# An extendable general platform for cluster analysis and validation

Brandon Troy Edwards

# AN EXTENDABLE GENERAL PLATFORM FOR CLUSTER ANALYSIS AND VALIDATION

---

A Thesis

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

Brandon Troy Edwards

June 2009

# AN EXTENDABLE GENERAL PLATFORM FOR CLUSTER ANALYSIS AND VALIDATION

---

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Brandon Troy Edwards

June 2009

Approved by:

[redacted]

Haiyan Qiao, Chair, Computer Science and Engineering — 6/11/2009 Date

[redacted]

Ernesto Gomez

[redacted]

Keith Schubert

# ABSTRACT

Cluster analysis plays an important role in data analysis and knowledge discovery. It is used in a large range of fields, including market research, city-planning, earthquake studies, and scientific research areas, such as bioinformatics. As the complexity and amount of data increases, improvement to old techniques as well as development of novel algorithms is needed. While cluster analysis is a useful, unsupervised technique for analyzing data, there are no clustering algorithms that can be uniformly applied to all sets of data. Many of the algorithms need input parameters that require the user to have some pre-existing knowledge about the data, such as the number of clusters the data holds. This thesis addresses current problems in the area of cluster analysis, such as estimating the number of clusters, detecting outliers, and offering useful visualization of multi-dimensional data. Solutions to these problems as well as a user-friendly clustering platform that aids the user in obtaining useful clustering results are presented.

## ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Haiyan Qiao, and my committee members, Drs. Ernesto Gomez and Keith Schubert for their guidance as I worked on my thesis.

Most of all I would like to thank my mother and father for their support during my entire career as a student.

TABLE OF CONTENTS

## LIST OF FIGURES

# 1. INTRODUCTION

Cluster analysis is an unsupervised knowledge discovery technique for data analysis. As databases grow larger and increase in complexity, cluster analysis strives to find more efficient and effective techniques for analyzing the data. It is one of the primary data analysis techniques used to extract useful information from a set of data objects.

Cluster analysis looks for data objects with similarities in a set of data and puts those with similar characteristics into a group called a cluster. Data objects that are within the same cluster are similar to one another, while data objects in different clusters are dissimilar to one another. A high quality clustering result contains high intra-cluster connectivity and low inter-cluster connectivity, where intra-cluster connectivity is defined as the average distance between data objects contained within a cluster, meaning a good result will have very compact clusters. Inter-cluster connectivity is the average distance between the clusters themselves. This means with a low inter-cluster connectivity, the clusters are spread far apart from one another.

A number of clustering algorithms have been proposed and while there is some benefit to have a large range of options available for clustering, it may become confusing to a user as to which algorithm will best suit their needs. It is important for a user to be able to select the most appropriate algorithm for their data set. Little work has been done to define performance criteria for clustering algorithms that will aid a user in selecting the best suited algorithm. This thesis implements a general platform which integrates several commonly used cluster analysis algorithms with ease of use and extendibility, along with various validation techniques to analyze the results of the algorithms. The user does not need any special knowledge in order to use this platform and based on the analysis of the results, the user will be able to decide which algorithm provides the highest cluster quality and which input parameters should be used for their dataset.

The following sections contained within this chapter will describe several commonly used clustering algorithms, distance measurements, and validation techniques, followed by the significance and objective of this thesis. The next chapter will go into the methodology, describing which algorithms were integrated into the platform and why. Chapter 3 will detail the implementation of the platform and the final chapter contains the conclusion of this thesis with a discussion on future work.

## *1.1 Background*

Clustering aims to classify data objects into a set of groups based on a measure of their similarities. By representing data objects with clusters you tend to lose some of the details of the data. Through this loss of information, simplification is gained in that the clustering result defines the patterns and structure that are found within the data. This information is useful and must be interpreted by users.

Cluster analysis is unsupervised, meaning it does not rely on a set of pre-described details on how to cluster the data, such as is the case with classification. This is what separates data clustering and data classification.

There are four basic steps in the path of cluster analysis [34]. First, given a dataset, some preprocessing and feature selection needs to be done in order to prepare the data for clustering. Next, whatever clustering algorithm will be used is executed and generates the clusters or structure of the data. Following that, cluster validation methods are used to determine the cluster quality of the results of the clustering algorithm. Finally, the results must be interpreted and some useful information garnered from them. During any step through this process, the previous step may be re-executed. For example, if the cluster quality given by the validation method is low, the clustering algorithm may be ran again with different input parameters to generate a new, more accurate result.

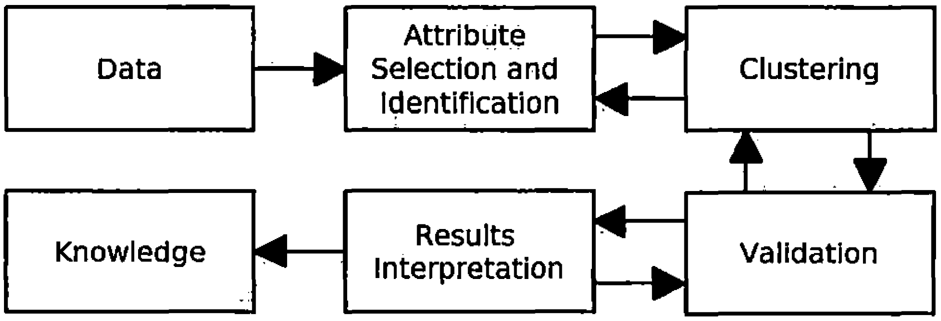


*Fig. 1.1:* Knowledge Discovery Flow Chart

One area where clustering is currently being used is marketing research. Market researches use cluster analysis to find relationships between different consumers. They generally gather their information through use of surveys and product test markets. This information can then be used in product placement, determining the target audience for a product, and in the development of new products.

Another area of current intense research being done in cluster analysis, is bioinformatics. Researches in this area use clustering algorithms to analyze micro array data and attempt to find patterns in which genes are expressed with certain medical conditions, such as patients with cancer. With this information researches can possibly define what combination of genes that are expressed can lead to an increased risk of certain cancers.

The most common method of evaluating the similarities within a set of data is through the use of a distance measurement, such as Euclidean distance. There are also methods used for the clustering strings, such as Cosine Similarity, which is the most commonly used method for document clustering, see [34]. Other fields where cluster analysis is commonly used in are pattern recognition, image analysis, city-planning, and earthquake studies.

### *1.1.1 Clustering Algorithms*

There are a myriad of different clustering algorithms that have been developed. Each algorithm has its own characteristics, advantages, and drawbacks. Not all algorithms generate useful results for any set of data. Certain algorithms lend themselves to certain types of data. To determine whether running a particular algorithm on a dataset provides a good result, it is important to note that a good clustering algorithm will provide a high quality result with high intra-cluster connectivity and low inter-cluster cluster connectivity. The following sections describe five major groups of clustering and specific examples of various algorithms within each group.

## *Partition-Based Algorithms*

Also called centroid clustering, partition-based clustering algorithms section off the data into groups and defines clusters with an elliptical shape. The most common algorithm in this area is k-means and its variations, such as fuzzy c-means.

K-Means provides a hard clustering, meaning each data point belongs to only one cluster. Fuzzy c-means on the other hand provides a soft clustering and takes a fuzzy value as a parameter. Based on this fuzzy parameter, the final results of the algorithm give each data point's relative connectivity to each cluster. No one data point belongs exclusively to a single cluster, each point has a membership value associated with each cluster.

K-Means is perhaps the simplest algorithm and probably the most commonly used clustering algorithm, which simply assigns data points to the nearest cluster center. This algorithm has a linear convergence rate and is relatively easy to implement. The algorithm works in an iterative fashion, assigning each point to the nearest cluster center. It then calculates the center of the cluster based on the average distance between each data point contained within the cluster. Once the cluster centers are found the algorithms reiterates, assigning points again to the nearest cluster center. The stopping point for the algorithm is either until a predetermined number of iterations have passed or until no data point changes its cluster membership during an iteration.

In its simplicity, k-means carries with it some disadvantages. It is susceptible to noise in the data set as each data point must belong to a cluster, and an outlier can distort the shapes of clusters [34]. Probably the largest disadvantage of k-means is that the user must provide the number clusters to look for within the data set. If the user has little or no knowledge of the data set to be clustered, this poses a problem that will greatly affect the results of the algorithm. One possible solution to this problem is to run the algorithm multiple times and analyze the results either manually, or using a cluster validation technique to gauge the optimal number of clusters within the data set. Another solution to this problem researchers have come up with is to combine k-means with a hierarchical algorithm. You first run the hierarchical algorithm and from that deduce the optimal number of clusters as input for the k-means algorithm [3].

One modification to k-means has been in the speed of the convergence rate. During the first iteration, the number of cluster centroids defined by the user are placed randomly within the bounds of the dataset prior to the data points being assigned to them. An improvement to the placement of the centroids can provide a faster convergence rate for the algorithm, see [1].

K-Means does not handle high-dimensional data well [34] and as the number of attributes in the data increases, K-Means tends to give increasingly poor results. The main advantages of this algorithm are its simplicity and fast convergence rate.

Fuzzy C-Means follows the same basic technique as k-means. Each data point is a member of a single cluster in k-means, however, in fuzzy c-means each data point can be associated with multiple clusters, with a varying degree of membership.

Fuzzy C-Means has similar disadvantages as K-Means, in that, the user must provide the number of clusters to the algorithm and the initial placement of the cluster centroids must be decided. Likewise, it also has a fast convergence rate.

## *Hierarchical Algorithms*

Hierarchical clustering outputs a dendrogram, a tree-like structure, and allows a user to see how the data set will be clustered at multiple levels with a varying number of clusters. At the top level is one cluster with every datapoint within the set contained inside that cluster. At the lowest level, the leaves of the dendrogram, every datapoint is its own cluster, so the number of clusters at the lowest level is equal to the number of datapoints contained in the data set.

