DHRUV GARG 16 BCE 1190

MACHINE LEARNING - DIGITAL ASSIGNMENT 2

CLUSTERING USING DIFFERENT ALGORITHMS

INTRODUCTION

Clustering is a type of **unsupervised learning** method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

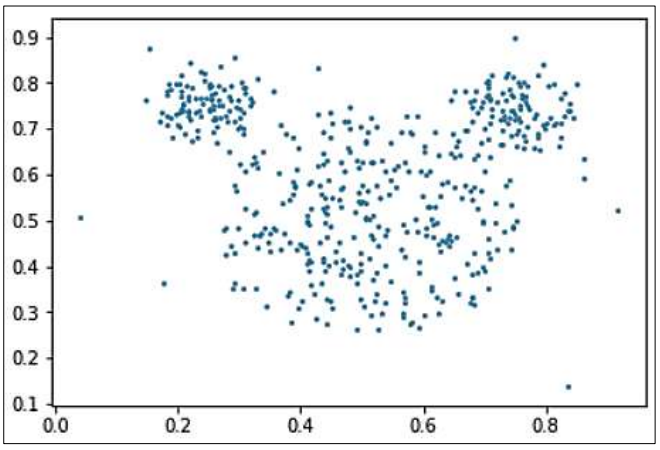
Clustering Methods:

1. Density-Based Methods: These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters. Eg: DBSCAN (Density-Based Spatial Clustering of Applications with Noise).
2. Hierarchical Based Methods: The clusters formed in this method forms a tree type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two categories:
    a. Agglomerative (bottom up approach)
    b. Divisive (top down approach)

    Eg: **AGNES, DIANA**, BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies) etc.
3. Partitioning Methods: These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter. Eg: **K-Means**, CLARANS (Clustering Large Applications based upon randomized Search) etc.
4. Grid-based Methods: In this method the data space are formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example STING (Statistical Information Grid), wave cluster, CLIQUE (CLustering In Quest) etc.
5. Clustering for Big Data: **SOM** (Self-Organizing Map)

IMPLEMENTATION

The dataset chosen was the **Mouse dataset**. It contained 500 rows and 3 columns. First 2 columns consisted of the X and Y coordinates of the left ear, right ear and head of the mouse. The third column actually had the label for the point, i.e. to which part of the mouse's face that point belonged. Since we are dealing with unsupervised learning, this column was dropped. First the data was read and checked for pre-processing. The data did not have missing values and was already scaled. Hence we did not need to use StandardScaler. However, the dataset did have some **noise**, and it can be seen from the following plot that while there are 3 clusters visible, there are some points which act as outliers for all the 3 classes.



The main objective of this assignment was to implement the 4 clustering algorithms:

a. K-Means
b. AGNES
c. DIANA
d. SOM

and infer from their clustering performance.

UNDERSTANDING K-MEANS THROUGH IMPLEMENTATION

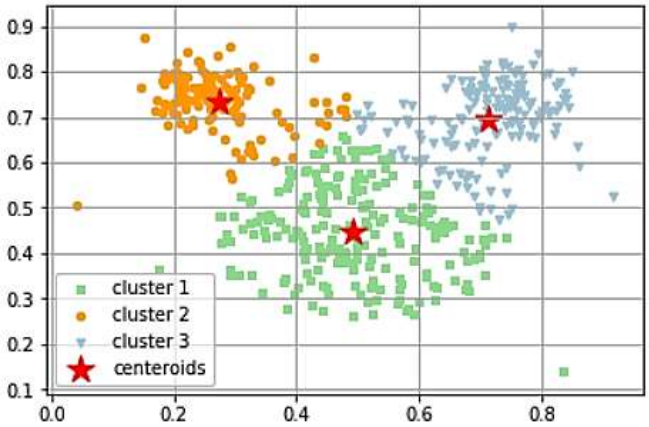
K-Means Algorithm:

1. Step 1: Initialization
   Randomly choose K examples (data points) from the dataset as initial centroids and that's simply because it does not know yet where the center of each cluster is.
2. Step 2: Cluster Assignment
   All the data points that are the closest to a centroid will create a cluster. Different distance metrics can be used while calculating the distance. Eg: Euclidean distance.
3. Step 3: Move the centroid
   After we have new clusters, we need centers. A centroid's new value is going to be the mean of all the examples in a cluster.
4. We'll keep repeating step 2 and 3 until the centroids stop moving, in other words, K-means algorithm is converged.

