An Accelerated MapReduce-Based K-prototypes for Big Data

Mohamed Aymen Ben HajKacem(✉), Chiheb-Eddine Ben N'cir, and Nadia Essoussi

LARODEC, Université de Tunis, Institut Supérieur de Gestion de Tunis, 41 Avenue de la Liberté, Cité Bouchoucha, 2000 Le Bardo, Tunisia
medaymen.hajkacem@gmail.com, {chiheb.benncir,nadia.essoussi}@isg.rnu.tn

Abstract. Big data are often characterized by a huge volume and a variety of attributes namely, numerical and categorical. To address this issue, this paper proposes an accelerated MapReduce-based k-prototypes method. The proposed method is based on pruning strategy to accelerate the clustering process by reducing the unnecessary distance computations between cluster centers and data points. Experiments performed on huge synthetic and real data sets show that the proposed method is scalable and improves the efficiency of the existing MapReduce-based k-prototypes method.

Keywords: K-prototypes · MapReduce · Big data · Mixed data

1 Introduction

Given the exponential growth and availability of data collected from different resources, analyzing these data has become an important challenge referred to as Big data analysis. Big data analysis usually refers to three mains characteristics also called the three Vs [7] which are respectively *Volume*, *Variety* and *Velocity*. Volume refers to the large scale data, Variety indicates the different data types and formats and Velocity refers to the streaming data [6]. One of the most important challenges in Big data analysis is how to explore the large amount of mixed data using machine learning techniques. Clustering is one of the machine learning techniques, which has been used to organize data into groups of similar data points called also clusters. Examples of clustering methods categories are hierarchical methods, density-based methods, grid-based methods, model-based methods and partitional methods [13]. However, traditional clustering methods are not suitable for processing large scale of mixed data. For example, k-prototypes clustering [18] which is one of the most popular method to cluster mixed data, it does not scale with huge volume of data [20].

To deal with this issue, Ben HajKacem et al. [3] have proposed a parallelization of k-prototypes method through MapReduce model. Although this method offers for users an efficient analysis of a huge amount of mixed data, it requires computing all distances between each of the cluster centers and the data points.

P. Milazzo et al. (Eds.): STAF 2016, LNCS 9946, pp. 13–25, 2016.
DOI: 10.1007/978-3-319-50230-4_2

However, many of these distance computations are unnecessary, because data points usually stay in the same clusters after first few iterations. Therefore, we propose in this paper an **A**ccelerated **M**ap**R**educe-based **k-p**rototypes method called AMR k-prototypes. The proposed method is based on pruning strategy to accelerate the clustering process by reducing the unnecessary distance computations between cluster centers and data points. The experiments show that the proposed method is scalable and outperforms the efficiency of the existing MapReduce-based k-prototypes method [3].

The organization of this paper is as follows: Sect. 2 presents related works in the area of Big data clustering. Then, Sect. 3 describes the proposed AMR k-prototypes method while Sect. 4 presents experiments that we have performed to evaluate the efficiency of the proposed method. Finally, Sect. 5 presents conclusion and future work.

2 Related Works

Big data clustering has recently received a lot of attentions to build parallel clustering methods. In this context, several parallel clustering methods have been designed in the literature [2,4,9,14,16,17,19,23]. Most of these methods use the MapReduce [5], which is a programming model for processing large scale data by exploiting the parallelism among a cluster of machines. For example, Zaho et al. [23] have proposed a parallelization of k-means method using MapReduce model. Kim et al. [14] have introduced an implementation of DBSCAN method through MapReduce model. Recently, a parallel implementation of fuzzy c-means clustering algorithm using MapReduce model is presented in [17]. Indeed, Big data are often characterized by the variety of attributes, including numerical and categorical. Nevertheless, the existing parallel methods can not handle different types of data and are limited to only numerical attributes.

To deal with mixed data, a pre-processing step is usually required to transform data into a single type since most of proposed clustering methods deal with only numerical or categorical attributes. However, transformation strategies is often time consuming and produce information loss, leading to undesired clustering results [1]. Thus, several clustering methods for mixed data have been proposed in the litterateur [1,8,11,15]. For instance, Huang [11] have proposed k-prototypes method which combines k-means [18] and k-modes [12] methods for clustering mixed data. Li and Biswas [15] have proposed Similarity-Based Agglomerative Clustering called SBAC, which is a hierarchical agglomerative algorithm for mixed data. Among the later discussed methods, k-prototypes remains the most popular method to cluster mixed data, because of its simplicity and linear computational complexity [8].