Hierarchical methods fall into two categories, divisive and agglomerative [34]. The divisive methods start with the entire data set as one cluster and iteratively breaks apart the clusters until each data point is an individual cluster. At each iteration, or level, the number of clusters is increased by one. Agglomerative works just the opposite, each data point starts as its own cluster and then the clusters are merged at each level until all the data points are contained within a single cluster. At each level for agglomerative, the number of clusters is decreased by one. In both methods, the number of levels in the dendrogram is equal to the number of datapoints in the dataset.

To decide which cluster should be split or merged, hierarchical algorithms employ several methods for measuring the intercluster distances. The simplest of these methods [12] is single-link, where the two closest datapoints within two clusters are compared. The clusters who are closest are then merged together for agglomerative algorithms. Another commonly employed linkage method is complete link, where unlike in the single-link method, the two datapoints who are furthest from each other in two respective clusters are compared. Again, the clusters who are determined to be closest by this method are merged for agglomerative algorithms. The method average link calculates the average distance between all pairwise points within two clusters. The centroid method calculates the location of the center of each cluster and compares clusters by measuring the distances between clusters by comparing their centers. What calculations are used for these methods can be found in section 1.1.2 and for more information on each of these linkage methods as well as examples of a few more, see [12].

Hierarchical is a greedy algorithm, once it decides what will be split or merged the decision cannot be refined at a later time. It has a polynomial time complexity and does not handle high dimensional data sets well. A variation of this algorithm, named hierarchical k-means, utilizes the tree structure of hierarchical clustering to estimate a reasonable value for the number of clusters in k-means for a particular data set [3].

### *Density-Based Algorithms*

Density-based clustering focuses on highly concentrated areas of data while being minimally affected by any outliers in the data set. It deals well with large spatial data sets, is affected very little by noise in the data set, but requires some parameters. Density-based algorithms can create arbitrarily shaped clusters around dense regions of the data while leaving low-density areas of the data unclustered to omit any outliers. This is unlike clustering algorithms in other areas which would make sure every piece of data was contained within a cluster. One of density-based algorithms' advantages is that it can actually recognize these outliers and make sure they do not cause contorted clusters to appear. This is however, all based upon the input parameters that are chosen.

The first density-based clustering algorithm, Density-Based Spatial Clustering of Applications with Noise, or DBSCAN [11], deals well with high dimensional data, however, a subsequent algorithm in the density-based area, DENCLUE (DENsity-based CLUStEring) [17], has both an increased efficiency and deals well with high dimensional data. The one problem that arises with DENCLUE is its two input parameters. The quality of the resulting clustering is highly dependent on the choice of these two parameters [22]. The first parameter determines the influence a data point has in its neighborhood and the second parameter is a density-attractor which defines the individual clusters [22]. Reducing the number of density-attractors can improve the performance of the algorithm [22]. DBSCAN also requires two input parameters, one of which defines how many points must be near each other in order for a cluster to be formed, which helps to determine which points are outliers, and the other parameter which defines how close the data points must be to each other, in order to be considered a cluster.

Another density algorithm, BRIDGE [20], attempts to merge the k-means algorithm and DBSCAN. In this algorithm, an iteration of k-means is run first, so the user must provide the number of clusters $k$. Through this use of k-means, the algorithm estimates the density threshold. So there is a tradeoff, while the computation speed of k-means is utilized and the advantage of DBSCAN to find arbitrary clusters and to be robust to noise, the user must still have some knowledge of the number of clusters in the dataset.

## *Grid-Based Algorithms*

Data mining applications place certain requirements on clustering algorithms that drive the area of grid-based clustering algorithms. These requirements include working well with datasets with a large number of attributes, easy interpretation of results, and not basing the clustering on a certain model or distribution [29]. Grid-based clustering algorithms fill this gap and are useful for clustering large multi-dimensional data sets, and as in density-based clustering algorithms, they regard regions of greater density than the surrounding areas as clusters [12]. Grid-based clustering algorithms have a reduced computational complexity in comparsion with other clustering algorithms for very large datasets.

Grid-based clustering algorithms generally follow five steps [12]. They first create a grid structure within the bounds of the dataset and separate equal spaced cells within the grid, whose size is predefined. The density of datapoints within each cell is then calculated and the cells are sorted by their evaluated densities. Cluster centers are identified and finally, neighboring cells are traversed.

One specific algorithm in this area is STING [33]. The dataset is divided into a grid of rectangles with a hierarchical structure. The top layer of this structure is a single cell and in each descending layer, each cell is divided into 4 more cells, called child cells. Each one of these cells is one quadrant of the parent cell. Whether a particular cell will be split or not is decided by the density of that cell. To make a query, the algorithm starts from the root cell and works it way down each descending layer to search for possible results. In this way, the algorithm has quick query searches, computationally. See [33] for more information about STING.

Another grid-based algorithm, WaveCluser, clusters spatial data based on wavelet transforms [13]. It carries with it many advantages, such as, not being affected by noise, being able to detect arbitrarily shaped clusters and a fast computation speed. Its computation time is linear over the number of objects to be processed in the dataset [12]. For more information on WaveCluser, see [13].

#### *Model-Based Algorithms*

Model-based clustering algorithms cluster data based on probability models. In model-based clustering it is assumed that the datapoints are generated by a mixture of probability distributions in which each component represents a different cluster [12]. With model-based algorithms we are stuck with selecting the right model and probability framework where as with the previous algorithm types mentioned, we look for the best method and find the optimal number of clusters.

Expectation Maximization (EM) is an optimization method for estimating some unknown parameters given a dataset [8]. EM alternates between estimating the unknowns and finding hidden variables. For a detailed example on EM, see [8].

### *1.1.2 Distance Measurements*

Distance measures are what is used to decide the similarity between not only a pair of objects, but also an object and a cluster, or a pair of clusters, see section 1.1.1 on how distance measurements are used in this way.

Distance measures have three basic criteria that need to be followed [31].

1. $D(x, y) \geq 0$

2. $D(x, y) = D(y, x)$

3. As $D(x, y)$ decreases, so must the value for the distance decrease in a correlating fashion.

Where, $D$ is the distance measure, and $x$ and $y$ are two different datapoints within the dataset.

The following subsections describe various commonly used distance measures.

*Euclidean*

Euclidean distance, see equation 1.2, is a special case of Minkowski distance, see equation 1.1, for $n = 2$, and is the most commonly used distance measurement [34, 12]. The distance $D$ between two points $i$ and $j$ is calculated as follows [34],

$$D_{ij} = \left( \sum_{l=1}^{d} |x_{il} - x_{jl}|^n \right)^{\frac{1}{n}} \tag{1.1}$$

$$D_{ij} = \left( \sum_{l=1}^{d} |x_{il} - x_{jl}|^2 \right)^{\frac{1}{2}} \tag{1.2}$$

For Euclidean distance, the square of the sum of the square roots of the corresponding attributes added together for two points gives the distance between those two points. The variables $i$ and $j$ have number of attributes from $1..d$ describing each of them. Each attribute for each point is paired, added together, and then the square root is taken. These values are then summed together and the square of this summation is taken to define the distance between the two points.

### *City-Block*

City-block distance, also known as Manhattan distance, is a special case of Minkowski, see equation 1.1, where $n = 1$ [34]. City-Block distance is the distance between two points measured along orthogonal axes. Equation 1.3 describes this special case of Minkowski.

$$D_{ij} = \sum_{l=1}^{d} |x_{il} - x_{jl}| \tag{1.3}$$

Where the distance $D$ of two points $i$ and $j$ with a number of attributes $d$ is calculated as the sum of the absolute values of the difference of the two points.

### *Cosine Similarity*

The most commonly used measure for document clustering [34], cosine similarity, relies on the attribute vectors of the documents to be compared. The similarity result given by this distance measure is between the value of -1 and 1, where -1 represents complete dissimilarity and a value of 1 represents an exact match.

$$S_{ij} = \cos\alpha = \frac{x_i^T x_j}{\|x_i\| \|x_j\|} \tag{1.4}$$

Equation 1.4 defines cosine similarity, where $x_i$ and $x_j$ are two transactions represented by a $d$-dimensional bit vector [12]. $S_{ij}$ is the similarity between these two objects.

*Pearson Correlation*

The Pearson correlation is a distance measure derived from the correlation coefficient and is used to evaluate the similarity between two variables. It is the most widely used distance measure for analyzing gene expression data, see [34].

$$D_{ij} = (1 - r_{ij})/2, \ \textit{where} \ r_{ij} = \frac{\sum_{l=1}^{d}(x_{il} - \overline{x_i})(x_{jl} - \overline{x_j})}{\sqrt{\sum_{l=1}^{d}(x_{il} - \overline{x_i})^2 \sum_{l=1}^{d}(x_{jl} - \overline{x_j})^2}} \tag{1.5}$$

Where, $x_i$ and $x_j$ are two data objects with an number of attributes equal to $d$.

*Mahalanobis Distance*

Mahalanobis distance lends itself to evaluating the similarity between objects in an ellipsoidal manner. Objects that are around the center of mass of a cluster will tend to have higher similarity values.

$$D_{ij} = (x_i - x_j)^T S^{-1} (x_i - x_j) \tag{1.6}$$

According to [34], a disadvantage of Mahalanobis is that it may be computationally slow.

### *1.1.3 Cluster Validation*

In order to figure out how good of a result a clustering algorithm comes up with, there needs to be some method to evaluate the result. Data clustering is unsupervised and it is reasonable to ask, especially for clustering algorithms which ask for the number of clusters as part of their input criteria, exactly how many clusters are in the data set. Several validation techniques have been developed that take the result of a clustering algorithm, as well as the corresponding data set, and come up with an index value that describes the goodness of the clustering algorithm's result. The index values generally range between 0 and 1, and depending on the algorithm, when the value approaches one these extremes, it indicates high cluster quality.

