Knowledge-Based Systems 117 (2017) 70–78

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

MEFASD-BD: Multi-objective evolutionary fuzzy algorithm for subgroup discovery in big data environments - A MapReduce solution

F. Pulgar-Rubio[a], A.J. Rivera-Rivas[a], M.D. Pérez-Godoy[a], P. González[a], C.J. Carmona[b,*], M.J. del Jesus[a]

[a] Department of Computer Science, University of Jaen, 23071, Jaen, Spain
[b] Department of Civil Engineering, Languages and Systems Area, University of Burgos, 09001, Burgos, Spain

ARTICLE INFO

Article history:
Received 30 March 2016
Revised 17 August 2016
Accepted 24 August 2016
Available online 24 August 2016

Keywords:
Subgroup discovery
Big data
Multi-objective evolutionary fuzzy systems
Apache Spark
MapReduce

ABSTRACT

Nowadays, there is an incredible increase of data volumes around the world, with the Internet as one of the main actors in this scenario and a growth rate above 30GB/s. The treatment of this huge amount of information cannot be carried out through traditional data mining algorithms in an efficient way and it is necessary to adapt and design new algorithms towards distributed paradigms such as MapReduce. This situation is a challenge for the community, investigated under the widely known term of big data.

This paper presents a new algorithm for the subgroup discovery task called MEFASD-BD. The algorithm is developed in Apache Spark based on the MapReduce paradigm, and it is able to tackle high dimensional datasets in an efficient way. In fact, this algorithm is the first approximation to big data within evolutionary fuzzy systems for subgroup discovery. MEFASD-BD implements novel MapReduce functions which are able to analyse the quality of the subgroups obtained for each map with respect to the original dataset in order to improve the quality of these subgroups. In addition, the final reduce function of the algorithm employs the token competition operator in order to select the best rules extracted in the different maps. An experimental study with high dimensional datasets is performed in order to show the advantages of this algorithm in this type of problems. Specifically, the results of the study show an important reduction of the runtime while keeping the values in the standard quality measures for subgroup discovery.

1. Introduction

The rapid progress in the development of information technologies is causing an exponential growth of the data volumes generated around the world. At present, the Internet generates more than 30GB/s: over 7100 tweets are sent, over 480 Instagram photos are uploaded, over 53,000 Google searches are done, over 120,000 YouTube videos are viewed, etc. [32]. The digital universe is being increased by a factor of 10 and predictions show that by 2020, 40 zettabytes of data will be stored [25]. These data are known as big data [43]. Big data is characterized by its well-known Vs: great Volumes of data, great Variety of formats, high Velocity in the generation of the data, mechanisms to ensure the Veracity of data and so on. These adjectives describe the difficulties of working with big data, and one of the main issues is that this huge amount of data can not be analysed using traditional techniques.

Data science appears with the objective of extracting knowledge from large volumes of data, but new technologies, instruments, etc., are needed to address this objective. The most popular example of these new paradigms is MapReduce [15,16], consisting of a distributed processing system using a divide and conquer approach. In recent years, there is a growing interest in the development of new algorithms using this type of approach for big data problems.

One of the promising areas where to develop new algorithms for the big data environment within data science and data mining is subgroup discovery (SD) [34,60]. SD is a data mining task in which, given a property of a data set (the variable of interest, or target variable) the objective is to discover subgroups whose behaviour is different from that of the complete set (that is, with a different statistical distribution). The knowledge extracted in an SD task is normally represented in the form of rules. In this sense, evolutionary algorithms (EAs) [26,31], which imitate the principles of natural evolution to address optimisation and learning problems, are well suited to perform the SD task due to their ability to reflect

* Corresponding author at: EPS Vena (A1-4127), University of Burgos, 09006, Burgos, Spain
E-mail addresses: fpulgar@ujaen.es (F. Pulgar-Rubio), arivera@ujaen.es (A.J. Rivera-Rivas), lperez@ujaen.es (M.D. Pérez-Godoy), pglez@ujaen.es (P. González), cjcarmona@ubu.es, ccarmona@ujaen.es (C.J. Carmona), mjjesus@ujaen.es (M.J. del Jesus).

http://dx.doi.org/10.1016/j.knosys.2016.08.021
0950-7051/

the interaction of variables in a rule-learning process also providing great flexibility in the representation [21]. In addition, multi-objective EAs are appropriate in this type of problems where different quality measures have to be optimized at a time. On the other hand, the use of fuzzy rules, based on fuzzy logic [64] allows to consider uncertainty and to represent continuous variables more closely to human reasoning.

However, existing approaches of evolutionary algorithms for SD are not able to work efficiently with the huge amount of data in a big data environment. To address this problem, a new proposal of SD algorithm for big data using the MapReduce approach is introduced in this paper based on the well-known NMEEF-SD algorithm [8]. The paper is organized as follows: Section 2 describes the concepts used along this paper, SD, evolutionary fuzzy systems (EFSs) and big data. The new algorithm for SD is described in Section 3, including a motivation for the introduction of this new algorithm and its main properties. In Section 4 the experimental study can be observed. Finally, concluding remarks and future challenges are outlined in Section 5.

2. Background

This section provides background information about the main concepts used in this paper. Section 2.1 presents the SD task and its main properties. Next, Section 2.2 introduces the concept of EFS and its use in SD. Finally, in Section 2.3 the MapReduce paradigm is presented.

2.1. Subgroup discovery

SD has been grouped within Supervised Descriptive Rule Induction [36] with other techniques such as Contrast Set Mining [4] or Emerging Pattern Mining [19]. The common objective of these techniques is to understand the underlying phenomena with respect to an objective class.

SD was initially introduced by Kloesgen [34] and Wrobel [60], and can be defined in the following way [61]:

> *"In subgroup discovery, we assume we are given a so-called population of individuals (objects, customers, ...) and a property of those individuals we are interested in. The task of subgroup discovery is then to discover the subgroups of the population that are statistically 'most interesting', i.e., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest."*

The representation of the knowledge is performed through rules (R) consisting of induced subgroup descriptions [24,37]. Each rule can be formally defined as:

$$R : Cond \rightarrow Target_{value}$$

where $Target_{value}$ is a value of the variable of interest for SD (it also appears as the target variable or *Class* in the literature), and *Cond* is commonly a conjunction of features (attribute-value pairs) which is able to describe an unusual statistical distribution with respect to the $Target_{value}$.

SD uses descriptive induction through supervised learning. Although supervised learning is widely used in classification, SD is a different task: SD attempts to describe knowledge by data in an interpretable way, while a classifier attempts to predict the target value of new data. This difference is illustrated in Fig. 1, where the model on the left represents a subgroup in a dataset with an independent, single and simple form; on the contrary the model on the right represents a classifier with a model composed of a set of dependant rules which are analysed overall with respect to accuracy.

