Impact of the type of rule in Fuzzy Emerging Pattern Mining on a Big Data Approach

A.M. García-Vico[1], P. González[1], C.J. Carmona[2], and M.J. del Jesus[1]

[1] Department of Computer Science, University of Jaén. 23071, Jaén (Spain)
agvico@ujaen.es, pglez@ujaen.es, mjjesus@ujaen.es
[2] Lenguages and Computer Technology Systems, Department of Civil Engineering, University of Burgos. 09006, Burgos (Spain)
cjcarmona@ubu.es

Abstract. New scenarios are now emerging with increasing amounts of data due to the massive use of the Internet and new technologies. Traditional methodologies are unable to handle this huge amount of data, so new approaches towards distributed paradigms, such as MapReduce, have to be designed. This environment is widely known in the literature as Big Data. The interpretability of the results is a key aspect in Emerging Pattern Mining. In this study, the influence of the type of rule used to extract knowledge in an evolutionary fuzzy algorithm for Emerging Pattern Mining in Big Data environment is analysed. The adaptation of the EvAEFP-Spark algorithm to extract disjunctive formal norm rules is also presented.

Keywords: Emerging pattern mining, evolutionary fuzzy systems, Big Data, Spark

1 Introduction

Today we live in a world in which huge amounts of data are continually being generated. This produces enormous volumes of data, with variety of formats that arrives at high velocity, making traditional data mining methods unable to process this "Big Data" [16]. The term "Big Data" is currently a hot topic in industry and academia [9], where most of the methods developed are based on the MapReduce paradigm [2]. A very important advantage of the MapReduce paradigm is that allows the creation of scalable methods, a key aspect in data science to obtain more accurate knowledge [25].

Emerging Pattern Mining (EPM) is part of the supervised descriptive rule discovery (SDRD) framework [18]. This data mining task tries to find patterns whose support increases significantly from one class, or dataset, to another [5]. EPM has been successfully applied in several cases in different fields [20, 19, 26]. However, the EPM task is extremely complex when the volume of data is huge due to the non-convexity of the problem [23]. When the size of the datasets increases, the number and complexity of the emerging patterns extracted by the EPM algorithms also greatly increases. This makes the interpretability of

the results obtained a key aspect to take into account. In this sense, rules are a suitable tool for the representation of knowledge in the extraction of information describing emerging patterns.

One of the more suitable approaches to deal with these problems are the evolutionary fuzzy systems (EFS) [13]. These systems are a hybridisation of evolutionary algorithms (EAs) [1] and fuzzy logic [27]. EAs are a well-known optimisation methods that efficiently search for the best solutions on huge search spaces. Additionally, the use of fuzzy logic allows to obtain a representation of the knowledge closer to human reasoning and makes it unnecessary to discretise numerical variables.

The objective of this work is to analyse the influence of the type of rule used to represent the knowledge in EPM in a Big Data environment using EFSs. To do so, we present an extension of a fuzzy EPM algorithm for Big Data problems, EvAEFP-Spark [12], allowing the use not only of canonical but disjunctive normal form (DNF) rules to represent the knowledge. To do so, the paper is organised as follows: Section 2 briefly introduces the main concepts used in this paper. Section 3 presents the extension of the EvAEFP-Spark algorithm, an evolutionary proposal for EPM under Big Data environments, in order to use DNF rules. Finally, Section 4 presents an experimental study for the comparison of the results of the algorithm using canonical or DNF rules and Section 5 introduces the concluding remarks.

2 Preliminaries

This section presents the main concepts used in the work. This way, Section 2.1 contains a brief background of EPM, Section 2.2 describes the concept of EFS, and finally Section 2.3 reviews the MapReduce paradigm.

