Data analysis : Lesson 6

Mounir Atiq

atiq.mounir@gmail.com

October 7, 2024

Clustering

Clustering assumes there are inherent classes from which observed data are drawn. It is a field of **unsupervised** learning and relies on **distance/proximity** between samples.

As real classes are unknown, most of the time clustering aims at optimizing concurrently two distance-based quantities :

- Minimizing inertia/dispersion within each clusters.
- Maximizing inertia/dispersion between different clusters.

It tends to provide data with a particular **structure** and thus play an important role in every unsupervised application of data analysis.

Chosen distance definition and data dimension have an important impact on results.

Proximity measures

A few examples of distance metrics used in clustering.

- Euclidian distance : $d(x,y) = \sqrt{\sum_{k=1}^{d} (x_k - y_k)^2} = ||x-y||_2$
- Manhattan distance : $d(x,y) = \sum_{k=1}^{d} |x_k - y_k| = ||x-y||_1$
- Minkowski distance : $d(x,y) = \left(\sum_{k=1}^{d} (x_k - y_k)^p \right)^{1/p} = ||x-y||_p$
- Cosine distance : $d(x,y) = \frac{\langle x,y \rangle}{||x||||y||} = \frac{x^T y}{||x||||y||}$
- Mahalanobis distance : $d(x,y) = \sqrt{\sum_{k=1}^{d} \frac{(x_k - y_k)^2}{\sigma_k^2}}$

Sometimes distances between sets or estimated distributions are also used.

K-means

K-means is an iterative algorithm that alternates between sample labeling according to cluster centers and cluster centers updates.

It only relies on distances to centers of mass but needs :

- Pre-determined fixed amount of clusters K.
- Initialization method for the first iteration cluster centers.

K-means initializes K cluster centers and then loops over the same steps :

- Affect each sample to the closest center.
- Recompute cluster centers as centers of mass once all data are labeled.
- Repeat until reaching centers "convergence" or maximum iteration.

Usually convergence criteria is simply defined relatively to a threshold on cluster centers variation over successive iterations.

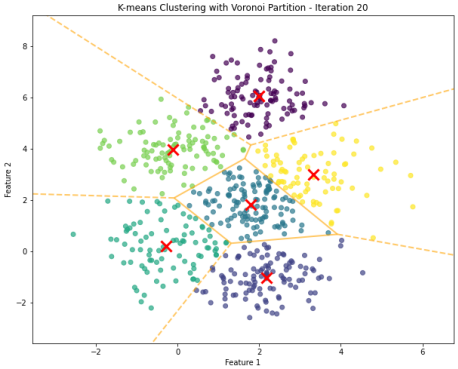


Figure 1: For given cluster centers, k-means partitioning corresponds to Voronoï diagram.

K-means

Algorithm 1 K-means Algorithm

Input: $K, (G_1, \dots, G_K)$ ▷ Number of clusters and initial cluster centers
Output: $(P_1, \dots, P_K)$ ▷ Partitioning of feature space
  while $N \leq N_{max}$ **do**
    for $1 \leq i \leq n$ **do** ▷ Affect each sample to the closest cluster
      $C_i \leftarrow \underset{k}{\arg\min}\, d(x_i, G_k)$
    end for
    for $1 \leq k \leq K$ **do** ▷ Recompute cluster centers
      $\{x_i : C_i = k\} = \{x_{i_1}, \dots, x_{i_{n_k}}\}$
      $G_k \leftarrow \frac{1}{n_k} \sum_{i=i_1}^{n_k} x_i$
    end for
  end while

