Study of Issues in Clustering of Data

Project report by

Mandeep Nayyar
MA13M1004

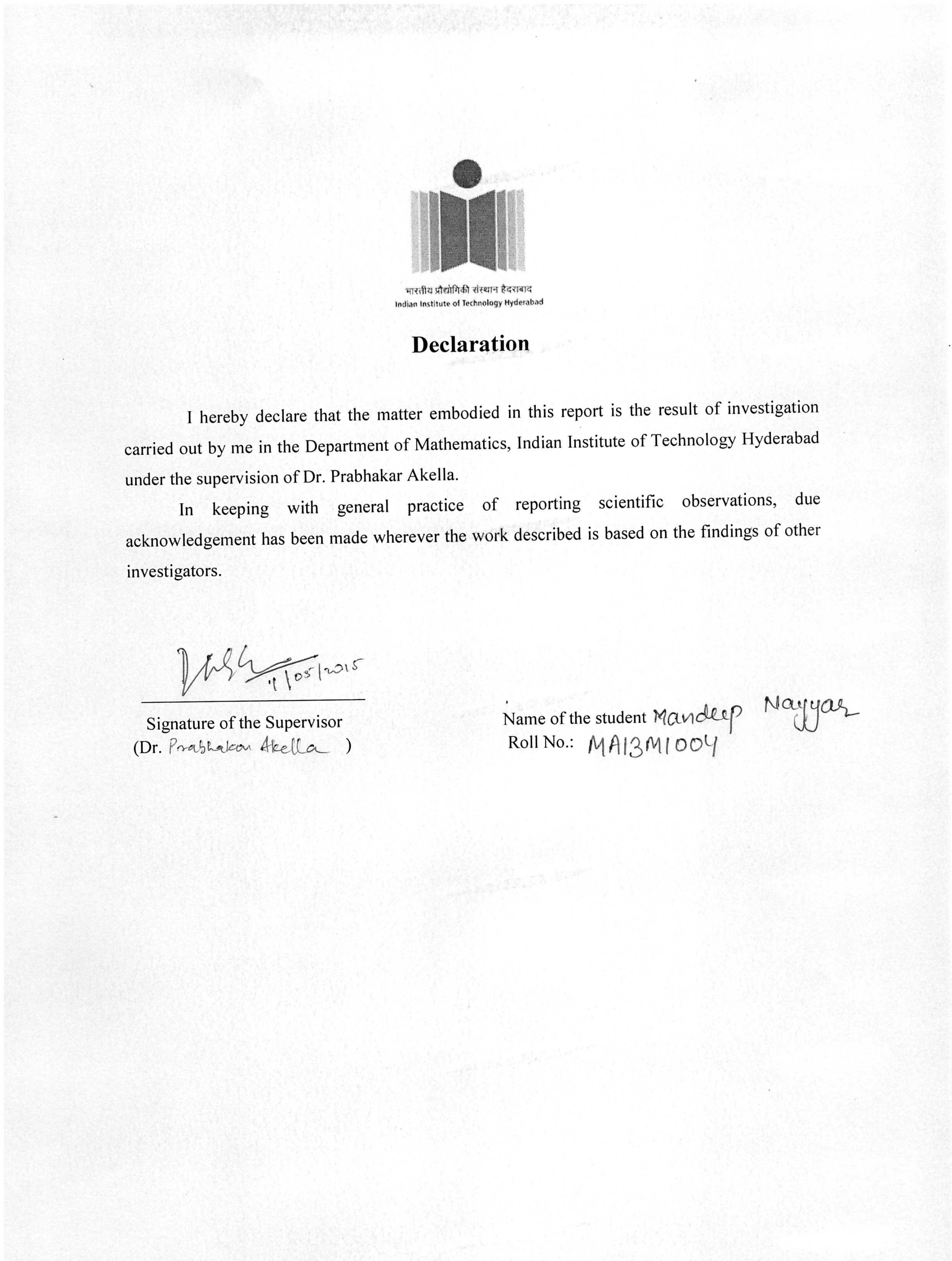
Department of Mathematics.
Indian Institute of Technology Hyderabad
November 2014.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my guide Dr. Prabhakar Akella for the continuous support he gave me during my project work, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of doing project and writing of this thesis.

I would also like to thank Prof. Vineeth N Balasubramanian and Prof. Sumohana Channappayya for clarifying my doubts and helping me with the ideas of implementing algorithms.

Last but not the least, I would like to thank my parents and my friends for their continuous support to me.



भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Declaration

I hereby declare that the matter embodied in this report is the result of investigation carried out by me in the Department of Mathematics, Indian Institute of Technology Hyderabad under the supervision of Dr. Prabhakar Akella.

In keeping with general practice of reporting scientific observations, due acknowledgement has been made wherever the work described is based on the findings of other investigators.

[illegible] 11|05|2015

Signature of the Supervisor
(Dr. Prabhakar Akella)

Name of the student Mandeep Nayyar
Roll No.: MA13M1004

Abstract

Data Mining is sorting through data to identify patterns and establish relationships between data points. It's purpose is to find valuable information hidden in the data. Due to wide availability of huge amount of data, and the imminent need for turning such data into useful information and knowledge for broad applications including market analysis and business management has made research and development on data mining to be flourishing. There are several parameters of data mining such as Classification, Association and Clustering. In this thesis we discuss mainly about Clustering.

In this thesis, we first study what clustering is, what are different types of clusterings and what are clustering techniques. Then, we study what are the issues in clustering. Since, there are many ways to do clustering we want to know which clustering displays the correct information of the data. As there are many definitions of clustering, among one of them is that we consider clustering to be the range of a function defined on the data set. This function is termed as clustering function. There are three natural properties that a clustering function should satisfy. These properties are scale invariance, richness and consistent. Scale invariant means if we increase the distance between every point in data, the clustering function should still be able to cluster data. Richness says that every possible partition of data should be a possible output. Consistent property says that if the distance between the points within cluster is decreased and distance between points in different cluster is increased, the clustering function gives same result. But, it has been proved that there is no clustering function that satisfies all three properties at the same time. So, instead of studying clustering function we move towards another notion $'$quality measure$'$. Quality measure is also a function that maps clustering of the data set to some positive real value. This value tells us how good our clustering is. Quality measure also has some basic natural properties and we have discussed these properties in thesis.

Then we study about $'$clusterability of data sets$'$. Clusterability of data set means that data has a good clusterable structure. There are four types of clusterability that we study- Variance ratio Clusterability, Separability clusterability, Worst pair ratio clusterability and Center perturbation clusterability. We study each of these notions and see what value of these notions will give us better clustering.

And then we study about how to cluster high dimensional data. It is natural to come across data set which has high dimension. In this thesis we will mainly study about $k-$ means clustering technique. Since we are dealing with high dimensional data, and the computational complexity of the clustering algo-

rithm increases as dimension increases, we need some method of dimensionality reduction. The method of dimensionality reduction we will study is *Principal Component Analysis*(PCA).

Next we will study $k-$ means algorithm, high dimensional clustering algorithm and subset high dimensional clustering algorithm used to cluster high dimensional data.

We will use normalized mutual information(NMI) and variance ratio clusterability to do comparative study between these algorithms. Mutual Information is the measurement of how much information the presence/absence of a term contributes to making the correct classification decision.

