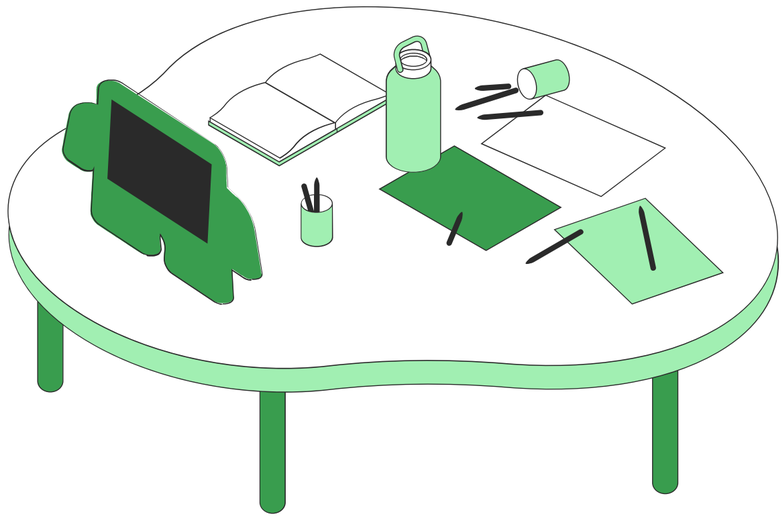
Clustering Techniques

Amr Kandil
Joseph Emad

Outline



Supervised vs Unsupervised Learning

Hard vs Soft Clustering

Distance-based Clustering

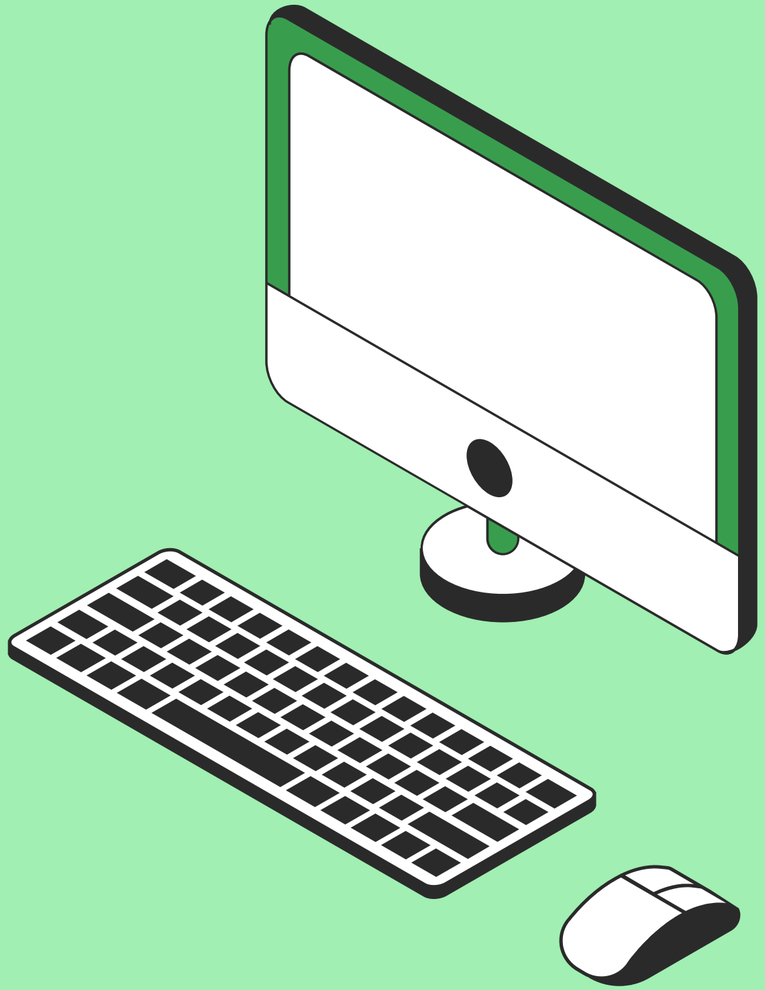
Density-based Clustering

Distribution-based Clustering

Fuzzy Clustering

Silhouette Score

Supervised vs Unsupervised Learning



Supervised Learning

Definition

- In Supervised learning, you train the machine using data which is well “labeled.” It means some data is already tagged with the correct answer. It can be compared to learning which takes place in the presence of a supervisor or a teacher.
- A supervised learning algorithm learns from labeled training data, helps you to predict outcomes for unforeseen data.

