Scalable Forecasting Techniques Applied to Big Electricity Time Series

Antonio Galicia, José F. Torres, Francisco Martínez-Álvarez, and Alicia Troncoso(✉)

Division of Computer Science, Universidad Pablo de Olavide, 41013 Seville, Spain
{agalde,jftormal}@alu.upo.es, {fmaralv,ali}@upo.es

Abstract. This paper presents different scalable methods to predict time series of very long length such as time series with a high sampling frequency. The Apache Spark framework for distributed computing is proposed in order to achieve the scalability of the methods. Namely, the existing MLlib machine learning library from Spark has been used. Since MLlib does not support multivariate regression, the forecasting problem has been split into h forecasting subproblems, where h is the number of future values to predict. Then, representative forecasting methods of different nature have been chosen such as models based on trees, two ensembles techniques (gradient-boosted trees and random forests), and a linear regression as a reference method. Finally, the methodology has been tested on a real-world dataset from the Spanish electricity load data with a ten-minute frequency.

Keywords: Big data · Scalable · Electricity time series · Forecasting

1 Introduction

It is known that advances in technology have meant that the amount of data being generated and stored is increasing to the point that 90% of the data in the world have been generated in the last years. The need to process this huge amount of data has become essential for the evolution of the data mining tools giving rise to the term big data. On the other hand, an essential component in the nature of the big data is that they are commonly indexed over time, called here big time series, and its prediction in future time periods can be extremely important in diverse areas such as energy, traffic, pollution and so forth.

Nowadays, the main existing frameworks for processing big time series have been developed by over the top tech companies like Google or Yahoo. Google developed the MapReduce technology [5], which divides input data for processing in *blocks* and then integrates the output information of each block in a single solution. Later, Yahoo developed Hadoop technology [22], an open code implementation of the MapReduce paradigm, currently integrated with the Apache foundation. The limitations of MapReduce in the implementation of algorithms, which iterate

I. Rojas et al. (Eds.): IWANN 2017, Part II, LNCS 10306, pp. 165–175, 2017.
DOI: 10.1007/978-3-319-59147-6_15

over the data, have required the creation of new tools, such as Spark [9], developed by the University of Berkeley and also today in the Apache Foundation. Spark installed on a Hadoop distributed file system (HDFS) allows in-memory parallel data processing, achieving a much higher processing speed than Hadoop. Apache Spark is also an open source software project that allows the multi-pass computations, provides high-level operators, uses diverse languages (Java, Python, R) in addition to its own language called Scala, and finally, offers the machine learning library MLlib [8].

In this work, a collection of scalable algorithms are proposed in order to forecast big data time series. In particular, representative prediction methods of different nature have been chosen such as models based on trees, linear regression and two ensembles techniques (gradient-boosted trees and random forests). The algorithms have been developed in the framework Apache Spark under the Scala programming language by using the library MLlib. All the methods have been tested on a real-world big time series related to energy consumption.

The rest of the paper is structured as follows. Section 2 reviews of the existing literature related to the machine learning algorithms for big data. In Sect. 3 the proposed methodology to forecast big data time series is introduced. Section 4 presents the experimental results corresponding to the prediction of the energy consumption. Finally, Sect. 5 closes the paper giving some final conclusions.

2 Related Work

The prediction of future events has always fascinated humankind. Not in vain, many of these efforts can be seen in everyday activities, such as weather forecasting, the prediction of exchange rate fluctuations or of pollution.

The methods for time series forecasting can be roughly classified as follows: classical Box and Jenkins-based methods such as ARMA, ARIMA, ARCH or GARCH [1] and data mining techniques (the reader is referred to [12] for a taxonomy of these techniques applied to energy time series forecasting). However, the majority of the data mining techniques cannot be applied when big data have to be processed due to the high computational cost. Therefore, big data mining techniques [21,24] are being developed for distributed computing in order to solve typical tasks as clustering, classification or regression. A brief description of the main advances is made below.

Increased attention has been paid to big data clustering in recent years [11,15]. A survey on this topic can be found in [7]. Specifically, several approaches have been recently proposed to apply clustering to big data time series. Namely, in [6] the authors propose a new clustering algorithm based on a previous clustering of a sample of the input data. The dynamic time warping was tested to measure the similarity between big time series in [16]. In [23] a data processing based on MapReduce was used to obtain clusters. A distributed method for the initialization of the k-means is proposed in [3].

