Journal of University of Science and Technology Beijing
Volume 11, Number 3, June 2004, Page 283

Automation

CABOSFV algorithm for high dimensional sparse data clustering

Sen Wu and Xuedong Gao

Management School, University of Science and Technology Beijing, Beijing 100083, China
(Received 2003-04-30)

Abstract: An algorithm, Clustering Algorithm Based On Sparse Feature Vector (CABOSFV), was proposed for the high dimensional clustering of binary sparse data. This algorithm compresses the data effectively by using a tool 'Sparse Feature Vector', thus reduces the data scale enormously, and can get the clustering result with only one data scan. Both theoretical analysis and empirical tests showed that CABOSFV is of low computational complexity. The algorithm finds clusters in high dimensional large datasets efficiently and handles noise effectively.

Key words: clustering; data mining; sparse; high dimensionality

1 Introduction

Clustering is one of the key tasks in data mining. It discovers data distributions in datasets, which groups the data objects into clusters so that the objects within a cluster are more similar than those in different clusters. As a branch of statistics, cluster analysis has been studied extensively for many years. Other contributing fields include data mining, statistics, machine learning, spatial database technology, biology and marketing. In data mining, effort focuses on effective and efficient clustering algorithms or methods for large datasets. The challenges of clustering in data mining are as follows: scalability, capability to handle high dimensional data, discovery of clusters with arbitrary shape, ability to deal with noisy data, insensitivity to the order of input records, interpretability and description of clustering result.

This paper focuses on high dimensional data clustering for binary attributes. Many clustering algorithms are good at handling low dimensional data, involving only several dimensions. It is challenging to cluster data objects in a high dimensional space, especially considering that such data can be very sparse and highly skewed [1].

1.1 Related work

From literature, it can be known that CLIQUE [2] and CURE (Clustering Using REpresentatives) [3] are good at handling high dimensional data clustering. CLIQUE, named for CLustering In QUEst, is a density and grid-based approach for high dimensional data clustering that provides automatic sub-space clustering of high dimensional data aiming at effective treatment of high dimensionality, interpretability of results and scalability. CURE is a bottom-up hierarchical clustering algorithm that employs a method that each cluster is represented by a certain fixed number of points that are generated by selecting well scattered points from the cluster and shrinking them toward the center of the cluster by a specified fraction. CURE can identify clusters having non-spherical shapes and wide variances in size. Both CLIQUE and CURE are effective for high dimensional data clustering. They successfully cluster data objects of 52 dimensions and 40 dimensions respectively. But they are both for interval-scaled variables. And the accuracy of CLIQUE clustering results may be degraded due to the simplicity of the method, while CURE has high computation complexity.

The reason that many algorithms are not good at handling high dimensional large datasets is as follows.

(1) First distance measures for low dimensional data clustering are not proper to be applied to high dimensional problems directly, especially when the data are highly sparse. In many cases the measures do not reflect the dissimilarity of objects truly, which must lead to poor quality of clusters. For example, CLARANS (Clustering Large Applications based upon RANdomized Search) [4], DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [5], BIRCH (Balanced Interative Reducing and Clustering using Hierarchies) [6] and STING (STatistical INformation Grid) [7] are all not good at handling

Corresponding author: Sen Wu, **E-mail:** wusen@manage.ustb.edu.cn

high dimensional datasets.

(2) Many clustering methods, typically the grid-based clustering approaches such as STING, STING+ CLIQUE [8], summarize the data in order to improve the efficiency of the algorithms. This makes the algorithms faster, yet may lower the quality and accuracy of the clusters due to the simplicity of the methods.

Among the above three algorithms, CLIQUE has good scalability as the number of data dimensions is increased just at the expense the low quality clustering result.

(3) Some clustering methods must scan data more than once in order to finish the clustering tasks. This reduces the efficiency of the algorithms. In addition, many algorithms are not scalable due to not considering how to work with limited resources. A typical example is that memory is much smaller than the size of large datasets.

