IE411 Term Project

Vlad Predovic

Monday, December 7th, 2015.

# Table of Contents

[Table of Contents 1](#_Toc436767578)

[Figures and Tables 2](#_Toc436767579)

[Introduction 3](#_Toc436767580)

[K-Means 3](#_Toc436767581)

[K-Medoid 3](#_Toc436767582)

[Implementation Details 4](#_Toc436767583)

[Input and Verification 4](#_Toc436767584)

[Selecting Initial Centroids/Medoids 4](#_Toc436767585)

[Forming and Finalizing Clusters 5](#_Toc436767586)

[*K-Means (Lloyd’s Algorithm)* 5](#_Toc436767587)

[*K-Medoid (Partitioning Around Medoids)* 6](#_Toc436767588)

[Assessing Cluster Variation 7](#_Toc436767589)

[User Guide 7](#_Toc436767590)

[Section I: Guide Introduction 8](#_Toc436767591)

[Section II: Data Input 8](#_Toc436767592)

[*Manual Input:* 8](#_Toc436767593)

[*Randomized Input:* 8](#_Toc436767594)

[*Input From File:* 8](#_Toc436767595)

[Section III: Results and Exporting Data 9](#_Toc436767596)

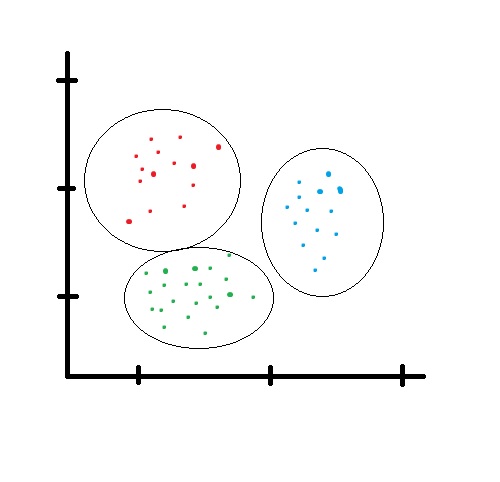
[Section IV: The Access Database 9](#_Toc436767597)

[References 10](#_Toc436767598)

# Figures and Tables

* *Figure1: Cluster Example………………………………………………………………….pg . 3*
* *Figure2: Distance between Points.……………………………………………………….pg . 3*
* *Figure3: K-Means Weakness…..………………………………………………………….pg . 3*
* *Figure4: Seed Randomization….………………………………………………………….pg . 5*
* *Figure5: Centroid Comparison..………………………………………………………….pg . 6*
* *Figure6: Form-Reference-Picture ……………………………………………………….pg . 8*

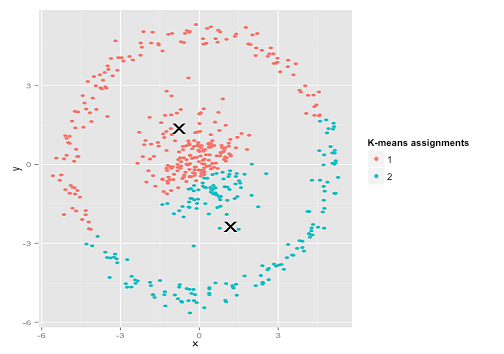
# Introduction

The Visual Basic program that goes along with this paper is designed to provide a quick and user friendly way to input data and perform cluster analysis. The program allows the user to pick between K-Means and K-Medoid, two different methods used for performing this type of analysis .

Cluster Analysis in a short statement is the grouping of data into separate categories based on a quantitative attribute. The grouping can be done through different methods but ultimately each data point becomes attributed to a distinct group. This results of this type of analysis are easier to understand graphically, but can be imagined to be somewhat like the picture on the right. A common use of this analysis is to try to differentiate data and see if it came from different sources.

*Fig. 1 Cluster Example*

K-Means

http://www.purplemath.com/modules/xyplane/dist07b.gif K-Means is a common method employed to solve the clustering problem and a commonly associated mathematical implementation to solve it is labeled as Lloyd’s algorithm. According to MacKay (2003), it is a competitive algorithm as the clusters compete to gain ownership of the data points. It begins with defining an n amount of centroids within the range of your data set. There are many different ways to place these centroids but ideally you would want them spread throughout the data. After this you associate each data point to the centroid which it has the lowest distance from. Since this program only allows 2-dimensional objects, finding the distances can be done using the Pythagorean theorem as in figure 2. Afterwards new centroids are deduced from all the objects that currently make up each cluster. The process loops, attributing each data point to a given cluster and deducing the center point until these stop changing altogether.

