LSIS: Large Scale Instance Selection Algorithm for Big Data

Reine Marie Marone[1], Fodé Camara[2], Samba Ndiaye[1]
[1]Department of mathematics, Cheikh Anta Diop University, Dakar, Senegal
[2]Departement of mathematics, Alioune Diop University, Bambey, Senegal
e-mail: fode.camara@uadb.edu.sn, reine.marie.marone@ucad.edu.sn

***Abstract*—Recently enormous volumes of data are generated in Information Systems, and data mining area is facing new challenges of transforming this "big data" into useful knowledge. To get from "big data" a manageable volume, we propose a large scale instance selection for reducing the initial dataset, leading to a reduction of both time taken and the computational resources that are necessary for performing the learning process, and improving the accuracy of classifier model. Our experimental results demonstrated that the proposed algorithms could scale well and efficiently process large datasets by selecting relevant instances for classification problem. The experimental results show also the contribution of the instance selection on the classification accuracy.**

Keywords- instance selection; big data; parallel computing; apache spark™

I. INTRODUCTION

Instance selection is an important data pre-processing issue for data mining especially with the context of big data where applications deal with very extremely large datasets. Big data can be defined as high volume, velocity and variety of data that has the potential to be mined for information, and require a new high-computational infrastructure to ensure successful data processing and analytics. Several frameworks for large-scale processing have tried to face the big data issues in last decade. Among them we can cite, Apache Hadoop and Apache SparkTM that are the two so popular big data frameworks. To get from "big data" a manageable volume, pre-processing task such as instance selection can be very crucial. However standard algorithms for instance selection are not suitable for Hadoop or Spark frameworks and must be re-designed (sometimes, entirely) to learn from large-scale datasets. That presents a big challenge for researchers, and for this reason we propose a new large-scale instance selection that we called LSIS (for Large Scale Instance Selection), using the Apache SparkTM.

The rest of the paper is organized as follows. Section II gives some preliminaries. Section III discusses related works. In section IV, we give the details of our proposition. In Section V, we evaluate the performance of our algorithm. Section VI concludes the paper and gives some future works.

II. PRELIMINARIES

A. Instance selection

Generally there are noisy or superfluous instances in datasets and therefore it is necessary to remove these instances. Instance selection methods allow reducing the size of data by removing irrelevant instances in order to improve the performance of an instance-based learning algorithm. In instance-based classifiers, instance selection techniques must be able to reduce training time of a classifier and obtain the same or even better classification rates than those achieved using the full data set [14].Classical instance selection methods can be divided into two: wrapper and filter. Wrapper techniques perform instance selection using a classification model and depend on accuracy achieved by the classifier. Separated test are performed in the blocks of instances of the dataset by training a model. Finally the accuracy of each model is evaluated and the subset with the high accuracy is selected [14]. Filter-based instance are usually not tailored to a classifier. These methods evaluated instances according to heuristics based on over all data characteristics in order to identify instances that can be safely removed from training data [14]. Filter-based instance selection techniques are typically faster than wrapper based techniques. Additional, filters are more general than wrappers and are closely coupled with a learning algorithm [14]. This is the reason why we propose in this paper a filter-based instance selection.

B. Spark Parallel Computing Framework

Apache SparkTM is an open-source cluster-computing framework. In contrast to Hadoop's disk-based MapReduce model, Spark's in-memory primitives provide performance up to 10 times faster for certain data mining applications [13]. By allowing user programs to load data into a cluster's memory and analyze it iteratively, Apache SparkTM is well suited to data mining algorithms, which are often iterative. Apache SparkTM requires a cluster manager and a distributed storage system. For distributed storage, Apache SparkTM can interface with a wide variety of systems, including Hadoop Distributed File System (HDFS), Cassandra and Amazon S3 [13].

Apache Spark comes with high-level APIs in Java, Scala and Python, and recently R.

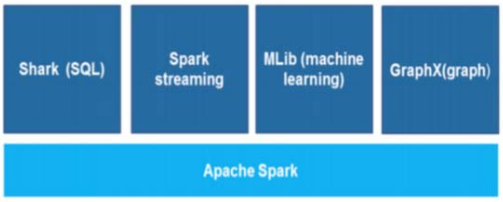

Figure 1.Apache Spark API's.

978-1-5090-6352-9/17/$31.00 ©2017 IEEE

It also supports a rich set of higher-level tools, including MLlib, a machine learning library that provides various algorithms designed to scale out on a cluster for classification, regression, clustering, collaborative filtering, and so on.

