

A Robust Seedless Algorithm for Correlation Clustering

Mohammad S. Aziz and Chandan K. Reddy

Department of Computer Science,
Wayne State University, Detroit, MI, USA
maziz@wayne.edu, reddy@cs.wayne.edu

Abstract. Finding correlation clusters in the arbitrary subspaces of high-dimensional data is an important and a challenging research problem. The current state-of-the-art correlation clustering approaches are sensitive to the initial set of seeds chosen and do not yield the optimal result in the presence of noise. To avoid these problems, we propose RObust SEedless Correlation Clustering (ROSECC) algorithm that does not require the selection of the initial set of seeds. Our approach incrementally partitions the data in each iteration and applies PCA to each partition independently. ROSECC does not assume the dimensionality of the cluster beforehand and automatically determines the appropriate dimensionality (and the corresponding subspaces) of the correlation cluster. Experimental results on both synthetic and real-world datasets demonstrate the effectiveness of the proposed method. We also show the robustness of our method in the presence of a significant noise levels in the data.

Keywords: Correlation clustering, principal component analysis.

1 Introduction

Clustering is one of the most popular techniques in the field of data mining [9]. The basic idea of clustering is to partition the data in such a way that the members of a partition are closer to each other and the members of different partitions are far apart. But many real-world applications often suffer from the “*curse of dimensionality*” and the measure of nearness becomes meaningless. In such scenarios, many feature selection and dimensionality reduction methods have been proposed to aid clustering [7]. However, these methods reduce the dimensionality by optimizing a certain criterion function and do not address the problem of data clustering directly. The result of using dimensionality reduction techniques for clustering high-dimensional data is far from satisfactory and rarely used in such scenarios. Moreover, it is possible that different clusters lie in different subspaces and thus cannot be identified using any dimensionality reduction or feature selection method (see Fig. 1(a)). To deal with such cases, subspace clustering methods such as CLIQUE [4], ENCLUS [6], SUBCLU [10] etc. have been proposed in the literature (see Fig. 1(b)). These methods attempt to find clusters in different subspaces. Searching every possible subspace is a computationally intensive task due to exponentially large search space. Thus, most of these methods use some form of an Apriori-based approach to identify the most interesting subspaces. Projected clustering [3,12] is one form of subspace clustering that uses the concept of projection. However, these subspace clustering or projected clustering methods have the following two limitations:

1. They are only able to separate clusters that are oriented in axis-parallel manner. They cannot find cluster in arbitrary-oriented subspace (see Fig. 1(c)).
2. They can only find clusters in the sense of locality.

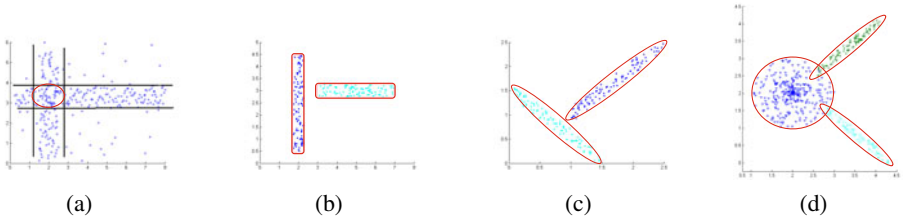


Fig. 1. Different kinds of correlation clusterings. (a) Projected clustering (b) Axis-Parallel correlation clusters. (c) Arbitrary oriented subspace clusters. (d) Arbitrary oriented subspace clusters with different dimensionality.

In subspace clustering, the clusters are formed in the subspaces rather than the full-dimensional space [4,6]. Finding arbitrary-oriented subspace cluster (Fig. 1(c)) involves exponential search space. To efficiently address the problem, most correlation based approaches use PCA to avoid searching unwanted regions of the search space. ORCLUS [2], which uses the same idea of axis-parallel PROCLUS [3], is the first PCA based method to find correlation clusters. 4C [5] also uses PCA, however it incorporates a density-based approach and hence, the number of clusters need not be pre-specified. Although these methods are able to keep the computational complexity low and do not suffer from a potentially infinite search space, their success in finding correlation clusters is highly dependent on the initial choice of the seeds. In addition, they usually do not produce optimal results in the presence of noise. Another problem with these methods is that the dimensionality of the correlation has to be pre-defined which is usually difficult for the end-user from the practical viewpoint. Yip et al. [13] proposed an algorithm called HARP [13], which exploits the data to adjust the internal threshold values dynamically at the runtime. However, this method faces difficulties in obtaining a low-dimensional cluster [11]. To avoid the problem associated with PCA-based methods such as choosing the initial set of seeds and susceptibility to noise, CASH algorithm [1] uses Hough transform to find correlations. Though this method does not use any initial seeds, its worst case time complexity is exponential, thus making it impractical for high-dimensional data. To solve these issues, we propose a novel PCA based algorithm which eliminates the problem of susceptibility to noise and the need for initial seeds to find correlation clusters. It can also simultaneously find correlation clusters of different dimensionality (see Fig. 1(d)) and the computational complexity is relatively low.