Regarding classification tasks, several MapReduce-based approaches in big data scenarios have been recently provided. A MapReduce-based framework

focused on several instance reduction methods is proposed in [20] to reduce the computational cost and storage requirements of the k Nearest Neighbors (kNN) classification algorithm. Also, several parallel implementations of the kNN algorithm based on Spark have been proposed in the literature [17,19]. Support vector machines (SVM) were recently adapted to the field of high performance computing giving rise to parallel SVMs [4].

In the regression field, there is still much research to be conducted, especially considering that very few works have been published. For instance, the ensemble techniques based on trees have been the most studied topic in the literature due to its easy adaptation to a distributed computing framework. Random forests have been applied to some particular problems showing a good performance for high-dimensional data [10]. On the other hand, regression trees have been built by parallel learning based on MapReduce on computer clusters in [14]. However, these methods based on a distributed computing have not used for big time series forecasting in to the best of authors' knowledge, and therefore, this work aims at filling this gap.

3 Methodology

This section describes the methodology proposed in order to forecast big data time series by using the MLlib library.

Given a time series recorded in the past up to the time t, $[x_1,...,x_t]$, the problem consists in predicting the h next values for the time series from a historical windows composed of w-values (h is known as the prediction horizon). This can be formulated as:

$$[x_{t+1}, x_{t+2}, \ldots, x_{t+h}] = f(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \tag{1}$$

where f is the model to be found by the forecasting method in the training phase.

Nevertheless, the existing regression techniques in MLlib do not support the multivariate regression, that is, the multi-step forecasting. Therefore, the first stage splits the problem into h forecasting subproblems as follows:

$$\begin{aligned} x_{t+1} &= f_1(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \\ x_{t+2} &= f_2(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \\ &\ldots \\ x_{t+h} &= f_h(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \end{aligned} \tag{2}$$

The existing possible relations between the h consecutive values $x_{t+1}, ..., x_{t+h}$ are missed with this formulation. However, if the prediction of previous values is used to predict the next values a greater error is obtained, as the errors are accumulated in the last time stamps of the prediction horizon. Additionally, to obtain h models $f_1, ..., f_h$ to predict h values has a greater computational cost than the building of a just model f to predict all the values.

The next stage consists in solving each forecasting subproblem in the Spark distributed computing framework by using the regression methods of the MLlib library. The main variable in Apache Spark is the Resilient Distributed Dataset (RDD), which is an immutable and partitioned collection of elements that can be operated in a distributed way. Thus, every RDD created is split in blocks of the same size approximately across the nodes that integrate the cluster, as it is shown in Fig. 1.

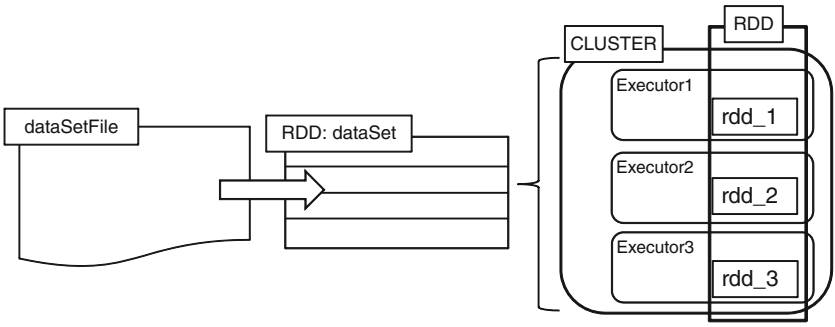

Fig. 1. A RDD variable in a spark cluster.

Once the dataset has been distributed, the MLlib algorithms firstly obtain a model from each worker node, and later, aggregate the predictions obtained for each model in a stage called reducer. It is important to highlight that RDD variables do not preserve the order, and therefore, all instances have to be indexed to deal with time series by using MLlib. An illustration of the methodology is presented in Fig. 2.