In general, with cluster validation techniques, the same clustering algorithm is run multiple times on the same set of data, but with a different set of values for the input criteria for the clustering algorithm in order to generate different sets of results. The validation technique would then be run on each set of results from the clustering algorithm, generating an index value for each. From these resulting index values it can be evaluated what input generated the best results on that particular data set.

It should be noted that there is currently no validation technique that can be used to evaluate the results of any given algorithm. Each technique specializes in a certain type of clustering algorithm, for instance, Hubert's $\Gamma$ Statistic, see section 2.2.1, will only give a reliable index value for partition-based clustering. Current validation techniques rely on compact clusters, however in new fields such as bioinformatics, validity checking is needed for sparse and arbitrarily shaped clusters [23]. There is a need to develop a cluster validation method that takes into account the intra-cluster quality, the inter-cluster separation and the geometry of the clusters using more than just a single point of reference, either multiple points or even a multi-dimensional curve [24]. Also, there are methods specifically for validating the results of fuzzy clustering algorithms [12, 35, 7].

There are three main areas of cluster validation techniques, external criteria, internal criteria, and relative criteria. Each of these are explained in detail in the following sections.

### *External Criteria*

The validation methods in the area of external criteria measure the validity of clustering results by comparing the results with a pre-specified structure. There are two different approaches to external criteria [12].

1. Comparing the resulting clustering structure $C$ to an independent partition of the data $P$, which was built to people's intuition about the clustering structure of the data set.

2. Comparing the proximity matrix $Q$ to the partition $P$.

Some common indices that use the first approach are the Rand statistic, Jaccard coefficient, and the Folkes and Mallows index [12]. For more information on these different indices, see [12].

For the second approach, Hubert's $\Gamma$ Statistic can be computed using the proximity matrix $Q$ and the partition $P$ [12]. For a detailed description on Hubert's $\Gamma$ Statistic see section 2.2.1 on page 32.

### *Internal Criteria*

Instead of comparing the results of a clustering algorithm with a pre-specified structure, as in external criteria, the internal criteria uses the structure and values of the dataset itself. Internal criteria are generally used for hierarchical clustering algorithms and single clustering schemes [22]. Internal and external criteria are both statistical methods and have a significant computational cost. They also rely on a clustering result matching with a prespecified scheme [21]. In the next section, relative criteria uses a much different method of validating results.

### *Relative Criteria*

The procedure for validation techniques in this area is to take the set of parameters of a clustering algorithm and analyze the results from multiple runnings of the algorithm given different parameters. The problem can be divided into two cases given whether the number of clusters is a parameter of the clustering algorithm or not [12].

1. If the number of clusters is not part of the parameters, to choose the optimal values for these parameters, the clustering algorithm is run with a large range of values for the input parameters until a range is identified in which the number of clusters stays constant. The parameter values that correspond to the middle of this range are then chosen as optimal parameters.

2. In the case that the number of clusters is not in the set of parameters of the clustering algorithm, the algorithm is run with minimum and maximum values for the number of clusters defined beforehand by the user. For each of the values of the number of clusters, the algorithm is run numerous times with a range of values for the rest of the parameters. The results are plotted with respect to the number of clusters and the best validity index is chosen [22].

For more information on relative criteria, see [12] and [22].

## *1.2 Significance*

Cluster analysis is a complex process. Not only does it require choosing the most appropriate algorithm for use with the dataset, but also the clustering result of the chosen algorithm is heavily influenced by the choice of input parameters and similarity metric. Many factors work together to tune the clustering results. Therefore, the validation of clustering results are significantly important and are used to analyze those clustering results, measure the goodnes of the results, and provide users a level of confidence in the results. However, based on the literature review, there is not a comprehensive survey on cluster validation published thus far. In this thesis, a comprehensive survey on cluster validity indices is conducted.

Cluster validation is a cohesive part along with clustering in the data analysis process, as shown in figure 1.1. It is desired to have a platform aid users in their choice of clustering algorithms and notify users of the quality of clustering results with cluster validation. Based on review, there exist a few platforms for cluster analysis [10, 14]. However, they are either limited in their scope or are far from user-friendly. One such platform, named Cluster 3.0 [10], allows a user to cluster micro array data for use with bioinformatics. Cluster 3.0 offers only a couple different choices of clustering algorithms, k-means and hierarchical, and the results generated by the platform take some time and expert knowledge in the field in order to decipher. Another such platform, RapidMiner [14], offers a variety of clustering. RapidMiner, however, requires extensive knowledge in order to be used. Users need to be trained in the program and require programming skills in order to use it. Furthermore, neither of them combine clustering algorithms with validation methods to verify the results.

No platform exists that offers a simple, user-friendly environment for cluster analysis. In this thesis, a platform is developed to fill that gap with an implementation of commonly used clustering algorithms and validation methods.

Based on literature review, there is little work done on compiling together several different clustering and validation methods for evaluating the clustering matrix. Validation techniques are an important tool for users in deciding which algorithm best suits their data. The techniques relieve work that would otherwise have to be done by the user in evaluating the results of each algorithm. The platform provides an analysis and summary of all the validation techniques and clustering algorithms run by the user on a dataset and from this information the user will be able to determine which algorithm will be most useful for their purposes with a minimal amount of knowledge on their own part, the bulk of the work having been done by the platform itself using these validation techniques.

This platform is extendable. New clustering algorithms and validation methods may easily be added and as long as the clustering algorithms adhere to the standard form, they may be used on any validation methods implemented in the platform. Along with extendability, the platform offers visualization of multi-dimensional data using principal component analysis and estimation of clusters through use of hierarchical clustering's dendrogram.

Beyond just the platform itself, this thesis covers the critical problem in data clustering research, the estimation of the number of clusters. Through use of the agglomerative hierarchical clustering algorithm the number of clusters in a dataset may be derived. This gives the user more information about the dataset and this information may also be used to obtain more accurate results from clustering algorithms that require the number of clusters as an input parameter.

In summary, this thesis provides a comprehensive validation methods review that is not available in current publications, develops an extendable platform that integrates clustering and validation while the existing clustering platforms offer clustering methods only without results evaluation. In addition, this thesis proposes an approach to estimate the number of clusters by analyzing hierarchical clustering, in which the impact of various merge criteria on the data distributions is studied.

## *1.3 Objective*

There are no perfect clustering techniques that will provide useful results for any particular set of data. The current clustering algorithms have shortcomings that need to be addressed. According to [34], a recent survey on clustering algorithms, some of the most important aspects of a novel clustering approach not shared amongst current approaches include being able to detect possible outliers, predict the number of clusters, and provide useful and simplified data visualization. The objective of this thesis is to solve these problems and create a general platform that offers several commonly used methods of clustering as well as validation techniques to evaluate the results generated by the clustering algorithms. Combining these techniques with cluster validation allows users to find the best clustering algorithm and input parameters to use for their set of data.

Users will be able upload their dataset, run multiple clustering algorithms on their dataset with varying input parameters and different distance measures to choose from.

The user may save the results of the algorithms and run several different validation methods on the results. The indices for the validation methods are given as well as an analysis option to review all results from all clustering algorithms and validation methods used on the dataset. The analysis also gives recommendations on which algorithm with which set of input parameters gave the best results according to the validation methods that were run.

# 2. METHODOLOGY

This chapter encompasses the clustering algorithms and validation methods that were implemented into the clustering platform. Each algorithm that was implemented into the platform is explained in detail and each were chosen based on their areas in order to obtain a wide array of techniques for clustering and validation. Partition-based, hierarchical, density-based, and model-based clustering algorithms were all chosen, as well as validation techniques for external, internal, and relative criteria.

## *2.1 Clustering Algorithms*

The clustering algorithms chosen to be implemented in the platform came from the different areas described in the literature review in the first chapter of this thesis. K-Means and fuzzy-c means are partition based algorithms very similar to each other in procedure. Both are simple and computationally fast algorithms that define centroid clusters. Agglomerative hierarchical is the common algorithm of hierarchical clustering. It is slow computationally, but provides a whole picture of the entire clustering. DBSCAN is a density-based clustering algorithm that is still computationally fast, but is not as affected by noise as the algorithms in the previous areas. Expectation Maximization is a model-based algorithm and is very different from the others in this platform, in that it is developed based off of probability models.

Following are detailed descriptions of each of the above clustering algorithms implemented into the platform.

### 2.1.1 K-Means

K-Means, already described in general during the literature review in section 1.1.1, was implemented into this platform due to how commonly it is used. It defines the idea of partition-based clustering. As the algorithm has already been described in general, this section will get into the specifics on how this algorithms works.

```
$(C_1, C_2, ..., C_k)$ = Initial cluster centers
while Datapoints change cluster membership do
    $d_{ij}$ = Distance between datapoint $i$ and cluster center $j$
    Assign datapoint $i$ to the cluster $j$ with the minimum distance
    Recalculate cluster centers
end
```

**Algorithm** 1: K-Means Algorithm.

First, the algorithm begins with a predefined number of clusters $k$. This must be defined by the user. In some instances the number of iterations for the algorithm to run through can also be defined by the user. Next, k-means partitions the dataset into $k$ sets. This is done by randomly assigning cluster centers equal to $k$ throughout the bounds of the dataset. The algorithm then assigns the points closest to each cluster center to that cluster. This is done by using a predefined distance measurement, usually Euclidean distance [12]. Now that each point is assigned to a cluster the cluster center is recalculated to the new center of the cluster that has been created. The cluster centers move locations and a new iteration of the algorithm begins, checking to see if any points should switch clusters now that the cluster centers have moved. This is repeated until either no point changes cluster membership or the predefined number of iterations has been met.