Unsupervised Learning

Definition

- Unsupervised learning is a machine learning technique, where you do not need to supervise the model. Instead, you need to allow the model to work on its own to discover information. It mainly deals with the unlabelled data.
- Unsupervised learning algorithms allow you to perform more complex processing tasks compared to supervised learning.

Supervised Learning

Benefits

- Supervised learning allows you to collect data or produce a data output from the previous experience.
- Helps you to optimize performance criteria using experience
- Supervised machine learning helps you to solve various types of real-world computation problems.

Unsupervised Learning

Benefits

- Unsupervised machine learning finds all kind of unknown patterns in data.
- Unsupervised methods help you to find features which can be useful for categorization.
- It is easier to get unlabeled data from a computer than labeled data, which needs manual intervention.

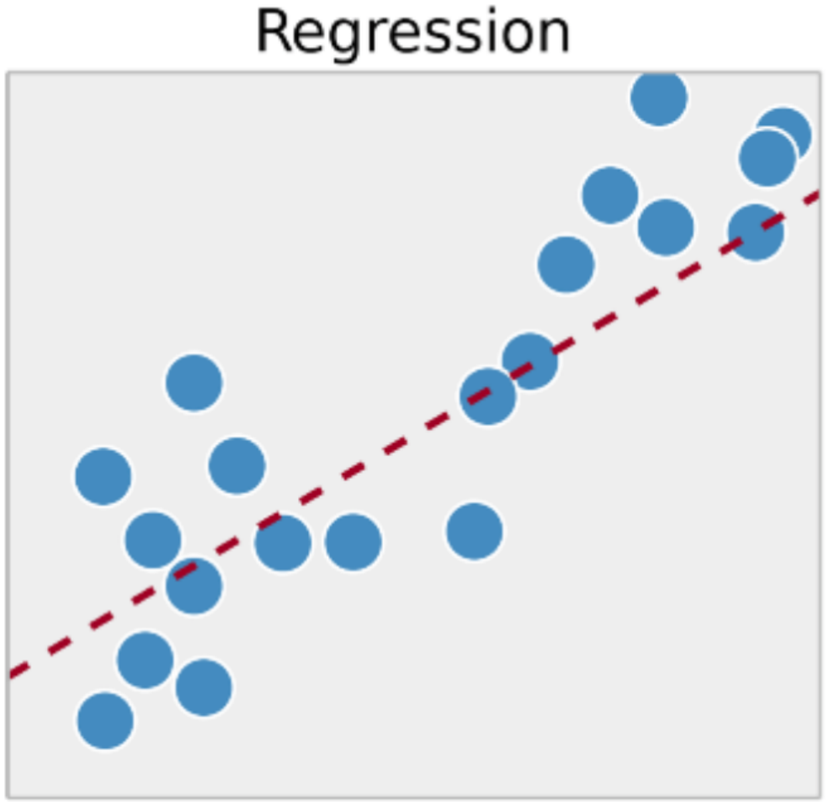
Supervised Learning

Examples

Regression:

Regression technique predicts a single output value using training data.

Example: You can use regression to predict the house price from training data. The input variables will be locality, size of a house, etc.