In the following, we present an accelerated MapReduce-based k-prototypes method to deal with large scale of mixed data.

3 An Accelerated MapReduce-Based K-prototypes for Big Data

We propose in this section an accelerated MapReduce-based k-prototypes method. Before presenting the proposed method, we first introduce the k-prototypes method [11], then the MapReduce model [5].

3.1 K-prototypes Method

Given a data set $X=\{x_1 \dots x_n\}$ containing n data points, described by m_r numerical attributes and m_t categorical attributes, the aim of k-prototypes [11] is to find k clusters where the following objective function is minimized:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} p_{ij} d(x_i, c_j), \tag{1}$$

where $p_{il} \in \{0, 1\}$ is a binary variable indicating the membership of data point x_i in cluster c_j, c_j is the center of the cluster c_j and $d(x_i, c_j)$ is the dissimilarity measure which is defined as follows:

$$d(x_i, c_j) = \sum_{r=1}^{m_r} \sqrt{(x_{ir} - c_{jr})^2} + \gamma_j \sum_{t=1}^{m_t} \delta(x_{it}, c_{jt}), \tag{2}$$

where x_{ir} represents the value of numeric attribute r and x_{it} represents the value of categorical attribute t for data point x_i, c_{jr} is the mean of numeric attribute r and c_{jt} is the most common value (mode) for categorical attributes t for cluster c_j. For categorical attributes, δ(p,q)=0 when $p = q$ and $\delta(p, q) = 1$ when $p \neq q$. γ_j is a weight for categorical attributes to cluster c_j. The optimization of the objective function J is performed using an alternating iterative process by looking for the optimal cluster centers. These two steps are alternated iteratively until convergence. The main algorithm of k-prototypes method is described in Algorithm 1.1.

Algorithm 1.1. Main algorithm of k-prototypes method

Input: X=$\{x_1 \dots x_n\}$, k
Output: Centers=$\{c_1 \dots c_k\}$
begin
  Choose k cluster centers randomly from X
  repeat
    Compute distance between data points and clusters using Eq. 2
    Update the cluster centers (Save the previous cluster centers as $Centers^{\wedge}$ to analyze the convergence)
  until $Centers^{\wedge} = Centers$;
end

3.2 MapReduce Model

MapReduce [5] is a parallel programming model designed to process large scale data sets among cluster nodes. The MapReduce model works as follows. The input and output of the computation is a set of $<key/value>$ pairs. The algorithm to be parallelized needs to be expressed by map and reduce functions. The map function is applied in parallel to each input $<key/value>$ pair and returns a set of intermediate $<key'/value'>$ pairs. Then, shuffle phase groups all intermediate values associated with the same intermediate key and passes them to the reduce function. The reduce function takes the intermediate key and set of values for this key. These values are merged together to produce a set of values. Figure 1 illustrates the flowchart of MapReduce model. The inputs and outputs are stored in an associated distributed file system that is accessible from any machine of the cluster nodes. The implementation of the MapReduce model is available in Hadoop[1]. Hadoop provides a distributed file system named Hadoop Distributed File System, (HDFS) that stores data on the nodes.

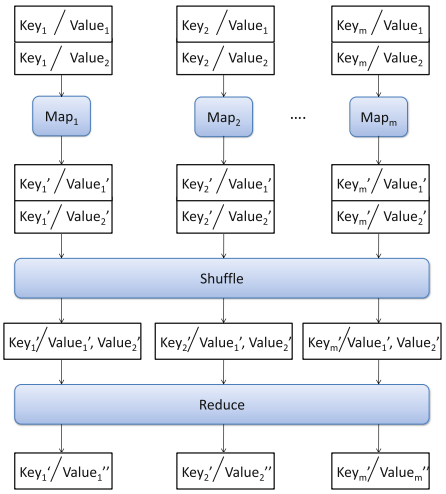

Fig. 1. MapReduce model flowchart

3.3 An Accelerated MapReduce-Based K-prototypes Method for Big Data (AMR K-prototypes)

To offer for users the possibility to build grouping from large scale of mixed type data, we propose the accelerated MapReduce-based k-prototypes method. The proposed method mainly consists of two functions: *map function* which performs the assignment of each data point to the nearest cluster, and *reduce function* which is devoted to update the new cluster centers. Then, we iterate

[1] http://hadoop.apache.org/.

calling the two functions several times until convergence. It is important to note that the initial cluster centers are chosen randomly.