K-Means is a simple, fast algorithm to implement. The time complexity is $O(n)$. The algorithm is subject to noise and outliers and does not work well with high-dimensional data.

### *2.1.2 Fuzzy-C Means*

Fuzzy-C Means (FCM) is one of the most popular fuzzy clustering algorithms [34]. FCM's execution is similar to that of k-means, see section 2.1.1, however each point may belong to two or more clusters. As k-means, FCM requires a few input parameters, $C$ the number of clusters, $m$ the fuzzifier, and $\epsilon$ the minimum improvement.

The following is the procedure for FCM using Euclidean for the distance function [25].

```
Initialize matrix $U^0 = [u_{ij}]$
repeat
    At $k$-step: calculate the centers vectors $C^k = [c_j]$ with $U^k$
        $c_j = \frac{\sum_{i=1}^{N} u_{ij}^m * x_i}{\sum_{i=1}^{N} u_{ij}^m}$    (2.1)
    Update $U^k, U^{k+1}$
        $u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{||x_i - c_j||}{||x_i - c_k||}\right)^{\frac{2}{m-1}}}$    (2.2)
until $||U^{k+1} - U^k|| < \varepsilon$
```

**Algorithm 2**: Fuzzy c-means algorithm.

Where $m$ is required to be a real number greater than 1, $u_{ij}$ is the degree of membership of datapoint $x_i$ in the cluster $j$, $x_i$ is the $i$th attribute of $d$-dimensional datapoint and $c_j$ is the $d$-dimension of the center of the cluster, and $k$ keeps track of the current iteration.

The results of the algorithm gives the membership values of each datapoint to each cluster. A datapoint will generally have a high membership value to one cluster and lower membership values to the rest of the clusters.

FCM suffers from similar problems as its counterpart, k-means [34]. The algorithm is susceptible to noise and requires careful consideration of the input variables, particularly the number of clusters. The one strong benefit it has over k-means is the fuzziness, it is not hard clustering, like K-Means.

### *2.1.3 Agglomerative Hierarchical*

As stated in the literature review for hierarchical clustering algorithms in section 1.1.1, the end result of the algorithm is a dendrogram. This dendrogram displays a complete clustering, as clusters are merged one point at a time at each level. The starting point for the algorithm is each object within the dataset considered to be its own cluster. The general agglomerative hierarchical clustering method can be summarized by the following procedure [34].

```
Start with $N$ singleton clusters
Calculate the proximity matrix for the $N$ clusters
repeat
    Search the minimal distance
        $D(C_i, C_j) = \min_{1 \le m,l \le N, m \ne l} D(C_m, C_l)$        (2.3)
    where $D$ is the distance between two clusters
    Combine cluster $C_i$ and $C_j$ to form a new cluster
until all objects are in the same cluster
```

**Algorithm 3**: Agglomerative hierarchical algorithm.

For the distance function, the most commonly used is the single linkage method [12]. On the first step of the agglomerative hierarchical algorithm, this method grabs the smallest distance value from the proximity matrix and those two clusters get combined. To update the proximity matrix to the next level, instead of recalculating the values, a new row and column are created in the matrix, representing the newly created cluster that contains two objects. The values in the row and column for this new cluster are generated by looking at the rows and columns of the objects that are contained within the cluster. The smaller of the two values is placed within cluster's row and column. The rows and columns for the two objects are then removed from the proximity matrix.

Once the algorithm has iterated through $n$ times, where $n$ is the number of objects in the dataset, the dendrogram has been completed and the algorithm has finished. The top of the dendrogram contains one cluster holding all the objects in the dataset. The results of the algorithm are usually displayed as the entire dendrogram and the distance between the merges at each level of the dendrogram.

Single linkage is not the only method used with hierarchical clustering. Complete, average, centroid, and median linkages are all used interpret the proximity of one cluster to another in agglomerative hierarchical. The main disadvantage of the nearest neighbor single linkage method is its inability to differentiate between two clusters that are in close proximity to one another. The furthest neighbor complete linkage method provides somewhat better results in that regard. The rest of the linkages offer better, but varying results when clusters are in a very close proximity to one another. Median linkage uses Euclidean distance as defined in equation 2.4, where the clusters $i$ and $j$ are defined recursively as half the distance between the centers of the two clusters that were merged to create them. Centroid linkage also uses the Euclidean distance function in equation 2.4, where $C_i$ (and $C_j$) is defined as in equation 2.5, where $n$ is the number of data points in the cluster. Average linkage calculates the average distance between all pairs of points in two clusters, as in equation 2.6. For further information about any of these linkages, see [12]. For results regarding the performance of these linkages on several datasets, see chapter 3.

$$D(C_i, C_j) = |C_i - C_j|_2 \tag{2.4}$$

$$\bar{C}_i = \frac{1}{n_i} \sum_{k=1}^{n} x_{ik} \tag{2.5}$$

$$D(i,j) = \frac{1}{n_i n_j} \sum_{k=1}^{n_i} \sum_{l=1}^{n_j} dist(C_{ik}, C_{jl}) \tag{2.6}$$

While agglomerative hierarchical clustering gives nice results for the entire clustering, the time complexity is $O(N^2)$, which makes the algorithm computationally expensive for large-scale datasets [34]. The algorithm is also subject to noise and outliers.

### *2.1.4 Density-Based Spatial Clustering of Applications With Noise*

Density-Based Spatial Clustering of Applications with Noise, or DBSCAN, is a density-based algorithm briefly introduced in section 1.1.1. This section goes into further detail, supplying definitions, pseudo-code, and the procedure for the algorithm.

DBSCAN requires two input parameters, Eps, and MinPts. Eps holds the radius of a particular neighborhood, while MinPts holds the minimum neighborhood size. These two parameters define [11]:

$$N_{Eps}(p) = \{q \in D \mid dist(p, q) \leq Eps\} \tag{2.7}$$

That is, a point $q$ in the dataset $D$ is in the neighborhood of point $p$ if the distance between $p$ and $q$ is within the range of the Eps parameter.

There are also a few other important definitions that go along with DBSCAN, the first being directly density-reachable. A point $p$ is directly density-reachable from a point $q$ with respect to Eps and MinPts if, $p$ belongs to $N_{Eps}(q)$ and $q$ fulfills the core point condition [11]:

$$|N_{Eps}(q)| \geq MinPts \tag{2.8}$$

Also, $p$ is considered to be density-reachable from $q$ with respect to Eps and MinPts if there is a chain of points $p_1, ..., p_n$, where $p_1 = q$, and $p_n = p$ such that $p_{i+1}$ is directly density-reachable from $p_i$. For the next definition density-connected, $p$ is considered density-connected to $q$ with respect to Eps and MinPts if there is a point $o$ such that both $p$ and $q$ are density-reachable from $o$. Finally, a cluster $C$ is formed when, for all $p$ and $q$, if $p$ is in $C$ and $q$ is density-reachable from $p$, then $q$ is also in $C$ as well as for all $p$ and $q$ in $C$, $p$ is density-connected to $q$ [11].

There are also a couple more important insights to help understand the process of this algorithm. First, if $p$ is a core point, and $O$ is the set of points density-reachable from $p$, then $O$ is a cluster. Second, if $C$ is a cluster and $p$ is a core point of $C$, then $C$ equals the set of density-reachable points from $p$. The implication of these two ideas is that finding the density reachable points of an arbitrary point generates a cluster and a cluster is uniquely determined by any of its core points [11].

Here is the process the algorithm goes through [11]:

```
Arbitrarily select a point $p$
while Not all datapoints have been visited do
    Retrieve all points density-reachable from $p$ with respect to Eps and MinPts
    if $p$ is a core point then a cluster is formed
    if $p$ is a border point then
        no points are density-reachable from $p$
        DBSCAN visits the next point in the dataset
    end
end
```
**Algorithm** 4: DBSCAN algorithm.

For each point, DBSCAN determines the Eps-environment and checks whether it contains more than MinPts data points. As a general guideline, the best values to choose for the Eps and MinPts are corresponding to the thinnest cluster in the dataset. Of course, this is difficult to do without any pre-knowledge of the dataset.

DBSCAN has a time complexity of $O(n * \log n)$. The algorithm is resistant to noise and can handle clusters of different shapes and sizes. However, it does not work well with datasets that have greatly varying densities or a large number of dimensions.

### *2.1.5 Expectation Maximization*

Expectation maximization (EM) is an iterative method that takes in some unknown parameters $\Theta$, given a dataset $U$. There are some missing variables $J$ within the dataset that need to found and removed. The idea is to maximize the posterior probability of the parameters $\Theta$ given the data $U$, marginalizing over $J$ [8].

$$\Theta^* = \max_{\Theta} \log P(U, \Theta) = \max_{\Theta} \log \sum_{J \epsilon \tau^n} P(U, J, \Theta) \tag{2.9}$$

EM iterates over estimating the unknowns $\Theta$ and the missing variables $J$ [8]. The E-step can be interpreted as constructing a local lower-bound to the posterior distribution, whereas the M-step optimizes the bound, thereby improving the estimate for the missing variables [8].

The procedure for the algorithm is as follows:

```
Initialize the distribution parameters
while the estimations of the distribution parameters are not convergent do
    Expectation (E): Computes an expectation of the likelihood by including the
    latent variables as if they were observed
    Maximization (M): Re-estimates the parameters by maximizing the expected
    likelihood found on the E step
end
```

**Algorithm 5**: Expectation maximization algorithm.

An advantage of model-based clustering is that it provides an estimated probability that a data object belongs to a cluster [6]. So, in the case that an object has a high correlation to two different clusters, it will be made apparent by this algorithm. The disadvantage is that the algorithm assumes that the data follows a particular model such as a Gaussian [6].