In our study we have taken four distance measures that are L_1 norm, L_2 norm, L_∞ norm and Inner Product and calculated the *NMI* and variance ratio clusterability across all the algorithms and we concluded that its better to divide the set into subsets and apply $k-$ means on each disjoint subsets rather then dividing the set into disjoint subsets and updating the cluster centers using a single pass algorithm.

Contents

Chapter 1

Introduction

We are in an age often referred to as the information age. In this information age, because we believe that information leads to power and success, we have been collecting tremendous amounts of information. Initially, with the advent of computers and means for mass digital storage, we started collecting and storing all sorts of data, counting on the power of computers to help sort through this amalgam of information. Unfortunately, these massive collections of data stored on disparate structures very rapidly became overwhelming. This initial chaos has led to the creation of structured databases and database management systems (DBMS). The efficient database management systems have been very important assets for management of a large corpus of data and especially for effective and efficient retrieval of particular information from a large collection whenever needed.

With the enormous amount of data stored in files, databases, and other repositories, it is increasingly important to develop powerful means for analysis and perhaps interpretation of such data and for the extraction of interesting knowledge that could help in decision-making. Data Mining, also popularly known as *Knowledge Discovery in Databases* (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. Data mining is the process of analyzing data from different perspectives and summarizing it into useful information. Data mining techniques are deployed to scour large databases in order to find novel and useful patterns. They also provide capabilities to predict the outcome of future observation.

The first and simplest analytical step in data mining is to describe the data summarize its statistical attributes (such as means and standard deviations), visually review it using charts and graphs, and look for potentially meaningful links among variables (such as values that often occur together).

The following describes a typical data mining example: A bank has data about clients to whom it gave loans in the past. The client data contains personal data, data describing the financial status and the financial behavior before and at the time the client was given the loan. The clients are divided into four classes. The first class contains all those clients who payed back the loans without any problems; the second class those who payed back with little problems here and there; the third contains those who should only get a loan after detailed checks because substantial problems of payback occurred in the past; and the forth class consists of all those who did not pay back at all. Using this data table a prediction model is created in order to predict the probability for each class for new clients. A good reference for concepts of data mining can be found in [8].

1.1 Issues in data mining

1.1.1 Scalability

Because of advances in data generation and collection, data sets with sizes of terabytes or even petabytes are becoming common. If data mining algorithms are to handle these massive data sets, then they must be scalable.

1.1.2 High Dimensionality

It is common to encounter data sets with hundreds or thousands of attributes. For example consider a data set that contains measurements of temperature at various locations. If the temperature measurement are taken repeatedly for an extended period, the number of dimension increases in proportion to the number of measurements taken. Data analysis techniques that were developed for low dimensional data often do not work well for such high dimensional data. Also, for some data analysis algorithms, the computational complexity increases rapidly as the dimensionality increases.

1.1.3 Heterogeneous and Complex Data

Data mining techniques often needs to handle data with heterogeneous attributes. Examples of such non-traditional types of data include collections of Web pages containing semi-structured text and hyperlinks; DNA data with sequential and three dimensional structure.

Chapter 2

Cluster Analysis

Suppose one has been assigned a task of arranging the books in a library. One will classify all the books of a particular subject and place them in a separate section. Each section containing books related to a particular subject is what we call as a cluster and the process of doing it is called clustering. We deal with clustering in almost every aspect of daily life. For example, a group of diners sharing the same table in a restaurant may be regarded as a cluster of people. In food stores items of similar nature, such as different types of fruits or vegetables are displayed in the same or nearby locations. In the following sections we will see what clustering is all about, what are techniques used for clustering and what are issues related with it.

2.1 What is Cluster Analysis?

The basic definition of Cluster analysis is that it groups objects of similar kind. The goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering.

Uses of Clustering

In Biology, cluster analysis is used to identify diseases and their stages. For example, by examining patients, who are diagnosed as depressed, one finds that there are several distinct sub-groups of patients with different types of depression. Biologists have applied clustering to analyze the large amounts of genetic information that are now available. For example, clustering has been used to find groups of genes that have similar functions. It is also used for

Information Retrieval, for example, the World Wide Web consists of billions of Web pages, and the results of a query to a search engine can return thousands of pages. Clustering can be used to group these search results into a small number of clusters, each of which captures a particular aspect of the query. The other use of clustering is to understand earth's climate. Understanding the Earth's climate requires finding patterns in the atmosphere and ocean. To that end, cluster analysis has been applied to find patterns in the atmospheric pressure of polar regions and areas of the ocean that have a significant impact on land climate.

2.2 Different types of Clustering

Hierarchical Clustering: Hierarchical clustering is an agglomerative (top down) clustering method. As its name suggests, the idea of this method is to build a hierarchy of clusters, showing relations between the individual members and merging clusters of data based on similarity.

Partitional Clustering: Partitional clustering decomposes a data set into a set of disjoint clusters. Given a data set of N points, a partitioning method constructs K ($N \geq K$) partitions of the data, with each partition representing a cluster. That is, it classifies the data into K groups by satisfying the following requirements: (1) each group contains at least one point, and (2) each point belongs to exactly one group.

Exclusive Clustering: In exclusive clustering data is grouped in an exclusive way, so that each data point belongs to only one definite cluster.

Overlapping Clustering: Overlapping clustering allows data objects to be grouped in 2 or more clusters. A real world example would be the breakdown of personnel at a school. Overlapping clustering would allow a student to also be grouped as an employee.

Fuzzy Clustering: In a fuzzy clustering, every object belongs to every cluster with a membership weight that is between 0 (absolutely doesn't belong) and 1 (absolutely belongs). In other words, clusters are treated as fuzzy sets.

Complete Clustering: A complete clustering assigns every object to a cluster.

Partial Clustering: Partial clustering allows some data objects to be left alone. The motivation for a partial clustering is that some objects in a data set may not belong to well-defined groups. Many times objects in the data set may represent noise or outliers. For example, some newspaper stories may share a common theme, such as global warming, while other stories are more generic or one-of-a-kind. Thus, to find the important topics in last month's stories, we may want to search only for clusters of documents that are tightly related by a common theme.

2.3 Clustering Techniques

2.3.1 Hierarchical Clustering

There are two basic approaches for generating a hierarchical clustering:

Agglomerative: Agglomerative hierarchical clustering is a bottom-up clustering method where clusters have sub-clusters, which in turn have sub-clusters, etc. It starts with the points as individual clusters and, at each step, merge the closest pair of clusters. This requires defining a notion of cluster proximity.

Advantages

1. It can produce an ordering of the objects, which may be informative for data display.

2. Smaller clusters are generated, which may be helpful for discovery.

Disadvantages

1. No provision can be made for a relocation of objects that may have been $'incorrectly'$ grouped at an early stage. The result should be examined closely to ensure it makes sense.

2. Use of different distance metrics for measuring distances between clusters may generate different results. Performing multiple experiments and comparing the results is recommended to support the veracity of the original results.

Divisive: Divisive Hierarchical Clustering is a top-down clustering method. It works in a similar way to agglomerative clustering but in the opposite direction. It starts with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. In this case we need to decide which cluster to split at each step and how to do the splitting.

