Business Club

Clustering

Business Club Analytics Team

July 2020

Introduction

Unsupervised Learning

Unsupervised learning refers to the use of artificial intelligence (AI) algorithms to identify patterns in data sets containing data points that are neither classified nor labeled. Unsupervised machine learning methods cannot be directly applied to a regression or classification problem, but can be instead used to discover the underlying structure of data.

In unsupervised learning, unsorted information is grouped according to the similarities between the given data points without any prior training. There’s no reference data at all.The unsupervised algorithm works with unlabeled data. Its purpose is exploration. If supervised machine learning works under clearly defined rules, unsupervised learning is working under the conditions of results being unknown and thus needed to be defined in the process.

Unlike in supervised, the ‘performance’ of unsupervised machine learning algorithms is not easy to be estimated.

The main applications of unsupervised learning include clustering, visualization, dimensionality reduction, finding association rules, and anomaly detection.

Clustering

Clustering is a type of unsupervised machine learning algorithm in which similar data points in a given dataset are grouped together. This essentially means that the data points with similar traits are in the same group. Elements in a cluster should be as similar as possible, and points in different groups should be as dissimilar as possible. Clustering is used for analyzing and grouping data, which does not include pre-labeled classes or class attributes. Clustering can be helpful for businesses to manage their data in a better way

Example: Detecting customer shopping patterns and grouping them accordingly.

Types of Clustering

- **Centroid-based Clustering** - The data is organised into non- hierarchical clusters and cluster centers (centroids) are assigned in such a way that the distance of a data point is minimum with the center. Eg. K-means clustering.

- **Connectivity-based Clustering** - Clusters are formed according to closeness of data points and the data points which are closer have similar characteristics to each other. Eg. Hierarchical clustering.

- **Density-based Clustering** - Density-based clustering locates regions of high density that are separated from one another by regions of low density.

- **Subspace Clustering** - It seeks to find clusters in different subspaces within the dataset. It localizes the search for relevant dimensions allowing them to find clusters that exist in multiple overlapping subspaces.

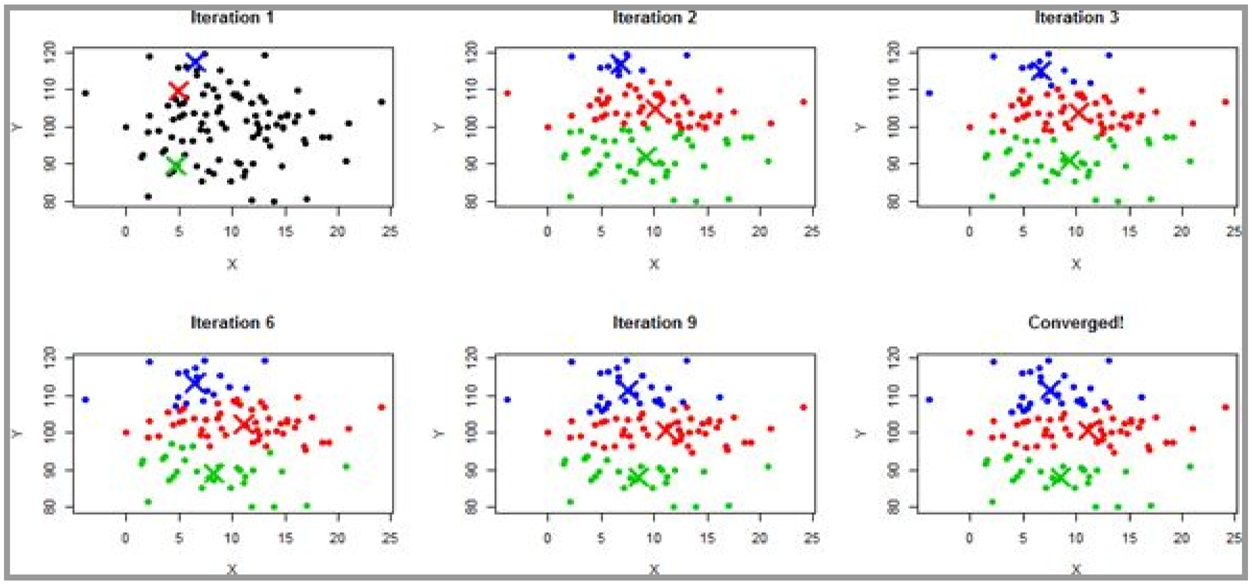
K-Means Clustering

K-means is a type of centroid-based clustering. It follows a procedure in which the no. of centroids (k) are specified, one for each cluster.
Different locations of centroids would give different results, so centroids should be preferably placed away from each other which would help in more distinction. Next, the data points are assigned to a cluster whose centroid is the nearest to them. The centroids are then reassigned and this process is repeated to get optimal clusters

Pseudo Code :

```
1. k = no. of clusters
2. place the centroids $c_1$ , $c_2$, .. $c_i$ randomly
3. for each data point $x_i$
        - find nearest centroid ($c_1$ , $c_2$, .. $c_i$ )
        - assign it to that cluster
4. for each new cluster, new centroid = mean of all points
5. repeat steps 3 and 4 until optimality
```

However, for better results, we first have to find the optimal no. of clusters, i.e. 'k'.



Time Complexity: The k-means algorithm is known to have a time complexity of O(n^2), where n is the input data size. This quadratic complexity debars the algorithm from being effectively used in large applications.