## 2.2 Validation Methods

The validation methods chosen to be in the platform were based upon their relevancy to the clustering algorithms being implemented. Each of these validation methods work well for the clustering algorithms that were chosen. For deatiled information on other validity methods not discussed in this thesis, see [21, 15, 5, 16, 26, 27].

The following sections describe the various validation methods in detail.

### 2.2.1 Hubert's Γ Statistic

Hubert's Γ Statistic is a validation method that measures the compactness of clusters. A high value of Γ indicates that there exist compact clusters. Equation 2.10 defines Hubert's Γ Statistic where $n$ is the number of data points in the dataset, and $M = \frac{n(n-1)}{2}$.

$$\Gamma = \frac{1}{M} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} Y_{ij} \tag{2.10}$$

Hubert's Γ Statistic is a good validation method for evaluating the results of partition-based clustering. As stated earlier in section 1.1.1, partition-based algorithms require the number of clusters as part of the input criteria. So, Hubert's Γ Statistic can be used to evaluate the results of multiple runnings of a partition-based clustering algorithm with a different number of clusters for each running to suggest what the appropriate number of clusters should be for that particular dataset.

### *2.2.2 Dunn's Index*

The Dunn's Index reveals clusters with low intra-cluster distances and high inter-cluster distances. Dunn's Index may be computed using the following,

$$D(U) = \min_{1 \leq i \leq c} \{ \min_{1 \leq j \leq c, j \neq i} \{ \frac{D(X_i, X_j)}{\max_{1 \leq k \leq c} \{\Delta(X_k)\}} \} \} \tag{2.11}$$

where $U$ is the dataset, $c$ is the number of clusters, $D(X_i, X_j)$ is the distance between clusters $C_i$ and $C_j$, and $\Delta(X_k)$ is the diameter of the cluster $C_k$. Large values for $D$ indicate that good clusters have been found. Based on this, the number of clusters chosen for a particular algorithm that generates the largest value for $D$ is the optimal number of clusters [12].

The Dunn's Index comes with its disadvantages, however. The algorithm is both sensitive to noise and running it is very time-consuming.

### *2.2.3 Silhouette Index*

Using equation 2.12, the Silhouette index [30] generates a Silhouette width for each cluster [4].

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.12}$$

The value of $a(i)$ is the average dissimilarity between the $i$th-object and all other objects contained within the same cluster, while $b(i)$ is the minimum average dissimilarity between the $i$th-object and all objects in the next closest cluster. Equation 2.12 returns a value for $s$ between $-1$ and $1$. If the value of $s$ approaches 1, this indicates a good clustering and that the number of clusters found by the algorithm is accurate for that particular data set. When the value of $s$ is near to 0, this indicates that objects contained within one cluster are an equal distance away from another cluster and could be contained within either cluster. If the value of $s$ approaches $-1$, this is an indication of a bad result, that the data objects have been misclassified and should be contained within a different cluster from the one they are currently in. Given several Silhouette indices for a particular algorithm run on the same dataset, but with different values for the input criteria, we can find which input values give the most accurate results. For more information on the Silhouette index, see [4, 28].

### *2.2.4 Davies-Bouldin Index*

Similar to that of the Dunn's Index, see 2.2.2, the Davies-Bouldin Index looks for clusters with high intra-connectivity and low inter-connectivity. Equation 2.13 defines the Davies-Bouldin Index, where $c$ is the number of clusters, $\Delta$ is the average distance of all data objects from the cluster to the cluster center, and $D(X_i, X_j)$ is the distance between cluster centers.

$$DB(U) = \frac{1}{c}\sum_{i=1}^{c}\max_{i\neq j}\{\frac{\Delta(X_i)+\Delta(X_j)}{D(X_i,X_j)}\} \tag{2.13}$$

Unlike in validation techniques such as Silhouette and Dunn's Index, given equation 2.13, a small value for Davies-Bouldin means the clustering result was good. A small value indicates compact clusters and large distances between cluster centers.

## 2.3 Principal Component Analysis

The goal of principal component analysis (PCA) is to reduce high-dimensional data into fewer dimensions, while still retaining as much important information about the data as possible [18]. New variables called principal components (PC) are created such that they contain most of the variation that was present in the original variables [12].

The first step in PCA is to subtract the mean from each of the data dimensions. The result of this is a dataset whose mean is zero. The next step is to calculate the covariance matrix. The covariance matrix is defined as [32],

$$C^{n \times n} = (c_{i,j}, c_{i,j} = cov(Dim_i, Dim_j)). \tag{2.14}$$

$C^{n \times n}$ is a matrix with $n$ rows and $n$ columns, and $Dim_x$ is the $x$th dimension. Next, calculate the eigenvalues and eigenvectors for the covariance matrix. The eigenvectors of the covariance matrix provide lines that characterize the data. The eigen vectors with the highest eigenvalues are the principal components of the dataset. Any eigenvectors thrown out at this point reduces the dimensionality of the data. This gives the feature vector, defined as [32],

$$FeatureVector = (eig_1, eig_2, ..., eig_n). \tag{2.15}$$

The smaller the eigenvalue, the less pertinent information is lost by throwing it out. To produce the final dataset $D$ we have [32],

$$D = FeatureVector' \times MeanData' \tag{2.16}$$

$D$ contains the final dataset with the datapoints in columns and the dimensions along rows. The *MeanData* is the adjusted dataset by subtracting the mean from each data dimension as described earlier. This dataset $D$ contains the reduced principal components with reduced dimensions based on how many eigenvectors were thrown out.

A few experiments have been done using PCA to reduce the number of attributes in the data before clustering. One of these experiments involved k-means, and it was found that reducing the data attributes significantly can provide an increase in the clustering accuracy [9]. Another study in bioinformatics found that reducing the dimensionality most often led to degradation of cluster quality and PCA should only be used before clustering when there is some prior knowledge about the dataset, especially its attributes [19].

# 3. IMPLEMENTATION AND RESULTS

This thesis implements an extendable and user-friendly platform for cluster analysis. The language of choice for the front-end of the platform is Java due to its good support of graphical user interfaces and platform independence, while the clustering algorithms and validation methods themselves are implemented in Matlab. Matlab was chosen for the back-end due to its high performance on heavy computation and visualization. NetBeans is used as the integrated development environment. One license for Matlab is currently installed and may be accessed through the Internet by the Java client. Several commonly used clustering algorithms as well as corresponding validation methods have been implemented to provide an integrated approach to cluster analysis and performance evaluation.

This chapter discusses the structure and utilization of the platform. Following that, results obtained through the clustering platform will be discussed. The system structure section will provide class and sequence diagrams, and explain the topology of the system. The section following that, platform utilization, will provide snapshots of the platform and describe how the platform is used. The final section of this chapter, results, explains the information garnered through the use of the platform and the contributions made to cluster analysis.

## *3.1 System Structure*

This platform follows the Model-View-Controller (MVC) architecture. In MVC, the model corresponds to the data used in the application and the rules that are used to manipulate that data, the view represents the graphical user interface (GUI), and the controller manages communication between the user and the model.

The basic structure in this platform is as follows. The Java client takes input parameters for the selected clustering algorithm and dataset from the user. All pertinent information is sent to a separate Java class. This Java class sends that data to Remote Matlab. Remote Matlab waits on the server for a request from the client. When a request is received, Remote Matlab runs the requested script. All heavy computation and visualization is done within Matlab and the results are saved on the server. When the Matlab has finished its computations, the Java client retrieves all important data and images from the server. These data and images are then displayed to the user.

All clustering information is accumulated as the user runs clustering algorithms on a dataset. This information is used by the validation methods. These validation methods follow the same path as the clustering algorithms, sending information to Remote Matlab and retrieving results and displaying them to the user.

A current session is ended when the user loads a new dataset into the Java client. All information about the previous dataset is removed, however files containing clustering results will still exist on the user's machine.

Remote Matlab is an open source tool generated for interactions between Java and Matlab. It allows remote requests from Java to run Matlab scripts. The tool is singlethreaded in that only one user can be using Matlab at one time for computations. Clients that try to connect to Remote Matlab while it is busy are put into a waiting queue. One client's tasks are fully completed before a new client's request is started. This tool is essential to this platform in that having these computations done in Java can take excessively longer than executing them in Matlab, a program that is perfect for this kind of work. The Java frontend's main purpose is for a user-friendly interface and to replace needing to have Matlab running on the user's machine.

The sequence diagram of the system is shown in figure 3.2 and the class diagram is shown in figure 3.1.

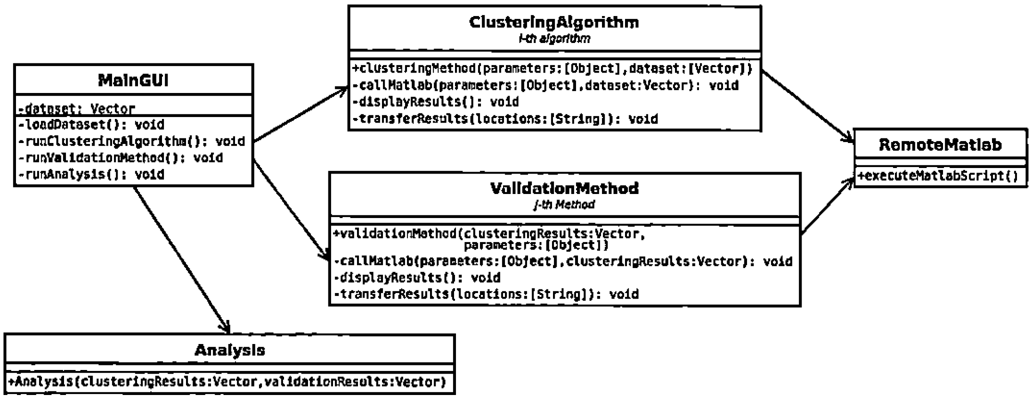

MainGUI
-dataset: Vector
-loadDataset(): void
-runClusteringAlgorithm(): void
-runValidationMethod(): void
-runAnalysis(): void
ClusteringAlgorithm
i-th algorithm
+clusteringMethod(parameters:[Object],dataset:[Vector])
-callMatlab(parameters:[Object],dataset:Vector): void
-displayResults(): void
-transferResults(locations:[String]): void
ValidationMethod
j-th Method
+validationMethod(clusteringResults:Vector, parameters:[Object])
-callMatlab(parameters:[Object],clusteringResults:Vector): void
-displayResults(): void
-transferResults(locations:[String]): void
RemoteMatlab
+executeMatlabScript()
Analysis
+Analysis(clusteringResults:Vector,validationResults:Vector)

*Fig. 3.1:* Class Diagram

The structure for the platform also allows for extendability to new clustering algorithms. A user may easily plug in a new clustering algorithm that is written either in Java or in Matlab.