*Fig. 2: Distance between points*

The K-Means method has some drawbacks. The easiest to represent of these is the idea that it tries to create circular groupings of data as in figure 1. If you had an example with a different type of pattern, the K-means method would fail such as the one shown in figure 3 to the right where it is obvious that all the data in the middle is one cluster and the large ring on the outside is a separate set of data.

*Fig. 3: K-Means Weakness*

K-Medoid

The K-medoid method is similar to the k-means method described above in that it attempts to attribute data objects into clusters depending on the distance of these points to the designated cluster centroids. One of the more common realizations of this method is the Partitioning Around Medoids (PAM) method which begins by selecting random existing data points as your medoids. Each data point is then associated to the closest medoid. Each data point in a cluster is then taken as the medoid and the total distance cost of all data points to this point is calculated. The central point of the configuration with the lowest cost is selected and that point then becomes the medoid. The algorithm loops as all the data is then re-associated to the closest of the new medoids. Once the medoids do not change anymore the algorithm ends.

The main drawback of the K-Medoid method is that it is usually much more expensive to run than the K-means method. In this case you have to calculate every distance from every point to all others each time you run an iteration. The latter method only requires that you calculate the distance to the given centroid points. However, according to Velmurugan (2010), the K-Medoid method is considerably more reliable thank the K-means method since it takes the ‘center-most’ data point in the cluster as its medoid instead of the average value of the cluster. This makes the algorithm less susceptible to being displaced by outliers. The K-means method is more comparable to the mean, and therefore an aggressive outlying point can severely skew the data.

# Implementation Details

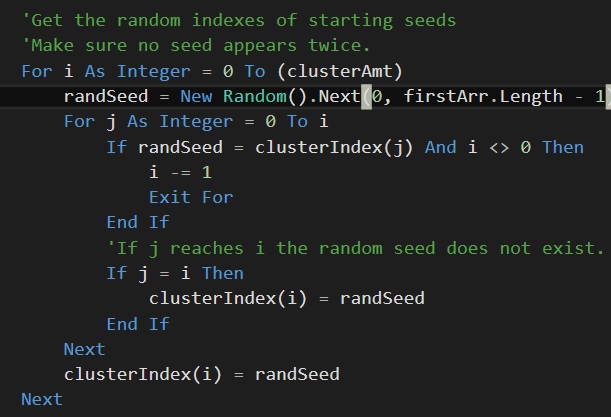
This next section will go into detail over the implementation of the algorithms used by the program. These are known as Lloyd’s algorithm and the Partitioning Around Medoids algorithm, used to solve the K-Means method and K-Medoids method respectively. They methods will be split into four subsections. Each one of these will outline the different paths taken for the respective section or clarify as to why similar code was utilized.

Input and Verification

Although this is done prior to running either algorithm, outlining the methods used to collect and verify the data is still important to understanding how it works. There are three separate ways to input data. Fifty numbers can be generated, fifty numbers can be inserted from a file, or the user can manually input data. If done manually, the program is set up so that you cannot input invalid data by disabling the specific buttons until conditions are met.

When inserting data, two separate global arrays will become filled with a maximum of 50 attributes each and a counter . Any amount of numbers up to 50 can be added, as the arrays will automatically resize to the amount of variables that were inserted. The quantity of clusters that will be seeded is also decided by the users, with a default value of two. When the Calculate button is pressed for either method, the amount of clusters chosen is immediately saved to a local variable in the associated sub-procedure.

Selecting Initial Centroids/Medoids

Selection the initial Centroids or Medoids is done in the same fashion, as the code is essentially reused with minor tweaks. After trying multiple methods including trying to standardize the data and seed it at a local level, the seeding was done using a randomizing algorithm as this appeared to be the most fair method.

The algorithm begins by picking a starting index between 0 and the amount of data points the user set. It then checks to make sure that this index was not previously picked. If it was, another random number is then picked. This repeats until the correct amount of clusters is reached as shown in figure 4. The data existing in the global arrays at the selected indexes then become the initial seeds. These values are transferred to a local 2-dimensional array titled ‘Medoids’ or ‘Centroids’ depending on which button was clicked.

*Fig. 4 Seed Randomization*

In the case that the first random number picked is the 0th index, this value is accepted. Otherwise an error occurs as the counter drops below 0, since an empty array has it’s values all initialized to 0.

Forming and Finalizing Clusters

### K-Means (Lloyd’s Algorithm)

The entire process of calculating the optimum centroids exists inside a Do-Until loop. This allows the algorithm to terminate smoothly if the initial seeds happen to be the optimal solution points. The Boolean variable Switchvar is set to True as soon as the loop is entered. This will be important later.

