PARALLEL MULTI-VIEW CONCEPT CLUSTERING ALGORITHM BASED ON SPARK

Xiaoming JIANG[1], Huamin YANG[2], Yuan REN[3,*], Zeming DU[4], Qingzong LIU[5]

To address the explosive growth of data in today's big data era, the increasing prevalence of multi-view and multi-modal data presents significant challenges for traditional stand-alone clustering algorithms. In recent years, with the continuous development of computer hardware, the popularization of cloud computing technology, and the reduction of storage costs, the parallel transformation of clustering algorithms is a way to face the above challenges. Conceptual decomposition, as an extension of non-negative matrix factorization (NMF), retains its advantages while removing the non-negativity constraint, thereby expanding its applicability. This paper uses a multi-view clustering algorithm based on concept decomposition, combined with Spark, the current mainstream parallel computing framework, to implement a Spark-based multi-view conceptual clustering parallel method, which improves the operating efficiency of the algorithm without reducing the accuracy rate. Experimental results validate the performance and efficiency of the proposed parallel method, demonstrating its suitability for big data clustering analysis.

Keywords: Cluster analysis, Multi-view clustering, Conceptual factorization, Parallel computing, Spark

1. Introduction

Due to the increase of data acquisition technologies (such as sensors, cameras, web crawlers, etc.) and data formats (e.g., text files, database files, structured files, HTML files, etc.) in recent years, multi-modal data has become increasingly common. Matrix factorization algorithms have been widely used in machine learning, data mining, and other fields, having the advantages of high efficiency, easy implementation, and strong scalability [1]. NMF (Non-negative matrix factorization) has been applied in the field of multi-view clustering and has obtained high performance [2]. However, NMF requires that each element in the data set must be non-negative, which largely limits its usage. To solve this limitation, researchers have proposed the concept factorization algorithm (Concept

[1] Changchun University of Science and Technology, Changchun, China
[2] Changchun University of Science and Technology, Changchun, China
[3] Changchun University of Science and Technology, Changchun, China, email: 1215749538@qq.com
[4] Changchun University of Science and Technology, Changchun, China
[5] Changchun University of Science and Technology, Changchun, China

Factorization, commonly referred to as CF algorithm). Inheriting the advantages of NMF algorithms, the CF algorithm removes the constraint on the non-negativity of the data set and extends the application of matrix factorization in the multi-view clustering field. Therefore, the CF algorithm has gradually been attention paid by domestic and foreign researchers. As the era of big data comes, the data volume is increasing with the rapid development of various fields. This has brought new challenges to the study of cluster analysis. To address the big data clustering problem, a more effective approach is to integrate distributed technology and put forward new clustering algorithms, or to transform existing traditional serial clustering algorithms. Many parallel clustering algorithms have been proposed in recent years [3]. Most of these algorithms are implemented based on the current mainstream distributed computing framework, such as Hadoop MapReduce and Spark. Spark is a distributed computing engine based on memory [4]. It is a MapReduce computing model as well as Hadoop MapReduce, but Spark can store intermediate calculation results in memory and reduce the program-to-disk I/O interaction, so Spark running efficiency is far higher than Hadoop MapReduce [5]. Because of its high efficiency and flexibility, Spark is more and more widely used in the field of multi-view clustering.

In this paper, based on the concept decomposition algorithm, a multi-view concept clustering algorithm based on concept decomposition is proposed. Then, the algorithm is applied in the Spark distributed computing platform to achieve a parallel approach based on Spark for multi-view concept clustering. The paper also conducts experiments on the proposed method to verify its effectiveness.

2. Concept Decomposition Basic Model

The main idea of concept decomposition is the same as that of non-negative matrix decomposition, for a given data matrix $X \in \mathbb{R}^{d \times n}$ (n is the number of samples in the data set, d is the feature dimension of the sample). It is necessary to find two matrices so that the multiplication result is infinitely close to the original data matrix X[6]. The objective function is expressed as:

$$X \approx XUV^T \tag{1}$$

Among them, $U \in \mathbb{R}^{d \times r}$ is the basis matrix, which is the coordinate basis in the new space. $V \in \mathbb{R}^{r \times n}$ is the coefficient matrix to replace the original data matrix for subsequent processing [7]. In addition, the loss function describing the approximation before and after concept decomposition is defined as:

$$\min_{U,V} \mathcal{O}_{CF} = ||X - XUV^T||_F^2 \tag{2}$$

Using an iterative multiplicative update algorithm to solve the objective function (2), the results are as follows:

$$u_{ij}^{t+1} \longleftarrow u_{ij}^{t+1} \frac{(KV)_{ij}}{(KUV^T V)_{ij}}$$
$$v_{ij}^{t+1} \longleftarrow v_{ij}^{t+1} \frac{(KU)_{ij}}{(VU^T KU)_{ij}} \tag{3}$$

In the formula, t means iteration t. It is a linear kernel function, any other kernel function can be used instead, such as polynomial kernel, Gaussian kernel, etc[8].

