A Transformation Approach Towards Big Data Multilabel Decision Trees

Antonio Jesús Rivera Rivas(✉), Francisco Charte Ojeda, Francisco Javier Pulgar, and Maria Jose del Jesus

Department of Computer Science, University of Jaén, Jaén, Spain
{arivera,fcharte,fpulgar,mjjesus}@ujaen.es

Abstract. A large amount of the data processed nowadays is multilabel in nature. This means that every pattern usually belongs to several categories at once. Multilabel data are abundant, and most multilabel datasets are quite large. This causes that many multilabel classification methods struggle with their processing. Tackling this task by means of big data methods seems a logical choice. However, this approach has been scarcely explored by now. The present work introduces several big data multilabel classifiers, all of them based on decision trees. After detailing how they have been designed, their predictive performance, as well as the execution time, are analyzed.

Keywords: Multilabel classification · Big data · Decision trees

1 Introduction

Pattern classification is among the most popular machine learning (ML) tasks. Usually, each data pattern is associated to one category (the class label). Starting from a set of previously labeled samples, classification algorithms train a model (the classifier). Once trained, the classifier can be shown new unlabeled patterns and it produces the predicted label as output. Decision trees (DT) are well-known classifiers [1], quick to build and easily interpretable. DT ensembles, such as Random Forest (RF) [2], are also quite effective, producing good predictive performance.

A large part of the data generated nowadays is made of patterns linked to several categories at once, instead of only one. Music clips can produce a subset of the existing emotions [3], images can be categorized into several groups [4], blog posts and questions in forums are assigned a set of tags [5], etc. The task of learning from data pattern which are assigned several labels is known as multilabel classification (MLC) [6]. The use of DTs in MLC, multilabel DTs (MDTs), is also a common option.

The amount of new images, videos, music clips, blog posts and other multilabel contents uploaded everyday to the Internet is impressive. As a consequence, MLC algorithms have to be able to process large datasets, a work that

I. Rojas et al. (Eds.): IWANN 2017, Part I, LNCS 10305, pp. 73–84, 2017.
DOI: 10.1007/978-3-319-59153-7_7

usually takes a long time. Facing this problem through Big Data (BD) techniques, notably by distributing the workload among a group of machines, seems a natural choice. Nonetheless, it is a barely explored alternative.

This work aims to propose five different MDT implementations based on BD techniques. Three of them are BD multilabel versions of the well-known ID3 [7], CART [8] and C4.5 [9] algorithms. The other two are ensembles of MDTs, as it is known that ensembles tend to improve classification results. In addition, ensembles are easier to parallelize in a BD environment than other techniques. The five proposals will be experimentally tested with a double goal. Firstly, the predictive performance of each alternative will be compared. Secondly, the improvement in running time as the number of parallel nodes is increased is analyzed.

The remainder of this paper is structured as follows. Section 2 introduces the foundations of MLC and some concepts related to DTs designed to work with BD infrastructure. The five proposed MDTs for BD are presented in Sect. 3. In Sect. 4 the experimental framework is detailed and results are discussed. Finally, some conclusions are drawn in Sect. 5.

2 Preliminaries

The methods presented in Sect. 3 are MDTs designed for BD environments. Therefore, it is essential to know the foundations of MLC, introduced in Subsect. 2.1, as well as some notions about BD infrastructures such as Hadoop and Spark, brought in Subsect. 2.2.

2.1 Multilabel Classification

Multilabel datasets (MLDs) emerge naturally in certain fields, such as music and video categorization [3,10], image tagging [4], document classification [11] or gene function identification [12]. An MLD can be defined as a subset of $A_1 \times A_2 \times ... \times A_f \times \mathcal{P}(\mathcal{L})$, being A_i the f input features and $\mathcal{P}(\mathcal{L})$ the powerset of $\mathcal{L}$, the full set of labels appearing in the data. There is no difference between the input space of an MLD and a traditional dataset. By contrast, the output space of the former is made of a set of 0s and 1s (labelset), stating which of the labels in $\mathcal{L}$ are relevant for each pattern. Therefore, a classifier have to be able to predict several outputs simultaneously.

Aside of the number of labels, which can be seen as the length of the labelset, several other metrics can be extracted from the samples' labelsets [13]. The two most common are label cardinality (Card) and label density (Dens). Y_i being the labelset of the ith-instance in the MLD $\mathcal{D}$, Card (1) is simply the average number of relevant labels in the MLD. Dens (2) is the normalized label cardinality[1].