Example of K-means iterations

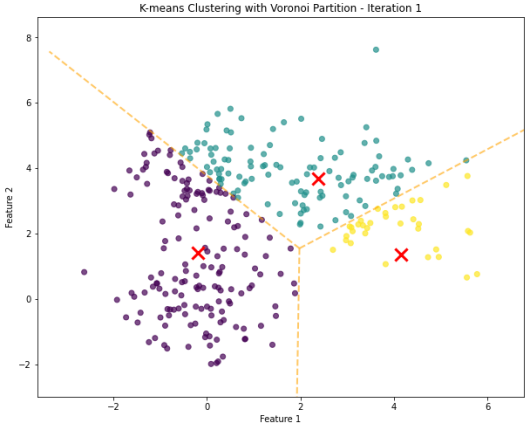


Figure 2: K-means centers convergence

Example of K-means iterations

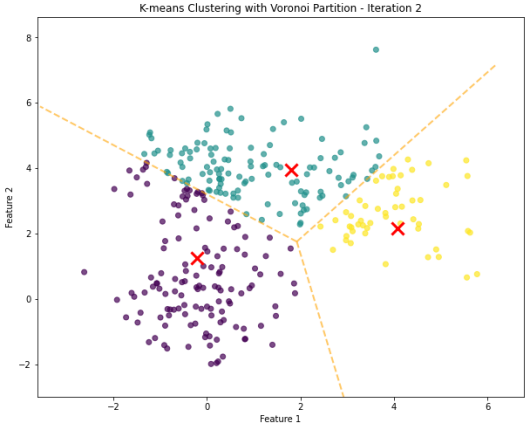


Figure 2: K-means centers convergence

Example of K-means iterations

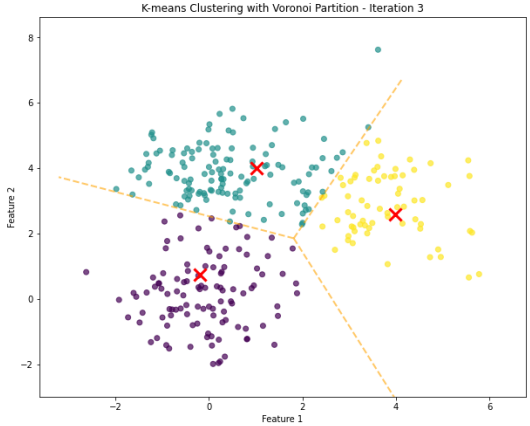


Figure 2: K-means centers convergence

Example of K-means iterations

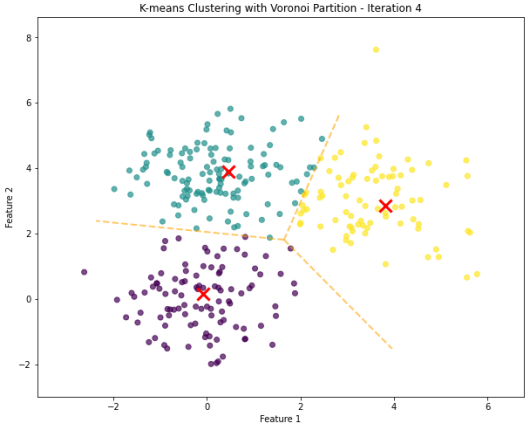


Figure 2: K-means centers convergence

Example of K-means iterations

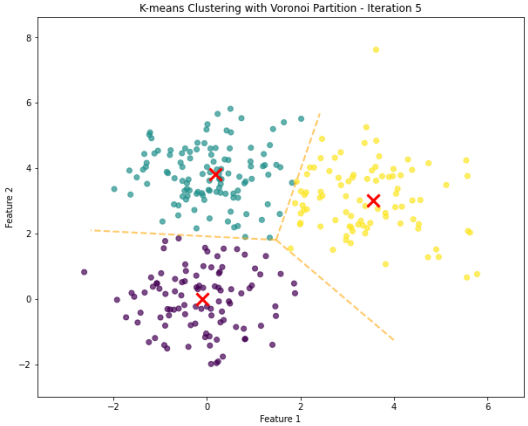


Figure 2: K-means centers convergence

Example of K-means iterations

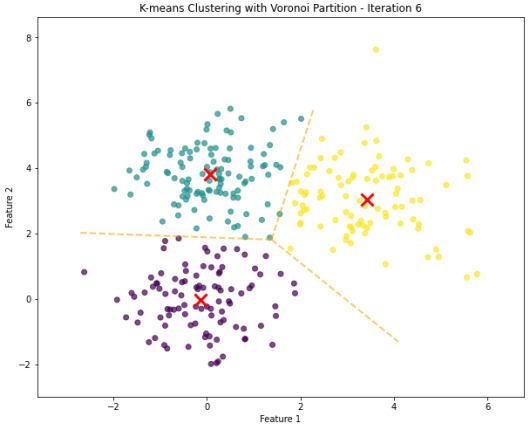


Figure 2: K-means centers convergence