While the platform has the clustering algorithms and validation methods, it does not have any data preprocessing. The dataset must be ready to go when the user loads it in.

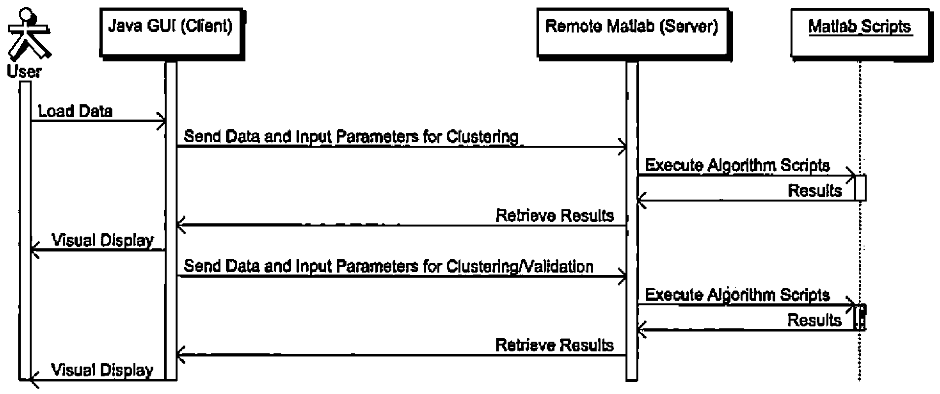
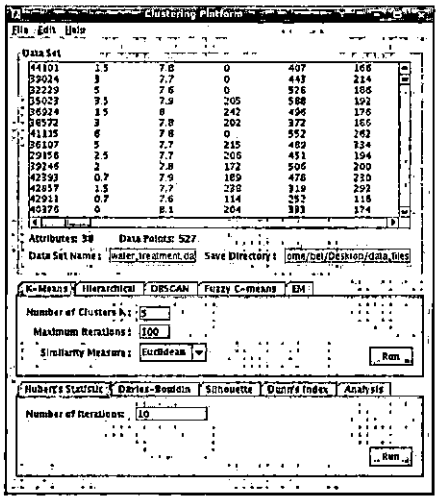

User
Java GUI (Client)
Remote Matlab (Server)
Matlab Scripts
Load Data
Send Data and Input Parameters for Clustering
Execute Algorithm Scripts
Results
Retrieve Results
Visual Display
Send Data and Input Parameters for Clustering/Validation
Execute Algorithm Scripts
Results
Retrieve Results
Visual Display

*Fig. 3.2:* Sequence Diagram

## *3.2 Platform Utilization*

Before doing anything, the first thing that needs to be done with the platform, which is shown in figure 3.3, is to load a fresh dataset with the application is launched. When the dataset is opened it is displayed within a window at the top of the main GUI. Once the dataset is loaded, the user must choose a clustering algorithm to work with. When a clustering algorithm has been selected from the tabs, the user decides on the input parameters and the similarity measure to be used. The algorithm is then executed by clicking the Run button and the dataset along with the similarity measure selection and parameters are sent to the Remote Matlab server where the computations are done. If the dataset contains more than two dimensions, for visual display purposes, PCA, see section 2.3, is used to get the two principal components which will be plotted. Once the execution has finished, the results are retrieved by the Java GUI and the visual results along with the text results are displayed. Next, the user may run more clustering algorithms on the dataset, or may run a validation method. A validation method is chosen from the tabs below the clustering algorithms. and any parameters that are needed are input by the user. Again, Run is selected to execute the validation method selected. The results of all the clustering algorithms that were executed on the dataset during the current session, along with information about the dataset, are sent to the Remote Matlab server where the computations are done. When computation is completed, the results are retrieved by the Java GUI and a histogram comparing the indices generated by the validation method for each clustering algorithm is displayed along with text results describing the indices. The user may run a different validation method at this point or continue to run more clustering algorithms. An analysis GUI is provided to give a summary of all the results generated during the session in tabular form. A new session is started when a new dataset is loaded into the platform.



| | | | | | |
|---|---|---|---|---|---|
| 44101 | 1.5 | 7.8 | 0 | 407 | 166 |
| 39024 | 3 | 7.7 | 0 | 443 | 214 |
| 32229 | 5 | 7.6 | 0 | 526 | 186 |
| 35023 | 3.5 | 7.9 | 205 | 588 | 192 |
| 36924 | 1.5 | 8 | 242 | 496 | 176 |
| 38572 | 3 | 7.8 | 202 | 372 | 186 |
| 41115 | 6 | 7.8 | 0 | 552 | 262 |
| 36107 | 5 | 7.7 | 215 | 482 | 134 |
| 29156 | 2.5 | 7.7 | 206 | 451 | 194 |
| 39246 | 2 | 7.8 | 172 | 506 | 200 |
| 42393 | 0.7 | 7.9 | 189 | 478 | 230 |
| 42857 | 1.5 | 7.7 | 238 | 319 | 292 |
| 42911 | 0.7 | 7.6 | 114 | 252 | 116 |
| 40376 | 0 | 8.1 | 204 | 393 | 174 |

Attributes: 38 Data Points: 527

Data Set Name: water_treatment.da Save Directory: ome/bel/Desktop/data_files

Number of Clusters k: 5

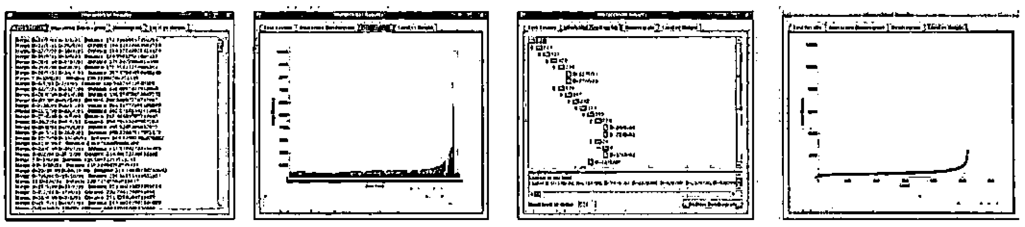
Maximum Iterations: 100

Similarity Measure: Euclidean

Number of Iterations: 10

*Fig. 3.3:* **The Main Interface Displaying Water Treatment Dataset**

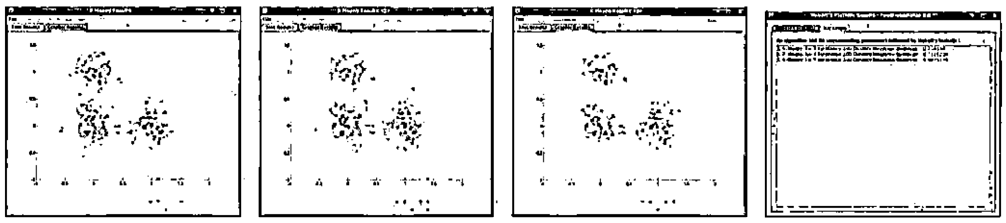
Figure 3.4 displays sample results of running the agglomerative hierarchical algorithm for the water treatment dataset, which has 38 attributes and 327 datapoints. The dataset comes from the daily measures of sensors in an urban waste water treatment plant [2]. The similarity measure used was single-link with Squared Euclidean Distance. Figure 3.4(a) shows us which two objects were merged at each level and the distance between those two objects when they were merged. The next figure, 3.4(b), gives a visual representation of which clusters were combined and jump in distance at each merge level. This dendrogram is the standard output for hierarchical clustering, see section 2.1.3 for more details. The third figure, 3.4(c), is another dendrogram, but in this case the user is able to interact with it. The user may look at a particular level to see the clusters, which points are contained within those clusters, and the merge distance from when the clusters were combined as well as the level at which they were combined. The final figure for hierarchical, 3.4(d), displays the level on one axis and the distance between the two objects being merged at that level. Given the information from these hierarchical results, we can infer how many clusters are in the data. Information on the number of clusters can then be used in algorithms such as k-means and fuzzy-c means. This is discussed in section 3.3.



(a) Text display. (b) Dendrogram. (c) Interactive dendroram. (d) Merge distance vs Level.

*Fig. 3.4:* Agglomerative Hierarchical Results for Water Treatment Data

Beyond just visual clustering results, validation methods may also be used to analyze the clustering results. These validation methods are important for determining cluster quality. In figure 3.5, the k-means clustering algorithm is executed multiple times on a two-dimensional dataset with a differing number of clusters $k$ for each execution. This is an artificial dataset used as a benchmark for the clustering algorithms and validation methods. The optimal number of clusters for this dataset is 3, and we can see that reflected in figure 3.5(d). In this display three values of $\Gamma$ are given, one for each execution of k-means. The first index value on the Hubert's $\Gamma$ Statistic display corresponds to the first figure 3.5(a) and so on down the line. The $\Gamma$ value for $k = 3$ is 0.794146, which is higher than the $\Gamma$ value for the other two results, which means Hubert's has confirmed that out of these results, $k = 3$ is the optimal number of clusters. Users may utilize this feature with their datasets to find the optimal number of clusters given several results with different parameters.