In this paper, we propose ROBUSt SEEdless Correlation Clustering (ROSECC) algorithm to find the correlation clusters in high-dimensional data. The main advantages of the ROSECC algorithm compared to the state-of-the-art methods proposed in the literature are:

1. It does not require initial seeds for the clustering and hence it is deterministic.
2. It can simultaneously identify correlation clusters with different number of dimensions.

3. It is robust to handle noise in the data and can obtain the desired correlation clusters despite the presence of significant noise levels in the data.

The rest of this paper is organized as follows: Section 2 describes the necessary definitions and notations. The proposed algorithm along with its computational complexity is described in Section 3. Section 4 outlines the experimental results on both synthetic and real-world datasets. Finally, Section 5 concludes our discussion.

2 Preliminaries

In this section, we will introduce some definitions that are needed to comprehend our algorithm. Our method is based on the projection distance of a point onto the principal vectors. Let $dp \in D$ is a datapoint and v is the principal vector of a member P_i of Partitionset P , then the projection distance of dp and P_i is given by the following equation:

$$PDist(dp, P_i) = \begin{cases} dp.v & \text{if } |P_i| > 1. \\ dist(dp, x) & \text{if } |P_i| = 1. \end{cases}$$

where $dist(dp, x) = \sqrt{\sum_j (dp_j - x_j)^2}$. dp_j and x_j corresponds to the j^{th} feature of the data points dp and x respectively. At the beginning of the generation of partitions, there will be only one single point x without any principal component. In such cases, the Euclidean distance from the point x will be used as the projection distance. A “Partitionset” (denoted by P_i) is defined as a set of datapoints which have lower projection distance to a particular component compared to the projection distance to any other component. A “Partition” is defined as a set of Partitionsets.

Definition 1. (Nearest Component): Let $dp \in D$ and $P = \{P_i\}$ where P_i 's are components, the nearest component of dp from P , denoted by $NComp(dp, P)$, is defined as the component for which the projection distance from dp is minimum. $NComp(dp, P) = \underset{P_i \in P}{\operatorname{argmin}}(PDist(dp, P_i))$. The corresponding distance is the “minimum projection”, denoted by $MProj(dp, P) = \min_{P_i \in P}(PDist(dp, P_i))$

Definition 2. (Farthest Candidate): Let $P = \{P_i\}$ where P_i is a Partitionset. The farthest candidate of P , denoted by $FCand(P)$, is defined as the data point for which the minimum projection distance is maximum. $FCand(P) = \underset{dp \in D}{\operatorname{argmax}}(MProj(dp, P))$

Theorem 1. Let a new component consisting of a single element $\{dp'\}$ is added to the component list P to become P' i.e. if $P' = P \cup \{dp'\}$. If $MProj(dp, P) > dist(dp', dp)$ then $MProj(dp, P') = dist(dp', dp)$

Proof

From the definition of the projection distance, we have $PDist(dp, \{dp'\}) = dist(dp', dp)$. Therefore, $MProj(dp, P) > dist(dp', dp) \Rightarrow MProj(dp, P) > PDist(dp, \{dp'\})$. Also, from the definition of the Minimum Projection, we get, $MProj(dp, P) \leq \forall_{P_i \in P} PDist(dp, P_i) \Rightarrow PDist(dp, \{dp'\}) < \forall_{P_i \in P} PDist(dp, P_i)$

$$\Rightarrow PDist(dp, \{dp'\}) \leq \forall_{P_i \in P'} PDist(dp, P_i).$$

Hence, from the definition of the projection distance, we get

$$MProj(dp, P') = PDist(dp, \{dp'\}) = dist(dp', dp). \quad \square$$

This theorem helps in reducing some of the unnecessary computations during the partitionset assignment step after obtaining a farthest candidate and the corresponding new component of a single datapoint. We can use the minimum projection value directly from the last iteration and compare it to the Euclidean distance from the farthest candidate. This is an important step that saves a lot of computation time.

