An efficient algorithm to solve high-dimensional data clustering: Candidate subspace clustering algorithm

Chin-Chieh Kao

AN EFFICIENT ALGORITHM TO SOLVE HIGH-DIMENSIONAL DATA CLUSTERING - CANDIDATE SUBSPACE CLUSTERING ALGORITHM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Chin - Chieh Kao

September 2013

AN EFFICIENT ALGORITHM TO SOLVE HIGH-DIMENSIONAL DATA CLUSTERING - CANDIDATE SUBSPACE CLUSTERING ALGORITHM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Chin - Chieh Kao

September 2013

Approved by:

[redacted]

Haiyan Qiao, Advisor, School of Computer Science and Engineering

8/28/2013
Date

[redacted]

Ernesto Gomez

[redacted]

Kerstin Voigt

ABSTRACT

The task of high dimensional data clustering is widely demanded in a variety of applications but very challenging due to the curse of dimension. When dimensionality increases, the distance metric usually becomes inaccurate while measuring the similarity between objects. Therefore most existing clustering algorithms only work in low dimension space.

In this project, a comprehensive literature review on high dimensional data clustering is conducted and a novel density-algorithm to perform high dimensional data clustering is developed. A new data clustering algorithm called Candidate Subspace clustering algorithm is an efficient algorithm to find all meaningful clusters in subsets of high dimensional space. Candidate Subspace clustering algorithm is able to detect arbitrarily shaped and positioned clusters in the specific subspace; it also improves the performance time as well.

Candidate Subspace clustering algorithm conducts clusterings in the agglomerative way. It effectively explores the subspaces and merges the clusters in n-Dimension into a $(n+1)$-Dimension subspace. Candidate Subspace clustering algorithm is used to test with two data sets: Iris data set and Seeds data set, which are widely used as bench market data in data clustering. Furthermore, a 5-D data that is made with 4-D Iris data and 1-D noise is used to test the validity of the algorithm. All the experimental results prove the effectiveness of the algorithm.

Matlab is used for the implementation of the proposed algorithm due to it's power of matrix computation and visualization.

ACKNOWLEDGEMENTS

I have made hard efforts in this project. However, it would not have any possibilities without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Dr.Qiao for her guidance and constant supervision. Whenever I have run into any problems, she is always willing to help me and offer me some valuable advice. Furthermore, she is very patient to help me whenever I get stuck somewhere on my project. Without her help, I could not accomplish the project.

I would also take this opportunity to express a deep sense of gratitude to Dr. Gomez and Dr. Voigt, for their guidance, which helped me in completing this project.

I am obliged to staff members of California State University San Bernardino, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my project.

I would like to express my gratitude towards my parents for their kind and unwavering support and encouragement of my education. I would like to thank Lily as well, she is always there with me no matter what happens.

My thanks and appreciation also go to my friends who have helped me out with their abilities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Overview

We are living in a world with data. We are exposed to a large amount of information in our daily life. We have to process the information autonomously, for example, to sort the information based on some relationships. How to quickly get the relevant information has always been a significant issue. Various techniques have been developed for this purpose. One method to process these data is to distinguish them into different groups, so that the similar data are grouped together. This type of activity has been done for a long time by human beings. Grouping data based on similarity by using computers is called data clustering.

Data clustering techniques are the methods to identify clusters given a data set, so that similar data are likely to be within the same cluster. The objective of data clustering is to identify clusters, and ideally the intro-cluster distance is minimize, and inter-cluster distance is maximized. People always confuse data clustering and classification, but there is some difference between the two. In classification, the task is to assign an object to the existing classes which already know. For example, to put a banana into group yellow, and put a strawberry into group red. In data clustering, the classes are not given, in other words, the data or objects are not labeled with classes. For example, given a group of fruits, we know the features of each fruit, but not the name. It is the task of data clustering to group the fruits based on similarity. The fruits could be grouped by colors or by species. In other words, data clustering is a technique that we use to discover hidden patterns. A good clustering method helps to divide data sets into clusters, where objects in the same cluster are compact, and objects in different clusters are distant from each other. Thus we can recognize the key element and analyze the information of the clusters easily.

Nowadays, data clustering has been widely used in different fields of life. In scientific imaging area, there is a major application of the clustering technique, such as gene data. Gene data clustering provides a powerful tool for studying relationships of genes. By using cluster analysis of the degree of similarity of genetic data between individuals and groups to infer population structures and assign individuals to groups that often correspond with their self-identified geographical ancestry.[4] Clustering is used to build groups of genes with related expression patterns (also known as coexpressed genes). Usually groups contain functionally related proteins, such as enzymes for a specific pathway, or genes that are co-regulated. For example, by using expressed sequence tags (ESTs) or DNA microarrays can be a powerful tool for genome annotation.[9] Cluster analysis is also widely used in market research. Market researchers use cluster analysis to understand the relationships between different groups of consumers/potential customers. Clustering can also be used to group all the shopping items available on the web into a set of unique products. For example, all the items on eBay can be grouped into unique products. In social science, cluster analysis can be used to identify the area where there are particular types of crime. By identifying these distinct areas or "hot spots", government can give warnings to citizens or tourists, to inform them to pay attention to their properties, or stay at home late at night for their own safety. Besides, where a similar crime has happened over a period of time, it is possible to manage law enforcement resources more effectively, etc.[9]

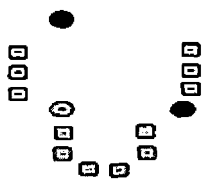
1.2 Current Work

Data clustering technology has a long history and there are many clustering methods available to use today.[10] Nowadays, it has been developed in many kinds of clustering technology. The choice of a particular method will depend on what type of output needed. In general, current clustering technology methods can be simplify as: [14]

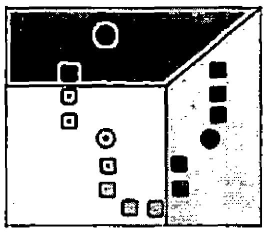
- Partition clustering algorithms, such as K-means and CLARA.
- Hierarchical clustering algorithms, such as BIRCH, CURE and CHAMELEON.
- Density-based clustering algorithms, such as DBSCAN.
- Grid-based clustering algorithms, such as STING and WaveCluster.

Partitioning clustering methods generally result in a set of "M" clusters with each of the objects belonging to one cluster, where the number of clusters "M" is given. Each cluster may be represented by a centroid or a cluster representative. The advantage of partitional methods is that they can be used for large data sets. One problem of partitional algorithms is the choice of "M". Compared with paritional methods' pros and cons, the hierarchical clustering methods are the most commonly used. The advantages of hierarchical clustering come at the cost of lower efficiency. This outputs a hierarchy, a structure that is more informative than the unstructured set of clusters. The primary disadvantage for hierarchical clustering is that it is only effective at splitting data from small amounts of data. When the data set is small, patterns and relationships between clusters are easily grouped. As the data grows, this disadvantage usually results in the loss of information. In density-based clustering, clusters are defined by density. The advantage of density-based clustering is that it can produce any arbitrary shapes. However, it cannot cluster data sets well with large differences in densities. Grid-based clustering are popular for mining clusters in a large multidimensional space where clusters are regarded as denser regions than their surroundings. The main advantage of grid-based methods' is their fast processing time which depend on the number of cells in each dimension. One problem of this method is that all clusters' boundaries are either horizontal or vertical. If no boundary is detected, an error will occur.

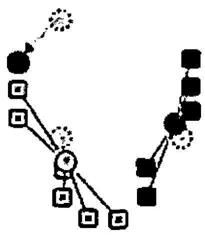
K-means is a typical partition clustering algorithm, and it is one of the most well-known methods for data clustering. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters). The goal of the K-means clustering algorithm is to represent the data set in a certain group very well to find k points and become a centroid of a data set, see Figure 1.1.



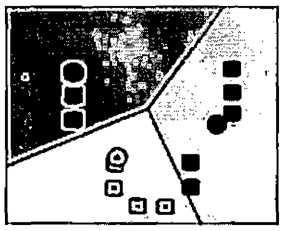
1. In this example, k = 3. Red, Green and Blue.



2. k clusters are created with the nearest mean.



3. The centroid of each k clusters becomes the new mean.



4. Repeat steps 2 and 3 until convergence has done.

Fig. 1.1: The basic clustering steps for K-means.

These centroids (Figure 1.1-1) should be chosen carefully because different locations might cause different results. The better choice is to place them as far away as possible from each other. These k points are also known as cluster centers, prototypes, centroids, or codewords, and so on. After figuring out these cluster centers, the next step is to do data compression and data classification. However, when using K-means algorithms to do data clustering, the clusters' quality can be easily affected by noise or outliers. The pseudo code of the K-means clustering algorithm is shown in Table 1.1.[6]

Tab. 1.1: The pseudo code of K-means clustering algorithm.

```
1. Place K points into the space represented by the objects that are
   being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the
   K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move.
   This produces a separation of the objects into groups from which
   the metric to be minimized can be calculated.
```

Another method to do clustering uses density-based clustering algorithms, such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN). DBSCAN algorithm use a density concept to separate each data and group them into groups based on density connected. There are two parameters that have to know and setup before using the algorithm, the setting radius (eps) and the minimum number of points required to form a cluster (minPts). At the beginning, finding an arbitrary starting point that has not been visited. If it is contents-neighborhood and reaches minPts, a cluster is started. Otherwise, the point will be labeled as a noise. Besides, if a point is found to be a dense part of a cluster, its neighborhood is also be part of that cluster. Keep doing these process until every point have visited and be labeled, which means the density-connected cluster is completely found. Moreover, the cluster contain three kinds of pint, they are core point, border point and noise point. A point is a core point if it has more than specified number of points (minPts) within radius (eps). A border point has fewer than minPts within radius (eps) but is in neighborhood of a core point. A noise point is any point that is not count as a core point or a border point.

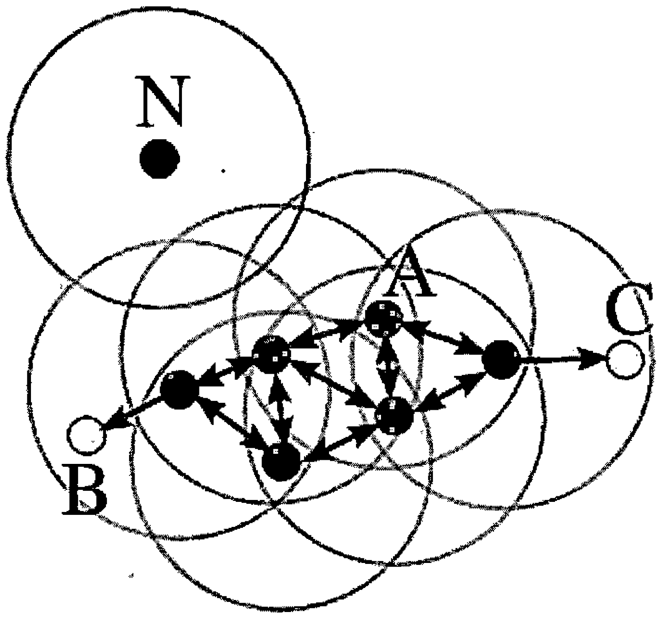


Fig. 1.2: The basic steps for DBSCAN clustering.

In Figure 1.2, point A is the core point, point B, C are labeled as border point. Point B and C are density reachable from point A, so it can be grouped as a cluster. Point N is a noise point. The algorithm is composed in Table 1.2. [2]

Tab. 1.2: The pseudo code of DBSCAN clustering algorithm.

```
Repeat
    1. To determine if the input object is centroid point or not.
    2. Find all objects which based on centroid point which are eps density
       reachable points.
Until compare with all objects

Repeat
    1. If all centroid points in eps reach the minpts, become a cluster.
    2. If two points in each two clusters are density reachable, combine as one
       cluster.
Until compare with all density reachable points which based on centroid point
```

1.3 High Dimensional Data Clustering - Challenge

Clustering technology is a useful starting point for analysis data set. However, when data's attribute becomes more and more, it becomes a high dimensional data set (usually one dimension represent one attribute) and it will become more challenge doing data clustering. "The approaches to clustering high dimensional data must deal with the ""curse of dimensionality"" ".[16] Richard Bellman says : "In view of all that we have said in the forgoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from the earliest days."[14] Which means, when dimension increases, the number of grids will increases exponentially. In other words, in general, data analysis techniques or data clustering works well at lower dimensions but often perform poorly when the dimensionality increases.

The main challenge[16] for high dimensional data clustering is that the effect of increasing dimensionality on distance or similarity. Generally speaking, most clustering technique is based on distance or similarity, and see how objects close to each other and become a cluster, otherwise, the cluster will become meaningless. "In high dimensional spaces, distances between points become relatively uniform".[15] For data distributions, look at Eq. 1.1.

$$\lim_{d \to 1} \frac{MaxDist - MinDist}{MinDist} = 0 \tag{1.1}$$

The difference of distance between closest and farthest object will become 0 when dimensionality has been increased. In the case, the notion of the nearest neighbor of a point is meaningless. In other words, the distance between the farthest object and the nearest object converges to 0 for increasing dimensionality "d". To explain this phenomenon, consider the difference between *MaxDist* and *MinDist*. The equation means difference between the closest and farthest neighbor based on the distance measurement. By experiments on simulated and real data sets, in L1 metric, (*MaxDist* - *MinDist*) increases with dimensionality. In L2 metric, it remains relatively constant. For Lr metric ($r \geq 3$), (*MaxDist* - *MinDist*) goes to 0 as dimensionality increase. To understand this more, take a look at L1,L2 and Lr metric. In Eq. 1.2,

$$P_{ij} = (\sum_{k=1}^{d} |X_{ik} - X_{jk}|^r)^{1/r} \tag{1.2}$$

where "r" is a parameter, "d" is the dimensionality of the data object, "X_{ik}" and "X_{jk}" are respectively, and the k^{th} known as the i^{th} and j^{th} objects of "X_i" and "X_j". When $r = 1$, the distance is called L1 norm or city block distance. If $r = 2$, it is the most common distance, which called L2 norm or Euclidean Distance. For previous discuss, one of the problem for doing high dimensional data set is in high dimensional spaces, distances between objects become relatively uniform. Besides, this problem can be the basic and fundamental issue when local neighborhood dismissing in the clustering process at high dimensionality.