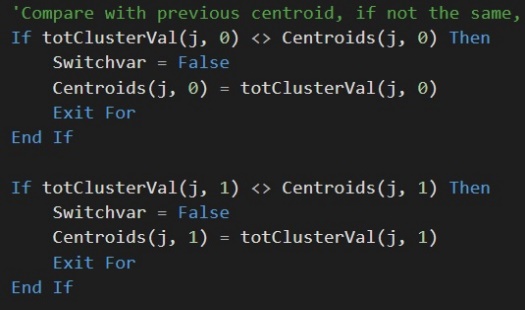
The first steps are all conducted for each individual data point separately by way of an encompassing outer-for loop. The steps taken are for each point are shown below:

* A loop calculates the distance between the given data point and each chosen centroid. The values are saved in a temporary one-dimensional array who’s size is equal to the amount of centroids selected. This inner loop ends.
* A function ‘minFinder’ returns the index of the temporary array in which the lowest distance calculated is found. The index value is add to a third global array the size of the amount of data points which represents the cluster that the given data point is connected to. The outer loop ends here.

A second outer loop begins, this time iterating through each cluster index. Now that the data points are all linked to a given centroid, they can be acted on categorically.

* An inner loop that runs through every data point checks if that point belongs to the given cluster by comparing the cluster index the outer-loop is going to with the value saved to the global array representing the cluster that the data point is associated to. This is done by comparing indexes. If it is, the attributes of that data point are added to a total, and a counter is incremented. Both inner and outer loop end.

A third outer loop begins once again iterating through each cluster index. We now have the total sum of the 2 dimensional data points for each cluster. We do not necessarily need a third loop, but it was put in as it helps separate the program into sections.

* These totals are divided by the quantity of elements in a given cluster to get the new centroid. If the centroid is different from the previous value help by that centroid with the same index, the Boolean variable ‘Switchvar’ is set to false.

*Fig. 5 Centroid-Comparison*

The Do-Until loop iterates until the Boolean variable returns true which would signify that the centroids are no longer changing as the switch was never triggered.

### K-Medoid (Partitioning Around Medoids)

Like the K-Means method, the algorithm employed for the K-Medoid method begins with an all-encompassing Do-While loop that iterates until a switch indicates that the Medoids are not changing anymore.

The next step is also functionally similar, as the distance between each point and medoid is calculated. The point then becomes associated to the medoid that is the shortest distance away. However, the Manhattan distance is used to calculate the distance between points as opposed to the Euclidean distance for the K-Means method.

The algorithm then loops through the amount of clusters entered by the user.

* The cost of each cluster is summed up by iterating through and calculating the distance between the medoid and the points associated to it. The resulting value is the initial cost of the cluster-medoid combination to begin the comparison process.

The next step involves iterating through each data point in a cluster and considering it as the medoid.

* For each data point as the medoid, the program finds the total distance to every data point in the cluster and compares it to the current total cost for the medoid found previously. If the total cost is less, then this point becomes the established medoid.

The results are the compared to the existing Medoids. If no change is noticed, the Do-Until loop will exit as this then means that the optimal medoids have been selected.

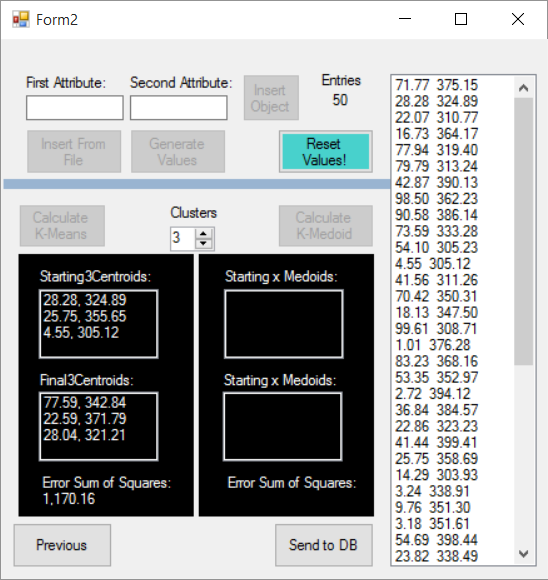
Assessing Cluster Variation

Assessing the variation in the data is done in the same way for both the K-Means and K-Medoid methods.

A for loop runs through each data point calculating the Euclidian distance to its associated cluster. This value represents the sum of the distances from each point to the closest cluster centroid/medoid and can also be called the Error Sum of Squares(SSE). The resulting value can be compared with other resulting values using the same data to assess the most efficient clusters.

# User Guide

This section provides a comprehensive guide to running the visual basic program. It assumes the user is beginning on the opening page that appears when the solution is ran and details the many effects that different combinations of inputs will have on the solution. It also is aimed to walk the user through how to correctly set up and run the program, as well as how to interpret the results.