3.3.1 Map Function:

During this function, we assign each data point to the nearest cluster by computing distance of Eq. 2 between data points and cluster centers. To reduce the number of distance computations, we propose a pruning strategy using triangle inequality. More precisely, the triangle inequality is used to prove that if cluster center c_1 is close to data point x, and some other cluster center c_2 is far away from another cluster center c_1, then c_1 must be closer than c_2 to x. The following theorem shows how to use the triangle inequality to reduce the distance computations and more details can be found in [10].

Theorem 1. *Let x a data point and let c_1 and c_2 cluster centers. If we know that d(c_1,c_2) $\geq$ 2 * d(x,c_1) $\Rightarrow$ d(x,c_1) $\leq$ d(x,c_2) without having to calculate d(x,c_2).*

Proof. According to triangle inequality, we know that d(c_1,c_2) $\leq$ d(x,c_1) + d(x,c_1) $\Rightarrow$ d(c_1,c_2) $-$ d(x,c_1) $\leq$ d(x,c_2). Consider the left-hand side d(c_1,c_2) $-$ d(x,c_1) $\geq$ 2 * d(x,c_1) $-$ d(x,c_1) = d(x,c_1) $\Rightarrow$ d(x,c_1) $\leq$ d(x,c_2).

After assigning each data point to nearest cluster, we update a local information about clusters. To do so, we first update the values of the numerical attributes of data points. Second, we update the frequencies of different values of categorical attributes of data points. Third, we update the number of data points assigned to clusters. Keeping these information is inexpensive and avoids the calculation over all data points for each iteration. Each time a data point changes cluster membership, the local information are updated. After few iterations, most data points remain in the same cluster for other iterations. Then, the map function outputs the local information about clusters to the reduce function.

Let $X=\{X^1 \dots X^m\}$ the input data set where X^g the portion of input data set associated to map function g. Let $Centers=\{c_1 \dots c_k\}$ the set of cluster centers. Let $SUM^g=\{sum_1^g \dots sum_k^g\}$ the set of sum of data values of numerical attributes relative to different clusters. Let $FREQ^g=\{freq_1^g \dots freq_k^g\}$ the set of frequencies of data values of categorical attributes relative to different clusters. Let $NUMBER^g=\{number_1^g \dots number_k^g\}$ the set of number of data points relative to different clusters. Let *new* (resp. *old*) the cluster index of data point x_i in the current (resp. previous) iteration. Let $Cluster-Cluster$ a matrix which records the distances between each pair of cluster centers where $Cluster-Cluster_{ij}$ returns the distance between c_i and c_j. The main steps of map function is described in Algorithm 1.2.

3.3.2 Reduce Function

During this function, we merge the local information which are produced from all map functions in order to calculate the new cluster centers. So, for each cluster, we first sum the numeric values of data points. Second, we compute the total

Algorithm 1.2. Map function

Input: $< key : g/value : X^g >, Centers$

Output: $< key^{'} : 1/value^{'} : SUM^g, FREQ^g, NUMBER^g >$

begin

$SUM^g \leftarrow \emptyset \quad FREQ^g \leftarrow \emptyset \quad NUMBER^g \leftarrow \emptyset$