Definition 3. (*Dimensionality of a cluster*): Let $P = \{P_i\}$ where P_i is a Partitionset and $\lambda_1, \lambda_2, \dots, \lambda_m$ be the eigen values of the Eigen decomposition of that partition set in descending order. The dimensionality of the cluster is defined as $Dim(P_i) = Min_k(\sum_1^k \lambda_i > \tau)$ where τ is a threshold.

When $\tau = 0.9$, 90% of the original variance of the data is preserved and the corresponding Partitionset is obtained as a cluster. The dimensionality of the cluster is the number of eigenvalues that cover most of the variance in the data of the Partitionset.

Definition 4. (*Acceptable Partitionset*): A Partitionset P_i is said to be an acceptable Partitionset, if $|P_i| \geq n_0$ and $\frac{1}{|P_i|} \sum_{dp \in P_i} PDist(dp, P_i) \leq \epsilon$

For a Partitionset to be acceptable, it must satisfy the following two conditions:

1. There must be sufficient number of data points (n_0) in the Partitionset.
2. The points in the correlation cluster must be closer ($< \epsilon$) to its principal component.

3 The ROSECC Algorithm

In this section, we will describe the details of the proposed ROSECC algorithm and also analyze the effect of different parameter values in the algorithm. The overall procedure is shown in Algorithm 1.

3.1 Algorithm Description

The five different steps in the ROSECC algorithm are described below.

Step 1: Finding k_0 principal components: For a given set of data points (*Data*), this step will find k_0 number of components in the data by iteratively generating a sufficient number of partitions and their corresponding principal components. The details of this step are described in Algorithm 2. Initially, the principal components are computed with the entire dataset. Then, the *Farthest Candidate* from the first principal component is taken and the dataset is partitioned into two different subsets depending on the distance of the data point to the previous principal component vector and the new seedpoint. PCA is independently applied on these two subsets, thus generating the two corresponding principal component vectors. The membership is updated and this process is repeated until k_0 number of components are obtained (see Fig. 2(a)). At this point, one might encounter any of these following scenarios:

1. Some principal components are obtained for data points that are not correlated at all. These are outliers or noise points.
2. There might be some components that contain more than 1-dimensional correlation.
3. More than one component and corresponding points might be obtained for the same cluster.

In the next three steps, these three situations are handled.

Step 2: Removing non-informative principal components: The components that may not represent a correlation cluster are removed in this step. A component is accepted for further consideration only if it satisfies the definition of acceptability given in the previous section. When the unacceptable components are removed, the corresponding set of points become members of the outlier set (O), which can then become a member of any other cluster in the later stages (see Fig. 2(b)).

Algorithm 1. $ROSECC(Data, k_0, k_1, n_0)$

- 1: **Input:** Data matrix ($Data$), Number of Partitionsets (k_0), Number of iterations (k_1), Threshold for datapoints (n_0)
 - 2: **Output:** Set of Correlation Clusters (P)
 - 3: **Procedure:**
 - 4: $P \leftarrow Generate_Components(Data, k_0, k_1)$
 - 5: $[P, O] \leftarrow Remove_Unacceptable_Components(P, n_0)$
 - 6: $[P, O] \leftarrow Include_Possible_Datapoints(O, P)$
 - 7: $P \leftarrow Merge_Possible_Components(P)$
 - 8: Return P
-

Step 3: Finding the dimensionality of the correlation: Though Step 1 successfully produces the set of components that possibly represent some correlation, identifying the dimensionality of the correlation is one of the key aspects of the algorithm. Practically, the correlation cluster lies in a low-dimensional space. Hence, we can assume that a high-dimensional correlation in the data will contain a one dimensional correlation with at least a fewer number of those datapoints. Correlation dimensionality of an acceptable component P (denoted by $Dim(P)$) is the number of eigenvectors needed for preserving the desired amount of variance.

