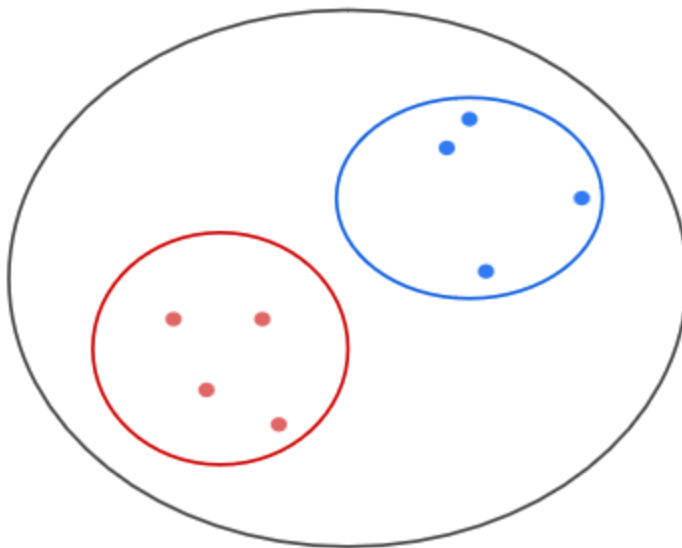


## Lesson20 Clustering (UnSupervised Learning)

Last Update June 1, 2021

Clustering known as **unsupervised learning**, that explores input data without being given an explicit output variable. An example is exploring customer demographic data to identify patterns.

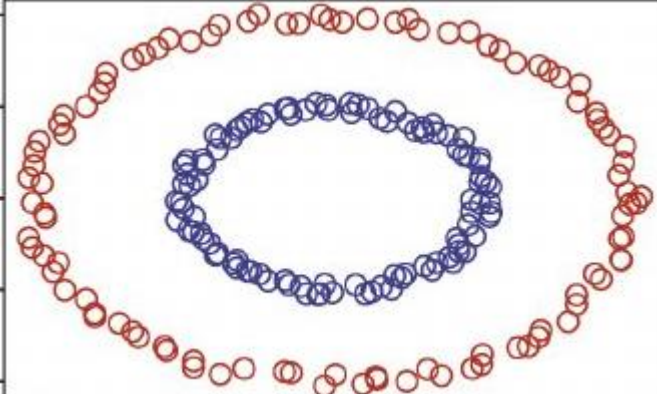
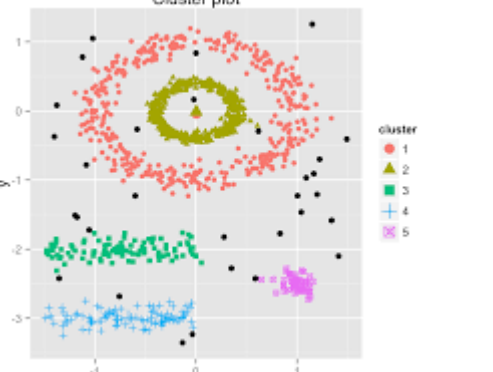
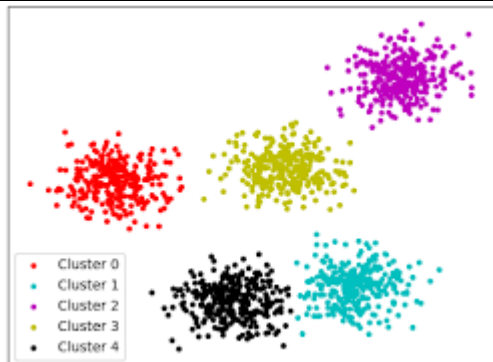
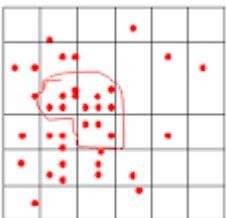
**UnSupervised** learning uses **clustering to** group sample data into groups that have similarity based on certain features.



You also can use **unSupervised learning** when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you.

The **difference** between classification and clustering is that **classification** uses predefined classes in which objects are assigned, while **clustering** identifies similarities **between** objects, which it groups according to those characteristics in common and which **differentiate** them from other .

Clusters may be formed in spherical form, density-base, hierarchy-based, partitioned and grid.

|                |   |
|----------------|---|
| spherical form |                       |
| density-base,  | <p>Cluster plot</p>  |
| Partitioned    |                     |
| Grid           |                      |

## Applications of Clustering

We can find clustering useful in the following areas:

**Data summarization and compression** – Clustering is widely used in the areas where we require data summarization, compression and reduction as well. The examples are image processing and vector quantization.

**Collaborative systems and customer segmentation** – Since clustering can be used to find similar products or same kind of users, it can be used in the area of collaborative systems and customer segmentation.

**Serve as a key intermediate step for other data mining tasks** – Cluster analysis can generate a compact summary of data for classification, testing, hypothesis generation; hence, it serves as a key intermediate step for other data mining tasks also.

**Trend detection in dynamic data** – Clustering can also be used for trend detection in dynamic data by making various clusters of similar trends.

**Social network analysis** – Clustering can be used in social network analysis. The examples are generating sequences in images, videos or audios.

**Biological data analysis** – Clustering can also be used to make clusters of images, videos hence it can successfully be used in biological data analysis

## Clustering Methods :

- **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. Example *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)* , *OPTICS (Ordering Points to Identify Clustering Structure)* etc.

- **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 category
  - **Agglomerative** (*bottom up approach*)
  - **Divisive** (*top down approach*)

examples *CURE (Clustering Using Representatives)*, *BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies)* etc.

- **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 example *K-means*, *CLARANS (Clustering Large Applications based upon Randomized Search)* etc.
- **Grid-based Methods** : In this method the data space is 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.

## Clustering Summary

| Method                | General Characteristics   |
|-----------------------|---|
| Partitioning methods  | <ul style="list-style-type: none"> <li>– Find mutually exclusive clusters of spherical shape</li> <li>– Distance-based</li> <li>– May use mean or medoid (etc.) to represent cluster center</li> <li>– Effective for small- to medium-size data sets</li> </ul>   |
| Hierarchical methods  | <ul style="list-style-type: none"> <li>– Clustering is a hierarchical decomposition (i.e., multiple levels)</li> <li>– Cannot correct erroneous merges or splits</li> <li>– May incorporate other techniques like microclustering or consider object “linkages”</li> </ul>  |
| Density-based methods | <ul style="list-style-type: none"> <li>– Can find arbitrarily shaped clusters</li> <li>– Clusters are dense regions of objects in space that are separated by low-density regions</li> <li>– Cluster density: Each point must have a minimum number of points within its “neighborhood”</li> <li>– May filter out outliers</li> </ul> |
| Grid-based methods    | <ul style="list-style-type: none"> <li>– Use a multiresolution grid data structure</li> <li>– Fast processing time (typically independent of the number of data objects, yet dependent on grid size)</li> </ul>   |

## CLUSTERING ALGORITHMS

|  |   |
|--|---|
| <b>K-means clustering</b>                    | Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans). This clustering algorithm computes the centroids and iterates until we it finds optimal centroid. It assumes that the number of clusters are already known. It is also called <b>flat clustering</b> algorithm. The number of clusters identified from data by algorithm is represented by 'K' in K-means |
| <b>Gaussian mixture model</b>                | A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters   |
| <b>Mean-Shift Algorithm</b>                  | It is another powerful clustering algorithm used in unsupervised learning. Unlike K-means clustering, it does not make any assumptions hence it is a non-parametric algorithm.  |
| <b>Hierarchical clustering</b>               | Splits clusters along a hierarchical tree to form a classification system. Can be used for Cluster loyalty-card customer  |
| <b>DBSCAN</b>                                | <b>DBSCAN Density-Based Spatial Clustering of Applications with Noise</b> is a density based clustered algorithm similar to mean-shift.   |
| <b>Agglomerative Hierarchical Clustering</b> | Hierarchical clustering algorithms actually fall into 2 categories: top-down or bottom-up. Bottom-up algorithms treat each data point as a single cluster at the outset and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all data points. Bottom-up hierarchical clustering is therefore called hierarchical agglomerative clustering <i>or</i> HAC.            |

## DBSCAN

DBSCAN groups together points that are close to each other based on a distance measurement (usually Euclidean distance) and a minimum number of points. It also marks as outliers the points that are in low-density regions.

The DBSCAN algorithm basically requires 2 parameters:

**eps:** specifies how close points should be to each other to be considered a part of a cluster. It means that if the distance between two points is lower or equal to this value (eps), these points are considered neighbors.

**minPoints:** the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region.

### Parameter estimation:

**eps:** if the eps value chosen is too small, a large part of the data will not be clustered. It will be considered outliers because don't satisfy the number of points to create a dense region. On the other hand, if the value that was chosen is too high, clusters will merge and the majority of objects will be in the same cluster. The eps should be chosen based on the distance of the dataset (we can use a k-distance graph to find it), but in general small eps values are preferable.

**minPoints:** As a general rule, a minimum minPoints can be derived from a number of dimensions (D) in the data set, as  $\text{minPoints} \geq D + 1$ . Larger values are usually better for data sets with noise and will form more significant clusters. The minimum value for the minPoints must be 3, but the larger the data set, the larger the minPoints value that should be chosen.

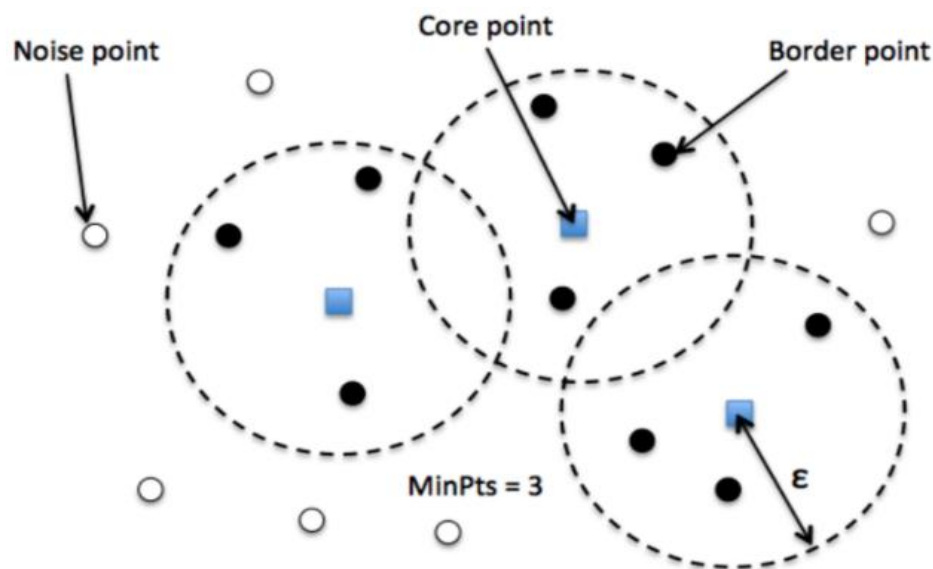
The DBSCAN algorithm should be used to find associations and structures in data that are hard to find manually but that can be relevant and useful to find patterns and predict trends.