2.3.2 Density Based Clustering

In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set. Objects in the sparse areas that are required to separate clusters are usually considered to be noise and border points. DBSCAN is the most important and efficient algorithm used to find the density based clusters. In DBSCAN algorithm two core points that are close enough-within a distance *Eps* of one another are put in the same cluster. Likewise, any border point that is close enough to a core point is put in the same cluster as the core point. Noise points are discarded.

2.3.3 K-means

In this thesis, we are going to study $k-means$ technique in depth.

Introduction

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. Next, we update these cluster centroids. We update the cluster centroids by taking mean of all the data points assigned to that centroids. This process continues until the points stop changing their clusters. In order to determine which centroid is closest to a particular data point we have to use a proximity measure. There are several proximity measures in use and typically one is chosen based on the data type that we are trying to cluster. Manhattan, Euclidean, cosine and Bregman divergence are all proximity measures that are commonly used. When we take our proximity measure as Euclidean norm, our objective is to minimize the sum of squared distance of the cluster to its centroid. The sum of squared error(SSE) is given by

$SSE = \Sigma_{i=1}^{K} \Sigma_{x \epsilon C_i} dist^2(x, c_i)$ where,

x is a data point in cluster C_i and c_i is the centroid of cluster C_i. So, $k-means$ clustering is defined as

Definition 1. *K-means clustering. Given a matrix $A \epsilon\ R^{n \times d}$(representing n points(rows), described with respect to d features (columns)) and a positive k denoting the number of clusters, find the $n \times k$ indicator matrix X_{opt} such that*

$X_{opt} = arg \min_{X \epsilon X} \|A - AA^T\|_F^2$

k-means Algorithm

Input: dataset consisting of n points say, $x_1, x_2, \cdots, x_n$, number of clusters k and threshold ϵ

```
1: Randomly take $k$ initial cluster centers $c_1, c_2, \cdots, c_k$.
2: for $i = 1 : n$
       for $j = 1 : k$
3: Calculate the distance of each data point $x_i$ from each initial cluster center $c_j$.
4: Assign the data points to its closest cluster center to get $k$ clusters say, $C_1, C_2, \cdots, C_k$.
       end for
end for
5: for $i = 1 : k$
       // Update the cluster centers
6: Update the cluster centers of each cluster by taking mean of clusters
   $c_i^{'} = \frac{1}{|C_i|} (\sum_{x_i \epsilon C_i} x_i + c_i)$
end for
7: Repeat steps 2 to 6
until: $SSE = \Sigma_{i=1}^{K} \Sigma_{x \epsilon C_i} dist^2(x, c_i) \leq \epsilon$
```

Chapter 3

Issues in Clustering

There have been many definitions of clustering. One of the definition is to represent the collection of objects as a set of abstract points, and define distances among the points to represent similarities, the closer the points, the more similar they are. The other definition is, given a data set we can think its clustering to be the range of some function defined on the data. This function is named as *clustering function*. So,

Definition 2. *[6] A clustering function f is a function that takes a set S of n points with pairwise distances between them, and returns a partition of S. This partition is the clustering of S.*

The clustering function is ought to satisfy three natural properties.

If d is a distance function, we write $\alpha{\cdot}d$ to denote the distance function in which the distance between i and j is $\alpha{\cdot}d(i,j)$.

Scale Invariance: For any distance function d and any $\alpha>0$, we have $f(d)$ $= f(\alpha{\cdot}d)$.

This is simply the requirement that the clustering function should not be sensitive to changes in the units of distance measurement, it should not have a built-in *lengthscale*.

The second property is that the output of the clustering function should be *rich* i.e. every partition of S is a possible output. The formal definition is,

Richness: Range(f) is equal to the set of all partitions of S.

The last property is the **Consistency** property. The clustering function is said to have consistency property if we shrink distances between points inside a cluster and expand distances between points in different clusters, we get the same result. To make it precise, we introduce the following definition.

Let d and d' be two distance functions on S. We say that d' is a Γ transformation of d if

(a) for all i, $j \in$ S belonging to the same cluster of Γ we have $d(i, j) < d(i, j)$; and

(b) for all i, $j \in$ S belonging to different clusters of Γ we have $d'(i, j) > d(i, j)$

Consistency. Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transformation of d, then $f(d') = \Gamma$.

Before telling what is the first issue with clustering, we define what do we mean by *refinement* of a partition.

Definition 3. *[6] We say that a partition Γ' is a refinement of a partition Γ if for every set $C' \in \Gamma'$, there is a set $C \in \Gamma$ such that $C' \subseteq C$.*

Define a partial order on the set of all partitions by $\Gamma' \leq \Gamma$, if Γ' is the refinement of Γ. Following the terminology of the partially ordered sets we say that a collection of partitions forms an *antichain* if it does not contain two distinct partitions such that one is a refinement of the other.

The issue with the clustering is that for each $n \geqslant 2$, there is no clustering function f that satisfies Scale Invariance, Richness, and Consistency.

This result is the immediate consequence of the following theorem.

Theorem 1. *[6] If a clustering function f satisfies Scale-Invariance and Consistency, then Range(f) is an antichain.*

The other issue with clustering is that centroid based clustering functions do not satisfy consistent property.

In centroid based clustering, we select k of the input points as centroids and then define clusters by assigning each point in data set to its nearest centroid. Let S be our data set, T be the set of k centroids and $g : R^+ \rightarrow R^+$ be any continuous, non-decreasing, and unbounded function. The aim of (k, g) centroid clustering function is to minimize the objective function $\Lambda_d^g(T) = \Sigma_{i \in S} g(d(i, T))$, where $d(i, T) = min_{j \in T} d(i, j)$. Then the partition of S into k clusters is given by assigning each point of S to the point of T closest to it.

Theorem 2. *[6] For every $k \geq 2$, and every function g (continuous, non-decreasing and unbounded) and for n sufficiently larger relative to k, the (k, g) centroid clustering function does not satisfy the consistency property.*

Now, as we know that there is no such clustering function that will satisfy all the required properties of clustering, we move towards a different approach. In this approach instead of axiomatizing the clustering function we define another notion quality measure and we axiomatize quality measure to see whether it satisfies all the properties of clustering.

3.1 Quality of the Clusterable data

Definition 4. *[1] A clustering quality measure is a function that maps pairs of the form(dataset,clustering) to some ordered set(say,the set of non negative real numbers) so that these values reflect how 'good' or 'cogent' that clustering is.*

We can cluster the data using different algorithms. Different clustering algorithm will aim to optimize different objective functions and are likely to output different clusterings of the same data set. Since it is often ambiguous which objective function is appropriate for clustering the data set, a user need to apply a clustering quality measure to choose between the outcomes of different algorithms. In this section we will introduce the measures that will state the quality of clustering.

There are two approaches that determines the quality of clustering. One is to axiomatize the clustering functions. Other is to develop the set of requirements (*'axioms'*) of clustering quality measures. This section focuses on the second approach. After introducing the axioms of quality measure we will introduce what are the properties of quality measure and then we will give examples of some quality measure that satisfy these axioms and properties.

For x , $y \in X$ and clustering C of X, we write $x \sim y$ whenever x and y are in the same cluster of clustering C and $x \nsim y$ otherwise.