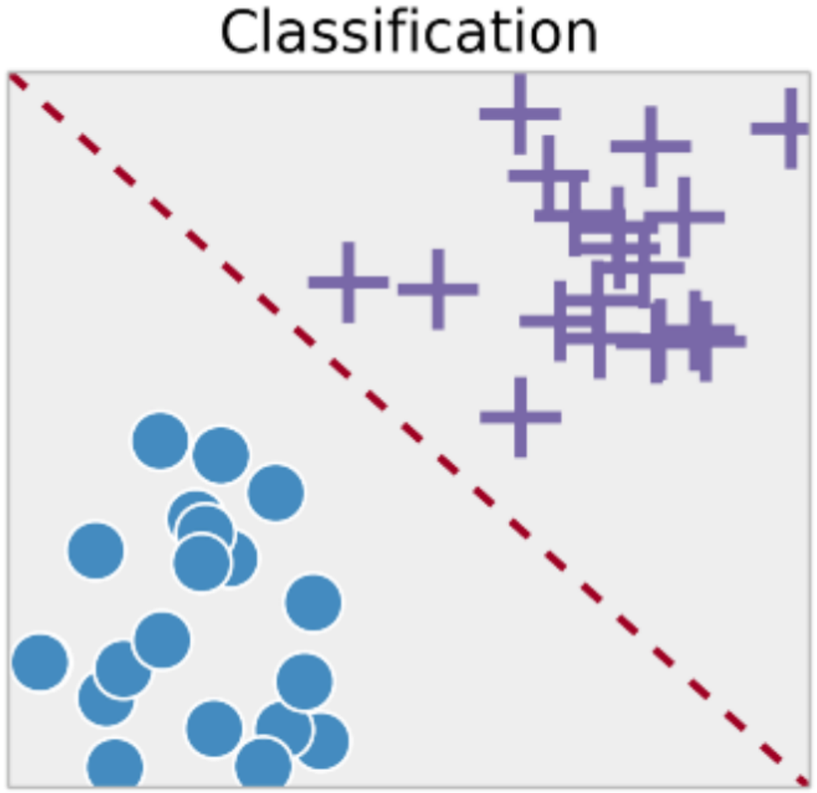
Supervised Learning

Examples

Classification:

Classification means to group the output inside a class. If the algorithm tries to label input into two distinct classes, it is called binary classification. Selecting between more than two classes is referred to as multiclass classification.

Example: Determining whether or not someone will be a defaulter of the loan.



Unsupervised Learning

Examples

Association

Association rules allow you to establish associations amongst data objects inside large databases. This unsupervised technique is about discovering **exciting relationships** between variables in large databases.

Example: people that buy a new home most likely to buy new furniture.

Clustering

Definition

Clustering is an unsupervised learning method in machine learning. It means that it is a machine learning algorithm that can **draw conclusions** from a given dataset on its own, without any kind of human intervention.

Clustering is the process of partitioning a set of data points into several groups so that data points in the same group are more similar to each other and different from data points in other groups. It is essentially a grouping of items based on their similarity and dissimilarity.

HARD CLUSTERING

Each data point either belongs to a cluster completely or not.

SOFT CLUSTERING

Instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned.

Case Study

General Case Study Description

Customer Personality Analysis is a detailed analysis of a company’s ideal customers. It helps a business to better understand its customers and makes it easier for them to modify products according to the specific needs, behaviors and concerns of different types of customers.

Dataset Description

General Information

Number of rows: 2240
Number of features: 29
Features divides into 4 categories:

- People
- Products
- Promotion
- Place

Dataset Description

Features

People

ID: Customer's unique identifier
Year_Birth: Customer's birth year
Education: Customer's education level
Marital_Status: Customer's marital status
Income: Customer's yearly household income
Kidhome: Number of children in customer's household
Teenhome: Number of teenagers in customer's **household**
Dt_Customer: Date of customer's enrollment with the company
Recency: Number of days since customer's last purchase
Complain: 1 if the customer complained in the last 2 years, 0 otherwise