These parameters can be understood if we explore two concepts called Density Reachability and Density Connectivity.

**Reachability** in terms of density establishes a point to be reachable from another if it lies within a particular distance ( $\epsilon$ ) from it.

**Connectivity**, on the other hand, involves a transitivity based chaining-approach to determine whether points are located in a particular cluster. For example,  $p$  and  $q$  points could be connected if  $p \rightarrow r \rightarrow s \rightarrow t \rightarrow q$ , where  $a \rightarrow b$  means  $b$  is in the neighborhood of  $a$ .

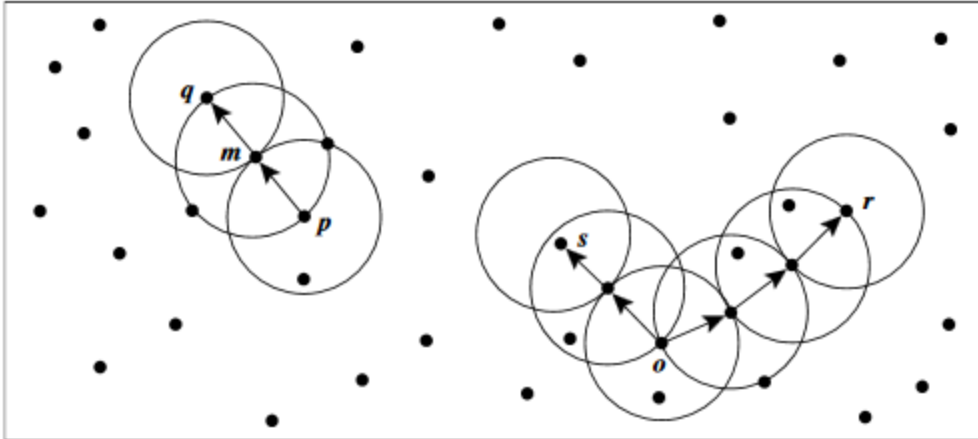
There are three types of points after the DBSCAN clustering is complete:



- **Core** — This is a point that has at least  $m$  points within distance  $n$  from itself.
- **Border** — This is a point that has at least one Core point at a distance  $n$ .
- **Noise** — This is a point that is neither a Core nor a Border. And it has less than  $m$  points within distance  $n$  from itself.

### Algorithmic steps for DBSCAN clustering

- The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).
- If there are at least 'minPoint' points within a radius of ' $\epsilon$ ' to the point then we consider all these points to be part of the same cluster.
- The clusters are then expanded by recursively repeating the neighborhood calculation for each neighboring point



## DBSCAN Algorithm

### Inputs:

**D:** a data set containing  $n$  objects,  
 **$r$  (eps):** the radius parameter,  
**MinPts:** the neighborhood density threshold.

**Output:** A set of density-based clusters.

### Method:

- (1) mark all objects as unvisited;
- (2) do
  - (3) randomly select an unvisited object  $p$ ;
  - (4) mark  $p$  as visited;
  - (5) if the  $r$ -neighborhood of  $p$  has at least MinPts objects
    - (6) create a new cluster  $C$ , and add  $p$  to  $C$ ;
    - (7) let  $N$  be the set of objects in the  $r$ -neighborhood of  $p$ ;
    - (8) for each point  $p_2$  in  $N$ 
      - (9) If  $p_2$  is unvisited
        - mark  $p_2$  as visited;
      - (10) if the  $r$ -neighborhood of  $p_2$  has at least MinPts points,
        - add those points to  $N$ ;
      - (11) if  $p_2$  is not yet a member of any cluster
        - add  $p_2$  to  $C$ ;
      - (12) end for
    - (13) output  $C$ ;



- (14) else
  - Mark p as noise;
- (15) loop until no object is un visited;

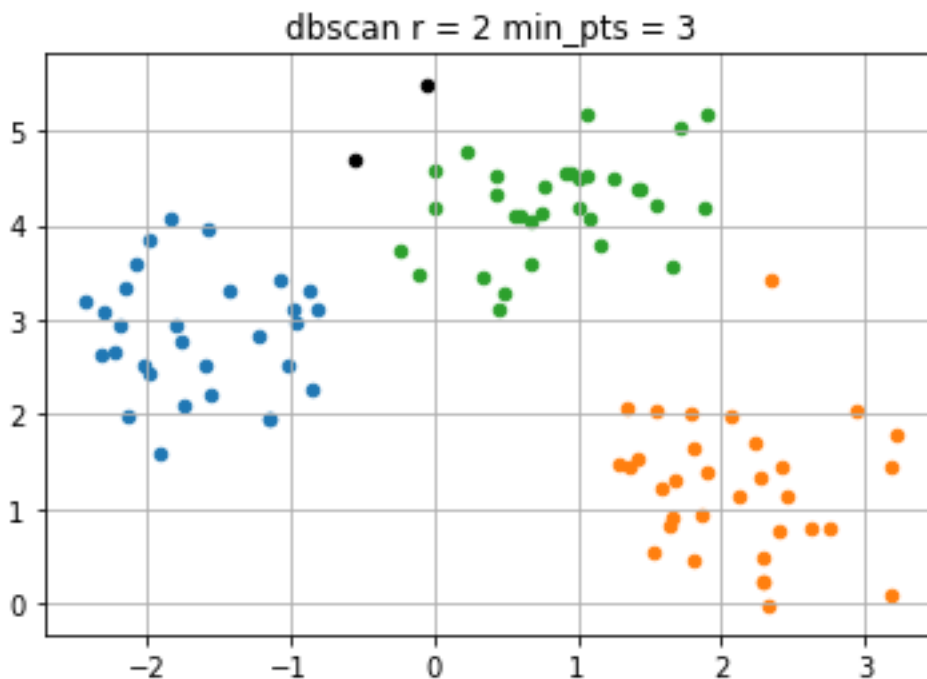
### Clustering HOMEWORK Question 1

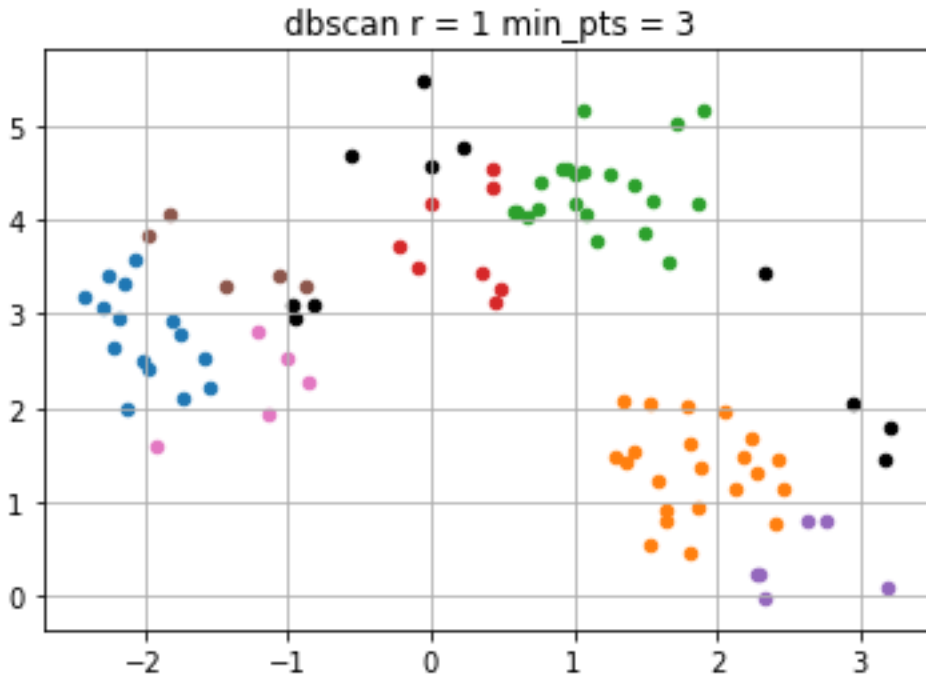
Code the Dbscan clustering algorithm. Plot the blobs and noise points.  
Use different values of r (eps) and minPts. You can make some blobs with the following code,

```
from sklearn.datasets import make_blobs  
points, classes = make_blobs(n_samples=n, centers=3, cluster_std=0.60, random_state=0)
```

Call your python file clustering\_homework1.py

You should get something like this:

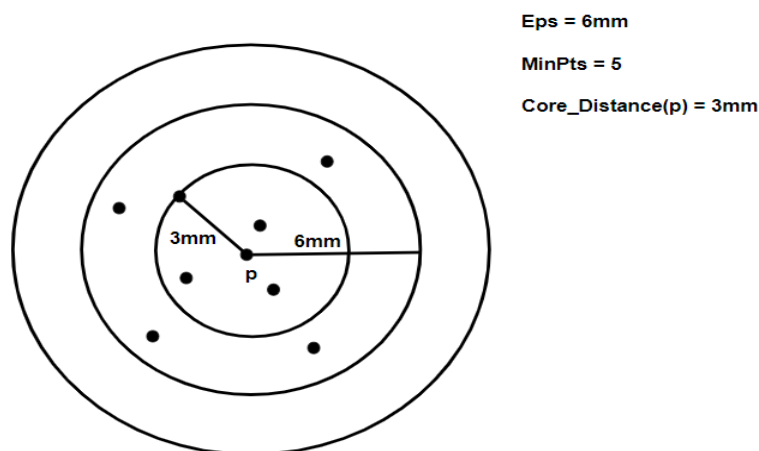




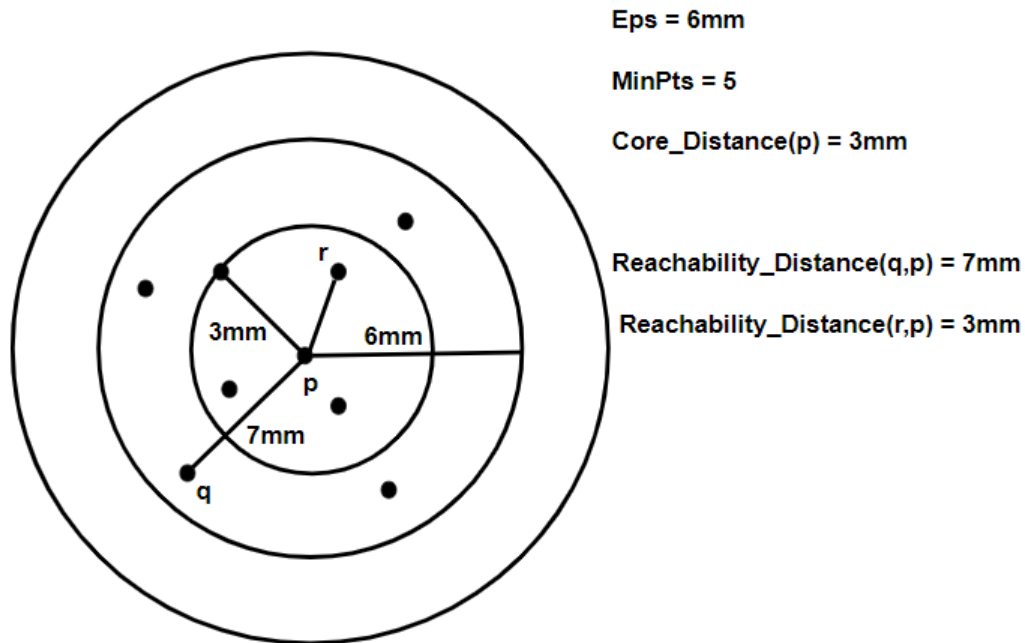
## OPTICS

**OPTICS** Clustering stands for **Ordering Points To Identify Cluster Structure**. It draws inspiration from the DBSCAN clustering algorithm. It adds two more terms to the concepts of DBSCAN clustering. They are:-

1. **Core Distance:** It is the minimum value of radius required to classify a given point as a core point. If the given point is not a Core point, then it's Core Distance is undefined.



