An approach to validity indices for clustering techniques in Big Data

José María Luna-Romera[1] · **Jorge García-Gutiérrez**[1] · **María Martínez-Ballesteros**[1] · **José C. Riquelme Santos**[1]

Abstract Clustering analysis is one of the most used Machine Learning techniques to discover groups among data objects. Some clustering methods require the number of clusters into which the data is going to be partitioned. There exist several cluster validity indices that help us to approximate the optimal number of clusters of the dataset. However, such indices are not suitable to deal with Big Data due to its size limitation and runtime costs. This paper presents two clustering validity indices that handle large amount of data in low computational time. Our indices are based on redefinitions of traditional indices by simplifying the intra-cluster distance calculation. Two types of tests have been carried out over 28 synthetic datasets to analyze the performance of the proposed indices. First, we test the indices with small and medium size datasets to verify that our indices have a similar effectiveness to the traditional ones. Subsequently, tests on datasets of up to 11 million records and 20 features have been executed to check their efficiency. The results show that both indices can handle Big Data in a very low computational time with an effectiveness similar to the traditional indices using Apache Spark framework.

Keywords Clustering · Big Data · Clustering validity indices · Intra-cluster distance

✉ José María Luna-Romera
jmluna@us.es

Jorge García-Gutiérrez
jorgarcia@us.es

María Martínez-Ballesteros
mariamartinez@us.es

José C. Riquelme Santos
riquelme@us.es

[1] Department of Computer Languages and Systems, ETSII, University of Seville, Seville, Spain

1 Introduction

In the last few years, available data has been increased considerably. Medicine, electricity, business or biology are some areas where data has been quickly generated [4,7,13,29,31,40]. This information needs to be processed in order to discover knowledge, but traditional techniques are limited by the size of the data. This fact supposes a challenge to the research community because traditional machine learning methods cannot deal with large volume of data. Therefore, such learning techniques need to be redesigned to be able to handle Big Data.

Among the traditional techniques to discover knowledge, clustering can be useful to analyze large datasets with the aim at finding groups with similar behavior. Clustering is formally defined in [15] as the process of grouping a set of data objects into multiple groups or clusters so that objects within a cluster have high similarity, but are very dissimilar to objects in other clusters. Dissimilarities and similarities are assessed based on the attribute values describing the objects and often involve distance measures. Each clustering method generates different solutions on the same dataset. Clustering analysis is also applied to detect unknown associations within the data.

In particular, clustering techniques based on partitioning methods find the most suitable partition of the objects of the dataset into a given number of groups optimizing a chosen partitioning criterion. Nevertheless, such methods require the optimal number of clusters that the dataset is going to be partitioned. For this task, there exist cluster validity indices (CVI) that help to calculate it. The application and usability of these indices has been proven in several works in the literature [1–3,25]. However, the traditional indices are not suitable to deal with large datasets due to the high computational time costs and their inability to be parallelized. Traditional CVIs

use pairwise distances, so such CVIs will have quadratic complexity. The use of this kind of CVIs on large data could take much longer to compute the evaluation measure than running the clustering algorithm.

Nowadays, some frameworks are able to deal with Big Data. One of the first frameworks that allowed processing large datasets was Apache Hadoop [9]. Hadoop allows to work across clusters of computers using simple programming models based on Google's MapReduce paradigm [9,14]. Additionally, one of the most used Big Data projects is the open-source cluster computing framework named Apache Spark [34]. Spark appeared as alternative to solve memory limitation that MapReduce suffered. MapReduce reads and writes from hard drive, as a result, it slows down the processing speed. Spark reduces the number of read/write cycles to disk and stores intermediate data in faster logical RAM memory. It uses an structured data, named RDD, especially designed for parallel computing that caches results in memory for processing large amounts of data [38]. Apache Spark contains an scalable Machine Learning library (MLlib) with a set of algorithms to handle classification, regression, decision tree, recommendation systems and clustering techniques [35].

The purpose of this paper is to show the limitations of traditional clustering indices and to present novel validity indices that can tackle Big Data, henceforth named BD-CVI. In particular, the proposed indices are implemented using Apache Spark framework. A data generator application is also presented to ensure the composition of data and to test the performance of the proposed indices. K-means method was selected for testing the performance of these CVIs and BD-CVIs.

The remainder of this paper is organized as follows. Section 2 presents an outline of the background about clustering including a description of traditional indices and the state-of-the-art of Big Data clustering. Section 3 defines the BD-CVI proposed in this paper. Section 4 shows the experimental analysis and the obtained results. Section 5 reports the conclusions drawn by this paper. Lastly, "Appendix A" details the dataset generator application that was implemented to be used in the experimentation.

2 Related work

In this section, clustering analysis is formally defined and a general classification of the main clustering methods is also presented (Sect. 2.1). In addition, main CVIs are described and classified by categories (Sect. 2.2). Furthermore, we provide a brief overview of previous works related to clustering analysis in Big Data (Sect. 2.3).

2.1 Clustering methods

Cluster analysis can be used as a mechanism to achieve a custom vision of the distribution of the data, to observe the features of each cluster and to target on a particular subset of data for other analysis. It is also used as a preprocessing step for further algorithms, such as classification or features selection, which would deal with the detected clusters and the selected features. In some cases, clustering analysis is also called automatic classification since clustering is a collection of similar data objects on each cluster, so data objects within the same cluster can be managed as an implicit class. Clustering can be found in the literature as data segmentation in some applications because large datasets are partitioned into groups by their similarity. Another use of clustering is the outlier detection, that could be defined as that data object that is far away from any cluster and it may be more interesting to not to include it in any of them. Some applications of clustering for outlier detection can be found in [26].

Clustering analysis is considered a branch of statistics, and it has been widely studied as distance-based cluster analysis. Clustering analysis implementations based on K-means or K-medoids have been developed into many statistical analysis software packages and systems [18]. In Machine Learning, clustering is a method of unsupervised learning, so the data object has no class label information. Clustering learns by observation instead of learning by examples. Some of the active research topics are focused on the scalability of clustering methods [6], the effectiveness of types of data [32], high-dimensional clustering techniques [18], and methods for clustering mixing numerical and nominal data in large datasets.

There exist many clustering algorithms in the literature, so it is difficult to set a categorization. In many cases, an algorithm can be classified into several categories due to its features. However, most of the clustering techniques could be classified into the following categories [12]:

Partitioning methods Given a set of n objects, a partitioning method constructs k groups of data, where each partition represents a cluster and $k \leq n$. It splits the data objects into k clusters such each cluster must contains at least one data object. Most of these methods are distance-based, so given k, which is the number of groups to build, a partition method sets an initial solution. Then, it iterates and try to improve the solution by moving objects between the groups. Despite each clustering method takes its own criterion, a satisfying partitioning is where data objects in the same cluster are close and objects in different clusters are far away. Clustering methods usually work properly with spherical-shaped cluster when this operation is used [18].

Hierarchical methods This kind of methods create a hierarchical decomposition of the given set of data objects. It

successively groups the objects close to one another until all the data objects are merged into one. This kind of clustering methods leads to smaller computation costs by not having to worry about a combinatorial number of different choices due to once a step is done it can never be undone.

Density-based methods This methods iteratively build a cluster as long as the density, defined as the number of objects in a cluster, exceeds some threshold. This kind of methods is used to detect outliers and discovers random-shape clusters.

Grid-based methods This kind of methods compute the object space into a finite number of cells that form a grid structure. They are independent of size of the dataset and dependent on the number of cells in each dimension in the quantized space.