2.1 Emerging Patterns Mining

Dong and Li [5, 7] defined EPM as the task of finding all patterns with a growth rate (GR) value greater than a threshold $\rho \geq 1$. Given two datasets D_1 and D_2, the GR of a pattern x is defined in Eq. 1.

$$GR(x) = \begin{cases} 0, & IF\ Sup_{D_1}(x) = Sup_{D_2}(x) = 0, \\ \infty, & IF\ Sup_{D_1}(x) \neq 0 \wedge Sup_{D_2}(x) = 0, \\ \frac{Sup_{D_1}(x)}{Sup_{D_2}(x)}, & another\ case \end{cases} \tag{1}$$

where $Sup_{D_i}(x) = \frac{count_{D_i}(x)}{|D_i|}$ is the support of the pattern in the dataset i. The aims of EPM are to find emerging tendencies in timestamped datasets, discriminative characteristics among classes or searching for differences among variables.

An important aspect to take into account is that EPM is a non-convex problem [23], i.e., more general patterns can have lower GR values with respect to more specific ones. Hence, it is usual to deal with huge search spaces in EPM,

also obtaining a large number of EPs. As a result, there have been attempts throughout the literature to filter or reduce the number of patterns obtained, in order to only obtain high quality EPs. To do so, concepts like jumping EPs [6], strong jumping EPs [8], or fuzzy EPs [10], among others, have been introduced and joined with different search strategies in order to make EPM faster. In the literature, different methods can be find following the border-based approach, like the DeEPS method [15], and following a tree-based approach, with methods like SJEP-C [8] and FEPM [10], among others.

2.2 Evolutionary Fuzzy Systems

An EFS is a hybridisation of fuzzy systems and a learning process in an EA [13]. EAs are algorithms based on natural evolution to deal with optimisation problems with complex search spaces. They have a good trade-off between the quality of the results and the time needed to obtain such results. EAs have been widely used in the literature in this kind of problems.

Fuzzy systems are based on fuzzy logic [27], so allowing to handle uncertainty. In addition, fuzzy systems allow the representation of numeric variables in a more human-readable way by using linguistic labels (LLs), where a fuzzy set represent a single label on the variable. These labels can be defined by the user or using uniform partitions if knowledge is unavailable.

EFSs can be applied in EPM as a process to search for EPs in a huge and complex search space. The objective is then a rule learning process that allows to obtain interpretable knowledge through the use of IF-THEN rules with fuzzy logic statements. These systems are known as fuzzy rule-based systems (FRBSs). They are a well-suited method to obtain simple and interpretable knowledge.

2.3 MapReduce

The EPs algorithms have difficulties when applied to Big Data problems. MapReduce [2, 3] is one of the most popular programming paradigms to deal with Big Data. It is composed of two main parts, the map and the reduce phases that works as follows: the map phase obtain some intermediate results from the input data, and the reduce phase joins all those intermediate results to produce the final result.

Spark [28] is one of the most popular frameworks implementing the MapReduce paradigm. It is a highly efficient cluster computing framework for large-scale data processing based on an intensive use of main memory. This fact allows Spark to improve the performance on iterative algorithms like EAs in orders of magnitude with respect to other frameworks such as Apache Hadoop [24].

3 The EvAEFP-Spark Algorithm

The algorithm EvAEFP-Spark [12] is an adaptation of the EvAEFP algorithm [11] for Big Data problems. EvAEFP-Spark extracts an undetermined number of

fuzzy EPs in order to describe trends from supervised learning. The fuzzy EPs can be formalised as rules in order to obtain more interpretable results. This paper presents an extension to the EvAEFP-Spark algorithm allowing the use of a different type of fuzzy rules to represent the knowledge extracted.

EvAEFP-Spark employs an EA with a "Chromosome = Rule" codification, where each individual in the population represents a potential rule. In this method, only the antecedent part of the rule is represented. This means that, in order to obtain rules for all possible values of the target class, this method must be executed once for each class.

The algorithm uses fuzzy rules when the features are continuous, and the fuzzy sets corresponding to the linguistic labels (LLs) are defined by means of the corresponding membership functions. These can be specified by the user or defined by means of a uniform partition if expert knowledge is not available. In this work, uniform partitions with triangular membership functions are used. An example with five uniform LLs can be observed in Fig. 1.