Step 4: Adding points to the correlation cluster: This step assumes that there are some acceptable components and a set of outliers obtained from step 1. It will also consider each outlier and tries to assign them to any one of the current components. If a datapoint cannot be assigned to any of the correlation clusters, then it remains to be an outlier. A data point dp in the outlier set O is includable to P_i , if $PDist(dp, P_i)$ is less than a threshold η (see Fig. 2(c)). *At this point, the mutual exclusiveness of the component does not hold and hence the algorithm will produce overlapping clusters.*

Step 5: Merging the correlation cluster: In this step, each component is checked for any possibility of merging with another component. Two components will be merged if

Algorithm 2. *Generate_Components(Data, k_0, k_1)*

```

1: Input: Data matrix (Data), Number of Components ( $k_0$ ), Number of iterations ( $k_1$ )
2: Output: Partition Set (P)
3: Procedure:
4:  $P = Data, k = 0$ 
5: repeat
6:    $k = k + 1; dp' = FCand(P); newp = \{dp'\}; P = P \cup newp$ 
7:   for each  $dp \in D$  do
8:     if  $MProj(dp, P) > dist(dp', dp)$  then
9:        $Comp(dp) = Comp(dp) - \{dp\}$  //Reassignment of cluster membership
10:       $newp = newp \cup \{dp\}$  //when the new partition is created
11:     end if
12:   end for
13:   for  $i = 1$  to  $k_1$  do
14:     for each  $dp \in D$  do
15:       for each  $P_i \in P$  do
16:         if  $MProj(dp, P) > PDist(dp, P_i)$  then
17:            $Comp(dp) = Comp(dp) - dp$ 
18:            $P_i = P_i \cup dp$ 
19:         end if
20:       end for
21:     end for
22:   end for
23: until  $k = k_0$  //Each iteration generates a new partition

```

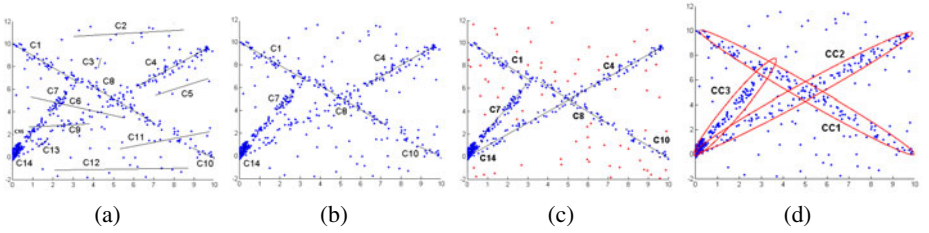


Fig. 2. Different steps of the proposed ROSECC algorithm. (a) Generation of a Partition-set: 15 partitions (C1-C15) are generated. (b) Removal of unacceptable Partitionsets: Only C1, C4, C7, C8, C10 and C14 remain after this step. (c) Addition of datapoints: the remaining points (points colored in red) are outliers. (d) Merging of Partitionsets: C1, C8 and C10 are merged to CC1; C4 and C14 are merged to CC2.

the coefficient of their representative vectors does not differ significantly. i.e. two components P_i and P_j are merged, if $|c_{il} - c_{jl}| < \delta, \forall l$ (see Fig. 2(d)).

The time complexity of our algorithm is $O(k_0^3 + Nd^2)$, which is typically dominated by the Nd^2 term for large-scale and/or high-dimensional data. This time complexity of the ROSECC algorithm is competitive to the other state-of-the-art techniques.

Table 1. Effect of the change in parameter values for the DS1 dataset

k_1	3				5			
n_0	1%-3%				1%-3%			
k_0	5	10	15	20	5	10	15	20
Accuracy	51.45%	70.34%	87%	87.35%	61.32%	95%	100%	100%

3.2 Tuning of the Parameters

The parameter k_0 is the number of components that are to be generated in the first step of the algorithm. This value should be at least twice the number of correlation clusters present in the data. The second parameter k_1 is the number of iterations that the algorithm runs for convergence with the new partition. From our experiments, we observed that very few iterations (fewer than 6) are necessary for convergence. The third parameter n_0 is required to check the acceptability of a component and should be atleast 1% of the number of data points. In Table 1, we show the effect of different parameter values for a synthetic dataset (DS1). Since we know the ground truth, we presented the accuracy as the percentage of correctly clustered datapoints. We can see that k_1 is good with value as low as 5, while the value of n_0 can be between 1% and 3%. Results are not optimal when $k_0 = 5$, but $k_0 = 15$ or more gives an accurate result. Hence, it is important to generate more number of components in the first step of the algorithm.

4 Results and Discussion

4.1 Synthetic Datasets

Our algorithm was tested successfully on various synthetic datasets that were generated to see the different aspects of the ROSECC algorithm. We discuss two different synthetic datasets and explain the data generation along with the corresponding results.