Similarity can also be one of the problem in high dimensional data clustering.[15] In higher dimensional cluster, there are usually contain hundred of thousands attributes from the data set, there may be some correlations among subsets of attributes. Different subsets of features are relevant for different clusters, which means there would be clusters in the particular subspace, and it could be many clusters at the same subspace. It is hard to tell which feature (data) is belong to which cluster. Furthermore, different correlations among the attributes may be relevant to different clusters. For example, text document. When clustering technique used on documentation,[14] such as finding an group thematically related to articles from the web page. Usually the data contains thousands of attributes and it is very sparse. To imagine that text document looks like high dimensional feature vector. The related documents will only have a similar word counting frequency in the subspace, and these subspace may be different for different groups of thematically relevant documents. Moreover, one document may be assigned to more then one group according to the similarities, which means it may overlap. For example, document A may share the same similarity with document B, but not document C in the given set X. However, in set Y, document A may share the same similarity with document C, but not document B.

Another problem is to project points from a higher dimensional subspace to a lower dimensional subspace.[16] Generally speaking, when doing projection from higher dimensional data to lower dimensional data, it can remove the noise. On the other hand, some of the data or important information may be missing. This problem may occur if there is a local noise but not global noise. When using projection technics to analysis the data,[12] given sets of attributes will become different clusters, which means different clusters only exists in different subsets of attributes. This kind of problem is called "local feature relevance problem".[11]

By using a three-dimensional data set as an example to explain this problem. In Figure 1.3, there are four clusters in figure (1a), when doing projection on the subspace A, the relevant attributes are x, y (figure (1b)). Project on the subspace B,

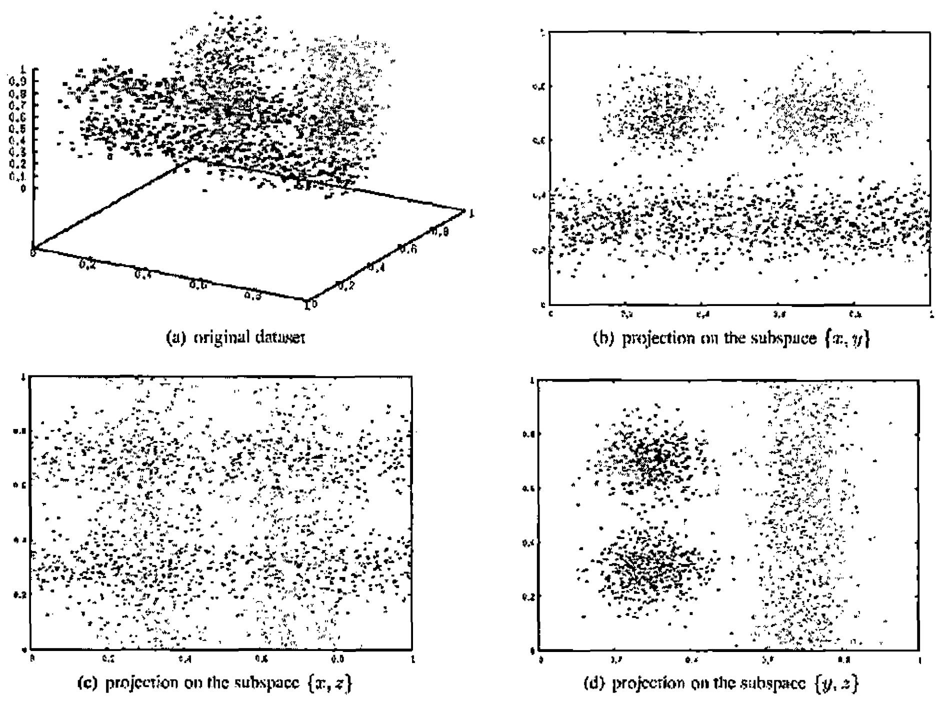


Fig. 1.3: Different way of projective cluster

the relevant attributes are x, z (figure (1c)). Project on the subspace C, the relevant attributes are y, z (figure (1d)). The clusters will occur in the corresponding projected subspace, but will not maintain the same cluster in the global space or even other subspaces. When lower dimensional clusters has occurred, it will also occurred in higher dimensional ones. The challenge for doing data clustering[14] which related to projection problem, is that hard to find appropriate subset of attributes to describe the similarity of objects belonging to the same group and hard to tell what different subsets of attributes are for different groups of objects.

2. RELATED WORK AND CONTRIBUTIONS

2.1 Recent Approaches for High Dimensional Data Clustering

Since the discussion on high dimensional data clustering's challenge in previous chapters, nowadays many possible solutions and algorithms has been designed to solve the specific problem. A possible solution for distance problem[10] is to provide a special distance equation for each problem which they want to solve. Thus, for the object which not relying on the locality assumption may not be affected by the problem of meaningless distance comparisons.

The other way is using Principal Component Analysis (PCA)[12] to solve the problem of projection clustering. To do the dimension reduction, may easily remove the noise, however, to keep the main or priority attribute is the most important subject.

Moreover, both feature selection and dimension reduction method based on PCA may be inappropriate if different clusters exist in different subspaces. Since the example using three-dimensional data set and it already presents a problem, what if the dimension grows to ten-dimensions, even a hundred-dimensions? The problem will occur with higher probability with increasing data dimensionality.[13]

2.2 Contributions

As discussed in previous chapters, although people provides many solutions and algorithms tried to solve the specific problem,[13] doing data clustering in high dimensional data is still a big challenge. Furthermore, there is still not a perfect and efficient algorithm have been designed in high dimensional data clustering. Base on previous observations and discussions, how to use a more efficient algorithm to solve high dimensional clustering problem instead using specific algorithm for specific problem? For example, maybe using one algorithm to solve projection and similarity problem at the same time. How to make the algorithm works more efficient and easier to understand ? How to avoid exhaustive search? In this project, a new clustering algorithm which eliminates the problems mentioned above and enables the user to gain all the clustering information contained in high-dimensional data. Instead of working from full dimension space, the new algorithm start from low dimension space, and find clusters in high dimension space. This project will require the knowledge of K-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN)[19] algorithms, and try to solve the exhaustive problem which discussed above. Moreover, the new algorithm which is called Candidate Subspace clustering algorithm also use the idea of Apriori rule.[18] Base on Apriori rule, the cluster will survive in (n) subspace only when it is existing in (n-1) subspace, otherwise the cluster will not occur in (n) subspace.

Instead of calculating all objects from the original data, finding the candidate_object for the corresponding candidate_space can have better performance. This has the following advantages:

- Candidate Subspace clustering algorithm is able to detect arbitrarily shaped and positioned clusters in the specific subspace.
- Without calculating all the objects from original data, Candidate Subspace clustering algorithm uses candidate_object corresponding to the candidate_space, it will work in high-dimensional data.
- The clustering results makes it easy to see which specific subspace has clusters, and which clusters exists in which subspace.

Candidate Subspace clustering algorithm is a type of agglomerative hierarchical clustering. Based on subspace (n), and find clusters in subspace (n+1). For using Iris data set as an example (it's a 4-D data set), to find cluster in 4-D based on subspace 3-D, cluster in 3-D based on subspace 2-D, etc. Furthermore, when finding if any subspace has a cluster or not, or want to know which specific cluster exists in which subspace, it can easily figure out the result by using Candidate Subspace clustering algorithm. For example, Candidate Subspace clustering algorithm can find clusters in each subspace. Figures 2.1, 2.2, 2.3, 2.4, 2.5 and 2.6 shows the cluster result in 2-D subspace. Figures 2.7, 2.8, 2.9 and 2.10 shows the cluster result in 3-D subspace. The cluster result in 4-D subspace will be discussed in next chapter.