[1] Card, Dens and many other multilabel characterization metrics can be easily obtained with the mldr package [14].

$$Card = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}|Y_i| \qquad (1) \qquad\qquad Dens = \frac{1}{|\mathcal{L}|}\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}|Y_i| \qquad (2)$$

MLC has been faced mainly through two different approaches [6], data transformation and method adaptation. The former aims to produce binary or multiclass datasets from the MLD, so that they can be processed with traditional classification algorithms. The latter, on the contrary, advocates for rewriting these traditional algorithms, making them able to work with multilabel data natively.

Although several transformation methods have been proposed in the literature, two of them stand out because are frequently used as foundation of many other algorithms. They are Binary Relevance (BR) and Label Powerset (LP).

- BR consists in producing as many binary datasets as labels there are in the MLD, training an independent classifier for each label. The predictions provided by these classifiers are joined to obtain the final labelset. Obviously, the number of models to build (and the time needed to do it) increases linearly with the number of labels.
- Whereas BR relies in binary classifiers, LP do it in multiclass ones. The trick lies in considering each distinct labelset as class identifier. The major drawback of this approach is that theoretically $2^{|\mathcal{L}|}$ different classes could exist.

Regarding the method adaptation approach, multilabel algorithms based on the best known models, such as trees [15], neural networks [16], support vector machines [12] or nearest neighbors [17], can be found in the literature.

2.2 Decision Trees for Big Data

The recent advances in communications and storage technologies have led to the emergence of big databases, in a context where "big" has to be understood as beyond the processing capacity of current personal computers. The answer to this scenario was the use of clusters of computers. For facing complex ML tasks different BD frameworks have been developed over time, such as Hadoop and Spark [18], both from the Apache Foundation.

Hadoop relies on an own distributed file system [19], named HDFS, and the approach to distribute the workload is the popular Map-Reduce [20]. Unlike Hadoop, Spark supports in-memory data sharing. This technique produces a noticeable improvement in running time, notably when multiple-pass computations over the data are needed. Depending on specific conditions, Spark runs as 100 times faster than Hadoop. In addition, a basic library of ML methods running over Spark, named MLlib [21], is available. Among the provided ML algorithms, a generic ID3/CART DT can be found.

The cornerstone of Spark is the Resilient Distributed Dataset (RDD). It represents a data collection that is distributed among a set of machines (cluster nodes). Spark is able to cache RDDs in memory, reusing them between successive

parallel operations. In the following, we refer to the number of nodes used to process the data as number of RDD partitions. The goal of an Spark cluster is to reduce the total running time by distributing the workload among its nodes. Therefore, the more nodes there are in the cluster, the less time will be spent in processing the data. However, partitioning and distributing the data is also a time-consuming task. Depending on the amount of data, this preparatory work could take longer than the reduction obtained by the parallelization. So the number of RDD partitions is a parameter that could need some adjustment.

3 Multilabel Decision Trees for Big Datasets

Taking as foundation the generic tree implementation provided by MLlib [21], the data mining library for Spark, three MDT algorithms were designed, ID3, CART and C4.5. All of them are based on the LP transformation, previously defined, so the labelsets are taken as class identifiers. Then, two ensembles of MDTs are also proposed, BR and RF. The details about these proposals are provided in the following subsections.

3.1 Classifiers Based on Single MDT

To learn a single MDT from an MLD, instead of a collection of binary trees, the LP transformation has been used. Thus, each labelset in an MLD is taken as the class identifier. In addition, multilabel versions of entropy and the Gini index, the metrics used to decide the variable used in each split of the tree, are needed.

Based on [15], and being $\mathcal{L}$ the full set of labels in the MLD, $p(l)$ the probability of l being relevant and $q(l) = 1 - p(l)$, the entropy measurement is defined as shown in (3). Analogously, (4) corresponds to the Gini index computation.

$$Entropy = -\sum_{l \in \mathcal{L}} p(l) \log p(l) + q(l) \log q(l) \tag{3}$$

$$Gini = 1 - \sum_{l \in \mathcal{L}} p(l)^2 + q(l)^2 \tag{4}$$

Based on the MLlib implementation of ID3 [7], a multilabel version using (3) and the LP transformation was implemented. In the same way, the MLlib's version of CART [8] was adapted to work with MLDs, using (4) and the same transformation. Since C4.5 is not available in MLlib, it was written as extension of the existing ID3 algorithm following [9]. This implied essentially implementing the pruning procedure of C4.5, producing smaller trees with a better ability of generalization.