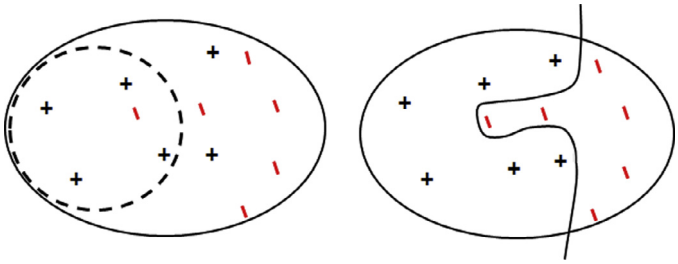

Fig. 1. Subgroup discovery model (left) Vs. classification model (right).

SD algorithms are based on four main elements [3]: the type of the target variable, such as binary, nominal or numeric; the search strategy, such as beam search (CN2-SD [39] or Apriori-SD [33]), exhaustive search (SD-Map [2] or Merge-SD [27]), amongst others; the description language used, focused on the interpretability of the knowledge extracted (considering that an SD model is interpretable if it is composed of a small set of rules with a low number of variables); and finally, the quality measures.

The latter is the most relevant due to the difficulty of analysis. In [7], three guidelines were presented in order to measure the quality of subgroups obtained by a SD model: interpretability, interest and trade-off between sensitivity and confidence. A complete analysis of the quality measures used throughout the SD literature can be found in [30], and the quality measures employed in this paper are summarised below:

- *Unusualness* is the weighted relative accuracy of a rule [38] and it measures interest and a trade-off between generality and precision. It can be computed as:

$$Unus(R_{i,j}) = \frac{n(Cond)}{n_s} \cdot \left(\frac{n(TargetValue_j \cdot Cond)}{n(Cond)} - \frac{n(TargetValue_j)}{n_s} \right) \quad (1)$$

  It can be described as the balance between the coverage of the rule $p(Cond_i)$ and its accuracy gain $p(TargetValue_j \cdot Cond) - p(TargetValue_j)$, where $n(Cond)$ is the number of examples which satisfy the conditions determined by the antecedent part of the rule, n_s is the number of total examples, $n(TargetValue_j \cdot Cond)$ is the number of examples which satisfy the conditions and also belong to the value for the target variable within the rule, and $n(TargetValue_j)$ are all the examples of the target variable. The domain of this quality measure is specified for each problem because there is a direct dependence with respect to the target variable.
- *Sensitivity* is the proportion of actual matches that have been classified correctly [34] and it has a component based on generality. It is computed as:

$$Sens(R_{i,j}) = \frac{n(TargetValue_j \cdot Cond)}{n(TargetValue_j)} \quad (2)$$

  This quality measures can be found in the literature as the Support based on the examples of the class, Recall or *TPrate*.
- *Fuzzy confidence* is an adaptation of the standar confidence measure for fuzzy rules [18]. This quality measures obtains the precision of one subgroup and it is defined as:

$$Cnf(R_{i,j}) = \frac{n(TargetValue_j \cdot Cond)}{n(Cond)} \quad (3)$$

The sensitivity and fuzzy confidence quality measures have a domain between 0 and 1, and the unusualness depends on the percentage of the majority class for the target variable (MajTV). In this case, an analysis of the domain for the unusualness was presented in [14]:

- Lower Bound Unusualness (LBU): $(1 - \%MajTV) * (0 - \%MajTV)$
- Upper Bound Unusualness (UBU): $\%MajTV * (1 - \%MajTV)$

A normalisation of the unusualness to the domain [0, 1] can be considered using the following Eq:

$$NormUnus(R_{i,j}) = \frac{Unus(R_{i,j}) - LBU}{UBU - LBU} \quad (4)$$

2.2. Evolutionary fuzzy systems in subgroup discovery

An EFS [29] is a hybridisation of fuzzy systems [64] augmented with a learning process based on evolutionary computation [20]. Fuzzy systems are one of the most important areas for the use of fuzzy set theory [65] and are characterised by their excellent ability to handle imprecision and uncertainty, and to describe the behaviour of complex systems without requiring a precise mathematical model. The most common fuzzy models consist of a collection of logical fuzzy rules and are known as fuzzy rule-based systems (FRBSs). The automatic design of FRBSs can be considered an optimization task or a search problem. In fact, the predominant type of EFSs is that focused on FRBSs which constitute an extension to classical rule-based systems, because they deal with "IF-THEN" rules, whose antecedent and consequent are composed of fuzzy logic statements, instead of classical ones.

On other hand, evolutionary computation, including EAs [26,31], genetic programming [35], and evolutionary programming [22], amongst others, have been employed thanks to their ability to deal with large search spaces and to find near-optimal solutions without a precise description of the problem. In addition, these types of algorithms are able to incorporate knowledge into the model. In fact, the incorporation of knowledge in FRBSs can be performed for example through the parameters of the fuzzy membership functions, or in the form of linguistic variables.

EFSs have been widely employed in SD throughout the literature because the SD task is a rule learning process that can be seen as an approximation problem in which the objective is the learning of the parameters of the model. In this task, the search space can be very complex and the search strategy used becomes a key factor. Therefore, the hybridisation between fuzzy logic and EAs is very well suited for this task. Among the SD algorithms based on EFSs in the literature, the main algorithms to be highlighted are SDIGA [18], FuGePSD [14] and NMEEF-SD [8]. These algorithms have effectively solved a wide number of real-world problems in different fields. In fact, interesting and unusual knowledge have been recently obtained in topics such as:

- Bioinformatics, with problems related to the acute sore throat [14] and the influenza A virus [6].
- Medicine, with the optimisation of the Emergency service in a concrete hospital from Madrid (Spain) [9].
- E-commerce, with the description of the clients related to a website about extra virgin olive oil [13].
- Industry, with the analysis of a concentrating photovoltaic problem [12].
- E-learning, with the description of relationships between the use of an e-learning platform and the marks obtained by the students [10,11,49].

2.3. MapReduce framework through Apache Spark in big data environments

MapReduce [15,16] is the most popular paradigm developed to deal with big data problems. It is developed by Google and allows

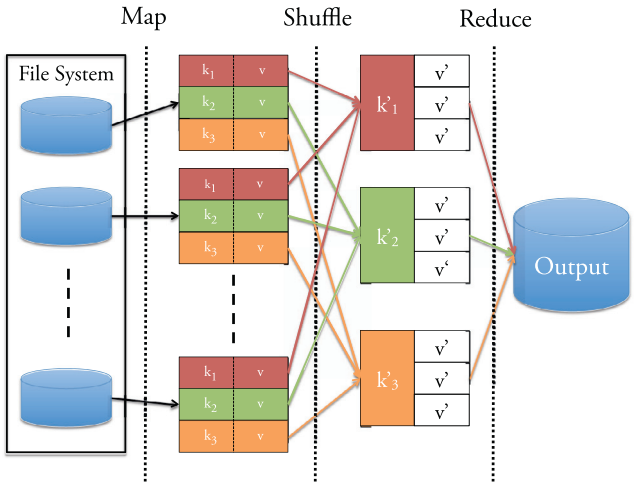

Fig. 2. Scheme operation for the *MapReduce* paradigm.

the processing of large datasets through automatic parallelization of the computation over a cluster of machines. In this way, the details of parallel computation are hidden for the programmers. Fig. 2 shows the scheme operation of the MapReduce framework. MapReduce is based on functional programming and works in two different steps:

- Map-function: The master node of the cluster segments the dataset in i independent subdatasets and distributes them to the worker nodes. Each one has a key-value pair (k_i, v) as input and output. Then, worker nodes process the subdatasets and generate a set of intermediate (k'_i, v') pairs. Finally, all values with the same key in the intermediate pairs are merged (named the shuffle phase) in order to speed up the computation in the following step.
- Reduce-function: The master node collects all the results of the subdatasets and combines them to obtain the final results. The reduce-function receives the intermediate key-value pairs and generates as final results the corresponding pair of key and value.