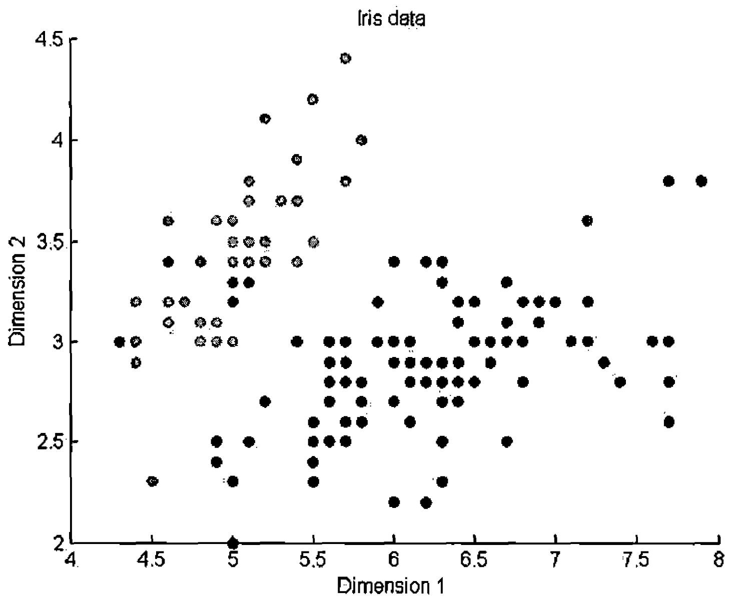


Fig. 2.1: Cluster result for Iris data set in subspace{1, 2}

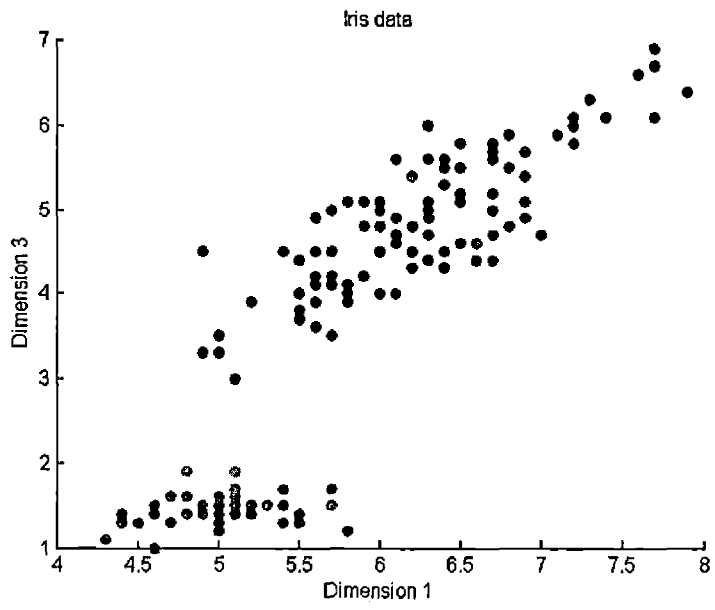


Fig. 2.2: Cluster result for Iris data set in subspace{1, 3}

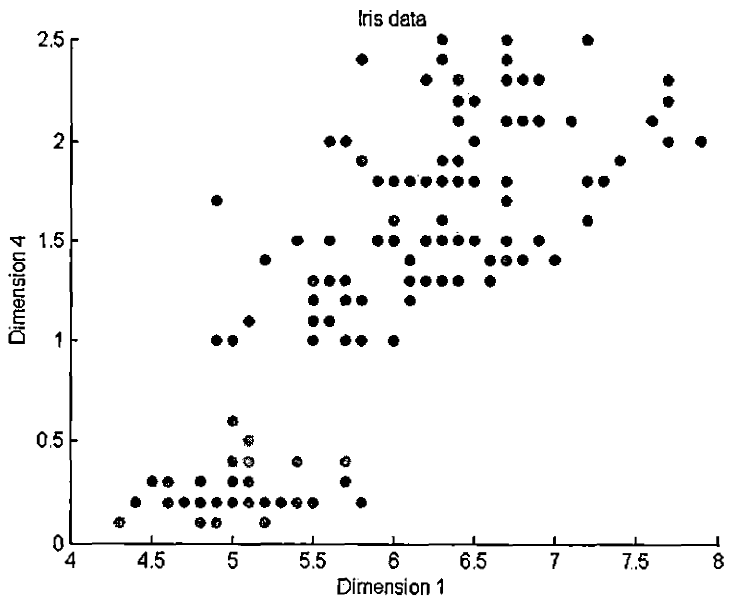


Fig. 2.3: Cluster result for Iris data set in subspace{1, 4}

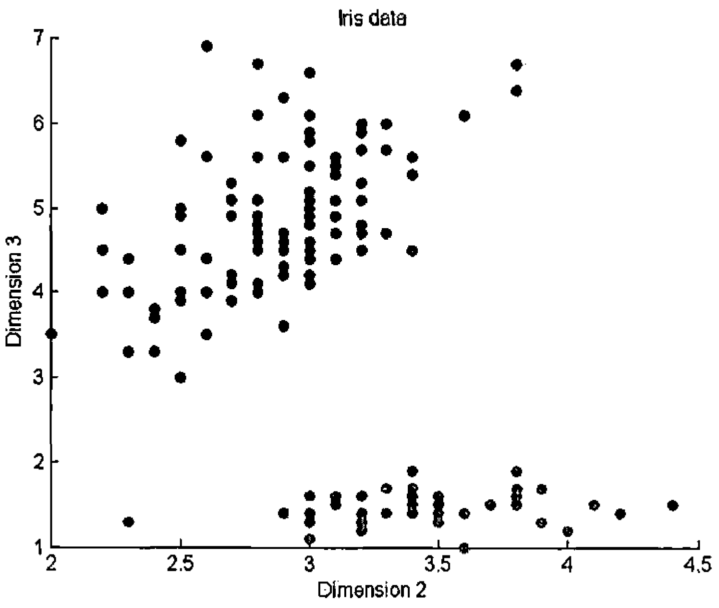


Fig. 2.4: Cluster result for Iris data set in subspace{2, 3}

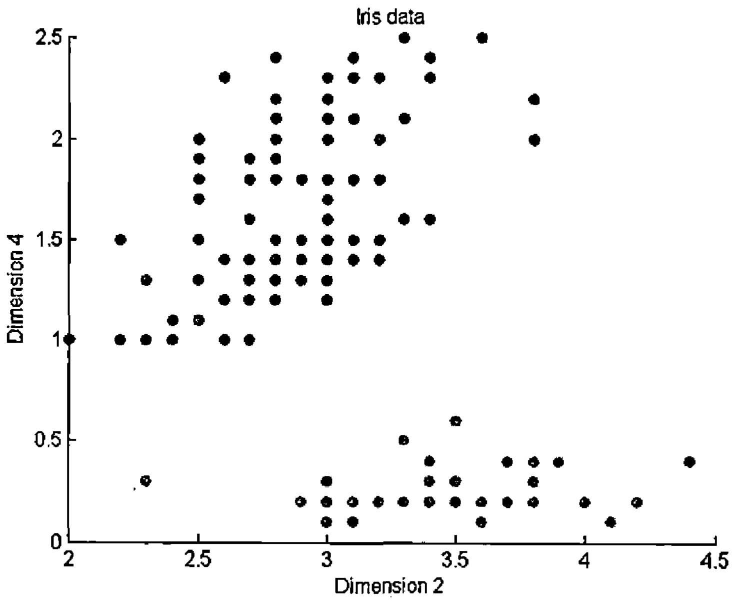


Fig. 2.5: Cluster result for Iris data set in subspace{2, 4}

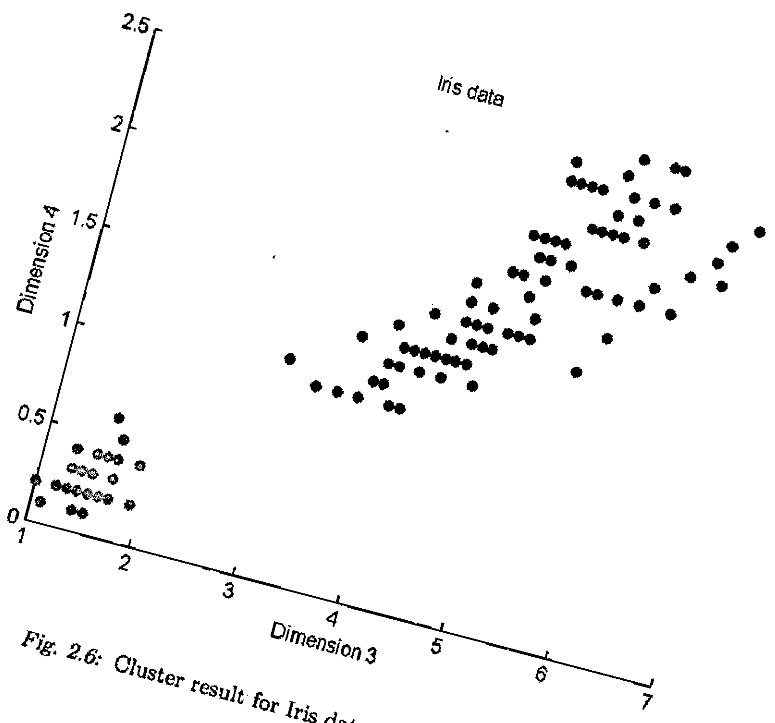


Fig. 2.6: Cluster result for Iris data set in subspace{3, 4}

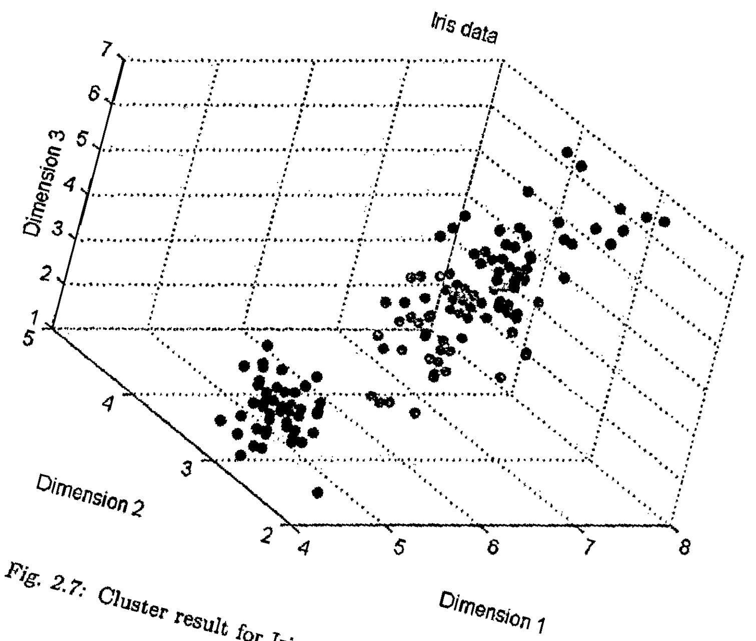


Fig. 2.7: Cluster result for Iris data set in subspace{1, 2, 3}

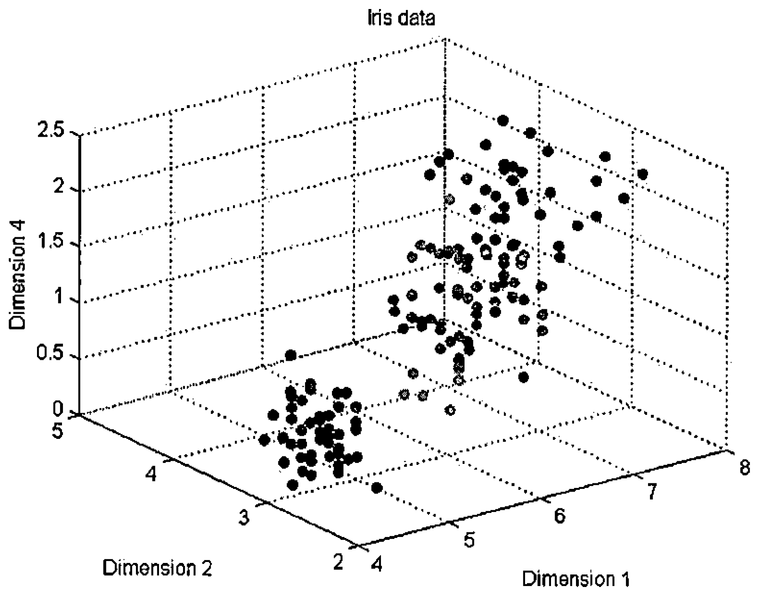


Fig. 2.8: Cluster result for Iris data set in subspace$\{1, 2, 4\}$

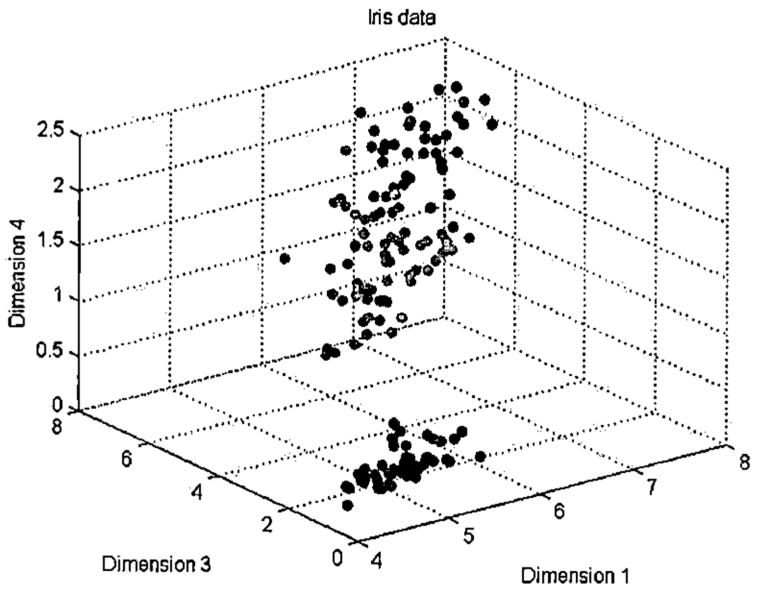


Fig. 2.9: Cluster result for Iris data set in subspace$\{1, 3, 4\}$

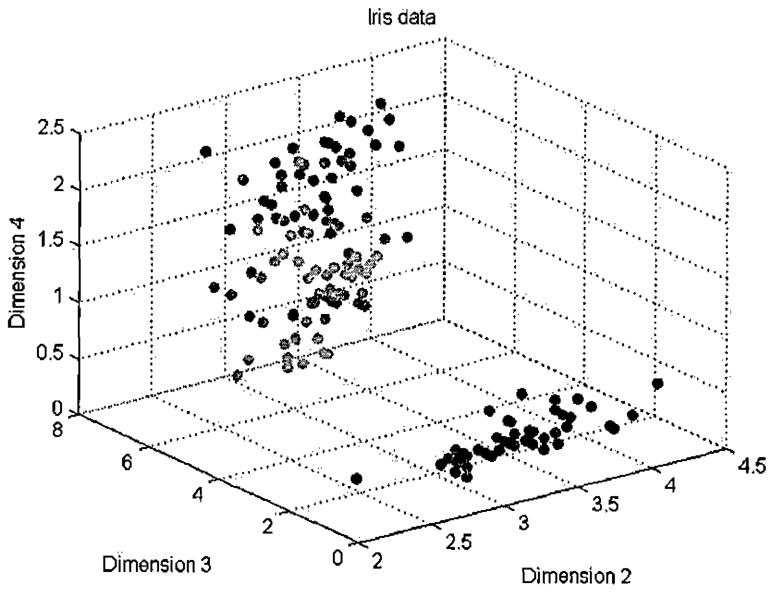


Fig. 2.10: Cluster result for Iris data set in subspace{2, 3, 4}

3. THE HIGH-DIMENSIONAL DATA CLUSTERING ALGORITHM

3.1 Preliminary Definition

In order to avoid "curse of dimension", a new algorithm which is called "Candidate Subspace clustering algorithm" has been designed. Candidate Subspace clustering algorithm is based particularly on the formal definitions by using the idea of Apriori rule. Besides, Candidate Subspace clustering algorithm can be combined with other clustering algorithms such as K-means and DBSCAN. K-means and DBSCAN clustering algorithms which are well defined and widely used. Using this clustering notion, Candidate Subspace clustering algorithm is much more effective than recent approaches.

In order to build "Candidate Subspace clustering algorithm", here are some basic ideas that need to notice.

Observation 1: Some objects in (n-1) subspace may not be dense enough in (n) subspace. Therefore, it may break down into two smaller clusters or not exist as a cluster.

- 1 cluster in (n-1) subspace $\rightarrow$ 2 or more clusters in (n) subspace
- 1 cluster in (n-1) subspace $\rightarrow$ zero cluster in (n) subspace

As the first observation shows, the increasing number of dimensions under (n-1) subspace developing is variable. Figures 3.1 and 3.2 shows when the dimension grows, the number of clusters increases because one big cluster becomes two smaller clusters. Figures 3.3 and 3.4 shows that there is possibly no cluster when the dimension

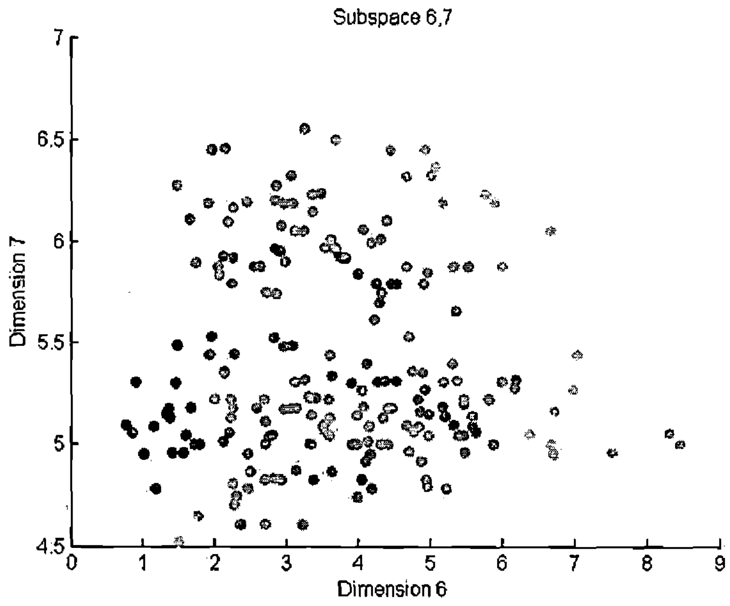


Fig. 3.1: Observation 1-1 : Seeds dataset's cluster result in subspace$\{6, 7\}$

increases.

Observation 2: Some objects in different clusters in (n) subspace might be close enough in (n-1) subspace, therefore can be merged into one cluster in (n-1) subspace.