The main difference between the classical implementation of these methods and the one made here, based on Spark, is that the latter parallelizes the task of evaluating the goodness of the attributes to be used in each split.

3.2 Classifiers Based on Ensembles

Tree ensembles, such as RF, are among the most popular and best performing classifiers. Since they train several independent models, ensembles are specially suitable for work distribution in a cluster. Each tree will be built independently, once the data partitions have been sent to each node.

The first ensemble is based on the BR transformation, using C4.5 as underlying classifier. Therefore, there will be as many trees as labels in the MLD. Each one will learn to differentiate patterns for which one label is relevant against all the others, as explained in Sect. 2. The predictions provided by the individual trees, at test time, are later combined to get the full labelset.

RF is proposed as the second MDT ensemble. As in BR, this approach also generates multiple trees. However each one is a MDT processing all labels, not a binary tree. A random subset of the input features is chosen to train the trees, as usual in RF. The trees are built with the multilabel C4.5 version described in the previous subsection. The maximum tree depth is set to 5 and the ensemble uses 100 trees, as recommended in [22].

4 Experimentation

In this section how the previously described methods have been empirically tested is explained. Section 4.1 outlines the experimental framework. The conducted experimentation has two main goals. Firstly, the classification performance produced by each one of the MDT implementations will be assessed in Sect. 4.2. Later, in Sect. 4.3, the execution time of each method, as well as the influence of the number of partitions in running time, will be analyzed. This way, the best algorithm could be chosen according to time restrictions and classification performance demands, as discussed in Sect. 4.4.

4.1 Experimental Framework

The MDTs have been tested using six MLDs[2] having disparate traits, as shown in Table 1. Three of them, medical [24], slashdot [25] and tmc2007 [11] come from the text domain, emotions [3] and scene [4] have their origin in the multimedia sources, while yeast [12] was produced from genetic data. The number of instances and attributes will mainly impact the execution time of the algorithms. On the other hand, the number of labels and cardinality are attributes that influence the predictive behavior of the models. Depending on the transformation applied to the data, the number of labels can also increase the running time.

Each MLD was partitioned following a 5 folds cross validation, thus each run used 80% of data to train the model and the remainder 20% as test patterns. Reported results are mean values over these 5 runs per MLD/method.

[2] All of them can be found in the RUMDR [23] repository.

Table 1. Main characteristics of the MLDs

Dataset	Instances	Attributes	Labels	Cardinality
emotions	593	72	6	1.869
medical	978	1 449	45	1.245
scene	2 407	294	6	1.074
slashdot	3 782	1 079	22	1.181
tmc2007	28 596	500	22	2.158
yeast	2 417	198	14	4.237

Aiming to analyze how the number of RDD partitions affected execution time, the training and testing was repeated six times using a different configuration. The used values are 1, 2, 4, 8, 16, 32 and 64. Theoretically, as the number of RDD partitions grows execution time should decrease, since the work is distributed among a larger amount of machines. The cluster used has 14 nodes and each node disposes of 2 x Intel Xeon E5-2670v2 and 64 GB of RAM.

The predictions made by each classifier were assessed by means of five performance metrics. Let Y_i be the ground truth labelset of the ith-instance, Z_i the predicted one, Δ the symmetrical difference, and $[\![]\!]$ the Iverson operator (returns 1 if the expression is true, 0 otherwise). Hamming Loss (5), Accuracy (6), F-Measure (8), and Subset Accuracy (9) are defined as follows. Hamming Loss is a loss metric, so the lower the value the better is performing the classifier. For the other four metrics, higher values are better.

$$Hamming\ Loss = \frac{1}{n}\frac{1}{k}\sum_{i=1}^{n}|Y_i \Delta Z_i| \tag{5}$$

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{6}$$

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|}, \quad Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|} \tag{7}$$

$$F\text{-}Measure = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}. \tag{8}$$

$$Subset\ Accuracy = \frac{1}{n}\sum_{i=1}^{n}[\![Y_i = Z_i]\!] \tag{9}$$

Micro F-Measure (10) differs from F-Measure in the way it is averaged. Instead of computing the metric for each instance, the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) for all the instances are aggregated, then the measure is computed. $\mathcal{L}$ denotes the full set of labels appearing in the MLD. Additional details about all these metrics can be found in [6].

$$Micro\ F\text{-}Measure = F\text{-}Measure(\sum_{l\in\mathcal{L}} TP_l, \sum_{l\in\mathcal{L}} FP_l, \sum_{l\in\mathcal{L}} TN_l, \sum_{l\in\mathcal{L}} FN_l) \quad (10)$$

In addition to the previous metrics, running times were also gathered to compare the influence of the number of RDD partitions in the total execution time.