3.1.1 Axioms of Quality measures

Scale Invariance[1] Quality measure m satisfies scale invariance if for every clustering C of (X, d), and every positive λ, $m(C, X, d) = m(C, X, \lambda d)$. **Isomorphism Invariance**

This axiom ensures that quality measures are independent of point description. That is, if the labels of all points are permuted, keeping the clustering fixed, the quality of the clustering should not change.

Definition 5. *[1] Clustering Isomorphism. Clusterings C and C' over (X, d) are isomorphic, $C \simeq C'$, if there exists a distance preserving isomorphism.*

Definition 6. *[1] Isomorphism Invariance. Quality measure m is isomorphism invariant for all clusterings C, C' over (X, d) ,$m(C, X, d) = m(C', X, d)$.*

This axiom says that following a permutation on the data point's labels, the output of a clustering function should be isomorphic to the output prior to the permutation.

Weak Local Consistency

Quality measure m is weakly locally consistent if $\forall$ clusterings C over (X, d), whenever d' is a C weakly locally consistent variant of d, then $m(C, X, d) \geq m(C, X, d')$.

Note that weak local consistency implies consistency and local consistency.

Co-final Richness Quality measure m satisfies co-final richness if for every pair of non-trivial clusterings C over (X, d) and C' over (X, d') there exists a C-consistent variant, d'', of d such that $m(C, X, d'') \geq m(C', X, d'')$.

3.1.2 Properties of quality measures

Consistent properties[1]

Definition 7. *(Locally Consistent Variant)[1] Distance function d' is a C locally consistent variant of d, for a clustering C over (X, d), if*

•For every cluster C_l of C there is a constant $c_l \leq 1$, such that $\forall x, y \in C_l, d'(x, y) = c_l d(x, y)$.

•There exists a $c \geq 1$ such that for every x, y, $d'(x, y) = c{\cdot}d(x, y)$.

Definition 8. *(Local Consistency)[1] Quality measure m is locally consistent if for all clusterings C over (X, d), whenever d' is a C locally consistent variant of d, then $m(C, X, d) \geq m(C, X, d')$.*

Local consistency has limited application in Euclidean spaces, where clustering often takes place. In Euclidean space, if we shrink each cluster uniformly, the distances between pairs of points in different clusters may change is a non-uniform manner.

Below is a more flexible version of consistency.

Definition 9. *(Weakly Locally Consistent Variant)[1] Distance function d' is a C weakly locally consistent variant of d, where C is a clustering over (X, d), if*

•For every cluster C_l of C there is a constant $c_l \leq 1$, such that $\forall$ x, $y \in C_l$, $d'(x, y) = c_l{\cdot}d(x, y)$.

•For every $x \not\sim y$, $d'(x, y) \geq d(x, y)$.

•For some set of points containing a point p_l from every cluster C_l, $\exists$ a constant $c \geq$ 1 such that, for every p_l, p'_l , $d'(p_l, p'_l\ ) = c \cdot d(p_l, p'_l\ )$.

Richness properties

Definition 10. *(Refinement)[1] A clustering C' of X is a refinement of clustering C of X if for every cluster C_i of C, $\exists$ a set of clusters in C' that partition C_i.*

Definition 11. *(Refinement Preference)[1] Quality measure m is refinement-preferring if for every clustering C of (X, d) that has a non-trivial refinement, there exists a non-trivial refinement C' of C such that $m(C', X, d) > m(C, X, d)$.*

For any refinement-preferring measure, given any clustering (that has a non-trivial refinement) over some data set, there is a non-trivial clustering of the data set with better quality. Thus, refinement-preferring measures are not rich.

3.1.3 Examples of quality measure

Loss-based quality measures Here, we will present quality measure for a loss based clustering. One of the example of a loss based clustering is k means whose objective function is to minimize SSE.

A *clustering loss function* is a function $\pounds$: $C_X \times D \to R^+ \cup \{0\}$, where C_X is the set of clusterings of data set X, and D is the family of distance functions over X.

Given a data set and a distance function d, we get clusterings of the set. Let C, C' be the clusterings of (X, d). The loss function assigns to each clustering a non-negative value. This value will tell us how good our clustering is compared to other clusterings.

£-Clustering quality

£-Clustering Quality is a quality measure that normalizes clustering loss function $\pounds$. Let C_{all} denote the 1-clustering of X, that is, the clustering that groups all points in X into the same cluster, then £-Clustering Quality of a clustering C over (X, d) is given by

$\pounds\text{-}CQ(C, X, d) = \frac{\pounds(C_{all}, X, d)}{\pounds(C, X, d)}$

Now while comparing two clusterings of a data set, we expect that clustering with lower loss to have better clustering quality. Whenever quality measure satisfy the property that clustering with lower loss are better, we say quality measure conforms with loss function.

This quality measure satisfies all the properties and axioms listed above.

Center-based quality measures A clustering $C = \{C_1, C_2, ..., C_k\}$ is center based if there exist points, called centers, $c_1 \in C_1$, $c_2 \in C_2$, . . . , $c_k \in C_k$, such that for all $x \in C_i$, $d(x, c_i) < d(x, c_j)$, for all $i \neq j$.

Relative Margin: This quality measure considers the ratio of the distance from each point in the dataset to its closest center, to the distance from the point to its second closest center. The smaller the ratio, the more it is sure to which cluster it belongs. We use average ratio as quality measure.

Let C be a center based clustering over (X, d)

The C-relative point margin of $x \in X$ is $C\text{-}RM_{X,d}(x) = \frac{d(x,c_i)}{d(x,c_j)}$, where c_i is the closest center to x, c_j is a second closest center to x.

Relative Margin: The relative margin of a center-based clustering C over (X, d) is $RM_{X,d}(C) = avg_{x \in X \setminus R}\ C\text{-}RM_{X,d}(x)$ where R is the set of centers in C. The range of relative margin is $[0, 1)$, and lower relative margin indicates a better clustering.

The above mentioned quality measures satisfy wide range of clustering techniques and are able to compute the clustering quality of a clustering in low polynomial time.

Now, we look at whether the given data is efficient enough to give a good clusterable structure. The next section deals with the study of clusterability of data sets.

3.2 Clusterability of Data Sets

Clusterability as the name suggests tells that whether the given data set has a good clusterable structure or not. Several notions of clusterability have been discussed in the literature. But for each pair of notions, there are data sets that are arbitrarily well clusterable by one of the notions, but poorly clusterable by the other notion. Some of the notions of clusterability that have been discussed in the literature are:

Variance ratio Clusterability

Variance is the measurement of the spread between numbers in a data set, it measures how far is the each number in the set from the mean. Variance of a cluster is the expected square distance to the centroids. In order to get a good clustering we want the variance within a cluster to be as small as possible and variance between clusters to be as large as possible. To measure the clustering quality [10] introduced a notion called variance ratio given by $VR_k(X) = \frac{\max\limits_{c \in C} B_c(X)}{W_c(X)}$, where, X is the dataset, C is the set of k-means optimal clusterings of X, $B_C(X) = \Sigma_{i=1}^{k} p_i \parallel centerofmass(X_i) - centerofmass(X) \parallel^2$ is the between-cluster variance of C and $W_C(X) = \Sigma_{i=1}^{k} p_i \sigma^2(X_i)$, the within-cluster variance of C.