- 2 clusters in (n) subspace $\rightarrow$ 1 cluster in (n-1) subspace

Compare with observation 1, a decreasing number of clusters occurred consistently during the decreasing of the number of dimensions. Figures 3.5 and 3.6 shows that when objects are close enough in (n-1) subspace, they might be merged into one bigger cluster in (n-1) subspace.

Conclusion: considering the results of both observations, determining how the number of clusters with the rank of dimensions, there is no deterministic relationship between clusters in (n-1) subspace and clusters in (n) subspace. However, a cluster

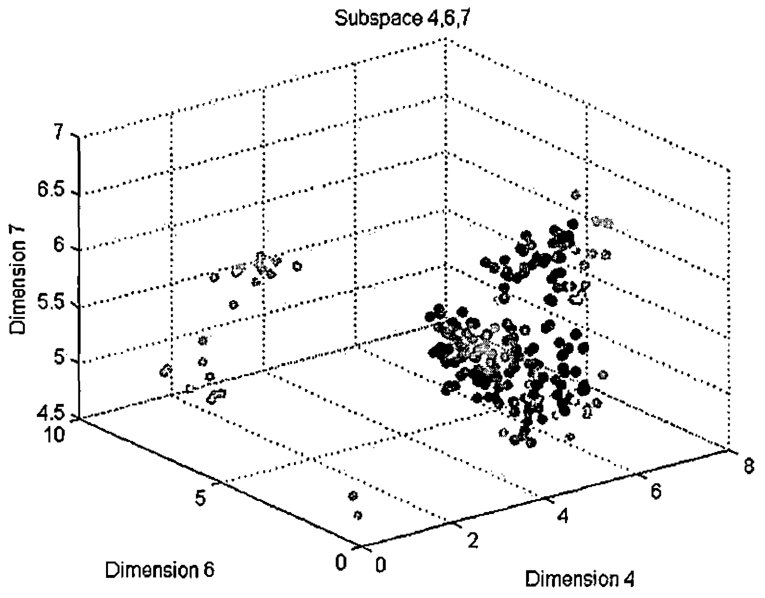


Fig. 3.2: Observation 1-1 : Seeds dataset's cluster result in subspace{4, 6, 7}

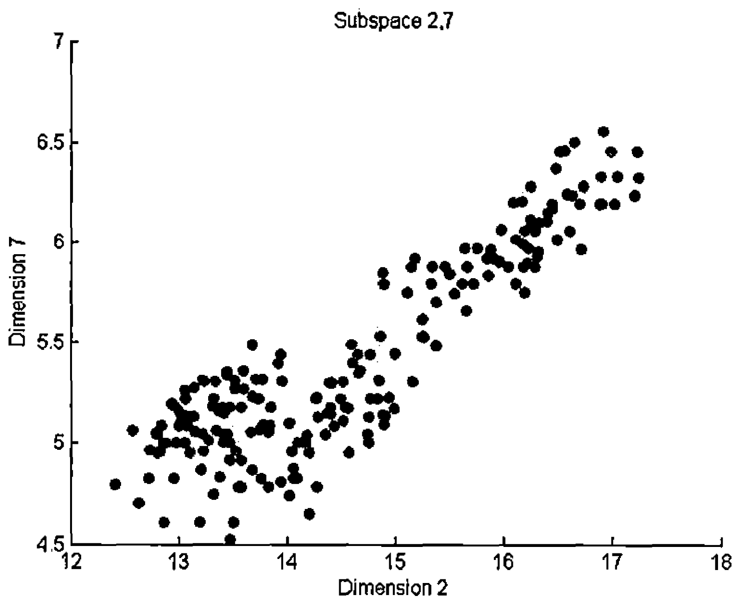


Fig. 3.3: Observation 1-2 : Seeds dataset's cluster result in subspace{1, 2}

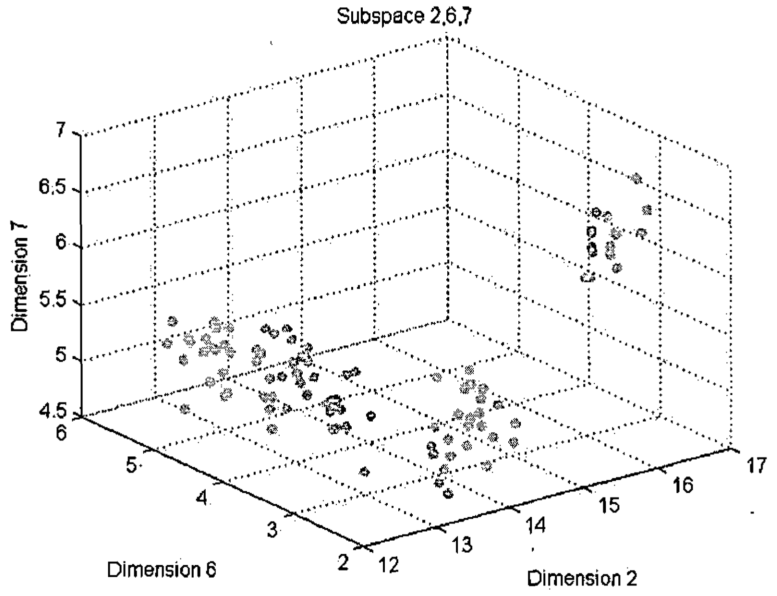


Fig. 3.4: Observation 1-2 : Seeds dataset's cluster result in subspace$\{1, 2, 6\}$

that exists in (n) subspace must exist in (n-1) subspace. If a group of data are not in the same cluster in (n-1) subspace, they are not in the same cluster in (n) subspace. When talking about (n-1) subspace, it is always a subset of (n) subspace.

How to make Candidate Subspace clustering algorithm more efficient than other algorithms? After following previous observations, the first step is to improve the execution time by using the idea of Apriori rule.[18] Apriori rule is a popular algorithm. It uses a breadth-first search strategy to count the support of item sets and uses a candidate generation function which exploits the downward closure property of support.[1] The Apriori rule calculates rules that express probabilistic relationships between items in frequent item sets. For example, a rule derived from frequent item sets containing A, B, and C might state that if A and B are included in a transaction, then C is likely to also be included.[3] By using this idea candidate_space and candidate_object are used in Candidate Subspace clustering algorithm.

When using the idea of Apriori rule, it is important to find a candidate_object

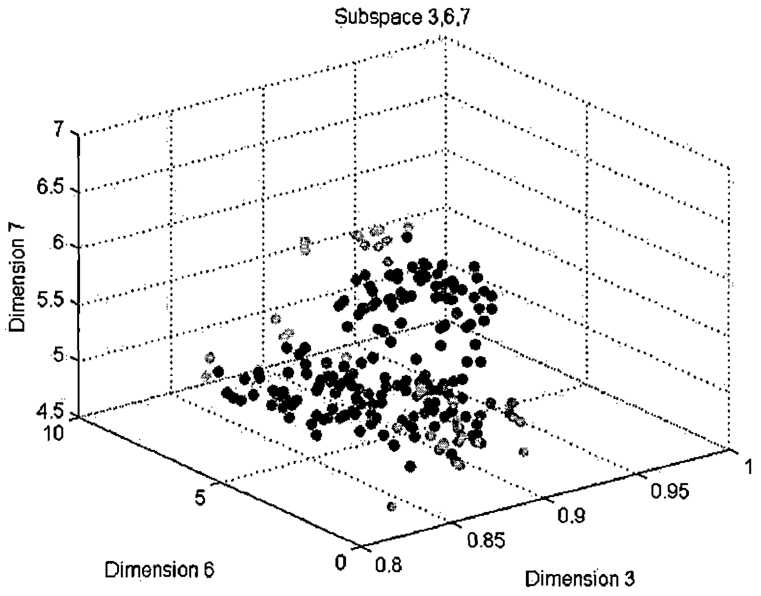


Fig. 3.5: Observation 2-1 : Seeds dataset's cluster result in subspace$\{3, 6, 7\}$

for each corresponding candidate_subspace. Following this process can improve the performance time. Candidate_subspace will be based on the cluster result in (n-1) subspace, to see if there is a cluster or not and then find candidate_space for (n) subspace. Moreover, a candidate_object is based on the corresponding candidate_space in (n-1) subspace, and one find candidate_object in (n) subspace. To understand how candidate_space and candidate_object work, here are two axioms.

Axiom 1: If there is no cluster in (n-1) subspace, then there is no cluster in (n) subspace extended from the (n-1) subspace.

First, when finding candidate_subspace, seeking for a candidate (n) subspace with clusters, all its (n-1) subspaces have to have at least one cluster. Otherwise, the (n) subspace will not be a candidate with clusters. For example, when clustering 4D data, if there is no cluster in 2-D subspace $\{4, 6\}$, then there is no cluster in 3-D subspace $\{4, 5, 6\}$ or $\{4, 6, 7\}$. See Figures 3.7, 3.8, and 3.9. These show that if no cluster exists in subset of the subspace, there will be no cluster in the subspace. Gray points means

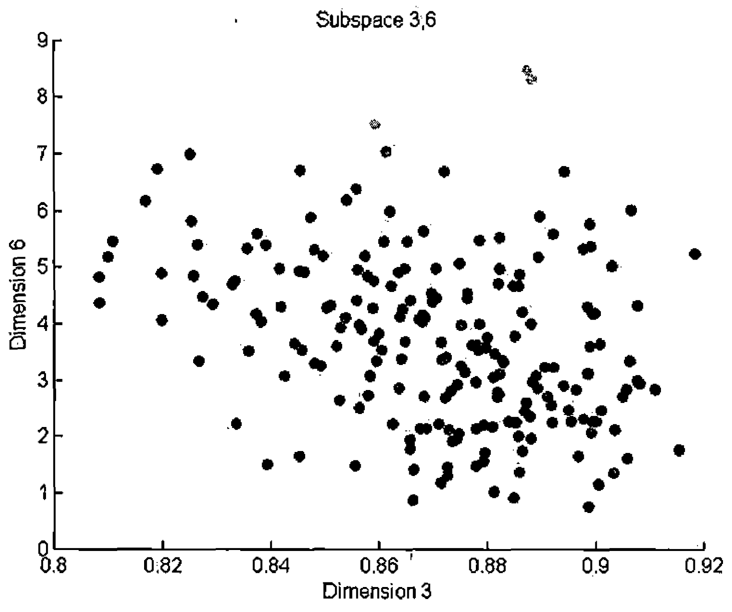


Fig. 3.6: Observation 2-1 : Seeds dataset's cluster result in subspace$\{3, 6\}$

noise, they does not belong to any clusters.

Axiom 2: When extending cluster from (n-1) to (n) subspace, if there exists a cluster C in (n-1) subspace, the corresponding cluster in (n) space based on cluster C will be no bigger in terms of cluster size.

Figures 3.10, 3.11, 3.12 and 3.13 show that a cluster's size in 3-D subspace $\{4, 5, 7\}$ is smaller than the size in 2-D subspace $\{4, 5\}$, $\{4, 7\}$ and $\{5, 7\}$. Different colors represent different clusters. Red points represent cluster 1, green points represent cluster 2, blue points represent cluster 3 and gray points mean noise.

To find candidate_object is similar with finding candidate space. First, to get candidate_space in (n) subspace from cluster results in (n-1) subspace. If there is no cluster in (n) candidate_space, then do not consider it as a (n+1) candidate_object in (n+1) candidate_space. Which means if no cluster exists in (n+1) candidate_space, then candidate_object is empty in (n+1) subspace. However, if there exists a cluster in (n) candidate_space, then find the intersection between each (n) candidate_space,

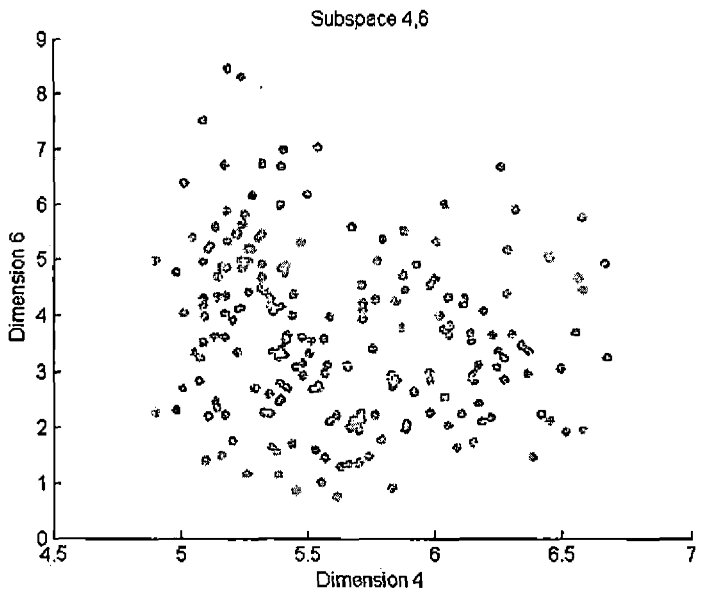


Fig. 3.7: Theorem 1 : Seeds dataset's cluster result in subspace$\{4,6\}$

and consider it as a $(n+1)$ candidate_object in $(n+1)$ candidate_space. For example, in order to find candidate_object in candidate_space $\{1,2,3\}$, finding the intersection between candidate_object $\{1,2\}$, $\{1,3\}$ and $\{2,3\}$. If there is no intersection between each candidate_object's subset, then candidate_space $\{1,2,3\}$ will not have a candidate_object $\{1,2,3\}$. Which means it does not have to calculate the pair of candidate_space $\{1,2,3\}$ and candidate_object $\{1,2,3\}$. There will be no cluster in candidate _space$\{1,2,3\}$. On the other hand, if there is one intersection object between candidate_object $\{1,2\}$, $\{1,3\}$ and $\{2,3\}$, then candidate_space $\{1,2,3\}$ has a corresponding candidate_object $\{1,2,3\}$.