4.2 Classification Performance

Firstly, the interest is in determining which one of the MDTs produces better classification results. The values corresponding to each evaluation metric are depicted in Fig. 1. Each bar plot shows results for the six MLDs processed with the five algorithms. As can be observed, the bar associated to RF is noticeable

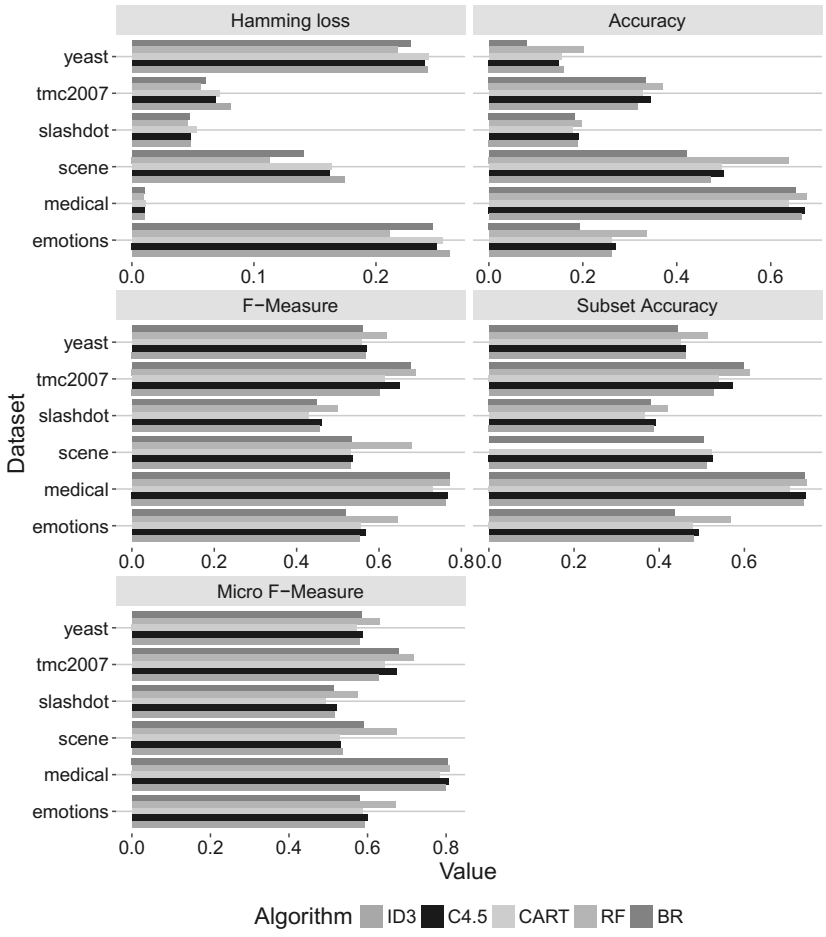

Fig. 1. Classification results

Table 2. Classification results

Metric	Dataset	ID3	C4.5	CART	RF	BR
Hamming Loss ↓						
	emotions	0.260	0.250	0.254	**0.211**	0.246
	scene	0.174	0.162	0.163	**0.113**	0.140
	yeast	0.242	0.240	0.243	**0.218**	0.228
	slashdot	0.048	0.048	0.053	**0.045**	0.047
	medical	**0.010**	**0.010**	0.011	**0.010**	0.011
	tmc2007	0.081	0.068	0.072	**0.056**	0.061
Accuracy ↑						
	emotions	0.481	0.492	0.479	**0.567**	0.436
	scene	0.512	0.526	0.522	**0.668**	0.504
	yeast	0.461	0.461	0.451	**0.513**	0.442
	slashdot	0.388	0.392	0.365	**0.420**	0.381
	medical	0.738	0.743	0.707	**0.745**	0.741
	tmc2007	0.528	0.572	0.541	**0.612**	0.598
Subset Accuracy ↑						
	emotions	0.261	0.269	0.261	**0.336**	0.194
	scene	0.472	0.500	0.496	**0.639**	0.421
	yeast	0.160	0.149	0.155	**0.202**	0.080
	slashdot	0.190	0.191	0.177	**0.198**	0.183
	medical	0.666	0.673	0.638	**0.675**	0.653
	tmc2007	0.316	0.344	0.327	**0.371**	0.334
F-Measure ↑						
	emotions	0.553	0.568	0.554	**0.644**	0.519
	scene	0.532	0.535	0.531	**0.678**	0.533
	yeast	0.568	0.570	0.557	**0.618**	0.560
	slashdot	0.455	0.460	0.429	**0.499**	0.447
	medical	0.763	0.767	0.731	**0.770**	**0.770**
	tmc2007	0.603	0.649	0.614	**0.690**	0.676
Micro F-Measure ↑						
	emotions	0.591	0.600	0.587	**0.670**	0.579
	scene	0.537	0.532	0.528	**0.673**	0.590
	yeast	0.580	0.587	0.573	**0.630**	0.585
	slashdot	0.517	0.522	0.492	**0.574**	0.514
	medical	0.797	0.806	0.785	**0.808**	0.805
	tmc2007	0.627	0.675	0.642	**0.718**	0.678

above the others (below for Hamming Loss) in many cases. BR seems to be the closer contender, performing at the same level than RF sometimes.

The exact values are provided in Table 2. Best results have been highlighted in bold. From these values, that RF is the best performer can be drawn. It achieves the highest (lowest for Hamming Loss) value in all cases, with only a pair of ties.

To better elucidate how each algorithm compare to others regarding classification performance, in Table 3[3] all of them have been ranked. The rightmost column shows the average ranking for all performance metrics. As can be seen, the second best performer is C4.5, ahead of the BR transformation. By contrast, CART gets the worst results.