When concept decomposition is actually applied to cluster analysis scenarios, the final clustering results can be obtained through the following two methods:

1) Specifies the number of clusters in the data set, let r=k, at this time, the subscript of the maximum value of each row vector element in the mapping matrix V is the cluster label of the clustering result;

2) Use other clustering algorithms to cluster the mapping matrix H.

The method proposed in this paper is to output the clustering results directly in the mapping matrix according to method (1).

3. Multi-View Concept Clustering

The execution of concept decomposition is a linear decomposition process. Therefore, the calculated new space can preserve the global characteristics of the original data, while the original local characteristics may be lost[9]. In order to preserve the original local characteristics of the data, the idea of manifold learning is introduced, that is, the local geometric features of the data are represented by the nearest neighbor graph[10]. The weight of the edge is defined by a kernel function, usually using a Gaussian kernel:

$$s_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{\sigma}}, & x_i \text{ and } x_j \text{ are close neighbors} \\ 0, & x_i \text{ and } x_j \text{ are not close neighbors} \end{cases} \tag{4}$$

The main advantage of this method is that it can keep the global characteristics of data while effectively retaining its local characteristics.

Where a is the variance of the data features. Based on the geometric structure between the sample points in the weight matrix S and the mapping matrix V is:

$$\begin{aligned} F &= \sum_{i,j=1}^{n} ||v_i - v_j||^2 s_{ij} \\ &= Tr(V^T DV) - Tr(V^T SV) \\ &= Tr(V^T LV) \end{aligned} \tag{5}$$

D is a diagonal matrix, defined as $d_{i,i}=\sum_{i} s_{ij}$ or $d_{i,i}=\sum_{j} s_{ij}$. h_i, h_j are row vectors. L=D-S represents the graph Laplacian matrix[11].

The general flow of the multi-view concept clustering algorithm (MVCC) is shown in Fig. 1.

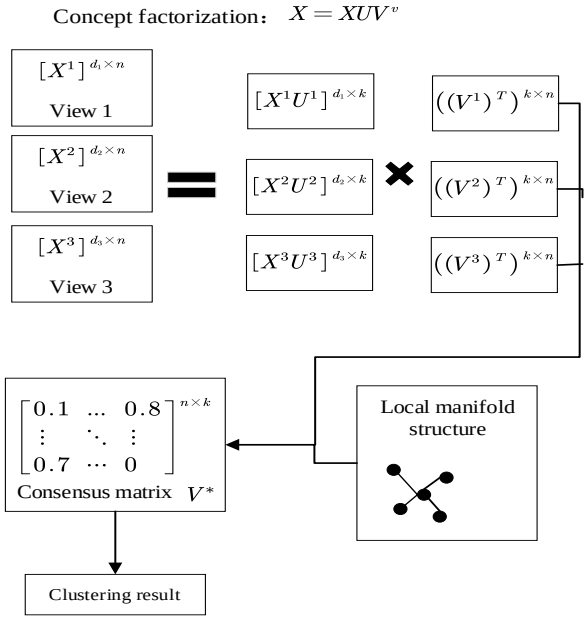

Fig. 1. Flowchart of the multi-view conceptual clustering algorithm

Assume a data set $X=\{X^i\}_{i=1}^{m}$ with m views. $X^v=\{x_j^v\}_{j=1}^{n}\in R^{d_v\times n}$ denotes the data matrix of the vth view, which has d_v features and n sample points. The main task of MVCC is to find a consistent mapping matrix. The algorithm also uses a penalty function to describe the approximation between the mapping matrix of each view and the consistent mapping matrix, and its F-norm representation is as follows:

$$\begin{aligned} &D_{(V^v,V^*)}=w_v||V^v-V^*||_F^2 \\ &s.t.\ w_v\geqslant 0,\sum_{v=1}^{m} w_v=1 \end{aligned} \tag{6}$$

where represents the weight of w_v, let $\Omega=[w_v]$. Combining equations (2)(5)(6) together, the objective function can be obtained:

$$\begin{aligned} &\min_{H^*}\mathcal{O}_{MVCC}=\sum_{v=1}^{m}\left\{\begin{matrix} ||X^v-X^vW^v(H^v)^T||_F^2+ \\ \alpha Tr((H^v)^T L^v H^v)+\beta w_v||H^v-H^*||_F^2 \end{matrix}\right\} \\ &s.t.\ w_{ij}\geqslant 0, h_{ij}^v\geqslant 0, h_{ij}^*\geqslant 0, w_v\geqslant 0, \sum_{v} w_v=1 \end{aligned} \tag{7}$$

Where α and β are balance factors, which are used to balance the local manifold regularization term and consistency penalty term, respectively.