However, when extending clustering from $(n\text{-}1)$ to (n) subspace, if there exists a cluster C in $(n\text{-}1)$ space, the corresponding cluster in (n) subspace based on cluster C will be no bigger in terms of cluster size. For example, give a candidate (n) subspace, starting clusters generated in $(n\text{-}1)$ subspace. Choose the $(n\text{-}1)$ subspace with the minimal number of clusters for efficiency. One cluster in $(n\text{-}1)$ subspace may not be

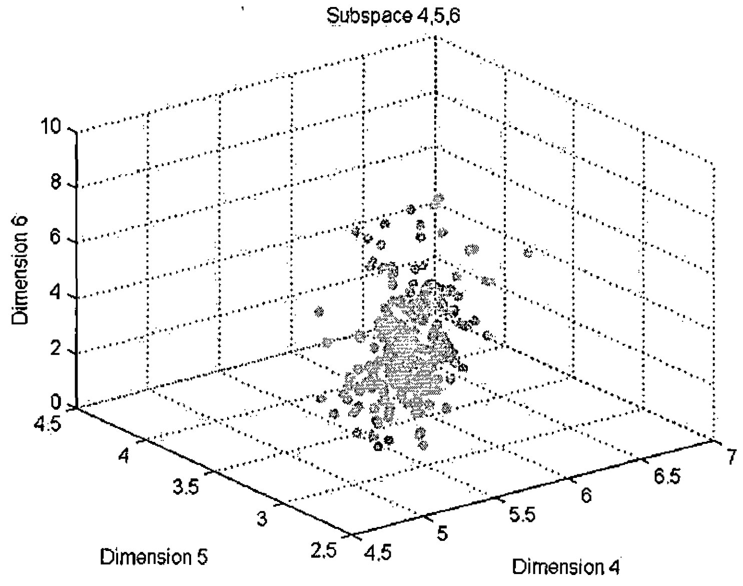


Fig. 3.8: Theorem 1 : Seeds dataset's cluster result in subspace{4, 5, 6}

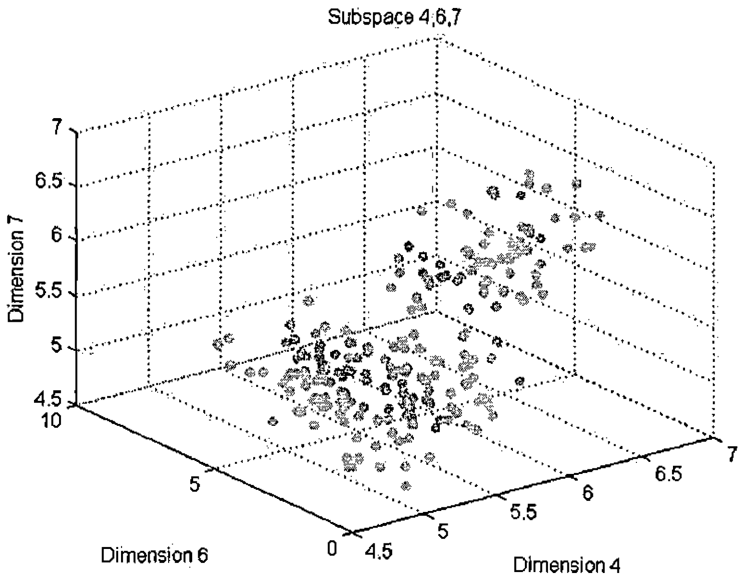


Fig. 3.9: Theorem 1 : Seeds dataset's cluster result in subspace{4, 6, 7}

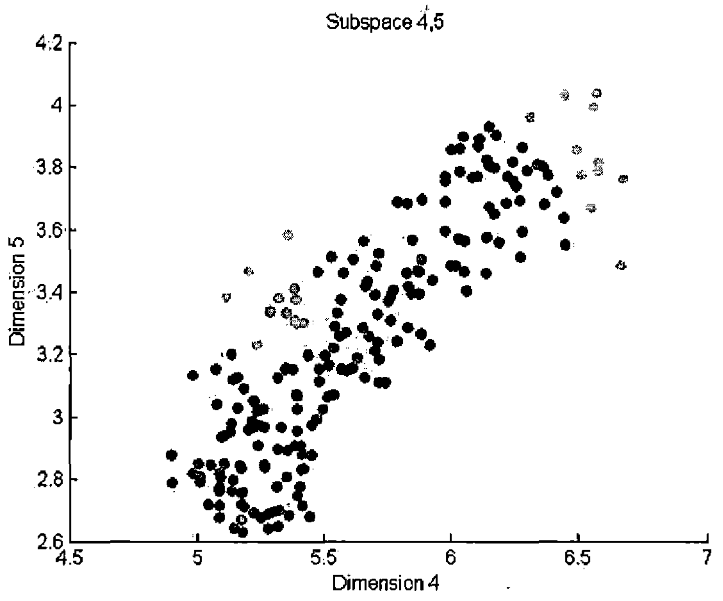


Fig. 3.10: Theorem 2 : Seeds dataset's cluster result in subspace$\{4, 5\}$

a cluster in (n) subspace and may contain one or break down to multiple clusters in (n) subspace.

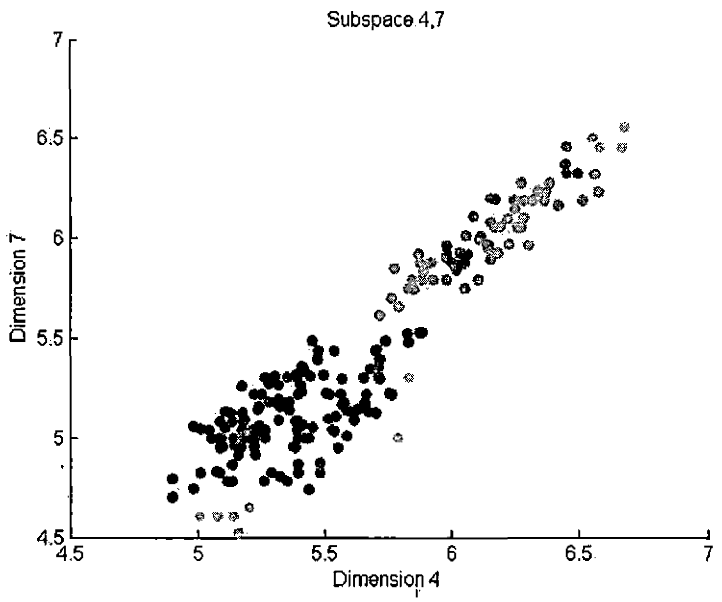


Fig. 3.11: Theorem 2 : Seeds dataset's cluster result in subspace$\{4, 7\}$

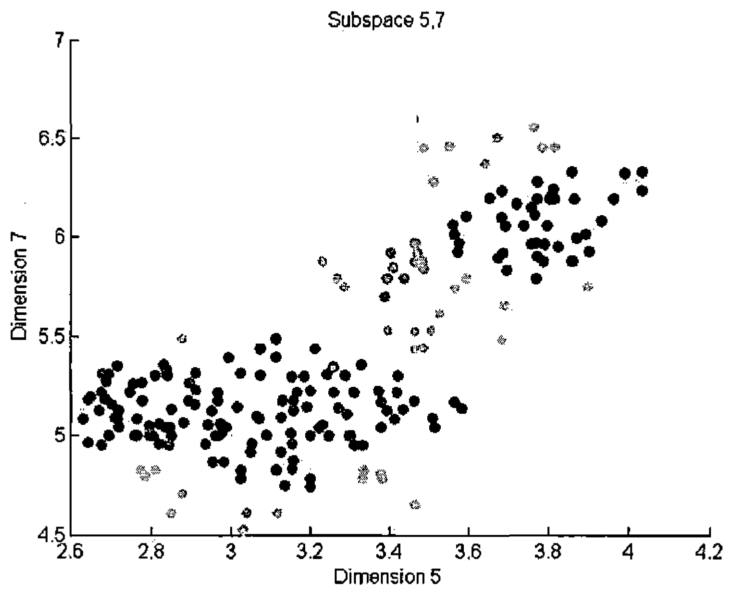


Fig. 3.12: Theorem 2 : Seeds dataset's cluster result in subspace$\{5, 7\}$

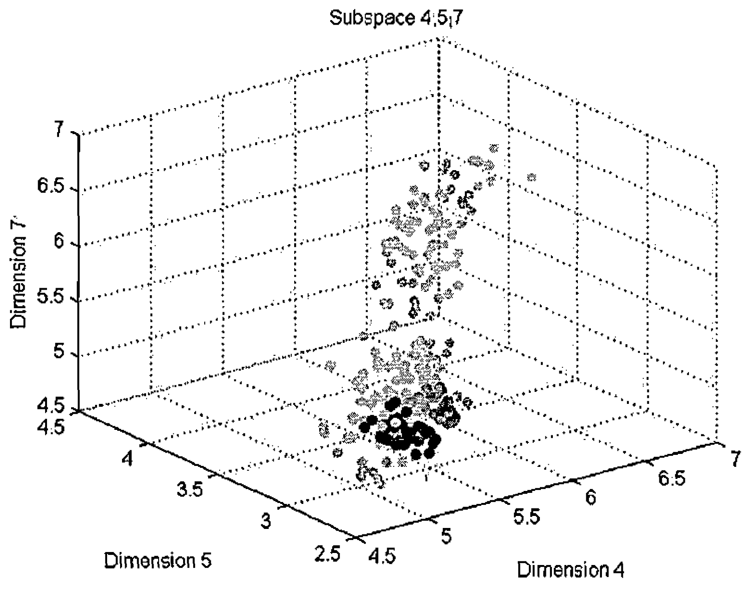


Fig. 3.13: Theorem 2 : Seeds dataset's cluster result in subspace$\{4,5,7\}$

3.2 Candidate Subspace Clustering Algorithm

Candidate Subspace clustering algorithm is a type of agglomerative hierarchical clustering algorithm, find the clusters in $(n+1)$ subspace based on (n) subspace. The following is the Candidate Subspace clustering algorithm.

First, there are some functions in MATLAB and can be used directly. By following the instruction in MATLAB, the function can be easily used. There are three functions that have been list out in MATLAB, which are SUBSET, INTERSECT and KMEANS.

According to MATLAB, the function SUBSET is used to find all possible combinations of k-elements in a set S. By using this function, it is used for finding the subset for candidate_subspace. Table 3.1 shows the function SUBSET.

Perhaps, the function INTERSECT is used to find the same elements among a number of sets. By using this function, it is used for finding the intersection between

Tab. 3.1: MATLAB function: SUBSET

SUBSETS :
p = SUBSETS (s, k)
Purpose: All possible combinations of k-elements in a set s
Input: Set s, k (k $\leq$ size (s))
Output: Set p

each candidate_object. Table 3.2 shows the function INTERSECT.

Tab. 3.2: MATLAB function: INTERSECT

INTERSECT: Set intersection
s = INTERSECT (a, b, ...)
Purpose: find common elements among a number of sets
Input: a number of sets (the number is not fixed)
Output: Set s

Next, running Candidate Subspace clustering algorithm and connect with existing algorithms, such as K-means and DBSCAN algorithms and get the cluster result. MATLAB already has K-means inside the function, as a result, it can be directly used. According to the function definition, "K-means uses a two-phase iterative algorithm to minimize the sum of point-to-centroid distances, summed over all K clusters".[17] There are two inputs for KMEANS function: "x" and "k". "X" represent the input data set, and "k" means how many centroid points to become a cluster. The output of KMEANS function is a column vector "Z". Each element in "Z" is a integer, represents the index of cluster it belongs to. For example, if Z[i] = 0, then object "i" does not belong to any cluster. If Z[i] = 1, then object "i" belong to cluster 1. Table 3.3 shows the function KMEANS.

Besides, Candidate Subspace clustering algorithm can also connect with DBSCAN to run the testing data. The purpose of DBSCAN is to clustering the data with Density-Based Scan algorithm with Noise. There are three inputs for DBSCAN func-

Tab. 3.3: MATLAB function: KMEANS

```
KMEANS: K-means algorithm
Z[i] = KMEANS (x, k)
Purpose: find cluster by using K-means algorithm
Input: x - input data set
       k - centroid point
Output: a column vector Z, each element in Z is a integer, represents the index
        of cluster it belongs to. If Z[i] =0, then object i does not belong to
        any cluster. If Z[i] =1,then object i belong to cluster 1
```

tion: "x", "minPt" and "Eps". "X" represent the data set with one pairs of "m" and "n", where "x.m" is a set of objects, "x.n" is a set of dimensions for a space. "MinPt" means the number of objects in a neighborhood of an object. "Eps" is the neighborhood radius. The output of DBSCAN function is a column vector "Z", the number of elements is "x.m" (number of objects). Each element in "Z" is a integer, represents the index of cluster it belongs to. For example, if Z[i] = 0, then object "i" does not belong to any cluster. If Z[i] = 1, then object "i" belong to cluster 1. Table 3.4 is the pseudo code of DBSCAN algorithm.