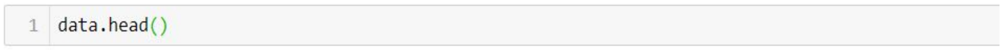
Determining the **optimal number of clusters** in a data set is a fundamental issue in k-means clustering, which requires the user to specify the number of clusters k to be generated. Some of the methods to determine the optimal no. of clusters are :-

Elbow Method

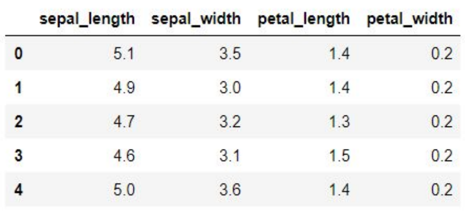
The **Elbow method** basically plots the graph of ‘sum of squared distances from centroid v/s no. of clusters’ and the point (no. of clusters) at which the distance stops getting reduced sharply is the point of optimality. If the line graph looks like an arm - a red circle in the below line graph, the "elbow" on the arm is the value of optimal k (number of the cluster). K-means is used to minimize SSE (sum of squared errors). SSE tends to decrease toward 0 as we increase k and SSE is 0 when k is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster. Therefore, we have to choose k in such a way that the error significantly reduces at that particular value of k.

Therefore, the ‘elbow’ is the point where the distortion declines the most.

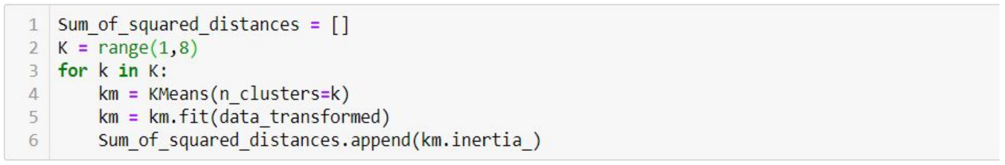
Here is a demonstration of the use of the elbow method for the ‘IRIS’ dataset.