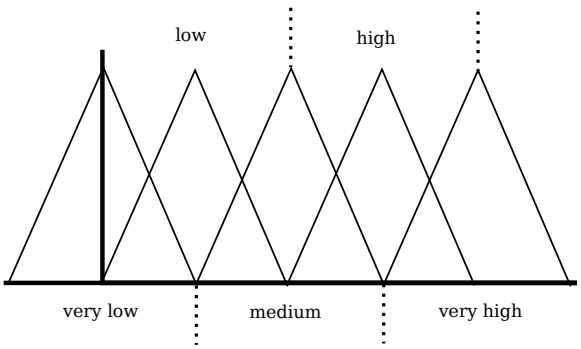

Fig. 1. Represention of a numeric variable with five uniform linguistic labels.

In the original version, EvAEFP-Spark uses canonical rules, where the antecedent of a rule is composed of a conjunction of value-variable pairs, and the value 0 is used to indicate that the variable is not considered for the rule. In Fig 2 a phenotype and genotype representations of an individual is presented.

$$\textit{Genotype}$$

x_1	x_2	x_3
2	0	3

$$\Downarrow$$

$$\textit{Phenotype}$$

$$IF\ (x_1 = LL_1^2) \wedge (x_3 = LL_3^3)\ THEN\ (x_{Obj} = Class)$$

Fig. 2. Representation of a canonical rule in the EvAEFP-Spark algorithm.

The extension presented in this paper allows the use of DNF rules, where each variable can take more than one value. A DNF rule represents the knowledge in a flexible and compact way and facilitates the extraction of more general rules. A fixed-length binary representation is used for a DNF rule, in which one bit for each of the possible values of every feature is stored. Therefore, if the corresponding bit contains the value 0 it indicates that the value is not used in the rule (value 1 indicates that the value is used). In Fig 3 a phenotype and genotype representations of an individual in DNF form is presented.

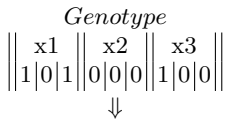

Fig. 3. Representation of a DNF rule in the EvAEFP-Spark algorithm.

3.1 Main properties and operational scheme

Algorithm EvAEFP-Spark is a mono-objective EA with an iterative rule learning (IRL) approach [22]. Therefore, the algorithm extracts the best rule of the evolutionary process and iterates until a stopping criteria is reached, i.e., the execution is stopped when the last pattern is not an EP, it does not cover new examples, or it has a null support. The evolutionary algorithm used is generational with elitism, where the best individual is kept in the population of the next generation, and uses as genetic operators a tournament selection [17] with two individuals, a two-point crossover operator [14] and a biased mutation operator presented in [4].

Fig 4 shows the operational scheme of EvAEFP-Spark. First, the algorithm reads the data and splits it into disjoint partitions, then sending each partition to a computing node. Next, the IRL approach begins by generating the initial population for the target class by means of a biased procedure. 50% of the individuals in the initial population are generated randomly and the remaining are generated with a maximum of an 80% of its variables initialised. At the end of the evolutionary process, the best rule is extracted and added to the final result set. This process is repeated while the stopping condition is not satisfied.

A key concept of an EA is the fitness function. This is used to evaluate the quality of an individual. In EvAEFP-Spark, the fitness function is defined as:

$$Fitness(R) = \sqrt{TPr * TNr} \tag{2}$$

where a geometric mean between the true positive rate (TPr) and the true negative rate (TNr) is used in order to obtain a balanced way to maximise the precision and generality of a potential rule.