It is important to remark that all map and reduce operations run in parallel, i.e. all the maps are run in an independent way and the reductions cannot be executed until their respective maps are finished.

Apache Hadoop [45,56] was one of the first open source implementations of the MapReduce paradigm, providing fast data transfer rates among machines by means of the Hadoop Distributed File System (HDFS). Hadoop and other big data paradigms promote data-parallel computations on clusters and provide fault tolerance and load balancing among other advantages. However, they lack of abstractions for the exploitation of distributed memory, a very important feature to improve scalability when big data problems are tackled.

A new MapReduce approach developed to solve these drawbacks is Spark [67], started in 2009 as an open source project in the UC Berkeley RAD Lab, later the AMPLab. Spark, through its main abstraction Resilient Distributed Datasets (RDDs), promotes fault-tolerant characteristics, enables the persistence of intermediate results in memory, and provides control of data partitioning and their manipulation using a rich set of operators. An RDD allows two types of operations such as transformations and actions: a transformation is a "lazy" operation that return a new RDD where all the calculations are scheduled, and action is an operation where all the processing queries are immediately computed and a result value is returned. Map operation is an example of

transformation and reduce operation is an example of action. From the point of view of efficiency, Spark applications run up to 100 times faster in memory, 10 times faster when running on disk, than Hadoop applications [66].

Apart from RDDs, Spark presents some remarkable characteristics [28] such as:

- Allows to develop multi-step data pipelines using directed acyclic graph (Spark Graph X) pattern.
- Runs on top of distributed file systems technologies such as Hadoop Distributed File System (HDFS).
- Provides 80 high-level operators or an interactively query data shell.
- Supports SQL queries (Spark SQL).
- Supports streaming data processing (Spark Streaming).
- Different machine learning libraries are available, such as MLlib or ml.
- Supports different programming languages such as Java, Scala, Python or R.

3. MEFASD-BD: Multi-objective evolutionary fuzzy algorithm for subgroup discovery in big data environments

This section presents a new SD proposal based on the MapReduce approach. First, Section 3.1 argues the motivations for the creation of the proposal. Section 3.2 presents the complete description of the proposed algorithm, MEFASD-BD, along with a flowchart.

3.1. Motivation

The explosion in the generation and collection of large datasets has further encouraged the knowledge extraction process and its analysis, in the believe that the exponential increase in the dimensionality of data would result in an increase of the precision of the models obtained. In fact, as previously mentioned, there is an emerging interest in big data, referring to massive amounts of data that are difficult to handle and analyse through traditional tools. However, standard algorithms are not usually able to deal with these huge datasets and it is necessary to adapt or redesign current learning algorithms in order to solve high dimensional problems because the abilities and capacities of data mining algorithms decrease when the size of the dataset increases. In fact, memory consumption and runtime increase significantly so that extremely powerful machines are needed to address the problem of knowledge extraction. A common solution to deal with the problem of dimensionality is the development of algorithms for distributed systems. In this way, the distribution and parallelization of the problem in different sub-problems could considerably reduce the use of memory and the runtime without the necessity of using very expensive machines. In fact, there are a number of algorithms implemented for distributed systems within data mining, such as Mahout for Hadoop or MLLib for Spark.

Throughout the literature the number of papers related to big data is growing in the last two years and specifically in data mining can be found papers about the use in imbalanced problems [23,42,47,54], feature selection [5,44,46], fusion of linguistic rules [48,50], prototype reduction [53], decision rules [59], fuzzy clustering [51], traffic flow forecasting [57,63], remote sensing [55], amongst others.

As with data mining algorithms in general, the interest in using SD algorithms to face big data problems is also growing. There are few works in the literature that examine the parallelization of the SD task. A first approach for SD on distributed data is introduced in [62]. It is important to note that in this work an important problem when trying to parallelize the SD task is described: when we divide the dataset into different subsets, the best local subgroups obtained in each subset does not necessarily yield a set containing the best global subgroups. So it is important to be sure of matching the distributions underlying the subsets with that of the complete dataset. Another contribution can be found in [40], where an algorithm is proposed that filters irrelevant subgroups due to their overlap, also proposing a procedure to ease the parallelization of the process by balancing the discovery effort among the different processes. In [52] is presented a parallel computing approach for SD where data to be analysed is not distributed, but instead portions of the overall search space is distributed to be considered on different computing nodes.

Although EAs have proven to be a very good approach to address SD problems [7] no proposals of distributed evolutionary SD algorithms have been developed so far to address the SD task in big data environments. In this paper, we introduce a new proposal of evolutionary SD algorithm adapted for the big data environment using the MapReduce paradigm. This will allow to execute SD algorithms in high dimensional datasets (above the one million of instances, for example) in limited space of time and with standard hardware. This algorithm is based in NMEEF-SD [8], an evolutionary SD algorithm that is well-known throughout the literature. NMEEF-SD obtains a set of general and accurate fuzzy rules thanks to the use of diverse genetic operators and a screening function based on the confidence. In addition, this algorithm employs two objectives for optimising problems because it is important the obtaining of good results in several quality measures for the SD task.

The algorithm has been developed using the Spark framework (rather than other frameworks such as Hadoop) due to some of its previously mentioned features. First, Spark includes abstractions for the exploitation of distributed memory, allowing to improve the scalability on big data problems. In addition, Spark applications run faster than Hadoop applications according to [66].

3.2. The core of the algorithm

The algorithm proposed to face SD problems in a big data environment, MEFASD-BD, is the first evolutionary SD algorithm for big data. MEFASD-BD follows the MapReduce approach and it is implemented in the Scala language trying to maximise the advantages of Apache Spark.

The base of MEFASD-BD is a multiobjective evolutionary algorithm following the well-known NSGA-II approach [17] and the basic ideas of the algorithm NMEEF-SD [8], using two quality measures as objectives, sensitivity (Eq. (2)) and unusualness (Eq. (1)) in order to obtain interesting fuzzy subgroups. Each candidate solution is codified with the "*Chromosome* = *Rule*" approach, where only the antecedent is represented. So, an execution for each value of the target variable is performed. At the end of the evolutionary process, a screening function based on confidence (Eq. (3)) is applied.

In summary, the algorithm works in the following way: the dataset is divided into partitions and each partition is loaded in a map. Then, the evolutionary process is carried out on each map as many times as values of the target variable, obtaining rules describing subgroups for each partition. However, as the subgroups have been obtained considering only partitions of the original dataset, now the algorithm computes global quality measures for these rules on the complete dataset. This is done because it is important to prevent the obtaining of partial subgroups that work well only for subsets of data. Finally, the best rules according to the global quality measures are selected using a token competition operator based on unusualness (Eq. (1)) obtaining a final reduced set of rules corresponding to the most relevant subgroups for the SD task.