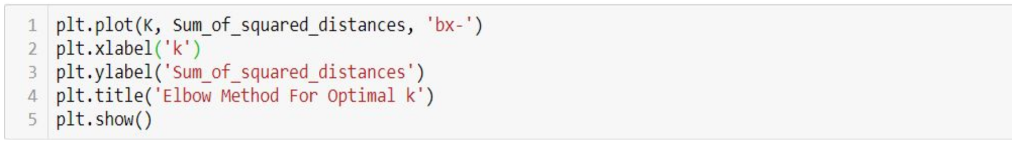
```
data.head()
```



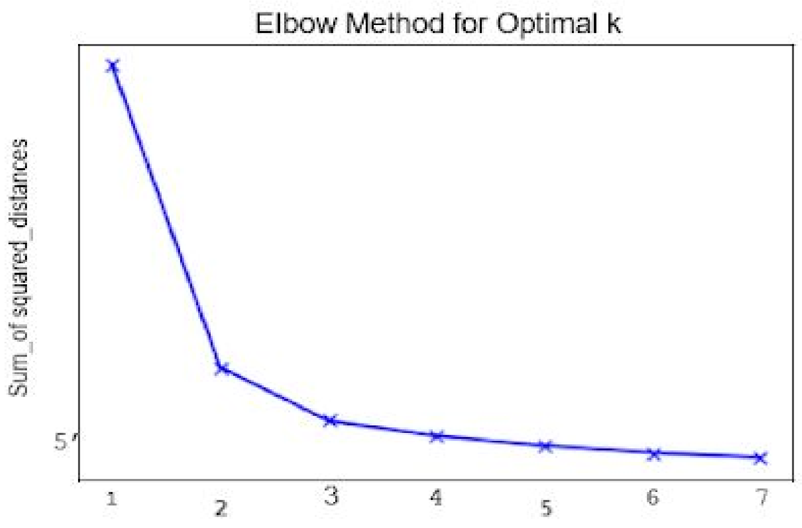
	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2



```
Sum_of_squared_distances = []
K = range(1,8)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(data_transformed)
    Sum_of_squared_distances.append(km.inertia_)
```



```
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



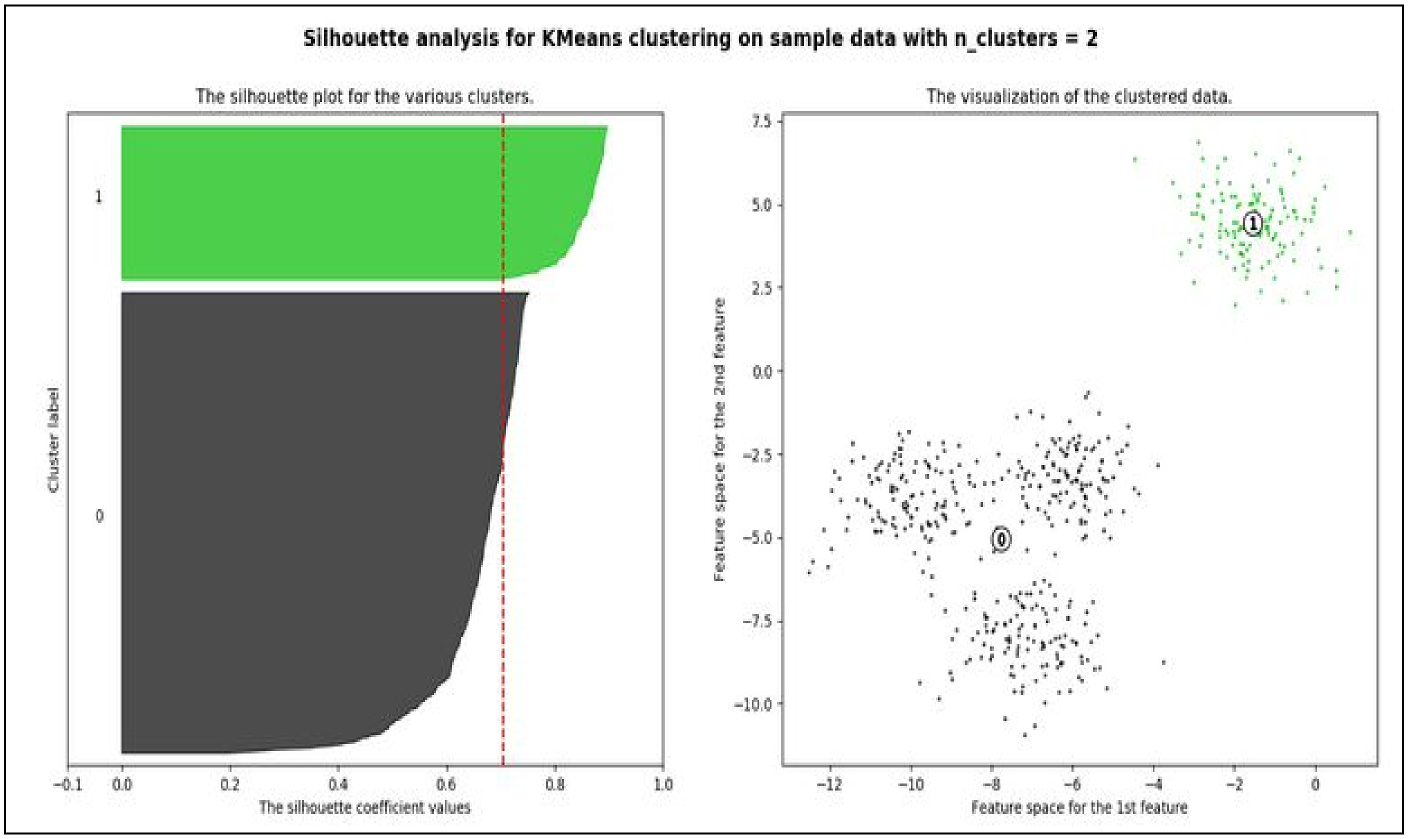
Silhouette

Silhouette can be applied to evaluate the validity of k-means clustering with different k values and select the best result. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighbouring clusters and thus provides a way to assess parameters like number of clusters visually. This measure has a range of [-1, 1].

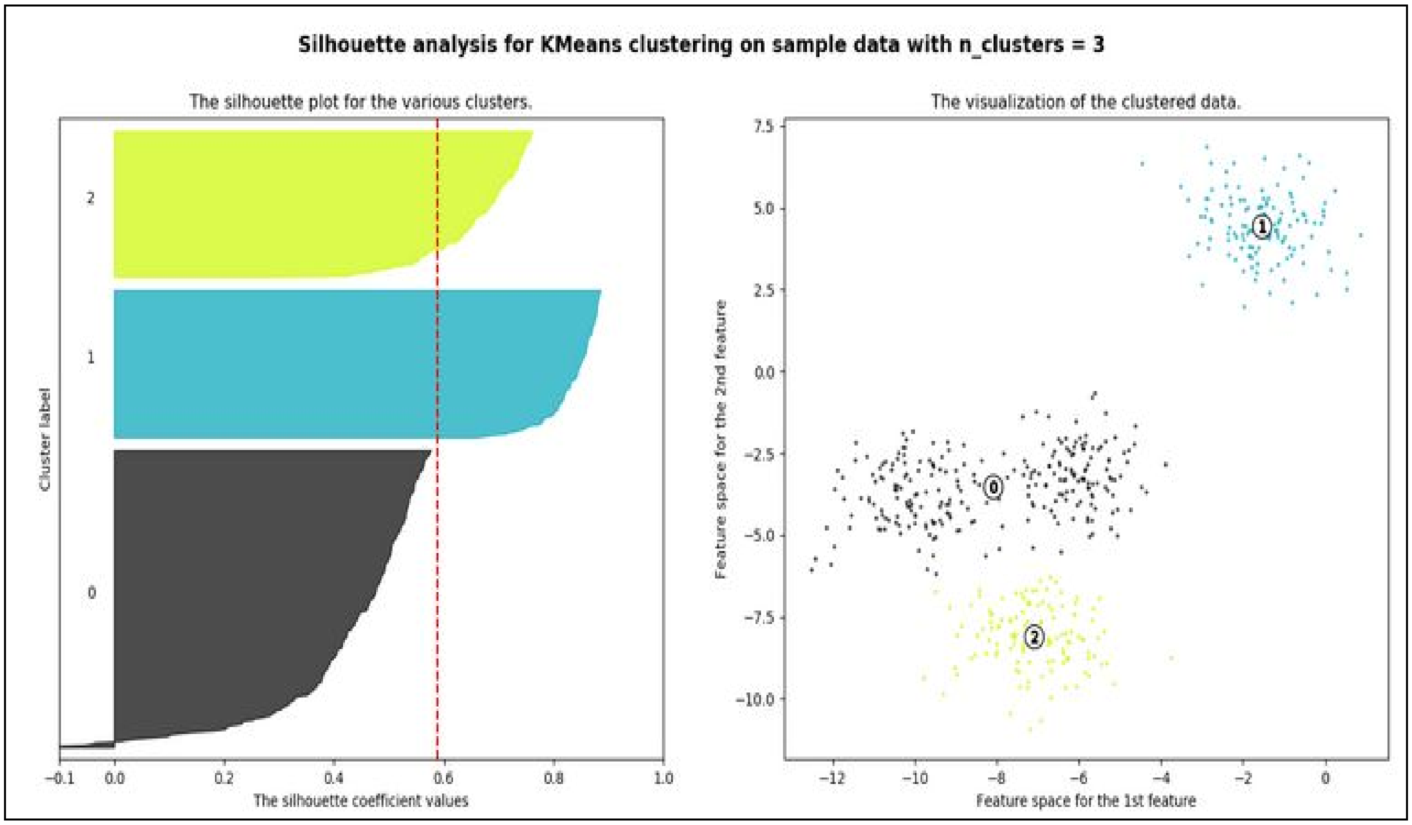
Silhouette coefficients near +1 indicate that the sample is far away from the neighbouring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighbouring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

In the example on the next page, the silhouette analysis is used to choose an optimal value for $n_{clusters}$. The silhouette plot shows that the $n_{clusters}$ values of 3 and 5 are a bad pick for the given data due to the presence of clusters with below average silhouette scores and also due to wide fluctuations in the size of the silhouette plots. Silhouette analysis is more ambivalent in deciding between 2 and 4.

