Laboratory 14 – Week 22

## Data Clustering

## 14.1 Introduction

**Firstly, this worksheet is one of the worksheets from which your laboratory worksheets portfolio of work will be assessed.**

The laboratory involves the application and analysis of the KMeans clustering algorithm. We will also use the MATLAB application to display some of the datasets we are working with.

## 14.2 Preliminaries

Make sure that you are familiar with the material in lecture “15.1 Bin Packing and Data Clustering”.

Download the file ClusterLab.txt from Appendix A. It consists of three columns which represents 100 points (*x*,*y*,*z*) in three dimensional space. Place this file somewhere that is easy to access. From this point on it will be assumed that it can be found in c:\temp\ClusterLab.txt.

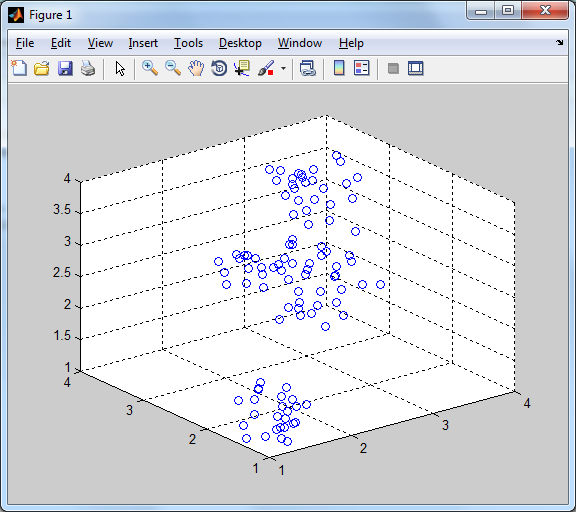
Open up the MATLAB application. You can do this by selecting the Windows button and typing “MATLAB” into the “Search programs and files” text box. Select MATLAB from the list.

Within the MATLAB command window type the following commands, one per line in the order they appear:

x = dlmread('c:\temp\ClusterLab.txt');

scatter3(x(:,1),x(:,2),x(:,3));

The first line reads the data into a variable (2D array) called x, the second line produces a 3D scatter plot of the data that should be similar to the diagram below:



You can rotate the graph by clicking the  button and selecting and dragging the corners.

The dataset appears to have three distinct clusters where two are quite close together. In fact the data should be clustered as follows:

|  |  |  |
| --- | --- | --- |
|  | Cluster 1: points 1-25 |  |
|  | Cluster 2: points 26-75 |  |
|  | Cluster 3: points 76-100 |  |

In this dataset and other datasets that we are going to cluster, each row is an instance or record of the data; this could correspond to survey data from people (a row for each person), or samples from experiments etc... The columns correspond to the measurements we may have taken (also known as features, variables or attributes), for example, height, weights, temperature etc... The objective of data clustering is to group or arrange the rows together into sets (clusters) where items in the same group (cluster) have similar measurements (columns).

We are going to use the KMeans algorithm and the Kappa metric to see if we can cluster these points, and then we are going to work with another dataset.

## 14.3 The KMeans Class

Create an Eclipse project called Lab14 and extract and add the KMeans.java from Appendix A.

The KMeans class has a number of methods that we are going to use:

**static** **public** **double**[][] ReadArrayFile(String filename,String sep)

This reads a text file of data (where the columns are separated by a string sequence, for example a comma in ClusterLab.txt) into a 2D array of doubles, where the rows correspond to the rows in the text file.

**static** **public** ArrayList<Integer> ReadIntegerFile(String filename)

This method reads in a list of integers (one per line) into an ArrayList. This can be used for reading in a clustering arrangement, see section 14.4.

**public** KMeans(**double** d[][],**int** nn,**int** ll)

This creates a new KMeans object. The parameters are: d is the dataset being clustered (rows are instances and columns are variables); nn is the number of variables in the dataset and ll is the numbers of rows in the dataset.

**public** ArrayList<Integer> RunIter(**int** nncc,**int** iter,ArrayList<Integer> real,**boolean** showlog)

This method runs the KMeans clustering algorithm for iter iterations. nncc is the number of clusters we are looking for, real is the expected clustering arrangement, (leave blank (empty) if unknown), showlog is a status flag and if it is false then no status or iteration data is shown. The method returns an ArrayList of Integers, representing the clustering of the input data.

**public** **static** **double** GroupingWK(ArrayList<Integer> a, ArrayList<Integer> b)

The GroupingWK method compares two clustering arrangements (represented as ArrayLists of Integers, a and b), using the Kappa metric as described in the lectures.

The return values of this method are as follows:

* A return value of -2 means the two inputs (a and b) are of different sizes
* A return value of -1 means the first vector is of size zero
* A return value of +1 is total agreement meaning identical clusters
* A return value of -1 is total disagreement (opposite arrangements)
* A return value of 0 is the expected agreement between two random clusters
* A real number between -1 and +1 is returned if no error is found

## 14.4 Exercise 1: Running the KMeans Algorithm

Within the KMeans class, a clustering arrangement of *n* items is represented by a *n* length vector, say *C*, where each element *ci = x* means that data item *i* is in cluster *x*. For example, imagine we had a result of {1,2,1,2,1,2} for 6 items, then rows/items {1,3,5} are in cluster 1 and rows/items {2,4,6} are in cluster 2.

We are now going to run the KMeans algorithm on the ClusterLab.txt dataset. Perform the following steps:

1. Create the expected clustering arrangement for the dataset as described in section 14.2. For example, the first 25 variables could be in cluster ID 1, the next 50 in cluster ID 2 and the last 25 in cluster ID 3. You will need to create an ArrayList<Integer> to store the expected arrangement in.
2. Read in the ClusterLab.txt dataset using the ReadArrayFile method within the KMeans class.
3. Create a new KMeans object, specifying the dataset, number of columns and number of rows as required.
4. Run the KMeans algorithm by calling the RunIter method, specify three clusters (since that is what we are expecting), ten iterations (this should be enough) and the expected clustering arrangement. Store the result in an ArrayList<Integer>.
5. Compare the result with the expected arrangement using the Kappa metric.

## 14.5 Exercise 2: How Consistent is KMeans?

Run the KMeans algorithm ten times and display the Kappa metric as in section 14.4 each time. What do you notice? Calculate the mean, maximum and minimum values.

## 14.6 Exercise 3: Clustering the Iris Dataset

We are now going to cluster [Fisher's](http://www-groups.dcs.st-and.ac.uk/~history/Biographies/Fisher.html) famous *Iris* dataset. Carry out the following steps:

1. Look at the following web address: <http://archive.ics.uci.edu/ml/datasets/Iris>.
2. Download the *Bezdek* version of the *Iris* dataset.
3. You will need to pre-process this dataset into a data text file and expected clustering text file. The best way to do this is to read the downloaded file into Excel.
4. Once you have imported the dataset into Excel, you should note that the actual data consists of 150 rows (instances) and 4 columns (variables). The assumption is that the rows will cluster according to the three classes, *Iris Setosa*, *Iris Versicolour* and *Iris Virginica*. You could use the search and replace facility in Excel to replace the class name for a cluster number (a different one per class).
5. Read in and cluster the dataset in the same manner as exercises 1 and 2.

## 14.7 Appendix A

The following class and text file contains the code and data you will need for this exercise sheet.

|  |
| --- |
|  |