How to chose the number of clusters 'k'?

Dissimilarity(C) is the sum of all the variabilities of k clusters.

Variability is the sum of all Euclidean distances between the centroid and each example in the cluster.

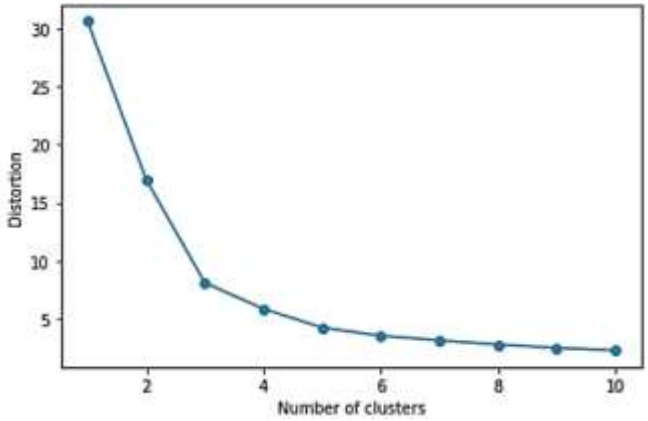


From here we can see that the left-ear and right-ear clusters formed also take up a part of the head area. Thus, although the clusters formed are good, we still have many points that are put into wrong clusters.

Since determining the number of clusters is a tricky affair, we have two tools at our disposal.

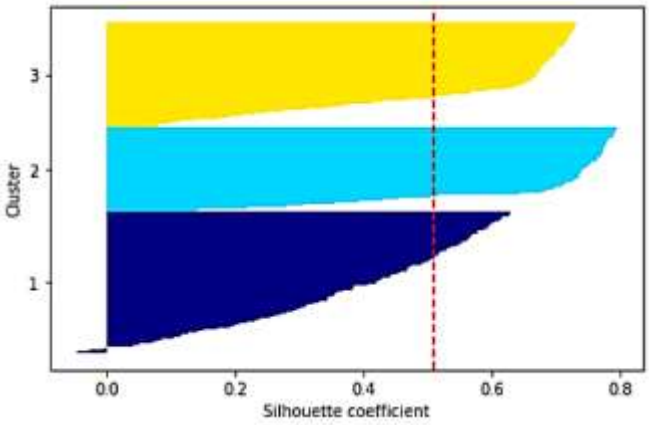
a. Elbow method

This method takes into account the distortion at each value of k. The point at which we get an inflection, we take that as the value of k.



b. Silhoutte coefficient

The silhouette value is a measure of how similar an object is to its own cluster compared to other clusters.



INFERENCE for K-Means

It is easy to understand and comprehend visually. Furthermore, it delivers training results quickly.

However, its performance is usually not as competitive as those of the other sophisticated clustering techniques because slight variations in the data could lead to high variance.

Furthermore, clusters are assumed to be spherical and evenly sized, something which may reduce the accuracy of the K-means clustering Python results.

UNDERSTANDING HIERARCHIAL CLUSTERING THROUGH IMPLEMENTATION

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. In general, the merges and splits are determined in a **greedy** manner. The results of hierarchical clustering are usually presented in a **dendrogram**.

AGNES Algorithm:

Agglomerative Nested clustering: This is a "bottom-up" approach - each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

This algorithm works by **grouping the data one by one** on the basis of the nearest distance measure of all the pairwise distance between the data point. It uses the Euclidean distance as the distance metric.

When should the groups be formed?

1) single-nearest distance or single linkage.

2) complete-farthest distance or complete linkage.

3) average-average distance or average linkage.

4) centroid distance.

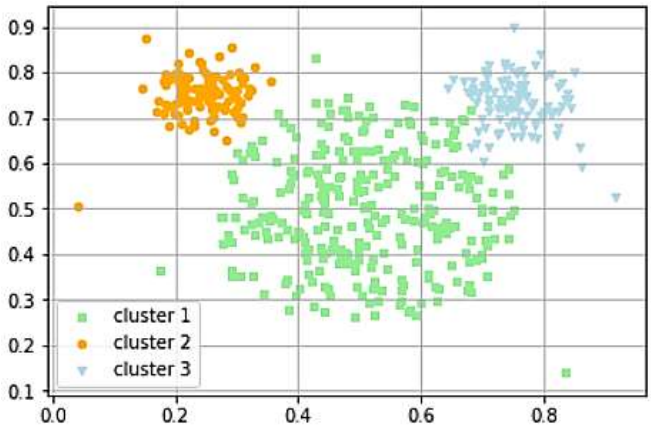
5) ward's method - sum of squared euclidean distance is minimized.