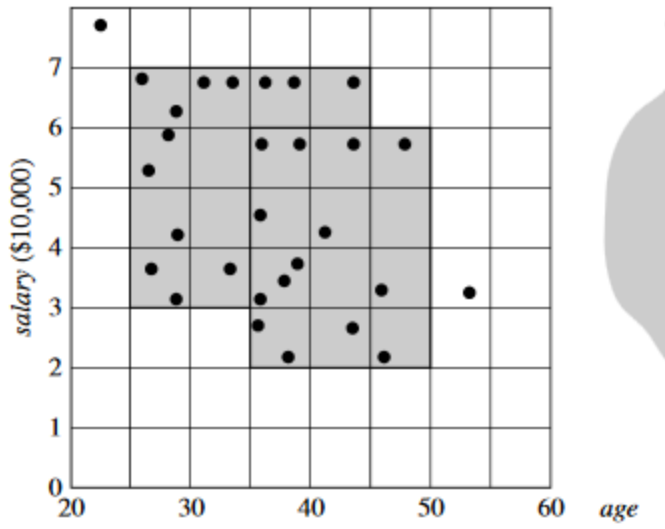
2. **Reachability Distance:** It is defined with respect to another data point  $q$  (Let). The Reachability distance between a point  $p$  and  $q$  is the maximum of the Core Distance of  $p$  and the Euclidean Distance (or some other distance metric) between  $p$  and  $q$ . Note that The Reachability Distance is not defined if  $q$  is not a Core point.



This clustering technique is different from other clustering techniques in the sense that this technique does not explicitly segment the data into clusters. Instead, it produces a visualization of Reachability distances and uses this visualization to cluster the data.

### Grid based clustering

The grid-based clustering approach uses a multi resolution grid data structure. It quantizes the object space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed. The main advantage of the approach is its fast processing time, which is typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space.



## K-MEANS CLUSTERING ALGORITHM

K-means clustering is a clustering algorithm that aims to partition  $n$  observations into  $k$  clusters. The data points are assigned to a cluster in such a manner that the sum of the squared distance between the data points and centroid would be minimum. It is to be understood that less variation within the clusters will lead to more similar data points within same cluster.

There are 3 steps:

|                       |  |
|-----------------------|--|
| <b>Initialization</b> | K initial “means” (centroids) are generated at random                            |
| <b>Assignment</b>     | K clusters are created by associating each observation with the nearest centroid |
| <b>Update</b>         | The centroid of the clusters becomes the new mean                                |

Assignment and Update are repeated iteratively until convergence

The end result is that the sum of squared errors is minimized between points and their respective centroids.

**Step 0:** make data point blobs

**Step 1:** specify the number of clusters, K

**Step 2:** randomly generate K centroids

**Step 3:** fill clusters with data points close to each cluster centroid

**Step 4:** calculate sum squares of distance

If sum square of distance is not changing then exit loop

**Step 5:** calculate new centroids from x and y cluster points

$$x = \frac{1}{n} \cdot \sum_{k=1}^n x_k$$

$$y = \frac{1}{n} \cdot \sum_{k=1}^n y_k$$

Repeat repeat steps 3 to 5

**Step 6:** plot clusters and centroids

## Clustering HOMEWORK Question 2

Code the k-means clustering algorithm, plot the blobs and centroids. You can make some blobs with the following code,

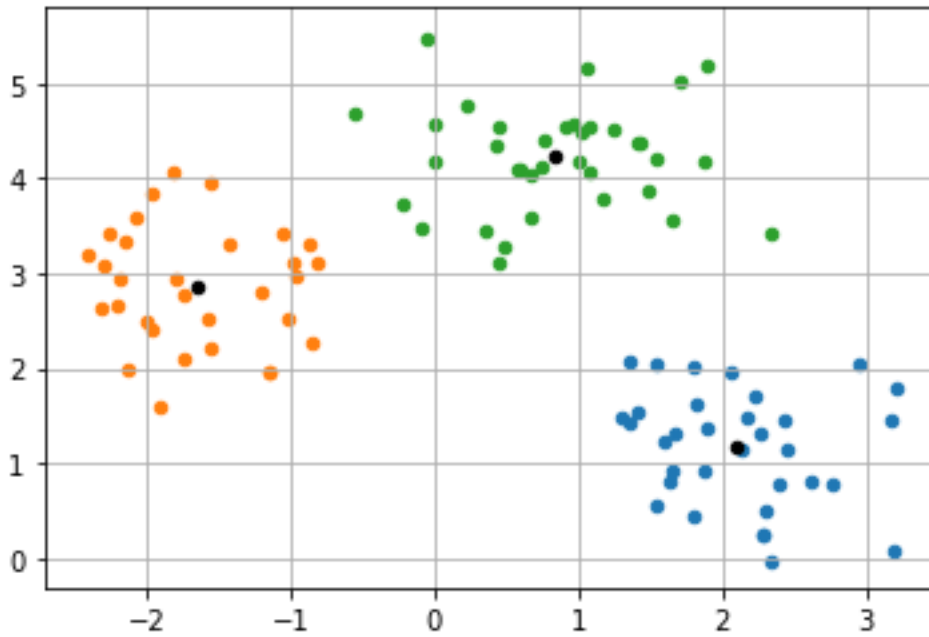
```
from sklearn.datasets import make_blobs
points, classes = make_blobs(n_samples=n, centers=3, cluster_std=0.60, random_state=0)
```

Try different numbers of blobs and K values.

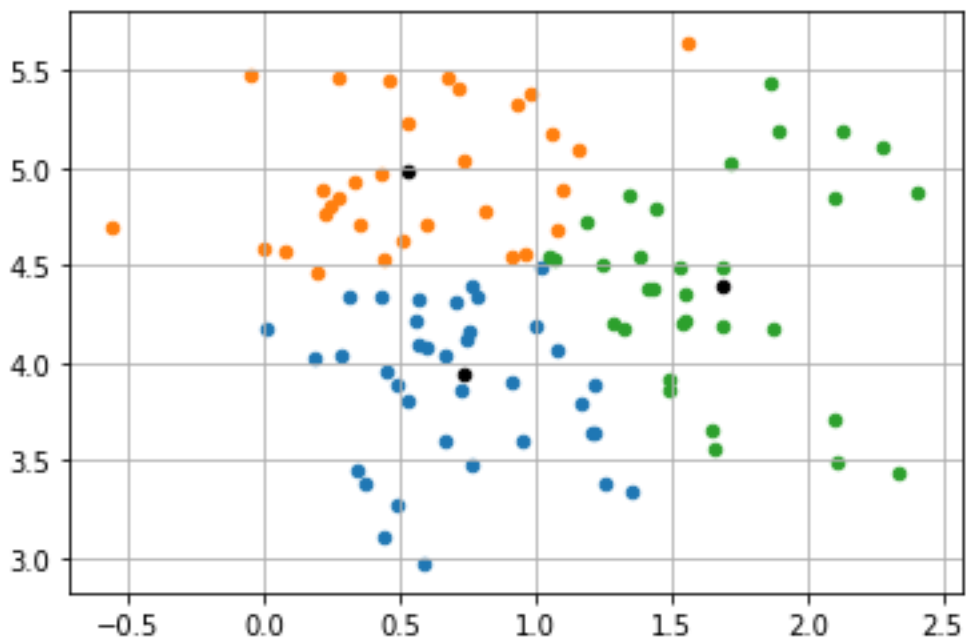
Call your python file clustering\_homework2.py

You should get something like this:

kmeans k = 3 iterations = 1000



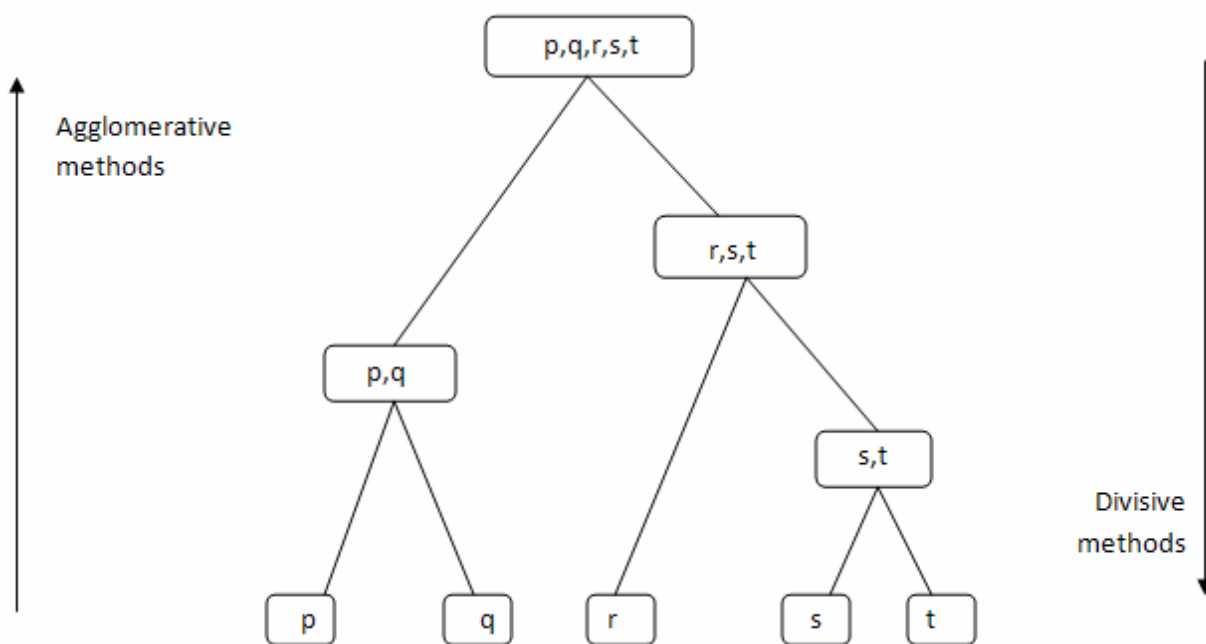
kmeans k = 3 iterations = 3



## Hierarchical Agglomerative Clustering

Hierarchical clustering algorithms group similar objects into groups called **clusters**. There are two types of hierarchical clustering algorithms:

- **Agglomerative** — Bottom up approach. Start with many small clusters and merge them together to create bigger clusters.
- **Divisive** — Top down approach. Start with a single cluster than break it up into smaller clusters.