(1) **DS1** : In this dataset, two one-dimensional correlations containing 200 data points generated from the origin with a slope of 1 and -1. Another correlation cluster (with 100 data points) is generated from the origin with a slope of 1.4. 100 random noise points are also added. The ROSECC algorithm found all the correlation clusters, even in the presence of sufficient noise and significant overlapping of the clusters (see Fig. 2(d)). Thus it shows that our algorithm is robust and only starts to break when the noise is greater than 20% in the data (see Table 2).

(2) **DS2**: In this dataset, one two-dimensional correlation cluster (circular shape) with 300 data points is generated. Two 1-dimensional correlation clusters that start from the edge of that cluster in two different directions, one with 150 datapoints and the other with 100 datapoints is added to the first cluster (see Fig. 3). This dataset is created to test the ability of the ROSECC algorithm to simultaneously identify the correlation clusters with different dimensionality which is one of key challenges in the correlation clustering problem.

Table 2. Robustness of ROSECC: Accuracy is measured in presence of different noise levels

Noise percentage	5%	10%	15%	20%	25%	30%	35%
Accuracy	100%	100%	100%	100%	93%	85%	70%

We compared the result of the ROSECC algorithm with the well-studied DBSCAN [8] and 4C [5] algorithms. In this dataset, all the datapoints are density connected (see Fig. 3). There are two 1-dimensional and one 2-dimensional correlation cluster in this dataset. As shown in the result, DBSCAN recognizes all the data points as a single cluster (Fig. 3(a)). Algorithm 4C recognizes either 1-dimensional clusters (Fig. 3(b)) or the 2-dimensional cluster (Fig. 3(c)) depending on the choice of the parameter corresponding to the number of dimensionality. On the other hand, the ROSECC algorithm was able to successfully identify all the three clusters simultaneously in a single run (Fig. 3(d)).

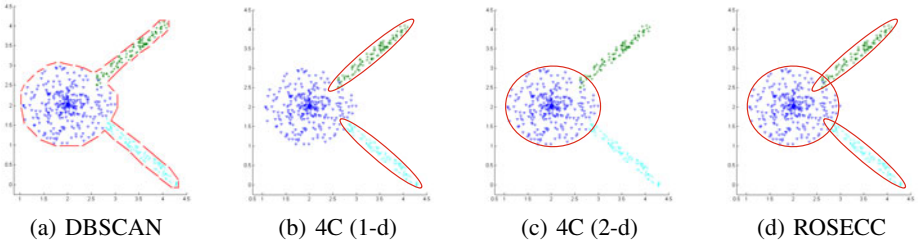


Fig. 3. Comparison results: ROSECC(d) finds all the desired correlation clusters in a synthetic dataset (DS2) where both DBSCAN(a) and 4C(b and c) fail

4.2 Real-World Datasets

We also show the performance of the ROSECC algorithm in finding correlation clusters in the following three real-world datasets.

(1) **Breast Cancer dataset:**¹ This dataset measures nine biomedical parameters characterizing breast cancer type in 683 humans. The algorithm found one 3-dimensional cluster and two 2-dimensional clusters. When the result is projected in a 3-D space using PCA, we found that the 3-dimensional correlation cluster is a pure cluster of benign cancer (see Fig. 4) and no malignant cancerous patient belongs to any correlation cluster and are considered to be outliers by our algorithm.

(2) **Wages dataset :** The wages dataset contains the statistics of the determinants of Wages from the 1985 Current Population Survey. It contains 534 observations on 11 features sampled from the original Current Population Survey of 1985 and can be

¹ <http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/>

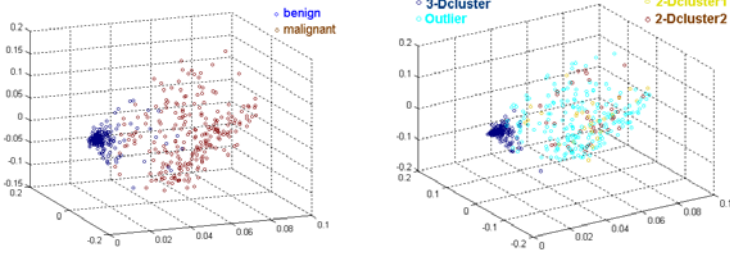


Fig. 4. Projection of the breast cancer dataset onto the 3-dimensional space using PCA. (a) Original class labels and (b) Result of the ROSECC algorithm.

downloaded from StatLib Data archive². ROSECC gives one 2-dimensional correlation (see Fig. 5) which basically gives $YE(\text{Years of Education}) + WE(\text{Years of work experience}) + 6 = \text{Age}$. It also gives four 1-dimensional correlation clusters;

- (i) $\text{Age} = WE + 18, YE=12$
- (ii) $\text{Age} = WE + 20, YE=16$
- (iii) $\text{Age} = WE + 24, YE=18$
- (iv) $\text{Age} = WE + 24, YE=14$.