(4) Many clustering algorithms themselves are of high time complexity. And for some of them, when the dimensionality of the data points is not small, the time complexity increases. For example, CURE can handle high dimensional clustering efficiently. Its time complexity is $O(n^2)$ for low dimensionality (n is the number of objects in a sample set), but the worst-case time complexity is $O(n^2 \log n)$ when dimensionality increases.

1.2 Features of CABOSFV

An algorithm that overcomes the above drawbacks named as CABOSFV was presented in this paper. The salient features of CABOSFV are discussed below.

(1) For high dimensional sparse data of binary variables, a new dissimilarity measure in CABOSFV algorithm named as Sparse Feature Dissimilarity (SFD) of a set (SFD) was put forward. SFD of a set represents the similar degree of all the objects in a set. The smaller the SFD is, the more similar the objects are. SFD of a set is very appropriate to act as distance measure for high dimensional sparse data clustering of binary variables.

(2) A new concept Sparse Feature Vector (SFV), a vector of four items, was introduced by which CABOSFV summarizes the information about a set or a cluster effectively that we need for making clustering decisions. So CABOSFV just stores the SFV of a set but not all the information about all the objects in the set. We can think of a cluster as a set of data points, but in fact only the SFV is stored. Hence data is reduced.

(3) Moreover we have proved that SFVs of clusters can be calculated incrementally and accurately when clusters are merged. That is, when two clusters are merged, the SFV of the new cluster can be calculated directly from the SFVs of the original two clusters. Consequently the quality of the clusters will not be affected.

(4) CABOSFV is an incremental method that does not require the whole data in advance. It implements creation and mergence of clusters by scanning the dataset only once. So CABOSFV is not I/O costly. Due to this feature and data reduction by SFV, CABOSFV is very efficient to handle large dataset.

(5) CABOSFV is of low computational complexity ($O(nk)$, where n is the number of objects, k is the number of clusters). Both theoretical analysis and empirical tests show that CABOSFV finds clusters in high dimensional large datasets for spare binary attributes efficiently and handles noise effectively.

2 Concepts and theorem for CABOSFV clustering

CABOSFV is very efficient and scalable to handle high dimensional sparse data clustering for binary attributes. Its salient features rely on two new concepts SFD of a set, SFV and a theorem SFV additivity. In this section, the definition of these two concepts and the theorem is given.

2.1 SFD of a set

We proposed a new measure to represent the dissimilarity of the objects in a set. We call it SFD of a set and define it as follows.

Definition 2.1 (SFD of a set): Given n objects, each object is described by m attributes. X is a set of objects, in which the number of objects is denoted as $|X|$, the number of attributes that equal 1 for all objects is indicated by a, and the number of attributes that equal 1 for some objects and equal 0 for other objects is indicated by e. SFD of set X, denoted as $\mathrm{SFD}(X)$, is defined as:

$$\mathrm{SFD}(X) = \frac{e}{|X| \times a} \tag{1}$$

$\mathrm{SFD}(X)$ represents the similarity of the objects in set $|X|$. It will serve as the distance measure in CABOSFV. The smaller the SFD is, the more similar the objects are.

2.2 SFV

CABOSFV compresses data effectively in order to

reduce data and improve the efficiency. A new tool, SFV, is presented to implement this task. SFV, a vector of four elements, contains all the information of a set we need in CABOSFV clustering process.

Definition 2.2 (SFV): Given n objects, each object is described by m attributes. X is a set of objects, in which the number of objects is denoted as $|X|$; the number of attributes that equal 1 for all objects is indicated by a, accordingly the sequence numbers of these attributes are $j_{s_1}, j_{s_2}, \cdots, j_{s_a}$; the number of attributes that equal 1 for some objects and equal 0 for other objects is indicated by e, the corresponding sequence numbers are $j_{ns_1}, j_{ns_2}, \cdots, j_{ns_e}$. The vector