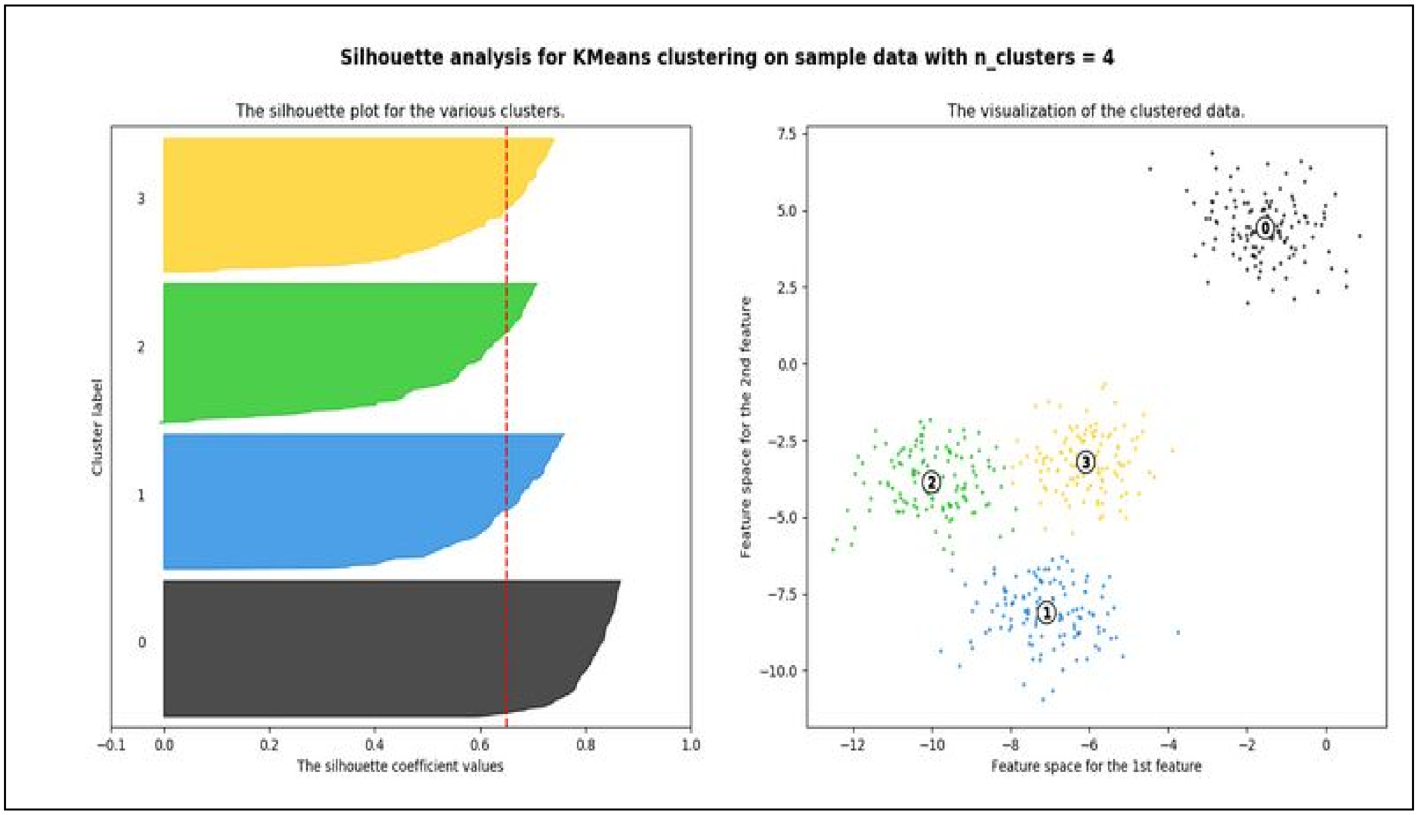
Also, from the thickness of the silhouette plot the cluster size can be visualized. The silhouette plot for cluster 0 when $n_{clusters}$ is equal to 2, is bigger in size owing to the grouping of the 3 sub clusters into one big cluster. However, when the $n_{clusters}$ is equal to 4, all the plots are more or less of similar thickness and hence are of similar sizes as can also be verified from the labelled scatter plot on the right.



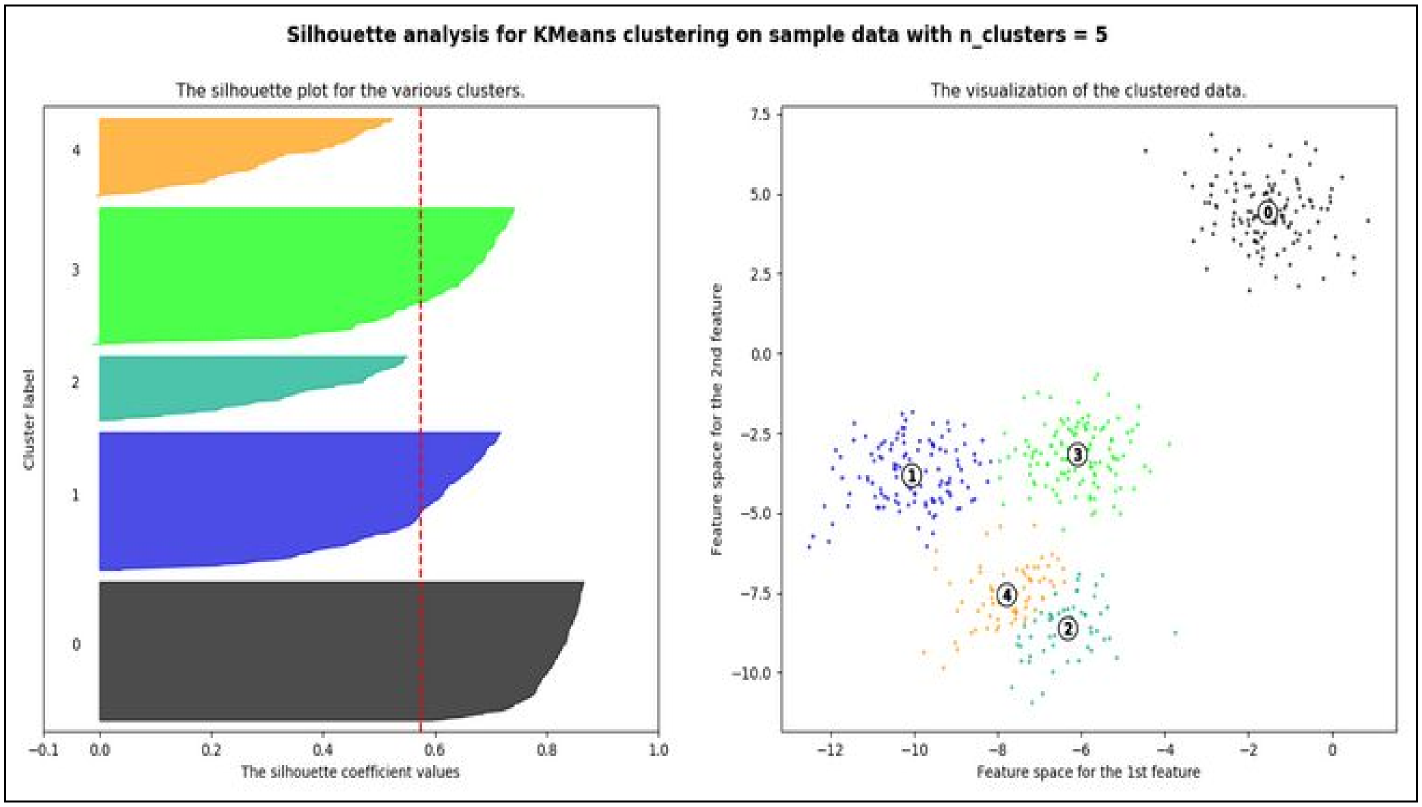
Silhouette analysis for KMeans clustering on sample data with n_clusters = 2
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 1st feature
Feature space for the 2nd feature



Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 1st feature
Feature space for the 2nd feature



Silhouette analysis for KMeans clustering on sample data with n_clusters = 4
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 1st feature
Feature space for the 2nd feature



Silhouette analysis for KMeans clustering on sample data with n_clusters = 5
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 1st feature
Feature space for the 2nd feature

Mathematical Analysis

