An Approach to Silhouette and Dunn Clustering Indices Applied to Big Data in Spark

Conference Paper · September 2016

DOI: 10.1007/978-3-319-44636-3_15

José María Luna-Romera
Universidad de Sevilla
19 PUBLICATIONS **175** CITATIONS
SEE PROFILE

María Martínez Ballesteros
Universidad de Sevilla
31 PUBLICATIONS **381** CITATIONS
SEE PROFILE

Jorge García-Gutiérrez
Universidad de Sevilla
48 PUBLICATIONS **610** CITATIONS
SEE PROFILE

José C Riquelme
Universidad de Sevilla
278 PUBLICATIONS **3,910** CITATIONS
SEE PROFILE

An Approach to Silhouette and Dunn Clustering Indices Applied to Big Data in Spark

José María Luna-Romera, María del Mar Martínez-Ballesteros, Jorge García-Gutiérrez, and José C. Riquelme-Santos

Department of Computer Science, Universidad de Sevilla
{jmluna,mariamartinez,jorgarcia,riquelme}@us.es,

Abstract K-Means and Bisecting K-Means clustering algorithms need the optimal number into which the dataset may be divided. Spark implementations of these algorithms include a method that is used to calculate this number. Unfortunately, this measurement presents a lack of precision because it only takes into account a sum of intra-cluster distances misleading the results. Moreover, this measurement has not been well-contrasted in previous researches about clustering indices. Therefore, we introduce a new Spark implementation of Silhouette and Dunn indices. These clustering indices have been tested in previous works. The results obtained show the potential of Silhouette and Dunn to deal with Big Data.

Keywords: Silhouette, Dunn, Clustering index, Big Data, Spark

1 Introduction

Nowadays every device that surrounds us generates data. 90% of the world data, around 10,000 exabytes, has been generated in the last three years and there exists an estimation to fourfold that amount of data by 2020 [10]. Big Data is mainly defined as a massive, heterogeneous and often-unstructured digital content that is difficult to process by using traditional tools and techniques [2]. From this scenario, old methodologies cannot be applied [9,16]. Apache Spark is an open source cluster computing framework in charged of working with Big Data [12] and it is based on Resilient Distributed Datasets, a type of structured data especially designed for parallel computing. This structure allows us to cache results in memory and reuse them to process huge amounts of data. Spark also includes a scalable machine learning library (MLlib) [14]. MLlib contains a set of common learning algorithms and utilities for classification, regression, clustering or filtering. In this paper, we are going to focus on clustering techniques, specifically on clustering indices.

K-Means and Bisecting K-Means clustering algorithms are included in MLlib. These clustering algorithms present an inconvenient: the number of clusters into which the dataset is going to be divided has to be known in advance. If the number of clusters is not known, different combinations may be tried until a good solution is reached. To solve this problem, there exists indices that help us define into how many clusters the dataset can be grouped.

K-Means and Bisecting K-Means in MLlib include a measure that calculates the sum of squared distances between the centroids (WSSSE) [13]. Unfortunately, this measure neither calculates the cluster consistency nor the well-separated distance between clusters. In addition, the results of WSSSE may not be correct and they may mislead the clustering. Therefore, we introduce a new Spark implementation of Silhouette and Dunn indices.

First, Silhouette is an index that refers to a method of consistency interpretation within clusters of data. The measure of Silhouette sets how the example is fixed to its cluster compared to others [8,11,15]. Second, Dunn is a validity index whose aim is to identify clusters with a high inter-cluster distance and low intra-cluster distance [6,7]. These indices have been well-contrasted in previous work [17,4,1]. However, they have not been developed for Big Data to the best of our knowledge.

Hence, we propose in this paper a novel Spark implementation of Silhouette and Dunn indices as a solution for the calculation of the optimal number.

This paper is structured as follows. Section 2 describes the research methodology used in this work. Section 3 presents the detailed results of the experiments. Finally, section 4 presents the conclusion of our experiments as well as future directions to follow in upcoming works.

2 Methods

In this section, we present the outlines of a new implementation of Dunn and Silhouette clustering indices based on Spark. We provide information about our Silhouette and Dunn indices (Section 2.1), the clustering algorithms that we have used for our experiments (Section 2.2) and the datasets that were used (Section 2.3). We also specify the hardware and software resources that have been used for our experiments (Section 2.4).

