Article

Distributed Correlation-Based Clustering Mechanism for Large-Scale Datasets

Kuei-Sheng Lee *, Meng-Feng Tsai, Chi-Sheng Huang

Department of Computer Science and Information Engineering, National Central University, No. 300, Zhongda Rd., Zhongli District, Taoyuan City 32001, Taiwan, ROC

* Correspondence: leegueishen@gmail.com

Abstract: In the field of machine learning, cluster analysis has always been a very important technology for determining useful or implicit characteristics in the data. However, the current mainstream cluster analysis algorithms require comprehensive analysis of the overall data to obtain the best parameters in the algorithm. As a result, handling large-scale datasets would be difficult. This research proposes a distributed related clustering mechanism for Unsupervised Learning, which assumes that if adjacent data are similar, a group can be formed by relating to more data points. Therefore, when processing data, large-scale datasets can be distributed to multiple computers, and the correlation of any two datasets in each computer can be calculated simultaneously. Later, results are processed through aggregation and filtering before assembled into groups. This method would greatly reduce the pre-processing and execution time of the dataset; in practical application, it only needs to focus on how the relevance of the data is designed. In addition, the experimental results show the accuracy, applicability, and ease of use of this method.

Keywords: Big Data; Clustering; Distributed system; Machine learning

1. Introduction

Clustering algorithms are widely used in statistical analysis or data processing. Their main goal is to divide similar datasets into the same group. A dataset suitable for clustering is a collection of points, which are objects belonging to some space. All spaces for which we can perform a clustering have a distance measure, giving a distance between any two points in the space. Although the goal is obvious, different approaches exit for clustering similar datasets. In general, in the clustering algorithm, the overall dataset would be examined and the parameters of the algorithm would be determined (such as the number of clusters [1,2], range of each processing, density within the range [3,4], and distance between datasets) to use unsupervised learning methods for clustering. Although it is unsupervised learning, when setting the parameters, it is actually necessary to observe the test results and adjust the parameters repeatedly to achieve the ideal clustering. In certain perspectives, the experimenter plays the role of the supervisor.

Research has been performed on the concept of more effective clustering for dealing with a large amount of data. When processing large-scale data clustering, the following goals must be achieved:

1. Avoid preceding tasks such as sorting of the data before clustering.
2. Avoid configuring difficult parameters such as the number of clusters or density before clustering.
3. Be able to cluster non-centroid (irregular) clusters.
4. Be able to apply a distributed framework to accelerate the processing of large-scale datasets.

Based on the above-mentioned research purposes, this study provides a related clustering mechanism that can be applied to large-scale datasets in a distributed structure. The users do not need to perform complex pre-processing of data when clustering, but only need to focus on a distance between any two points. This research can be applied to fields such as astronomical data processing, social network analysis, or health insurance medical records, providing an effective method for large-scale unknowable data clustering, so that the purpose of data exploration and data mining is truly achieved.

2.Method

The basic idea of this research is simple to understand. Adjacent data in a group must be similar to each other. When the data are expanded and linked based on this feature, a complete cluster can be assembled until there are no similar data. The similarity characteristics between datasets can be easily determined as long as there is a small cluster of samples. However, when this idea is applied to large-scale dataset, the problem of "combinatorial explosion" will occur [5]. Since we assume that all the data are not sorted and categorized, it is impossible to know whether the data are adjacent and can be used for similarity test calculations. Combining all the datasets for calculation is a feasible method; however, in this method, the large-scale data to be processed originally will be expanded into a larger or even more complex dataset. If the following combination Formula (1) is used for calculating, after combining the original 1000 data, a dataset of 500,000 will need to be processed.

$$C_n^m = \frac{m!}{n!(m-n)!} \tag{1}$$

Therefore, we propose a clustering method that allows large-scale data to be processed using a parallel computing model that quickly processes and obtains clustering results. This method will be executed according to the following steps, as shown in Figure 1:

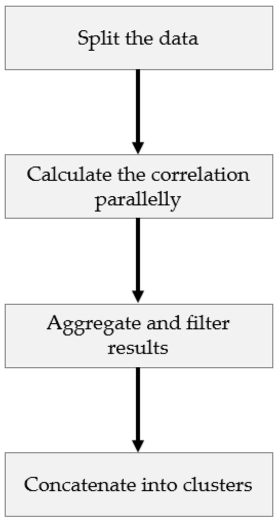
Split the data
Calculate the correlation parallelly
Aggregate and filter results
Concatenate into clusters

Figure 1. The method flow diagram

2.1 Splitting the data

To apply parallel processing to large-scale datasets from distributed computers, it is necessary to consider that each cluster of data processed parallelly cannot have its own uniqueness; it must be able to adapt to current computer resources, and the data volume of each cluster can be adjusted arbitrarily [6,8]. The method adopted here to split the data is to use the initial point filtering method in the K-means++ algorithm for determining the centroid

[9]. Furthermore, the most important parameter K in the K-means++ algorithm can be set as a multiple of the total number of computers in the distributed framework. If there are 3 computers that can be processed parallelly, the K value can be set to 3, 6, 9, or even larger. The focus of obtaining the value is to allow all computers to process parallelly without any of them going into the idle state.

Because each initial point of the K-means++ algorithm would be dispersed and move as far away as possible, it would help to evenly divide the large-scale dataset into each computer processing parallelly. After the K value has been determined, a calculation with the K-means++ algorithm can be performed to determine the initial point (the centroid) to which all the datasets belong and record the distance between neighboring data points and the centroid during the calculation process. This is because if the data are simply split by the K-means++ algorithm, they would lose the transitive closure between the clusters and let this K value become the final number of clusters.

However, there is an intersection between clusters we need to split and other clusters; hence, there is the opportunity to associate the data processed parallelly. Therefore, to divide the data, the farthest distance of all data points from the centroid is used; then, the displacement value recommended by the Cure algorithm is increased by 20% [10]. Furthermore, each centroid is allowed to cover all the data within this length range, while allowing the data to overlap, as shown in Figure 2.

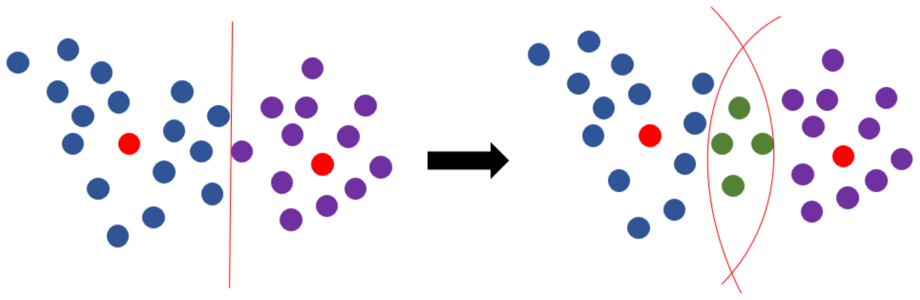

Figure 2. Changing from clustering close to the centroid to extending from the longest distance

2.2 *Calculate the correlation parallelly*

Here, we present the most important setting of this method, which is to regulate the distance between any two data to comply with the principle of similarity. Here, the Euclidean distance (Formula 2) is used to calculate the distance between any two data points, so the value d needs to be set. When the distance between any two points is within the length d, the two data points are considered to be similar.

$$d_{12} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \tag{2}$$

In the calculation process, we also determine the secondary memory mechanism that is often used for large-scale dataset processing. In the secondary memory, we place the index of each data point, as well as its x-axis and y-axis values, so that each data point can be quickly called and calculated by index. In addition, the results associated with each data point in the secondary memory can be summarized. If the index value of the data is the key value, the value is the index collection that is correlated to those data, as shown in Figure 3. The second memory can be a relational database or NoSQL database.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0010.v1

index	axis
19	0.95415778,4.39985544
54	-1.57480456,2.84532424
211	-0.83380144,8.13041771
173	0.09985625,7.79849768
250	-1.74736532,2.77770451

index	correlation
277	173, 213, 229, 250
250	229, 277
285	240, 242
242	285, 15, 290
240	242

Figure 3. Secondary memory mechanism

2.3 *Aggregate and filter results*

After the parallel calculation, all the correlation results in this dataset are obtained. However, to avoid the unreasonable state of data combining of the certain two clusters into one cluster (see for example Figure 4) simply due to a single linear correlation, we will set a threshold and filter it. When a certain point x has only two similar points y and z, and y and z also have only two similar points (including x), then x should be filtered and excluded.