## Some pros and cons of Hierarchical Clustering

### Pros

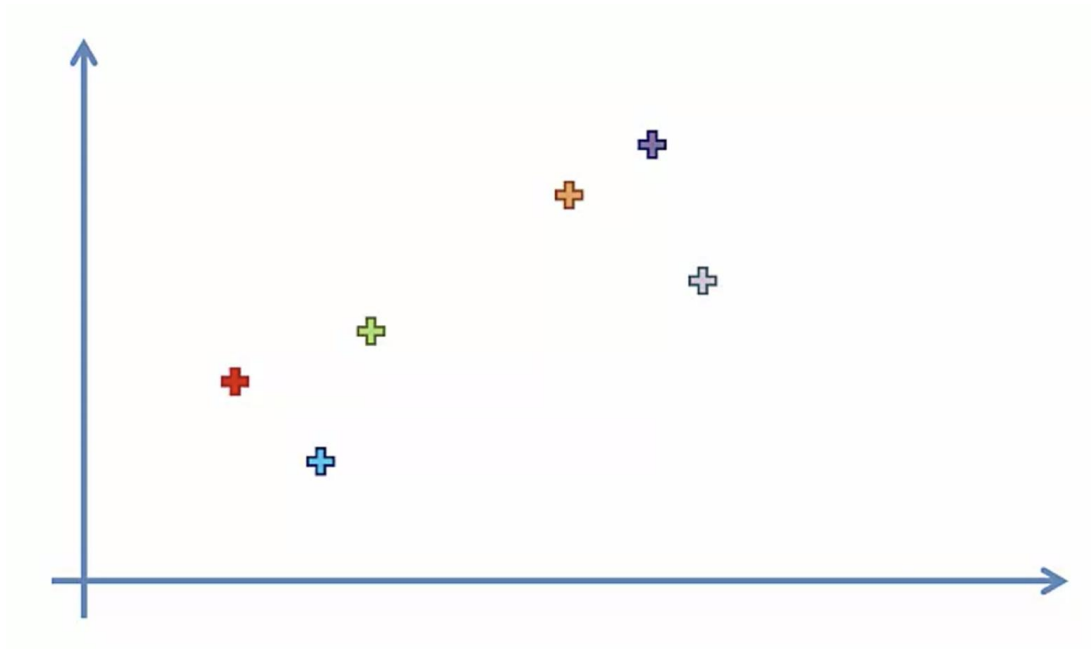
- No assumption of a particular number of clusters (i.e. k-means)
- May correspond to meaningful taxonomies

### Cons

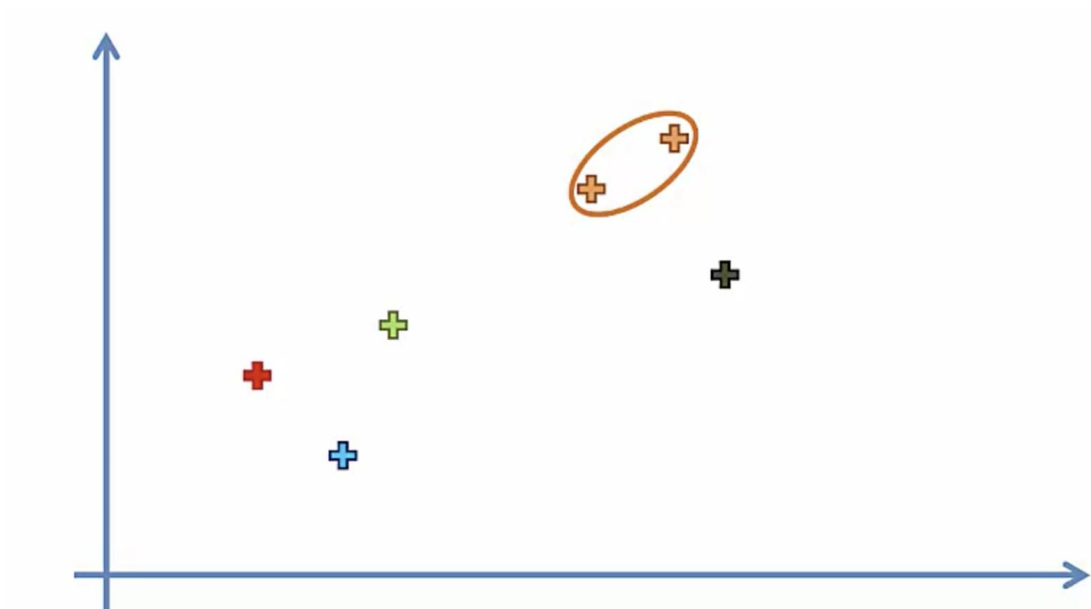
- Once a decision is made to combine two clusters, it can't be undone
- Too slow for large data sets,  $O(n^2 \log(n))$

## How Hierarchical **Agglomerative Clustering** works:

1. Make each data point a cluster

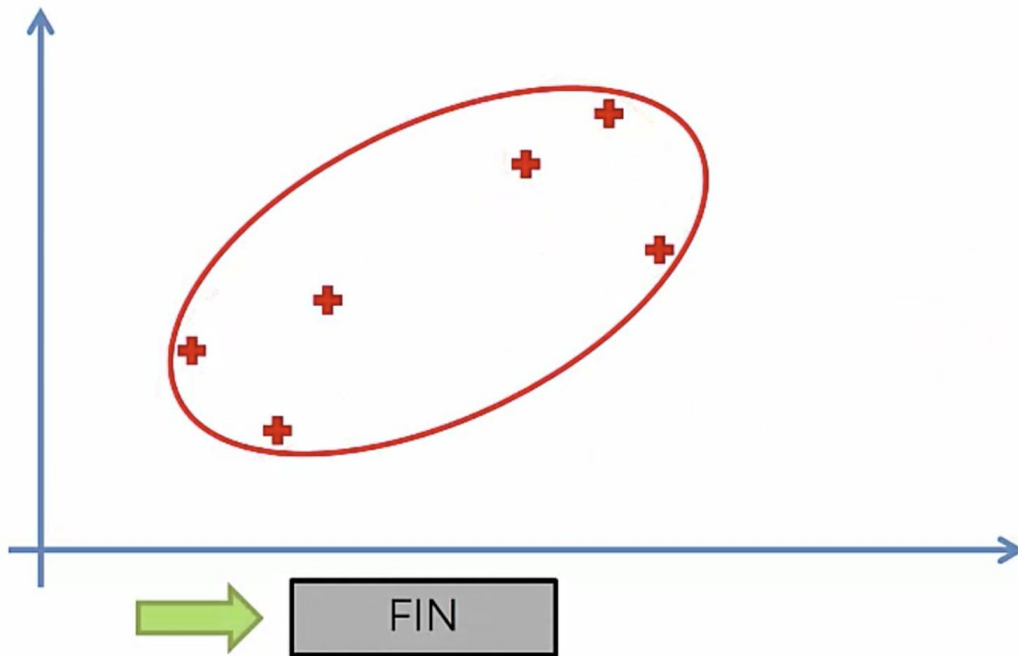


2. Take the two closest clusters and make them one cluster





3. Repeat step 2 until there is only one cluster

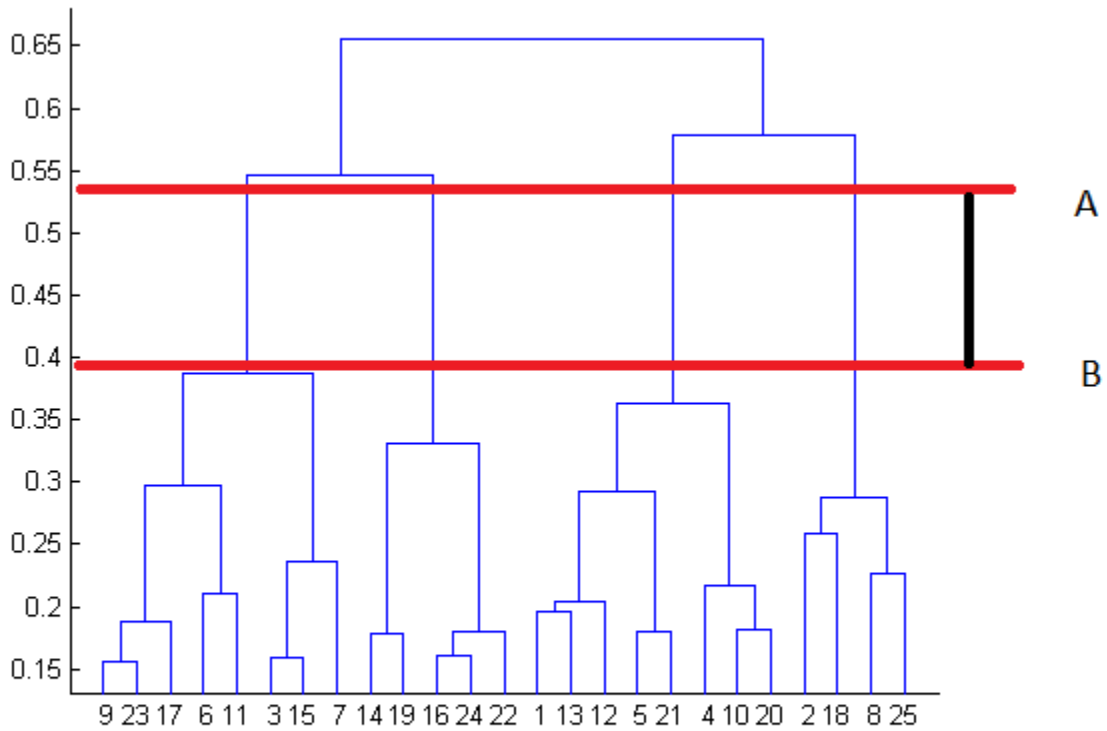


## Dendrograms

We can use a dendrogram to visualize the history of groupings and figure out the optimal number of clusters.