Next, a complete description of this algorithm for each stage together a flowchart with its operation are presented.

3.2.1. Map stage in MEFASD-BD

The first stage of the algorithm is determined by the number of nodes that the programmer decides to use. In this way, the training dataset is divided into the given number of disjoint subsets, i.e. number of *Maps*. Spark divides automatically the original training file in subsets with a proportional number of instances in each partition. Next, the evolutionary process is executed in each map once for each value of the target variable. In this way, an initial rule set (*RS*) for each partition is obtained where all subgroups obtained are non-dominated rules with respect to sensitivity and unusualness.

Algorithm 1 Map function in MEFASD-BD.

```
1: function MAP(Maps)
2:     repeat
3:         t ← 0
4:         Generate P(t)
5:         repeat
6:             Generate offspring Q(t)
7:             R(t) ← P(t) ⋃ Q(t)
8:             Generate all non-dominated fronts from R(t)
9:             if the Pareto front evolves then
10:                Complete P(t+1) with fronts in order
11:            else
12:                Re-initialisation on coverage in P(t+1)
13:            end if
14:            t++
15:        until Number of evaluations is reached
16:        RS ⋃ Pareto front with high confidence
17:        Following Target_value
18:    until Not Target_value
19:    return RS
20: end function
```

Pseudo-code of this function can be analysed in Algorithm. 1 where an independent *RS* is obtained for each partition. The algorithm is executed for each value of the target variable ($Target_{value}$), where first a main population (P_t) is generated, and then an evolutionary process controlled by a number of evaluations is executed. Within this process, an offspring population (Q_t) is created based on P_t through genetic operators, and both populations are joined in R_t. This new population is divided in non-dominance fronts and depending on the evolution of the Pareto front an operator of re-initialisation is applied. This operator can be analysed in [8]. The final *RS* is composed of all the non-dominated rules obtained in the final Pareto front for each $Target_{value}$ as long as they reach a good confidence value.

3.2.2. Compute global measures in MEFASD-BD

At this moment, the algorithm has obtained a set of rules for each map considering only its own subset of data, i.e. rules obtained are computed with respect to a chuck of data instead of the whole dataset. In this way, it would be necessary to analyse these rules with respect to the complete training dataset, so it is necessary to compute the quality of each rule with respect to the remaining data spread in the remaining partitions. Although this task could be solved through a shuffle function, this re-distribution would involve copying data across machines provoking complex and costly operations. In Spark, this operation can be solved through the use of broadcast variables which are shared in all the nodes of the cluster. In fact, a broadcast variable with the original dataset allows to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. In this way, Spark distributes this type of variables using efficient broadcast algorithms to reduce communication cost.

The objective of this stage is therefore the obtaining of rules that represent the contained information in the dataset analysed with the major possible quality. In fact, this function is key in order to achieve interesting subgroups in big data environment. The calculation of global measure for each rule is obtained through the analysis of the rules with respect to the whole dataset. Pseudo-code of this stage of the algorithm is presented in Algorithm 2. In

Algorithm 2 Global measures procedure in MEFASD-BD.

```
1: procedure GLOBAL MEASURES(Maps, Rules, Dataset)
2:     i ← 0
3:     repeat
4:         Map ← Analysing map(Maps,i)
5:         j ← 0
6:         repeat
7:             R ← Extract rule(Map,j)
8:             Compute global quality measures (R, Dataset)
9:             if GlobalConfidence(R) is lower than a threshold then
10:                Delete (R, Map)
11:            end if
12:            j++
13:        until Not Rules
14:        i++
15:    until Not Maps
16: end procedure
```

this way, the algorithm takes one rule (*R*) for one map (*Map*) and it computes the global quality measures with respect to the complete dataset (*Compute global quality measures (R, Dataset)*). This process is repeated for all rules (*Rules*) obtained for each map (*Maps*) in the previous stage. In addition, it is very important to remark that the deletion of rules with values of global confidence under a threshold defined by the experts in a parameter is also performed. This filter improves the final set of rules obtained because the rules with low values of precision with respect to the whole dataset are discarded.

3.2.3. Reduce stage in MEFASD-BD

This is the final stage of the MEFASD-BD algorithm where a reduction of the rules obtained in each partition is carried out. The main objective is the combination of all the sets of rules with suitable levels of quality, computed with local information, in order to reduce the final set of rules to only the most significant ones. As can be observed in Algorithm 3 all the sets of rules (obtained by

Algorithm 3 Reduce function in MEFASD-BD.

```
1: function REDUCE(Maps)
2:     Complete RS ← RS_1 ⋃ RS_2 ⋃ ... ⋃ RS_n
3:     Order rules by global Unusualness
4:     Apply Token Competition operator
5:     Delete rules without tokens
6:     return Complete RS
7: end function
```

the map procedure) are combined in a big set of rules.

Next, a sorting based on the global unusualness computed in the previous stage is performed by applying the token competition operator [41,58] at the end of the process. This operator has been used by FuGePSD in [14] with excellent results and the main objective is to improve the diversity among the individuals at phenotipic level emulating the behaviour in a natural environment, also reducing the number of rules whose individuals have zero tokens.

MEFASD-BD considers an instance as a token and all the individuals must compete for this instance. When an individual matches an example provided, a flag will indicate that the example is seized and hence other individuals cannot capture it. In this way, the diversity is improved with respect to the phenotype, i.e. individuals that overcome the token competition describe information about examples of the dataset which are not covered by others rules. This process is applied for each individual in such a way that individual with the highest global unusualness exploits its niches