$$\mathrm{SFV}(X) = (|X|, S(X), \mathrm{NS}(X), \mathrm{SFD}(X)) \tag{2}$$

is called the SFV of set X, where $|X|$ is the number of objects in set X; $S = \{j_{s_1}, j_{s_2}, \cdots, j_{s_a}\}$ is the set of sequence numbers of the attributes that equal 1 for all objects in set X; $\mathrm{NS} = \{j_{ns_1}, j_{ns_2}, \cdots, j_{ns_e}\}$ is the set of sequence numbers of the attributes in set X that equal 1 for some objects and equal 0 for other objects; $\mathrm{SFD}(X)$ is the SFD of set X. According to Definition 2.1 (SFD of a set) we know that $a = |S|$ and $e = |\mathrm{NS}|$, thus we have

$$\mathrm{SFD}(X) = \frac{|\mathrm{NS}|}{|X| \times |S|} \tag{3}$$

If set X contains only one object, $|X|$, the number of objects in the set, equals 1; the elements in set S are the sequence numbers of the attributes that equal 1 for the only object; NS is Φ; $\mathrm{SFD}(X)=0$; and $\mathrm{SFV}(X) = \{1, S, \Phi, 0\}$.

SFV summarizes the sparse state of all the objects in a set including the SFD. This summary is not only efficient because it stores much less than all the data objects in the set, but also sufficient for calculating all the measurements we need for making clustering decisions in CABOSFV based on the additivity of SFV, which will be discussed successively.

2.3 Additivity of SFV

In this part we will give the definition of the additivity of SFV first, then present the additivity theorem of SFV. The definition and the theorem, together with the definitions of SFD of a set and SFV, are at the core of CABOSFV.

Definition 2.3 (additivity of SFV): Given n objects, each object is described by m attributes. X and Y are two sets of objects with no intersection, corresponding SFV are

$$\begin{cases} \mathrm{SFV}(X) = (|X|, S(X), \mathrm{NS}(X), \mathrm{SFD}(X)) \\ \mathrm{SFV}(Y) = (|Y|, S(Y), \mathrm{NS}(Y), \mathrm{SFD}(Y)) \end{cases} \tag{4}$$

We define the additivity of SFV as

$$\mathrm{SFV}(X) + \mathrm{SFV}(Y) = (N, S, \mathrm{NS}, \mathrm{SFD}) \tag{5}$$

where

$N = |X| + |Y|$;

$S = S(X) \cap S(Y)$;

$\mathrm{NS} = (\mathrm{NS}(X) \cup \mathrm{NS}(Y) \cup S(X) \cup S(Y)) \setminus (S(X) \cap S(Y))$

$$\mathrm{SFD} = \frac{|\mathrm{NS}|}{N \times |S|}.$$

We can prove that, when two sets are merged, the SFV of the new set can be calculated directly from the SFVs of the original two sets according to the additivity defined above. We have the SFV additivity theorem as follows.

Theorem 2.1 (SFV additivity theorem): Given n objects, each object is described by m attributes. X and Y are two sets of objects with no intersection. Union of X and Y is indicated by $X \cup Y$, corresponding SFVs of X, Y and $X \cup Y$ are

$$\begin{cases} \mathrm{SFV}(X) = (|X|, S(X), \mathrm{NS}(X), \mathrm{SFD}(X)) \\ \mathrm{SFV}(Y) = (|Y|, S(Y), \mathrm{NS}(Y), \mathrm{SFD}(Y)) \\ \mathrm{SFV}(X \cup Y) = (|X \cup Y|, S(X \cup Y), \mathrm{NS}(X \cup Y), \\ \qquad \mathrm{SFD}(X \cup Y)) \end{cases} \tag{6}$$

Then

$$\mathrm{SFV}(X \cup Y) = \mathrm{SFV}(X) + \mathrm{SFV}(Y) \tag{7}$$

Due to definition of SFV and SFV additivity theorem, SFVs of clusters can be stored and calculated accurately when clusters are merged. So CABOSFV is not only efficient because it only stores a summary of the data, but also sufficient for getting the information we need for clustering because of the additivity theorem.