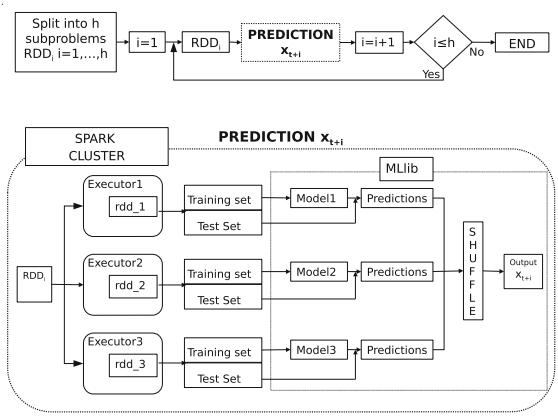

Fig. 2. Illustration of the proposed methodology.

Regression methods from MLlib have been selected according to cover different paradigms such as linear models, models based on trees and, finally, techniques ensembles.

The models based on trees have been mainly proposed because interpretable results are always desirable for the end-user. Furthermore, the ensemble techniques usually improve the results obtained by a single regressor in addition to obtain very good results for many real applications. Finally, a linear model has been selected as a state-of-the-art reference method. A brief description of the methods used for each paradigm is made below.

Within the models based on trees, a greedy algorithm [18] that performs a recursive binary partitioning of the feature space in order to build a decision tree has been used. The tree predicts the same value for all instances that reach the same leaf node. The root nodes are selected from a set of possible splits, but no from all attributes, by maximizing the information gain. In this approach, the possible split candidates are a quantile over the block of the data, which is being processed by a certain worker machine in the cluster. Moreover, once the splits are ordered, a maximum number of bins is allowed.

Two ensemble of decision trees have been considered: random forests [2] and the gradient-boosted trees (GBTs) [13]. Both algorithms learn ensembles of trees, but the training processes are very different. GBTs train one tree at a time, being the longer training than random forests, which can train multiple trees in parallel. Random forests improves the performance when the number of trees increases, however, GBTs can present overfitting if the number of trees grows too large.

Random forests is an ensemble of decision trees trained separately in the same way as detailed above for individual decision trees. The trees generated are different because of different training sets from a bootstrap subsampling and different random subsets of features to split on at each tree node are used. To make a prediction on a new instance, a random forest makes the average of the predictions from its set of decision trees.

GBTs iteratively train a sequence of decision trees. On each iteration, the algorithm uses the current ensemble to predict the label of each training instance and then compares the prediction with the true label by computing the mean square error. The training instances with poor predictions are re-labeled, and therefore, in the next iteration, the decision tree will help correct for previous mistakes.

Finally, a linear regression has been selected as linear model. The well-known stochastic gradient descent method has been used to minimize the mean square error for the training set in order to obtain the model.

4 Results

This section presents the results obtained from the application of the proposed methodology to electricity consumption big data time series to predict the 24 next values, that is, the forecast horizon set to $h = 24$ (4 h). Hence, Sect. 4.1

describes the used dataset. The experimental setup carried out is detailed in Sect. 4.2. Finally, the results are discussed in Sect. 4.3.

4.1 Datasets Description

The time series used is related to the electrical energy consumption, which ranges from January 1st 2007 at 00:00 am to June 21st 2016 at 23:40 am. The consumption is measured every ten minutes during this period. This makes a time series with a total length of 497832 measurements, which have been split into 298608 samples for the training set corresponding to the period from January 1st, 2007 at 00:00 am to September 8th 2012 at 10:30 am and 199080 samples for the test set corresponding to the period from September 8th 2012 at 10:40 am to June 21st 2016 at 11:40 pm.

4.2 Design of Experiments

The experimental setting of the algorithms is as follows:

1. The number of past values used to predict the 24 next values has been set to 144 (window $w = 144$), which represents all the values for a whole day.
2. In the linear regression, the stochastic gradient descent method requires an adequate number of iterations and rate of learning in order to guarantee the convergence of the optimization technique. In this work, values of $1.0E-10$ for the rate and 100 for the iterations have shown to be suitable.
3. The number of trees and the maximum depth are the main inputs for random forests and GBTs. Different depth levels have been tested for both ensembles, namely, four and eight. A number of five trees has been set for GBTs and values of 50, 75, 100, 125 and 150 trees for random forests.