Definition 12. *[2] (Variance Ratio). The variance ratio of X for k is* $VR_k(X) = \max\limits_{c \in C} \frac{B_c(X)}{W_c(X)}$, *where C is the set of k-means optimal clusterings of X.*

The range of variance ratio is $[0,\infty)$ and higher values of variance ratio indicate better clusterability.

Separability Clusterability

The separability notion of clusterability captures how sharp is the drop in the loss function when moving from a $(k-1)$- clustering to a k-clustering. This notion was introduced in [2].

Definition 13. *[2] (Separability). A data set X is* (k, ϵ)*- separable if* $OPT_k(X) \leq OPT_{k-1}(X)$.

The range of separability is [0, 1). A data set has better separability than another data set if it is separable for smaller ϵ.

Worst pair ratio Clusterability

The minimum distance between two points in different clusters of a clustering C is called the split between the two clusters, and the minimal split between two clusters is called the split of C; that is, $split_C(X) = min_{x \nsim y} \parallel x - y \parallel$. The maximum distance between two points within a cluster in C is called the width of the cluster, and the maximal width of a cluster in C is called the width of C, $width_C(X) = max_{x \sim y} \parallel x - y \parallel$.

Definition 14. *[2] (Worst Pair Ratio). The worst pair ratio of X is WPR(X)* $= \{max \frac{split_C(X)}{width_C(X)} \mid C$ *a clustering of* $X\}$.

The range of worst pair ratio is $[0,\infty)$ and higher values of worst pair ratio mean better clusterability.

Center Perturbation Clusterability

In a center-based clustering, each point in a cluster is closer to its own cluster center than to the center of any other cluster. If the given data set has a $'good'$ clusterable structure, perturbing the centers won't affect the clusterability of data set.

Given a loss function L, let $OPT_{\mathbf{L},k}(X) = \min\{L(C) \mid C$ is a k-clustering of $X\}$, the loss of a k-clustering of X that minimizes L.

Definition 15. *[2] (ϵ-close). Two center-based clusterings, C and C' of X, are ϵ-close, if there exist centers $c_1, c_2, \ldots, c_k$ of C, and centers $c'_1, c'_2, \ldots, c'_k$ of C', such that for all $i \leq k$, $\parallel c_i - c'_i \parallel \leq \epsilon$.*

Definition 16. *[2] (Center Perturbation Clusterability). A data set X is (ϵ, δ)-CP clusterable for k (for ϵ, $\delta \geq 0$), if for every clustering C of X that is ϵ -close to some optimal k-clustering of X, $\boldsymbol{L}(C) \leq (1+\delta) OPT_{\mathbf{L},k}(X)$.*

Now, after studying quality measures and clusterability of data sets, we look at the ways by which we can make our algorithm strong enough so that it can handle noise.

3.3 Robustness in the presence of noise

The important feature of the clustering algorithm is that it should be able to cluster all the points in the data. However, it is the often the case that datasets one wishes to cluster contains a significant subset which is unstructured, such a subset is referred as *noise*, which tends to disrupt clustering algorithms and makes it difficult to detect the cluster structure of remaining points. We want our clustering algorithm to be noise robust, this can be done by transforming the original algorithm to robustified algorithm. Till now, two such robustified paradigms have been introduced. Before introducing these paradigms, we define some notations.

We consider a scenario in which the input dataset X consists of two part: a clusterable subset I, which is also called the un-noised data, and an added noise set $X \setminus I$ (the identity of which is not known to the clustering algorithm). We consider two clustering algorithms, the original one, A , and its robustified transformation $R_p(\mathrm{A})$ that is obtained by using our paradigm with a robustifying parameter p. A robustifying parameter, p, denotes the degree to which an algorithm should be robustified to noise; For example, the number of extra clusters that can be used or a notion of distance beyond which a point is considered an outlier. A robustifying paradigm, $Rp(\mathrm{A})$, is a function that takes a clustering algorithm A and returns a robustified clustering algorithm $R_p(\mathrm{A})$ based on the robustifying parameter p. We refer to A as the ground clustering algorithm of $R_p(\mathrm{A})$.

Definition 17. *[4] p-Increased Paradigm. The p-Increased Paradigm is a robustifying paradigm, $RI_p(\cdot)$, that takes as input a $(k; g)$-centroid algorithm and returns a $(k + p; g)$- centroid algorithm.*

The next paradigm is parameterized by the distance after which a point should be considered an outlier. To define this paradigm, we first introduce a class of algorithms. Given a space E and distance function d, the δ-truncated distance function corresponding to d is the function d' such that $d'(x; y) = \min\{\delta,\ d(x,\ y)\}$ for $x,\ y \in E$. The $(k,\ g)$-δ-truncated algorithm is an objective based algorithm that, given $X \subseteq E$, first optimizes the function.

Definition 18. *(δ-Truncated Paradigm)[4]. The δ-Truncated Paradigm is a robustifying paradigm, $RT_\delta(.)$, that takes as input a (k, g)-centroid algorithm and returns a (k, g) δ-truncated algorithm.*

3.3.1 Measures of Robustness

Previously, robustness of the algorithm to the addition of noise was measured by comparing the output of the same algorithm on both noised and un-noised data. This approach lead to pessimistic results about the possibility of achieving noise robustness. The approach that works well for measuring the robustness of the algorithm is by comparing the output of a robustified algorithm on noisy data to the output of its corresponding ground algorithm on the unnoised data. Before defining the robustness measures based on this approach, we fix some notations.

Let A denote any clustering algorithm (ground clustering) and $A' = R_p(A)$ be the robustifying paradigm with parameter p corresponding to A. Given $I \subseteq X$, $A(I)$ denotes the clustering of I using ground algorithm, and $A'(x)$ denotes the clustering of X by the robustified algorithm. We consider I to be robust (to $X \setminus I$) with respect to the $R_p(A)$ algorithm if certain properties of $A(I)$ are preserved in $A'(X)$. For any $x \in X$, we use $\mu(x)$ and μ'(x) to denote the centers of $A(I)$ and $A'(X)$, respectively, to which x belongs.

Cluster centers are commonly used to compress and represent data. The distances between points and their corresponding centers can be viewed as the distortion of such a compression. Therefore, it is essential to have clustering algorithms where this value does not grow significantly in the presence of noise. The first robustness measure measures how much this distortion is affected by the addition of noise to the input data.

• **α-distance-robust**. A subset $I \subseteq X$ is α-distance-robust with respect to A' if for all $y \in I$, $d(y, \mu) \leq d(y, \mu(y')) + \alpha$.

An algorithm is considered robust, if it separates the input using some low-cost clusters that cover the un-noised data. The next robusteness measure captures this property by computing minimal cost of a subset of clusters that covers at least $\mid I \mid$ points in total.

• **β-cost-robust**. Let Λ be an objective (cost) function. $I \subseteq X$ is a β-cost-robust with respect to A' for Λ , if there exists $C^* \subseteq A'(X)$, such that $|\bigcup C^*| \geq \mid I \mid$ and $\Lambda(C^*) \leq \Lambda(A(I)) + \beta$.

3.4 Purity and Normalized Mutual Information

Typical objective functions in clustering formalize the goal of attaining high intra-cluster similarity (documents within a cluster are similar) and low inter-cluster(documents from different clusters are dissimilar). This is an internal criterion for the quality of a clustering. An alternative to internal criteria is direct evaluation in the application of interest. We will discuss about two external quality measures, *Purity and Normalized Mutual Information*[5].