2.2 Cluster validity indices

As stated in the introduction, this paper is focused on partitioning clustering methods because they need the number of clusters into which the dataset is going to be partitioned. Knowing a priori a proper approximation to the number of cluster could be very useful for any clustering algorithm, especially hierarchical ones. For this task, several CVIs have been proposed in the literature. A summary of the most representative CVIs of each category are presented as follows [10,33]:

- *Indices measuring compactness of clusters* These indices measure both the distance between the points that belong to the same cluster and the compactness of them:

  *Maximum Cluster Diameter (Δ) [16] is the highest diameter among all the clusters in the dataset. It is calculated by the maximum distance between two points that belong to the same cluster.
  *Average Within-Cluster Distance (W) [33] measures the average distance of the points that belong to the same cluster.

- *Indices measuring separation between clusters* This category evaluates the separation of the clouds of points and grades it when there is a gap between them:

  *Average Between-Cluster Distance (β) [33] is the average distance of the points that are in different clusters.

- *Indices measuring relationships between compactness and separation* This category measures the ratio between the compactness of the clusters and the existing separation between them:

  *Silhouette [30] is a measure that sets how compacted are the points that belong to the same cluster against the separation between the clusters.

  *Dunn [11] measures the relation between the minimum inter-cluster distance and the maximum intra-cluster distance.

  *Davies and Bouldin [8] is a measure that uses data object quantities and features inherent to the dataset to set the compactness and separation of the clusters.

  *Calinski and Harabasz [5] is based on getting a relation between the inter-cluster distance and the intra-cluster distance.

2.3 Clustering in Big Data

Clustering analysis in Big Data has been the main focus of a lot of researchers in the last years. Some of the most relevant papers in this field are analyzed below.

Two C-means algorithms based on the canonical polyadic decomposition and the tensor-train network for clustering Big Data are proposed in [39]. They stated that the algorithms are suitable for Big Data clustering in Internet of Things systems with low-end devices since they can achieve a high compression rate for heterogeneous samples to save the memory space significantly.

Mohammed et al. [27] proposed a new cluster algorithm named FireflyClust, that it can deal with text documents in a hierarchical line. FireflyClust can handle Big Data, overcoming other methods such as Bisect K-means, hybrid Bisect K-means and Practical General Stochastic Clustering Method. In this case, the algorithm does not need the number of cluster as input parameter.

An effective K-means algorithm design was proposed in [37]. The algorithm is based on MapReduce programming model that acquires a fast detection speed with a high scale-up. However, no method was applied to identify the optimum number of clusters, so the algorithm was tested using different k number of clusters.

Jerome and ätönen [20] proposed a hierarchical clustering technique for classifying anomalies into clusters and providing information regarding the behavior of the anomaly cluster by analyzing its centroid in Big Data. Overall, it is easier to detect anomalies than finding out reasons for anomalous behavior. This technique was also used to determine the severity of the anomaly by using a failure significance metric.

A novel cluster center fast determination clustering algorithm for Big Data was proposed in [21]. The algorithm is based on the density and distance distribution of the data objects to determine the cluster center quickly by constructing the normal distribution function.

Kim et al. [22] suggested an optimized combinatorial clustering algorithm for noisy performance with random sampling for Big Data. The algorithm outperforms conventional approaches through various numerical and qualitative

thresholds such as mean and standard deviation of accuracy and computation speed.

Tong et al. [36] proposed Scalable Clustering Using Boundary Information, a highly flexible and scalable clustering scheme. To achieve this, such algorithm firstly identifies the border points of the dataset, and then it groups boundary points into suitable clusters and includes the rest points to their nearest border point. The obtained reports confirm similar results than the standard DBSCAN method, but such method is able to handle Big Data.

A novel method for assessing the robustness of clusters for partitioning algorithms is introduced in [23]. However, such method is not applied to Big Data, and moreover, the experiments have been carried out with supervised data where classes have been used as clusters.

3 Big Data indices

As stated in Introduction, the main purpose of this paper is to provide efficient and suitable CVIs able to deal with Big Data. The proposed BD-CVIs are approximations of traditional indices because those indices require high computational cost and they are unable to be parallelized. Firstly, traditional CVIs are described in Sect. 3.1. Secondly, the definition of BD-CVIs is in Sect. 3.2.

3.1 Traditional CVIs

From the traditional indices, we have selected those that had the best performance in the experiments (Sect. 4.3.3), and thus Silhouette and Dunn are detailed below:

Let Ω be the space of the objects 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_0 the centroid of Ω.

Let $x_i \in A_k$, the distance from x_i to the own cluster A_k is defined:

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

where $a_k(x_i)$ represents the dissimilarity of x_i to all other points within the same cluster k and

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

$b_k(x_i)$ is the smallest average dissimilarity of $x_i \in A_k$ to the points in other clusters.

– *Silhouette* is defined in [30] as (Eq. 4):

$$s_k(x_i) = \frac{b_k(x_i) - a_k(x_i)}{max\{a_k(x_i), b_k(x_i)\}} \tag{3}$$

$$Silhouette = \frac{1}{|\Omega|} \sum_{x_i \in \Omega} s_k(x_i) \tag{4}$$

Silhouette index ranges $[-1, 1]$, where good values are near 1 and -1 closer values are bad clustering solutions.

– *Dunn* index is defined in [11] and its purpose is to identify compact and well-separated clusters. For a given number of clusters N, the following equation defines the Dunn index:

$$Dunn = \frac{\min\limits_{k=1...N} \{d(C_k, C_j),\ k \neq j\}}{\max\limits_{k=1...N} \{\max d(x_i, x_j), i \neq j, x_i, x_j \in A_k\}} \tag{5}$$

In a compact and well-separated clusters dataset, the distances between the clusters are wide and the distances between the points of the same cluster are small. Hence, a high value of the Dunn index means a compact and well-separated clusters solution.

3.2 BD-CVIs

In this subsection, BD-Silhouette and BD-Dunn are going to be formally introduced:

– *BD-Silhouette* is defined by two approaches to intra-cluster and inter-cluster mean distances.
inter-cluster (Eq. 6) is the average of distances between each cluster centroid and global centroid C_0:

$$inter\text{-}cluster = \frac{1}{N} \sum_{k=1}^{N} d(C_k, C_0) \tag{6}$$

where C_0 is the center of the centroids of the clusters.
intra-cluster (Eq. 8) distance is defined as the average of the distances between each point to the centroid of the cluster to which it belongs (Eq. 7).

$$r_k = \frac{1}{|A_k|} \sum_{x_i \in A_k} d(x_i, C_k) \tag{7}$$

$$intra\text{-}cluster = \frac{1}{|N|} \sum_{x_i \in A_k}^{N} r_k \tag{8}$$

Traditional Silhouette index takes *intra-cluster* distance instead, that is defined by the average distance between

the points that belong to the same cluster (Eq. 1). The *intra-luster* distance is the main difference in BD-Silhouette.

Equation 9 represents BD-Silhouette that has been defined as the ratio between the difference of the *inter-cluster* and the *intra-cluster*, and the maximum of them.

$$BD\text{-}Silhouette = \frac{inter\text{-}cluster - intra\text{-}cluster}{max\{inter\text{-}cluster, intra\text{-}cluster\}} \tag{9}$$

BD-Silhouette returns a value in $(-1, 1)$, depending on the consistence of the cluster and the separation between them. The higher the cluster number is, the lower *intra-cluster* is because the points of the dataset tend to be more compact. BD-Silhouette takes the value -1 if a single cluster is defined for all the examples and tends to 1 when the number of clusters is increased. BD-Silhouette would be 1 in the extreme case of each data object being a cluster. Therefore, an optimal value for the number of clusters would be the first maximum of BD-Silhouette, which maximizes the coherence of the cluster with the lowest k possible.

