COT 4521 Final Project Report
Clustering Algorithms: A comparison between K-Means Clustering and DBSCAN Clustering

Param Vipulkumar Chokshi
University of South Florida
Department of Computer Science and Engineering
pvc@usf.edu

1. Introduction

Clustering is an unsupervised machine learning task. It is also referred to as cluster analysis.In clustering algorithms, we provide the algorithm a lot of input data with no labels and let it find any groupings in the data it can.Those groupings are called clusters. A cluster is a group of data points that are similar to each other based on their relation to surrounding data points. Clustering is used for things like feature engineering or pattern discovery.

Types of Clustering Algorithms:
Density-based
In density-based clustering, data is grouped by areas of high concentrations of data points surrounded by areas of low concentrations of data points.
Distribution-based
With a distribution-based clustering approach, all of the data points are considered parts of a cluster based on the probability that they belong to a given cluster.
Centroid-based
These types of algorithms separate data points based on multiple centroids in the data. Each data point is assigned to a cluster based on its squared distance from the centroid. This is the most commonly used type of clustering.
Hierarchical-based
Hierarchical-based clustering is typically used on hierarchical data, like you would get from a company database or taxonomies. It builds a tree of clusters so everything is organized from the top-down.

2. Project Objective and Setup

The goal of the project was to implement two clustering algorithms:
K-Means clustering algorithm (Centroid Based)
DBSCAN clustering algorithm (Density-based spatial clustering of applications with noise) (Density Based)

The implementations were compared to scikit-learn's (a python library for machine learning) implementations to prove correctness. One of the main objectives was to compare K-Means with DBSCAN on different type of toy datasets to evaluate pro and cons of the respective clustering algorithm.

The environment used to implement these project comprised of python along with some essential libraries.

Linear algebra: numpy
Plotting: matplotlib
Custering algorithms and datasets: SciKit-Learn

The toy datasets used to compare the algorithms are as follows:

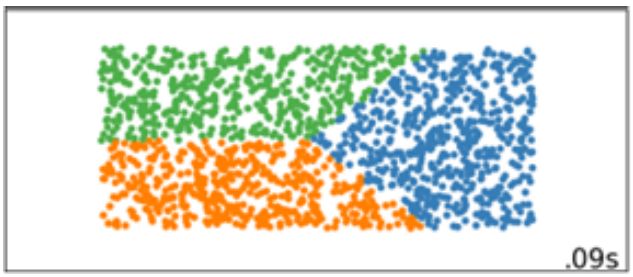


Figure 1: No structure: Randomly distributed data

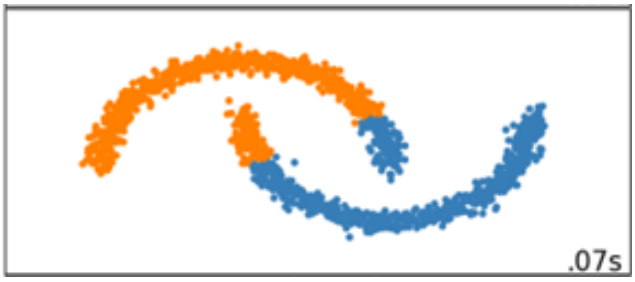


Figure 2: Noisy Moons: Irregularly or curve shaped data.

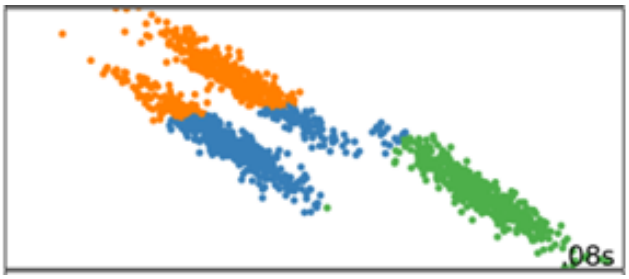


Figure 3: Anisotropic Blobs: Blobs skewed along a line

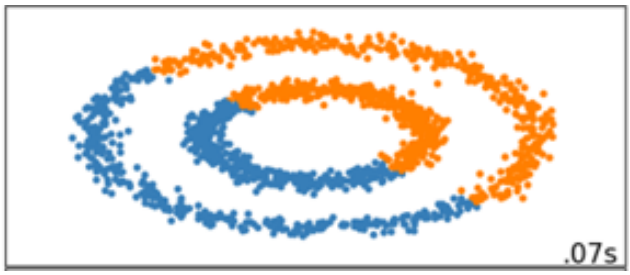


Figure 4: Noisy Circles: Irregularly or curve shaped data.