Let $K^v=(X^v)^T X, R=V^v-V^*$ (K^v is also a linear kernel), (7) can be rewritten in the form of matrix trace:

$$\min_{V^*}\mathcal{O}_{MVCC}=\sum_{v=1}^{m}\left\{\begin{array}{c}Tr(K^v)-2Tr\left((U^v)^T K^v V^v\right)+\\ Tr\left((U^v)^T L^v V^v\right)+\\ \alpha Tr\left((V^v)^T L^v V^v\right)+\beta w_v Tr\left((R^v)^T R^v\right)\end{array}\right\} \tag{8}$$
$$s.t.\ w_{ij}\geqslant 0, h_{ij}^v\geqslant 0, h_{ij}^*\geqslant 0, w_v\geqslant 0, \sum_v w_v=1$$

As can be seen from (8), it is a very challenging task to find the global optimal solution for all variables simultaneously [12]. In the following content, an iterative optimization algorithm based on the multiplication rule will be introduced, which aims to find the optimal local solution for each variable. The solution process is as follows:

1) Fixed V^v, V^* and Ω, solve U^v:

At this time, it can be converted into the following formula to solve:

$$\min_{U^v}\mathcal{O}=\sum_{v=1}^{m}\left\{\begin{array}{c}-2Tr\left((U^v)^T K^v V^v\right)+\\ Tr\left((U^v)^T K^v U^v (V^v)^T V^v\right)\end{array}\right\} \tag{9}$$
$$s.t.\ u_{ij}^v\geqslant 0$$

Equivalent to:

$$(u_{ij}^v)^{t+1}\longleftarrow (u_{ij}^v)^t\frac{(K^v V^v)_{ij}}{\left(K^v U^v (V^v)^T V^v\right)_{ij}} \tag{10}$$

2) Fixed U^v, V^* and Ω, solve V^v:

Let ψ_{ij} denote the Lagrange operator. Due to the nonnegative constraint on v_{ij}^v, and let $\Psi=[\psi_{ij}]$, Then the Lagrangian function $\mathcal{L}(V^v)$ for V^v is expressed as:

$$\begin{aligned}\mathcal{L}(V^v)=&Tr(K^v)-2Tr\left((U^v)^T K^v U^v (V^v)^T V^v\right)\\ &+\alpha Tr\left((V^v)^T L^v V^v\right)\\ &+\beta w_v Tr\left((R^v)^T R^v\right)-Tr(\Psi V^v)\end{aligned} \tag{11}$$

Calculate the partial derivative of V^v to (11), and apply the KKT condition $\psi_{ij}v_{ij}=0$, we can get:

$$(v_{ij}^v)^{t+1}\leftarrow (v_{ij}^v)^t\frac{(K^v U^v+\alpha S^v V^v+\beta\omega V^*)_{ij}}{\left(V^v (U^v)^T K^v U^v+\alpha D^v V^v+\beta\omega V^v\right)_{ij}} \tag{12}$$

Orthogonalize U^v and V^v to ensure the uniqueness of the solution. It can be seen that the following formula:

$$\begin{aligned} U^v &\leftarrow U^v (Z^v)^{\frac{1}{2}} \\ V^v &\leftarrow V^v (Z^v)^{-\frac{1}{2}} \\ Z^v &= \mathrm{diag}\left(\sum_i v_{i1}^v, \ldots, \sum_i v_{i1}^v\right) \end{aligned} \tag{13}$$

3) Fixed U^v, V^v and Ω, solve V^*:

At this point (8) is transformed into solving the following formula:

$$\begin{aligned} \min \mathcal{O}(\Omega) &= \sum_v w ||V^v - V^*||_F^2 \\ s.t. \ \sum_v w &= 1, w \geqslant 0 \end{aligned} \tag{14}$$

Finding the partial derivative of V^* to (14) and making it equal to 0 can be solved to get:

$$V^* = \frac{\sum_v \omega V^v}{\sum_v \omega} = \sum_v \omega V^v \tag{15}$$

4) Fixed U^v, V^v and V^*,solve Ω:

At this time, the minimization problem of (8) will degenerate into a convex optimization problem, expressed as:

$$\begin{aligned} \min \mathcal{O}(\Omega) &= \sum_v w ||V^v - V^*||_F^2 \\ s.t. \ \sum_v w &= 1, w \geqslant 0 \end{aligned} \tag{16}$$

In order to effectively avoid the occurrence of invalid solutions, the introduction of 2-norm sparse regularization can make the obtained perspective weights more practical, specifically expressed as the following formula:

$$\begin{aligned} \min \mathcal{O}(\Omega) &= \sum_v w ||V^v - V^*||_F^2 + \gamma ||\Omega||^2 \\ s.t. \ \sum_v w &= 1, w \geqslant 0 \end{aligned} \tag{17}$$

Where the parameter γ controls the smoothness of the weight vector Ω. Solving equation (17) can be regarded as a quadratic programming task.

4. Parallel Multi-View Concept Clustering Algorithm

Through the analysis of the above-mentioned multi-view concept clustering algorithm MVCC, it can be seen that the main steps of the algorithm are independent [13]. Therefore, the MVCC algorithm can be used for distributed

parallel computing, thereby improving the performance and efficiency of the algorithm [14].