Products

MntWines: Amount spent on wine in last 2 years
MntFruits: Amount spent on fruits in last 2 years
MntMeatProducts: Amount spent on meat in last 2 years
MntFishProducts: Amount spent on fish in last 2 years
MntSweetProducts: Amount spent on sweets in last 2 years
MntGoldProds: Amount spent on gold in last 2 years

Dataset Description

Features

Promotion

INumDealsPurchases: Number of purchases made with a discount
AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise
AcceptedCmp2: 1 if customer accepted the offer in the 2nd campaign, 0 otherwise
AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise
AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise
Response: 1 if customer accepted the offer in the last campaign, 0 otherwise

Place

NumWebPurchases: Number of purchases made through the company's website
NumCatalogPurchases: Number of purchases made using a catalogue
NumStorePurchases: Number of purchases made directly in stores
NumWebVisitsMonth: Number of visits to company's website in the last month

Feature Engineering

Main Changes

- Replace "**Dt_Customer**" with a new feature called "**loyalty_days**", which indicates the number of days since the first purchase.
- Replace "**Year_Birth**" feature by "**Age**" feature which is calculated by subtracting the year of birth from the current year.
- Add all amounts spent by the customer into one new feature called "**Total_Purchases**" indicating the total amount spent by the customer in various categories over the span of two years.
- Add the kids and teenagers features into a new feature called "**Total_Children**" to indicate total children in a household.
- Create three categories in the "**Education**" by simplifying its value counts.

Feature Engineering

Examples

```
# Create Age feature
df["Age"] = 2022-df["Year_Birth"]

# Create Total Purchases feature
df["Total_Purchases"] = df["MntWines"] + df["MntFruits"] +
                        df["MntMeatProducts"] + df["MntFishProducts"] +
                        df["MntSweetProducts"] + df["MntGoldProds"]

# Create Total Children feature
df["Total_Children"]=df["Kidhome"]+df["Teenhome"]

# Create Total Offers Taken as a percentage of all offers
df["Total_Offers_Taken"] = (df["AcceptedCmp1"] + df["AcceptedCmp2"] +
                            df["AcceptedCmp3"] + df["AcceptedCmp4"] +
                            df["AcceptedCmp5"] + df["Response"]) / 6
```

Feature Engineering

Examples

```
[ ]  df["Education"].unique()

     array(['Graduation', 'PhD', 'Master', 'Basic', '2n Cycle'], dtype=object)
```

```
#Segmenting education levels in three groups
df["Education"]=df["Education"].replace({"Basic":"Undergraduate",
                                         "2n Cycle":"Postgraduate",
                                         "Graduation":"Graduate",
                                         "Master":"Postgraduate",
                                         "PhD":"Postgraduate"})
df["Education"].unique()
```

```
array(['Graduate', 'Postgraduate', 'Undergraduate'], dtype=object)
```

```
[ ]  df["Marital_Status"].unique()

     array(['Single', 'Together', 'Married', 'Divorced', 'Widow', 'Alone',
            'Absurd', 'YOLO'], dtype=object)
```

Feature Engineering

Examples

```
! pip install category_encoders
import category_encoders as ce
encoder= ce.BinaryEncoder(cols='Education',return_df=True)
df=encoder.fit_transform(df)
encoder= ce.BinaryEncoder(cols='Marital_Status',return_df=True)
df=encoder.fit_transform(df)
```

Drop Redundant Features

Examples

```
#Dropping some of the redundant features
df = df.drop(["Dt_Customer", "Z_CostContact",
              "Z_Revenue", "Year_Birth", "ID"],
              axis=1)
```

```
df = df.drop(["Wines", "Fruits", "Meat",
              "Fish", "Sweets", "Gold"],
              axis=1)
```