1. Determine the largest vertical distance that doesn't intersect any of the other clusters
2. Draw a horizontal line at both extremities
3. The optimal number of clusters is equal to the number of vertical lines going through the horizontal line

For example in the case below, best choice for number of clusters will be

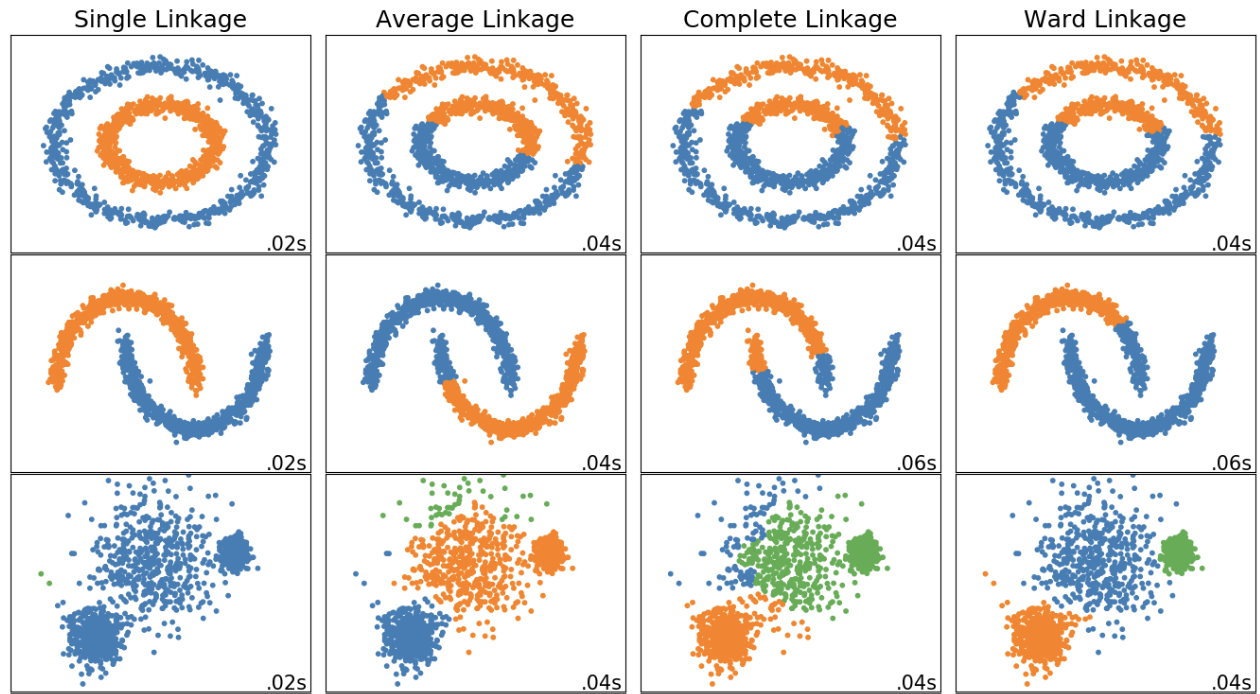


### Linkage Criteria

Similar to gradient descent, you can tweak certain parameters to get drastically different results.

This example aims at showing characteristics of different linkage on datasets

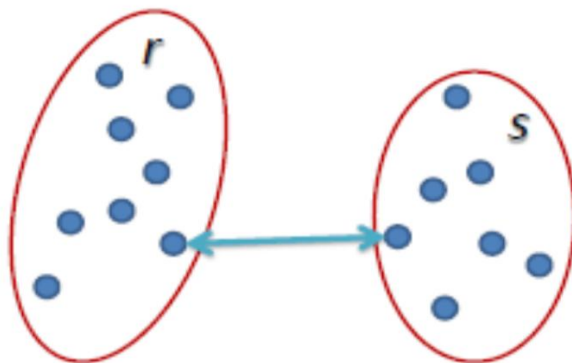
- noisy\_circles
- noisy\_moons
- blobs



The linkage criteria refers to how the distance between clusters is calculated. Take the two closest clusters and make them one cluster.

### Single Linkage

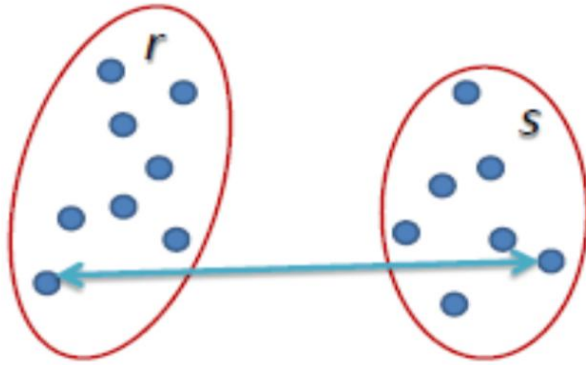
The distance between two clusters is the shortest distance between two points in each cluster



$$L(r, s) = \min(D(x_{ri}, x_{sj}))$$

## Complete Linkage

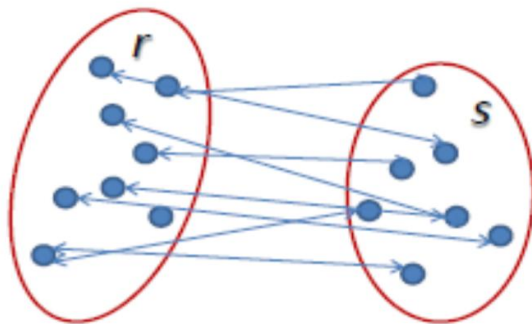
The distance between two clusters is the longest distance between two points in each cluster



$$L(r, s) = \max(D(x_{ri}, x_{sj}))$$

## Average Linkage

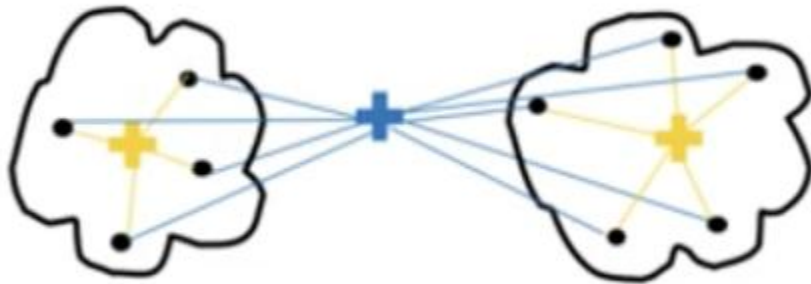
The distance between clusters is the average distance between each point in one cluster to every point in other cluster



$$L(r, s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

## Ward Linkage

The distance between clusters is the sum of squared differences within all clusters

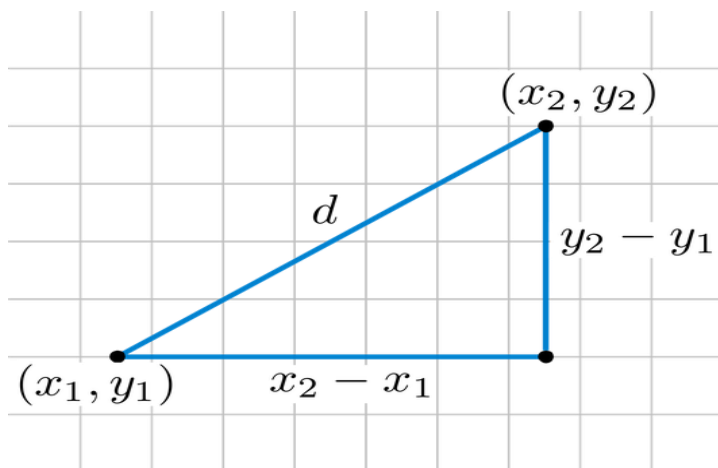


## Distance Metric

The method you use to calculate the distance between data points will affect the end result.

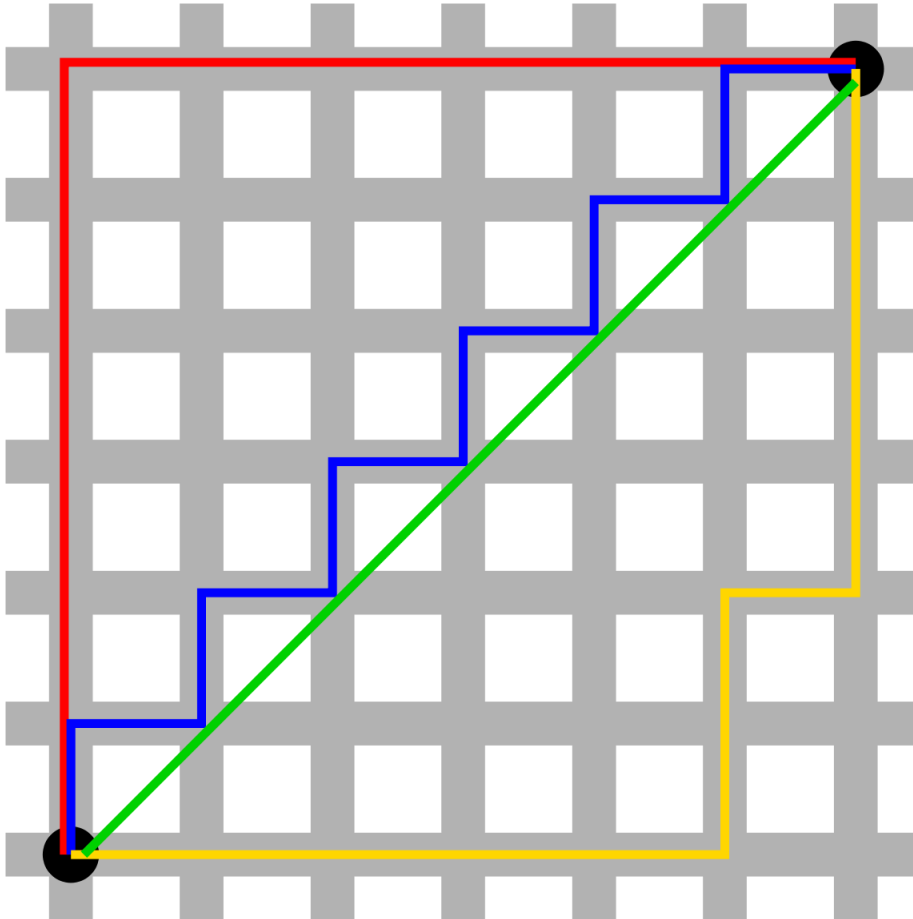
## Euclidean Distance

The shortest distance between two points. For example, if  $x=(a,b)$  and  $y=(c,d)$ , the Euclidean distance between  $x$  and  $y$  is  $\sqrt{(a-c)^2+(b-d)^2}$