2.1 BD-Silhouette and BD-Dunn indices

Our work is an approach of Silhouette and Dunn indices implemented in Spark. We will denote our indices as BD-Silhouette and BD-Dunn respectively. Our indices are approaches due to the limitations that Spark offers for applying quadratic complexity algorithms.

Let Ω be the space of the objects with $x \in \Omega$ with a given distance d.

Then $\{A_k\}_{k=1..N}$ is a set of clusters so that $\bigcup_k A_k = \Omega$, and $A_i \cap A_j = \emptyset \quad \forall i \neq j$.

C_k is the centroid of A_k, and C_o the centroid of Ω.

We suppose $X_i \in A_k$, then the distance from the object x_i to the cluster A_k is defined.

$$a_k(x_i) = \frac{1}{|A_k| - 1} \sum_{\substack{x_j \in A_k \\ x_j \neq x_i}} d(x_i, x_j) \tag{1}$$

The distance from x_i to the nearest cluster is defined $x_i \in A_k$

$$b_k(x_i) = min_{j=i..N}^{j \neq k}\{a_j(x_i)\} \tag{2}$$

interMean is defined as the average of the distances from centroids to the global center. *intraMean* is the average of the distances between each point to its centroid. With this *intracluster* calculation we avoid quadratic complexity, and it sets it as lineal order.

$$interMean = \frac{1}{N}\sum_{k=1}^{N} d(c_k, C_o) \tag{3}$$

$$intraMean = \frac{1}{|\Omega|}\sum_{x\in\Omega} d(x_i, C_k)\ with\ x_i\ in\ A_k \tag{4}$$

Formula 5 represents the BD-Silhouette that has been defined as the ratio between the difference of the interclusterMean and the intraclusterMean, and the maximum of them.

$$BD\text{–}Silhouette = \frac{interMean - intraMean}{max\{interMean, intraMean\} with\ x_i \in A_x} \tag{5}$$

BD-Silhouette returns a value from 0 to 1, depending on the consistence of the cluster and the separation between them. The higher the cluster number is, the lower intraclusterMean is because dataset's points tend to be more compacted which make BD-Silhouette tends to 1.

Furthermore, as it is specified in formula 6, BD-Dunn is the ratio between the minimum of the distances from the centroids to the global center and the maximum of the distances from each point in the set to its centroid. This way BD-Dunn returns a value that seeks the best number of clusters attending to its consistency and as much separation as possible.

$$BD\text{–}Dunn = \frac{min_{k=1..N}\{d(C_k, C_0)\}}{max_{x_i\in\Omega}\{d(x_i, c_k)\}} \tag{6}$$

The implemented algorithms return all the values of these formulas from 2 to a given number. This number sets the maximum of the quantity of clusters we could suppose optimal. BD-Dunn results show a set of values whereby we have to keep the first maximum. It represents the best number of clusters whose inter-cluster and intra-cluster distances are optimized.

Algorithm 1 shows the pseudo-code of BD-Silhouette method with precise details of the functions utilized from Spark. In the following, we describe the most significant instructions, enumerated from 1 to 13.

As input, we receive the dataset *data*, the maximum number of cluster that we want to check *maxNumClusters* and the number of iterations for the algorithm *numIterations*.

Algorithm 1 BD-Silhouette implementation

```
Require: $data$, $maxNumClusters$, $numIterations$
 1: $totalData = data.count()$
 2: for $i = 2$ to $maxNumClusters$ do
 3:     $clusters = Cluster.train(data, i, numIterations)$
 4:     $intraMean = clusters.computeCost(data)/totalData$
 5:     $centroides = sc.parallelize(clusters.clusterCenters)$
 6:     $clusterCentroids = Cluster.train(centroides, 1, numIterations)$
 7:     $interMean = clusterCentroids.computeCost(centroides)/i$
 8:     if $interMean \geq intraMean$ then
 9:         $max = interMean$
10:     else
11:         $max = intraMean$
12:     end if
13:     $Silhouette = (interMean - intraMean)/max$
14: end for
```

First, we create a cluster model from inputs (Instruction 3). In this way we get the *intraMean* by calculating the average of the distances between each point in the dataset to its centroid (Instruction 4). Next stage is the calculation of *interMean*. To this end we create a model from the centroids (Instruction 4) and we get the average of the distances between each centroid to the global center (Instruction 6-7). In this way we get the *interMean* value. The last step is to obtain the maximum between *intraMean* and *interMean* (Instruction 8-12) and get the silhouette value (Instruction 13). This sequence has to be done as many times as the *maxNumClusters* input is set (Instruction 2) for these instructions in order to calculate silhouette index for each number of clusters.