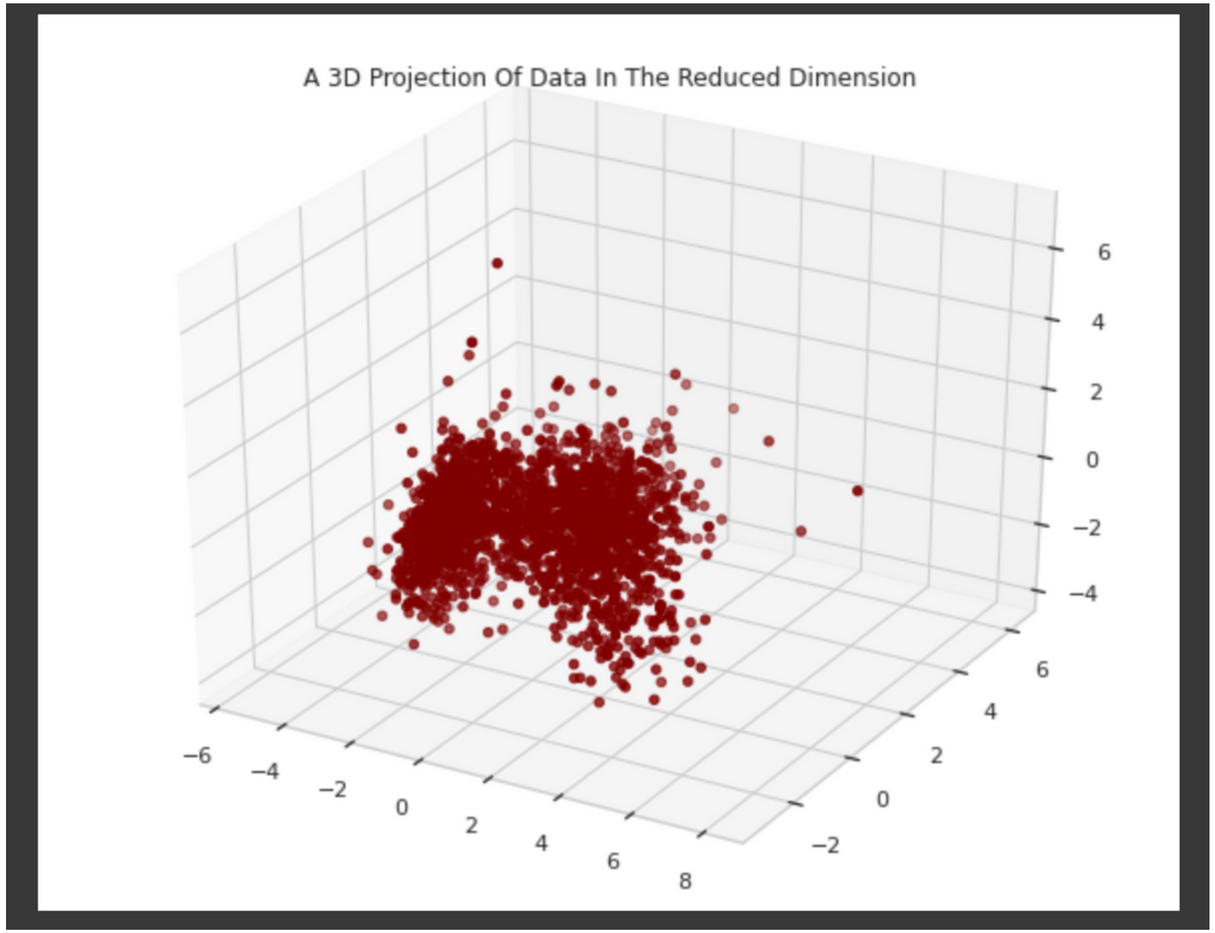
```
df = df.drop(["Kidhome", "Teenhome"], axis=1)
```

```
df = df.drop(["AcceptedCmp1", "AcceptedCmp2", "AcceptedCmp3",
              "AcceptedCmp4", "AcceptedCmp5", "Response"],
              axis=1)
```

