems for big data: progress and opportunities. Int J Comput Intell Syst. 2016;9(sup1):69–80.
20. Hariri RH, Fredericks EM, Bowers KM. Uncertainty in big data analytics: survey, opportunities, and challenges. J Big Data. 2019;6(1):44.
21. Lopez V, del Rio S, Benitez JM, Herrera F. On the use of mapreduce to build linguistic fuzzy rule based classification systems for big data. In: Fuzzy systems (FUZZ-IEEE), 2014 IEEE international conference on, IEEE. 2014. pp. 1905–12.
22. del Río S, López V, Benítez JM, Herrera F. A MapReduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. Int J Comput Intell Syst. 2015;8(3):422–37.
23. López V, del Río S, Benítez JM, Herrera F. Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. Fuzzy Sets Syst. 2015;258:5–38.
24. Elkano M, Galar M, Sanz J, Bustince H. CHI-BD: a fuzzy rule-based classification system for big data classification problems. Fuzzy Sets Syst. 2017;348:75–101.
25. Segatori A, Marcelloni F, Pedrycz W. On distributed fuzzy decision trees for big data. IEEE Trans Fuzzy Syst. 2018;26(1):174–92.
26. Segatori A, Bechini A, Ducange P, Marcelloni F. A distributed fuzzy associative classifier for big data. IEEE Trans Cybern. 2018;48(9):2656–69.
27. Ferranti A, Marcelloni F, Segatori A, Antonelli M, Ducange P. A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data. Inf Sci. 2017;415:319–40.
28. Fernandez A, Almansa E, Herrera F. CHI-SPARK-RS: an spark-built evolutionary fuzzy rule selection algorithm in imbalanced classification for big data problems. In: 2017 IEEE international conference on fuzzy systems (FUZZ-IEEE), IEEE. 2017. pp. 1–6.
29. Barsacchi M, Bechini A, Ducange P, Marcelloni F. Optimizing partition granularity, membership function parameters, and rule bases of fuzzy classifiers for big data by a multi-objective evolutionary approach. Cogn Comput. 2019;11:367–87.
30. Elkano M, Uriz M, Bustince H, Galar M. On the usage of the probability integral transform to reduce the complexity of multi-way fuzzy decision trees in big data classification problems. In: 2018 IEEE international congress on Big Data. 2018. pp. 25–32.
31. Márquez A, Márquez F, Peregrín A. A scalable evolutionary linguistic fuzzy system with adaptive defuzzification in big data. In: 2017 IEEE international conference on fuzzy systems (FUZZ-IEEE), IEEE. 2017. pp. 1–6.
32. López S, Márquez AA, Márquez FA, Peregrín A. Evolutionary design of linguistic fuzzy regression systems with adaptive defuzzification in big data environments. Cogn Comput. 2019;11:388–99.
33. Cózar J, Marcelloni F, Gámez JA, de la Ossa L. Building efficient fuzzy regression trees for large scale and high dimensional problems. J Big Data. 2018;5(1):49.
34. Bharill N, Tiwari A, Malviya A. Fuzzy based scalable clustering algorithms for handling big data using apache spark. IEEE Trans Big Data. 2016;2(4):339–52.
35. Wu J, Wu Z, Cao J, Liu H, Chen G, Zhang Y. Fuzzy consensus clustering with applications on big data. IEEE Trans Fuzzy Syst. 2017;25(6):1430–45.
36. Hidri MS, Zoghlami MA, Ayed RB. Speeding up the large-scale consensus fuzzy clustering for handling big data. Fuzzy Sets Syst. 2018;348:50–74.
37. Pulgar-Rubio F, Rivera-Rivas A, Pérez-Godoy MD, González P, Carmona CJ, del Jesus M. MEFASD-BD: multi-objective evolutionary fuzzy algorithm for subgroup discovery in big data environments-a mapreduce solution. Knowl Based Syst. 2017;117:70–8.