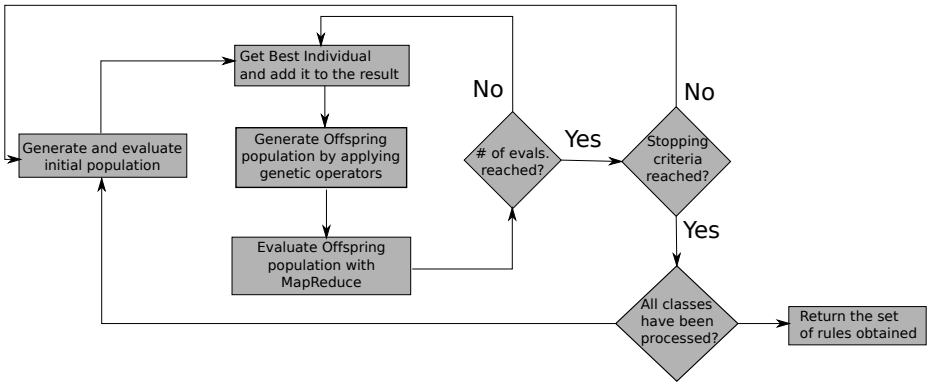

Fig. 4. General Operational scheme of the EvAEFP-Spark algorithm.

3.2 MapReduce approach

The fitness function is the most expensive task of an EA because it involves traversing the complete dataset once for each individual in the population, in order to calculate the TPr and TNr. When using EAs in a Big Data environment for the EPM task, it is necessary to express the results by means of a confusion matrix in order to analyse the fitness function. Table 1 represents a confusion matrix for an emerging rule with tp as the number of examples correctly covered, fn such as the number of examples for the class not covered, fp is the number of examples incorrectly covered, and tn is the number of examples non-covered for the not class. The measures used in the fitness function can be easily calculated from this confusion matrix, since $TPr(R) = \frac{tp}{tp+fn}$ and $TNr = \frac{tn}{tn+fp}$.

Table 1. Confusion matrix for a rule

	Predicted	
Real	Positive	Negative
Positive	tp	fn
Negative	fp	tn

In EvAEFP-Spark the fitness function has been developed following the MapReduce paradigm in order to reduce the complexity when evaluating the population. On each generation of the EA, the map phase sends those individuals generated by genetic operators and, thus, not evaluated yet, to the data partitions. Then, on each map, a confusion matrix is calculated for each individual. After that, the reduce phase performs the sum of all the possible confusion matrices, getting only one confusion matrix for each individual with the correct values to calculate the measures back on the driver. A scheme of this MapReduce approach is shown in Figure 5.

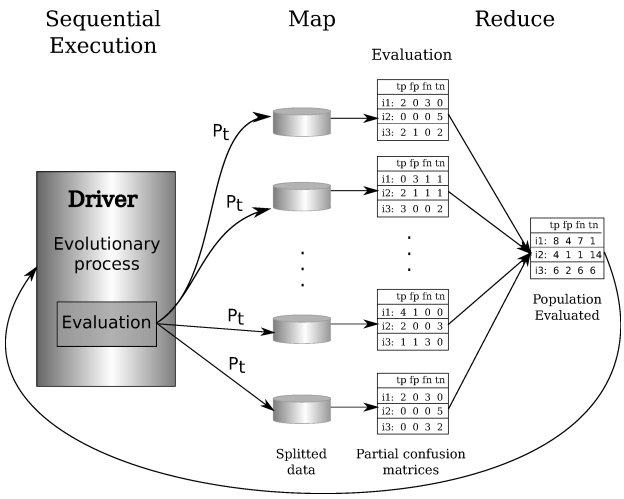

Fig. 5. Scheme of the evaluation procedure of the EvAEFP-Spark algorithm following a MapReduce approach.

4 Experimental study

An experimental study with some big datasets available in UCI repository [21] has been performed in order to compare the results of the EvAEFP-Spark algorithm with both canonical and DNF rules. The main characteristics of these datasets, namely, the name of the dataset, the number of instances, the number of features and their types (Real, Integer and Nominal), and the number of classes are summarised in Table 2.

Table 2. Characteristics of the datasets used in the experimental framework.

Dataset	Instances	Features (R/I/N)	Classes
census	299284	41 (1/12/28)	3
kddcup	494020	41 (26/0/15)	23
rlcp	5749132	11 (11/0/0)	2
susy	5000000	18 (18/0/0)	2

For the comparison, a 5-fold stratified cross validation scheme is used in order to avoid the arbitrariness of the classical k-fold cross validation and the dependence of the results with respect to the partitioning performed. The parameters used in the execution of the algorithm are summarised in Table 3.