Due to the data transmission of the distributed computing framework, disk IO and network delay need to be considered. Therefore, putting data stored as close together as possible can greatly improve the performance of the system [15]. Therefore, an important concept of Spark is: to only move tasks, not move data. By storing data distributedly on multiple nodes, the Driver node can obtain the required resources from the resource manager and then transmit the corresponding computing tasks to each node, thereby realizing parallel processing of multiple nodes [16].

The basic idea of the multi-view concept clustering parallel method is:

1) Assign a computing node to each view to calculate the association matrix U^v and the mapping matrix V^v;

2) Allocate a computing node for the data set to calculate the consistency mapping matrix V^*;

3) Assign a calculation node to the view weight to calculate the view weight Ω.

It can be seen from the above algorithm idea that the consistency matrix V^* and the weight vector Ω are in a global position. Therefore, consider defining V^* and Ω as global variables.Through distributed storage of multi-view data set $\{X^i\}_{i=1}^m$ in HDFS, U^v and V^v stored in each partition are generated to realize parallel computing.

To sum up, the application of the MVCC algorithm in distributed parallel computing has obvious advantages. By storing data tightly, reducing the cost of data transmission, and adopting the strategy of only moving tasks, the MVCC algorithm has achieved remarkable results in performance and efficiency. In addition, Spark's distributed computing framework provides strong support for multi-node parallel processing, which further enhances the practical application value of the algorithm. With the continuous development of distributed computing technology, the MVCC algorithm is expected to become an important pillar in the field of big data processing in the future.

In Spark, it is the executors in each worker node that actually perform computing operations. If the calculation of the executor needs to use the variables in the driver, you need to apply for a copy of the variables from the driver[17]. The number of tasks in the executor must be consistent with the number of variable copies. This consumes a lot of memory resources and communication costs. Spark provides a broadcast variable to help developers solve this problem[18]. Broadcast variables are read-only variables that can be written and customized by developers. They are cached in the executor, which can effectively reduce the communication overhead between nodes and improve system performance.

The steps of the parallel multi-view concept clustering algorithm are as follows:

Algorithm 1 Parallel algorithm for multi-view clustering based on Spark

Input: Multi-view data matrix$\{X^i\}_{i=1}^m$, number of neighbors p, parameter $\{\alpha,\beta,\gamma\}$, number of clusters k.

Output: Clustering result

Algorithm Description:

Initialization:

1.Store the data set in the distributed file system HDFS;

2.Read data set from HDFS, create view matrix RDD;

3.Use the map() function to convert the view matrix RDD to a graph Laplacian matrix RDD;

4.Initialize the weight vector Ω and use it as a broadcast variable;

5.Use map() to perform kmeans on each view, and initialize U^v, V^v, V^* with the result;

6.Cache U^v, V^v, and use V^* as a broadcast variable.

Iterative Calculation:

1. Fixed V^v, V^* and Ω, Calculate U^v from the formula (10) using map();

2. Fixed U^v, V^* and Ω, Calculate V^v from the formula (12) using map();

3. Use map() to perform orthogonal normalization on U^v and V^v according to formula (13);

4. Use collectAsMap() to collect U^v and V^v operation results and cache them in memory;

5. Fix U^v, V^v and Ω, calculate V^* according to formula (15), and update the consistency matrix V^* to the broadcast variable;

6. Fix U^v, V^v and V^*, update Ωaccording to formula (17), and update the weight matrix Ω to broadcast variables.

7. According to the formula (8), it is judged whether the maximum number of iterations or the convergence condition is reached, and if it is reached, the iteration is exited, otherwise steps 1-7 in the iterative calculation are repeated;

8. Execute $arg\max_{1\leqslant j\leqslant k} H_{:,j}^*$ to get the final clustering result.

The flowchart of the parallel multi-view conceptual clustering algorithm is shown in Fig.2.