Let X = {X_1, X_2, ..., X_n} be a set of n data points. X_i($1 \le i \le n$) is one of the data points, which can be represented as [$x_{i,1}$, $x_{i,2}$, ..., $x_{i,m}$], where m is the number of features. Given the set of data points X, an integer k($2 \le k \le n$) and k initial centroids in the domain of X, the k-means algorithm aims to find a clustering of X into k clusters such that it minimises the k-means Cost Function, which is defined as as follows:

$$CF(X,C) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{i,l} d_E(X_i, C_l)$$

where dE(·,·) is the squared Euclidean distance, C={C_1, C_2, ..., C_k}, which is a set of cluster centroids after clustering, and $w_{i,l}$ is the indicator function, which equals to 1 when X_i is in C_l and 0 when X_i is not in C_l. For a single data point X_i, its Silhouette value sil(i) is calculated as:

$$sil(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

where a(i) is the distance of X_i to its own cluster, which is defined as:

$$a(i) = \frac{\sum_{\substack{p=1 \\ p \neq i}}^{n} w_{p,h} d_E(X_i, X_p)}{n_h - 1}$$

where n_h is the number of data points in the cluster h. b(i) is the distance of X_i to its closest neighbouring cluster, which is defined as:

$$b(i) = \underset{l \neq h}{minimum} \frac{\sum_{p=1}^{n} w_{p,l} d_E(X_i, X_p)}{n_l}$$