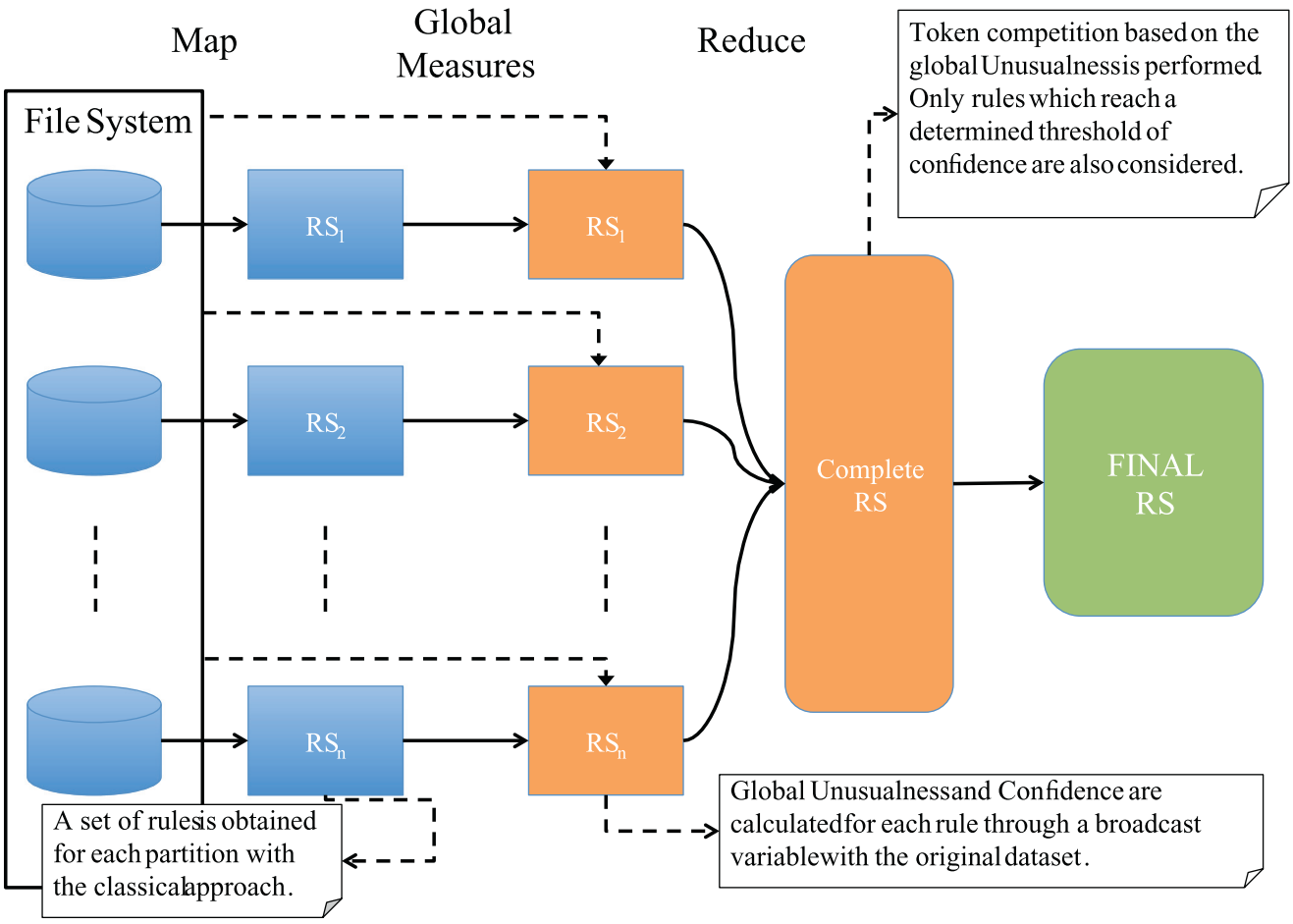

Fig. 3. Flowchart of the MEFASD-BD algorithm.

by seizing as many tokens as it can. The other individuals entering the same niches will have their strength decreased, since they cannot compete with the stronger ones.

Fig. 3 reflects the complete operating scheme of the new MEFASD-BD algorithm, following the MapReduce paradigm and using different quality measure during the process: local unusualness and sensitivity to obtain rules in the Map function, global fuzzy confidence for screening rules with a low value in confidence in the global measures function, and finally global unusualness in the token competition operator in the Reduce function.

4. Experimental framework and analysis

This section describes the main details of the experimental study performed. Section 4.1 summarises the datasets used in the study and the parameters of the algorithm. Section 4.2 presents the hardware and software features of the equipment used in the experiments. Finally, in Section 4.3 results and analysis for the complete experimental study are shown.

4.1. Datasets and parameters

The experimental study has been performed with high dimensional datasets with a high number of instances. Specifically, we have used datasets from the KEEL Repository[1] [1]. The main characteristics of these datasets are summarised in Table 1, including the number and type of the attributes (real, integer or nominal), the number of instances, the number of classes, and the size of each dataset.

Table 1
Datasets used in the experimental study.

Name	#Attributes (R/I/N)	#Examples	#Classes	Size (MB)
RLCP	10 (11/0/0)	5,749,132	2	319.56
KDD	41 (26/0/15)	494,020	23	99.26
Census	41 (1/12/28)	142,521	3	74.96
Fars	29 (5/0/24)	100,968	8	62.44
Connect	42 (0/0/42)	67,557	3	8.80
Shuttle	9 (0/9/0)	58,000	7	1.55

For the standard algorithms, the parameters considered by the authors in the original papers have been employed, i.e. NMEEF-SD [8], SDIGA [18] and FuGePSD [14]. With respect to the new evolutionary algorithm for big data environments, the following parameters have been used for each map: sensitivity and unusualness as objectives, three linguistic labels and a minimum confidence of 0.6; and for the evolutionary process, 51 individuals for the population, a maximum number of evaluations of 10,000, and finally, 0.60 and 0.10 for crossover and mutation probability, respectively.

4.2. Run environment

The experimental study has been carried out in the computation cluster of the Advanced Studies Center in Information and

[1] http://www.keel.es/datasets.php.

Table 2
Parameters used in Spark.

Parameter	Value
Number of nodes that compose the server	14
Maximum number of cores for each node	18
Maximum RAM memory for each node	64 GB
Maximum size for the serialize tasks	100 MB
CPU frequency of each node	2.5 Ghz

Table 3
Average results for the EFSs in subgroup discovery.

Ds	Algorithm	Rules	Vars	UNUS	SENS	FCNF	T(sec)
Shuttle	NMEEF-SD	17.20	3.44	0.517	1.000	0.815	706
	SDIGA	8.00	4.60	0.561	0.992	0.393	893
	FuGePSD	5.40	3.18	0.678	0.936	0.894	1980
Connect	NMEEF-SD	42.00	2.67	0.518	0.819	0.671	704
	SDIGA	3.20	2.00	0.500	0.951	0.347	1322
	FuGePSD	1.20	3.10	0.518	0.418	0.607	11032
Fars	NMEEF-SD	3.80	1.49	0.776	0.759	0.863	929
	SDIGA	9.00	15.76	0.512	0.240	0.195	2756
	FuGePSD	2.80	2.90	0.756	0.822	0.843	54321
Census	NMEEF-SD	44.80	3.62	0.598	0.841	0.964	531
	SDIGA	2.60	2.00	0.512	0.896	0.574	2543
	FuGePSD	2.20	1.87	0.518	0.649	0.964	55321
KDD	NMEEF-SD	121.80	11.02	0.848	0.964	0.923	29875
	SDIGA	20.60	8.16	0.501	0.280	0.389	69222
	FuGePSD	–	–	–	–	–	–
RLCP	NMEEF-SD	5.20	1.65	0.801	0.997	0.956	14723
	SDIGA	–	–	–	–	–	–
	FuGePSD	–	–	–	–	–	–

Communication Technologies of the University of Jaen[2]. The cluster is composed of several nodes, including a master node and the remaining as computation nodes. The main features and parameters of the cluster are summarised in Table 2.

With respect to the software, this cluster is based on the Bullx Linux Server, which is specifically designed for high-performance computing and based on RedHat Enterprise Linux (release 6.3). The required package for the execution of Spark has been installed in the cluster.

Finally, it must be noted that the values of the quality measures presented in the experimental study are the averages of the different executions. Specifically, executions are determined by a 5-fold cross-validation and 3 executions for each dataset, i.e. values for normalised unusualness (*UNUS*), sensitivity (*SENS*) and fuzzy confidence (*FCNF*) are the average results of 15 executions. In the experimental study are also presented the number of rules (*Rules*), the number of variables (*Vars*) in the antecedent part of the subgroups, and the runtime measured in seconds (*T*(*sec*)).