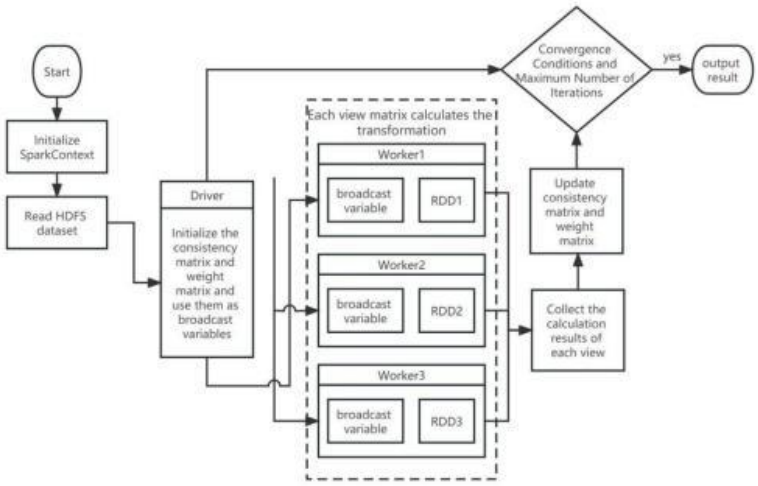

Fig. 2. Flowchart of the parallel multi-view conceptual clustering algorithm

5. Experiment

5.1 Lab Environment

In this experiment, we use the "master-slave" architecture to test. The whole experimental system includes a master node and two slave nodes, with three hosts participating together. In order to ensure the smooth progress of the experiment, we have strictly planned the hardware configuration and software environment of the master node and the slave node. Specifically, the master node is equipped with a high-performance processor, sufficient memory, and fast storage devices to ensure stable operation when processing a large amount of data. Table 1 and Table 2 list the specific parameters of the host configuration and development environment in detail.

Table 1

Host configuration information

	Spark01	Spark02	Spark03
CPU	8 cores	14cores	12cores
Memory	16G	16G	16G
SSD	1T	512G	1T

Table 2

Development environment

	Spark01	Spark02	Spark03
operating system	Debian11.3	Debian11.1	Debian11.1
JDK	11.0.16	11.0.16	11.0.16
Hadoop	Hadoop-3.3.4	Hadoop-3.3.4	Hadoop-3.3.4
Spark	Spark-3.3.1	Spark-3.3.1	Spark-3.3.1
Python	3.9.13	3.9.13	3.9.13
Anaconda	22.9.0	22.9.0	22.9.0

PySpark	3.3.1	3.3.1	3.3.1
IDE	VS Code		

5.2 Dataset Description

In this paper, the following three data sets are used in the experiment:

1) BBC data set

The BBC data set is a new data set, that contains 685 news reports from the BBC, which can be divided into five categories (business, entertainment, politics, sports, and technology). It is described by four views, and the dimensions are 4659-dimensional, 4633-dimensional, 4665-dimensional, and 4684-dimensional. These four dimensions reflect the richness and diversity of data sets and provide different levels of analytical perspectives.

2) Handwritten Digits(HD)

The HD data set is a multimodal data set of handwritten images. The data set has 10,000 samples, the two view dimensions are 784, 256, and 10 clusters. These two views respectively represent different feature attributes of handwritten images.

3) ALOI Dataset

ALOI is a multimodal object image data set. The data set has 11,025 samples, and the three view dimensions are 77, 64, 64, and 100 clusters. Compared with other similar data sets, ALOI is unique in its rich multimodal features.

4) 3Sources Dataset

3Sources contains 948 news articles collected from three different media outlets. Among these news reports, 169 were reported by three media at the same time, and this part of the articles was selected as a multi-view data set, and each news source was regarded as a view of the data set.

5) ORL Dataset

The ORL data set contains a face data set consisting of 400 pictures of 10 different photos of 40 people. The uniqueness of this data set is that it contains three different types of feature sets, which represent different visual characteristics of human faces, and provides multi-angle and multi-scale analysis methods.

See Table 3 for information about each data set. Although the datasets utilized in this study are representative, they may not fully capture the diversity and complexity of real-world data. Issues such as data skew and highly heterogeneous distributions were not extensively tested, which could affect the framework's generalizability in broader applications. Addressing these challenges will require additional experimentation with more diverse and complex datasets.

Table 3

The Information of Data set

Data Set	Number of samples	Number of views	Number of clusters
BBC	685	4	5
HD	10000	2	10

ALOI	11025	3	100
3Sources	169	3	6
ORL	400	3	40

5.3 Evaluation Indicators

5.3.1 NMI

Normalized mutual information (NMI) is an information measurement method. The essence of clustering is to find patterns or structures in data, and NMI provides a way to quantify the clustering effect. It is defined as follows:

$$NMI = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{18}$$

In the formula, X and Y are N-dimensional vectors, $I(X,Y)$ is the mutual information of X and Y, and H(X) and H(Y) are the entropy of X and Y respectively. The NMI value ranges from [0,1], the closer the NMI value is to 1, the better the clustering effect.

5.3.2 ACC

ACC is accuracy, defined as follows:

$$ACC = \frac{1}{N}\sum_{i}^{N} y_i = map(p_i) \tag{19}$$

In the formula, p_i is the label generated by the i-th sample in the cluster, y_i is its true label, and $map(\cdot)$ represents the redistribution of the cluster label, which is usually implemented by the Hungarian algorithm. The ACC value ranges from [0,1], and the closer the ACC value is to 1, the better the clustering effect.

5.3.3 Speedup Ratio

The acceleration ratio is an important performance evaluation index, which is mainly used to measure the efficiency of parallel computing systems in quantitative performance improvement. Its definition is as follows:

$$S_p = \frac{T_1}{T_p} \tag{20}$$

In the formula, T_1 is the running time under a single node, T_p is the running time of parallel computing, and the larger S_p is, the higher the parallelization efficiency of the algorithm is.