Below we present Dunn algorithm implementation. BD-Dunn algorithm receives the same input parameters: the dataset *data*, the maximum number of cluster that we want to check *maxNumClusters* and the number of iterations for the algorithm *numIterations*. Dunn algorithm implementation returns a set of values, one per number of cluster. Because of that it is in a loop and each iteration return one value of Dunn (Instruction 1).

The first step is to create a model from inputs (Instruction 2). With this model, we calculate the euclidean distance between each point and his centroid. And of all of these distances, we get the maximum one (Instruction 3). Next stage is to get the minimum distance between the centroids to global center (Instructions 4-6). To get this, the first step is to create a model with all the points (Instruction 5). With this model we can measure the distances between each centroid, and the global center. And with all of them, we get just the minimum one (Instruction 6). The last step is divide the minimum between the maximum (Instruction 7).

Algorithm 2 BD-Dunn implementation

```
Require: $data$, $maxNumClusters$, $numIterations$
1: for $i = 2$ to $maxNumClusters$ do
2:     $clusters = Cluster.train(data, i, numIterations)$
3:     $max = data.map\{x => Vectors.sqdist(x, clusters.predict(x))\}.max$
4:     $centroids = sc.parallelize(clusters.clusterCenters)$
5:     $clusterCent = Cluster.train(data, 1, numIterations)$
6:     $min = centroids.map\{x => Vectors.sqdist(x, clusterCent.clusterCenters)\}.min$
7:     $dunn = min/max$
8: end for
```

BD-Dunn method return a set of values, one for each iteration. From all the results the first maximum have to be chosen in order to choose the more compacted and well separated clusters.

2.2 Clustering Algorithms

As said above, machine learning algorithms from Spark library have been used for this study. In particular we have used K-Means (KM) and Bisecting K-Means (BKM). KM is a clustering algorithm commonly used that clusters the data into a predefined number of clusters. It splits data into k clusters in which each tuple belongs to the cluster with the nearest distance to the centroid. Moreover, BKM is a hierarchical clustering algorithm which seeks to build a hierarchy of clusters.

2.3 Datasets

In this experimental study we will use three synthetic Big Data (Table 1). These datasets have been specially generated for this task. We know the optimal number of cluster for each one in advance. The synthetic datasets have been specially generated for having well distinguished clusters. To achieve that objective, we have implemented an algorithm that receive the number of examples in each cluster and the number of features of the dataset. The algorithm generates an output file with random numbers in its columns and each column is in different range. After filling the cluster, the columns pivot, and it repeats the same operation. Between the ranges there is enough separation to not create overlaps between the clusters. With these conditions we make sure that the datasets have the number of clusters that we set.

Table 1. Description of used datasets

Dataset	#Examples	#Features	#Cluster
RandomSetN4	50,000,000	4	4
RandomSetN5	50,000,000	5	5
RandomSetN7	50,000,000	7	7

Table 1 summarizes the characteristics of these datasets. Table shows datasets dataset by rows, and by columns we can find the number of examples (#Examples), the number of features (#Features) and the number of cluster that has been established (#Cluster). As we said previously, we know the number of clusters for the datasets in advance. ”RandomSetN4” has four clusters, ”RandomSetN5” has five clusters and ”RandomSetN7” is the largest dataset with seven clusters. We may say that we are talking about Big Data because all the datasets have fifty millions instances with different number of features.

2.4 Hardware and software

All the experiments have been executed on Ubuntu Server 14.04 with the following features: Intel Xeon E7- 4820 processor with 8 cores (16 threads) and a clock speed of 2.00 GHz. Its cache was 18 MB and it counted with 64 GB of RAM. Spark 1.5.2 with Hadoop 2.6.0 was used with 20GB for the driver memory.

3 Experimental Results

In this section we describe the results that were carried out to analyse the implementation of BD-Silhouette and BD-Dunn clustering indices applied to Big Data. In order to test our algorithms, we have applied two clustering methods, K-means(KM) and Bisecting K-means(BKM). For these executions we have set the maximum number of cluster as 10 and were applied on three different datasets in Section 2.3.