for $i \leftarrow 1 \ldots k$ **do**

  for $j \leftarrow 1 \ldots k$ **do**

    $Cluster - Cluster_{ij} \leftarrow \mathrm{d}(c_i, c_j)$

foreach $x_i \in X^g$ **do**

  for $j \leftarrow 1 \ldots k$ **do**

    if $p_{ij} = 1$ **then**

      $old \leftarrow j$

  minDistance$\leftarrow \mathrm{d}(s_i, c_{old})$

  for $j \leftarrow 1 \ldots k$ **do**

    if $minDistance \leq 2 * Cluster - Cluster_{jold}$ **then**

      j $\leftarrow$ j+1

    else

      % *Distance computation*

      Distance$\leftarrow \mathrm{d}(x_i, c_j)$

      if $distance < minDistance$ **then**

        minDistance $\leftarrow$ distance

        $new \leftarrow$ j

  if $new \neq old$ **then**

    $sum^g_{new} \leftarrow sum^g_{new} + x_i$

    $sum^g_{old} \leftarrow sum^g_{old} - x_i$

    $freq^g_{new} \leftarrow freq^g_{new} + 1$

    $freq^g_{old} \leftarrow freq^g_{old} - 1$

    $number^g_{new} \leftarrow number^g_{new} + 1$

    $number^g_{old} \leftarrow number^g_{old} - 1$

return $< 1/SUM^g, FREQ^g, NUMBER^g >$

end

frequencies of different values of categorical attributes relative to the data points. Third, we sum the number of total data points. Given the above information, we can compute both the mean and mode value of the new cluster centers. Once the new cluster centers are computed, the proposed method moves to the next iteration until convergence. The convergence is achieved when cluster centers become stable for two consecutive iterations. We notate that the new cluster centers are stored in HDFS to be ready for next iteration.

Let $NewCenters$=$\{newc_1 \ldots newc_k\}$ the set of new cluster centers. Let Highest-Freq($freq_j$) a function which returns the mode value of cluster j from $freq_j$. The main steps of reduce function is described in Algorithm 1.3.

Algorithm 1.3. Reduce function

Input: $< key : 1/value : SUM^1, FREQ^1, NUMBER^1, \ldots, SUM^m, FREQ^m, NUMBER^m >$

Output: $< key^{'} : 1/value^{'} : NewCenters >$

begin

$NewCenters \leftarrow \emptyset$

for $j \leftarrow 1 \ldots k$ **do**

for $g \leftarrow 1 \ldots m$ **do**

$sum_j \leftarrow sum_j + sum_j^g$

$freq_j \leftarrow freq_j + freq_j^g$

$number_j \leftarrow number_j + number_j^g$

for $j \leftarrow 1 \ldots k$ **do**

Calculation of mean value

$newc_j \leftarrow sum_j / number_j$

Calculation of mode value

$newc_j \leftarrow Highest - Freq(freq_j)$

return $< 1/NewCenters >$

end

4 Experiments and Results

In this section, we describe the experiments which are performed to evaluate the efficiency of the proposed AMR k-prototypes method. First, the execution environment, and the information of the data sets used are given. Then, the evaluation measures are presented, and the experimental results are provided and discussed.

4.1 Environment and Data Sets

The experiments are performed on Hadoop cluster running the latest stable version of Hadoop 2.7.1. The Hadoop cluster consists of 4 machines. Each machine has two Pentium(R) Core i5 (2.70 GHz) CPU E5400 and 1 GB of memory. The operating system of each machine is Ubuntu 14.10 server 64 bit. We conducted the experiments on the following data sets:

- Synthetic data set: four series of mixed data sets generated using the data generator developed in[2]. The data sets range from 1 million to 4 million data points. Each data point is described using 3 numeric and 3 categorical attributes. In order to simplify the names of the synthetic data sets, we used names with specific pattern based on the data size. For example: the Sy1M data set consists of 1 million data points.
- KDD Cup data set (KDD): This is a real data set which consists of data about TCP connections simulated in a military network environment. Each connection is described using 7 numeric and 3 categorical attributes. The clustering

[2] https://projets.pasteur.fr/projects/rap-r/wiki/SyntheticDataGeneration.

process for this data set detects type of attacks among all the connections. KDD data set was obtained from UCI machine learning repository[3].
– Cover Type data set (Cover): This is a real data set which represents cover type for 30×30 meter cells from US Fores. Each measurement is described using 5 numeric and 3 categorical attributes. The clustering process for this data set identifies types of trees. Cover data set was obtained from UCI machine learning repository[4]. Statistics of these data sets are summarized in Table 1.