5.4 Experimental Results and Algorithm Evaluation

5.4.1 Parallel Performance Experiment

In this experiment, the serial and parallel algorithms of multi-view concept clustering are mainly concerned, and a total of 10 clustering processes are empirically analyzed for the five data sets shown in Table 3. During the experiment,

the time spent in each round of clustering will be recorded and the average value will be calculated. In addition, the stability of the experimental results is measured by calculating the standard deviation, which is the final experimental result. The experimental results are shown in Table 4, Table 5, and Fig. 3 respectively. In these tables and charts, the vertical axis MVCC represents the serial multi-view concept clustering algorithm, and the horizontal axis represents different data sets (as mentioned above).

Table 4 shows the experimental results of five data sets when using a serial multi-view concept clustering algorithm. It can be observed that different data sets have certain differences in clustering effect. This is because the distribution and correlation of features may be different in different data sets, which leads to different clustering results. Table 5 shows the experimental results of five data sets under the parallel multi-view concept clustering algorithm. Similar to Table 4, we can see that the parallel algorithm also shows some differences in different data sets. However, it is worth noting that, compared with a serial algorithm, the parallel algorithm has achieved a better clustering effect on some data sets. Fig. 3 shows the experimental results visually. As can be seen from the figure, the parallel algorithm does achieve better clustering results on some data sets, especially on data sets 5 and 6. However, on other data sets, the serial algorithm shows higher clustering accuracy. This shows that in practical application, choosing the appropriate clustering algorithm needs to be weighed according to the characteristics and needs of specific data sets.

We can see from the experimental data recorded in Table 4, that the minimum standard deviation of the running time of MVCC and PMVCC (for different node numbers) running 10 times under different data sets is 0.0036, and the maximum is 0.0178, which does not exceed 0.03, which belongs to a small deviation range, so the errors in the experimental process, statistical omissions, and the contingency of the experiment are excluded, and the data reliability is relatively high.

Table 4

Running time of algorithm(s)

	BBC	HD	ALOI	3Sources	ORL
MVCC	9.23 (0.0077)	95.64 (0.0047)	58.77 (0.0111)	6.20 (0.0063)	38.68 (0.0102)
PMVCC (1 node)	9.46 (0.0178)	97.27 (0.0054)	60.25 (0.0137)	6.25 (0.0064)	39.23 (0.0083)
PMVCC (2 node)	8.95 (0.0132)	75.39 (0.0075)	49.13 (0.0064)	6.07 (0.0058)	32.55 (0.0083)
PMVCC (3 node)	8.52 (0.0092)	62.76 (0.0044)	36.26 (0.0036)	5.85 (0.0102)	25.92 (0.0056)

Table 5 uses the MVCC running time as the standard according to the statistical results in Table 4, calculates the speedup ratio of using PMVCC under

different node numbers according to formula (20), and draws the speedup ratio line graph in Fig. 3 based on this. From the overall experimental results, PMVCC can accelerate the traditional serial algorithm.

Table 5

Algorithm acceleration ratio

	BBC	HD	ALOI	3Sources	ORL
MVCC	1	1	1	1	1
PMVCC (1node)	0.9757	0.9832	0.9754	0.9920	0.9860
PMVCC (2 node)	1.0313	1.2686	1.1962	1.0214	1.1883
PMVCC (3 node)	1.0833	1.5239	1.6208	1.0598	1.4923

5.4.2 Clustering Effect Experiment

In this experiment, the multi-view concept clustering algorithm is used to cluster the same data set many times. The main purpose of the experiment is to compare and analyze the performance of serial and parallel algorithms in clustering accuracy (ACC) and standard mutual trust (NMI) under different numbers of nodes, so as to evaluate the stability and effect of the algorithms.

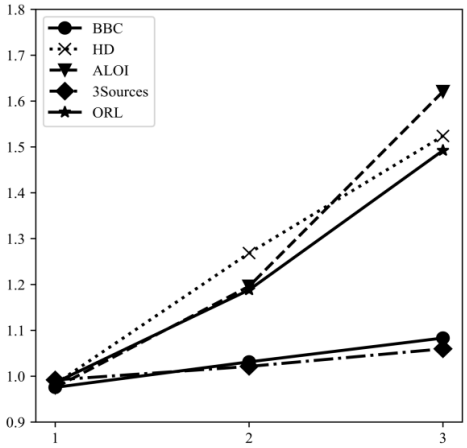

Fig. 3. Algorithm acceleration ratio under different node numbers

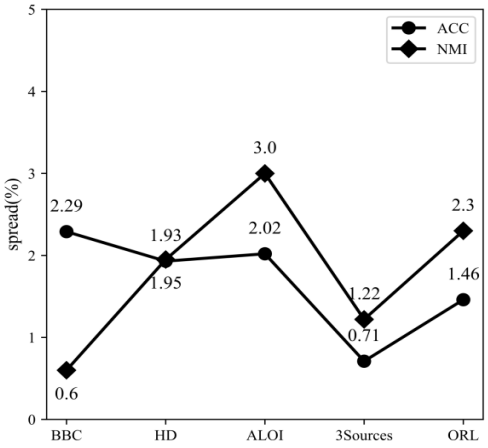

Fig. 4. Spread between the highest and lowest values of ACC and NMI(%)