(a) K-Means, k=3. (b) K-Means, k=4. (c) K-Means, k=5. (d) Hubert's $\Gamma$ Statistic.

*Fig. 3.5:* K-Means With Different Numbers of Clusters Analyzed by Hubert's $\Gamma$ Statistic

## *3.3 Results*

Several contributions are made through the use of this platform. Users are able to estimate the number of clusters, multi-dimensional data is able to be visualized through the use of PCA, and both clustering algorithms and validation methods are integrated together, allowing users to check the clustering quality.

In figure 3.6, PCA is being to used to draw the clustering results for k-means in two-dimensions. The dataset being used is the 38 attribute water treatment data. PCA is used only for visualization purposes in the platform, it is not used to modify the dataset prior to clustering.

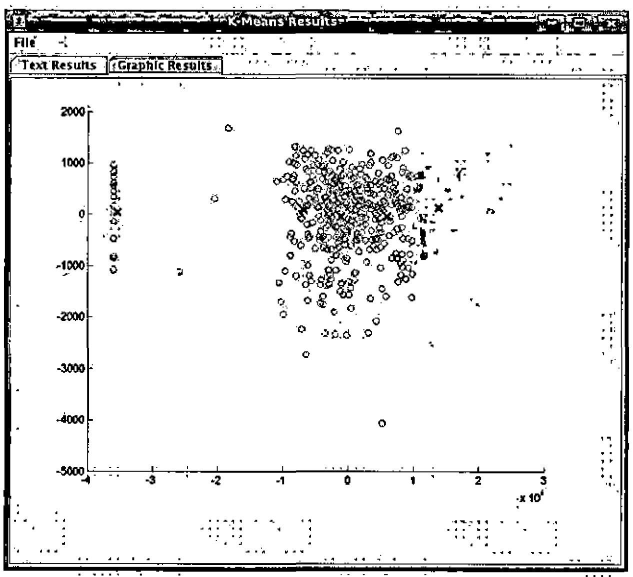


*Fig. 3.6:* Principal Component Analysis Displaying Clusters in Two-Dimensions for 38 Attribute Water Treatment Data

As has been stated, this platform integrates clustering algorithms and validation methods. Using these validation methods, the user may quickly see the cluster quality of each algorithm, helping the user to choose the appropriate algorithm for their dataset as well as the optimal input parameters and similarity measure.

If we take a look at the dendrograms in figure 3.4, we can visually see the jumps between merge distances toward the top levels of the tree structure. This is indicative of how many clusters are in the dataset. Merge distances of similar sizes throughout the entire dendrogram would tell us that the data objects themselves have a high similarity to one another. When we see a jump towards the end, like in these dendrograms, this either shows us two clusters being merged together, or an outlier being merged into a cluster. Be it the former, we can count these clusters and all other clusters at that level to get an estimate of the optimal number of clusters for the dataset. Through use of the interactive dendrogram and the text results, we can see whether it was just an outlier data object being merged, or two clusters. This information about the optimal number of clusters, though just an estimate, is useful as input for the k-means or fuzzy-c means algorithms. If a user has little or no knowledge about the number of clusters in their data, but would like to find results using the k-means method, then this hierarchical to k-means method may be invoked.

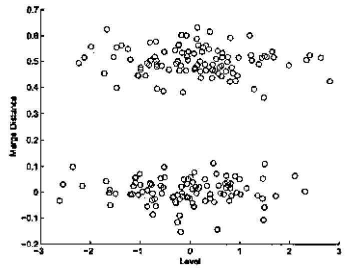
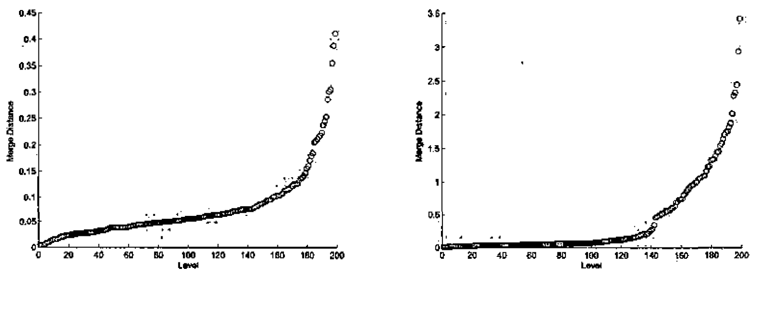


*Fig. 3.7:* Two Cigar-Shaped Clusters

This method does not work in all cases, however, and a different way of calculating cluster centers and thus altering their proximity to one another, must be considered. Given two cigar-shaped clusters whose data is in close proximity to one another, as in figure 3.7, the typical methods of deciding which clusters are closest to each other, such as single and complete linkage, no longer offer an accurate result in identifying the number of clusters in the dataset. Figure 3.8(a) displays the level vs. height diagram generated by using single linkage for the two cigar-shaped clusters. The level in this diagram refers to what iteration the algorithm is on and the height refers to the distance between the two clusters that are being merged. For the dataset in figure 3.7, a center of mass calculation is used. Using center of mass and calculating distance using the Euclidean distance measurement, we can see more clearly defined clusters in the level vs. height diagram. Figure 3.8(b) displays the level vs. height diagram for using center of mass on the two cigar-shaped clusters. The clusters are defined in these level vs. height diagrams by noting large jumps in distance between levels. The larger the distance of these jumps, the more clearly defined the clusters are.



(a) Single Linkage (b) Center of Mass

*Fig. 3.8:* Level vs. Height Charts for Cigar-Shaped Clusters

Comparing figures 3.8(b) and 3.8(a), the obvious jump in the center of mass diagram can be seen, and therefore the separation between the two clusters more easily identified. The small jump at level 143 in figure 3.8(b) is where some of the outliers begin to start merging first into the top cluster and then into the bottom cluster.

Figure 3.3 displays the cluster of outlier points that are merged together with the bulk of the top cluster that is already formed. This is a good way to identify possible outliers.

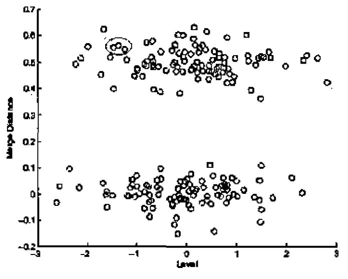


*Fig. 3.9:* Outliers Being Merged for Center of Mass

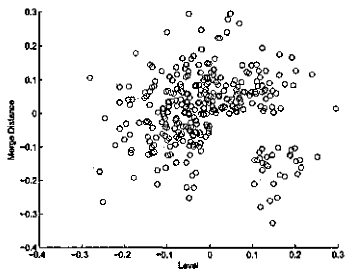
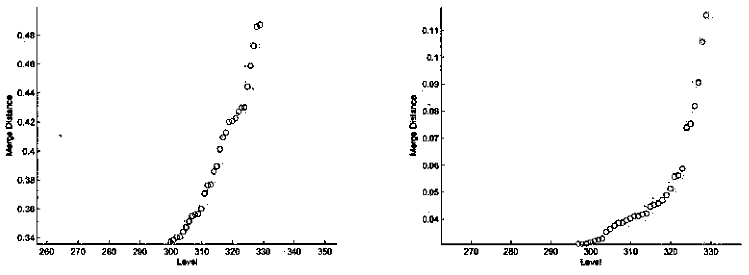


*Fig. 3.10:* Higher Mass Cluster With Smaller Mass Cluster

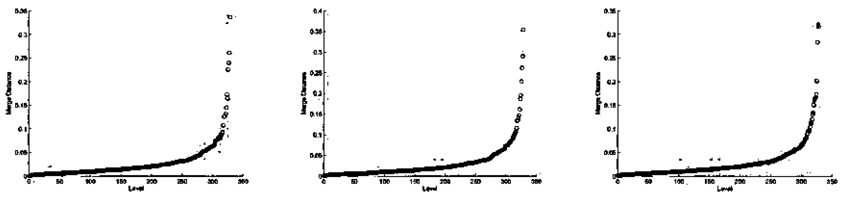
Another case where center of mass works well can be seen in figure 3.10, where one cluster with a much larger concentration of data points is in close proximity to a smaller cluster with much fewer data points. The resulting level vs. height diagram for center of mass can be seen in figure 3.11(a) and the single linkage level vs. height diagram can be seen in figure 3.11(b). Center of mass gives a clearer result as to the number of clusters in the dataset than other existing proximity calculations offered by hierarchical clustering.



(a) Center of Mass (b) Single Linkage

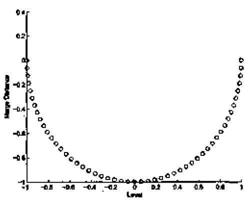
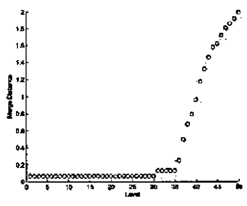
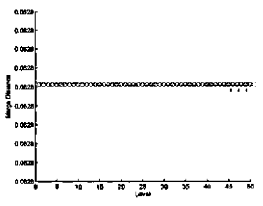
*Fig. 3.11:* Center of Mass and Single Linkage Comparison for Higher Mass Cluster With Smaller Mass Cluster

Figure 3.12 offers the level vs. height diagrams for centroid, average, and median linkages for the dataset seen in figure 3.10. As can be seen, center of mass offers comparable results. While they all generally out perform single linkage in regards to clearly defining the clusters in a dataset, they each offer varying results for different datasets. There is not one method that outshines the rest in every case. It is important to decipher which method will offer the best results. This can be determined by testing the dataset with agglomerative hierarchical using each of the methods and comparing the results.