Example of K-means iterations

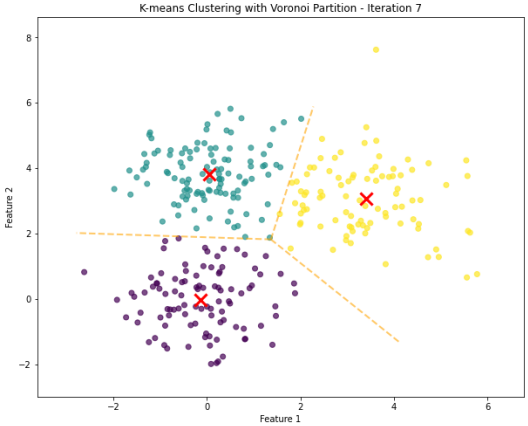


Figure 2: K-means centers convergence

Example of K-means iterations

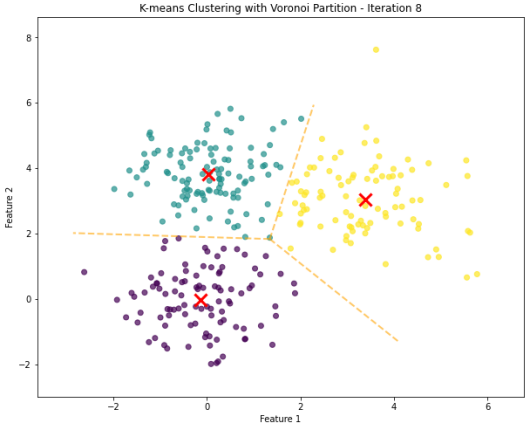


Figure 2: K-means centers convergence

Example of K-means iterations

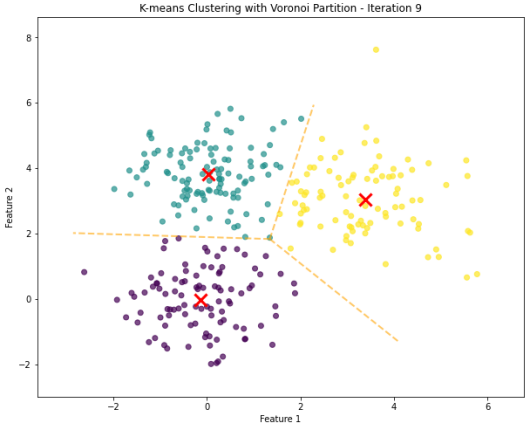


Figure 2: K-means centers convergence

Example of K-means iterations

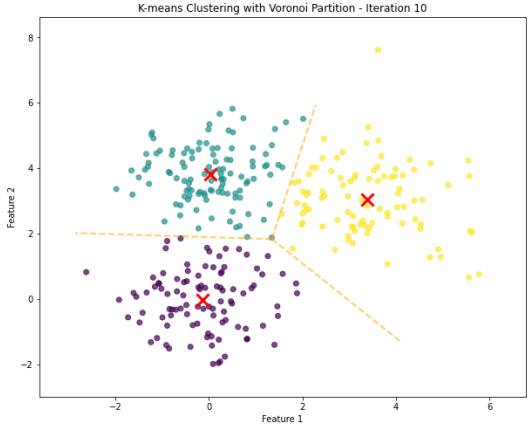


Figure 2: K-means centers convergence

Hierarchical clustering

The principle of this algorithm is start from one cluster per sample and to merge recursively clusters according to their distance until the desired amount of clusters.