3 CABOSFV clustering algorithm

3.1 Two-layers structure

CABOSFV clustering process can be illustrated by a 'Bottom-Up' structure of two layers as shown in **figure 1**. In the bottom layer there are the n objects that will be clustered according to their similarity, and in the top layer there are k clusters that we get by CABOSFV clustering. For each object, which is treated as a special set containing only one object, and for each cluster, the most important information we need

keep in clustering process is the SFV. Assuming the SFD threshold, the upper limit, for a final cluster is b, the clustering process is as follows.

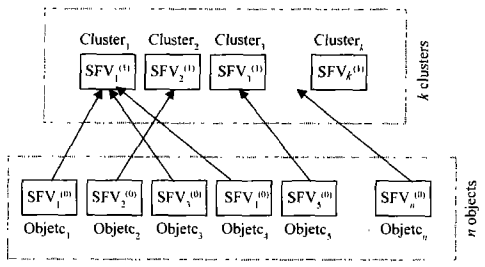

Figure 1 Two layers structure of CABOSFV clustering process.

Regard n individual objects as n object sets respectively. The SFV of each object set is known because that, $\mathrm{SFV}(X)=\{1,S,\Phi,0\}$ exists for a set X with only one object according to the definition of SFV, and S is the sequence numbers of the attributes that equal 1 for the only object, which is also known in advance. Scan SFV from first object set, the creation of clusters and the mergence of object sets will be implemented during the scanning process. First cluster 1 is created containing object 1. Then check if object 2 can be assigned into cluster 1 together with object 1 according to the SFV additivity theorem and SFD threshold of a cluster b. If the SFD of the merged set is not greater than the upper limit b, put object 2 in cluster 1; otherwise, create cluster 2 containing object 2. After that check if object 3 can be assigned into cluster 1 or cluster 2. If object 3 can be put in both cluster 1 and cluster 2, put object 3 in the cluster having a lower SFD after the mergence. If object 3 can be assigned neither into cluster 1 nor into cluster 2, create a new cluster 3 containing object 3. Similarly, each rest object is either assigned into an existing cluster or into a new cluster until all the objects have been scanned. Thus the clustering process is finished with just one data scan.

3.2 Algorithm steps

Now we provide the detailed algorithm steps. Given n objects, object i is described by m attributes $x_{i1},x_{i2},\cdots,x_{im}$, the SFD threshold for a cluster is b, steps of CABOSFV clustering are as follows.

Step 1. View each object as a set containing only one object and denote it as $X_i^{(0)}$, $i\in\{1,2,\cdots,n\}$. Accordingly the SFV is denoted as $\mathrm{SFV}(X_i^{(0)})$, $i\in\{1,2,\cdots,n\}$.

Step 2. According to the additivity theorem of SFV, calculate $\mathrm{SFV}(X_1^{(0)}\cup X_2^{(0)})=\mathrm{SFV}(X_1^{(0)})+\mathrm{SFV}(X_2^{(0)})$. Compare the SFD of the merged set with the threshold b. If it is not greater than b, merge $X_1^{(0)}$ and $X_2^{(0)}$, and the merged set is a initial cluster denoted as $X_1^{(1)}$; otherwise, $X_1^{(0)}$ and $X_2^{(0)}$ are regarded as tow initial clusters, represented respectively by $X_1^{(1)}$ and $X_2^{(1)}$. The number of clusters is indicated by c.

Step 3. For object set $X_3^{(0)}$, calculate

$$\mathrm{SFV}(X_3^{(0)}\cup X_k^{(1)})=\mathrm{SFV}(X_3^{(0)})+\mathrm{SFV}(X_k^{(1)}),\quad k\in\{1,2,\cdots,c\} \tag{8}$$

and search i_0 in order that

$$\mathrm{SFD}(X_3^{(0)}\cup X_{i_0}^{(1)})=\min_{k\in\{1,2,\cdots,c\}}\mathrm{SFD}(X_3^{(0)}\cup X_k^{(1)}) \tag{9}$$

If $\mathrm{SFD}(X_3^{(0)}\cup X_{i_0}^{(1)})$ is not greater than threshold b, merge $X_3^{(0)}$ and $X_{i_0}^{(1)}$, the merged initial cluster is still denoted as $X_{i_0}^{(1)}$; otherwise, view $X_3^{(0)}$ as a new initial cluster, denoted as $X_{c+1}^{(1)}$, and the number of clusters is increased by 1, or $c:=c+1$.