The experimentation has been launched on a cluster, which is composed of three nodes: the master and two slaves nodes. Each node has two Intel Xeon E7-5820K processors at 3.3 GHz, 15 MB cache, 6 cores per processor and 16 GB of main memory working under Linux Ubuntu. The cluster works with Apache Spark 2.0.2 and Hadoop 2.6.

Finally, the well-known mean relative error (MRE) measure has been selected to assess the accuracy of the predictions. Its formula is:

$$MRE = \frac{1}{N} \sum_{i=1}^{N} \frac{|\widehat{x}_i - x_i|}{x_i} \tag{3}$$

where $\widehat{x}_i$ stands for the predicted values and x_i for the actual consumption values.

4.3 Electricity Consumption Big Data Time Series Forecasting

Table 1 summarizes the MRE obtained by all methods based on trees when predicting the test set. A study of how the number of trees has an influence on the error is made for the random forests ensemble. In addition, the depth of the trees used for all methods has been analyzed. It can be seen that a greater accuracy is provided when the depth of the trees increases due to trees more specific are obtained. By contrast, it seems that the number of trees to be used by the random forest has not a high impact over the error, and therefore, fifty trees was a sufficient number to obtain a good performance of the method.

Table 1. MRE for different depth levels and number of trees.

	Decision tree	Random forests					GBTs
Number of trees	1	50	75	100	125	150	5
Depth 4	5.1516	4.2823	4.2583	4.2415	4.2415	4.2427	4.3402
Depth 8	2.8783	2.2005	2.1853	2.1842	2.1810	2.1773	2.7190

Table 2 shows the MRE for the methods based on trees when a depth of 8 and a number of 50 trees for random forests has been used. Additionally, it shows the MRE obtained by means of a linear regression as baseline method to establish a benchmarking. All non linear methods based on trees achieved better errors than the linear regression, namely a difference of 5% approximately. Although the best results are obtained by the random forests ensemble technique, it can be concluded that the decision tree is the more adequate method in terms of accuracy and CPU time to predict big data time series.

Table 2. MRE for the test set and CPU time for training.

	MRE (%)	Time (seconds)
Linear regression	7.3395	553
Decision tree	2.8783	81
Random forests	2.2005	277
GBTs	2.7190	417

Figures 3 and 4 present the predicted values along with the actual values for the random forest algorithm for the two days from the test set leading to the largest and smallest errors, respectively. The worst prediction corresponds to an error of 9.12% associated to the period from December 24th 2013 at 10:50 am to December 25th 2013 at 10:40 am and the error of the best prediction is 0.67% corresponding to the day from September 20th 2012 at 10:40 am to September 21st 2012 at 10:30 am. It can be noted that the worst day is a special day, namely, Christmas Eve.

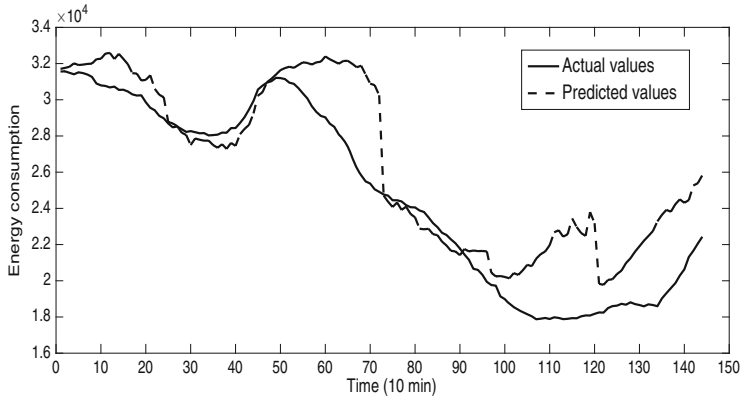

Fig. 3. The day corresponding to the worst prediction when using random forests.