C. Spark Programming Model

Spark use MapReduce paradigm that is a programming model and has an interesting benefit for big data applications because it simplifies the processing of massive volumes of data through its efficient and cost-effective mechanisms.

Map-Reduce programming model is composed of two subsequent methods that handle data computations: (i) the Map function and (ii) the Reduce function [15].

D. Architecture of Spark

Apache Spark uses master/worker architecture and has a cluster manager that dispatches work for the cluster[16]. A spark cluster has a single coordinator called master and several slaves/workers. The driver program that runs on the master node of the spark cluster schedules the job execution and negotiates with the cluster manager. Worker is running spark instance where is created a distributed agent responsible for the execution of tasks called executor. Executor runs several individual tasks in a given Spark job. Figure below illustrates the architecture of Apache Spark Cluster.

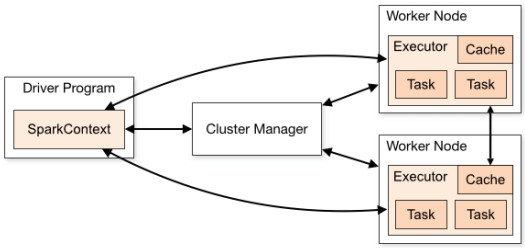

Figure 2. Architecture of Apache Spark Cluster.

III. Related Works

Instance reduction is an important task in the data preparation phase of data mining. In the literature, various approaches for instances reduction have been proposed. According to the strategy used, we can divide the instance selection methods into two groups: (i) supervised instances filters and (ii) unsupervised instances filters. [1] gives a comparative study of them. The main drawback of instance selection methods is their computational complexities that are generally quadratic $O(n^2)$, where n is the number of instances [2]; thus, the majority of them are impracticable in datasets with many thousands of instances as in big data context [2].

TABLE I: Summary on State-of-the-art Of non Parallel Instance Selection Methods

Algorithm	Year	Reference
CDIS	2016	[5]
LSBo	2015	[7]
LDIS	2015	[6]
SV-kNNC	2006	[8]
CNN	1968	[3]

Another drawback is related to the fact that most of the instance selection algorithms are tailored for nearest neighbor classifier, so the instances selected with these algorithms are often only suitable for nearest neighbor classifiers.

To deal with big data, novels proposals were presented recently in. [8] [9] [10]. All these large-scale algorithms [8] [9] [10] develop the same idea; get manageable volume from big data by reducing the memory required to store the data, and therefore accelerating the classification algorithms.

TABLE II: Summary on State-of-the-art of Parallel Instance Selection Methods

Large scale Algorithm	Year	Reference
MRDIS	2017	[9]
MRVIS	2016	[10]
MRPR	2015	[11]

IV. Our Proposal

A. Problem definition

To address the data explosion issues, we propose both scale-up and data reduction algorithm. We can formalize this as follows: Let D be a massive training set with a big number of instances, our Large-Scale Instance Selection algorithm consists to find a subset S of instances from D such as $f(S) \geq f(D)$ using Map-reduce, the well-known paradigm of parallel computing. Notice that f represents the SVM classifier model.

B. The LSIS algorithm

SVM is one of the top picks in data mining and has been used successfully to classify linearly separable and nonlinearly separable data with high accuracy. In the literature many studies reveal that rank features using SVM models yields good performances. For this reason, in LSIS algorithm we combine Support vector machine (SVM) with our criterion to score the instances in order to get better results.

LSIS algorithm can be broken into three steps:

In step 1: For each instance, each element encountered in the instance will be associated with the value 1. The value 1 means that the attribute was encountered once in the instance. This corresponds to the following statement of the algorithm:

Foreach *instance line* $I_i \subset I$
  Foreach a_i *in instance line*
    mapToPair(a_i =>(a_i,1))
  EndForeach
EndForeach

Then for each element a_i, the sum of the obtained values 1 is computed in order to get the total number of appearance

of a_i in the data set (frequency of the element). It represents a reduce operation under spark which is done as follows:

$rdd\ [(a_i, f_i)] = reduceByKey(\_+\_)$

It is also used to calculate the maximum frequency of each element as follows:

$rdd[(a_i, f_{max})] = rdd[(a_i, f_i)].$
$reduceByKey(math.max(\_, \_))$

In Step 2: The weight vector of svm classifier is first computed,and the score of each instance is calculated as follows:

$$Score\ (I_j) = \sum_{i=1}^{m} (w_i / norm(w))/[\ (f_i)_{max} - f_i)]$$

Where w_i, $norm(w)$, f_i and f_{max} represents respectively the weight of the attribute a_i, the norm of weight vector w, the frequency of the ith-attribute of instance I_j, and the maximal frequency of the ith-attribute values.