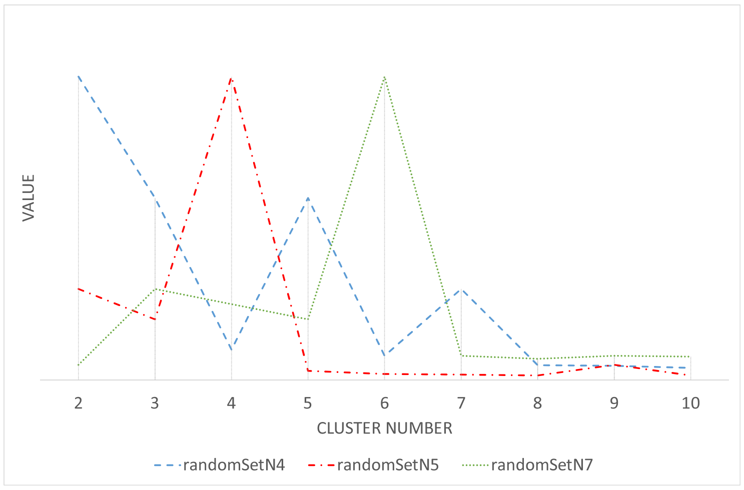

Fig. 1. BKM WSSSE

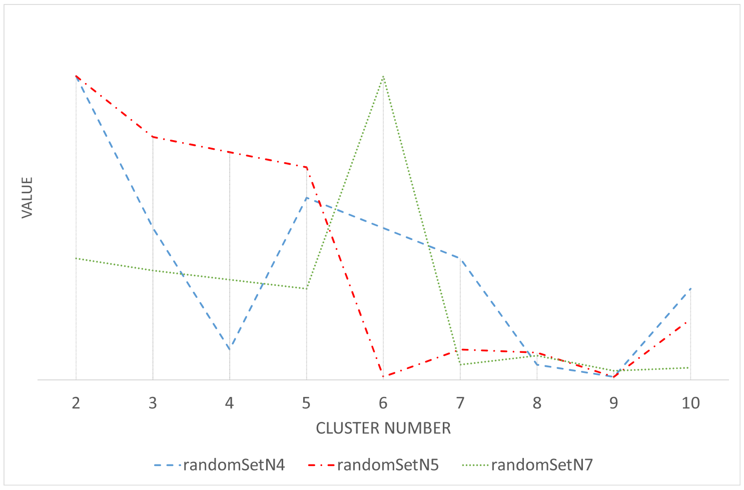

Fig. 2. KM WSSSE

Results of WSSSE are illustrated in Figure 1 and Figure 2. These results have been drawn in the same graph due to space problems in the paper. Figure 1 shows the results of WSSSE using BKM for each dataset. RandomSetN4 has four clusters, however the graph reports minimum values at four, six and eight. RandomSetN5 has five clusters and the graph reports minimums at three, five, and ten. Similarly, the dataset with seven groups, its graph illustrates minimum values in two, five and eight. The same situation is found in Figure 2. Unfortunately, WSSSE provides results that might mislead choosing the optimal number of clusters. .