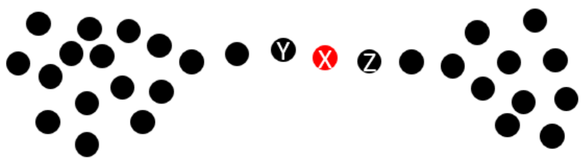

Figure 4. Remove the state of linear connection

2.4 *Concatenate into clusters*

Finally, each cluster of similar results needs to be concatenated into clusters. Each cluster has its index objects that are correlated. If two clusters have an intersection of index objectes, they can be concatenated into a new cluster, as shown in Figure 5. The calculation is recursively repeated until the final number of clusters is constant. This is actually the clustering method of hierarchical clustering, which continuously concatenates small clusters into large clusters in a hierarchical structure [11,12]. When the concatenation is completed, the real clustering results will be obtained.

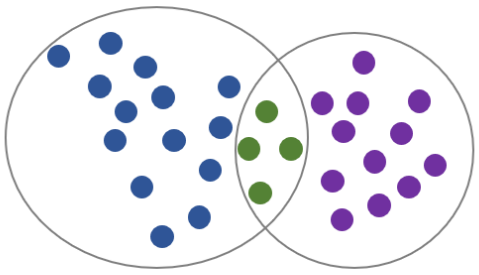

Figure 5. The intersection of two Index objects

With the steps mentioned above, a large dataset can be reasonably divided to facilitate parallel calculations. Then, through the results of parallel calculation, the data are aggregated and filtered to obtain the final clustering result.

3.Experiment

To implement the proposed method, we used Scikit-learn (also known as scikits.learn or sklearn) to produce test data. Scikit-learn is a machine learning package used by Python's

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 November 2020 doi:10.20944/preprints202011.0010.v1

programming language. In the experiment, we generated three types of data. Each data sample consists of 1000 data, mixed with 5% of noise. The relevant parameters for data forming are shown in Table 1:

Table1. Data parameters produced by sklearn.datasets

Data Type	Number	Parameter	noise
blobs	1000	centers = 4, cluster_std = 0.5, random_state = 0	
circles	1000	factor = 0.5	0.05
moons	1000		0.05

Based on the above parameters, various datasets were generated. Among these, the dataset of blobs formed into four obvious clusters. The dataset of circles was a circle with the outer circle covering the inner circle, divided into two clusters. Finally, moons were generated by two arc-shaped clusters, interlaced with each other. The appearance of the test data clusters is shown in Figure 6.

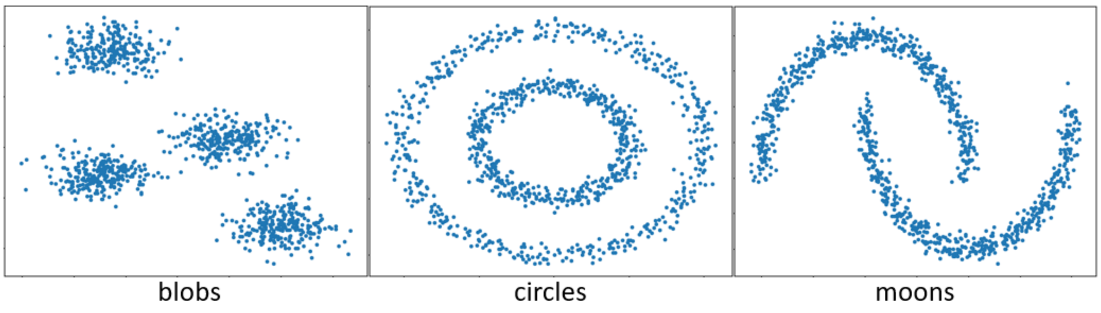

Figure 6. Uncluttered test data

To perform distributed parallel calculation, all data must be split. Here, they are split into 9 zones (Figure 7) and 3 zones (Figure 8). The split data can be configured to different computers for parallel calculation using the same calculation method to obtain faster performance, which is also a key point when processing large-scale datasets.