Tab. 3.4: The pseudo code of DBSCAN

```
DBSCAN: Clustering the data with Density-Based Scan algorithm
         with Noise (DBSCAN)
Z[i] = DBSCAN (x, minPt, Eps)
Purpose: find cluster by using DBSCAN algorithm
Input: x - data set with one pairs of m and n, where x.m is a set of objects,
         x.n is a set of dimensions for a space
       minPt - the number of objects in a neighborhood of an object
       Eps - neighborhood radius
Output: a column vector Z, each element in Z is a integer, represents the index
        of cluster it belongs to. If Z[i] =0, then object i does not belong to
        any cluster. If Z[i] =1,then object i belong to cluster 1
```

After understand how to use the function, K-means and DBSCAN, the algorithm "Candidate Subspace clustering algorithm" is show at Table 3.5. The purpose of Candidate Subspace clustering algorithm is that when given clustering results in (k) space, find clusters in (k+1) space. It based on cluster result in (k) space, find out the pair of candidate_space and candidate_object as a input data "x" for the next iteration and find the cluster result. When there is no pare of candidate_space and candidate_object or each one of them is empty, the loop will break out. In order to run Candidate Subspace clustering algorithm, there are some initial sets. First, the initial input for Candidate Subspace clustering algorithm are : "m", "n", "DATA" and "X". "M" is number of objects to cluster, "n" is number of dimensions in each object. "DATA" is a m by n matrix, "X" is a set from 1, 2... to m. Next, initial sets are : temp(k).com, a(1).candidate_space{i} and a(1).candidate_object{i}. Where temp(k).com is combinational subspaces in k-D, k=1, 2... to n, a(1).candidate_space{i} = temp(1).com and a(1).candidate_object{i} = X. Table 3.5 shows the pseudo code of Candidate Subspace clustering algorithm.

Tab. 3.5: The pseudo code of Candidate Subspace clustering algorithm

```
for j=1 to n
  a(1).cluster_result{j} = KMEANS(data(X, j), k)
  //a(1).cluster_result{j} = DBSCAN(data(X(m), j), minPt, Eps)
end for loop

for k=1, 2, ... n
  //narrow possible subspaces in (k+1)-D
  for each subspace s in temp(k).com
     If a(k).cluster_result{j} is not empty
          delete the elements in temp(k+1).com that contains s
     end if
  end for loop

  //find candidate space and candidate object in (k+1)-D
  for element i = 1, 2 to size of (temp(k+1).com)
     a(k+1).candidate_space{i} = temp(k+1).com;
     a(k+1).candidate_object{i}= INTERSECT (SUBSETS
                                      (a(k+1).candidate_space{i},k).DATA)
     if a(k+1).candidate_space{i} or a(k+1).candidate_object{i} is empty
          break
     else
          a(k+1).cluster_result{i}  = KMEANS(
          data(a(k+1).candidate_object{i}, a(k+1).candidate_space{i}), k)
          //a(k+1).cluster_result{i}  = DBSCAN(
           data(a(k+1).candidate_object{i}, a(k+1).candidate_space{i})
             , minPt, Eps)
     end if
  end for loop
repeat
```

4. PERFORMANCE EVALUATION

To present the simulation results and compare Candidate Subspace clustering algorithm with two other clustering algorithms: K-means and DBSCAN. Where K-means simply assumes K spherical clusters and looks for their centers, in this case, there are group of three clusters in the testing data. DBSCAN will calculate the Eps distance between each object which contain the MinPts and try to find the best result for each clusters.

For running Candidate Subspace clustering algorithm, connect it with K-means and DBSCAN algorithm, and compare with the original K-means and DBSCAN clustering algorithm. The performance of Candidate Subspace clustering algorithm is illustrated on the Iris data set and Seeds data set. These Iris data set[5] and Seeds data set[8] are taken from UCI machine learning data set repository. When running Candidate Subspace clustering algorithm connect with DBSCAN, it is hard to find the best solution for (MinPts, Eps). After tested on hundreds of sets about (MinPts, Eps), the best solution for Iris data set is (4, 0.4). For Seeds data set, the best solution is (7, 0.625). After running the clustering algorithm, the result will compare Candidate Subspace clustering algorithms with original K-means and DBSCAN by using testing data sets.

4.1 Iris Data Set

The Iris data contains 3 classes (Iris Setosa, Versicolor and Virginica owers), 4 attributes (sepal length in cm, sepal width in cm, petal length in cm and petal width in cm), each class has 50 cases. The maximal number of dimension which Matlab can draw is up to 3D. In order to draw the picture, applying PCA (principal component analysis)[7] on Iris data set to reduce it's dimension to 2D. The first principal component would be called pcl, the second principal component is called pc2, etc. The entire Iris data set has been shown in Figure 4.1 on two dimensions, pcl and pc2, which are much more informative than the other two.

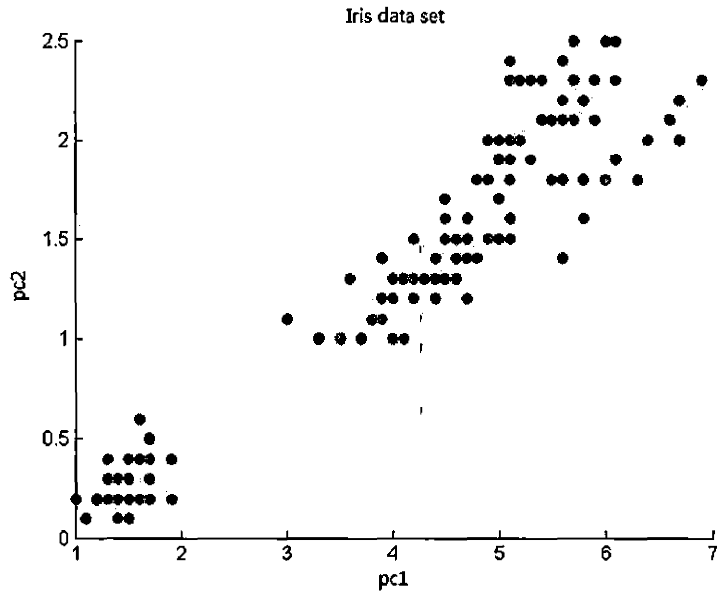


Fig. 4.1: Original 150 Iris data displayed using the PCA features.

First, when running Candidate Subspace clustering algorithm connect with K-means, the clustering result are shows in Figure 4.2. All objects are labeled by numbers: 1-50 for the Setosa , 51-100 for the Versicolor, and 101-150 for the Virginica. The Candidate Subspace clustering algorithm created three clusters corresponding to Setosa, Versicolor, and Virginica varieties. Thus the objects 53, 78 which belong to the Virginica are incorrectly grouped into the cluster associated with the Versicolor. Moreover, the objects 102, 107, 114, 115, 120, 122, 124, 127, 128, 134, 139, 143, 147, 150 which belong to the Versicolor, are wrongly classified into the cluster associated with the Virginica.

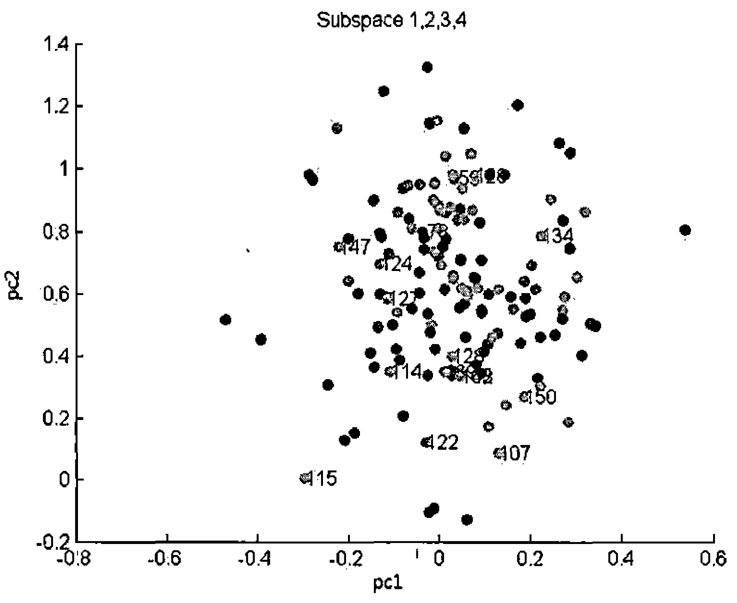


Fig. 4.2: Iris data set's CS cluster result.

The cluster results which containing numbers of grains classified properly and numbers of grains classified wrongly into clusters associated with Setosa, Versicolor, and Virginica varieties are shown in Table 4.1.

Tab. 4.1: Clustering results of the Candidate Subspace clustering algorithm and K-means clustering algorithm for the Iris data set

		CS			K-means	
	correct	incorrect	total	correct	incorrect	total
Setosa	50	0	50	50	0	50
Versicolor	36	2	38	48	14	62
Virginica	48	14	62	36	2	38

According to the results of the Candidate Subspace clustering algorithm, there is no incorrect object for Setosa. These result are equal if K-means clustering algorithm is used. The Setosa variety is best recognized using both techniques.

For the other two varieties, the created clusters containing 38 elements (Versicolor) and 62 elements (Virginica). To the Versicolor, 36 kernels were classified correctly, while 2 of the other varieties were incorrectable identified as the Versicolor variety. For the Virginica, 48 kernels were correctly identified and 14 kernels of the Versicolor variety were wrongly identified as the Virginica variety. The clustering results by using K-means algorithm were the following: the Versicolor variety: out of 50 kernels, 36 were correctly identified and 14 incorrectly identified; the Virginica variety: our of 50 kernels, 48 were correctly identified and 2 incorrectly identified. This implies, that Versicolor and Virginica varieties are similar with respect to geometric parameters.

The percentages of correctness of both methods are summarized in Table 4.2. Candidate Subspace clustering algorithm achieved accuracy of about 100% for the Setosa, 72% for the Versicolor and 96% for the Virginica. Compare with K-means algorithm, it achieved accuracy of about 100%, 96%, and 72% respectively.

Tab. 4.2: Comparison of performance of the Candidate Subspace clustering algorithm and K-means clustering algorithm for the Iris data set, correctness percentages are reported

	CS	K-means
	Correctness	Correctness
Setosa	100%	100%
Versicolor	72%	96%
Virginica	96%	72%

Moreover, when running Candidate Subspace clustering algorithm connect with DBSCAN clustering algorithm, the clustering result are shows in Figure 4.3. All objects are labeled by numbers: 1-50 for the Setosa, 51-100 for the Versicolor, and 101-150 for the Virginica. The Candidate Subspace clustering algorithm also created three clusters corresponding to Setosa, Versicolor, and Virginica varieties. Thus the objects 71,73 and 84 which belong to the Versicolor, are wrongly classified into the cluster associated with the Virginica. The object 65 is the noise.

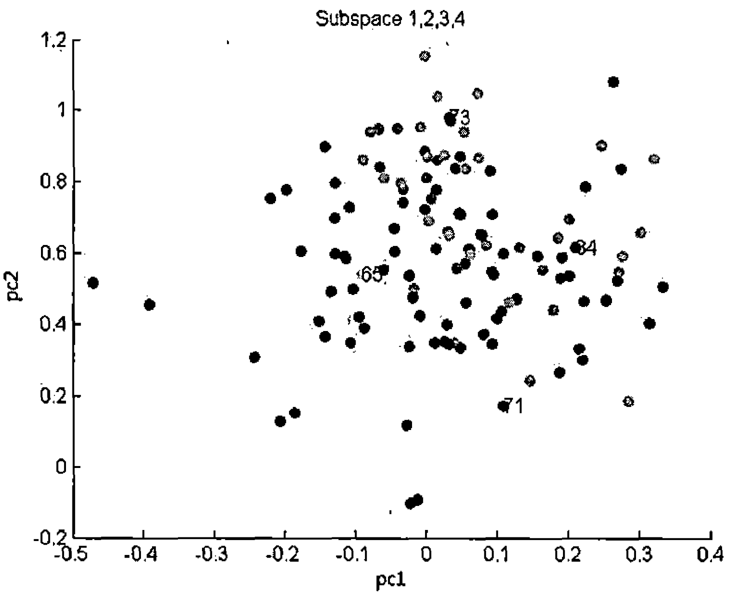


Fig. 4.3: Iris data set's CS cluster result.

Clustering results, containing numbers of grains classified properly and numbers of grains classified wrongly into clusters associated with Setosa, Versicolor, and Virginica varieties, are shown in Table 4.3.

Tab. *4.3:* Clustering results of the Candidate Subspace clustering algorithm and DBSCAN clustering algorithm for the Iris data set

		CS			DBSCAN	
	correct	incorrect	total	correct	incorrect	total
Setosa	47	0	47	47	0	47
Versicolor	39	0	39	39	0	39
Virginica	35	3	38	35	3	38

According to the results of running Candidate Subspace clustering algorithm, there is no incorrect object for Setosa and Versicolor. These results are equal if DBSCAN clustering algorithm is used. The Setosa variety is best recognized using both techniques.

For the other two varieties, the Candidate Subspace clustering algorithm created clusters containing 39 elements (Versicolor) and 38 elements (Virginica). To the Versicolor, 39 kernels were classified correctly. For the Virginica, 35 kernels were correctly identified and 3 kernels of the Versicolor variety were wrongly identified as the Virginica variety. The clustering results by using DBSCAN algorithm were the following: the Versicolor: out of 50 kernels, 47 were correctly identified; the Virginica: out of 50 kernels, 35 were correctly identified and 3 incorrectly identified. This implies, that Versicolor and Virginica varieties are similar with respect to geometric parameters.

The percentages of correctness of both methods are summarized in Table 4.4. Candidate Subspace clustering algorithm achieved accuracy of about 94% for the Setosa, 78% for the Versicolor and 70% for the Virginica. Compare with DBSCAN, it algorithm achieved accuracy of about 94%, 78% and 70% respectively.