The sil(i) ranges from −1 to 1. When a(i) is much smaller than b(i), the sil(i) is close to 1 to show this data point is well clustered. In the opposite way, the sil(i) is close to −1 to show it is badly clustered.

The Silhouette value of a whole cluster or a full clustering is defined as the average value of sil(i) across all the data involved:

$$Sil = \frac{1}{n}\sum_{i=1}^{n} sil(i)$$

Therefore, the Silhouette value for a full clustering Sil also ranges from −1, which shows a very bad clustering, to 1, which shows a perfect clustering.

Hierarchical Clustering

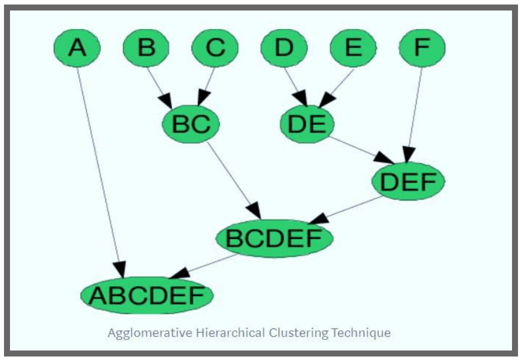
Hierarchical clustering constructs a (usually binary) tree over the data. The leaves are individual data items, while the root is a single cluster that contains all of the data. Between the root and the leaves are intermediate clusters that contain subsets of the data. The main idea of hierarchical clustering is to make “clusters of clusters” going upwards to construct a tree.

There are two main conceptual approaches to forming such a tree. Hierarchical agglomerative clustering (HAC) starts at the bottom, with every datum in its own singleton cluster, and merges groups together. Divisive clustering starts with all of the data in one big group and then chops it up until every datum is in its own singleton group.

Agglomerative Clustering

Basic Agglomerative Clustering Algorithm