– *BD-Dunn* simplifies the original Dunn index to facilitate its computation in Big Data, since it does not have to calculate in the denominator the distance between each pair of points of the dataset. On the contrary, original Dunn seeks the minimum distance between the centroids and the maximum distance between all the points that belong to the same cluster. Thus, BD-Dunn (Eq. 5) 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.

$$BD{-}Dunn = \frac{\min\limits_{k=1\ldots N}\{d(C_k, C_0)\}}{\max\limits_{k=1\ldots N}\max\limits_{x_i \in A_k}\{d(x_i, C_k)\}} \tag{10}$$

BD-Dunn takes the value 0 if we define a single cluster for all the examples. However, BD-Dunn tends to infinity when the number of clusters increases. In the extreme case of each example belong to a different cluster, its value cannot be calculated because the denominator is zero.

Figure 1 illustrates a distribution of a dataset with 2 features and 3 clusters in different colors. The clusters are represented by circles, and the points are the red dots. Each cluster i has its centroid denoted by C_i, and the global centroid as C_0 is also represented. Blue cluster has also highlighted some points in the cluster. In the figure, *inter-cluster* distance is represented by the red cluster as $d(C_1, C_0)$ that measures the distance between the centroid of the red cluster and the global centroid. The *intra-cluster* distance is represented in the blue cluster as the distance between the point X_1 and its centroid C_2.

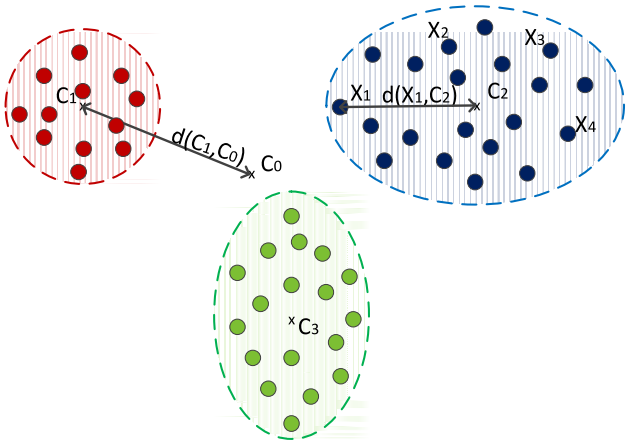

Fig. 1 Representation of 3 clusters with *inter-cluster* and *intra-cluster* distance and the global centroid (C_0)

As it happens with traditional CVIs, BD-CVIs return a value on each clustering solution. To get the optimum number of clusters of a large dataset, BD-CVI could be calculated on each clustering execution. The optimal number of clusters is chosen following a different criterion on each BD-CVI. BD-Silhouette and BD-Dunn are growing indices. BD-Silhouette reaches 1 when $k = N$, and BD-Dunn tends to infinity. Thus, in both BD-CVIs, the first maximum is a satisfactory solution because it maximizes the clustering coherence with the lowest number of clusters possible.

Figure 2 illustrates the graphical representation of the results of BD-Silhouette and BD-Dunn for a dataset with 5 clusters and 500,000 instances each. BD-Silhouette value increases with the number of clusters. Such index marks a possible optimal number of clusters when there is a change of trend in the values. In Fig. 2, BD-Silhouette is increased by the number of clusters until $k = 5$. This change of trend indicates that $k = 5$ may be an optimal number of clustering. BD-Dunn reveals the optimal number of clusters with the first maximum value of its plot. Figure 2 shows a first maximum value on $k = 5$, where the line of BD-Dunn increases with the number of clusters and decreases in $k = 6$. This inflection point indicates that $k = 5$ could be the optimal number of clusters.

4 Experimental study

The experimental setup and the results are detailed in this section. A comparative framework is also presented to test

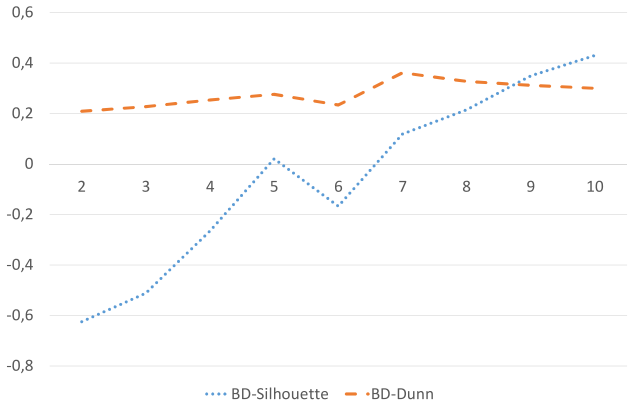

Fig. 2 BD-Silhouette and BD-Dunn results for a dataset with 5 clusters and 500,000 instances each

both CVIs and to test which CVIs and BD-CVIs have the best performance (Sects. 4.3.3, 4.4.3).

4.1 Experimental setup

4.1.1 Software and hardware

In this paper, we compared the results of traditional CVIs and the proposed BD-CVIs with the datasets described in Sect. 4.1.2. A clustering algorithm is required to test the performance of the proposed BD-CVIs. As stated in Sect. 2, K-means is a partitioning method that previously needs the number of clusters into which the dataset is going to be partitioned, so that this algorithm has been selected in the experimental study. In addition, it is the paradigmatic clustering algorithm [19] and it is one of the available algorithms in Spark MLlib [35]. In the case of traditional CVIs, we have used the K-means package available in the Weka Software developed in Java [17].

Two different execution environments were used in our experiments. On the first hand, traditional CVIs have been tested in the EC2 instances from Amazon Web Services (AWS) that count with Intel Xeon E5-2666 v3 (Haswell) processors, 3.75 GB RAM memory and enough hard disk to manage datasets originally stored in AWS S3. On the other hand, BD-CVIs were executed in AWS Elastic Map Reduce. 5 instances of $m3.xlarge$ that each one counts with Intel Xeon E5-2670 v2 (Ivy Bridge) processors with 4 vCPU, 15 GB RAM memory and 2 SSD of 40 GB were used.

4.1.2 Generated datasets

A total of 28 datasets have been used which are generated using the dataset generator application described in "Appendix A". In order to test the limits of the CVIs and the novel BD-CVIs, several combinations of number of clusters and number of instances were applied. Table 1 shows the main features of the generated datasets in the experiments. There are 4 different groups of datasets with 5, 7, 9 and 11 clusters. Each group of datasets contains 7 datasets, with 1000, 2000, 5000, 10,000, 100,000, 500,000 and 1 million instances per cluster. Thus, a dataset with 5 clusters and 1000 instances per cluster has a total of 5000 instances. The datasets are easily identified following the next pattern: $C - N\{k, M\}$ where C is the optimal number of clusters of the dataset, N is the number of instances of each cluster multiplied by a thousand if it is followed by a k, or by a million if it is followed by a M. For example, 5–10k dataset has 5 clusters and each one contains 10,000 instances, so it contains a total of 50,000 instances. All the datasets were created with 20 features, the standard deviation was 0.05, and the mean was 0.25 and 0.75 to ensure the separation of the clusters.