Table 6

Cluster evaluation the result of ACC(%)

	BBC	HD	ALOI	3Sources	ORL
MVCC	**76.84** (0.0035)	83.24 (0.0025)	53.46 (0.0178)	77.01 (0.0041)	60.04 (0.0047)
PMVCC (1 node)	75.89 (0.0112)	83.57 (0.0033)	53.61 (0.0062)	76.93 (0.0156)	**60.93** (0.0091)
PMVCC (2 node)	75.08 (0.0084)	**84.88** (0.0102)	54.08 (0.0093)	76.88 (0.0136)	60.23 (0.0038)
PMVCC (3 node)	75.23 (0.0088)	84.67 (0.0028)	**54.56** (0.0074)	**77.43** (0.0183)	60.55 (0.0028)

Fig.3 shows a consistent increase in acceleration ratio, experimental limitations prevented a comprehensive exploration of its scalability. We were unable to test the acceleration limit by adding more nodes or identifying the platform period and inflection point of algorithm acceleration.

The experimental results are shown in Table 6, Table 7, and Fig. 4. From these tables and graphs, we can clearly see the clustering effect of serial and parallel algorithms under different node numbers. It is worth noting that in most cases, with the increase in the number of nodes, ACC and NMI values will show an upward trend. This shows that increasing the number of nodes is helpful to improve the clustering effect. However, in some specific cases, the increase in the number of nodes does not bring a better clustering effect, which may be related to the characteristics of the data set.

In order to comprehensively evaluate the applicability of these two algorithms in different scenarios, five representative data sets are selected for experiments. During the experiment, each data set was clustered 10 times, and the average accuracy rate (ACC) and Nash-Minton similarity index (NMI) of each operation were calculated. Tables 6 and 7 show the average and standard deviation of the experimental results. As can be seen from the table, MVCC and PMVCC have achieved good clustering results under different node numbers [19]. In order to visually represent the performance of each algorithm on each data set, the best indicators obtained by each data set under different algorithm conditions are marked in bold in the table. By comparing the statistical data in Table 6 and Table 7, we can further analyze the performance differences of different algorithms on various data sets. In order to more intuitively show the difference between the highest and lowest values of clustering indicators of each data set under different algorithm conditions, Fig. 4 is calculated and drawn, which is the line chart of the difference between the highest and lowest values of clustering indicators obtained by each data set under different algorithm conditions. The difference does not exceed 5%. It can be seen that the parallelized multi-view concept clustering algorithm will not significantly improve the clustering accuracy of the serial algorithm, and the clustering effect of the algorithm is in a relatively stable state.

Table 7

Cluster evaluation the result of NMI(%)

	BBC	HD	ALOI	3Sources	ORL
MVCC	**65.33** (0.0110)	78.46 (0.0055)	80.47 (0.0106)	69.83 (0.0166)	58.73 (0.0016)
PMVCC (1 node)	64.94 (0.0037)	79.82 (0.0038)	82.55 (0.0106)	69.57 (0.0084)	**60.11** (0.0011)
PMVCC (2 node)	65.26 (0.0049)	**80.02** (0.0055)	82.18 (0.0118)	70.17 (0.0192)	59.62 (0.0058)
PMVCC (3 node)	65.11 (0.0058)	78.96 (0.0058)	**82.95** (0.0187)	**70.43** (0.0185)	59.82 (0.0100)

Based on the above experimental results, it can be shown that the parallel multi-view concept clustering algorithm can have a considerable acceleration effect on the original serial algorithm on the premise of ensuring that the accuracy of the clustering results does not decrease.

6. Conclusion

With the rapid development of information technology, the challenge of expanding data volume has become increasingly prominent. To address this, this paper proposes a solution based on the current mainstream distributed computing engine, Spark. The experimental results demonstrate that this method significantly improves operational efficiency while ensuring accuracy.

First, while Fig.3 shows a consistent increase in acceleration ratio, experimental limitations prevented a comprehensive exploration of its scalability. We were unable to test the acceleration limit by adding more nodes or identifying the platform period and inflection point of algorithm acceleration. These aspects will be key focuses of our future research.

Secondly, as a memory-based parallel computing engine, Spark faces potential challenges of memory overflow when handling large-scale datasets. While Spark’s data-spilling mechanism addresses such issues to some extent, it may lead to significant performance degradation. Further exploration and optimization of memory management strategies tailored to practical application scenarios are required to address this limitation effectively.

Furthermore, although the datasets utilized in this study are representative, they may not fully capture the diversity and complexity of real-world data. Issues such as data skew and highly heterogeneous distributions were not extensively tested, which could affect the framework's generalizability in broader applications. Addressing these challenges will require additional experimentation with more diverse and complex datasets.