Tab. 4.4: Comparison of performance of the Candidate Subspace clustering algorithm and DBSCAN clustering algorithm for the Iris data set, correctness percentages are reported

	CS	DBSCAN
	Correctness	Correctness
Setosa	94%	94%
Versicolor	78%	78%
Virginica	70%	70%

Tab. 4.5: Execution time when running the algorithm with Iris data set.

	CS	Original
K-means	0.472 sec	0.376 sec
DBSCAN	0.946 sec	0.647 sec

Table 4.5 shows the execution time when running algorithms with Iris data set. The execution time of Candidate Subspace clustering algorithm connect with K-means is slower then running K-means and DBSCAN. The reason is that Candidate Subspace clustering algorithm has to go through all the candidate space. Although K-means and DBSCAN faster than Candidate Subspace clustering algorithm, they make strong assumptions that the candidate space exists in and only in the full space.

This assumption can not hold in high-dimensional data clustering problem.

4.2 Seeds Data Set

The Seeds data contains 3 (Rosa, Kama and Canadian)classes, 7 attributes (area A, perimeter P, compactness C = $4*pi*A/P^2$, length of kernel, width of kernel, asymmetry coefficient and length of kernel groove), each class has 70 cases. The data's projection on the axes of the two greatest principal components is presented in Figure 4.4.

After running Candidate Subspace clustering algorithm connect with K-means, the clustering result are show in Figure 4.5. All objects are labeled by numbers: 1-70 for the Kama , 71-140 for the Rosa and 141-210 for the Canadian. The Candidate Subspace clustering algorithm created three clusters corresponding to Rosa, Kama, and Canadian varieties. Thus the objects 17, 20, 24, 27, 28, 40, 60, 61, 62, 63, 64 and 70 which belong to the Rosa are incorrectly grouped into the cluster associated with the Canadian. Moreover, the objects 101, 125, 123, 133,134,135,136, 138,139 and140 which belong to the Kama, are wrongly classified into the cluster associated with the Rosa. In addition, the object 38, which belong to the Rosa are wrongly classified into the cluster associated with the Kama.

Clustering results, containing numbers of grains classified properly and numbers of grains classified wrongly into clusters associated with Rosa, Kama, and Canadian varieties, are shown in Table 4.6.

Tab. 4.6: Clustering results of the Candidate Subspace clustering algorithm and K-means clustering algorithm for the Seeds data set

		CS			K-means	
	correct	incorrect	total	correct	incorrect	total
Rosa	57	10	67	51	10	61
Kama	60	1	61	66	1	67
Canadian	70	12	82	70	12	82

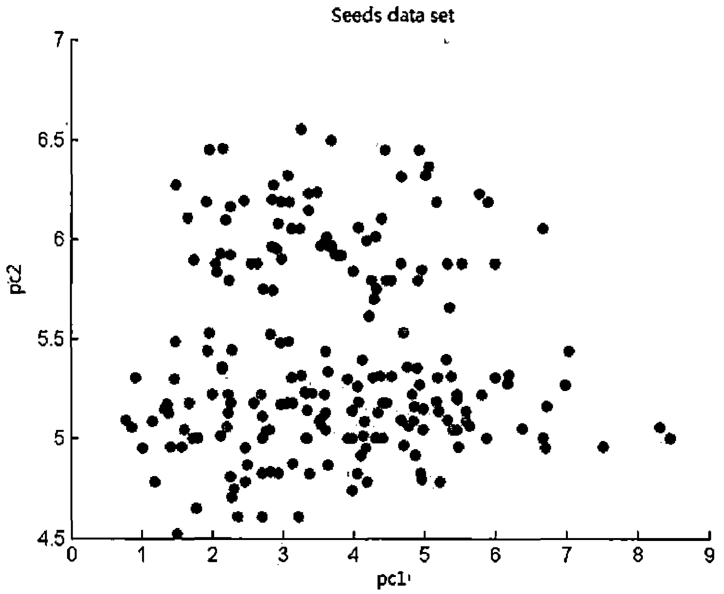


Fig. 4.4: Original 210 Seeds data displayed using the PCA features.

According to the cluster results of Candidate Subspace clustering algorithm, only 1 of the Kama were classified wrongly as the Rosa variety. These result are equal if K-means clustering algorithm is used. The Rosa variety is best recognized using both techniques. For the other two varieties, the Candidate Subspace clustering algorithm created clusters containing 61 elements (Kama) and 82 elements (Canadian). To the Kama, 60 kernels were classified correctly, while 1 of the other varieties were incorrectable identified as the Kama variety. For the Canadian, 70 kernels were correctly identified and 12 kernels of the Kama variety were wrongly identified as the Canadian variety. The clustering results by using K-means algorithm were the following. The Kama variety: out of 70 kernels, 60 were correctly identified and 12 incorrectly identified; the Canadian variety: our of 70 kernels, 68 were correctly identified and 2 incorrectly identified. This implies, that Kama and Canadian varieties are similar with respect to geometric parameters.

The percentages of correctness of both methods are summarized in Table 4.7. Can-

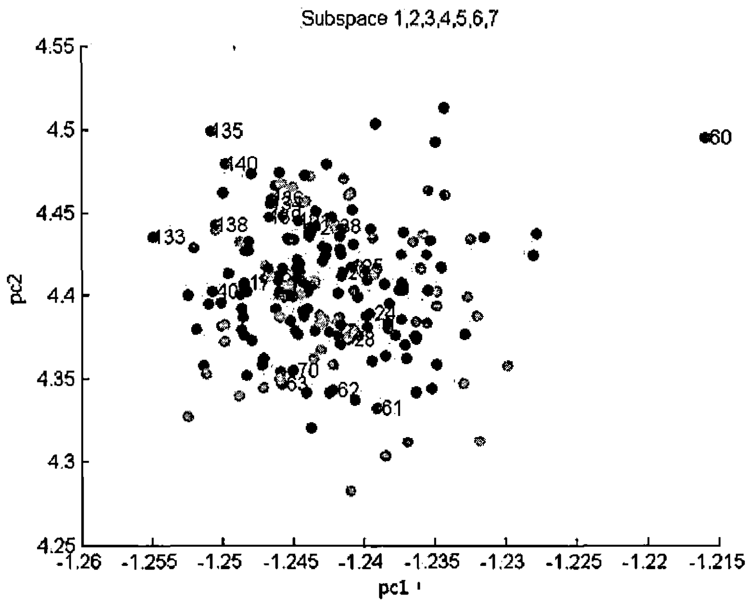


Fig. 4.5: **Seed data set's CS cluster result.**

didate Subspace clustering algorithm achieved accuracy of about 81% for the Rosa, 86% for the Kama and 100% for the Canadian. Compare with K-means algorithm, it achieved accuracy of about 72%, 94% and 100% respectively.

Tab. 4.7: **Comparison of performance of the Candidate Subspace clustering algorithm and K-means clustering algorithm for the Seeds data set, correctness percentages are reported**

	CS	K-means
	Correctness	Correctness
Rosa	81%	72%
Kama	86 %	94%
Canadian	100 %	100%

Next, after running Candidate Subspace clustering algorithm connect with DB-SCAN, the clustering result are show in Figure 4.6. Thus the objects 20, 63 and 64 which belong to the Rosa are incorrectly grouped into the cluster associated with the Canadian. Moreover, the object 136 which belong to the Kama, are incorrect classified into the cluster associated with the Rosa. In addition, the objects 149, 161 and 199 which belong tö the Rosa are wrongly classified into the cluster associated with the Kama. The clustering results are shown in Table 4.8.

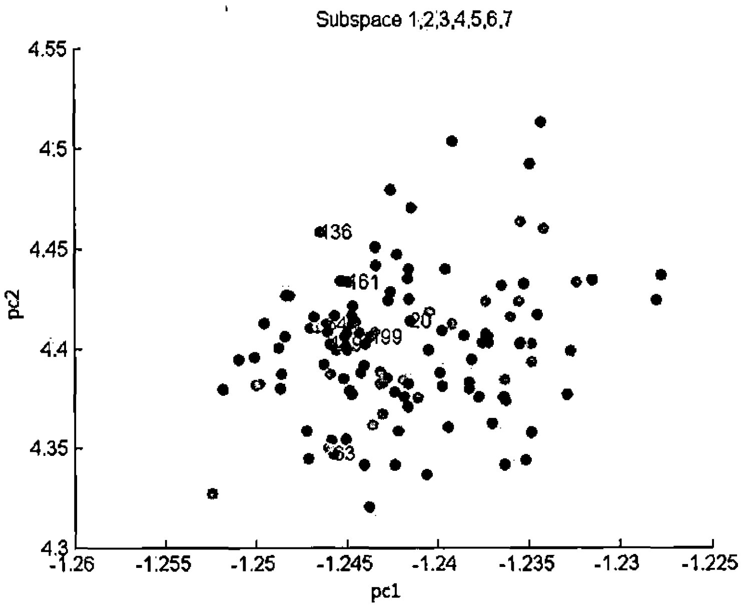


Fig. 4.6: Seed data set's CS cluster result.

Tab. 4.8: Clustering results of the Candidate Subspace clustering algorithm and DBSCAN clustering algorithm for the Seeds data set

		CS			DBSCAN	
	correct	incorrect	total	correct	incorrect	total
Rosa	33	3	36	33	3	36
Kama	32	1	33	32	1	33
Canadian	54	3	57	54	3	57

According to the results of the Candidate Subspace clustering algorithm, only 1 of the Kama were classified wrongly as the Rosa variety. These result are equal if DBSCAN clustering algorithm is used. For the other two varieties, the Candidate Subspace clustering algorithm created clusters containing 33 elements (Kama) and 57 elements (Canadian). To the Kama, 32 kernels were classified correctly, while 1 of the other varieties were incorrect identified as the Kama variety. For the Canadian, 54 kernels were correctly identified and 3 kernels of the Kama variety were wrongly identified as the Canadian variety. The clustering results by using DBSCAN algorithm were the same with using Candidate Subspace clustering algorithm. This implies, that Candidate Subspace clustering algorithm and DBSCAN algorithm are similar with respect to geometric parameters. Besides, the reason why the clustering result 's total number of each class were less than 70 is because when using DBSCAN clustering algorithm, it is easily effect by noise and it will not consider the noise as an candidate object from (n) subspace to (n+1) subspace.

The percentages of correctness of both methods are summarized in Table 4.9. Candidate Subspace clustering algorithm achieved accuracy of about 47% for the Rosa, 45% for the Kama and 77% for the Canadian. Compare with DBSCAN algorithm, it achieved accuracy of about 47%, 45% and 77% respectively. Table 4.10 shows the execution time when running algorithms with Seeds data set.

Tab. 4.9: Comparison of performance of the Candidate Subspace clustering algorithm and DBSCAN clustering algorithm for the Seeds data set, correctness percentages are reported

	CS	DBSCAN
	Correctness	Correctness
Rosa	47 %	47%
Kama	45 %	45%
Canadian	77 %	77%

Tab. 4.10: Execution time when running the algorithm with Seeds data set.

	CS	Original
K-means	5.3 sec	3.4 sec
DBSCAN	6.9 sec	5.3 sec

4.3 NewIris Data

In order to prove Candidate Subspace clustering algorithm works and to show its advantages, by add one noise (dimension) to Iris data set to show that it can find any subspace's cluster if there is a cluster exist. The number of fifth column were create randomly. The NewIris data will contain 5 attributes, the data's projection show in Figure 4.7.

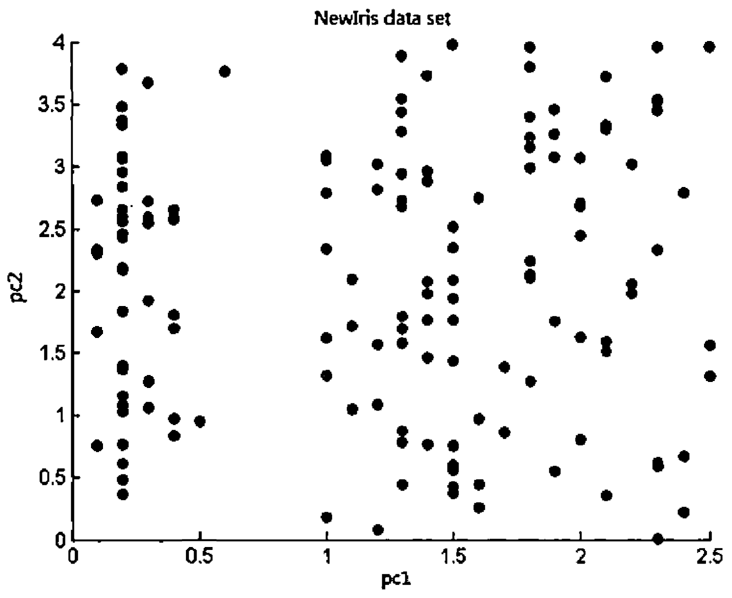


Fig. 4.7: Original 150 NewIris data set displayed using the PCA features.

When running Candidate Subspace clustering algorithm connect with K-means, take a look at subspace $\{1,2,3,4\}$, the clustering result are show in Figure 4.8. It created three clusters corresponding to Setosa, Versicolor, and Virginica varieties. Thus the objects 53 and 78 which belong to the Versicolor were incorrectly grouped into the cluster associated with the Virginica. Moreover, the objects 102, 107, 114, 115, 120, 122, 124, 127, 128, 134, 139, 143, 147 and 150 which belong to the Versicolor, are wrongly classified into the cluster associated with the Virginica.

Clustering results, containing numbers of grains classified properly and numbers of grains classified wrongly into clusters associated with Setosa, Versicolor, and Virginica varieties is show in Table 4.11.