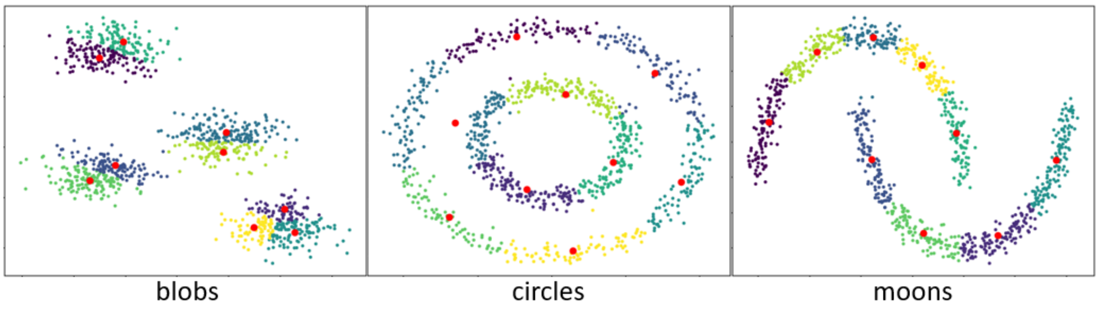

Figure 7. Nine centroid division status

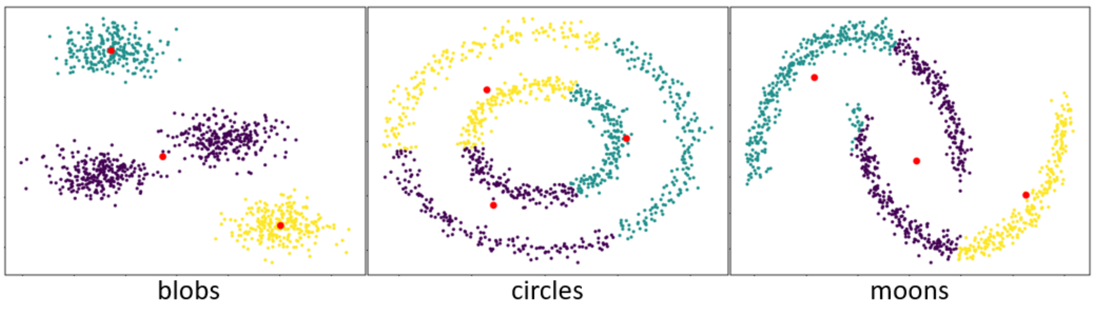

Figure 8. Three centroid division status

After the parallel calculation was completed, the correlated data were concatenated transitively to form clusters. The clustering diagram after the experiment is shown in Figure 9 and the actual number of clusters is the same as the default number of clusters of the original data, as shown in Table 2.

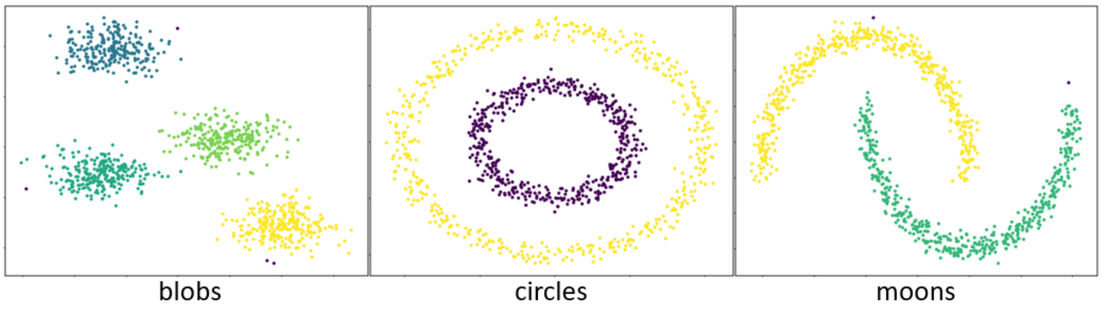

Figure 9. Actual clustering **results**

Table 2. Experiment results

Data type	Numbers	Original clusters	Cluster results
blobs	1000	4	4
circles	1000	2	2
moons	1000	2	2