It need to define distance between clusters, for that there are three common definitions :

- Minimum distance: $d(C_1, C_2) = \min_{(x_i, x_j) \in C_1 \times C_2} d(x_i, x_j)$.
- Maximum distance: $d(C_1, C_2) = \max_{(x_i, x_j) \in C_1 \times C_2} d(x_i, x_j)$.
- Average distance: $d(C_1, C_2) = \frac{1}{n_1 n_2} \sum_{(x_i, x_j) \in C_1 \times C_2} d(x_i, x_j)$.
- Ward distance : $d(C_1, C_2) = \frac{n_1 n_2}{n_1 + n_2} d(G_1, G_2)$.

If a desired amount of cluster K is provided, the algorithm just stop merging while reaching this number.

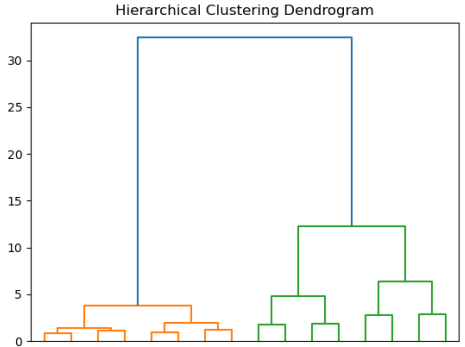


Figure 3: Dendogram of hierarchical clustering on Iris dataset.

DBSCAN

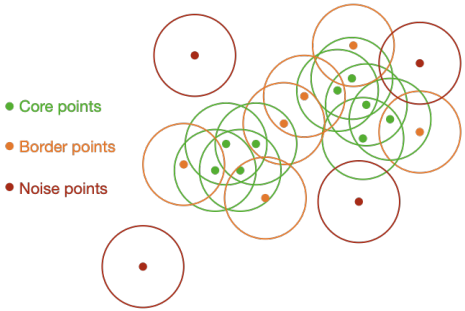
Contrary to k-means, DBSCAN does not need to specify a number of cluster or to initialize cluster centers.

It relies on two notions :

- **Neighborhoods :**
  $V_\epsilon(x_i) = B_d(x_i, \epsilon) = \{x_j : \quad d(x_i, x_j) < \epsilon\}$
- **Connectivity degree :** $\text{Card}(V_\epsilon(x_i))$

DBSCAN algorithm runs through all samples assessing these two notions :

- It depends on two parameters : neighborhood size ϵ and a minimum connectivity degree C_{min}.
- It redefines samples into three categories : **core** points, **border** points and **noise** points



DBSCAN

Algorithm 2 DBSCAN Algorithm

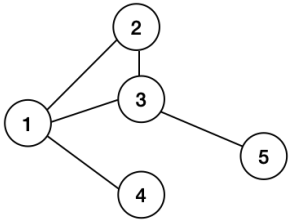
Input: ϵ, C_{min} ▷ Neighborhood size and minimum connectivity degree
Output: $y = x^n$
  $K \leftarrow 0$
  $P_{core} \leftarrow \{\}, P_{border} \leftarrow \{\}, P_{noise} \leftarrow \{\}$
  while $i \leq n$ **do** ▷ Detect core points and affect them to a cluster
    if $\mathrm{Card}(V_\epsilon(x_i)) \geq C_{\min}$ **then**
      $P_{core} \leftarrow \mathrm{append}(x_i)$ ▷ x_i is a **core** point
      if $\exists x_j : x_i \in V_\epsilon(x_j)$ **then**
        $C_i \leftarrow C_j$
      else ▷ New cluster
        $K \leftarrow K + 1, \quad C_i \leftarrow K$
      end if
    end if
  end while
  for $1 \leq i \leq n$ **do** ▷ Detect border and noise points
    if $\exists x_j \in P_{core} : x_i \in V_\epsilon(x_j)$ **then**
      $P_{border} \leftarrow \mathrm{append}(x_i), \quad C_i \leftarrow C_j$
    else
      $P_{noise} \leftarrow \mathrm{append}(x_i)$
    end if
  end for