Table 1 Generated datasets with number of clusters, total number of instances and the size in MB

Dataset	Clusters	Instances	Size (MB)
5–1k	5	5000	1.00
5–2k	5	10,000	2.00
5–5k	5	25,000	5.00
5–10k	5	50,000	10.00
5–100k	5	500,000	100.00
5–500k	5	2,500,000	501.00
5–1M	5	5,000,000	1003.52
7–1k	7	7000	1.40
7–2k	7	14,000	2.80
7–5k	7	35,000	7.00
7–10k	7	70,000	14.00
7–100k	7	700,000	140.00
7–500k	7	3,500,000	703.00
7–1M	7	7,000,000	1402.88
9–1k	9	9000	1.81
9–2k	9	18,000	3.62
9–5k	9	45,000	9.03
9–10k	9	90,000	18.00
9–100k	9	900,000	181.00
9–500k	9	4,500,000	903.00
9–1M	9	9,000,000	1802.24
11–1k	11	11,000	2.21
11–2k	11	22,000	4.41
11–5k	11	55,000	11.05
11–10k	11	110,000	22.10
11–100k	11	1,100,000	221.00
11–500k	11	5,500,000	1095.68
11–1M	11	11,000,000	2211.84

Table 2 Distance of traditional CVIs to the optimal solution by dataset

	Silhouette	Dunn	David–Bouldin	Calinski–Harabasz	Δ	W	β
5–1k	**0**	**0**	**0**	**0**	**0**	**0**	**0**
5–2k	**0**	**0**	**0**	**0**	**0**	**0**	**0**
5–5k	**0**	**0**	**0**	**0**	**0**	**0**	**0**
5–10k	**0**	**0**	**0**	**0**	**0**	**0**	**0**
7–1k	1	**0**	1	1	1	1	1
7–2k	**0**	**0**	**0**	**0**	**0**	**0**	**0**
7–5k	1	1	1	1	2	2	2
7–10k	2	**0**	2	2	3	2	3
9–1k	2	1	1	2	2	2	2
9–2k	1	2	1	1	1	2	1
9–5k	**0**	1	**0**	1	**0**	**0**	**0**
9–10k	**0**	2	**0**	2	2	3	3
11–1k	1	**0**	2	2	**0**	3	3
11–2k	**0**	**0**	**0**	**0**	**0**	**0**	**0**
11–5k	**0**	2	**0**	**0**	4	**0**	**0**
11–10k	2	2	2	2	2	2	3
Total	10	11	10	14	17	16	17

The hits results are highlighted in bold

4.2 Results

This section is divided in two subsections. Section 4.3 contains the results for the traditional CVIs and Sect. 4.4 shows the results for BD-CVIs. Each subsection includes the effectiveness results and the computational cost. Section 4.3 includes a statistical analysis to compare the effectiveness among CVIs. Section 4.4 provides a statistical analysis to compare the execution time among BD-CVIs.

After executing the CVIs, the effectiveness and execution time of each index are measured. The effectiveness of the indices is calculated by the absolute value of the difference to the optimal solution. Thus, an index with 0 value is considered that it correctly predicts the optimal number of clusters, whereas an index with 2 means that predicts two number above or below the optimal solution. The goodness of fit of the indices is given by the sum of the absolute values of the differences between the real value and the estimated. Therefore, the lower the value, the better the index. The statistical analysis was carried out using the open-source platform StatService [28].

4.3 Results of traditional CVIs

4.3.1 Effectiveness

Table 2 shows the distance to the optimal number of clusters given by the CVIs on each dataset. The datasets were chosen until execution time was under 86,400 s (1 day). The correctly predicted clusters are highlighted in bold, and the last row of the table is the total of distances of each CVI. The lower the value is, the better the CVI is.

Silhouette, Dunn and David–Bouldin obtained the best results since they had the lowest total of distances. The worst results (highest distances) were obtained by Maximum Cluster Diameter (Δ), Average Within-Cluster Distance (W) and Average Between-Cluster Distance (β).

Figure 3a, b shows graphically the number of cluster by Silhouette, Dunn, Davies–Bouldin and Calinski–Harabasz. Δ, W and β were not included in these figures because their results were not so positive. The optimal results are represented by big red dots, so each CVI whose point is on it means that correctly predicted the optimal number of clusters. Datasets with 5 and 7 clusters are included in Fig. 3a and datasets with 9 and 11 clusters are in Fig. 3b.

CVIs obtained very good results in Fig. 3a. Almost all the represented CVIs correctly predicted the optimal solution. Dunn correctly predicted all the datasets except 7–5k. However, Dunn was the only one CVI that set the optimal number of clusters in two datasets in Fig. 3b. 11–10k, 9–1k and 9–2k number of clusters were not estimated by any of the CVI in this figure.

The results of the CVIs do not directly depend on the number of instances of the dataset. The results may be influenced by the number of clusters of the dataset. The lesser number of clusters have the dataset, the greater is the ratio of correct predictions. The optimal number of clusters for datasets with 5 clusters were correctly predicted by all the indices. The optimal number of clusters for datasets with 7 clusters was the most difficult to predict.

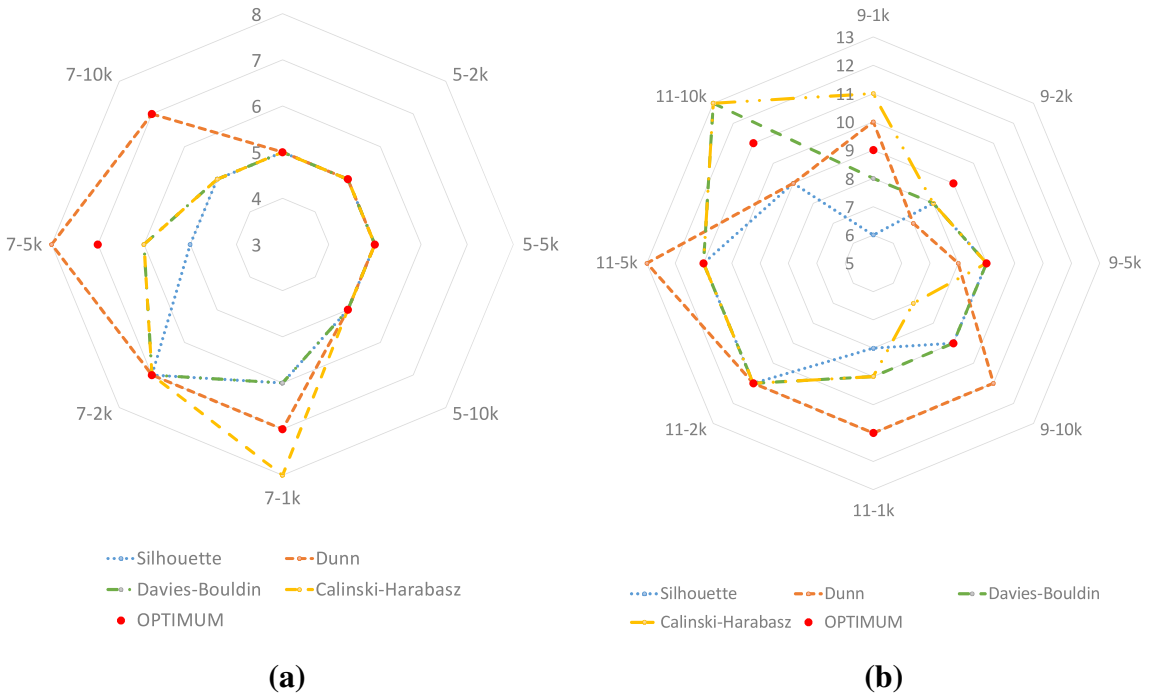

Fig. 3 Results of traditional CVI Silhouette, Dunn, Davies–Bouldin, Calinski–Harabasz for different datasets. Optimal solution by a red dot. **a** Results for datasets with 5 and 7 clusters. **b** Results for datasets with 9 and 11 clusters (color figure online)