38. Fernandez-Bassso C, Ruiz MD, Martin-Bautista MJ. Fuzzy association rules mining using spark. In: International conference on information processing and management of uncertainty in knowledge-based systems. Springer. 2018. pp. 15–25.
39. Gacto MJ, Alcalá R, Herrera F. Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures. Inf Sci. 2011;181(20):4340–60.
40. Wang H, Xu Z, Pedrycz W. An overview on the roles of fuzzy set techniques in big data processing: trends, challenges and opportunities. Knowl Based Syst. 2017;118:15–30.

41. Dean J, Ghemawat S. Mapreduce: a flexible data processing tool. Commun ACM. 2010;53(1):72–7.
42. Lyubimov D, Palumbo A. Apache Mahout: Beyond MapReduce. 1st ed. South Carolina: CreateSpace Independent Publishing Platform; 2016.
43. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX conference on networked systems design and implementation. NSDI'12. Berkeley, CA, USA: USENIX Association; 2012. p. 15–28.
44. Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Freeman J, Tsai D, Amde M, Owen S, Xin D, Xin R, Franklin MJ, Zadeh R, Zaharia M, Talwalkar A. Mllib: machine learning in apache spark. J Mach Learn Res. 2016;17(1):1235–41.
45. Carbone P, Katsifodimos A, Ewen S, Markl V, Haridi S, Tzoumas K. Apache flink: stream and batch processing in a single engine. Bull IEEE Comput Soc Tech Comm Data Eng. 2015;36(4)28–38.
46. Guillaume S, Charnomordic B. Fuzzy inference systems: an integrated modeling environment for collaboration between expert knowledge and data using FisPro. Expert Syst Appl. 2012;39(10):8744–55.
47. Cordón O, del Jesus MJ, Herrera F. A proposal on reasoning methods in fuzzy rule-based classification systems. Int J Approx Reason. 1999;20(1):21–45.
48. Chi Z, Yan H, Pham T. Fuzzy algorithms: with applications to image processing and pattern recognition. In: Advances in fuzzy systems–applications and theory. vol. 10. World Scientific, Singapore. 1996.
49. Antonelli M, Ducange P, Marcelloni F. A fast and efficient multi-objective evolutionary learning scheme for fuzzy rule-based classifiers. Inf Sci. 2014;283:36–54.
50. Fernandez A, Lopez V, del Jesus MJ, Herrera F. Revisiting evolutionary fuzzy systems: taxonomy, applications, new trends and challenges. Knowl Based Syst. 2015;80:109–21.
51. Khan GM. Evolutionary computation. In: Evolution of artificial neural development. 2018. pp. 29–37.
52. Ducange P, Marcelloni F. Multi-objective evolutionary fuzzy systems. In: International workshop on fuzzy logic and applications. Springer. 2011. pp. 83–90.
53. Fazzolari M, Alcalá R, Nojima Y, Ishibuchi H, Herrera F. A review of the application of multi-objective evolutionary fuzzy systems: current status and further directions. IEEE Trans Fuzzy Syst. 2013;21(1):45–65.
54. Deb K. Multi-objective optimization. In: Burke EK, Kendall G, editors. Search methodologies. Berlin: Springer; 2014. p. 403–49.
55. Ishibuchi H, Yamamoto T. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. Fuzzy Sets Syst. 2004;141(1):59–88.
56. Cococcioni M, Ducange P, Lazzerini B, Marcelloni F. A pareto-based multi-objective evolutionary approach to the identification of mamdani fuzzy systems. Soft Comput. 2007;11(11):1013–31.
57. Botta A, Lazzerini B, Marcelloni F, Stefanescu DC. Context adaptation of fuzzy systems through a multi-objective evolutionary approach based on a novel interpretability index. Soft Comput. 2009;13(5):437–49.
58. Fazzolari M, Alcalá R, Herrera F. A multi-objective evolutionary method for learning granularities based on fuzzy discretization to improve the accuracy-complexity trade-off of fuzzy rule-based classification systems: D-MOFARC algorithm. Appl Soft Comput. 2014;24:470–81.