Graph definition

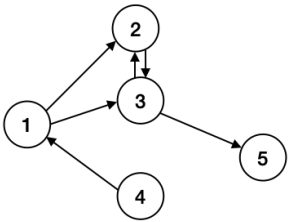
Graphs

A graph is a couple of vertices (or nodes) and edges $G = (V, E)$,

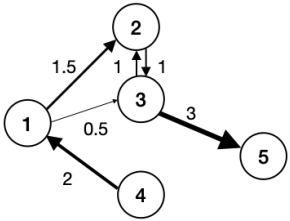
- Each edge $e = (v_i, v_j) \in V^2$ is a connection between two nodes.
- If the graph is **directed**, nodes order matters : $e = (v_i, v_j)$ means "from node v_i to node v_j".
- Graphs can be **weighted** : each edge e_k is associated with a weight w_k



(a) Undirected graph



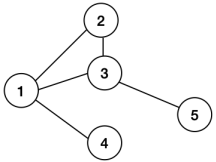
(b) Directed graph



(c) Weighted directed graph

Similarity and Laplacian matrices

Similarity and Laplacian matrices fully describe graphs.

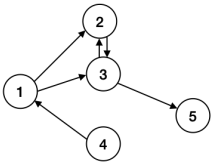


Similarity matrix :

$$S = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Degree matrix :

$$D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

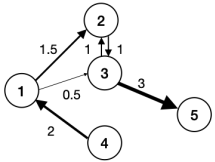


Similarity matrix :

$$S = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Degree matrix :

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



Similarity matrix :

$$S = \begin{pmatrix} 0 & 1.5 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 3 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Degree matrix :

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- **Laplacian** matrix : $L = D - S$, **Normalized Laplacian** matrix : $L = I - D^{-1/2} S D^{-1/2}$

They are symmetric for **undirected** graphs but not in general for **directed** graphs.
Columns of Laplacian matrix sum up to zero.

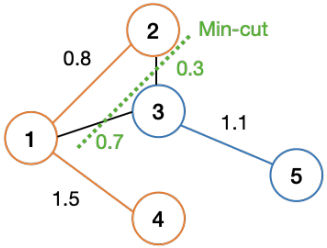
Similarity and Laplacian matrices

Spectral clustering uses spectral decomposition of undirected graph Laplacian matrices.

- First eigenvalue of Laplacian matrix is always zero.
- Multiplicity of zero eigenvalue corresponds to the number of disjoint subgraphs.

Graph min-cut edge problem

- Cut partitioning assessment : $\text{cut}(P_1, P_2) = \sum_{(i,j) \in P_1 \times P_2} w_{ij}$
- Notion of "volume" : $\text{Vol}(P) = \sum_{v_i \in P} \deg(v_i)$
- **Normalized cut** : $\text{cut}^{(\text{norm})}(P_1, P_2) = \text{cut}(P_1, P_2)\left(\frac{1}{\text{Vol}(P_1)} + \frac{1}{\text{Vol}(P_2)}\right)$



Clustering on eigenvectors corresponding to lowest eigenvalues is related to this partitioning problem.

Spectral clustering

Any data can be modeled as a graph between samples.

Spectral clustering steps

1. Compute Laplacian matrix L
2. Perform spectral decomposition of symmetric matrix L
3. Consider the sub-matrix of K first eigenvectors
4. Apply **any clustering method** on rows of this sub-matrix
5. Affect graph nodes according to row clusters

We need first to define a **similarity** matrix either from sample distances or directly from data.
Common similarity measures : $S_{ij} = \widehat{\sigma}_{ij}$ or $S_{ij} = \exp\left(-\frac{d^2(x_i,x_j)}{\sigma^2}\right)$.
Then, a graph can be built from S :

- Fully connected
- Keeping edges only on nearest neighbors
- Keeping edges only if on distances lower than a threshold
- Keeping only a maximum degree for each node