Table 3 Average of elapsed time of CVIs in seconds

	Silhouette	Dunn	David–Bouldin	Calinski–Harabasz	Δ	W	β
5–1*k*	9.34	9.33	0.01	2.00	1.99	1.99	7.31
5–2*k*	40.54	40.55	0.01	8.48	8.55	8.59	31.99
5–5*k*	224.46	224.49	0.02	47.20	47.43	47.65	176.91
5–10*k*	1586.37	1584.48	0.48	346.66	340.48	341.01	1238.27
7–1*k*	18.51	18.51	0.01	3.87	3.92	3.92	14.50
7–2*k*	73.48	73.62	0.01	15.73	15.41	15.44	58.14
7–5*k*	906.27	905.93	0.03	450.55	431.62	453.36	368.85
7–10*k*	34,771.06	34,540.32	0.43	91,003.92	86,179.63	99,349.86	3707.81
9–1*k*	20.96	21.12	0.01	2.45	2.35	2.51	18.24
9–2*k*	281.41	278.27	0.02	137.06	136.44	138.55	137.44
9–5*k*	4823.15	4686.26	0.08	14,269.38	10,143.56	9776.66	1330.02
9–10*k*	–	–	0.49	–	–	–	16,552.37
11–1*k*	35.35	34.26	0.01	4.09	3.79	3.55	29.34
11–2*k*	497.75	495.99	0.02	246.05	248.54	246.80	246.21
11–5*k*	8945.03	9178.18	0.14	23,155.30	20,852.84	21,976.94	2653.13
11–10*k*	–	–	1.13	–	–	–	36,766.93

4.3.2 Execution time

Table 3 shows the execution time in seconds of each traditional CVI. Figure 4 is added for better understanding of Table 3. Time is generally increased with the number of instances of the dataset. The lowest execution time was obtained by Davies–Bouldin with very high difference respect to the other CVIs. Davies–Bouldin lasts 1.13 s for the largest dataset (11–10k), while there were some indices whose execution time was higher than 86,400 s (1 day). Such cases are marked as "–". There is a significant increase in the runtime of traditional CVIs for datasets with more than

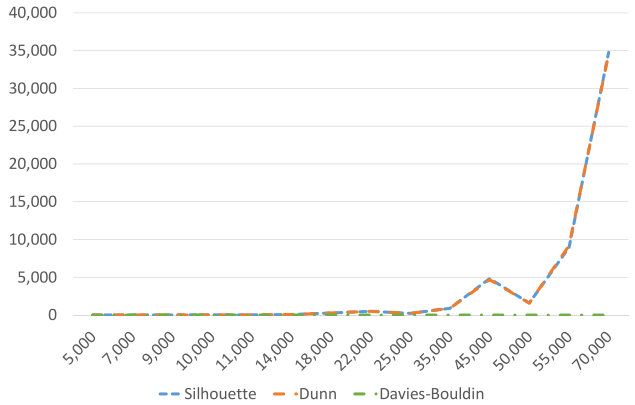

Fig. 4 Representation of traditional CVIs time by the number of instances by dataset

Table 4 Sorted ranking of traditional CVIs for Friedman test

CVI	Ranking
David–Bouldin	3.406
Silhouette	3.531
Dunn	3.656
Calinski–Harabasz	3.969
Δ	4.250
W	4.500
β	4.688

50,000 instances. It can be noted that runtime is four times higher in those datasets even though the growth in the number of instances is less than 50%.

4.3.3 Statistical analysis

After the results generation of the traditional CVIs, a statistical analysis was applied to check if significant differences exist among the effectiveness of the multiple CVIs. The non-parametric Friedman test is shown in Table 4. The highest result for a ranking would be 1, and the worst would be 7. As the ranking shows, Davies–Bouldin was in the first position with 3.406, followed by Silhouette and Dunn with 3.531 and 3.656, respectively.

The statistic for Friedman was 5.0625, distributed according to a Chi-square distribution with 6° of freedom. The p value for Friedman was 0.5358 and higher than 0.05. Therefore, the null hypothesis was accepted that they all behaved in a similar way with a level of significance of $\alpha = 0.05$.

Given the results of Table 3, it makes no sense to perform a statistical test to show that Davies–Bouldin was the fastest CVI.

4.4 Results of BD-CVIs

4.4.1 Effectiveness

BD-CVIs were applied to all datasets from Table 1, including those datasets used in Sect. 4.3. Davies–Bouldin was also included in these experiments for its great results in terms of efficiency in the previous experiments.

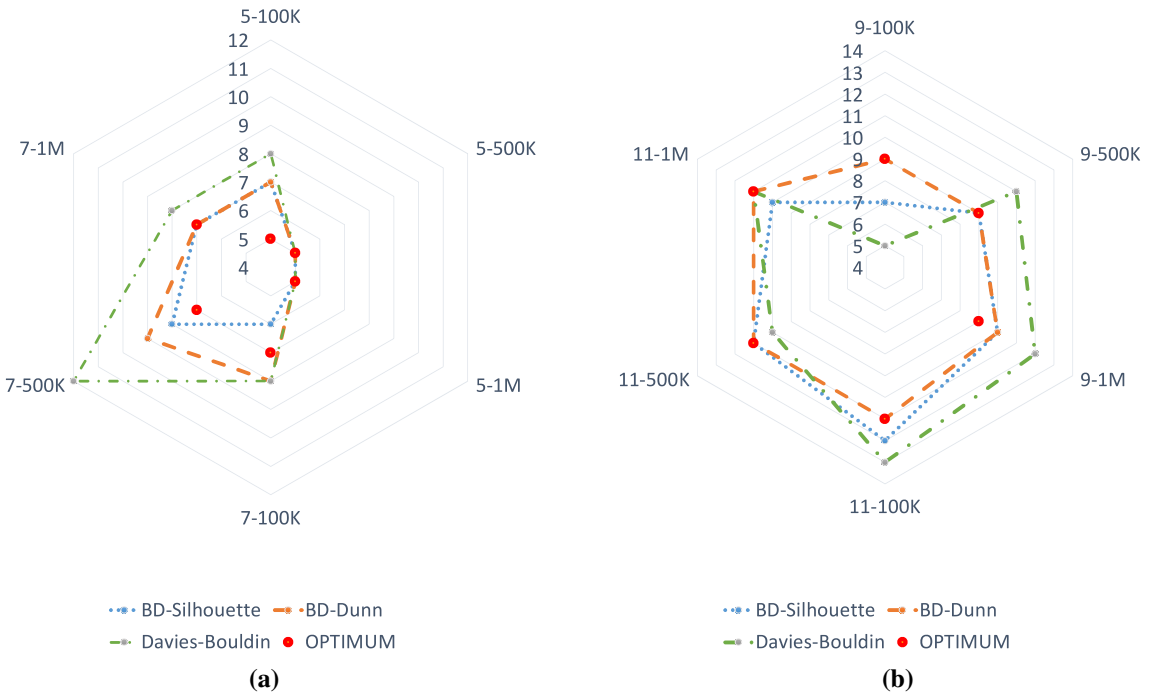

Fig. 5 Results of BD-Silhouette, BD-Dunn and Davies–Bouldin for different datasets. Optimal solution by a red dot. **a** Results for datasets with 5 and 7 clusters. **b** Results for datasets with 9 and 11 clusters (color figure online)

Table 5 Distance of BD-CVIs to the optimal solution by dataset

	BD-Silhouette	BD-Dunn	Davies–Bouldin
5–1*k*	**0**	**0**	**0**
5–2*k*	**0**	**0**	**0**
5–5*k*	**0**	**0**	**0**
5–10*k*	**0**	**0**	**0**
5–100*k*	2	2	3
5–500*k*	**0**	**0**	**0**
5–1*M*	**0**	**0**	**0**
7–1*k*	**0**	**0**	**0**
7–2*k*	**0**	**0**	**0**
7–5*k*	**0**	**0**	**0**
7–10*k*	**0**	**0**	**0**
7–100*k*	1	1	1
7–500*k*	1	2	2
7–1*M*	**0**	**0**	1
9–1*k*	**0**	**0**	**0**
9–2*k*	**0**	**0**	**0**
9–5*k*	**0**	**0**	**0**
9–10*k*	1	3	1
9–100*k*	2	**0**	4
9–500*k*	**0**	**0**	2
9–1*M*	1	1	3
11–1*k*	**0**	**0**	1
11–2*k*	**0**	**0**	**0**
11–5*k*	**0**	**0**	**0**
11–10*k*	1	1	2
11–100*k*	1	**0**	2
11–500*k*	**0**	**0**	1
11–1*M*	1	**0**	**0**
Total	11	10	23

The best results are highlighted in bold

Figure 5a, b shows graphically the results of each BD-CVI by dataset. Red dots highlight the optimal results of each dataset. There were some datasets whose optimal solution was not given by any BD-CVI. However, BD-Silhouette and BD-Dunn correctly predicted most of the datasets; meanwhile, Davies–Bouldin was too far to the optimal like in datasets 7–500*k* or 9–1*M*.