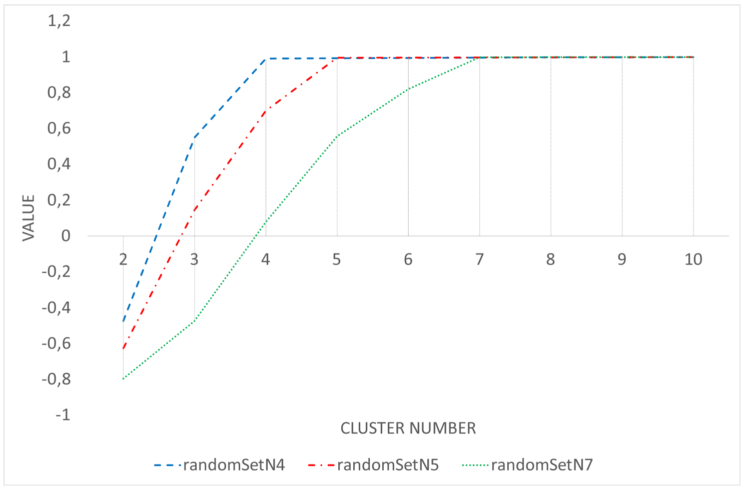

Fig. 3. BKM BD-Silhouette

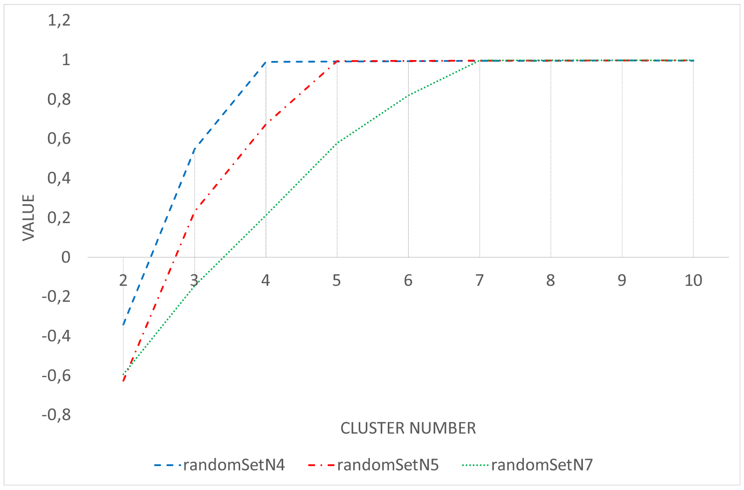

Fig. 4. KM BD-Silhouette

As we can see in Figure 3 and Figure 4, BD-Silhouette results for KM and BKM are represented. A well defined "elbow" is shown at the optimal number of cluster at each dataset. RandomSetN4 curve is increased until it reaches the maximum where it draws an elbow in the graph at four. Therefore, the optimal number of clusters for RandomSetN4 is four. The same situation happens for the other datasets. RandomSetN5 has its elbow at five, and RandomSetN7 has it at seven.

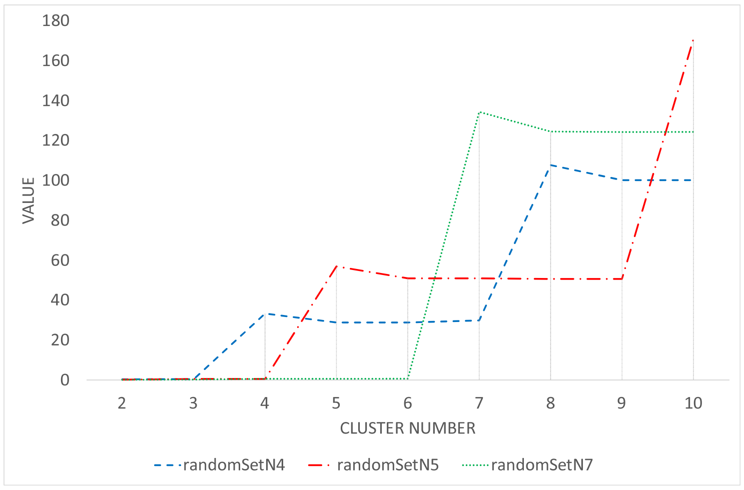

Fig. 5. BKM BD-Dunn Results

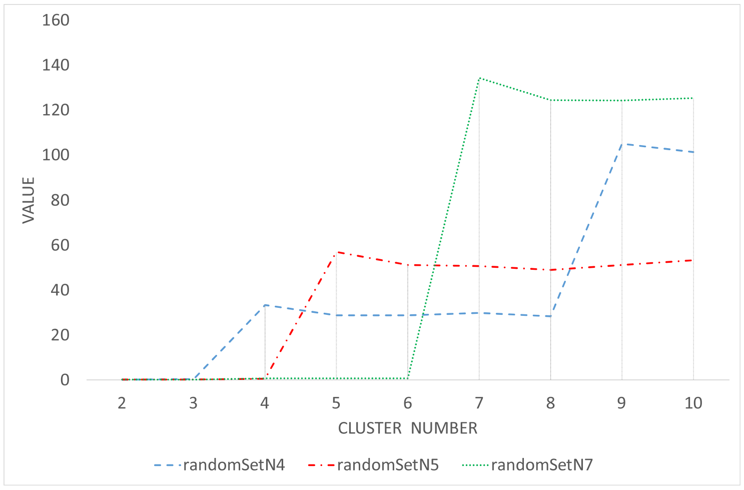

Fig. 6. KM BD-Dunn Results

Furthermore, we have BD-Dunn index graphs results. The optimal number of clusters for the dataset is represented by the first maximum in the graph. Therefore, as we can see in Figure 5 and Figure 6, in the case of RandomSetN4, the first maximum is reached at four. For RandomSetN5 the first maximum is reached at five, and it is at seven for RandomSetN7.