Table 1. Summary of the data sets

Data set	Number of data points	Number of attributes	Domain
Sy1M	1.000.000	6 (3 Numeric, 3 Categorical)	Synthetic
Sy2M	2.000.000	6 (3 Numeric, 3 Categorical)	Synthetic
Sy3M	3.000.000	6 (3 Numeric, 3 Categorical)	Synthetic
Sy4M	4.000.000	6 (3 Numeric, 3 Categorical)	Synthetic
KDD	4.898.431	10 (7 Numeric, 3 Categorical)	Detection intrusion
Cover	581.012	8 (5 Numeric, 3 Categorical)	Agriculture

4.2 Evaluations Measures

In order to evaluate the quality of the obtained results, we use Sum Squared Error (SSE) [21] which is defined as follows.

– The Sum Squared Error [21] is one of the most common partitional clustering criteria and its general objective is to obtain a partition which minimizes the squared error. This criterion is defined as follows:

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{k} d(c_j, x_i). \tag{3}$$

We used in our experiments the Speedup and Scaleup [22] measures to evaluate the performance of AMR k-prototypes method, which are defined as follows.

– The Speedup [22] is measured by fixing the data set size while increasing the number of machines to evaluate the ability of parallel algorithm to scale with increasing the number of machines of the Hadoop cluster, which is calculated as follows:

$$Speedup = \frac{T_1}{T_h}, \tag{4}$$

where T_1 the running time of processing data on 1 machine and T_h the running time of processing data on h machines in the Hadoop cluster.

[3] https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data.
[4] http://archive.ics.uci.edu/ml/datasets/Covertype.

– The Scaleup [22] is a measure of speedup that increases with increasing data set sizes to evaluate the ability of the parallel algorithm for utilizing the Hadoop cluster effectively, which is calculated as follows:

$$Scaleup = \frac{T_{n_1}}{T_{h*n_h}}, \tag{5}$$

where T_{n_1} the running time of processing data with size of n on 1 machine and T_{h*n_h} the running time of processing data with size of $h*n$ on h machines of the Hadoop cluster.

4.3 Results

We first evaluate the performance of the pruning strategy to reduce the unnecessary distances computations. Tables 2 and 3 report the number of distance computations performed by AMR k-prototypes compared to existing MapReduce-based k-prototypes (MR k-prototypes) method [3] for synthetic and real data sets respectively using ten runs. A different initialization of cluster centers have been used over the ten runs, whereas within each run the same initialization of cluster centers has been used for the different methods. The number of iterations is fixed as 10 for each run. From Tables 2 and 3, we can observe that the proposed method can reduce a lot of distance computations over MR k-prototypes method on both synthetic and real data sets. More importantly, this reduction becomes more significant with the increase of k. For example, the number of distance computations is reduced by 46.12% when $k = 50$ and by 78.09% when $k = 100$ for Sy4M data set.

Table 4 presents results obtained with AMR k-prototypes versus MR k-prototypes in terms of SSE values for real data sets. From Table 4, we can observe that the proposed method produces the same SSE values compared to MR-KP method. Therefore, we can conclude that AMR-KP avoids unnecessary distance

Table 2. Comparison of the number of distance computations for the synthetic data sets (averaged over 10 runs)

Data set	Number of distance computations ($*10^8$)	
	MR k-prototypes	AMR k-prototypes
Sy1M (K = 50)	5.0000 (± 0.01)	4.6553 (± 0.17)
Sy2M (K = 50)	10.0000 (± 0.03)	9.3087 (± 0.28)
Sy3M (K = 50)	15.0000 (± 0.01)	10.3966 (± 0.22)
Sy4M (K = 50)	20.0000 (± 0.01)	10.8620 (± 0.18)
Sy1M (K = 100)	10.0000 (± 0.02)	2.2671 (± 0.15)
Sy2M (K = 100)	20.0000 (± 0.01)	4.4502 (± 0.31)
Sy3M (K = 100)	30.0000 (± 0.01)	6.8056 (± 0.25)
Sy4M (K = 100)	40.0000 (± 0.03)	9.0711 (± 0.33)

Table 3. Comparison of the number of distance computations for the real data sets (averaged over 10 runs)