This way we go on grouping the data until one cluster is formed. Now on the basis of dendogram graph we can calculate how many number of clusters should be actually present.

WORKING of AGNES:

We begin with disjoint clustering having L(0) = 0. After finding the least distance pair of clusters in the current clustering, we merge them into a single cluster and update the distance matrix.

If the datapoints are in one cluster, we stop the algorithm. Else we continue from the second step.



We can see that the points have been clustered in a much better manner than K-Means clustering. The left-ear and right-ear clusters do not encroach onto the data points meant to be in the head cluster.

DIANA Algorithm:

Divisive Hierarchial clustering is a "top-down" approach - all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

The basic principle of divisive clustering was published as the DIANA (DIvisive ANAlysis Clustering) algorithm. Initially, all data is in the same cluster, and the largest cluster is split until every object is separate.

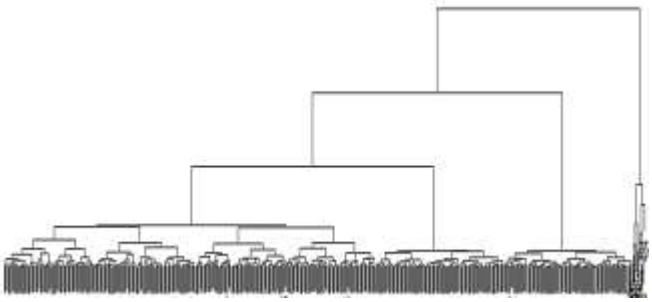
Because there exist $O(2^n)$ ways of splitting each cluster, heuristics are needed. DIANA chooses the object with the maximum average dissimilarity and then moves all objects to this cluster that are more similar to the new cluster than to the remainder.

WORKING of DIANA:

Divisive Hierarchical clustering- It is just the reverse of Agglomerative Hierarchical approach.

[I implemented DIANA in R]

Given below is the dendrogram for DIANA clustering.



INFERENCE for Hierarchial clustering

Advantages

1. No apriori information about the number of clusters required.
2. Easy to implement and gives best result in some cases.

Disadvantages

1. Algorithm can never undo what was done previously.
2. Time complexity of at least O(n2 log n) is required, where ‘n’ is the number of data points.
3. Based on the type of distance matrix chosen for merging different algorithms can suffer with one or more of the following:
    a. Sensitivity to noise and outliers
    b. Breaking large clusters

   c. Difficulty handling different sized clusters and convex shapes
4. No objective function is directly minimized
5. Sometimes it is difficult to identify the correct number of clusters by the dendogram.

UNDERSTANDING SOM CLUSTERING THROUGH IMPLEMENTATION

A self-organizing map (SOM) is a clustering technique that helps you uncover categories in large datasets, such as to find customer profiles based on a list of past purchases.

It is a special type of **unsupervised neural networks**, where neurons are arranged in a single, 2-dimensional grid, which can take the shape of either rectangles or hexagons.

Through multiple iterations, neurons on the grid will gradually coalesce around areas with high density of data points.

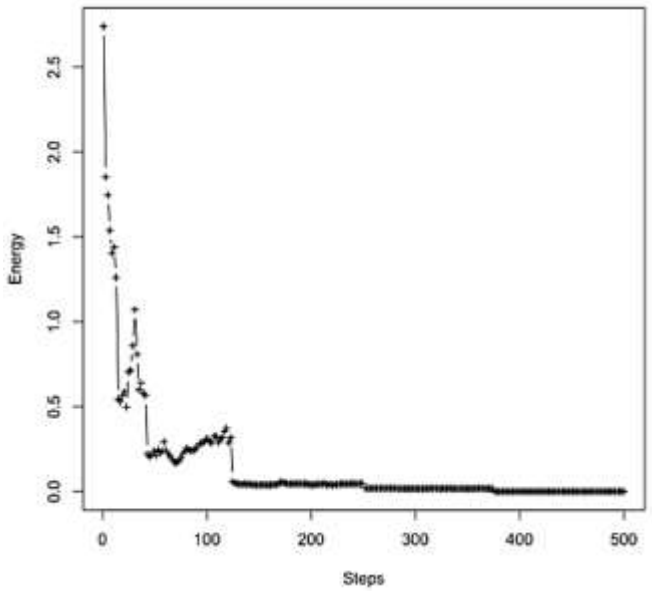
Hence, areas with many neurons might reflect underlying clusters in the data. As the neurons move, they bend and twist the grid to more **closely reflect the overall topological shape** of our data.