For the experiment, we used a desktop computer with Intel 16G RAM, and an Intel Core i7-4510U CPU on a Microsoft Windows 10 Professional Operational System, and Python-based programs. The difference between stand-alone processing and parallel processing is already obvious in the data splitting process. Table 3 lists the execution time of splitting under different numbers for blobs data.

Table 3. Experiment performance

Numbers	Split numbers	Execution time (seconds)
1000	1	2
1000	9	1
2000	1	7
2000	9	2

3000	1	15
3000	9	6

The experimental results indicate that this method can indeed cluster correctly, and under the condition of parallel calculation, the processing efficiency is greatly improved.

4.Related Work

Many related methods have been developed for clustering; more common algorithms related to the research are as follows:

4.1 K-means++

K-means++ algorithm is improved from K-means. The K-means algorithm randomly selects the number of data points as the centroid of the cluster at the beginning based on the number of clusters. Subsequently, in each iteration of the calculation, the data closest to the current cluster centroid is calculated, which becomes the result of the current cluster; then, the new centroid for each cluster is recalculated. This method is iterated repeatedly until convergence, and finally the centroid become the centroid of the cluster results. In comparison with K-means, which randomly selects the centroid of the cluster, K-means++ selects one center point at a time at the start, and each time the selection is done, the data farther from the center point would have a higher probability to be the next center point. This allows the initial centroid to be more evenly distributed. When the selected initial centroid can be more evenly distributed, the number of iterations required by the algorithm would be effectively reduced.

4.2 Mini-batch K-Means

Instead of using the full dataset at each iteration, the algorithm is capable of using mini-batches, moving the centroids just slightly at each iteration. This reduces computation cost by orders of magnitude compared to the classic batch algorithm while yielding significantly better solutions than online stochastic gradient descent [13].

4.3 DBSCAN

DBSCAN is an algorithm that does not rely on clustering centroids and does not need to set the number of clusters. The principle is to check whether the data in the specific range meet the standard density. If the data in the specific range meet the standard density, the adjacent data can be continuously extended and expanded. With this feature, DBSCAN can continue to cluster data with the same density; therefore, any shape can be clustered and noise data in the cluster can be eliminated simultaneously.

4.4 CURE

The CURE algorithm is a clustering method using representative points. The algorithm first treats each data point as a cluster, and then merges the closest clusters until the number of clusters reaches the required number. The CURE algorithm would not use all points or the centroid and radius to represent a cluster but select a fixed number of uniformly distributed points from each cluster as representative points to describe the cluster. Furthermore, these points are multiplied by an appropriate shrinkage factor. In each iteration of the calculation, they are brought closer to the centroid of the cluster. With this feature, the cluster can be extended to show clusters of any shape.

4.5 Agglomerative clustering

A hierarchy of clusters is built from the bottom up, at each iteration agglomerative clustering connects the nearest pair of clusters. This approach scales very well to large numbers of instances or cluster, it can capture clusters of various shapes, it produces a flexible and informative cluster tree instead of forcing you to choose a particular cluster scale, and it can be used with any pairwise distance. If we provide a connectivity matrix, a sparse m by m matrix that indicates which pairs of instances are neighbors, it can scale to large numbers of instances.

4.6 Birch

Birch (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm was designed specifically for very large datasets, and it can be faster than batch K-Means, with similar results, as long as the number of features is not too large (<20). It builds a tree structure during training containing just enough information to quickly assign each new instance to a cluster, without having to store all the instances in the tree. this allows it to use limited memory, while handle huge datasets.