Remaining Features

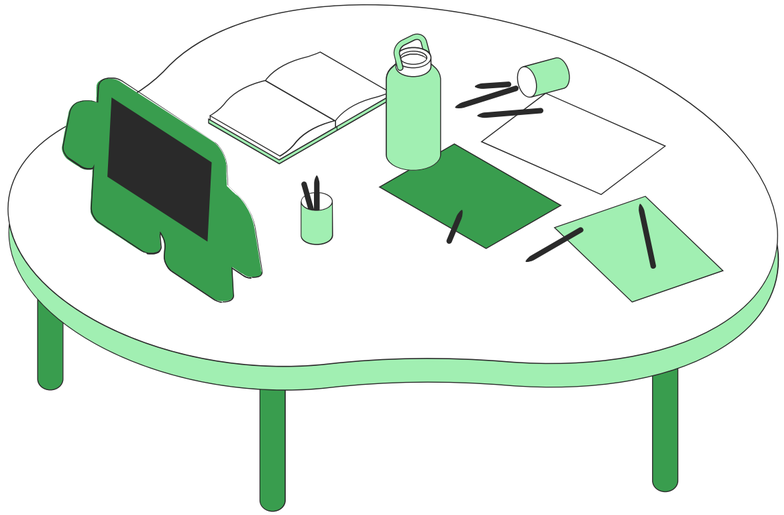
After Feature Engineering

Number of rows: 2240
Number of features: 18
New more meaningful features
All numerical features



Distance-Based Clustering

DISTANCE-BASED CLUSTERING



Hierarchical

Partitioning

Calculating Distances

Different Algorithms

Euclidean

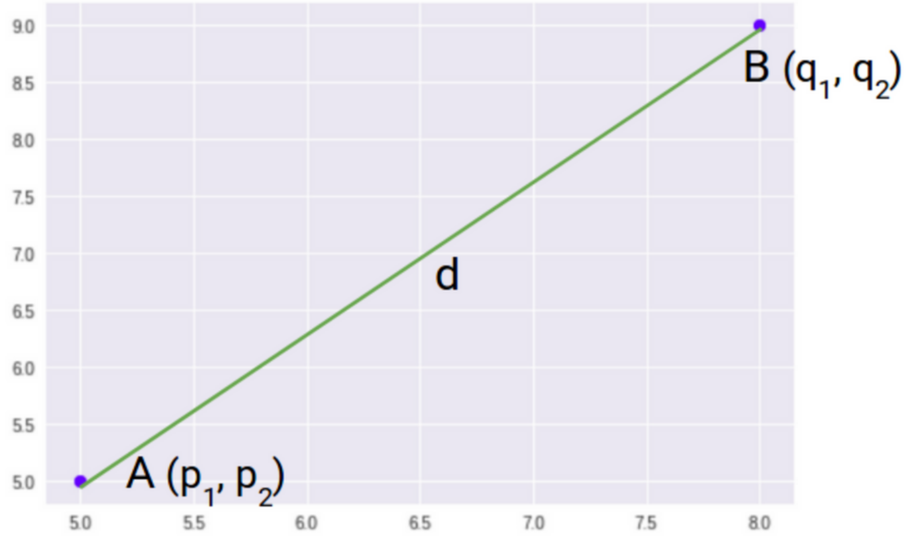
Manhattan

Minkowski

Cosine Similarity

Euclidean Distance

Definition



- It corresponds to the L2-norm of the difference between the two vectors.
- Euclidean distance in a 2-dimensional space is given as:

$$||x-y||_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$$

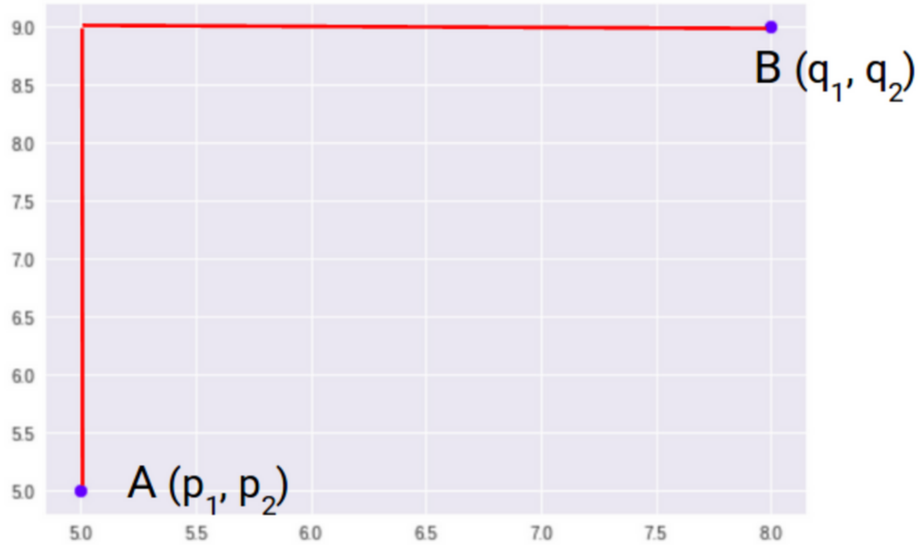
General Form:

$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$

Manhattan Distance

Definition

- Take the sum of absolute distances in both the x and y directions.
- Corresponds to the L1-norm of the difference between the two vectors. It can be computed as:
- So, the Manhattan distance in a 2-dimensional space is given



$$d = |p_1 - q_1| + |p_2 - q_2|$$

General Form:

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$

Minkowski Distance

Definition

- Minkowski Distance is the generalized form of Euclidean and Manhattan Distance.
- The General formula for Minkowski Distance is given as:

$$D = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

Cosine Similarity

Definition

Cosine similarity between two vectors corresponds to their dot product divided by the product of their magnitudes. If x and y are vectors, their cosine similarity is:

$$cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

Agglomerative Clustering

Agglomerative Clustering is a bottom-up strategy in which each data point is originally a cluster of its own, and as one travels up the hierarchy, more pairs of clusters are combined. In it, two nearest clusters are taken and joined to form one single cluster.