However, all of the data points of these four clusters are also the members of the 2-dimensional clusters, which suggests that these four 1-dimensional correlations are actually on the 2-dimensional plane of the 2-dimensional cluster. This is an interesting example because there are quite a few data points in the 1-dimensional clusters and our algorithm identifies those as separate clusters as well.

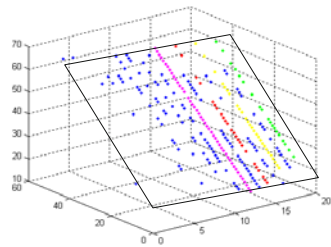


Fig. 5. ROSECC identifies the clusters in the Wages dataset

(3) **Glass Identification dataset:** The glass identification dataset³ consists of 214 samples and 9 attributes measuring the refractive index and weight percentage of different metals in the corresponding oxide in the glass. We ran our algorithm to find the clusters in this dataset and the wdbc dataset. We measure the performance of the ROSECC algorithm by using the class label as ground truth and calculating the F1 measure. Table 3 shows the results of the ROSECC algorithm in both the datasets are better than the other state-of-the-art methods.

Table 3. Experimental results of F1 measure for real-world datasets

Dataset	CLIQUE [4]	SUBCLU [10]	PROCLUS [3]	ROSECC
Glass	0.45	0.49	0.57	0.58
WDBC	0.42	0.43	0.47	0.49

² http://lib.stat.cmu.edu/datasets/CPS_85_Wages

³ <http://archive.ics.uci.edu/ml/datasets/Glass+Identification/>

5 Conclusion

We proposed ROSECC, a novel PCA based robust seedless correlation clustering algorithm. ROSECC does not require the initial set of seeds for clustering and is robust to noise in the data compared to the other PCA based approaches which typically require initial seeds and a pre-defined dimensionality parameter. It incrementally partitions the data space and eventually finds the correlation clusters in any arbitrary subspace even in the presence of overlapping clusters. Using several synthetic and real-world datasets, we demonstrated the advantages of the ROSECC algorithm compared to the other methods available in the literature for identifying correlation clusters.

References

1. Achtert, E., Böhm, C., David, J., Kröger, P., Zimek, A.: Robust clustering in arbitrarily oriented subspaces. In: Proceedings of SIAM International Conference on Data Mining (SDM), pp. 763–774 (2008)
2. Aggarwal, C., Yu, P.: Finding generalized projected clusters in high dimensional spaces. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 70–81 (2000)
3. Aggarwal, C.C., Wolf, J.L., Yu, P.S., Procopiu, C., Park, J.S.: Fast algorithms for projected clustering. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 61–72 (1999)
4. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 94–105 (1998)
5. Bohm, C., Kailing, K., Kroger, P., Zimek, A.: Computing clusters of correlation connected objects. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 455–466 (2004)
6. Cheng, C., Fu, A.W., Zhang, Y.: ENCLUS: Entropy-based subspace clustering for mining numerical data. In: Proceedings of the ACM conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 84–93 (1999)
7. Ding, C.H.Q., He, X., Zha, H., Simon, H.D.: Adaptive dimension reduction for clustering high dimensional data. In: Proceedings of the IEEE International Conference on Data Mining, pp. 147–154 (2002)
8. Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the ACM conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 226–231 (1996)
9. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
10. Kailing, K., Kriegel, H., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: Proceedings of SIAM International Conference on Data Mining (SDM), pp. 246–257 (2004)
11. Kriegel, H., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3(1), 1–58 (2009)
12. Yip, K.Y., Cheung, D.W., Ng, M.K.: On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In: Proceedings of the 21st International Conference on Data Engineering (ICDE), pp. 329–340 (2005)
13. Yip, K.Y., Ng, M.K.: Harp: A practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 16(11), 1387–1397 (2004)