Step 4. For $X_i^{(0)}$, $i\in\{4,5,\cdots,n\}$, do similarly as in Step 3.

Step 5. From all the initial clusters $X_k^{(1)}$, $k\in\{1,2,\cdots,c\}$ obtained from above steps, get rid of outlier clusters that contain few objects. The remainder clusters are the final result of CABOSFV clustering.

3.3 Time complexity

For n objects with m attributes, if the number of clusters obtained from CABOSFV clustering is k, we know from the above steps that the time complexity of CABOSFV is $O(nk)$.

We have tested the actual efficiency of CABOSFV clustering and the capability to handle high dimensional sparse data by computer with Pentium 500 CPU and 128 M memory. **Figure 2** illustrates the average calculating time when 10 clusters were generated from 100 and 200 dimensional sparse data with the objects number increasing from 50000 to 100000 by interval 10000. From the figure it is known that the calculating time is about 7 s when the number of objects is 100000.

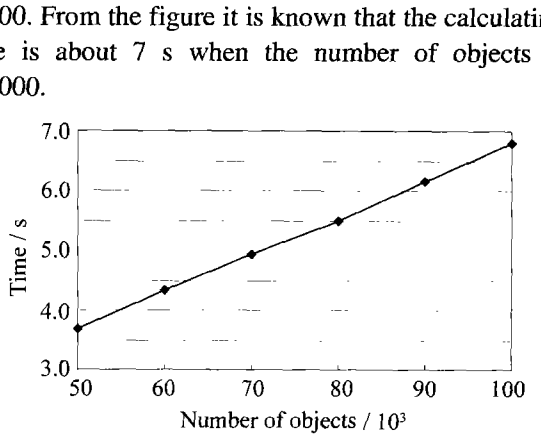

Figure 2 Scalability with the number of data records.

Empirical tests show that CABOSFV is very effective and efficient to handle high dimensional large datasets for sparse data clustering problem.

4 A numeric example of CABOSFV clustering

4.1 Problem statement

Assuming there are 15 customers and the sequence numbers of the customers are denoted as 1,2,…,15. Each customer is characterized by 48 attributes, the orders of 48 products. The sequence numbers of products are indicated by 1,2,…,48. The sequence numbers of products ordered by customers are listed in **table 1**. Now we will group the customers into clusters according to the similarity of their ordering products. This is a clustering problem of 48 dimensions.

Table 1 Product varieties ordered by the customers

Customer	Products ordered by the customer
1	1,3,4,5,6,7,8,10,11,12,22,23,25,26,34,35,36,37,43
2	1,3,4,5,6,7,8,10,11,12,20,21,22,26,28,35,39
3	1,3,6,7,22,24
4	1,3,4,6,7,8,10,22,24,26,29,35,42
5	1,3,4,5,6,8,10,11,15,22,23,26,28
6	1,8,22,23
7	1,3,4,6,7,8,10,11,22,26
8	1,3,4,5,6,8,10,17,18,22,23,28
9	1,3,4,5,8,22,26,28,29
10	1,3,4,6,7,8,10,11,12,14,16,17,22,23,24,26,28,29,30,35,47
11	1,3,4,5,6,8,10,16,18,20,23,28,29,34,48
12	1,3,4,5,6,7,8,10,11,13,22,28,41,45
13	1,3,4,5,6,7,8,10,11,16,19,22,26,28,29,30,35,36,37,43,44,45
14	1,2,3,4,5,8,22,23,24,26,27,28,39
15	1,3,4,5,6,8,10,11,22,26,28

4.2 CABOSFV clustering result

Assuming the SFD threshold of a cluster b=0.5, we get all the initial clusters as shown in **table 2**. Among all the initial clusters, $X_2^{(1)}$, $X_4^{(1)}$, $X_6^{(1)}$, $X_7^{(1)}$ and $X_8^{(1)}$ all contain only one customer. They are outlier clusters and should be removed from final clusters. So the final result of CABOSFV clustering is 3 clusters, which are {1,2,5,12,15}, {4,7,10} and {8,9}.