Table 5 shows the distances to the optimal solution of each BD-CVI by dataset. This table shows that the optimal number for datasets with 5 clusters was correctly set by the three indices. Davies–Bouldin did not guess any dataset that BD-Silhouette or BD-Dunn could not. In fact, if BD-Silhouette and BD-Dunn set correctly the optimal number, BD-Davies–Bouldin did it too. There were two cases where BD-Silhouette was the only one BD-CVI that sets the optimal number of clusters correctly.

Table 6 Average of elapsed time of BD-CVIs in seconds

	BD-Silhouette	BD-Dunn	Davies–Bouldin
5–1*k*	0.10	0.06	0.64
5–2*k*	0.11	0.05	0.52
5–5*k*	0.09	0.05	0.63
5–10*k*	0.13	0.10	0.80
5–100*k*	0.22	0.29	0.75
5–500*k*	0.48	0.68	1.26
5–1*M*	0.95	1.67	2.66
7–1*k*	0.11	0.05	0.64
7–2*k*	0.09	0.06	0.61
7–5*k*	0.11	0.08	0.70
7–10*k*	0.12	0.10	0.74
7–100*k*	0.25	0.33	0.66
7–500*k*	0.61	0.94	1.61
7–1*M*	7.15	4.99	6.56
9–1*k*	0.10	0.06	0.80
9–2*k*	0.11	0.06	0.77
9–5*k*	0.10	0.08	0.71
9–10*k*	0.10	0.08	0.83
9–100*k*	0.66	0.85	1.97
9–500*k*	1.72	3.22	3.97
9–1*M*	16.83	9.98	11.42
11–1*k*	0.09	0.06	0.80
11–2*k*	0.12	0.09	0.90
11–5*k*	0.12	0.10	0.98
11–10*k*	0.15	0.20	0.99
11–100*k*	0.28	0.41	1.10
11–500*k*	1.47	2.79	3.69
11–1*M*	25.06	15.23	17.61

4.4.2 Execution time

Table 6 illustrates the total time in seconds after applying BD-CVIs on each dataset. Figure 6 was added to better understand the behavior of Table 6. In datasets where traditional CVIs took more than a day, BD-CVIs took less than 25 s. It is noteworthy that BD-CVIs perform similarly to traditional CVIs. In fact, there is a change in trend of datasets with more than 6 millions instances, as happened in traditional CVIs when the number of instances was 50,000. In the case of BD-CVIs, the runtime had a 400% increase when the number of instances was higher than 6 millions even though the number of instances only has an increment of 100%.

4.4.3 Statistical analysis

Two statistical tests were applied to check the significance in the differences of BD-CVI results, in terms of effectiveness and execution time.

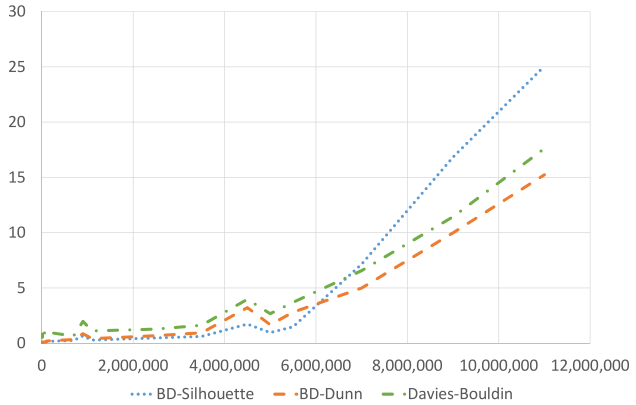

Fig. 6 Representation of BD-CVIs time by the number of instances by dataset

Table 7 Sorted ranking of effectiveness BD-CVI for Aligned Friedman test

BD-CVI	Ranking
BD-Silhouette	35.17
BD-Dunn	36.51
Davies–Bouldin	55.80

Table 8 Post hoc analysis using Holm's procedure and BD-Silhouette as the control algorithm

BD-CVI	p	z	Holm
Davies–Bouldin	0.002	3.164	0.025
BD-Dunn	0.837	0.205	0.050

The effectiveness statistical analysis using Aligned Friedman test is shown in Table 7. Aligned Friedman was used because the test is applied to a dataset with less than 5 features. As the ranking shows, BD-Silhouette was in the first position with 35.17, followed by BD-Dunn with 36.51 and Davies–Bouldin with 55.80 in the last position.

The statistic for Aligned Friedman was 20.45 according to a Chi-square distribution with 2° of freedom. The p value for Aligned Friedman was 0.0 and lower than 0.05. Therefore, the null hypothesis was rejected that they all behaved in a similar way with a level of significance of $\alpha = 0.05$.

Post hoc testing was applied because the null hypothesis that was rejected. Table 8 shows the p values, z value and Holm's α, using BD-Silhouette as the control CVI since it obtained the best ranking. Holm's procedure rejects those hypotheses that have a p value ≤ 0.05.

In execution time statistical analysis, Aligned Friedman test is shown in Table 9. As the ranking shows, BD-Dunn was in the first position with 30.12, followed by BD-Silhouette with 33.44 and Davies–Bouldin with 63.92 in the last position.

The statistic for Friedman was 20.214, distributed according to a Chi-square distribution with 2° of freedom. The p value for Friedman was 0.0 and lower than 0.05. Therefore, the null hypothesis was rejected (that they all behaved in a similar way) with a level of significance of $\alpha = 0.05$.

Table 9 Sorted ranking of execution time BD-CVI for Aligned Friedman test

BD-CVI	Ranking
BD-Dunn	30.12
BD-Silhouette	33.44
Davies–Bouldin	63.92

Table 10 Post hoc analysis using Holm's procedure and BD-Dunn as the control algorithm

BD-CVI	p	z	Holm
Davies–Bouldin	0.000	5.1852	0.025
BD-Silhouette	0.6104	0.5095	0.050

Table 10 shows the p values, z value and Holm's α, using Dunn as the control CVI since it obtained the best ranking. Holm's procedure rejects those hypotheses that have a p value ≤ 0.0083.

4.5 Discussion

Experimental results show that BD-CVIs may be used to provide the optimal number of clusters of large datasets. In this paper, BD-Silhouette and BD-Dunn have achieved better results in lower time than the rest of the indices.

Results show that finding the optimal number of clusters is not a trivial task. There were some datasets that were not correctly solved. The results in this study indicate that Silhouette, Dunn and Davies–Bouldin were the CVIs with the highest success rate. This fact is particularly significant because it helps to construct new CVIs that are suitable to work with Big Data.

This study also found that BD-CVIs had even more difficult to provide the optimal number of cluster of a dataset. The results of this study indicate that there were complex datasets where no BD-CVI correctly predicted the optimal number of clusters. All these support the notion that getting the optimal number of clusters is not a minor task. However, the results of this study show that BD-Silhouette and BD-Dunn are good choices to predict the optimal number of clusters as the results were promising.