To compute *purity*, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by total number of documents.

Formally:
$purity(\Omega, C) = \frac{1}{N}\ \Sigma_k max_j |\omega_k \cap c_j|$
where $\Omega = \omega_1, \omega_2, \cdots, \omega_k$ is the set of clusters and $C = c_1, c_2, \cdots, c_j$ is the set of classes.

High purity is easy to achieve when the number of clusters is large, in particular, purity is 1 if each document gets its own cluster. Thus, we cannot use purity to trade off the quality of the clustering against the number of clusters. A measure that allows us to make this tradeoff is *normalized mutual information or NMI*

Before stating what is *normalized mutual information*, we will define what do we mean by *mutual information.*

Definition 19. *Mutual Information. Mutual Information is the measurement of how much information the presence/absence of a term contributes to making the correct classification decision. Mutual Information is denoted by I and is given by*

$$I(\Omega; C) = \sum_k \sum_j P(w_k \cap c_j) log\ \frac{P(w_k \cap c_j)}{P(w_k)P(c_j)}$$
$$= \sum_k \sum_j \frac{|w_k \cap c_j|}{N} log \frac{N|w_k \cap c_j|}{|w_k||c_j|}$$

$where P(w_k), P(c_j) and P(w_k \cap c_j)$ *are the probabilities of a document being in cluster* w_k*, class* c_j *and in the intersection of* w_k *and* c_j *respectively.*

These two equations are equivalent for the maximum likelihood estimates of the probablities(i.e., the estimate of each probability is the corresponding relative frequency).

So, *normalized mutual information* or *NMI* is given by
$NMI(\Omega, C) = \frac{I(\Omega, C)}{[H(\Omega) + H(C)]/2}$ where,
H is the entropy given by
$H(\Omega) = -\sum_k P(w_k) log P(w_k)$

$$= -\sum_{k} \frac{|w_k|}{N} \log \frac{|w_k|}{N}$$

where, again, the second equation is based on maximum likelihood estimates of the probabilities.

$I(\Omega; C)$ measures the amount of information by which our knowledge about the classes increases when we are told what the clusters are. The minimum of $I(\Omega; C)$ is 0 if the clustering is random with respect to class membership. In that case, knowing that a document is in a particular cluster does not give us any new information about what its class might be.

Chapter 4

Clustering of high dimensional data

Now we will study clustering of high dimensional data where the number of samples is smaller than the dimensionality of data.

Density Estimation helps in obtaining information and gaining understanding about the distribution of the underlying data set. Since, clustering can also be done using density estimation, analyzing the number of samples required for accurately recovering the underlying distributions, referred to the problem of sample complexity, is a challenging open problem.

It is well known that the number of samples needed for accurate density estimation is at least exponential in the dimensionality. So, the problem that arises is,

Is it possible to achieve accurate clustering results when the data dimensionality is larger than the number of samples to be clustered?

The problem of computationally expensive was partially solved in [3] where the authors studied a special case of this problem where data points were sampled from a mixture of two isotropic Gaussians. The authors showed that when the cluster centers are d dimensional s-sparse vectors (i.e. there are no more than s non-zero entries), the sample complexity can be reduced to $O(s^2 \log d)$. But now, the result has been proved for more general case. It has been shown that when there are K clusters, $K > 2$, data points that are sampled from a mixture of $K \geq 2$ spherical Gaussians with s-sparse centers, require only $O(s \log d)$ samples to reliably estimate the cluster centers. And finally it has been proved that it is indeed possible to reliably cluster high-dimensional data even when the number of samples is smaller than the dimensionality as long as the cluster centers are sparse.

Since, we are dealing with high dimensional data, we will study a method that can lower the dimension of the data and still gives efficient results.

4.1 Principal Component Analysis

4.1.1 Introduction

PCA is the method used to reduce the number of features that represent data. The benefits of this *dimensionality reduction* include providing a simpler representation of the data, reduction in memory, and faster classification. In PCA we project the data from a higher dimension to a lower dimensional manifold such that the error incurred by reconstructing the data in the higher dimension is minimized. Principal component analysis uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components.

It is useful when we have data on number of variables (possibly large number of variables), and believe that there is some redundancy in those variables. Here, redundancy means that some of the variables are correlated with one another, possibly because they are measuring the same construct. Because of this redundancy, it is possible to reduce the observed variables into a smaller number of principal components, that will account for most of the variance in the observed variables.

4.1.2 Idea behind PCA

The main idea behind principal component analysis is to first find a direction that corresponds to maximal variance between the data points. The data is then projected on the hyperplane orthogonal of that direction. We obtain a new data set, and find a new direction of maximal variance. We may stop the process when we have collected enough directions.

4.1.3 Assumptions of PCA

1. Linearity. PCA assumes the data set to be linear combinations of the variables.

2. PCA assumes that directions of maximum variance contains features representing the data but there is no guarantee that the directions of maximum variance will contain good features for discrimination.

3. PCA assumes that components with larger variance correspond to interesting dynamics and lower ones correspond to noise.

4.1.4 PCA Interpretation

PCA can be interpreted in two different ways.

1.Maximize the variance of projection along each component.

2.Minimize the reconstruction error (ie. the squared distance between the original data and it's estimate).

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. When we have a set of data points, we deconstruct the set into eigenvectors and eigenvalues. An eigenvector is a direction, and an eigenvalue is a number, which tells how much variance is there in the data in that direction. The eigenvector with the highest eigenvalue is therefore the principal component.

4.1.5 Computing PCA using covariance method

Definition 20. *Covariance. Covariance is a measure of how changes in one variable are associated with changes in a second variable. Specifically, covariance measures the degree to which two variables are linearly associated.*

Let X and Y be two vectors of dimension n. Then, the covariance between X and Y is given by
$cov(X,Y) = \frac{\Sigma_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{n-1}$

Suppose we have a dataset consisting of n observations where each observation has m variables and we want to reduce the data so that each observation can be described with only l variables,i.e, $l < m$. Given below is the algorithm for computing PCA using covariance matrix.

Step 1. Compute the covariance matrix C. C will be a $m \times m$ symmetric matrix where each entry c_{ij} is the covariance between variable i and variable j.

Step 2. Compute eigenvalues and corresponding eigenvectors of the covariance matrix C.

Step 3. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $m \times k$ dimensional matrix. These eigenvectors are the principal components.

Step 4. Use this $m \times k$ eigenvector matrix to transform the observations onto the new subspace.

Next, we study a method for doing feature selection.

4.2 LASSO

LASSO is a regression method proposed by R.Tibershani in 1996. Similar to ordinary least squares regression, LASSO minimizes the residual sum of squares but poses a constraint to the sum of absolute values of the coefficients being less then a constant. This simple modification also allows LASSO to perform variable selection because the shrinkage of the coefficients is such that some coefficients can be shrunk exactly to zero. The lasso estimator β is defined by

$\beta = argmin \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |\beta_j|$

or equivalently,

$\beta = argmin \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \beta_j x_{ij})^2$ subject to $\sum_{j=1}^{p} |\beta_j| \leq t$

where n is the number of objects, p is the number of variables and λ is a parameter, which can be tuned in order to set the shrinkage level, the higher the λ is, the more coefficients are shrunk to zero.