```
1. Compute the proximity matrix, if necessary
2. **repeat** {
        a. Merge the closest two clusters
        b. Update the proximity matrix to reflect the proximity between the new
           cluster and the original clusters }
3. **until** Only one cluster remains
```

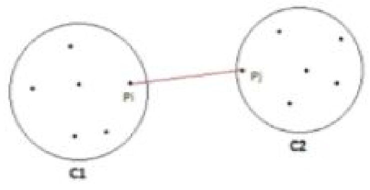


Agglomerative Hierarchical Clustering Technique

Defining Proximity between Clusters

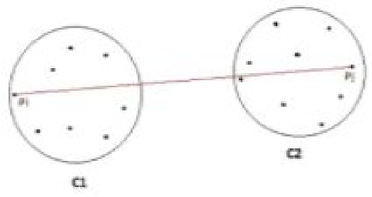
Single Link or Min

Here, the proximity of two clusters is defined as the minimum of the distance between any two points in the two different clusters. The single link technique is good at handling non-elliptical shapes, but is sensitive to noise and outliers.



Complete Link or MAX or CLIQUE

For the complete link or MAX version of hierarchical clustering, the proximity of two clusters is defined as the maximum of the distance between any two points in the two different clusters. Complete link is less susceptible to noise and outliers, but it can break large clusters and it favours globular shapes.



Group Average

For the group average version of hierarchical clustering, the proximity of two points is defined as the average pairwise proximity among all pairs of points in the different clusters. Thus, for group average, the cluster proximity proximity(C_i , C_j) of clusters C_i and C_j, which are of size m_i and m_j, respectively, is expressed by the following equation:

$$proximity(C_i, C_j) = \frac{\sum_{\substack{\mathbf{x} \in C_i \\ \mathbf{y} \in C_j}} proximity(\mathbf{x}, \mathbf{y})}{m_i * m_j}.$$

Ward's Method and Centroid Methods

For Ward's method, the proximity between two clusters is defined as the increase in the squared error that results when two clusters are merged. Thus, this method uses the same objective function as K-means clustering.

Centroid methods calculate the proximity between two clusters by calculating the distance between the centroids of clusters.

The Lance-Williams Formula for Cluster Proximity

Any of the cluster proximities that we have discussed can be viewed as a choice of different parameters, in the Lance-Williams formula, for the proximity between clusters Q and R, where R is formed by merging clusters A and B. In this equation, p(., .) is a proximity function, while m_A, m_B, and m_Q are the number of points in clusters A, B, and Q, respectively. In other words, after we merge clusters A and B to form cluster R, the proximity of the new cluster, R, to an existing cluster, Q, is a linear function of the proximities of Q with respect to the original clusters A and B.

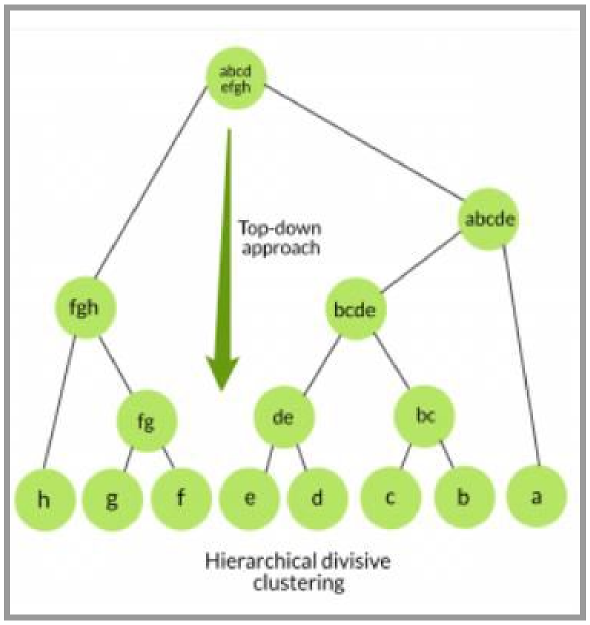
$$p(R,Q) = \alpha_A \, p(A,Q) + \alpha_B \, p(B,Q) + \beta \, p(A,B) + \gamma \, |p(A,Q) - p(B,Q)|$$

Any hierarchical clustering technique that can be expressed using the Lance-Williams formula does not need to keep the original data points. Instead, the proximity matrix is updated as clustering occurs.

Clustering Method	$\alpha_\mathbf{A}$	$\alpha_\mathbf{B}$	β	γ
Single Link	$1/2$	$1/2$	0	$-1/2$
Complete Link	$1/2$	$1/2$	0	$1/2$
Group Average	$\frac{m_A}{m_A+m_B}$	$\frac{m_B}{m_A+m_B}$	0	0
Centroid	$\frac{m_A}{m_A+m_B}$	$\frac{m_B}{m_A+m_B}$	$\frac{-m_A m_B}{(m_A+m_B)^2}$	0
Ward's	$\frac{m_A+m_Q}{m_A+m_B+m_Q}$	$\frac{m_B+m_Q}{m_A+m_B+m_Q}$	$\frac{-m_Q}{m_A+m_B+m_Q}$	0

Divisive Clustering

Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. In this case, we need to decide which cluster to split at each step and how to do the splitting.



Basic Divisive Clustering Algorithm

Given a dataset, at the top we have all the data in one single cluster. The cluster is split using some flat clustering method such as K-Means.