As introduced in [12], when running algorithm EvAEFP-Spark it is necessary to specify the number of partitions used to divide the input data. In this experiments 128 partitions have been used. To perform the comparison between

Table 3. Parameters used for the EvAEFP-Spark algorithm in the experiments.

Parameter	Value
Number of fuzzy labels	3
Number of evaluations	10000
Population length	100
Crossover probability	0.60
Mutation probability	0.01

the results of the two types of representations for the knowledge extracted, the quality of the EPs extracted is analysed through the following quality measures:

- GR. It is the main quality measure in EPM and it is defined in Eq. 1. However, the range for this quality measure ($[0, \infty]$) implies the necessity to measure the percentage of patterns that are EPs on the test set.
- True Positive Rate (TPr). It measures the ratio of true positive examples covered by the rule, giving a generality measure of the rule obtained.
- False Positive rate (FPr). It measures the ratio of false positive examples covered by the rule, giving a precision measure for the rule obtained.

Table 4. Results of the algorithm EvAEFP-Spark with canonical and DNF rules

Dataset	Rule	GR	TPR	FPR	Time
census	Can	0.819	0.534	**0.430**	**2100**
	DNF	**0.913**	**0.649**	0.467	2264
kddcup	Can	0.602	0.416	**0.002**	**11960**
	DNF	**1.000**	**0.748**	0.748	13596
rlcp	Can	**1.000**	0.951	**0.013**	**2621**
	DNF	**1.000**	**0.957**	0.014	22067
susy	Can	**0.533**	0.302	**0.211**	**9929**
	DNF	**0.533**	**0.386**	0.299	29160

The results of the experimental study are shown in Table 4. In general, the values obtained using canonical and DNF rules are very similar, with a small advantage in favour of the DNF rules. However, the minimum advantage of the DNF representation is greatly diluted considering the huge amount of time expended in obtaining such patterns. This time difference is much more relevant with the volumen of data is higher. In addition, they are more complex and less interpretable rules than canonical rules.

5 Conclusions

An extension of the EvAEFP-Spark algorithm to handle DNF rules is presented in this paper. In a DNF rule, more than one value can be taken by each variable,

so allowing to represent the knowledge in a flexible and compact way and facilitating the extraction of more general rules. The experiments on datasets in a Big Data environment does not show a significant advantage of one representation over the other. However, the time to obtain DNF rules is significantly higher than canonical. Therefore, selecting one type or another of rules will mainly depend on the type of knowledge to be extracted.

Acknowledgments

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

References

1. T. Bäck, D. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*. Oxford University Press, 1997.
2. J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters", *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
3. J. Dean and S. Ghemawat, "MapReduce: A flexible data processing tool", *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.
4. M. J. del Jesus, P. González, F. Herrera, and M. Mesonero, "Evolutionary Fuzzy Rule Induction Process for Subgroup Discovery: A case study in marketing", *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 4, pp. 578–592, 2007.
5. G. Z. Dong and J. Y. Li, "Efficient Mining of Emerging Patterns: Discovering Trends and Differences", in *Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 1999, pp. 43–52.
6. G. Z. Dong, J. Y. Li, and X. Zhang, "Discovering jumping emerging patterns and experiments on real datasets", in *Proc. on International Database Conference Heterogeneous and Internet Databases*, 1999, pp. 155–168.
7. G. Z. Dong and J. Y. Li, "Mining border descriptions of emerging patterns from dataset pairs", *Knowledge and Information Systems*, vol. 8, no. 2, pp. 178–202, 2005.
8. H. Fan and K. Ramamohanarao, "Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers", *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 6, pp. 721–737, 2006.
9. A. Fernandez, S. del Rio, V. Lopez, A. Bawakid, M. J. del Jesus, J. M. Benitez, and F. Herrera, "Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks", *Wiley Interdisciplinary Reviews-Data Mining and Knowledge Discovery*, vol. 4, no. 5, pp. 380–409, 2014.
10. M. García-Borroto, J. Martínez, and J. Carrasco-Ochoa, "Fuzzy emerging patterns for classifying hard domains", *Knowledge and Information Systems*, vol. 28, no. 2, pp. 473–489, 2011.
11. A. M. García-Vico, J. Montes, J. Aguilera, C. J. Carmona, and M. J. del Jesus, "Analysing Concentrating Photovoltaics Technology through the use of Emerging Pattern Mining", in *Proc. of the 11th International Conference on Soft Computing Models in Industrial and Environmental Applications*. Springer, 2016, pp. 1–8.