*Note: Figure Six below serves as a reference when looking through the guide. The view represents how the program appears after running K-Means analysis on 50 randomly generated values.*

*Fig. 6 Form-Reference-Picture*

Section I: Guide Introduction

Upon starting up the program you should be taken to an initial form that introduces the program. The first page of the solution presents the user with the concept of data clustering. Afterwards, it gives a brief numbered explanation on how to run the program and how to manage your results.

Clicking next takes the user to a page which explains in a few sentences basics of the K-Means method. It then gives a quick listing of the pros and cons of this method meant to give the user a better idea of whether or not they should employ the K-Means method. Finally the page provides a sentence on what the data that program displays represents.

Clicking next a second time will take them to the K-Medoids page. The page is in the same format as the prior, with a quick introduction to the method and a listing of pros and cons, ending with an explanation of how the data is presented. In-depth descriptions of both these methods can be found in the Implementation Details section of this write-up.

Clicking next again will initiate the main part of the program.

Section II: Data Input

The writing of this program focused on giving the user narrow pathways of execution to avoid as many errors as possible. Buttons will constantly enable and disable depending on the actions the user takes, making it easy to understand what is still available to the user.

The user has three separate options to input data:

### Manual Input:

Inserting data is done by filling in the text boxes labeled ‘First Attribute:’ and Second Attribute’ on the top left corner of the form. A valid number needs to be inserted into each field in order for the ‘Insert Object’ button to become enabled, preventing the user from entering invalid data. The data points appear in the list box on the right side of the form and a counter next to the “Insert Object’ button displays the count of times the user has inserted. However, manually inserting data will also disable both other options of inserting data. Additionally, at least three more data points must be entered than the amount of clusters currently chosen in order to enable the calculate buttons.

### Randomized Input:

Clicking the button ‘Generate Values’ will insert fifty random values between 0 and 100 into the system. These will also be displayed in the textbox to the right. Doing so will disable all three buttons with insertion capabilities and enable the buttons used to perform cluster analysis.

### Input From File:

The button ‘Insert From File’ will insert data from a text file named ‘**Numbers.txt’** placed in the subdirectory bin\debug of the visual studio project. The data in the file must be inserted with one space between each number. The program will align the first fifty numbers and the second fifty numbers such that data point 1 would consist of the first number that appeared in the file and the fifty-first number that appeared. Doing so will once again disable all other insertion buttons and enable calculation buttons.

After data input is complete user now has the option of changing the quantity of clusters using the ‘NumericUpDown’ box in the center of the screen. At this point all data necessary for calculations has been inserted. The next step is to press a Calculate button depending on what

algorithm better suits the needs of the user. Alternatively, pressing the light blue ‘Reset Values’ button would result in a complete reset of the form. This can be done at any point in order to run a different test or generate different input.

Section III: Results and Exporting Data

From this point on it is assumed that the user has input all necessary data and pressed either calculate buttons. The data appears in the exact same fashion regardless of the test that is ran within the confines of the black rectangular squares. The centroid/medoid values appear in *x, y* format. The initial randomized seeds are displayed under ‘Starting x Centroids/Medoids’ as well The final optimal x, y values are displayed under ‘Final x Centroids/Medoids’. Finally, the Error Sum of Squares(SSE) is displayed at the bottom of the current black rectangle.

Pressing the ‘Send to DB’ button on the bottom right side of the form will save the results displayed in Section III to an access located in the same directory as the execution file. A message box will appear with the statement *"If this is the only message displayed, you have successfully saved your data".* As stated, any additional messages will provide details as to why the program failed.

Section IV: The Access Database

The access fill that accompanies this project can be found in the same folder as the main executable for the solution (TermProject\_Predovic). Pressing the ‘Send to DB’ button will add one entry every time.

In case of loss it can be easily be recreated by recreating the description below, paying strict attention to formatting such as lower/upper case letters.

* The access file is titled **Database1**
* The file is made up of one table named **TBL\_ClusterResults**. Creating additional tables will not affect the program.
* The table has five separate columns:
  + **ID**: Auto increments for each entry.
  + **Run\_Date**: Automatically populates with today’s date.
  + **No\_Clusters**: Represents the number of clusters.
  + **Centroids\_Medoids**: Depicts Final cluster Centroids/Medoids
  + **Error\_Sum\_of\_Squares**: Value of the objective function for the instance ran.

# References

 MacKay, David(2003).Chapter 20. An Example Inference Task: Clustering (PDF).Information Theory, Inference and Learning Algorithms. Cambridge University Press. pp. 284–292.

*Velmurugan, T, (2010). Computational Complexity between K-Means and K-Medoids Clustering Algorithms for Normal and Uniform Distributions of Data Points.* *Journal of Computer Science,* *6*(3), 363-368.