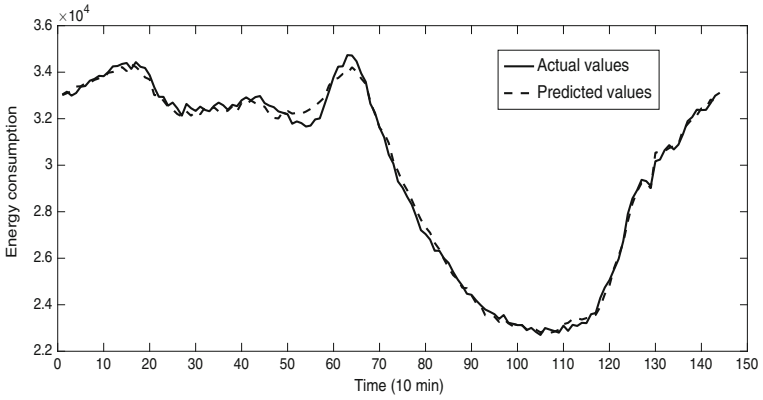

Fig. 4. The day corresponding to the best prediction when using random forests.

Finally, the training time versus the length of the time series for all algorithms proposed here are shown in the Fig. 5. The execution time has been obtained with time series of two, four, eight, sixteen and thirty and two times the length of the original time series. It is necessary to highlight the building of the dataset from the time series for each subproblem is not included in the training time as that is not made in a distributed way, but in an iterative way. From this figure, it can be observed that the most scalable method is the decision tree.

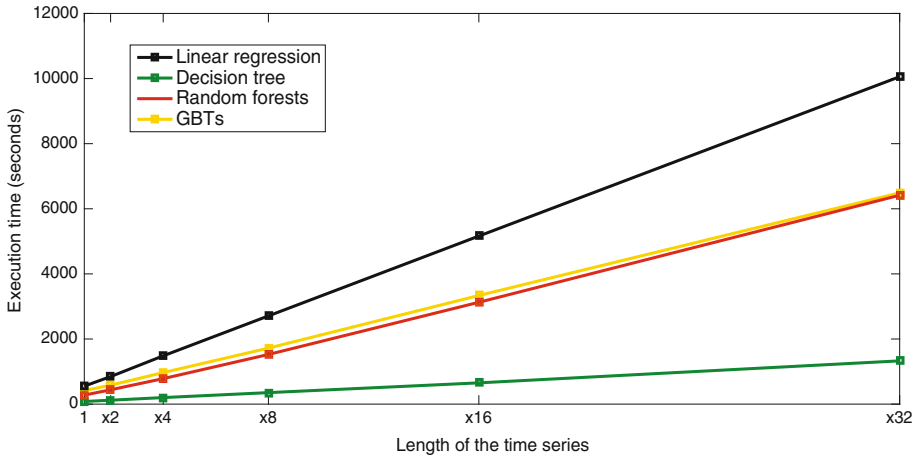

Fig. 5. Runtime and scalability for all algorithms.

5 Conclusions

In this work, a new formulation has been proposed for multi-step forecasting problems in order to be able to use the MLlib library from Apache Spark framework. The use of this library guarantees that the methods applied to predict the energy consumption for the next twenty four values are scalable, and therefore, they can be used for big data time series. A pool of linear and non linear methods have been selected, e.g., methods based on trees, ensemble techniques based on trees and a linear regression. Results for the Spanish electricity demand time series have been reported, showing the good performance of the methods proposed here and the grade of scalability for each of them.

Future work is directed towards solving the forecasting subproblems in a distributed way by using technology based on multithreads.

Acknowledgments. The authors would like to thank the Spanish Ministry of Economy and Competitiveness and Junta de Andalucía for the support under projects TIN2014-55894-C2-R and P12-TIC-1728, respectively.

References

1. Box, G., Jenkins, G.: Time Series Analysis: Forecasting and Control. Wiley, New York (2008)
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
3. Capó, M., Pérez, A., Lozano, J.A.: A Recursive k-means initialization algorithm for massive data. In: Proceedings of the Spanish Association for Artificial Intelligence, pp. 929–938 (2015)