Table 2 Initial clusters obtained from CABOSFV clustering

Initial clusters	Customers	N	S	NS	SFD
$X_1^{(1)}$	1,2,5, 12,15	5	1,3,4,5,6,8,10,11,22	7,12,13,15,20,21,23,25,26, 28,34,35,36,37,39,41,43,45	0.400
$X_2^{(1)}$	3	1	1,3,6,7,22,24		0
$X_3^{(1)}$	4,7,10	3	1,3,4,6,7,8,10,22,26	11,12,14,16,17,23,24,28, 29,30,35,42,47	0.481
$X_4^{(1)}$	6	1	1,8,22,23		0
$X_5^{(1)}$	8,9	2	1,3,4,5,8,22,28	6,10,17,18,23,26,29	0.500
$X_6^{(1)}$	11	1	1,3,4,5,6,8,10,16,18,20, 23,28,29,34,48		0
$X_7^{(1)}$	13	1	1,3,4,5,6,7,8,10,11,16,19, 22,26,28,29,30,35,36,37, 43,44,45		0
$X_8^{(1)}$	14	1	1,2,3,4,5,8,22,23,24,26,27,28,39		0

Notes: N, S, NS and SFD are four elements of SFV.

5 Conclusions

A new clustering algorithm CABOSFV was proposed. Due to the definition of SFV and the additivity theorem of SFV, CABOSFV is very effective and efficient for high dimensional sparse data clustering with binary attributes. Both theoretical analysis and empirical tests showed CABOSFV clustering has following salient features:

(1) It is very effective for high dimensional sparse data clustering. We have tested the algorithm by dataset of up to 200 dimensions and got the right result.

(2) The time complexity is $O(nk)$, where n is the number of objects, k is the number of clusters.

(3) CABOSFV is very efficient to handle large dataset due to the low time complexity. We have listed in last section that the average computation time is about 7 seconds for 100000 data objects of 100 and 200 dimensions to be grouped into 10 clusters.

(4) It is not sensitive to noise. CABOSFV can identify outlier objects and thus they can be removed. In the above example, customer 3, customer 6, customer 11, customer 13 and customer 14 are all outliers and are assigned into the outlier clusters $X_2^{(1)}$, $X_4^{(1)}$, $X_6^{(1)}$, $X_7^{(1)}$ and $X_8^{(1)}$ respectively, which are removed from the final result.

References

[1] Jiawei Han and Micheline Kamber, *Data Mining Concepts and Techniques* [M], Academic Press, San Diego, 2001.

[2] R. Agrawal, J. Gehrke, and D. Gunopulos, *et al.*, Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, [in] *Proceedings of the ACM SIGMOD Conference on Management of Data* [C], Seattle, 1998, p.94.

[3] S. Guha, R. Rastogi, and K.C. Shim, An efficient clustering algorithm for large databases, [in] *Proceedings of the ACM SIGMOD Conference on Management of Data* [C], Seattle, 1998, p.73.

[4] T.N. Raymond and Jiawei Han, Efficient and effective clustering methods for spatial data mining, [in] *Proceedings of 20th International Conference on Very Large Data Bases* [C], Santiago, 1994, p.144.

[5] M. Ester, H. Kriegel, and J. Sander, *et al.*, A density-based algorithm for discovering clusters in large spatial databases with noise, [in] *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)* [C], Portland, 1996, p.226.

[6] Tian Zhang, R. Ramakrishnan, and M. Livny, BIRCH: an efficient data clustering method for very large databases, [in] *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* [C], Montreal, 1996, p.103.

[7] Wei Wang, Jiong Yang, and R. Muntz, STING: a statistical information grid approach to spatial data mining, [in] *Proceedings of the 23rd VLDB Conference* [C], Athens, 1997, p.186.

[8] W. Wang, J. Yang, and R. Muntz, STING+: an approach to active spatial data mining, [in] *Proceedings of the Internatioal Conference on Data Engineering* [C], Sydney, 1999, p.116.