According to the results on subspace $\{1,2,3,4\}$, it performs as same as original Iris

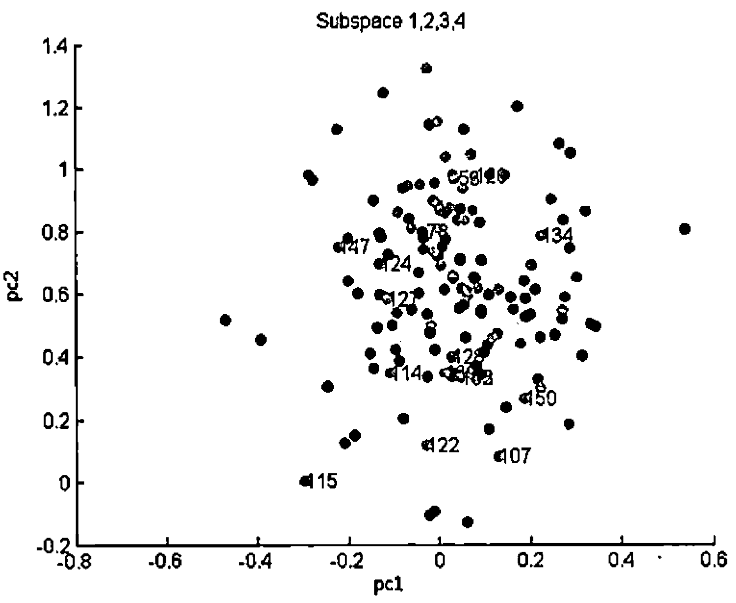


Fig. 4.8: NewIris data set's CS cluster result on subspace$\{1, 2, 3, 4\}$.

Tab. 4.11: Clustering results of the Candidate Subspace clustering algorithm and K-means clustering algorithm for the NewIris data set

		CS			K-means	
	correct	incorrect	total	correct	incorrect	total
Setosa	50	0	50	50	0	50
Versicolor	36	2	38	48	14	62
Virginica	48	14	62	36	2	38

data set which has shown above. The percentages of correctness of both methods are summarized in Table 4.12. The accuracy of about 100% for the Setosa, 72% for the Versicolor and 96% for the Virginica. Compare with K-means algorithm, it achieved accuracy of about 100%, 96% and 72% respectively.

Moreover, when running Candidate Subspace clustering algorithm connect with DBSCAN, the clustering result on subspace $\{1, 2, 3, 4\}$ are show in Figure 4.9. It also created three clusters corresponding to Setosa, Versicolor and Virginica varieties. Thus the objects 71,73 and 84 which belong to the Versicolor, are wrongly classified

Tab. 4.12: **Comparison of performance of the Candidate Subspace clustering algorithm and K-means clustering algorithm for the NewIris data set, correctness percentages are reported**

	· CS	K-means
	Correctness	Correctness
Setosa	100%	100%
Versicolor	72	96%
Virginica	96%	72%

into the cluster associated with the Virginica. The object 65 is the noise.

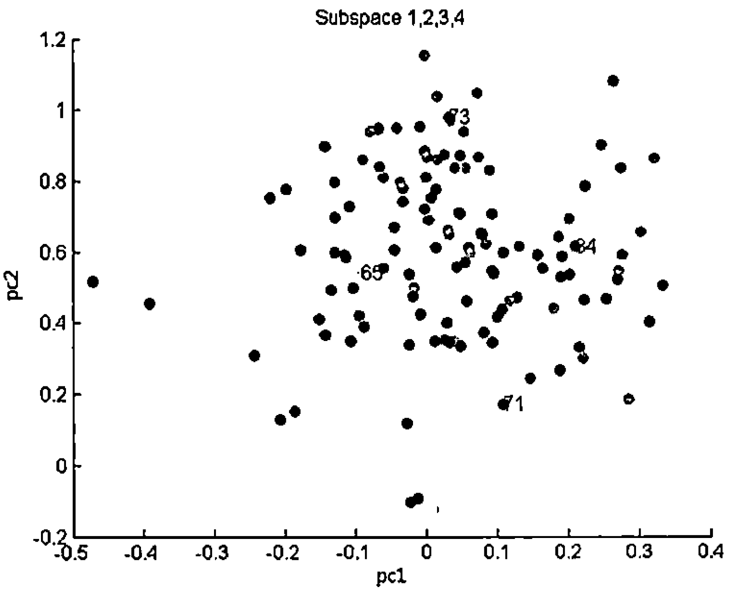


Fig. 4.9: **NewIris data set's CS cluster result.**

Clustering results, containing numbers of grains classified properly and numbers of grains classified wrongly into clusters associated with Setosa, Versicolor, and Virginica varieties is show in Table 4.13.

According to the cluster result on subspace $\{1, 2, 3, 4\}$, there is no incorrect object for Setosa and Versicolor. These results are equal if DBSCAN clustering algorithm is used. The Setosa variety is best recognized using both techniques. The percentages of correctness of both methods are summarized in Table 4.14. Candidate Subspace

Tab. 4.13: Clustering results of the Candidate Subspace clustering algorithm and DBSCAN clustering algorithm for the Iris data set

		CS			DBSCAN	
	correct	incorrect	total	correct	incorrect	total
Setosa	47	0	47	47	0	47
Versicolor	39	0	39	39	0	39
Virginica	35	3	38	35	3	38

clustering algorithm achieved accuracy of about 94% for the Setosa, 78% for the Versicolor and 70% for the Virginica. Compare with DBSCAN algorithm, it achieved accuracy of about 94%, 78% and 70% respectively.

Tab. 4.14: Comparison of performance of the Candidate Subspace clustering algorithm and DBSCAN clustering algorithm for the Iris data set, correctness percentages are reported

	CS	DBSCAN
	Correctness	Correctness
Setosa	94%	94%
Versicolor	78%	78%
Virginica	70%	70%

When seeking subspace $\{1, 2, 3, 4\}$ in NewIris data set, the cluster result are the same with Iris data. This means that Candidate Subspace clustering algorithm can find meaningful cluster in subspace correctly. It can perform feature abstraction task which is conducted by domain experts, along with clustering.

5. CONCLUSIONS

There are several clustering algorithms and applications that have discussed in previous chapters. In different problems, it need to choose which clustering algorithm should use to solve the specific problems. For example, when dealing with documentation retrieval problem, a set of relevant documents has to be found in millions of documents of dimensionality more than 1000. To handle these problems, it would be easier if some of the data obtained were used in decision making rather than directly using the entire data sets. Therefore, clustering algorithms would be a good idea to use for achieving data abstraction.

However, when data grows more and more into hundreds of thousands of data, there will be some challenges of doing clustering algorithm in high dimensional data. The main challenge for doing data clustering is when it is extending to high dimensional data, which called "curse of dimension". As discussed in previous chapters, most clustering techniques are based on distance or similarity and calculate how objects close to each other become a cluster. However, when dealing with high dimensional data, similarity measurement will become harder and harder or even it will become meaningless. Besides, when doing projection on high dimensional data, the clusters will occur in the corresponding projected subspace, but will not maintain the same cluster in the global space or even other subspaces. Projection can remove the noise, but some of the data or important information might be missing. This is called local feature relevance problem, which means different clusters only exist in different subsets of attributes. As a result, doing data clustering on high dimensional data is different from low dimensional data, and these differences will affect the clustering results. Moreover, several recent approaches on doing data clustering with high dimensional data has been described. All of these approaches have been widely used in a number of areas, and successfully applied, although there still has some improvements needs to be done with techniques, and understand their strengths and limitations. Besides, not all clustering types will be suitable for all types of data.

In this paper, a new data clustering algorithm is designed to finding clusters in high dimensional data in each subspace. Candidate Subspace clustering algorithm it starts from low dimension space, and find clusters in high dimension space. Based on Apriori rule, Candidate Subspace clustering algorithm is an efficient algorithm to compute all data sets in subspaces of high dimensional data. In Candidate Subspace clustering algorithm, it builds the cluster from lower subspace ((n-1) subspace) to higher subspace ((n) subspace). The advantage of Candidate Subspace clustering algorithm are it can detect any arbitrarily shaped and positioned clusters in the specific subspace; instead of calculating all the objects from original data, Candidate Subspace clustering algorithm reduced the number of subspaces and objects for each subspace and works in high-dimensional data; with the cluster result, it is easy to observe a specific subspace which has cluster and which cluster survives in which subspace.

The results of Candidate Subspace clustering algorithm performs equally or better than the classical K-means algorithm. However, when connected with DBSCAN to run the algorithm, the result of total objects in clusters are easily affected by the noise because DBSCAN not count noise into cluster, and K-means will contain all object into clusters. Candidate Subspace clustering algorithm performed well and the percentages of correctness were comparable. The amount of 134 grains when Candidate Subspace clustering algorithm connected with K-means algorithm. When testing with Iris data set, giving almost 90% of the total was classified properly. The

Setosa is better recognized than Versicolor and Virginica because of the correctness percentage is the best. For Seeds data set, 187 objects were found and gave almost 89% of the total, Kama is the best recognizable class. When running Candidate Subspace clustering algorithm connects with DBSCAN, only 121 was found on Iris data set and 119 was found on Seeds data set. It is because DBSCAN is easily affected by noise, especially when dimension grows, and the noise will not count as a candidate object for subspace (n+1). As a result, the total object was found is less than K-means algorithm, because K-means algorithm counts every objects to be cluster, there is no problem with noise affection.

Furthermore, Candidate Subspace clustering algorithm is a type of agglomerative hierarchical clustering algorithm. It finds cluster in (n) subspace based on (n-1) subspace and see if there is a cluster or which cluster exists in the corresponding subspace. When added one noise column as the fifth dimension on Iris data set, it become a 5-D data set. After running Candidate Subspace clustering algorithm connect with K-means algorithm and DBSCAN algorithm, it can easily find meaningful cluster result for 4-D subspace. Moreover, it contains five 4-D subspaces, there are $\{1,2,3,4\}$, $\{1,2,3,5\}$, $\{1,2,4,5\}$, $\{1,3,4,5\}$ and $\{2,3,4,5\}$. When looking on subspace $\{1,2,3,4\}$, the cluster result perform exactly as same as original 4-D Iris data set, which means, Candidate Subspace clustering algorithm works well, it can find cluster in each subspace correctly.

Last but not least, Candidate Subspace clustering algorithm can find meaningful clusters in subspaces. Moreover, unlike current clustering algorithms which assumes all the dimensions are valid features for clustering. The new algorithm can perform feature abstraction task which is conducted by domain experts, along with clustering.

An approach for future work is the development of an more efficient index structure for candidate_space and candidate_object. When data dimensions become higher and higher, a better index support could further improve the efficient of Candidate

Subspace clustering algorithm. Moreover, a parallel version of Candidate Subspace clustering algorithm is envisioned for further improving it's scalability.

REFERENCES

[1] Association rule learning. http://en.wikipedia.org/wiki/Association_rule_learning.

[2] Dbscan algrithm. http://www.360doc.com/content/11/0608/14 /7000788_122449612.shtml.

[3] Dbscan algrithm. http://docs.oracle.com/cd/B28359_01/datamine.111 /b28129/algo_apriori.htm.

[4] Human genetic clustering. http://en.wikipedia.org/wiki/Human_genetic_clustering.

[5] Iris data set. http://archive.ics.uci.edu/ml/datasets/Iris.

[6] K-means clustering. www.en.wikipedia.org/wiki/K-means_clustering.

[7] Principal component analysis. https://en.wikipedia.org/wiki /Principal_component_analysis.

[8] Seeds data set. http://archive.ics.uci.edu/ml/datasets/seeds.

[9] Wikipedia, the free encyclopedia. http://en.wikipedia.org.

[10] M.N. Murty A.K. Jain and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys, Vol. 31, No. 3*, 1999.

[11] Daniel Barbara and Ping Chen. Using the fractal dimension to cluster datasets. *George Mason University, ISE Dept., MSN 4A4, Fairfax, VA 22030 USA*, 2003.

[12] Joel L. Wolf Philip S. Yu Charu C. Aggarwal, Cecilia Procopiuc and Jong Soo Park. Fast algorithms for projected clustering. *Philadelphia PA, ACM SIGMOD International Conference on Management of Data*, 1998.

[13] Hans-Peter Kriegel Peer Kroger Ina Muller-Gorman Elke Achtert, Christian Bohm and Arthur Zimek. Detection and visualization of subspace cluster hierarchies. *Institute for Informatics, Ludwig-Maximilians-Universitat Munchen, Germany*, 2002.

[14] Peer Kroger Hans-Peter Kriegel and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data. 3, 1, Article 1*, March 2009.

[15] Hans-Peter Kriegel Karin Kailing and Peer Kroger. Density-connected subspace clustering for high-dimensional data. *Institute for Computer Science University of Munich, Germany, Lake Buena Vista, FL*, 2004.

[16] Levent Ertoz Michael Steinbach and Vipin Kumar. The challenges of clustering high dimensional data. *Department of Statistics, University of Washington, Seattle, Washington*, 1998.

[17] H. Spath. Cluster dissection and analysis: Theory, fortran programs, examples, translated by j. goldschmidt, halsted press. 1985.

[18] Anita Wasilewska. Apriori algorithm. Department of computer science, 2007. www.cs.sunysb.edu/ cse634/lecture.../07apriori.pdf.

[19] GUO Ji-dong YUE Shi-hong, LI Ping and ZHOU Shui-geng. Using greedy algorithm: Dbscan revisited. *Journal of Zhejiang University SCIENCE*, July 2003.