4. Cavallaro, G., Riedel, M., Richerzhagen, M., Benediktsson, J.A.: On understanding big data impacts in remotely sensed image classification using support vector machine methods. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **8**, 4634–4646 (2015)
5. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
6. Ding, R., Wang, Q., Dan, Y., Fu, Q., Zhang, H., Zhang, D.: Yading: fast clustering of large-scale time series data. Proc. VLDB Endow. **8**(5), 473–484 (2015)
7. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Zomaya, A.Y., Khalil, I., Sebti, F., Bouras, A.: A survey of clustering algorithms for big data: taxonomy & empirical analysis. IEEE Trans. Emerg. Top. Comput. **5**, 267–279 (2014)
8. Machine Learning Library (MLlib) for Spark. On-line (2016). http://spark.apache.org/docs/latest/mllib-guide.html
9. Hamstra, M., Karau, H., Zaharia, M., Knwinski, A., Wendell, P., Spark, L.: Lightning-Fast Big Analytics. O' Really Media, USA (2015)
10. Li, L., Bagheri, S., Goote, H., Hassan, A., Hazard, G., Risk adjustment of patient expenditures: a big data analytics approach. In: Proceedings of the IEEE International Conference on Big Data, pp. 12–14 (2013)
11. Luna-Romera, J.M., Martínez-Ballesteros, M., García-Gutiérrez, J., Riquelme-Santos, J.C.: An approach to Silhouette and Dunn clustering indices applied to big data in spark. In: Luaces, O., Gámez, J.A., Barrenechea, E., Troncoso, A., Galar, M., Quintián, H., Corchado, E. (eds.) CAEPIA 2016. LNCS, vol. 9868, pp. 160–169. Springer, Cham (2016). doi:10.1007/978-3-319-44636-3_15
12. Martínez-Álvarez, F., Troncoso, A., Asencio-Cortés, G., Riquelme, J.C.: A survey on data mining techniques applied to electricity-related time series forecasting. Energies **8**(11), 13162–13193 (2015)
13. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent. In: Proceedings of the Neural Information Processing Systems Conference, NIPS, pp. 512–518 (1999)
14. Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J.: PLANET: massively parallel learning of tree ensembles with mapreduce. In: Proceedings of the Very Large Databases, pp. 1426–1437 (2009)
15. Perez-Chacon, R., Talavera-Llames, R.L., Martinez-Alvarez, F., Troncoso, A.: Finding electric energy consumption patterns in big time series data. In: Omatu, S. (ed.) Proceedings of the International Conference on Distributed Computing and Artificial Intelligence. Advances in Intelligent Systems and Computing, vol. 474. Springer, Cham (1991)
16. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Addressing big data time series: mining trillions of time series subsequences under dynamic time warping. ACM Trans. Knowl. Discov. Data **7**(3), 267–279 (2014)
17. Reyes-Ortiz, J.L., Oneto, L., Anguita, D.: Big data analytics in the cloud: spark on Hadoop vs MPI/OpenMP on Beowulf. Procedia Comput. Sci. **53**, 121–130 (2015)
18. Rokach, L., Maimon, O.: Top-down induction of decision trees classifiers - a survey. IEEE Trans. Syst. Man Cybern. Part C **35**(4), 476–487 (2005)
19. Talavera-Llames, R.L., Pérez-Chacón, R., Martínez-Ballesteros, M., Troncoso, A., Martínez-Álvarez, F.: A nearest neighbours-based algorithm for big time series data forecasting. In: Martínez-Álvarez, F., Troncoso, A., Quintián, H., Corchado, E. (eds.) HAIS 2016. LNCS, vol. 9648, pp. 174–185. Springer, Cham (2016). doi:10.1007/978-3-319-32034-2_15

20. Triguero, I., Peralta, D., Bacardit, J., García, S., Herrera, F.: MRPR: a mapreduce solution for prototype reduction in big data classification. Neurocomputing **150**, 331–345 (2015)
21. Tsai, C.-W., Lai, C.-F., Chao, H.-C., Vasilakos, A.: Big data analytics: a survey. J. Big Data **2**(1), 21 (2015)
22. White, T.: Hadoop, The Definitive Guide. O' Really Media, USA (2012)
23. Zhao, W., Ma, H., He, Q.: Parallel k-means clustering based on mapreduce. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5391, pp. 674–679. Springer, Heidelberg (2009). doi:10.1007/978-3-540-95885-7_24
24. Zhou, L., Pan, S., Wang, J., Vasilakos, A.V.: Machine learning on big data: opportunities and challenges. Neurocomputing **237**, 350–361 (2017)