[3] Names of metrics have been abbreviated to better fit them as column captions.

Table 3. Average ranking by metric

Algorithm	HL	Acc	F-M	SA	MF-M	Avg. rank
RF	1.000	1.000	1.083	1.000	1.000	1.017
C4.5	2.833	2.250	2.333	2.333	2.500	2.450
BR	2.333	4.000	3.250	4.333	3.167	3.417
ID3	4.333	3.583	3.833	3.500	3.667	3.783
CART	4.500	4.167	4.500	3.833	4.667	4.333

4.3 Execution Time Analysis

The main goal of distributing the workload among a group of machines is to reduce the total execution time taken by the process. In this case, the process is the training of each classifier. The number of RDD partitions have been set to different values, aiming to analyze at which extent increasing the parallelization level decreases running time.

Since it has been already proven that ID3 and CART produce poor classification performance, time analysis will be focused in the other three MDT implementations. Figure 2 shows execution times in seconds for each MLD and method. The X axis is common to all plots, indicating the number of RDD partitions. Y axes are independent, stating the running time in seconds. Experiments taking longer than 10 h were discarded, this is the reason to the lacking of data points in tmc2007 for 1 and 2 partitions.

As would be expected, in general running time decreases as the number of RDD partitions grows. However, there are a few exceptions such as RF while trained with emotions, medical and yeast. In these cases increasing the number of partitions from 32 to 64 implies a deterioration instead of an improvement, taking significantly longer. This could be explained by the fact that the process of dividing the problem and distributing it among the machines in the cluster, takes longer than the savings obtained by sharing the workload.

4.4 Discussion

From the observation of the previous results, choosing the best MDT alternative is a matter of deciding what is most important in each case, predictive performance or running time. To obtain the best possible classification of new patterns RF is the correct choice, with a large advantage over the other algorithms. RF is an ensemble, a collection of C4.5 trees each of them generated from a random subset of the features. Therefore, obtaining better results than a single C4.5 classifier is not strange. BR is also an ensemble, but each one of the trees is focused in predicting one label only, working independently of the other trees. The approach of creating several trees taking the relationship among labels into account, through the LP transformation, proves to be superior.