## Manhattan Distance

Imagine you were in the downtown center of a big city and you wanted to get from point A to point B. You wouldn't be able to cut across buildings, rather you'd have to make your way by walking along the various streets. For example, if  $x=(a,b)$  and  $y=(c,d)$ , the Manhattan distance between  $x$  and  $y$  is  $|a-c| + |b-d|$

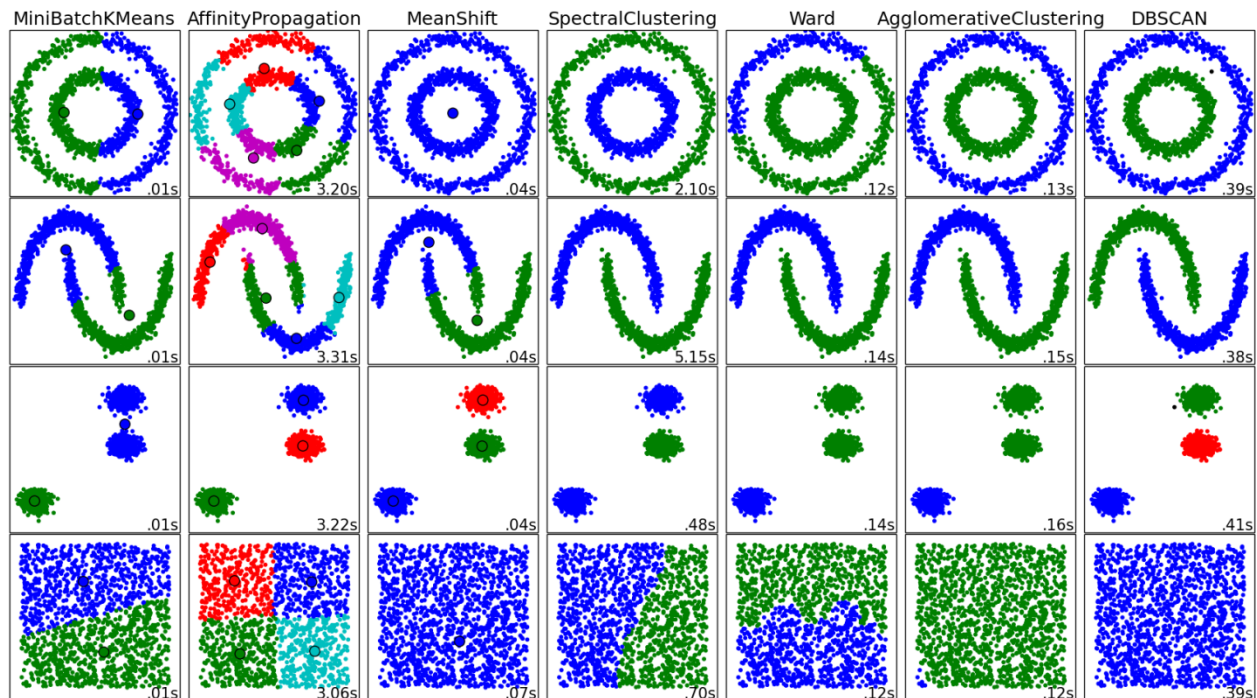


## SKLEARN Clustering Algorithms located in sklearn.cluster

| Method name                                  | Parameters  | Scalability  | Use case  | Geometry (metric used)                          |
|--|---|--|---|---|
| <a href="#">K-Means</a>                      | number of clusters  | Very large<br><code>n_samples</code> ,<br>medium<br><code>n_clusters</code> with<br><a href="#">MiniBatch code</a> | General-purpose,<br>even cluster size,<br>flat geometry, not<br>too many clusters     | Distances<br>between points                     |
| <a href="#">Affinity propagation</a>         | damping, sample preference  | Not scalable with<br><code>n_samples</code>  | Many clusters,<br>uneven cluster size,<br>non-flat geometry                           | Graph distance<br>(e.g. nearest-neighbor graph) |
| <a href="#">Mean-shift</a>                   | Bandwidth   | Not scalable with<br><code>n_samples</code>  | Many clusters,<br>uneven cluster size,<br>non-flat geometry                           | Distances<br>between points                     |
| <a href="#">Spectral clustering</a>          | number of clusters  | Medium<br><code>n_samples</code> , small<br><code>n_clusters</code>  | Few clusters, even<br>cluster size, non-flat<br>geometry                              | Graph distance<br>(e.g. nearest-neighbor graph) |
| <a href="#">Ward hierarchical clustering</a> | number of clusters<br>or distance<br>threshold                            | Large <code>n_samples</code><br>and <code>n_clusters</code>  | Many clusters,<br>possibly<br>connectivity<br>constraints                             | Distances<br>between points                     |
| <a href="#">Agglomerative clustering</a>     | number of clusters<br>or distance<br>threshold, linkage<br>type, distance | Large <code>n_samples</code><br>and <code>n_clusters</code>  | Many clusters,<br>possibly<br>connectivity<br>constraints, non<br>Euclidean distances | Any pairwise<br>distance                        |
| <a href="#">DBSCAN</a>                       | neighborhood size   | Very large<br><code>n_samples</code> ,<br>medium<br><code>n_clusters</code>  | Non-flat geometry,<br>uneven cluster sizes  | Distances<br>between nearest<br>points          |
| <a href="#">OPTICS</a>                       | minimum cluster<br>membership   | Very large<br><code>n_samples</code> , large<br><code>n_clusters</code>  | Non-flat geometry,<br>uneven cluster<br>sizes, variable<br>cluster density            | Distances<br>between points                     |
| <a href="#">Gaussian mixtures</a>            | Many  | Not scalable   | Flat geometry, good<br>for density<br>estimation                                      | Mahalanobis<br>distances to<br>centers          |
| <a href="#">Birch</a>                        | branching factor,<br>threshold, optional<br>global clusterer.             | Large <code>n_clusters</code><br>and <code>n_samples</code>  | Large dataset,<br>outlier removal,<br>data reduction.                                 | Euclidean<br>distance<br>between points         |

This example aims at showing characteristics of different clustering algorithms on datasets

noisy\_circles  
noisy\_moons  
blobs  
no\_structure



## k-means Clustering using sklearn

sklearn has a k-means clustering module located in **sklearn.cluster**

In our k-means sklearn program we first make 2 random blobs and then calculate the center centroids. We then make another set of random blobs and plot the centroids of the first set over the second set of blobs.

To make blobs we use the sklearn **make\_blobs** and import the make\_blobs library

```
from sklearn.datasets.samples_generator import make_blobs
```



```
x, y_true = make_blobs
(n_samples = n, centers = 3, cluster_std = 0.60, random_state = 0)
```

x are the x,y points of the blobs where as y\_true are the 3 classification numbers.

We uses the kmeans model from sklearn.cluster

```
import sklearn.cluster import KMeans
```

We make KMeans model with 3 clusters

```
kmeans = KMeans(n_clusters = 3)
```

and then train the model with the **fit** function

```
kmeans.fit(x)
```

we use **predict** method from the kmeans model to predict the classification numbers

```
y_kmeans = kmeans.predict(x)
```

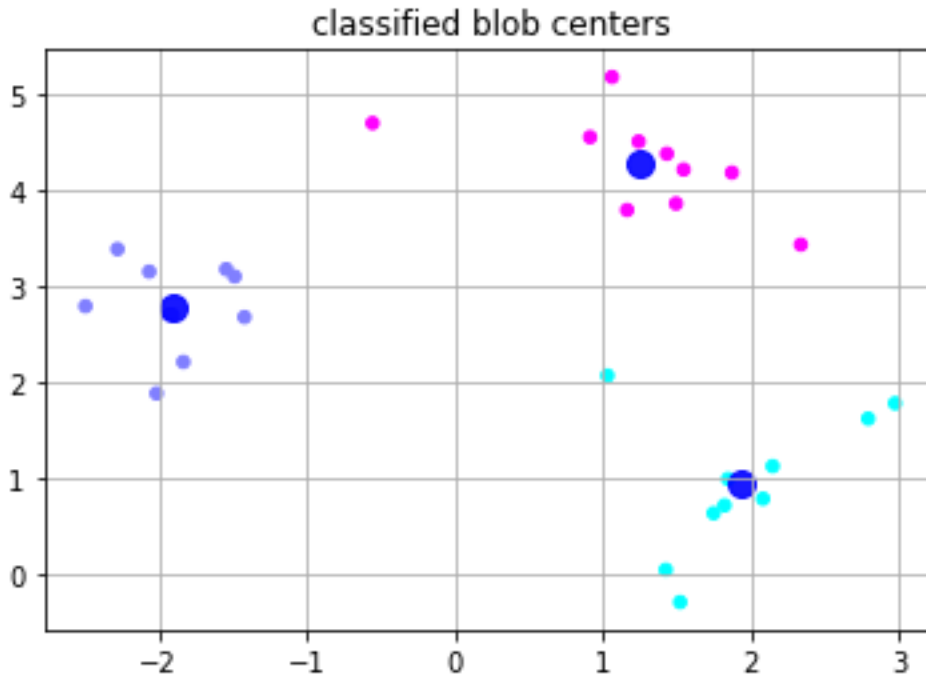
we then get the center centroids of first set of blobs

```
centers = kmeans.cluster_centers_
```

and plot them on the seconds set of blobs

our output:

```
prediction classification:
[2 1 1 1 0 2 0 1 0 1 1 1 1 0 2 2 0 2 1 2 0 0 2 2 2 0 0 0 1 2]
classification set 2:
[0 2 2 2 1 0 1 2 1 2 2 2 2 1 0 0 1 0 2 0 1 1 0 0 0 1 1 1 2 0]
centers of first set of blobs:
[[ 1.93984407  0.94738267]
 [-1.89753197  2.77697024]
 [ 1.25131979  4.27778921]]
```



Our complete program as follows

```

"""
kmeans.py
clustering using kmeans algorithm
"""

import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
#from sklearn.datasets.samples_generator import make_blobs

# generate 3 2d blobs
# x are the xy points
# y_true are the blob numbers
n = 30
x, y_true = make_blobs(n_samples = n, centers = 3, cluster_std = 0.60, random_state = 0)

print('blobs set 1:')
print(x)
print('classification set 1:')
print(y_true)

```

```

# plot blobs
plt.scatter(x[:, 0], x[:, 1], s = 20);
plt.title("blobs")
plt.grid()
plt.show()

# make KMeans model
kmeans = KMeans(n_clusters = 3)

# train the model
kmeans.fit(x)

# predict the classification numbers
y_kmeans = kmeans.predict(x)

print('prediction classification:')
print(y_kmeans)

# make some more blobs
x, y_true = make_blobs(n_samples = n, centers = 3, cluster_std = 0.60, random_state = 0)

print('blobs set 2:')
print(x)
print('classification set 2:')
print(y_true)

# plot the second set of blobs using color map 'cool'
plt.scatter(x[:, 0], x[:, 1], c = y_kmeans, s = 20, cmap = 'cool')

# get the centers of first set of blobs
centers = kmeans.cluster_centers_
print('centers of first set of blobs:')
print(centers)

# plot centers of first set of blobs on second set of blobs
# to check for accuracy
plt.scatter(centers[:, 0], centers[:, 1], c = 'blue', s = 100, alpha = 0.9);
plt.title("classified blob centers")
plt.grid()
plt.show()