4.3. Analysis of the results

The main objective of this experimental study is to show the benefits of the algorithm MEFASD-BD in environments with high dimensionality. However, an analysis of the most important EFSs for SD is presented first in Table 3. As can be observed, the runtime increases when also increases the complexity of the dataset. In fact, algorithms SDIGA and FuGePSD have problems getting results with datasets such as KDD or RLCP. On the other hand, algorithm NMEEF-SD has a good behaviour with this type of datasets but with quite high runtime periods.

Next, once the difficulties of the EFSs for SD in high dimensional datasets is shown, a study of the new proposal is performed using different number of maps. The main objective is to show the advantages of big data algorithms within SD task, where an important reduction of the runtime while maintaining the quality of the results is presented. To analyse the results of an SD algorithm, it is usual to take into account the interpretability, the trade-off between generality and precision, and the interest. In addition, this experimental study considers the runtime of the algorithm using different number of maps. Table 4 shows the results of this experimental study with different datasets (*Ds*), including the number of Maps used (*Maps*), the runtime in seconds, and the average results for each quality measure.

Table 4
Average results of MEFASD-BD for each dataset.

Ds	Maps	Rules	Vars	UNUS	SENS	FCNF	T(sec)
Shuttle	2	1.00	3.00	0.919	0.849	0.885	821
	4	1.00	3.00	0.919	0.849	0.885	429
	8	1.00	3.00	0.919	0.849	0.885	237
	16	1.00	3.00	0.919	0.849	0.885	138
	32	1.00	3.00	0.919	0.849	0.885	96
	64	1.80	3.40	0.767	0.611	0.927	81
	128	1.80	3.20	0.767	0.611	0.927	68
	192	1.60	3.10	0.805	0.671	0.917	66
Connect	2	11.40	3.11	0.530	0.644	0.701	448
	4	16.20	3.21	0.531	0.636	0.702	235
	8	21.40	3.45	0.540	0.393	0.743	127
	16	21.00	3.74	0.538	0.359	0.754	74
	32	25.80	3.60	0.541	0.295	0.757	44
	64	30.00	3.67	0.541	0.326	0.750	31
	128	32.00	3.82	0.541	0.328	0.752	28
	192	33.00	3.82	0.542	0.332	0.750	28
Fars	2	4.00	2.40	0.742	0.680	0.803	1519
	4	5.20	3.23	0.713	0.581	0.737	793
	8	5.80	2.69	0.722	0.582	0.798	440
	16	5.40	2.29	0.741	0.655	0.841	240
	32	6.60	2.63	0.726	0.594	0.810	171
	64	6.60	2.31	0.734	0.625	0.858	98
	128	6.40	2.24	0.740	0.626	0.866	84
	192	8.20	2.42	0.721	0.586	0.850	80
Census	2	22.20	5.62	0.638	0.787	0.967	881
	4	25.00	5.80	0.641	0.778	0.968	449
	8	30.20	6.00	0.645	0.763	0.969	238
	16	32.80	5.82	0.655	0.739	0.971	130
	32	35.20	6.04	0.652	0.739	0.970	81
	64	36.80	6.19	0.657	0.726	0.972	62
	128	43.20	6.04	0.670	0.680	0.975	65
	192	44.20	6.06	0.671	0.668	0.976	77
KDD	2	12.80	10.68	0.785	0.898	0.855	50515
	4	13.80	10.77	0.777	0.949	0.923	25043
	8	16.20	9.89	0.771	0.894	0.893	13713
	16	19.00	10.59	0.769	0.921	0.921	7211
	32	17.80	10.68	0.784	0.979	0.925	4028
	64	17.20	8.98	0.790	0.974	0.927	2371
	128	17.60	9.11	0.804	0.979	0.921	1600
	192	15.80	8.94	0.817	0.976	0.930	5678
RLCP	2	5.50	2.72	0.811	0.997	0.950	38393
	4	6.00	2.83	0.828	0.997	0.934	8536
	8	7.20	2.90	0.852	0.997	0.939	4146
	16	7.00	2.88	0.849	0.997	0.940	2415
	32	7.00	2.94	0.849	0.997	0.940	1370
	64	7.00	2.88	0.849	0.997	0.940	1039

Table 4 shows the results. In this table can be observed the following:

- The interpretability of the models obtained in the executions with different maps varies between executions with a low and high number of maps. In this way, there is an increase of the number of rules while the number of variables is maintained. This is a logical situation due to the nature of the algorithm: as a rule set is obtained for each map, the more maps used implies the more number of rules obtained. However, it is important to remark that the token competition operator applied in the reduce function follows to optimise the final number of

[2] http://ceatic.ujaen.es/en.

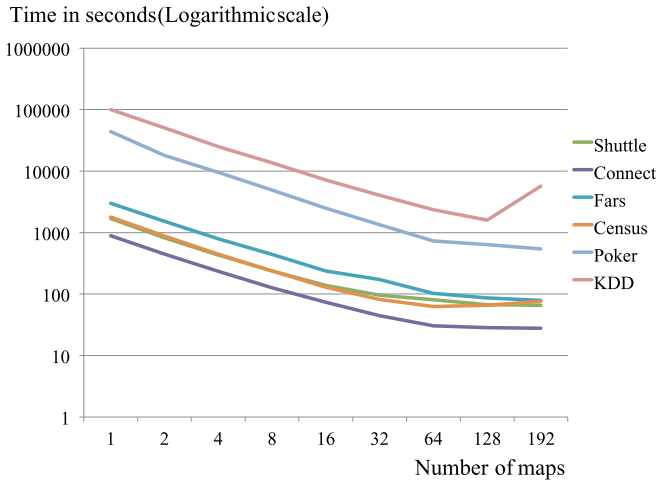

Fig. 4. Runtime of the different datasets.

rules obtained, with only a soft increase when using a high number of maps.

- The values of unusualness (interest and novelty of the rules) are maintained for all the datasets except of shuttle; i.e. the new algorithm has a good behaviour with respect to the gain of information of the subgroups obtained, and there is no loss of information even when a high number of maps is used. The problem in dataset shuttle is related to the use of a number of maps greater than 32, since this dataset is not large enough to make many divisions.
- Respect to the relation between sensitivity and confidence, it is important to note that the generality of the subgroups is reduced when the precision increases; i.e. the algorithm tends to excessive specialization when increasing the number of divisions. In fact, this reduction of the generality is associated to the increase of the number of subgroups obtained, although among all the subgroups appear rules with very similar values to those obtained in the original problem.
- The runtime of the MEFASD-BD algorithm is reduced when the number of maps is increased, as can be observed in Fig. 4. This means that one of the reasons why the new algorithm for high dimensional datasets has been developed, reducing runtime, is satisfied; in fact, reductions close to 90% of the runtime are obtained in complex datasets such as KDD or RLCP. It must be also noted the difficulty of obtaining results for the algorithm with the RLCP dataset when the number of maps is greater than 100; that is why Table 4 does not include results for 128 and 192 maps for the RLCP dataset. This problem appears for memory limitations of the machines.
- The experimental study shows the difficulty of determining a general optimal number of maps, because it is possible to find better results with a number of maps or another depending on the problem. For example, the connect dataset has an important loss of sensitivity when the number of maps is greater than 4, the shuttle dataset has a good behaviour with 32 maps, and fars and census datasets obtain the best results with 64 or 128 maps, in the same way as for RLCP and KDD. Definitely, the study shows a soft relation between the number of maps and the complexity of the datasets, i.e. it would be possible to obtain better results using 64 maps or more with high dimensional datasets.