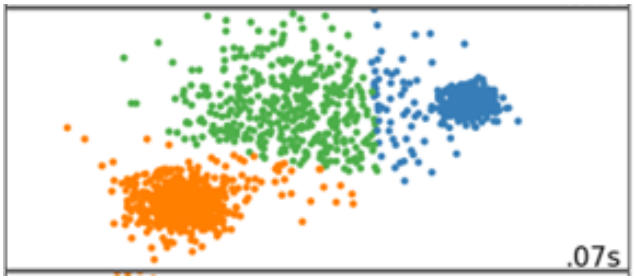


Figure 5: Blobs with varied Variances

Figure 6: Regular blobs

3. Algorithms

3.1. K-Means Clustering

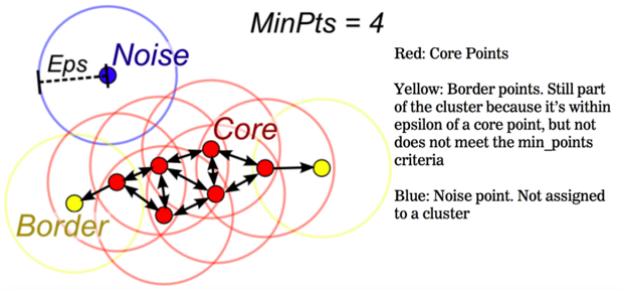
The KMeans algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares (Euclidean distance). This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.It creates a Voronoi diagram with the input K sites. Each region represents a cluster.

Algorithm 1 K-Means Cloistering Algorithm

Input: K. set of points $x_1 ... x_n$
Place centriods $c_1 ... c_k$ at random locations
while Cluster assignments are changing **do**
  for each point x_i **do**
    find the nearest centroid c_j
    assign the point x_i to cluster j
  end for
  for each cluster j in 1 to K **do**
    new centroid c_j = men of all points x_i assigned to cluster j in previous step
  end for
end while

3.2. DBSCAN

The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. The central component to the DBSCAN is the concept of core samples, which are samples that are in areas of high density. A cluster is therefore a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples). There are two parameters to the algorithm, min_samples and eps, which define formally what we mean when we say dense. Higher min_samples or lower eps indicate higher density necessary to form a cluster.



Algorithm 2 DBSCAN

DBSCAN (D, eps, MinPts)
$C \leftarrow 0$
for each unvisited point P in dataset D **do**
  mark P as visited
  $NeighborPts \leftarrow regionQuery(P, eps)$
  if sizeOf(NeighborPts) ¡ MinPts **then**
    mark P as NOISE
  else
    $C \leftarrow$ next cluster
    expandCluster (P, NeighborPts, C, eps, MinPts)
  end if
end for

expandCluster(P, NeighborPts, C, eps, MinPts)
add P to cluster C
for each point P' in NeighnorPts **do**
  if P' is not visited **then**
    mark P' as visited
    $NeigborPts' \leftarrow regionQuery(P', eps)$
    if sizeOf(NeighborPts') ¿= MinPts **then**
      $NeighborPts \leftarrow NeighborPts \cup NeighborPts'$
    end if
  end if
  if P' is not yet a member of any cluster **then**
    add P' to cluster C
  end if
end for

regionQuery(P, eps)
return all points within P's eps-neighborhood (including P)

4. Results and Conclusions

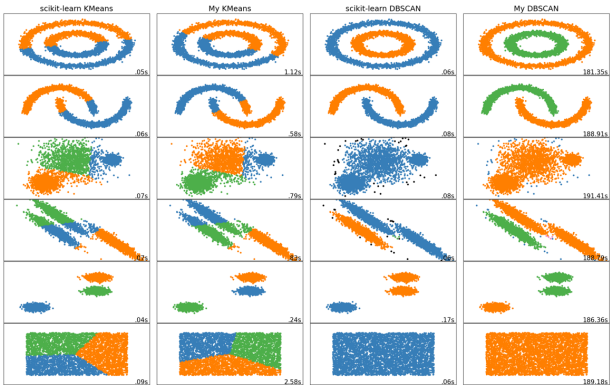


Figure 7: K-Means vs DBSCAN

1. Clusters formed by K-Means are convex in shape. Thus it performs bad on irregular shapes like the noisy spheres, and moons. On the other hand DBSCAN can form arbitrary shapes based on density, and thus performs better on the spherical and moon shaped data sets.

2. In general K-Means algorithm is more efficient for larger data sets compared to DBSCAN. This was also seen in my implementations of both the algorithms

3. K-Means algorithm is no way to detect outliers. Thus it generalizes the noisy point to the closest cluster. While in DBSCAN, outliers are not assigned to any clusters, as evident in the figure above.

4. K-Means algorithm uses euclidean/manhattan distance as a similarity metirc. Thus in the anisotropic dataset, where the similarity is not based on euclidean distance (this because the blobs are skewed and not circular), the algorithm performs poorly.

5. For sparse data, or for data with variances the DBSCAN algorithm performs very poorly, as evident from its performance on blobs with variances, and randomly distributed data. In general varying density creates problems for the algorithm. K-means performs better here since the number of cluster are mentioned prior to the cluster formation.

6. DBSCAN is also very sensitive to the chosen values for epsilon and minimum points to from a cluster. As seen in the above example, two different cluster/ blobs are considered to be on singe blob.