59. Antonelli M, Ducange P, Lazzerini B, Marcelloni F. Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity and partition integrity. Soft Comput. 2011;15(12):2335–54.
60. Baralis E, Garza P. I-prune: Item selection for associative classification. Int J Intell Syst. 2012;27(3):279–99.
61. Abdelhamid N, Ayesh A, Thabtah F, Ahmadi S, Hadi W. MAC: a multiclass associative classification algorithm. J Inf Knowl Manage. 2012;11(02):1250011.
62. Alcala-Fdez J, Alcala R, Herrera F. A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. IEEE Trans Fuzzy Syst. 2011;19(5):857–72.
63. Antonelli M, Ducange P, Marcelloni F, Segatori A. A novel associative classification model based on a fuzzy frequent pattern mining algorithm. Expert Syst Appl. 2015;42(4):2086–97.
64. Zhang C, Zhang S. Association rule mining: models and algorithms. Berlin: Springer; 2002.
65. Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: SIGMOD Rec. vol. 29. New york: ACM. 2000. pp. 1–12.
66. Quinlan JR. Induction of decision trees. Mach Learn. 1986;1(1):81–106.
67. Altay A, Cinar D. In: Kahraman C, Kabak Ö, editors. Fuzzy decision trees. Cham: Springer; 2016. pp. 221–61.
68. Witten IH, Frank E, Hall MA, Pal CJ. Data mining: practical machine learning tools and techniques. Burlington: Morgan Kaufmann; 2016.
69. Pecori R, Ducange P, Marcelloni F. Incremental learning of fuzzy decision trees for streaming data classification. In: 2019 conference of the international fuzzy systems association and the European society for fuzzy logic and technology (EUSFLAT 2019). Paris: Atlantis Press. 2019/08.
70. Fernández A, del Río S, Bawakid A, Herrera F. Fuzzy rule based classification systems for big data with MapReduce: granularity analysis. Adv Data Anal Classif. 2016;11:711–30.
71. Chu C-T, Kim SK, Lin Y-A, Yu Y, Bradski G, Olukotun K, Ng AY. Map-reduce for machine learning on multicore. In: Advances in neural information processing systems. 2007. pp. 281–8.
72. Pecori R, Ducange P, Marcelloni F. Incremental learning of fuzzy decision trees for streaming data classification. In: 2019 conference of the international fuzzy systems association and the European society for fuzzy logic and Technology (EUSFLAT 2019). Atlantis Press. 2019.
73. Pedrycz W. Granular computing: analysis and design of intelligent systems. Boca Raton: CRC Press; 2016.
74. Antonelli M, Ducange P, Lazzerini B, Marcelloni F. Multi-objective evolutionary design of granular rule-based classifiers. Granul Comput. 2016;1(1):37–58.
75. Xu LD, Duan L. Big data for cyber physical systems in industry 4.0: a survey. Enterp Inf Syst. 2019;13(2):148–69.
76. Mohammadi M, Al-Fuqaha A, Sorour S, Guizani M. Deep learning for iot big data and streaming analytics: a survey. IEEE Commun Surv Tutor. 2018;20(4):2923–60.

77. Kayes A, Rahayu W, Dillon T, Chang E, Han J. Context-aware access control with imprecise context characterization through a combined fuzzy logic and ontology-based approach. In: OTM confederated international conferences "On the move to meaningful internet systems". Springer. 2017; pp. 132–53.
78. Pecori R, Suraci V, Ducange P. Efficient computation of key performance indicators in a distance learning university. Inf Discov Deliv. 2019;47:96–105.
79. Adadi A, Berrada M. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE Access. 2018;6:52138–60.
80. Fernandez A, Herrera F, Cordon O, del Jesus MJ, Marcelloni F. Evolutionary fuzzy systems for explainable artificial intelligence: why, when, what for, and where to? IEEE Comput Intell Mag. 2019;14(1):69–81.

Publisher's Note