Data set	Number of distance computations ($*10^8$)	
	MR k-prototypes	AMR k-prototypes
KDD (K = 50)	24.4921 (± 0.56)	3.8136 (± 0.26)
Cover (K = 50)	19.7198 (± 0.17)	2.1147 (± 0.54)
KDD (K = 100)	48.9843 (± 0.28)	6.2263 (± 0.44)
Cover (K = 100)	38.2515 (± 0.58)	1.6948 (± 0.23)

Table 4. Comparison of the SSE values for the real data set (averaged over 10 runs)

Data set	SSE ($*10^8$)	
	MR k-prototypes	AMR k-prototypes
KDD (K = 50)	8.8131 (± 0.17)	8.8131 (± 0.17)
Cover (K = 50)	6.5124 (± 0.33)	6.5124 (± 0.33)
KDD (K = 100)	7.6916 (± 0.25)	7.6916 (± 0.25)
Cover (K = 100)	5.2678 (± 0.19)	5.2678 (± 0.19)

computations while still always producing exactly the same quality result as MR-KP method.

Then, we evaluate the speedup of the proposed method when the data set grows. Figure 2 shows the speedup results on the synthetic data sets. As the size of the data set increases, the speedup of AMR k-prototypes becomes approximately linear, especially in the case of Sy3M and Sy4M data sets. In addition, Fig. 2 shows that when the data size is 1 million, the performance of 4 machines of the Hadoop cluster is not significantly improved compared to that of 2 machines. The reason is that the time of processing 1 million data points is not very bigger than the communication time among the machines and time occupied by fault tolerance. Therefore, we can conclude that the larger the data set, the better the speedup.

To study the scalability of the proposed method, we have evaluated scaleup measures when we increase the size of the data set in direct proportion to the number of machines of the Hadoop cluster. The Sy1M, Sy2M, Sy3M and Sy4M data sets are processed on 1, 2, 3, 4 machines respectively. Figure 3 illustrates the scaleup results on the synthetic data sets. The scaleup has almost a constant ratio and ranges between 1 and 1.06. For example, the scaleup for Sy1M is 1 while for Sy4M it is 1.06, which is a very small difference. Therefore, we can conclude that the proposed method is scalable.

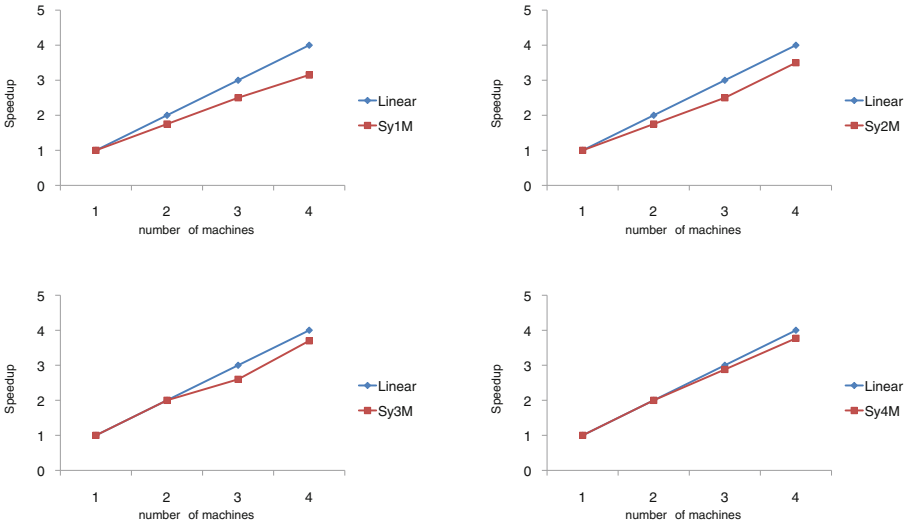

Fig. 2. Speedup results

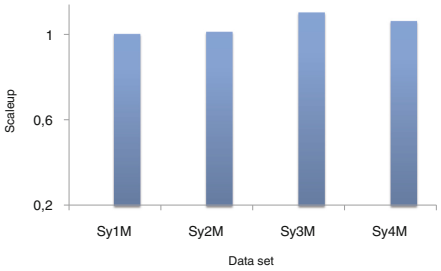

Fig. 3. Scaleup results

5 Conclusion

In this paper, we have proposed an accelerated MapReduce-based k-prototypes method to deal with large scale of mixed data. The proposed method is based on pruning strategy to reduce the unnecessary distance computations. The experiment results show that our method is scalable and can improves the efficiency of existing MapRedced-based k-prototypes method without decreasing the quality. A proper initialization of k-prototypes method is crucial for obtaining a good final solution. Thus, we plan to propose an efficient initialization of k-prototypes using MapReduce model in the future work.