WORKING of SOM:

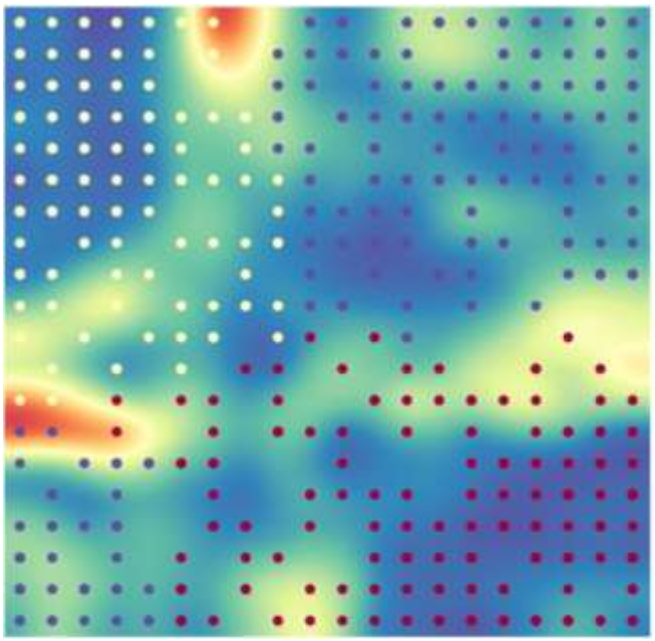
1. Initially, neurons in the SOM grid start out in random positions, but they are gradually massaged into a mould outlining the shape of our data.

2. We can see that the grid's shape stabilizes after a couple of hundred iterations.

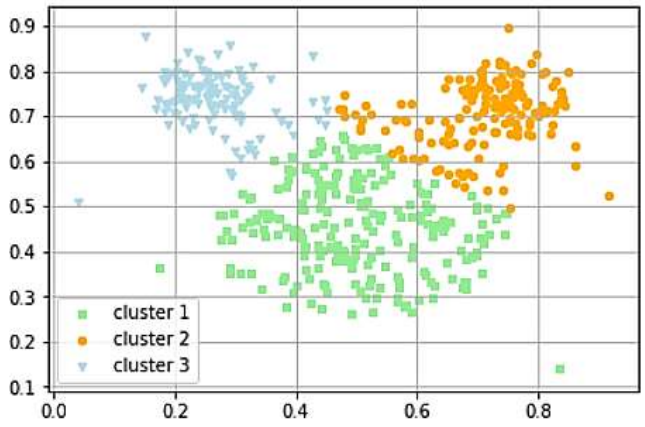
3. To check that the algorithm has converged, we can plot the evolution of the SOM's energy—initially, the SOM evolves rapidly, but as it reaches the approximate shape of the data, the rate of change slows down.



4. To get an overview of how many data points each neuron corresponded to, we can plot a frequency map of the grid, shown below.



5. From the frequency map, we can see a clear divide separating a top left neuron cluster from a top right cluster and a bottom right cluster.
6. To verify that there is indeed a divide, we can plot the different clusters, which visualizes how much neurons differ from each other in 2-dimensional space.



INFERENCE for SOM clustering

Advantages

1. **Easy to understand.** Unlike Multidimensional Scaling or N-land, people can quickly pick up on how to use them in an effective manner.
2. They **work very well:** They classify data well and then are easily evaluating for their own quality so you can actually calculated how good a map is and how strong the similarities between objects are.

Disadvantages

1. **Finds different similarites among the sample vectors.** SOMs organize sample data so that in the final product, the samples are usually surrounded by simliar samples, however similar samples are not always near each other. If you have a lot of shades of a colour, not always will you get one big group with all the purples in that cluster. Sometimes the clusters will get split and there will be two groups of purple. Using colors we could tell that those two groups in reality are similar and that they just got split, but with most data, those two clusters will look totally unrelated.

RESULT:

As we could observe from this implementation and through Python / R documentations, there are different scenarios where the different algorithms would be the most useful, ie. they work for different sizes of data and different sizes of clusters. In this implementation, all 4 do a fairly good job. **Hierarchial clustering outperformed the other two in this implementation.**

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points