In terms of time, traditional indices last so much time compared with BD-CVI. The results of this study indicate that the biggest dataset used with traditional indices last more than a day; however, BD-CVIs in the same dataset lasted less than 1 minute. These observations provide evidence that suggests that the use of traditional indices is very limited due to the size of the datasets.

5 Conclusions

In this paper, two novel CVIs implemented in Spark have been proposed to be applied in datasets considered as Big Data. The proposed indices are based on Silhouette and Dunn indices, but modified and optimized to deal with Big Data.

The experimental study indicates that our Big Data indices are very competitive. We have tested its effectiveness and time execution with datasets of different sizes (different number of clusters and different number of instances). The main achievements obtained are the following:

- Two clustering indices based on traditional Silhouette and Dunn indices.
- BD-Silhouette and BD-Dunn has allowed us to estimate the optimal number of clusters of datasets that may be considered Big Data.
- Computational time of these indices is drastically reduced compared with traditional indices.
- The size of the dataset does not directly influence to the effectiveness of the BD-CVIs.
- The software of this contribution can be found as a spark-package at http://spark-packages.org/package/josemarialuna/clusterIndices.
- The source code of these indices can be found at https://github.com/josemarialuna/ClusterIndices.

As a future work, we intend to include some approaches to other CVIs that also obtained suitable results in their traditional version. Further research is needed to study the outcomes because some of the BD-CVI results were not enough clear. It would also be useful to explore the results of our BD-CVIs with no round-shape clusters datasets. Additionally, it would be also interesting to research the results of BD-CVIs taking into consideration using the inter-cluster distances between the centroids instead of using the global centroid.

Acknowledgements 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.

Appendix A: Datasets generation

In this work, we needed suitable datasets to cluster the data and to know in advance how many clusters have them. We have developed an application that generates especially designed datasets with predefined number of clusters. To make sure that the clusters are well formed and separated, the points of the datasets follow a normal distribution with different mean values and a low standard deviation. Datasets are generated introducing as input parameters the following items: the total number of clusters of the dataset, the number of instances per cluster, the number of features of the instances and the standard deviation. As we mention before, feature values of data in clusters follow a normal distribution, and for this purpose, data is randomly generated with the given standard deviation and the mean, that by default is 0.25 or 0.75. With this random generation of points, we ensure that clusters are well separated and it will make easier to be identified by the CVIs.

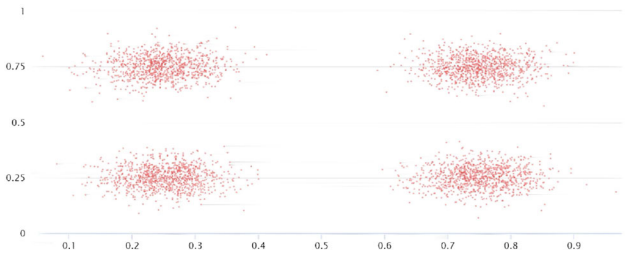

Fig. 7 Generated dataset with 4 clusters and 2 features generated with a mean of 0.25 and 0.75 and a standard deviation of 0.05

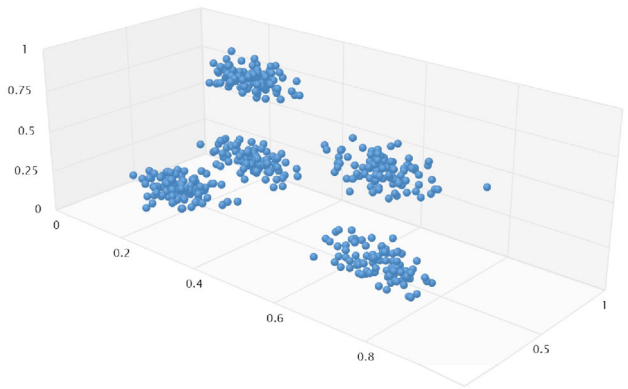

Fig. 8 Generated dataset with 5 clusters and 3 features with a mean of 0.25 and 0.75 and a standard deviation of 0.05

Figures 7 and 8 illustrate two basics example datasets that were generated by the application. Those figures show the distribution of the points in 2D and 3D. These datasets, and those generated for the experiments, were created with an average of 0.25 and 0.75, and a standard deviation of 0.05. Figure 7 corresponds to a dataset with 2 features and 4 clusters using 1000 instances per cluster. As it can be seen, in this figure there are 4 clear groups of points that correspond with the clusters. There are also some points that are not close to a big cloud of points and this is due to data points are randomly generated following a normal distribution a its standard deviation is 0.05.

A similar situation is found in Fig. 8. This figure is a 3D representation of a dataset with 3 features where 5 clusters can be easily identified. Each cluster counts with 100 instances and, as it happened in Fig. 7, there are also some points that are separated from the central cloud of points.

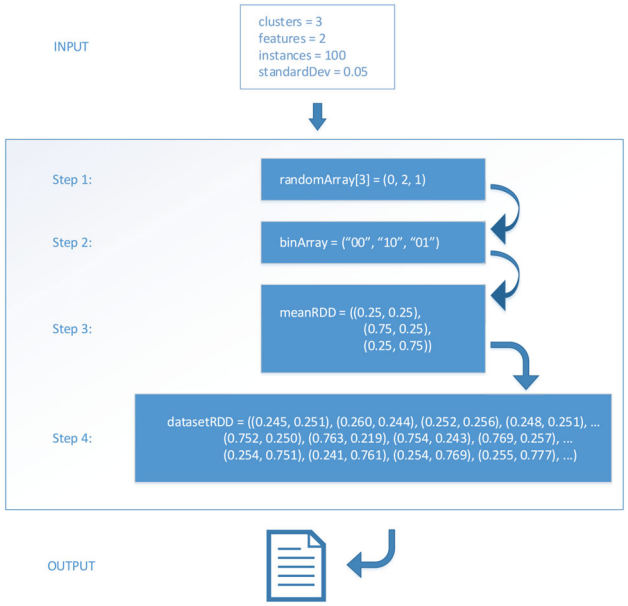

Fig. 9 Flowchart of dataset generator algorithm

Figure 9 shows graphically step by step how the application generates a dataset. The figure shows the input of the algorithm represented by a white box at the top, the application with the steps represented by a darker box, and the output file at the bottom of the figure. In our example the application receives the next input parameters: 3 clusters, 2 features, 100 instances per cluster, and a standard deviation of 0.05.

1. *Step 1* the application receives the input parameters and an array named $randomArray$, whose size is the number of clusters. $randomArray$ is randomly generated with the numbers in the interval [0, $numberofclusters$) without repetition. In the figure is shown that $randomArray$ has size 3, and the random included numbers are (0, 2, 1).
2. *Step 2* the values of $randomArray$ are parsed to binary, and they are saved into an array named $binArray$. In our example, $randomArray$ was (0, 2, 1), so $binArray$ becomes (00, 10, 01).
3. *Step 3* $meanRDD$ is an RDD object that takes its values from $binArray$. The values of the array are individually taken, and if it is 0, it sets 0.25; or if it is 1, it sets 0.75. In our example, "00" becomes (0.25, 0.25), "10" becomes (0.75, 0.25), and "01" becomes (0.25, 0.75).
4. *Step 4* on this step, the values of each data object in the dataset are generated and saved into an RDD object. It generates $instances$ value random numbers following a normal distribution with the standard deviation given as input parameter ($standardDev$), and the mean is set by the value of $meanRDD$. Each value of $meanRDD$ will be the data objects of each cluster. In our example, the application will generate 100 data objects with and a standard deviation of 0.05, and a mean of 0.25 for the first feature, and a 0.25 for the second feature ((0.245, 0.251), (0.260, 0.244), (0.252, 0.256)…). The data objects of the second clusters take (0.75, 0.25) as the values for the mean and generate the following data objects: ((0.752, 0.250), (0.763, 0.219), (0.754, 0.243)…). And the third cluster has the following data objects: ((0.254, 0.751), (0.241, 0.761), (0.254, 0.769)…).
5. Once $datasetRDD$ is built, the application saves the data into an output file.