Each instance I_i is mapped to the pair $(I_j, score)$:

$mapToPair(I_j => (I_j, score))$

In Step 3: Finally, the workers send the pairs $(I_i, score)$ to the master that returns the k instances with the best score.

Algorithm **LSIS**

Input: Dataset D (with n instances and m attributes)

Output: k best inliers

Begin

/*First step*/

Map begin

1: $I = flatmap(line => f.getInstance())$

2: Foreach *instance line* $I_i \subset I$

3: *Foreach* a_i *in instance line*

4: $mapToPair\ (a_i => (a_i, 1))$

5: *EndForeach*

6: ***EndForeach***

End.

Reduce begin

$rdd\ [(a_i, f)] = reduceByKey(\_+\_)$

$rdd[(a_i, f_{max})] = rdd[(a_i, f_i)].reduceByKey(math.max(\_, \_))$

End

/*Second step*/

Map begin

Foreach *instance line* $I_i \subset I$

$mapToPair(I_i => (I_i, score))$ *where*

$$score = \sum_{i=1}^{m} (w_i / norm\ (w))/(f_{max} - f_i)$$

EndForeach

End

Return select ***instances*** *according to the* ***k*** highest *scores*

End.

V. Performance Evaluation

The experiments were performed on a cluster consisting of 4 workers nodes and one head node. The head node has 2 cores Intel® Xeon® E5 processors running at 2.60 GHz, with 28 GB memory and a 200 GB disk. And each worker node has 4 cores Intel® Xeon® E5 processors running at 2.60 GHz, with 14 GB memory and 200 GB disk. The computing nodes are all running at the Linux-based HDInsight (Spark) cluster and HDI 3.3.

We used two benchmark real-world big data sets chosen in [12] and some statistics of those datasets are presented in Table III.

TABLE III. Characteristics of Benchmark Datasets

Name	|I|	|a|
real-sim	72,309	20,958
kddb-raw-libsvm	19, 264,097	1, 163,024

Table 3: Characteristics of benchmark datasets

In our experimentations, we focus on two evaluation measures: *accuracy* and *scalability.*

A. Accuracy Evaluation

We evaluated the accuracy according to the instances reduction percentage. Table 4 and Figure 3 show that LSIS can improve the classification accuracy. We use 10-fold cross validation to evaluate the effectiveness of selected features using SVM classifier.

What is especially remarkable is that for all datasets, the classification accuracy is much better for a subset of 50 percent of instances.

TABLE IV: Classifier Accuracy according to the Percentage of Instance Reduction

% reduction	Classifier Accuracy	
	kdd-libsvm	real-sim
100	0,8606	0,8923
70	0,8620	0,9243
60	0,8638	0,9263
50	0,8644	0,9260

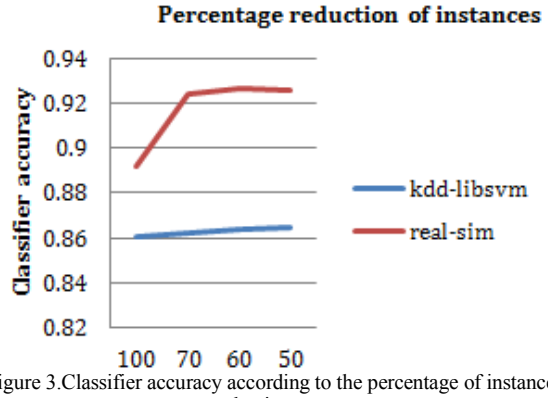

Figure 3.Classifier accuracy according to the percentage of instance reduction

B. Scalability Evaluation

After discussing the performance of our proposition in terms of classification accuracy, we evaluate the scalability of our algorithm by

proportionally increasing the number of nodes while keeping the same conditions to perform.

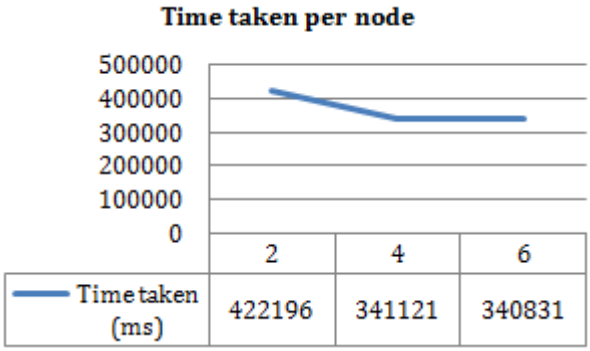

Figure 4. Scalability evaluation

The performance results demonstrate that our solution offers good computational efficiency. The time of selecting features decreases significantly when the number of nodes increases. That reveals logarithmic behavior as the number of cores increased.