Spectral clustering example on graph

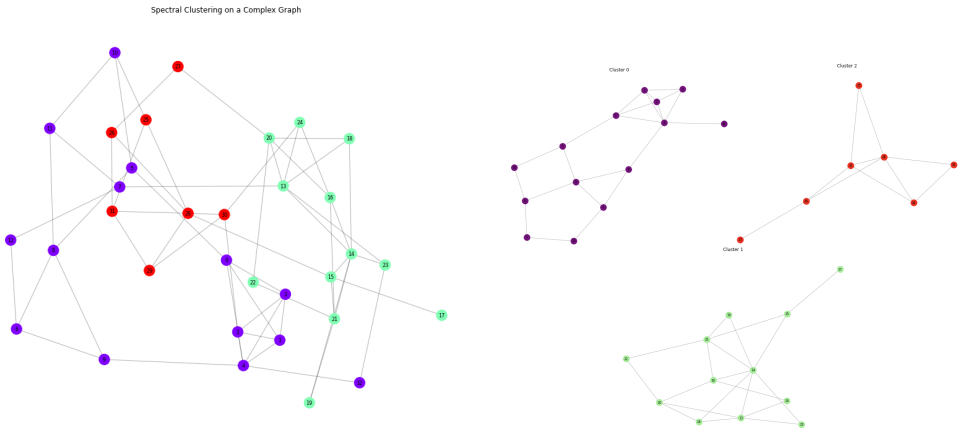


Figure 5: Example of graph spectral clustering

Clustering algorithms comparison / Iris dataset

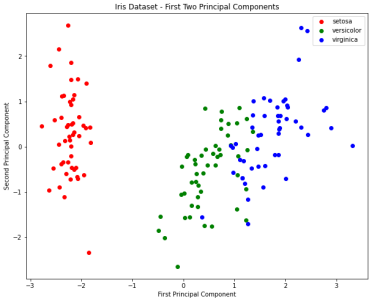


Figure 6: Original classes of Iris dataset

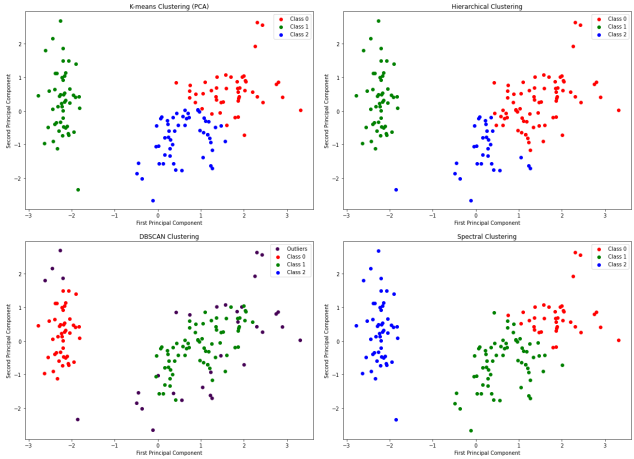


Figure 7: Clustering results on Iris dataset

Clustering algorithms comparison / Synthetic dataset

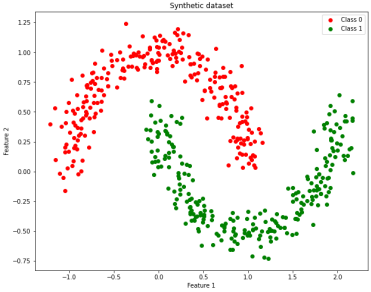


Figure 8: Original classes of Synthetic dataset

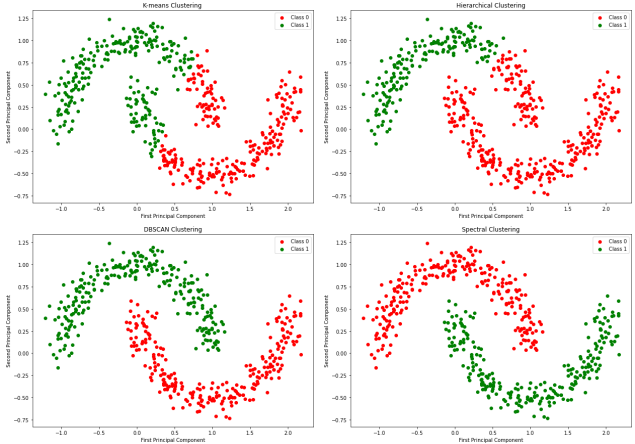


Figure 9: Clustering results on Synthetic dataset