As would be expected, the running times for the ensembles, BR and RF, are longer than for the single C4.5 MDT. However, increasing the number of RDD

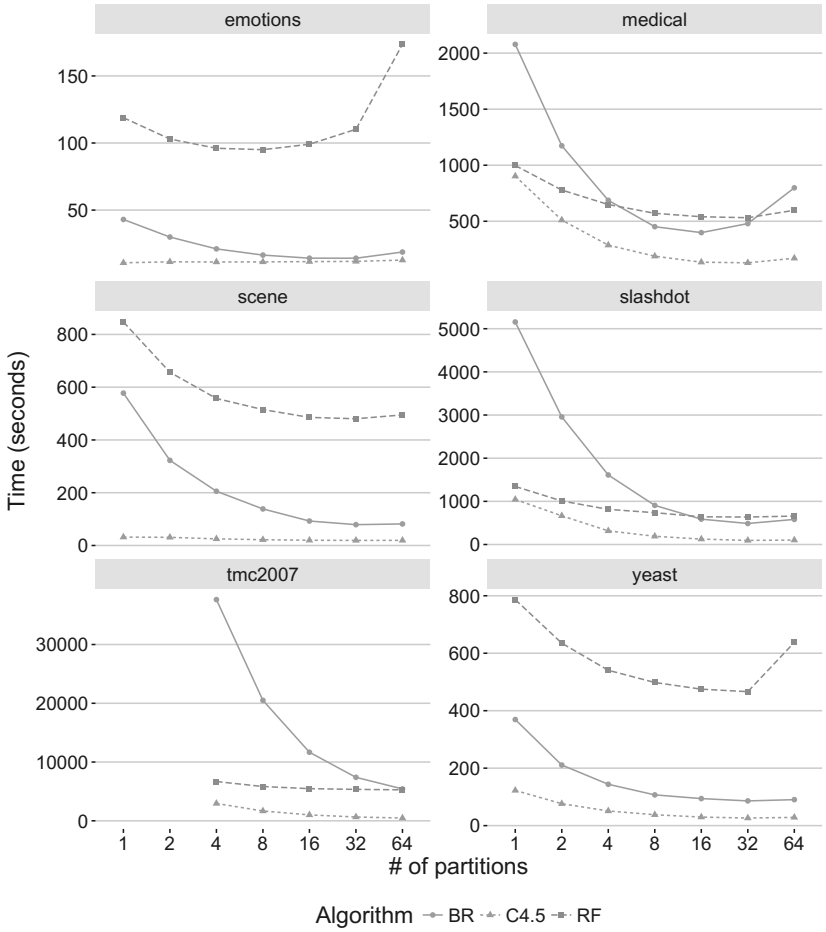

Fig. 2. Execution time vs number of partitions

partitions reduces these times until they get quite close in some cases. As can be observed in Fig. 2, BR is slower with the MDLs which have more labels, such as tmc2007 and slashdot, since it has to produce a larger set of classifiers. On the contrary, RF is more affected by the number of samples and attributes, as it has to produce bigger trees as these numbers grow. In general, for MLDs with many labels RF will produce the best classification results in less time than BR, although depending on the number of RDD partitions some surpassing could exists (as with the medical MLD). C4.5 running times are always the lowest, but they only benefit from distributing the work among machines with the larger MLDs. As can be stated from Fig. 2, for MLDs such as emotions, scene and yeast the line for C4.5 is almost flat.

Overall, given its predictive performance and for being able to reduce running time as the number of RDD partitions is increased, RF seems to be the best decision when it comes to choose a multilabel decision tree algorithm for big data environments.

5 Conclusions

The amount of new data patterns produced every day is huge, mainly in form of images, videos, sounds and texts due to the emergence of services such as Flickr, Instagram, YouTube! and personal blogs. These patterns are multilabel in nature, and could be grouped/categorized/tagged into several classes. Hence the interest in having methods able to perform multilabel classification with big databases.

In this work five different decision tree based methods have been proposed. Three of them are multilabel versions of well-known ID3, CART and C4.5 algorithms, using an adapted entropy/Gini metric and based on the LP transformation. The other two, RF and BR, are ensembles of classifiers, following two distinct approaches. The former trains several trees with a subset of the input features and all labels, while the latter trains an individual tree for each label with all the features.

A two-way experimental study has been conducted. The first part has led as result that RF is the best choice when only predictive performance matters. The second part analyzed how the total running time could be reduced by increasing the number of RDD partitions. The behavior of RF and BR was dependent of the MLD characteristics, noticeably the number of labels. As ensemble methods, their running time was always higher than that of single-MDT classifiers such as C4.5.