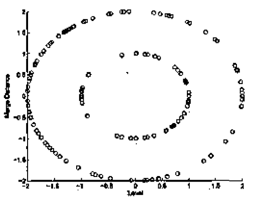
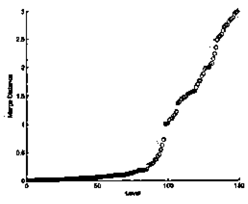
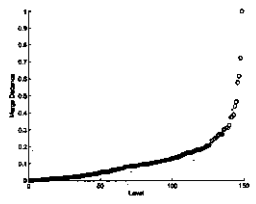
(a) Centroid Linkage (b) Average Linkage (c) Median Linkage

*Fig. 3.12:* Level vs. Height Diagrams for Higher Mass Cluster With Smaller Mass Cluster

(a) U-Shaped Plot (b) Center of Mass (c) Single Linkage

*Fig. 3.13:* Level vs. Height Diagrams for U-Shaped Plot

(a) Rings (b) Center of Mass (c) Single Linkage

*Fig. 3.14:* Level vs. Height Diagrams for Concentric Rings

Figures 3.13 and 3.14 are two more examples of center of mass. The former is a comparison of single linkage and center of mass on a U-shaped plot. Single linkage's result is a line in this case, because the distance between datapoints is equal in this example. Center of mass, however, displays many jumps throughout the level vs. height chart. This is due to the fact that the caculated center for each cluster moves as each new datapoint and cluster are merged into it. In figure 3.14 a somewhat different situation is happening. The dataset for this example contains two rings. The distance between datapoints within each ring in this example is not equal, unlike the U-shaped plot. The last several jumps in figure 3.14(c), starting at level 146, are where large clusters in the inner ring begin to merge with large clusters formed on the outer ring. This happens because the distance between the clusters within each ring has become greater than the distance between each ring. The center of mass level vs.

height chart in this case displays a jump at level 98. This is the exact moment where a cluster within the inner ring merges with a cluster on the outer ring. This happened much quicker for center of mass than single linkage, because the the centers of the clusters are recalculated with each merge. The centers for the clusters in each ring move toward the center of the plot with each merge. Eventually, a cluster from the outer ring and one from the inner ring were close enough to be merged together. Both of the former plots are indications of where center of mass does not perform well, but still provides us with at least some information. Significant jumps in merge distance are used to indicate possible outliers or clusters being merged together, but in these cases the merge distances are just a side effect of the cluster center calculation for center of mass.

# 4. CONCLUSION

Cluster analysis is an important tool for data exploration. Whether the algorithm groups the data around centers, groups by high density regions, or creates a hierarchical structure, they all serve the purpose of extracting some meaningful information out of the data. The path through clustering data includes the initial preprocessing of the dataset, the execution of the clustering algorithm on the dataset, using a validation method to determine cluster quality, and finally, interpreting the results. There is no universal clustering algorithm that can be applied to any dataset. Each algorithm applies to specific datasets and each validation method applies to certain groups of clustering algorithms.

An extendable general platform for clustering and validation has been implemented. This platform allows users to choose from several different algorithms and similarity measures, as well as a number of validation methods. These validation methods aid the user in picking the appropriate clustering algorithm and input parameters, as well as determining overall cluster quality. The platform is also extendable, allowing a user to easily plug in their own clustering algorithm written either in Java or Matlab.

Solutions to certain problems in the field of cluster analysis, such as estimating the number of clusters, have also been presented. Utilizing center of mass, a method for obtaining the center of a cluster for agglomerative hierarchical, a user can more accurately decipher the number of clusters in a dataset given the results of running the agglomerative hierarchical algorithm on the dataset. This information then allows the user to run additional clustering algorithms with more accurate input parameters or just makes the results of agglomerative hierarchical more useful in more situations.

## *4.1 Future Work*

Currently, this platform assumes the data loaded into it has been pre-processed and does not perform any feature selection or extraction. An important improvement would be to include some way to deal with missing data and to omit any non-useful attributes. This could possibly take the form of using PCA to find the principle components and using that information, locate the pertinent attributes.

## REFERENCES

[1] D. Arthur and S. Vassilvitskii. K-Means++: The Advantages of Careful Seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[2] A. Asuncion and D.J. Newman. UCI Machine Learning Repository, 2007. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[3] R. Harrisan B. Chen, P.C. Tai and Y. Pan. Novel Hybrid Hierarchical-K-means Clustering Method for Microarray Analysis. *IEEE Computational Systems Bioinformatics Conference Workshops*, 2005.

[4] N. Bolshakova and F. Azuaje. Cluster Validation Techniques for Genome Expression Data. *Signal Process.*, 83(4):825–833, 2003.

[5] F. Boutin and M. Hascoet. Cluster Validity Indices for Graph Partitioning. In *IV '04: Proceedings of the Information Visualisation, Eighth International Conference*, pages 376–381, Washington, DC, USA, 2004. IEEE Computer Society.

[6] C. Tang D. Jiang and A. Zhang. Cluster Analysis for Gene Expression Data: A Survey. *IEEE Trans. on Knowl. and Data Eng.*, 16(11):1370–1386, 2004.

[7] K.H. Lee D. Kim and Doheon Lee. Fuzzy Cluster Validation Index Based On Inter-Cluster Proximity. *Pattern Recogn. Lett.*, 24(15):2561–2574, 2003.

[8] F. Dellaert. The Expectation Maximization Algorithm, February 2002.

[9] C. Ding and X. He. K-Means Clustering Via Principal Component Analysis. In *ICML '04: Proceedings of the Twenty-First International Conference On Machine Learning*, page 29, New York, NY, USA, 2004. ACM.

[10] M. Eisen. "Cluster 3.0". http://bonsai.ims.u-tokyo.ac.jp/mdehoon/software/cluster/software.htm.

[11] M. Ester, H.P. Kriegal, H. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Datasets with Noise. In *Proc. 2nd Int. Conf. KDDD*, pages 1232–1239, 1996.

[12] C. Ma G. Gan and J. Wu. *Data Clustering Theory, Algorithms, and Applications.* SIAM, 2007.

[13] S. Chatterjee G. Sheikholeslami and A. Zhang. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In *Proceedings of the 24th VLDB Conference*, 1998.

[14] Rapid-I GmbH. "RapidMiner". http://www.rapid-i.com/.

[15] M. Halkidi and M. Vazirgiannis. Clustering Validity Assessment: Finding the Optimal Partitioning of a Data Set. *Data Mining, IEEE International Conference on*, 0:187, 2001.

[16] M. Halkidi and M. Vazirgiannis. A Density-Based Cluster Validity Approach Using Multi-Representatives. *Pattern Recogn. Lett.*, 29(6):773–786, 2008.

[17] A. Hinneburg and D.A. Keim. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. *American Association for Artificial Intelligence*, 1998.

[18] I. T. Jolliffe. *Principal Component Analysis.* Springer, October 2002.

[19] W.L. Ruzzo K. Yeung. An Empirical Study on Principal Component Analysis for Clustering Gene Expression Data. *Bioinformatics*, 17:763–774, 2001.

[20] H. Liu M. Dash and X. Xu. Merging Distance and Density Based Clustering. In *DASFAA '01: Proceedings of the 7th International Conference on Database Systems for Advanced Applications*, pages 32–39, Washington, DC, USA, 2001. IEEE Computer Society.

[21] M. Vazirgiannis M. Halkidi and Y. Batistakis. Quality Scheme Assessment in the Clustering Process. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 265–276, London, UK, 2000. Springer-Verlag.

[22] Y. Batistakis M. Halkidi and M. Vazirgiannis. Clustering Algorithms and Validity Measures. In *Proceedings of the 13th International Conference on Scientific and Statistical Database Management*, pages 3–22, Washington, DC, USA, 2001. IEEE Computer Society.

[23] Y. Batistakis M. Halkidi and M. Vazirgiannis. Cluster Validation Methods : Part 1. *SIGMOD Rec.*, 31(2):40–45, 2002.

[24] Y. Batistakis M. Halkidi and M. Vazirgiannis. Clustering Validity Checking Methods: Part II. *SIGMOD Rec.*, 31(3):19–27, 2002.

[25] M. Matteucci. “A Tutorial On Clustering Algorithms”. http://home.dei.polimi.it /matteucc/Clustering/tutorialhtml/index.html.

[26] U. Maulik and S. Bandyopadhyay. Performance Evaluation of Some Clustering Algorithms and Validity Indices. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(12):1650–1654, 2002.

[27] E.J. Pauwels and G. Frederix. Finding Salient Regions in Images: Nonparametric Clustering for Image Segmentation and Grouping. *Comput. Vis. Image Underst.*, 75(1-2):73–85, 1999.

[28] S. Petrović. A Comparison Between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS Clusters. In *Proceedings of the 11th Nordic Workshop on Secure IT-systems*, pages 53–64, 2006.

[29] D. Gunopulos R. Agrawal, J. Gehrke and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Rec.*, 27(2):94–105, 1998.

[30] P. Rousseeuw. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[31] D. Tritchler S. Fallah and J. Beyene. Estimating Number of Clusters Based on a General Similarity Matrix with Application to Microarray Data. *Statistical Applications in Genetics and Molecular Biology*, 7, 2008.

[32] L. Smith. A Tutorial on Principal Component Analysis. February 2002.

[33] J. Yang W. Wang and R.R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[34] R. Xu and D. Wunsch II. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16(3), May 2005.

[35] X. Zhang Y. Zhang, W. Wang and Y. Li. A Cluster Validity Index for Fuzzy Clustering. *Information Sciences*, 178(4):1205 – 1218, 2008.