```

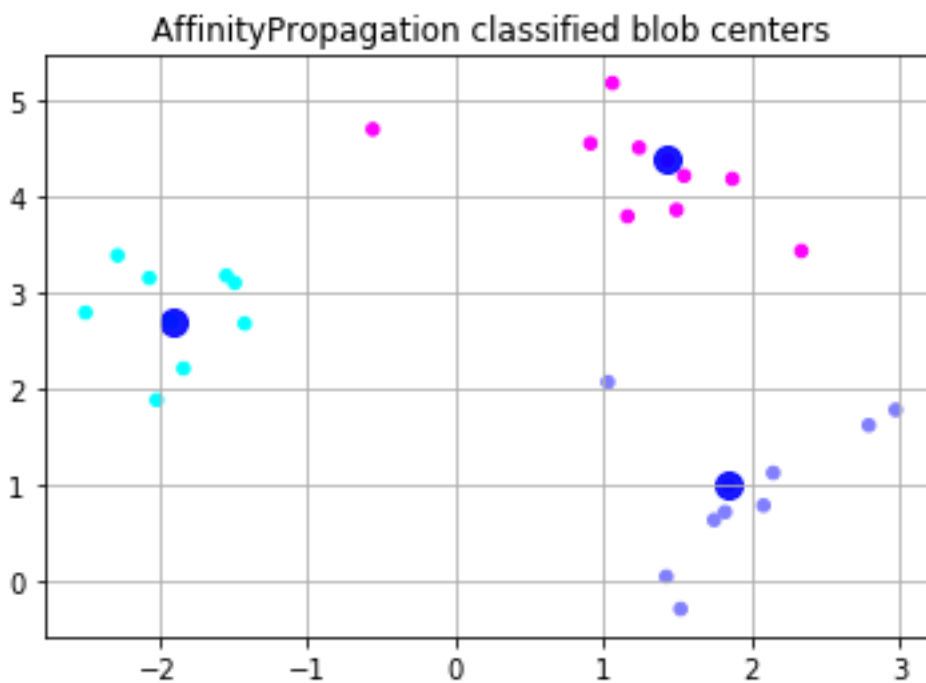
## TO DO

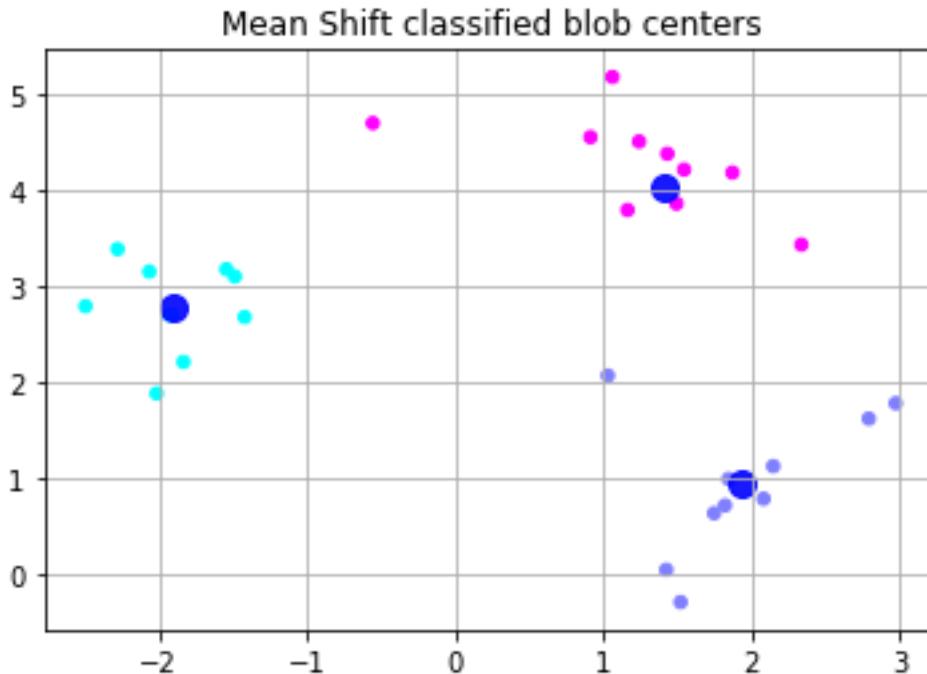
Type in the sklearn example k-means clustering program and run it. Then try these other following clustering algorithms offered by sklearn. MeanShift and AffinityPropagation

(1) clustering = MeanShift(bandwidth=2)

(2) clustering = AffinityPropagation

You should get something like this:





## Hierarchical Agglomerative Clustering using sklearn

Sklearn has the **AgglomerativeClustering** classifier located in module `sklearn.cluster`

```
class sklearn.cluster.AgglomerativeClustering(n_clusters=2, *, affinity='euclidean',
memory=None, connectivity=None, compute_full_tree='auto', linkage='ward',
distance_threshold=None)
```

As before we make 3 blobs using the sklearn **make\_blobs** function located in **sklearn.datasets.samples\_generator**.

The **AgglomerativeClustering** classifier has the **fit\_predict** method that returns a list of labels corresponding to into the X indexes.

For example if the first entry in the labels array is 2 this mean the x coordinates (`x[0][0]`, `x[0][1]`) is blob 2. Where `x[0][0]` is the x coordinate and `x[0][1]` is the y coordinate, since x is a 2 dimensional array of x and y coordinates.

When we plot the blobs on a scatter plot each blob label gets a specific color as denoted in the plot legend. Since we have 3 blobs each blob is assigned a number between 0 and 2 from the classifiers.

We will also plot a **dendrogram** using the scipy **dendrogram** function located in `scipy.cluster.hierarchy`

For our example program we first import all the required libraries.

```
#hierarchical_clustering.py  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn.datasets import make_blobs  
#from sklearn.datasets.samples_generator import make_blobs  
from sklearn.cluster import AgglomerativeClustering  
import scipy.cluster.hierarchy as sch  
  
# make 3 blobs  
X, y_true = make_blobs(n_samples = 30, centers = 3, cluster_std = 0.60, random_state  
= 0)  
  
# make denogram  
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))  
plt.title('Blob Dendrogram')  
plt.xlabel('x')  
plt.ylabel('Euclidean distances')  
plt.show()  
  
# make AgglomerativeClustering  
clustering = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidean', linkage =  
'ward')  
  
# fit and predict and return array of labels  
labels = clustering.fit_predict(X)  
  
# print label predictions  
print("label predictions: ",labels)
```

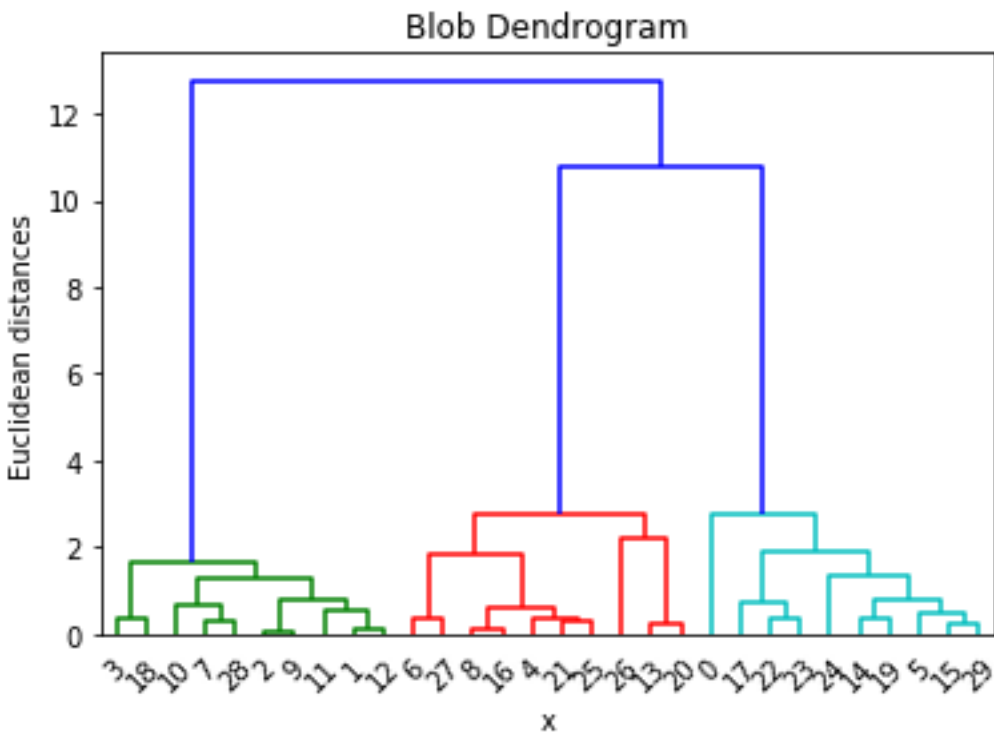
```

# Visualizing the clusters
plt.scatter(X[labels == 0, 0], X[labels == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[labels == 1, 0], X[labels == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[labels == 2, 0], X[labels == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.title('AgglomerativeClustering')
plt.legend()
plt.show()

```

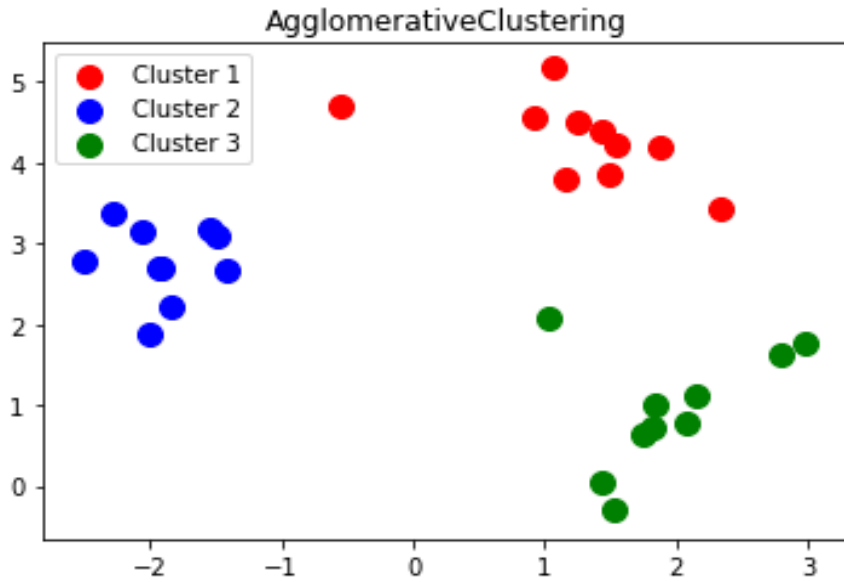
Here is the dendrogram plot:



Here is the output

```
label predictions: [0 1 1 1 2 0 2 1 2 1 1 1 1 2 0 0 2 0 1 0 2 2 0 0 0 2 2 2 1 0]
```

Here is the plot:



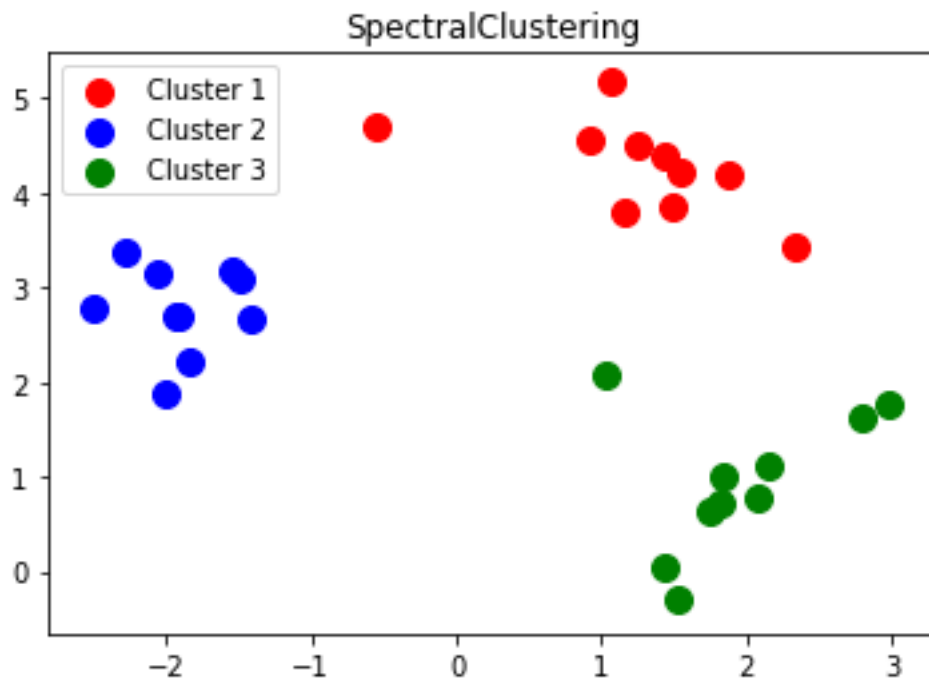
Todo

**(1) Try SpectralClustering located in sklearn.cluster import SpectralClustering**

```

clustering = SpectralClustering(n_clusters=3,random_state=0)
clustering.fit(X)
labels = clustering.labels_

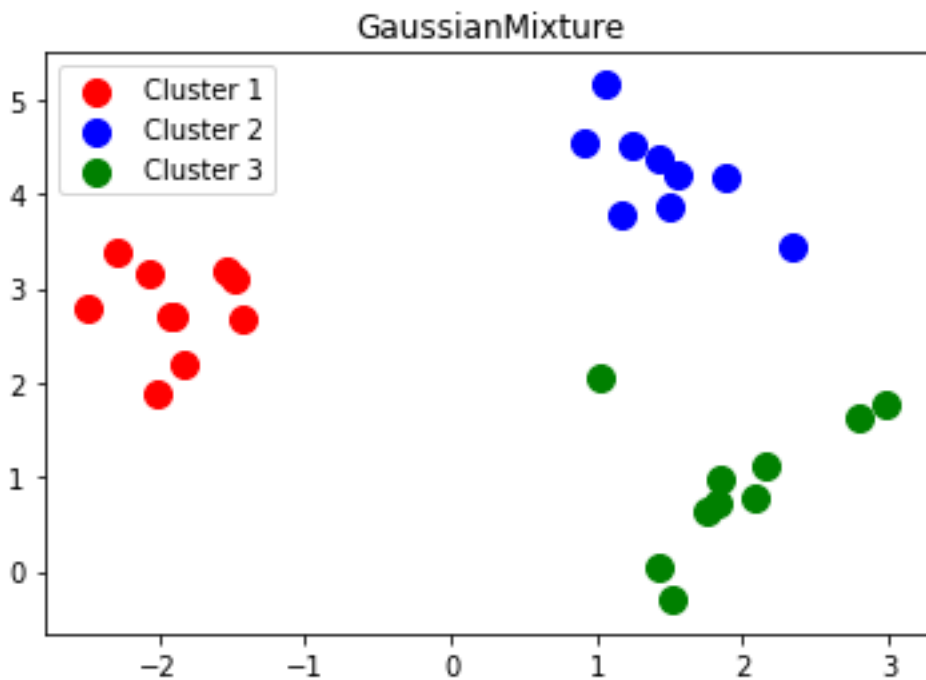
```





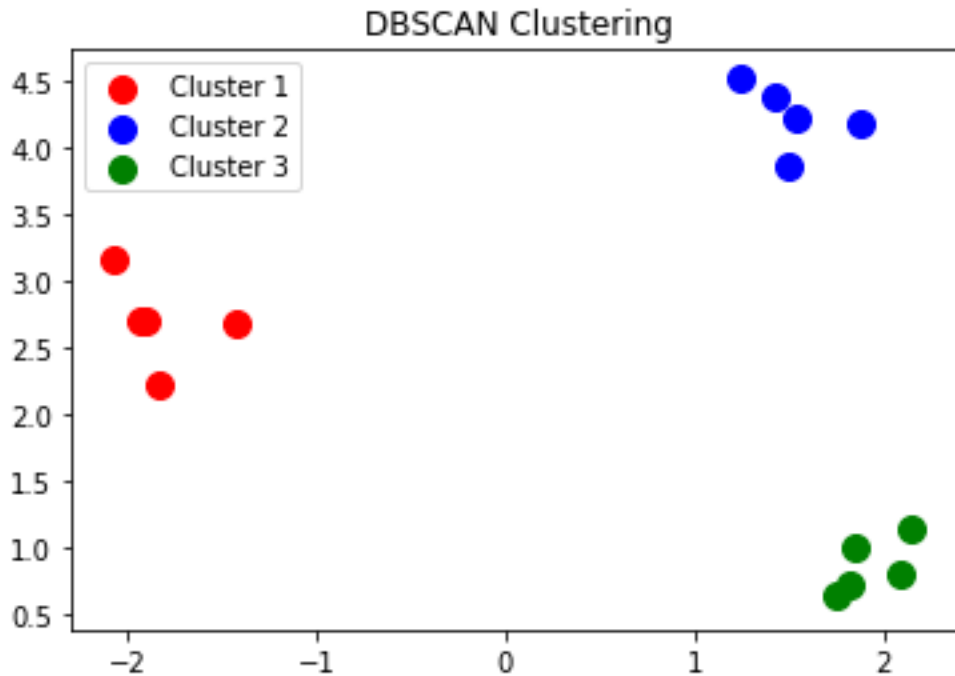
## (2) try GaussianMixture in sklearn.mixture import GaussianMixture

```
clustering = GaussianMixture(n_components=4)
clustering.fit(X)
labels=clustering.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis');
plt.show()
```



## (3) try DBSCAN located in sklearn.cluster import DBSCAN

```
clustering = DBSCAN()
clustering.fit(X)
labels = clustering.labels_
```



## SEGEMENTATION

Segmenting is the process of putting customers into groups based on similarities, and clustering is the process of finding similarities in customers so that they can be grouped, and therefore segmented.

Clustering is the process of using machine learning and algorithms to identify how different types of data are related and creating new segments based on those relationships. Clustering finds the relationship between data points so they can be segmented.

## CLUSTERING HOMEWORK Question 3

Make a 2 dimensional array to store customers and the products they buy. Have about two to three customers in one column and have 8 to 10 corresponding products in the adjacent column. Have about 12 to 15 rows of customers and their corresponding products. Have products groups like shirts, pants, socks, shoes, television, computer monitor etc. Do not place products randomly but assign products to different classes of customers. For example 1 class of customers buys clothing. Another class of customers buys electronics and another class of customers buys computer stuff.

Using a clustering algorithm of your choice, find the customer clusters and the products they buy.

Find out which customers do not have certain products and suggest products for them to buy. You need to use numbers for your customers and products. Either use a lookup dictionary or a encoding mechanism. Use **enumerate** to get which product corresponds to each customers after the clustering process

```
for product_index, customer_class in enumerate(cluster_classes):  
    product_number = x[i][1]
```

Call your python program clusteringhomework.py

You should get something like this:

K-Means

prediction classification:

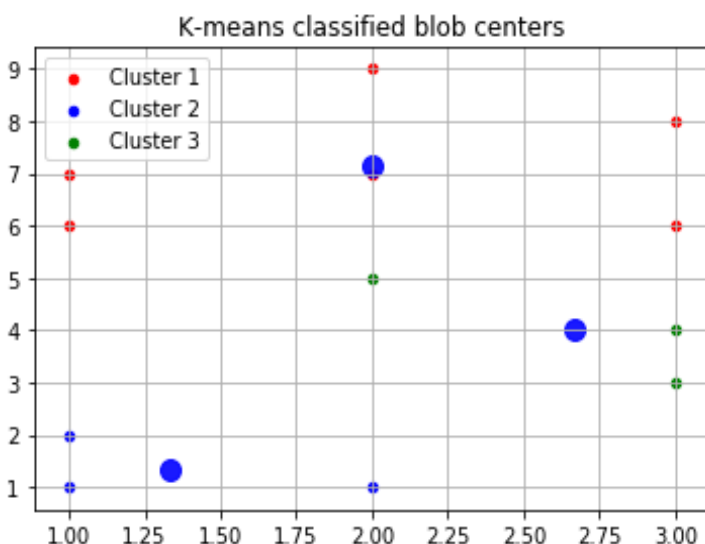
```
[1 1 0 0 2 0 0 1 2 2 0 0]
```

centers of blobs:

```
[[2.    7.16666667]
```

```
[1.33333333 1.33333333]
```

```
[2.66666667 4.    ]]
```



customer: 1 has:

- cell\_phone
- monitor
- mouse
- shoes

Would you like to buy?

- television
- radio
- shirt
- pants
- computer

customer: 2 has:

- television
- radio

Would you like to buy?

- shirt
- pants
- computer
- cell\_phone
- monitor
- shoes
- mouse

customer: 3 has:

- computer
- shirt
- pants

Would you like to buy?

television

radio

cell\_phone

monitor

shoes

mouse

END