The source code of this application can be found at [24].

References

1. Abdi, A., Hassanzadeh, Y., Ouarda, T.: Regional frequency analysis using Growing Neural Gas network. J. Hydrol. **550**, 92–102 (2017)
2. Alok, A., Saha, S., Ekbal, A.: Semi-supervised clustering for gene-expression data in multiobjective optimization framework. Int. J. Mach. Learn. Cybern. **8**(2), 421–439 (2017)
3. Berikov, V., Pestunov, I.: Ensemble clustering based on weighted co-association matrices: error bound and convergence properties. Pattern Recognit. **63**, 427–436 (2017)
4. Boone, C., Skipper, J., Hazen, B.: A framework for investigating the role of big data in service parts management. J. Clean. Prod. **153**, 687–691 (2017)
5. Calinski, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat. Theory Methods **3**(1), 1–27 (1974)
6. Chen, W.-Y., Song, Y., Bai, H., Lin, C.-J., Chang, E.Y.: Parallel Spectral Clustering in Distributed Systems. IEEE Trans. Pattern Anal. Mach. Intell. **33**(3), 568–586 (2011)
7. Daki, H., El Hannani, A., Aqqal, A., Haidine, A., Dahbi, A.: Big Data management in smart grid: concepts, requirements and implementation. J. Big Data **4**(1), 13 (2017)
8. Davies, D., Bouldin, D.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI–1**(2), 224–227 (1979)
9. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
10. Dubes, R., Jain, A.K.: Clustering techniques: the user's dilemma. Pattern Recognit. **8**(4), 247–260 (1976)
11. Dunn, J.: Well-separated clusters and optimal fuzzy partitions. J. Cybern. **4**(1), 95–104 (1974)
12. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A .Y., Foufou, S., Bouras, A.: A survey of clustering algorithms for big data: taxonomy and empirical analysis. IEEE Trans. Emerg. Top. Comput. **2**(3), 267–279 (2014)
13. Gallos, L., Korczyński, M., Fefferman, N.: Anomaly detection through information sharing under different topologies. Eurasip J. Inf. Secur. **1**, 2017 (2017)
14. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google File System, vol. 37, pp. 29–43. ACM Press, New York, USA (2003) **(cited By 2613)**
15. Han, J., Kamber, M., Pei, J.: Cluster analysis: basic concepts and methods. In: Data Mining: Concepts and Techniques, pp. 443–495. Elsevier, USA (2012)
16. Hennig, C., Liao, T.: How to find an appropriate clustering for mixed-type variables with application to socio-economic stratification. J. R. Stat. Soc. Ser. C Appl. Stat. **62**(3), 309–369 (2013)
17. Holmes, G., Donkin, A., Witten, I.: WEKA: a machine learning workbench. In: Proceedings of ANZIIS '94—Australian New

Zealnd Intelligent Information Systems Conference, Number JANUARY 1994, pp. 357–361. (1994)
18. Jacques, J., Preda, C.: Functional data clustering: a survey. Adv. Data Anal. Classif. **8**(3), 231–255 (2014)
19. Jain, A .K.: Data clustering: 50 years beyond K-means. Pattern Recognit. Lett. **31**(8), 651–666 (2010)
20. Jerome, R .B., ätönen, K .H.: Anomaly detection and classification using a metric for determining the significance of failures. Neural Comput. Appl. **28**(6), 1265–1275 (2017)
21. Jinyin, C., Xiang, L., Haibing, Z., Xintong, B.: A novel cluster center fast determination clustering algorithm. Appl. Soft Comput. **57**, 539–555 (2017)
22. Kim, J., Lee, W., Song, J .J., Lee, S.-B.: Optimized combinatorial clustering for stochastic processes. Clust. Comput. **20**(2), 1135–1148 (2017)
23. Lord, E., Willems, M., Lapointe, F.-J., Makarenkov, V.: Using the stability of objects to determine the number of clusters in datasets. Inf. Sci. **393**, 29–46 (2017)
24. Luna-Romera, J.M.: Clustering Synthetic Big Datasets Generator. https://github.com/josemarialuna/CreateRandomDataset (2017). Accessed 20 July 2017
25. Mazinan, A.: On cluster validity indices with its application to interleaved radar pulse separation through fuzzy-based representation. Evol. Syst. **7**(4), 243–254 (2016)
26. Miller, Z., Dickinson, B., Deitrick, W., Hu, W., Wang, A.H.: Twitter spammer detection using data stream clustering. Inf. Sci. **260**, 64–73 (2014)
27. Mohammed, A.J., Yusof, Y., Husni, H.: Fireflyclust: an automated hierarchical text clustering approach. Jurnal Teknologi, **79**(5), 11–22 (2017)
28. Parejo, J.A., Garcia, J., Ruiz-Cortes, A., Riquelme, J.C.: Statservice: Herramienta de análisis estadistico como soportepara la investigacion con metaheuristicas. In: Actas del VIII Congreso Expañol sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados. Albacete, España (2012)
29. Perez-Chacon, R., Talavera-Llames, R., Martinez-Alvarez, F., Troncoso A.: Finding Electric Energy Consumption Patterns in Big Time Series Data. In: Omatu, S., et al. (eds.) Distributed Computing and Artificial Intelligence, 13th International Conference. Advances in Intelligent Systems and Computing, vol. 474, pp. 231–238. Springer, Cham (2016)
30. Rousseeuw, P.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**(C), 53–65 (1987)
31. Rumson, A .G., Hallett, S .H., Brewer, T .R.: Coastal risk adaptation: the potential role of accessible geospatial Big Data. Mar. Policy **83**, 100–110, (2017)
32. Sagi, T., Gal, A., Barkol, O., Bergman, R., Avram, A.: Multi-source uncertain entity resolution: transforming holocaust victim reports into people. Inf. Syst. **65**, 124–136 (2017)
33. Sevilla-Villanueva, B., Gibert, K., ànchez-Marrè, M.S.: Using CVI for Understanding Class Topology in Unsupervised Scenarios, pp. 135–149. Springer, Cham (2016)
34. Spark, A.: Apache Spark, Lightning-Fast Cluster Computing. https://spark.apache.org/ (2017). Accessed 20 June 2017
35. Spark, A.: MLlib is Apache Spark's Scalable Machine Learning Library. https://spark.apache.org/mllib/ (2017). Accessed 20 June 2017
36. Tong, Q., Li, X., Yuan, B.: A highly scalable clustering scheme using boundary information. Pattern Recognit. Lett. **89**, 1–7 (2017)
37. Yang, M., Mei, H., Huang, D.: An effective detection of satellite images via k-means clustering on hadoop system. Int. J. Innov. Comput. Inf. Control **13**(3), 1037–1046 (2017)
38. Zaharia, M.. Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Presented as Part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 15–28, San Jose, CA, USENIX (2012)
39. Zhang, Q., Yang, L.T., Chen, Z., Li, P.: High-order possibilistic c-means algorithms based on tensor decompositions for big data in IoT. Inf. Fusion **39**, 72–80 (2018)
40. Zhang, R., Xu, C., Duan, Z.: Novel antigenic shift in HA sequences of H1N1 viruses detected by big data analysis. Infect. Genet. Evol. **51**, 138–142 (2017)