4.2.1 Selection of Tuning Parameter

Selection of tuning parameter is very important as it has a big influence on the performance of the estimator. Cross-validation is considered the simplest and most widely used method for the minimization of the prediction error. The most common forms of cross-validation are $k-$fold and leave one-out cross-validation.

4.2.2 $k-$ fold cross-validation

In $k-$fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k1$ subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds are then averaged to produce a single estimation. Suppose the data consists of n observations.

4.2.3 Leave one-out cross-validation

The choice $k = n$ in $k-$fold cross-validation is known as leave one-out cross-validation, in this case we have n subsamples and for the i^{th} subsample, the fit is computed using all the data after omitting i^{th} observation.

Chapter 5

Algorithms for clustering high dimensional data

Given below is the datasets we have used for implementing our clustering algorithms.

5.1 Yale Dataset

The Yale dataset consists of 165 grayscale images in GIF format of 15 individuals. There are 11 images of each individual one per subject, one per different facial expression or configuration: center-light, with glasses, happy, left-light, without glasses, normal, right-light, sad, sleepy, surprise and wink.

As each image is a matrix of pixel values, the size of matrix of each image in yale dataset is 243×320. After vectorizing the image, we get 1-dimensional vector of size 1×77760. So, the size of total dataset is 165×77760.

Download link :

$http://vision.ucsd.edu/datasets/yale_face_dataset_original/yalefaces.zip$

Following is the one of the image from the Yale dataset.

5.2 Algorithms

We have applied PCA on each of the algorithm to reduce the dimension of the dataset and to lower the computational complexity of the algorithm and all the algorithms have been implemented in MATLAB.

Figure 5.1: Image from Yale dataset

5.2.1 High dimensional clustering algorithm

An efficient clustering algorithm was designed that only needs to go through all the data points once to obtain an accurate estimation of cluster centers. This is in contrast to many clustering algorithms, such as k-means which require going through the data set multiple times before the final centers can be determined.

We will state what that algorithm is and how it works. Let $D = \{x_1, x_2, ..., x_n\}$ be the set of n data points to be clustered into K clusters, where each $x_i \in R^d$ is a vector of d dimensions. The proposed algorithm is an iterative procedure. Without loss of generality, we assume $n = T(2^m - 1)$ for some integers T and m. The algorithm first randomly divides the collection of n data points into m subsets, denoted by S_1,...,S_m, with $\mid S^i \mid = T2^{i-1}$. The initial guess for cluster centers is denoted by $\hat{c_1}^1$,...,$\hat{c_k}^1$ Given the initial cluster centers, the algorithm iteratively updates them. At each iteration t, it uses the data points in S^t, and identify, for each data point $x_i^t \in S^t$, its closest cluster $\hat{k_i}^t$ by using some suitable distance metric.

After computing the cluster memberships, next step is to update the cluster centers, given the estimated cluster centers at iteration t, $\{\hat{c_k}^t\}_{k=1}^K$, we denote by $\hat{S_K}^t$ the subset of data points in S^t that are assigned to $\hat{c_k}^t$. At iteration $t + 1$, we take the new cluster center as the average of data points in $\hat{S_K}^t$

```
N = Istack;
m = 4;
K = 15;
T = 11;
[coeff,score,latent,tsquared,explained,mu] = pca(N);
n = N * coeff(:,1:150);
[idx clustercenters] = kmeans(n,15);
```

```
c = clustercenters;
f = randperm(165);
S = [ ];
ma = [ ];
arr=[ ];
IP=[ ];
ar=[ ];
B=[ ];
for i = 0 : 3
    A=[ ];
    X=[ ];
     N=[ ; ];
    q = 11 * (2^i − 1);
    p = 11 * (2^i);
for j = q+1:1:q+p
    for l=1:K
         ma(j,l)= sum(n(f(j),:).*c(l,:));
end
    [Y,arr] = max(ma(j,:));
     A = [A arr];
end
for l=1:K
    J=numel(find(A==l));
     X=[X J];
    N(l,1:J)= find(A==l);
              if X(l)==0
                   c(l,:)=c(l,:);
              else
                   c(l,:)=sum( n(f(N(l,1:J)),:))/X(l);
end
end
clear A;
clear X;
clear N;
clear S;
end
for i = 1:165
    for l=1:K
```

```
        IP(i,l) = sum(n(i,:).*c(l,:));
end
    [Y1 ar] = max(IP(i,:));
    B = [B ar];
end
     for d = 1:165
        for j = 1:15
            distancematrix(d,j) = $norm(n(d,:) - c(j,:))^2$;
end
end
[M,I] = min(distancematrix');
```

5.2.2 k-means

This algorithm is the implementation of what we have studied in 5.2.2

```
    n = Istack./255;          //n is the dataset
K = 15;                       //number of clusters
datadim = length(n(1,:));         //dimension of data
nbData = length(n(:,1));          //number of points in data
Matrix = [ ];
distancematrix = [ ];
    //initializing the centroids randomly
datamin = min(n);
datamax = max(n);
datadiff = datamax - datamin ;
centroid = 255*rand(K, datadim);
for i=1 : 1 : length(centroid(:,1))
centroid( i , : ) = centroid( i , : ) .* datadiff;
centroid( i , : ) = centroid( i , : ) + datamin;
end                      // end init centroids
posdiff = 10000000;
while posdiff > 160.0
assignment = [ ];      //assign each datapoint to the closest centroid
for d = 1 : nbData;
   mindiff = ( n( d, :) - centroid( 1,:) );
   mindiff = mindiff * mindiff';
curAssignment = 1;
for c = 2 : K;
```

```
        diff2c = ( n( d, :) - centroid( c,:) );
        diff2c = diff2c * diff2c';
    if( mindiff ≥ diff2c)
         curAssignment = c;
         mindiff = diff2c;
    end
    end
    assignment = [ assignment; curAssignment];     //assign the d-th dataPoint
    end
         // for the stoppingCriterion
    oldPositions = centroid;
    centroid = zeros(K, datadim);      // recalculate the positions of the centroids
    pointsInCluster = zeros(K, 1);
    for d = 1: length(assignment);
         centroid( assignment(d),:) = centroid( assignment(d),:) + n(d,:);
         pointsInCluster( assignment(d), 1 ) = pointsInCluster( assignment(d), 1
) + 1;
    end
    for c = 1: K;
        if( pointsInCluster(c, 1) = 0)
        centroid( c , : ) = centroid( c, : ) / pointsInCluster(c, 1);
    else
        //set cluster randomly to new position
        centroid( c , : ) = (rand( 1, datadim) .* datadiff) + datamin;
    end
    end
         //stoppingCriterion
    for i = 1:length(centroid(:,1))
         Matrix(i,1) = norm(centroid(i,:) - oldPositions(i,:));
    end posdiff = max(Matrix);
    end
          // calculating the closest centroid
    for d = 1:nbData
         for j = 1:length(centroid(:,1))
              distancematrix(d,j) = norm(n(d,:)-centroid(j,:));
    end
    end
    [M, I] = min(distancematrix');
```