5.Conclusion

We proposed a method to process large-scale datasets in a distributed clustering manner and verified it through experimental results. In the entire method, only two parameters are required. One parameter is a factor that determines the similarity between the data, such as by checking whether the distance between two data is close enough. The other parameter is the number of distributed processing and it does not affect the results of the experiment, because it is only configured to match the computer conditions of the current experimental environment. At the same time, it resolves the conflicts between the distribution and transitive closures in parallel calculation and splitting. By appropriately expanding the range of splitting, the relationship between parallel calculation and splitting can be maintained to continue the transitive closure of the overall data. Finally, the parallel processing results were collected and excellent experimental data could be obtained.

Like most clustering algorithms, this research is still limited by the curse of dimensionality because high dimensional data will invalidate the distance gap between data. Therefore, if the original dataset is of high dimensionality, it is necessary to consider processing the original data in a reduced dimensionality method before applying this clustering method [14,17].

In addition, because this research is based on the similarity of data on the theoretical basis, it cannot process clustering of different densities, which implies that if there are clusters of different densities in the dataset, this method would not be applied for clustering. This is the biggest limitation of this research.

This research has proposed a simple clustering method that can be implemented on a distributed system. Furthermore, because of the characteristics of the method, the clusters of various shapes can be processed. In our experiment, the traditional two-dimensional dataset is used to check whether the calculated distance between the two points is close enough. However, according to the spirit and principle of this research, it will not be limited to processing two-dimensional data suitable for the Euclidean distance. This method can be applied to any two datasets that can be distinguished to analyze their similarity as well as to other fields.

References

1. Yuan, Chunhui, and Haitao Yang. "Research on K-value selection method of K-means clustering algorithm." J—Multidisciplinary Scientific Journal 2.2 (2019): 226-235.
2. Jain, Anil K. "Data clustering: 50 years beyond K-means." Pattern recognition letters 31.8 (2010): 651-666.

3. Price-Jones, Natalie, and Jo Bovy. "Blind chemical tagging with DBSCAN: prospects for spectroscopic surveys." Monthly Notices of the Royal Astronomical Society 487.1 (2019): 871-886.
4. Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd. Vol. 96. No. 34. 1996.
5. Henard, Christopher, et al. "Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines." *IEEE Transactions on Software Engineering* 40.7 (2014): 650-670.
6. Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.
7. Hosseini, B.; Kiani, K. A Robust Distributed Big Data Clustering-based on Adaptive Density Partitioning using Apache Spark. Symmetry 2018, 10, 342.
8. Gao, X.; Yang, M. Understanding and Enhancement of Internal Clustering Validation Indexes for Categorical Data. Algorithms 2018, 11, 177.
9. Arthur, David, and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Stanford, 2006.
10. Guha, Sudipto, Rajeev Rastogi, and Kyuseok Shim. "CURE: an efficient clustering algorithm for large databases." ACM Sigmod record 27.2 (1998): 73-84.
11. Zhang, Hao, et al. "In-memory big data management and processing: A survey." IEEE Transactions on Knowledge and Data Engineering 27.7 (2015): 1920-1948.
12. Xu, Qin, et al. "Efficient synthetical clustering validity indexes for hierarchical clustering." Expert Systems with Applications (2020): 113367.
13. Sculley, David. "Web-scale k-means clustering." Proceedings of the 19th international conference on World wide web. 2010.
14. Day, William HE, and Herbert Edelsbrunner. "Efficient algorithms for agglomerative hierarchical clustering methods." Journal of classification 1.1 (1984): 7-24.
15. Pearson, Karl. "LIII. On lines and planes of closest fit to systems of points in space." The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2.11 (1901): 559-572.
16. Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller. "Kernel principal component analysis." International conference on artificial neural networks. Springer, Berlin, Heidelberg, 1997.
17. Roweis, Sam T., and Lawrence K. Saul. "Nonlinear dimensionality reduction by locally linear embedding." science 290.5500 (2000): 2323-2326.