Acknowledgments. This work is partially supported by the Spanish Ministry of Science and Technology under project TIN2015-68454-R.

References

1. Kotsiantis, S.: Supervised machine learning: a review of classification techniques. In: Proceedings of Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real World AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, pp. 3–24. IOS Press (2007)
2. Svetnik, V., Liaw, A., Tong, C., Culberson, J.C., Sheridan, R.P., Feuston, B.P.: Random forest: a classification and regression tool for compound classification and qsar modeling. J. Chem. Inf. Comput. Sci. **43**(6), 1947–1958 (2003)
3. Wieczorkowska, A., Synak, P., Raś, Z.: Multi-label classification of emotions in music. In: Kłopotek, M.A., Wierzchoń, S.T., Trojanowski, K. (eds.) Intelligent Information Processing and Web Mining. AISC, vol. 35, pp. 307–315. Springer, Heidelberg (2006)
4. Boutell, M., Luo, J., Shen, X., Brown, C.: Learning multi-label scene classification. Pattern Recogn. **37**(9), 1757–1771 (2004)
5. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: QUINTA: a question tagging assistant to improve the answering ratio in electronic forums. In: Proceedings of IEEE International Conference on Computer as a Tool, EUROCON 2015, pp. 1–6. IEEE (2015)
6. Herrera, F., Charte, F., Rivera, A.J., Del Jesus, M.J.: Multilabel Classification: Problem Analysis, Metrics and Techniques. Springer, Heidelberg (2016)

7. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
8. Steinberg, D., Colla, P.: CART: Tree-Structured Non-Parametric Data Analysis. Salford Systems, San Diego (1995)
9. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993). ISBN 1-55860-238-0
10. Snoek, C.G.M., Worring, M., van Gemert, J.C., Geusebroek, J.M., Smeulders, A.W.M.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: Proceedings of 14th ACM International Conference on Multimedia, MULTIMEDIA 2006, pp. 421–430 (2006)
11. Srivastava, A.N., Zane-Ulman, B.: Discovering recurring anomalies in text reports regarding complex space systems. In: Aerospace Conference, pp. 3853–3862. IEEE (2005)
12. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems, vol. 14, pp. 681–687. MIT Press (2001)
13. Herrera, F., Charte, F., Rivera, A.J., del Jesus, M.J.: Case studies and metrics. Multilabel Classification, pp. 33–63. Springer, Cham (2016). doi:10.1007/978-3-319-41111-8_3
14. Charte, F., Charte, D.: Working with multilabel datasets in R: the mldr package. R. J. **7**(2), 149–162 (2015)
15. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: Raedt, L., Siebes, A. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 42–53. Springer, Heidelberg (2001). doi:10.1007/3-540-44794-6_4
16. Zhang, M.: Ml-rbf: RBF neural networks for multi-label learning. Neural Process. Lett. **29**, 61–74 (2009)
17. Zhang, M., Zhou, Z.: ML-KNN: a lazy learning approach to multi-label learning. Pattern Recogn. **40**(7), 2038–2048 (2007)
18. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. HotCloud **10**(10–10), 95 (2010)
19. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
20. Gillick, D., Faria, A., DeNero, J.: Mapreduce: distributed computing for machine learning, Berkley, 18 December 2006
21. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al.: Mllib: machine learning in apache spark. J. Mach. Learn. Res. **17**(34), 1–7 (2016)
22. del Río, S., López, V., Benítez, J.M., Herrera, F.: On the use of mapreduce for imbalanced big data using random forest. Inf. Sci. **285**, 112–137 (2014)
23. Charte, F., Charte, D., Rivera, A., de Jesus, M.J., Herrera, F.: R ultimate multilabel dataset repository. In: Martínez-Álvarez, F., Troncoso, A., Quintián, H., Corchado, E. (eds.) HAIS 2016. LNCS (LNAI), vol. 9648, pp. 487–499. Springer, Cham (2016). doi:10.1007/978-3-319-32034-2_41
24. Crammer, K., Dredze, M., Ganchev, K., Talukdar, P.P., Carroll, S.: Automatic code assignment to medical text. In: Proceedings of Workshop on Biological, Translational, and Clinical Language Processing, BioNLP 2007, pp. 129–136. Association for Computational Linguistics (2007)
25. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Mach. Learn. **85**, 333–359 (2011)