VI. CONCLUSION

The large amount of data that is available in any research field poses new problems for data mining methods. In this paper, we presented a novel large-scale instance selection for getting manageable volume from them by selecting the relevant instances. We demonstrated that the proposed algorithm can scale well and efficiently process large datasets. We also found that our instance selection algorithm can improve the SVM classifier. In the future, we plan to compare our algorithm to the existing ones.

ACKNOWLEDGMENT

We have recipient of a Microsoft azure sponsored account. We would like to thanks Microsoft for this opportunity that have helped us to evaluating the performance of our algorithm. Without their Apache Spark cluster, we would never have made it to this point.

REFERENCES

[1] S. Garcia, J. Derrac, J. Cano, F. Herrera. **Prototype selection for nearest neighbor classification: Taxonomy and empirical study**. Pattern Anal. Mach. Intell. IEEE Trans., 34 (3) (2012), pp. 417–435 http://dx.doi.org/10.1109/TPAMI.2011.142

[2] Á. Arnaiz-González, J.-F. Díez-Pastor, J.J. Rodríguez, C. García-Osorio. **Instance selection of linear complexity for big data**. Knowl. Based Syst., 000 (2016), pp. 1–13.

[3] P. Hart. **The condensed nearest neighbor rule (corresp.)**. Inf. Theor. IEEE Trans., 14 (3) (1968), pp. 515–516

[4] E. Leyva, A. González, R. Pérez. **Three new instance selection methods based on local sets: a comparative study with several approaches from a bi-objective perspective.** Pattern Recognit., 48 (4) (2015), pp. 1523–1537. http://dx.doi.org/10.1016/j.patcog.2014.10.001

[5] Carbonera, Joel Luis, and Mara Abel. **A novel density-based approach for instance selection**. IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), 2016.

[6] Carbonera, Joel Luis, and Mara Abel. **A density-based approach for instance selection**. IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), 2015.

[7] E. Leyva, A. González, and R. Pérez, **Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective**, Pattern Recognition, vol. 48, no. 4, pp. 1523–1537, 2015.

[8] Srisawat, A., Phienthrakul, T., Kijsirikul, B. **SV-kNNC: An Algorithm for Improving the Efficiency of k-Nearest Neighbor.** PRICAI'06 proceedings of the 9th Pacific Rim international conference on Artificiel Intelligence. Guilin, China — August 07 - 11, 2006.

[9] Álvar Arnaiz-González, Alejandro González-Rogel, José-Francisco Díez-Pastor, Carlos López-Nozal. **MR-DIS: democratic instance selection for big data by MapReduce.** Artificial Intelligence (2017), pp. 1-9, doi:10.1007/s13748-017-0117-5.

[10] Junhai Zhai, Xizhao Wang, Xiaohe Pang. **Voting-based instance selection from large data sets with MapReduce and random weight networks**. Information Sciences. Volumes 367-368, Pages 1066-1077.

[11] Isaac Triguero, Daniel Peralta, Jaume Bacardit, Salvador García, Francisco Herrera, **MRPR: A MapReduce solution for prototype reduction in big data classification**.Neuro Computing, Volume 150, Part A, 20 February 2015, Pages 331–345.

[12] Chih-Chung Chang and Chih-Jen Lin**. LIBSVM Data: Classification (Binary Class).** https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

[13] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica. **Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing**. *Technical Report UCB/EECS-2011-82, EECS Department, University of California, Berkeley*, July 2011.

[14] Andronicus A. Akinyelu and Aderemi O. Adewumi.**Improved Instance Selection Methods for Support Vector Machine Speed Optimization.**Security and Communication Networks,Volume 2017 (2017), Article ID 6790975, 11 pages.

[15] Kyong-Ha Lee, Bongki Moon, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung.**Parallel Data Processing with MapReduce: A Survey.**SIGMOD Record, December 2011 (Vol. 40, No. 4).

[16] Priya Dahiya , Chaitra.B and Usha Kumari. **Survey on Big Data using Apache Hadoop and Spark.**International Journal of Computer Engineering In Research Trends, 4(6):pp:195-201,June - 2017.

ICCC 2017

Theory and Technology of Data Engineering