5.2.3 Subset High dimensional clustering algorithm

To improve the results of $k-$means algorithm, we modified it using the concepts from High Dimensional Clustering Algorithm(H.D.C.A). First, we randomly take k initial centers and randomly divide the initial dataset into m subsets as in H.D.C.A. Then using the data points from the first subset we update our initial cluster centers using $k-$means algorithm. Next, we apply $k-$ means to updated centroids and data points from second subset, and repeat this process till m subsets.

```
    N = Istack;
m = 4;
K = 15;
T = 11;
[coeff,score,latent,tsquared,explained,mu] = pca(N);
n1 = N * coeff(:,1:155);
n = n1./255;
[idx clustercenters] = kmeans(n,15);
centroid = clustercenters;
f = randperm(165);
datadim = length(n(1,:));
nbData = length(n(:,1));
Matrix = [ ];
distancematrix = [ ];
datamin = min(n);
datamax = max(n);
datadiff = datamax - datamin ;
posdiff = 10000;
while posdiff > 160.0
assignment = [];
for i = 0: 1 : 3;
    q = 11 ∗ (2^i − 1);
    p = 11 ∗ (2^i);
   for d = q+1:1:q+p
     mindiff = ( n( d, :) - centroid( 1,:) );
     mindiff = mindiff * mindiff';
     curAssignment = 1;
       for c = 2 : K;
       diff2c = ( n( d, :) - centroid( c,:) );
       diff2c = diff2c * diff2c';
```

```
if( mindiff ≥ diff2c)
    curAssignment = c;
    mindiff = diff2c;
end
end
assignment = [ assignment; curAssignment];
end
oldPositions = centroid;
centroid = zeros(K, datadim);
pointsInCluster = zeros(K, 1);
for d = 1: length(assignment);
centroid( assignment(d),:) = centroid( assignment(d),:) + n(d,:);
pointsInCluster( assignment(d), 1 ) = pointsInCluster( assignment(d), 1 )
+ 1;
end
for c = 1: K;
    if( pointsInCluster(c, 1)  = 0)
    centroid( c , : ) = centroid( c, : ) / pointsInCluster(c, 1);
else
    centroid( c , : ) = (rand( 1, datadim) .* datadiff) +
datamin;
end
end
for i = 1:length(centroid(:,1))
     Matrix(i,1) = norm(centroid(i,:) - oldPositions(i,:));
end
posdiff = max(Matrix);
end
clear assignment;
end
for d = 1:nbData
    for j = 1:length(centroid(:,1))
        distancematrix(d,j) = norm(n(d,:)-centroid(j,:));
end
end
[M,I] = min(distancematrix');
```

5.2.4 Results

We have implemented all three algorithms using four distance measures that are L_1 norm, L_2 norm, L_∞ norm and Inner product. A good reference for different families of distance measures can be found in [7]. We calculated *NMI*, Variance ratio clusterability for all the algorithms, and took the average values of 3-4 runs. All the values have been shown in the table below

Norm/Measure	$k-$means	H.D.C.A	S.H.D.C.A
L_1/NMI	0.6442	0.4190	
L_1/V.R.C	4.8×10^{28}	1×10^{28}	
L_2/NMI	0.7787	0.5992	0.6116
L_2/V.R.C	4.5×10^{22}	6×10^{28}	8×10^{27}
L_∞/NMI	0.7456	0.6329	0.7753
L_∞/V.R.C	5×10^{28}	2.9×10^{28}	1.37×10^{28}
Inner product/NMI	0.3949	0.3482	
Inner product/V.R.C	7×10^{27}	2.5×10^{27}	

Chapter 6

Conclusions

In this work, we have studied the theoretical foundations of clustering. We started with what is cluster analysis and then studied some clustering techniques. Then we get into what are issues in clustering and how to conclude whether our clustering is good or not. For this, we studied some measures such as variance ratio clusterability, worst pair ratio clusterability and separability clusterability that tell how good is our clustering. Then we studied how to cluster high dimensional data.

Since, dealing with high dimensional data is computationally expensive, we looked into a method called Principal Component Analysis that can lower the dimension of the data and still gives efficient results. Principal component analysis uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. Then we studied some algorithms to cluster high dimensional data. To implement our algorithms we took a standard database called the Yaledata base which consists of 165 grayscale images of 15 different people. The first algorithm we looked at is $k-$means algorithm, the main idea of $k-$means algorithm is to define k centers, one for each cluster,then each point is assigned to its nearest cluster center. Next, we update these cluster centers by taking mean of all the data points assigned to that centers. This process continues until the points stop changing their clusters.Then we studied high dimensional clustering algorithm in which we first randomly take user defined number of clusters and then randomly divide the dataset into subsets and at each step we update our cluster centers using the points in these subsets. Then we studied a modified algorithm using the concepts from $k-$ means and high dimensional clustering algorithm. In this algorithm first we randomly take k

initial centers and randomly divide the initial dataset into m subsets as in H.D.C.A. Then using the data points from the first subset we update our initial cluster centers using $k-$means algorithm. Next, we apply $k-$ means to updated centroids and data points from second subset, and repeat this process till m subsets. Then to do comparative study between all these algorithms we took two measures called Normalized mutual information and variance ratio clusterability and calculated their values for all the algorithms using four distance measures that are L_1 norm, L_2 norm, L_∞ norm and Inner Product. From the table we have made, we can see that $k-$ means gives best results followed by subset high dimensional clustering algorithm and high dimensional clustering algorithm. From this, we conclude that its better to divide the set into subsets and apply $k-$ means on each disjoint subsets rather then dividing the set into disjoint subsets and updating the cluster centers using a single pass algorithm.

Bibliography

[1] Margareta Ackerman, Shai Ben-David (2008): Measures of Clustering Quality: A Working Set of Axioms for Clustering, *Advances in Neural Information Processing Systems (NIPS)*,21.

[2] Margareta Ackerman, Shai Ben-David(2009) Which Data Sets are 'Clusterable'?- A Theoretical Study of Clusterability, *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS), Florida, USA*, 5.

[3] (2013) Minimax theory for high-dimensional guassian mixtures with s-sparse mean separation, *Neural Information Processing Systems (NIPS)*,2139-2147

[4] Shai Ben-David, Nika Haghtalab(2014) Clustering in the Presence of Background Noise, *Proceedings of the 31 st International Conference on Machine Learning, Beijing, China*, 32.

[5] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schtze(2009) An Introduction to Information Retrieval, *Cambridge University Press Cambridge, England.*

[6] Jon Kleinberg. (2002): An Impossibility Theorem for Clustering, *Advances in Neural Information Processing Systems (NIPS)*, 15.

[7] Sung-Hyuk Cha. (2007): Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions, *INTERNATIONAL JOURNAL OF MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES*, 1,300-307.

[8] Pang-Ning Tan, Michael Steinbach, Vipin Kumar(2005) Introduction to Data Mining, *Addison Wesley.*

[9] Tibshirani, Robert(1996): Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society. Series B (Methodological)*, 58.

[10] Jinfeng Yi, Lijun Zhang, Jun Wang, Rong Jin, Anil K. Jain (2014): A Single-Pass Algorithm for Efficiently Recovering Sparse Cluster Centers of High-dimensional Data, *Proceedings of the 31st International Conference on Machine Learning, Beijing, China*, 32.