According to these results, the new MEFASD-BD algorithm for big data environments shows an excellent behaviour in this type of problems, with a huge reduction of the runtime while maintaining the quality of the subgroups obtained.

5. Concluding remarks and future work

This paper presents the MEFASD-BD algorithm, the first proposal of an SD algorithm for big data environments. The new algorithm uses the basic ideas of the algorithm NMEEF-SD in any of its stages and it is implemented in Scala using the Spark Apache approach.

MEFASD-BD implements new functions based on the MapReduce paradigm and uses both global and local quality measures in order to sort and consider only rules with good gain of information with respect to the whole dataset. Moreover, the algorithm implements a token competition operator in the reduce phase to obtain a high reduction in the final number of rules also preserving the diversity in the final population.

The results of the experiments performed demonstrate a good behaviour of the new algorithm in high dimensional datasets, with a huge reduction of the runtime and maintaining the levels of interpretability, trade-off between sensitivity and confidence, and interest respect to the original algorithm.

The good results of the new algorithm make us think that this is a promising line of work. As a future work, we consider the implementation of other SD algorithms for big data environments, the development of new operators and the study of new quality measures for this task. It would be also interesting to apply these algorithm in real problems, such as bioinformatics, where huge dimensional datasets are used; in fact, in this area the advance in fields such as genetic map is generating huge amounts of information that could be analysed from the point of view of SD.

Acknowledgement

This work was supported by the Spanish Ministry of Economy and Competitiveness under project TIN2015-68454-R (FEDER Founds).

References

[1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287.
[2] M. Atzmueller, F. Puppe, SD-Map - a fast algorithm for exhaustive subgroup discovery, in: Proceedings of the 17th European Conference on Machine Learning and 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, vol. 4213 of LNCS, Springer, 2006, pp. 6–17.
[3] M. Atzmueller, F. Puppe, H.P. Buscher, Towards knowledge-intensive subgroup discovery, in: Proceedings of the Lernen - Wissensentdeckung - Adaptivität - Fachgruppe Maschinelles Lernen, 2004, pp. 111–117.
[4] S. Bay, M. Pazzani, Detecting group differences: mining contrast sets, Data Min. Knowl. Discovery 5 (2001) 213–246.
[5] V. Bolón-Canedo, N.S.-M. no, A. Alonso-Betanzos, Recent advances and emerging challenges of feature selection in the context of big data, Knowl.-Based Syst. 86 (2015) 33–45.
[6] C.J. Carmona, C. Chrysostomou, H. Seker, M.J. del Jesus, Fuzzy rules for describing subgroups from influenza a virus using a multi-objective evolutionary algorithm, Appl. Soft Comput. 13 (8) (2013) 3439–3448.
[7] C.J. Carmona, P. González, M. del Jesus, F. Herrera, Overview on evolutionary subgroup discovery: analysis of the suitability and potential of the search performed by evolutionary algorithms, WIREs Data Min. Knowl. Discovery 4 (2) (2014) 87–103.
[8] C.J. Carmona, P. González, M.J. del Jesus, F. Herrera, NMEEF-SD: Non-dominated multi-objective evolutionary algorithm for extracting fuzzy rules in subgroup discovery, IEEE Trans. Fuzzy Syst. 18 (5) (2010) 958–970.
[9] C.J. Carmona, P. González, M.J. del Jesus, M. Navío, L. Jiménez, Evolutionary fuzzy rule extraction for subgroup discovery in a psychiatric emergency department, Soft Comput. 15 (12) (2011) 2435–2448.
[10] C.J. Carmona, P. González, M.J. del Jesus, C. Romero, S. Ventura, Evolutionary algorithms for subgroup discovery applied to e-learning data, in: Proceedings of the IEEE International Education Engineering, 2010, pp. 983–990.
[11] C.J. Carmona, P. González, M.J. del Jesus, S. Ventura, Subgroup discovery in an e-learning usage study based on moodle, in: Proceedings of the International Conference of European Transnational Education, 2011, pp. 446–451.
[12] C.J. Carmona, P. González, B. García-Domingo, M.J. del Jesus, J. Aguilera, MEFES: an evolutionary proposal for the detection of exceptions in subgroup discovery. An application to concentrating photovoltaic technology, Knowl.-Based Syst. 54 (2013) 73–85.