Despite these limitations, this paper provides a detailed discussion of the proposed framework, including its construction, the implementation of a multi-view concept clustering algorithm on Spark, and an analysis of the results. We believe that this approach offers an effective method for tackling the challenges of expanding data volume.

In conclusion, while the distributed computing engine based on Spark and the multi-view concept clustering algorithm demonstrate great potential, there are still unresolved issues and challenges that need to be addressed. Through continuous optimization and further research, we are confident that this framework will play an increasingly significant role in future applications and contribute to the advancement of related technologies and industries.

Acknowledgments

This work was supported by the scientific research project of Jilin Provincial Department of Education (JJKH20230854KJ).

REFERENCES

[1] FU Lele, LIN Pengfei, Athanasios V. Vasilakos et al. An Overview of Recent Multi-View Clustering. Neurocomputing, Volume 402,2020, Pages 148-161, ISSN 0925-2312.

[2] LIN Yuan, YANG Xiaofei, XING Zhiwei et al. Latent Multi-View Semi-Nonnegative Matrix Factorization with Block Diagonal Constraint. Axioms, 2022, 11, 722.

[3] Mao Yimin, Gan Dejin, Liao Lefa and others. Parallel partition clustering algorithm based on Spark framework and ASPSO. Journal of Communications, 2022, 43 (03): 148-163.

[4] Sayantan Mitra, Mohammed Hasanuzzaman, and Sriparna Saha. 2020. A Unified Multi-view Clustering Algorithm Using Multi-objective Optimization Coupled with Generative Model. ACM Trans. Knowl. Discov. Data 14, 1, Article 2 (January 2020), 31 pages.

[5] Hussain Syed Fawad, Khan Khadija, Jillani Rashad. Weighted multi-view co-clustering (WMVCC) for sparse data. APPLIED INTELLIGENCE. Volume 52, 2021, Page 398-416, ISSN 0924-669X.

[6] MAO Y M, GAN D J, LIAO LF et al. Parallel Division Clustering Algorithm Based on Spark Framework and ASPSO. Journal on Communications, 2022, 43 (03): 148-163.

[7] Liu Dongjiang, Li Jianhui. A Parallel Graph Clustering Algorithm Based on Spark. Journal of Systems Simulation, 2020, 32 (6): 1038-1050.

[8] LIU D J, LI J H. Study of Parallelized Graph Clustering Algorithm Based on Spark. Journal of System Simulation, 2020, 32 (06): 1038-1050.

[9] WANG Jiankai. Clustering Algorithm for Big Datasets with Mixed Attribute Features under Spark. Mathematical Problems in Engineering, vol.2022, 11 pages, 2022.

[10] Zineb Dafir, Said Slaoui. An Efficient Parallel Algorithm for Clustering Big Data based on the Spark Framework. International Journal of Advanced Computer Science and Applications (IJACSA), Vol.13, Iss. 7, (2022).

[11] Keyvan Golalipour, Ebrahim Akbari. From Clustering to Clustering Ensemble Selection: A Review. Engineering Applications of Artificial Intelligence, Volume 104, 2021, 104388, ISSN 0952-1976.

[12] Tahani Alqurashi, WANG Wenjia. Clustering Ensemble Method. J. Mach. Learn. & Cyber. 10,1227–1246 (2019).

[13] WU Guoqing, CAO Liqiang, TIAN Hongyun et al. HY-DBSCAN: A Hybrid Parallel DBSCAN Clustering Algorithm Scalable on Distributed-memory Computers. Journal of Parallel and Distributed Computing, Volume 168, 2022, Pages 57-69, ISSN 0743-7315.

[14] YOU Congzhe, SHU Zhenqiu, FAN Honghui. Non-negative Sparse Laplacian Regularized Latent Multi-View Subspace Clustering. 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Year 2020, Pages 210-213.

[15] Liu Weiming, Cui Yu, Mao Yimin. Parallel K-means Algorithm Based on MapReduce and MSSA. Application Research of Computers, 2022, 39 (11).

[16] LIU W M, CUI Y, MAO Y M. Parallel K-means Algorithm based on MapReduce and MSSA. Application Research of Computers, 2022 Vol. 39 No. 11.

[17] Shipeng Cao, Zhuoran Tang, Weiwei Zuo et al. Key Technology and Application of Distributed Parallel Computing for Machine Learning. Journal of Intelligent Systems, 2021, 16 (05): 919-930.

[18] ZOU Zhenwan, LI Feng, YANG Huiting et al. Research on Parallel CKLDC-means Clustering Algorithm Based on Hadoop Platform. 2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2022, pp. 196-199.

[19] Wang, H., Yang, Y., Zhang, X., & Peng, B. (2020). Parallel Multi-View Concept Clustering in Distributed Computing. NEURAL COMPUTING & APPLICATIONS, 32(10), 5621–5631.