```
1. repeat {
    a. Choose the best cluster among all the clusters to split
    b. Split that cluster by the flat clustering algorithm }
2. until Each data is in its own singleton cluster
```

Fuzzy C-Means

Clustering can be either hard or fuzzy type. In the first category, the patterns are distinguished in a well-defined cluster boundary region. But due to the overlapping nature of the cluster boundaries, some classes of patterns may be specified in a single cluster group or dissimilar group. To reduce such limitations fuzzy type clustering came into the picture. Fuzzy clustering problems can be grouped into three branches:

(a) Based on fuzzy relation;

(b) Based on fuzzy rule learning; and

(c) Based on optimization of an objective function.

The fuzzy clustering based on the objective function is quite popularly known to be Fuzzy c-means clustering (FCM). In FCM method, the pattern may belong to all the cluster classes with a certain fuzzy membership degree.

Structure

The FCM approach uses a fuzzy membership. There are three basic operators in the FCM method: the fuzzy membership function, partition matrix and the objective function.

Let us consider a set of n vectors ($X = (x_1,x_2,\ldots x_n)$ $2<=c<=n$) for clustering into c groups. Each vector x_i є R^s is described by s real valued measurements which represent the features of the object x_i. A membership matrix known as Fuzzy partition matrix is used to describe the fuzzy membership. The set of fuzzy partition matrices (cxn) is denoted by M_{fc} and is defined as:

$$M_{fc} = \{W \in R^{cn} | w_{ik} \in [0,1], \forall i,k; \sum_{i=1}^{c} w_{ik} = 1, \forall k; 0 < \sum_{k=1}^{n} w_{ik} < n, \forall i\}$$

where $1 \le i \le c$, $1 \le k \le n$.

The objective function of the fuzzy c-means algorithm is computed by using membership value and Euclidean distance.

$$J_m(W,P) = \sum_{\substack{1 \le k \le n \\ 0 \le i \le c}} (w_{ik})^m (d_{ik})^2$$

$$(d_{ik}) = ||x_k - p_i||$$

where m є (1,∞) is the parameter which defines the fuzziness of the resulting clusters and d_{ik} is the Euclidian distance from object x_k to the cluster center p_i.

The minimization of the objective function J_m through FCM algorithm happens by iterative updation of the partition matrix using the following two equations:

$$p_i = \sum_{k=1}^{n} (w_{ik})^m x_k / \sum_{k=1}^{n} (w_{ik})^m$$

$$w_{ik}^{(b)} = \sum_{j=1}^{c} 1/[(d_{ik}^{(b)}/d_{jk}^{(b)})^{2/m-1}]$$

The FCM membership function is calculated as:

$$\mu_{i,j} = \left[\sum_{t=1}^{c} \left(\frac{\|x_j - v_i\|_A}{\|x_j - v_t\|_A} \right)^{\frac{2}{m-1}} \right]^{-1}$$

$\mu_{i,j}$ is the membership value of jth sample and ith cluster. The number of clusters is represented by c. x_j is the jth sample and v_i cluster center of the ith cluster. $|| \, ||_A$ represents the norm function.

Algorithm

1. **Initialize** the number of clusters c.
2. Select an inner product metric Euclidean norm and the weighting metric (fuzziness).
3. **Initialize** the cluster prototype $P^{(0)}$, iterative counter $b = 0$.
4. Then calculate the partition matrix $W^{(0)}$.
5. Update the fuzzy cluster centers $P^{(b+1)}$.
6. If $||P^{(b)}-P^{(b+1)}|| < \varepsilon$ then stop, otherwise **repeat** step **2** through **4**.

DBSCAN

DBSCAN stands for **Density Based Spatial Clustering of Applications with Noise**. Density-based clustering locates regions of high density that are separated from one another by regions of low density. In the center-based approach of defining density, density is estimated for a particular point in the data set by counting the number of points within a specified radius, ϵ, of that point. This includes the point itself. This is called the epsilon neighbourhood of the point which is defined as follows:

$$N(p) = \{q \in D \mid dist(p, q) \leq \epsilon\}.$$

This method is simple to implement, but the density of any point will depend on the specified radius. For instance, if the radius is large enough, then all points will have a density of m, the number of points in the data set. Likewise, if the radius is too small, then all points will have a density of 1.

Classification of Points

Core Points

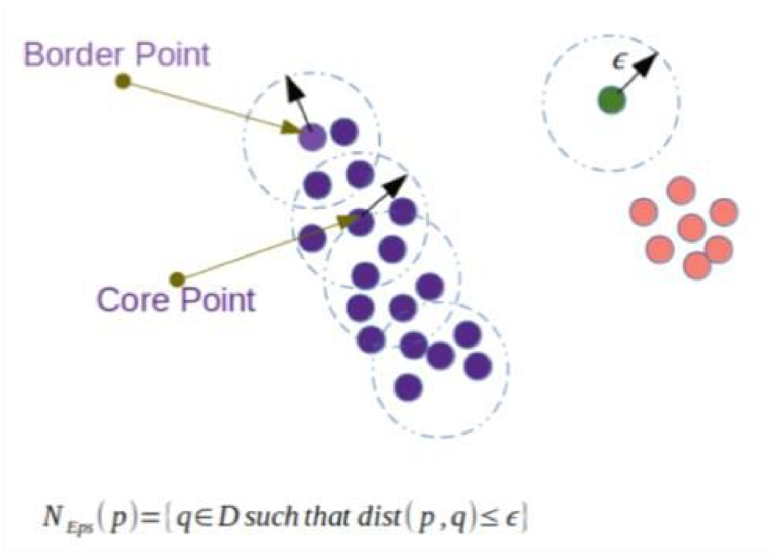
These points are in the interior of a density-based cluster. A point is a core point if the number of points within a given neighbourhood around the point as determined by the distance function and a user-specified distance parameter, ϵ, exceeds a certain threshold, MinPts, which is also a user-specified parameter.

Border Points

A border point is not a core point, but falls within the neighbourhood of a core point.

Noise Points

A noise point is any point that is neither a core point nor a border point.


Border Point
Core Point
ϵ
$N_{Eps}(p) = \{q \in D \text{ such that } dist(p,q) \le \epsilon\}$

The DBSCAN Algorithm

Given the previous definitions of core points, border points, and noise points, the DBSCAN algorithm can be informally described as follows.

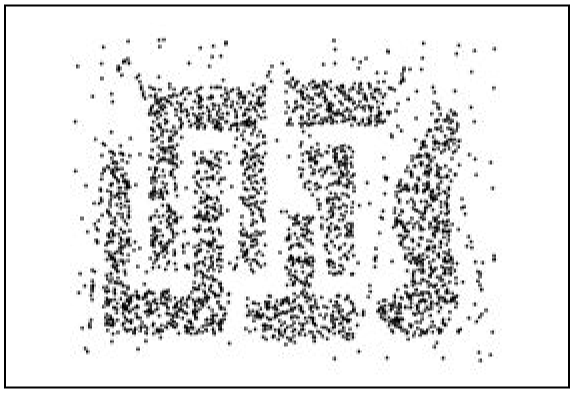
1. Label all points as core, border or noise points.
2. Eliminate noise points.
3. Put an edge between all core points that are within ϵ distance of each other.
4. Make each group of connected core points into a separate cluster.
5. Assign each border point to one of the clusters of its associated core points.

Selection of DBSCAN Parameters

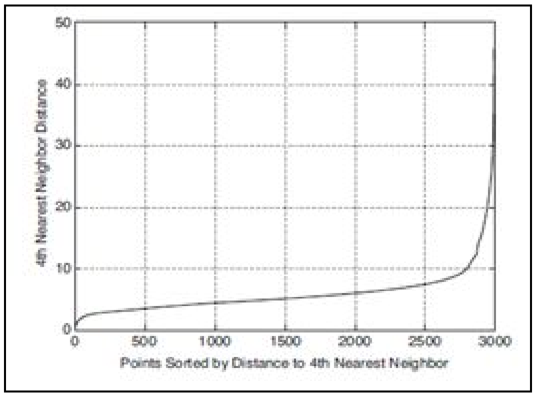
There is the issue of how to determine the parameters ϵ and MinPts. The basic approach is to look at the behaviour of the distance from a point to its kth nearest neighbour, which we will call the k-dist. For points that belong to some cluster, the value of k-dist will be small if k is not larger than the cluster size.

Therefore, if we compute the k-dist for all the data points for some k, sort them in increasing order, and then plot the sorted values, we expect to see a sharp change at the value of k-dist that corresponds to a suitable value of ϵ. If we select this distance as the ϵ parameter and take the value of k as the MinPts parameter, then points for which k-dist is less than ϵ will be labelled as core points, while other points will be labelled as noise or border points.

Strengths and Weaknesses



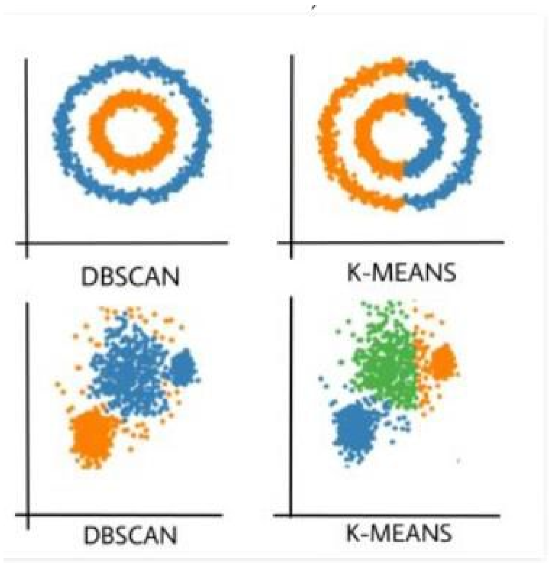
Sample Data



K-Dist plot

Because DBSCAN uses a density-based definition of a cluster, it is relatively resistant to noise and can handle clusters of arbitrary shapes and sizes. Thus, DBSCAN can find many clusters that could not be found using K-means.

However, DBSCAN has trouble when the clusters have widely varying densities. Also, DBSCAN can be expensive when the computation of nearest neighbours requires computing all pairwise proximities, as is usually the case for high-dimensional data.


DBSCAN
K-MEANS
DBSCAN
K-MEANS