[13] C.J. Carmona, S. Ramírez-Gallego, F. Torres, E. Bernal, M.J. del Jesus, S. García, Web usage mining to improve the design of an e-commerce website: Orolivesur.com, Expert Syst. Appl. 39 (2012) 11243–11249.
[14] C.J. Carmona, V. Ruiz-Rodado, M.J. del Jesus, A. Weber, M. Grootveld, P. González, D. Elizondo, A Fuzzy genetic programming-based algorithm for subgroup discovery and the application to one problem of pathogenesis of acute sore throat conditions in humans, Inf. Sci. 298 (2015) 180–197.
[15] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, in: Operating Systems Design and Implementation (OSDI), 2004, pp. 137–150.
[16] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[17] K. Deb, A. Pratap, S. Agrawal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[18] M.J. del Jesus, P. González, F. Herrera, M. Mesonero, Evolutionary Fuzzy Rule Induction Process for Subgroup Discovery: A case study in marketing, IEEE Trans. Fuzzy Syst. 15 (4) (2007) 578–592.
[19] G.Z. Dong, J.L. Li, L. Wong, New Generation of Data Mining Applications, Chap. the Use of Emerging Patterns in the Analysis of Gene Expression Profiles for the Diagnosis and Understanding of Diseases, John Wiley, 2005, pp. 331–354.
[20] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computation, Springer, 2003.
[21] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, F. Herrera, Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study, IEEE Trans. Evol. Comput. 14 (6) (2010) 913–941.
[22] D.B. Fogel, Evolutionary Computation - Toward a New Philosophy of Machine Intelligence, IEEE Press, 1995.
[23] D. Galpert, S. Río, F. Herrera, E. Ancede, A. Antunes, G. Aguero-Chapin, An effective big data supervised imbalanced classification approach for ortholog detection in related yeast species, BioMed Res. Int. 2015 (2015) 1–12.
[24] D. Gamberger, N. Lavrac, Expert-guided subgroup discovery: methodology and application, J. Artif. Intell. Res. 17 (2002) 501–527.
[25] J. Gantz, D. Reinsel, The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east, IDC iView 2007 (2012) 1–16.
[26] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., 1989.
[27] H. Grosskreutz, S. Rueping, On subgroup discovery in numerical domains, Data Min. Knowl. Discovery 19 (2) (2009) 210–216.
[28] M. Hamstra, H. Karau, M. Zaharia, A. Konwinski, P. Wendell, Learning Spark: Lightning-Fast Big Data Analytics, O'Reilly Media, 2015.
[29] F. Herrera, Genetic fuzzy systems: Taxomony, current research trends and prospects, Evol. Intell. 1 (2008) 27–46.
[30] F. Herrera, C.J. Carmona, P. González, M.J. del Jesus, An overview on subgroup discovery: foundations and applications, Knowl. Inf. Syst. 29 (3) (2011) 495–525.
[31] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press,
[32] InternetLiveStats.com, Internet live stats (2016). URL http://www.internetlivestats.com/one-second/.
[33] B. Kavsek, N. Lavrac, APRIORI-SD: Adapting association rule learning to subgroup discovery, Appl. Artif. Intell. 20 (2006) 543–583.
[34] W. Kloesgen, Explora: a multipattern and multistrategy discovery assistant, in: Advances in Knowledge Discovery and Data Mining, American Association for Artificial Intelligence, 1996, pp. 249–271.
[35] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.
[36] P. Kralj-Novak, N. Lavrac, G.I. Webb, Supervised descriptive rule discovery: a unifying survey of constrast set, emerging pateern and subgroup mining, J. Mach. Learn. Res. 10 (2009) 377–403.
[37] N. Lavrac, B. Cestnik, D. Gamberger, P.A. Flach, Decision support through subgroup discovery: three case studies and the lessons learned, Mach. Learn. 57 (1–2) (2004) 115–143.
[38] N. Lavrac, P.A. Flach, B. Zupan, Rule Evaluation Measures: A Unifying View, in: Proceedings of the Ninth International Workshop on Inductive Logic Programming, vol. 1634 of LNCS, Springer, 1999, pp. 174–185.
[39] N. Lavrac, B. Kavsek, P.A. Flach, L. Todorovski, Subgroup discovery with CN2-SD, J. Mach. Learn. Res. 5 (2004) 153–188.
[40] F. Lemmerich, M. Rohlfs, M. Atzmueller, Fast discovery of relevant subgroup patterns, in: Twenty-Third International FLAIRS Conference, 2010.
[41] K.S. Leung, Y. Leung, L. So, K.F. Yam, Rule learning in expert systems using genetic algorithm: 1, concepts, in: K. Jizuka (Ed.), Proc. of the Second International Conference on Fuzzy Logic and Neural Networks, 1992, pp. 201–204.
[42] V. López, S. Río, J.M. Benítez, F. Herrera, Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data, Fuzzy Sets Syst. 258 (2015) 5–38.
[43] M. Minelli, M. Chambers, A. Dhiraj, Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses, John Wiley and Sons, 2013.
[44] D. Peralta, S. Río, S. Ramíez-Gallego, I. Triguero, J.M. Beníez, F. Herrera, Volutionary feature selection for big data classification: A mapreduce approach, Math. Probl. Eng. 2015 (2015) 1–11.
[45] A.H. Project, 2015, Apache hadoop, http://hadoop.apache.org/.
[46] J. Qian, P. Lv, X. Yue, C. Liu, Z. Jing, Hierarchical attribute reduction algorithms for big data using mapreduce, Knowl.-Based Syst. 73 (2015) 18–31.
[47] S. Río, V. López, J.M. Benítez, F. Herrera, On the use of mapreduce for imbalanced big data using random forest, Inf. Sci. 285 (2014) 112–137.
[48] S. Río, V. López, J.M. Benítez, F. Herrera, A MapReduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules, Int. J. Comput. Intell. Syst. 8 (3) (2015). 442–437
[49] C. Romero, P. González, S. Ventura, M.J. del Jesus, F. Herrera, Evolutionary algorithm for subgroup discovery in e-learning: a practical application using moodle data, Expert Syst. Appl. 36 (2009) 1632–1644.
[50] J. Sanchez-Riera, K. Hua, Y. Hsiao, T. Lim, S. Hidayati, W. Cheng, A Comparative study of data fusion for rgb-d based visual recognition, Pattern Recogn. Lett. 73 (2016) 1–6.
[51] I. Timón, J. Soto, H. Pérez-Sánchez, J. Cecilia, Parallel implementation of fuzzy minimals clustering algorithm, Expert Syst. Appl. 48 (2016) 35–41.
[52] D. Trabold, H. Grosskreutz, Parallel subgroup discovery on computing clusters-first results, in: Big Data, 2013 IEEE International Conference on, IEEE, 2013, pp. 575–579.
[53] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, MRPR: A MapReduce solution for prototype reduction in big data classification, Neurocomputing 150 (2015a) 331–345.
[54] I. Triguero, S. Río, V. López, J. Bacardit, J.M. Benítez, F. Herrera, OSEFW-RF: the winner algorithm for the ECBDL'14 big data competition: an extremely imbalanced big data bioinformatics problem, Knowl.-Based Syst. 87 (2015b) 69–79.
[55] L. Wang, H. Geng, P. Liu, K. Lu, J. Kolodziej, R. Ranjan, A.Y. Zomaya, Particle swarm optimization based dictionary learning for remote sensing big data, Knowl.-Based Syst. 79 (2015) 43–50.
[56] T. White, 2012, Hadoop, The Definitive Guide, O'Reilly Media.
[57] A. Wibisono, W. Jatmiko, H. Wisesa, B. Hardjono, P. Mursanto, Traffic big data prediction and visualization using fast incremental model trees-drift detection (fimt-dd), Knowl.-Based Syst. 93 (2016) 33–46.
[58] M.L. Wong, K.S. Leung, Data Mining Using Grammar Based Genetic Programming and Applications, Kluwer Academics Publishers, 2000.
[59] S. Wongthanavasu, J. Ponkaew, A cellular automata-based learning method for classification, Expert Syst. Appl. 49 (2016) 99–111.
[60] S. Wrobel, An Algorithm for Multi-relational Discovery of Subgroups, in: Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery, vol. 1263 of LNAI, Springer, 1997, pp. 78–87.
[61] S. Wrobel, Inductive Logic Programming for Knowledge Discovery in Databases, in: Relational Data Mining, Springer, 2001, pp. 74–101. Chap.
[62] M. Wurst, M. Scholz, Distributed Subgroup Mining, in: Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006), Springer, 2006, pp. 421–433.
[63] D. Xia, B. Wang, H. Li, Y. Li, Z. Zhang, A Distributed spatial-temporal weighted model on mapreduce for short-term traffic flow forecasting, Neurocomputing 179 (2016). 246–26
[64] L.A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning. Parts I, II, III, Information Science 8–9 (1975). 199–249, 301–357, 43–80.
[65] L.A. Zadeh, Soft computing and fuzzy logic, IEEE Softw. 11 (6) (1994) 48–56.
[66] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, I. Stoica, Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing, in: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, 2012.
[67] M. Zaharia, M. Chowdhury, M. Franklin, S. Shenker, I. Stoica, Spark: cluster computing with working sets, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, 2010. 10–10