References

1. Ahmad, A., Dey, L.: A k-mean clustering algorithm for mixed numeric and categorical data. Data Knowl. Eng. **63**(2), 503–527 (2007)
2. Bahmani, B., Moseley, B., Vattani, A., Kumar, R., Vassilvitskii, S.: Scalable k-means++. Proc. VLDB Endowment **5**(7), 622–633 (2012)
3. Ben Haj Kacem, M.A., Ben N'cir, C.E., Essoussi, N.: MapReduce-based k-prototypes clustering method for big data. In: Proceedings of Data Science and Advanced Analytics, pp. 1–7(2015)
4. Cui, X., Zhu, P., Yang, X., Li, K., Ji, C.: Optimized big data k-means clustering using mapReduce. J. Supercomput. **70**(3), 1249–1259 (2014)
5. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
6. Gandomi, A., Haider, M.: Beyond the hype: big data concepts, methods, and analytics. Int. J. Inf. Manag. **35**(2), 137–144 (2015)
7. Gorodetsky, V.: Opportunities, challenges and solutions. In: Information and Communication Technologies in Education, Research, and Industrial Applications, pp. 3–22
8. Ji, J., Bai, T., Zhou, C., Ma, C., Wang, Z.: An improved k-prototypes clustering algorithm for mixed numeric and categorical data. Neurocomputing **120**, 590–596 (2013)
9. Hadian, A., Shahrivari, S.: High performance parallel k-means clustering for disk-resident datasets on multi-core CPUs. J. Supercomput. **69**(2), 845–863 (2014)
10. Hamerly, G., Drake, J. Accelerating Lloyd's algorithm for k-means clustering. In: Partitional Clustering Algorithms, pp. 41–78 (2015)
11. Huang, Z.: Clustering large data sets with mixed numeric and categorical values. In Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 21–34(1997)
12. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Disc. **2**(3), 283–304 (1998)
13. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. (CSUR) **31**(3), 264–323 (1999)
14. Kim, Y., Shim, K., Kim, M.S., Lee, J.S.: DBCURE-MR: an efficient density-based clustering algorithm for large data using mapReduce. Inf. Syst. **42**, 15–35 (2014)
15. Li, C., Biswas, G.: Unsupervised learning with mixed numeric and nominal data. Knowl. Data Eng. **14**(4), 673–690 (2002)
16. Li, Q., Wang, P., Wang, W., Hu, H., Li, Z., Li, J.: An efficient k-means clustering algorithm on mapReduce. In: Proceedings of Database Systems for Advanced Applications, pp. 357–371 (2014)
17. Ludwig, S.A.: MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability. Int. J. Mach. Learn. Cybern. **6**(6), 923–934 (2015)
18. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 14, no. 1, pp. 281–297 (1967)
19. Shahrivari, S., Jalili, S.: Single-pass and linear-time k-means clustering based on mapReduce. Inf. Syst. **60**, 1–12 (2016)
20. Vattani, A.: K-means requires exponentially many iterations even in the plane. Discrete Comput. Geom. **45**(4), 596–616 (2011)
21. Xu, R., Wunsch, D.C.: Clustering algorithms in biomedical research: a review. Biomed. Eng. IEEE Rev. **3**, 120–154 (2010)

22. Xu, X., Jäger, J., Kriegel, H.P.: A fast parallel clustering algorithm for large spatial databases. In: High Performance Data Mining, pp. 263–290 (2002)
23. Zhao, W., Ma, H., He, Q. Parallel k-means clustering based on mapReduce. In: Proceedings of Cloud Computing, pp. 674–679 (2009)