4 Conclusions and further work

In this paper we have developed a solution to calculate the optimal number of clusters in Big Data. The use of Apache Spark has provided us a way to parallelize the processes to calculate the optimal number of Big Data through Silhouette and Dunn approaches.

The main achievements obtained are the following:

– An approach to Silhouette and Dunn indices applied to Big Data.
– These approaches allow us to calculate the optimal number of clusters for Big Data using traditional clustering indices.
– The source code of this technique can be found in the next repository: https://github.com/josemarialuna/ClusterIndex.

As future work, we aim to focus on the development of an implementation of Davies–Bouldin [5,3] index applied to Big Data. All the experiments have been tested in a single machine thus next experiments will be tested in a cluster in order to check the algorithms in terms of network efficiency, load balancing and data locality. We Another direction for future work is to enhance our Silhouette and Dunn approaches focused in computational time.

Acknowledgement

This work has been supported by the Spanish National Research Project TIN2014-55894-C2-1-R. J.M. Luna-Romera holds an FPI scholarship from the Spanish Ministry of Education.

References

1. Abdelkader, M., Abd-Almageed, W., Srivastava, A., Chellappa, R.: Silhouette-based gesture and action recognition via modeling trajectories on riemannian shape manifolds. Computer Vision and Image Understanding 115(3), 439–455 (2011)
2. Al-Jarrah, O.Y., Yoo, P.D., Muhaidat, S., Karagiannidis, G.K., Taha, K.: Efficient machine learning for big data: A review. Big Data Research 2(3), 87 – 93 (2015), http://www.sciencedirect.com/science/article/pii/S2214579615000271, big Data, Analytics, and High-Performance Computing
3. Bezdek, J., Pal, N.: Some new indexes of cluster validity. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 28(3), 301–315 (1998)
4. Cane, J., O'Connor, D., Michie, S.: Validation of the theoretical domains framework for use in behaviour change and implementation research. Implementation Science 7(1) (2012)
5. Davies, D.L., B.D.: A cluster separation measure. Trans. Pattern Analysis and Machine Intelligence (1979)
6. Dunn, J.: Well-separated clusters and optimal fuzzy partitions. Journal of Cybernetics 4(1), 95–104 (1974)
7. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17(2-3), 107–145 (2001)
8. Jain, A., Murty, M., Flynn, P.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
9. Katal, A., Wazid, M., Goudar, R.: Big data: Issues, challenges, tools and good practices. pp. 404–409 (2013)
10. Lake, P., Drake, R.: Information Systems Management in the Big Data Era, chap. Introducing Big Data, pp. 1–18. Springer International Publishing, Cham (2014), http://dx.doi.org/10.1007/978-3-319-13503-8_1
11. Rousseeuw, P.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20(C), 53–65 (1987)
12. Spark, A.: Lightning-fast cluster computing . https://spark.apache.org/ (2016), [Online; accessed 01-April-2016]
13. Spark, A.: Clustering - spark.mllib. http://spark.apache.org/docs/latest/mllib-clustering.html (2016), [Online; accessed 01-April-2016]
14. Spark, A.: Machine Learning Library (MLlib) Guide. http://spark.apache.org/docs/latest/mllib-guide.html/ (2016), [Online; accessed 01-April-2016]
15. Starczewski, A., Krzyzak, A.: Performance evaluation of the silhouette index. vol. 9120, pp. 49–58 (2015)
16. Yang, C., Liu, C., Zhang, X., Nepal, S., Chen, J.: A time efficient approach for detecting errors in big sensor data on cloud. IEEE Transactions on Parallel and Distributed Systems 26(2), 329–339 (2015)
17. Zhu, L., Du, T., Qu, S., Wang, K., Zhang, Y.: An improved clustering algorithm for big data based on k-means with optimized clusters' number (2015)