12. A. M. García and C. J. Carmona and P. González and M. J. del Jesus, "A first approach to handle fuzzy emerging patterns mining on big data problems: the EvAEFP-Spark algorithm", in *Proc. of the 2017 International Conference on Fuzzy Systems (FUZZ-IEEE 2017)*, 2017 (Accepted).
13. F. Herrera, "Genetic fuzzy systems: taxomony, current research trends and prospects", *Evolutionary Intelligence*, vol. 1, pp. 27–46, 2008.
14. J. H. Holland, "Adaptation in natural and artificial systems", *University of Michigan Press*, 1975.
15. J. Y. Li, G. Z. Dong, K. Ramamohanarao, and L. Wong, "DeEPs: A New Instance-Based Lazy Discovery and Classification System", *Machine Learning*, vol. 54, no. 2, pp. 99–124, 2004.
16. S. Madden, "From databases to big data", *IEEE Internet Computing*, vol. 16, no. 3, 2012.
17. B. L. Miller and D. E. Goldberg, "Genetic Algorithms, Tournament Selection, and the Effects of Noise", *Complex System*, vol. 9, pp. 193–212, 1995.
18. P. Kralj-Novak, N. Lavrac, and G. I. Webb, "Supervised Descriptive Rule Discovery: A Unifying Survey of Constrast Set, Emerging Pateern and Subgroup Mining", *Journal of Machine Learning Research*, vol. 10, pp. 377–403, 2009.
19. R. Sherhod, V. J. Gillet, T. Hanser, P. N. Judson, and J. D. Vessey, "Toxicological knowledge discovery by mining emerging patterns from toxicity data", *Journal of Chemical Information and Modeling*, vol. 5, no. S-1, p. 9, 2013.
20. G. Tzanis, I. Kavakiotis, and I. P. Vlahavas, "Polya-iep: A data mining method for the effective prediction of polyadenylation sites", *Expert Systems with Applications*, vol. 38, no. 10, pp. 12 398–12 408, 2011.
21. A. Asuncion and D. J. Newman, "UCI Machine Learning Repository", 2007. [Online]. Available: http://www.ics.uci.edu/ mlearn/MLRepository.html
22. G. Venturini, "SIA: A Supervised Inductive Algorithm with Genetic Search for Learning Attributes based Concepts", in *Proc. European Conference on Machine Learning*, ser. LNAI, vol. 667. Springer, 1993, pp. 280–296.
23. L. Wang, H. Zhao, G. Dong, and J. Li, "On the complexity of finding emerging patterns", in *Proc. of the 28th Annual International Computer Software and Applications Conference*, vol. 2, 2004, pp. 126–129.
24. T. White, *Hadoop, The Definitive Guide*, O'Reilly Media, 2012.
25. X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data", *IEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014.
26. Y. Yu, K. Yan, X. Zhu, and G. Wang, "Detecting of PIU Behaviors Based on Discovered Generators and Emerging Patterns from Computer-Mediated Interaction Events", in *Proc. of the 15th International Conference on Web-Age Information Management*, ser. LNCS, vol. 8485. Elsevier, 2014, pp. 277–293.
27. L. A. Zadeh, "The concept of a linguistic variable and its applications to approximate reasoning. Parts I, II, III", *Information Science*, vol. 8-9, pp. 199–249,301–357,43–80, 1975.
28. M. Zaharia, M. Chowdhury, M. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets", in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10, 2010, pp. 10–10.