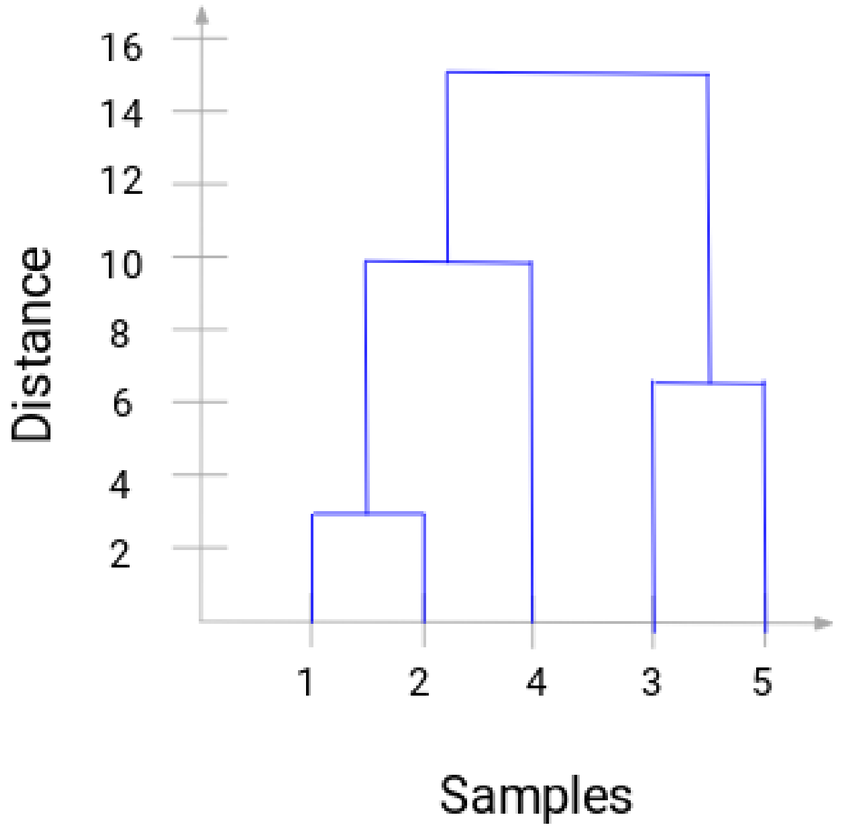
Divisive Clustering

The divisive clustering algorithm is a top-down clustering strategy in which all points in the dataset are initially assigned to one cluster and then divided iteratively as one progresses down the hierarchy.

Dendograms

Definition

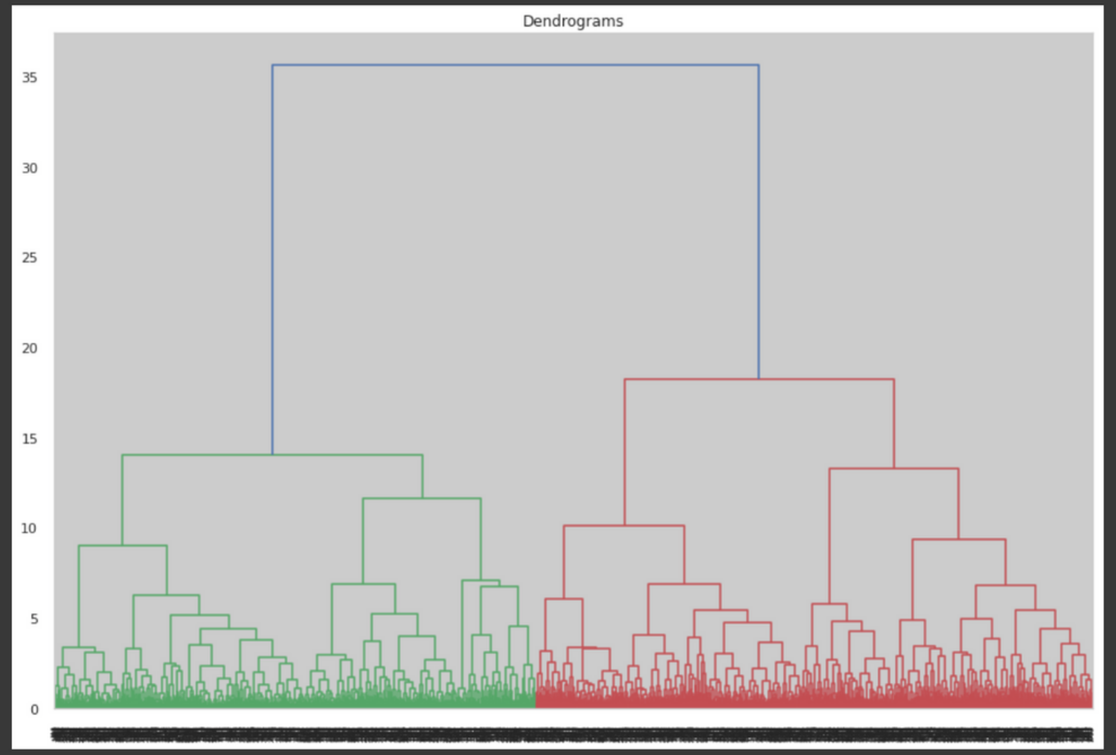
A dendrogram is a **visual representation** of the hierarchical connection between items. It's most often produced as a result of hierarchical clustering. A dendrogram's main purpose is to figure out the best approach to assign items to clusters according to similarities and dissimilarities and our desired problem.



Dendograms

Case Study Application

```
[ ]  import scipy.cluster.hierarchy as shc
     plt.figure(figsize=(15, 10))
     plt.title("Dendrograms")
     dend = shc.dendrogram(shc.linkage(data_scaled_copy, method='ward'))
```



Proximity Matrix

Definition

Proximity matrix simply stores the distances between each two points.

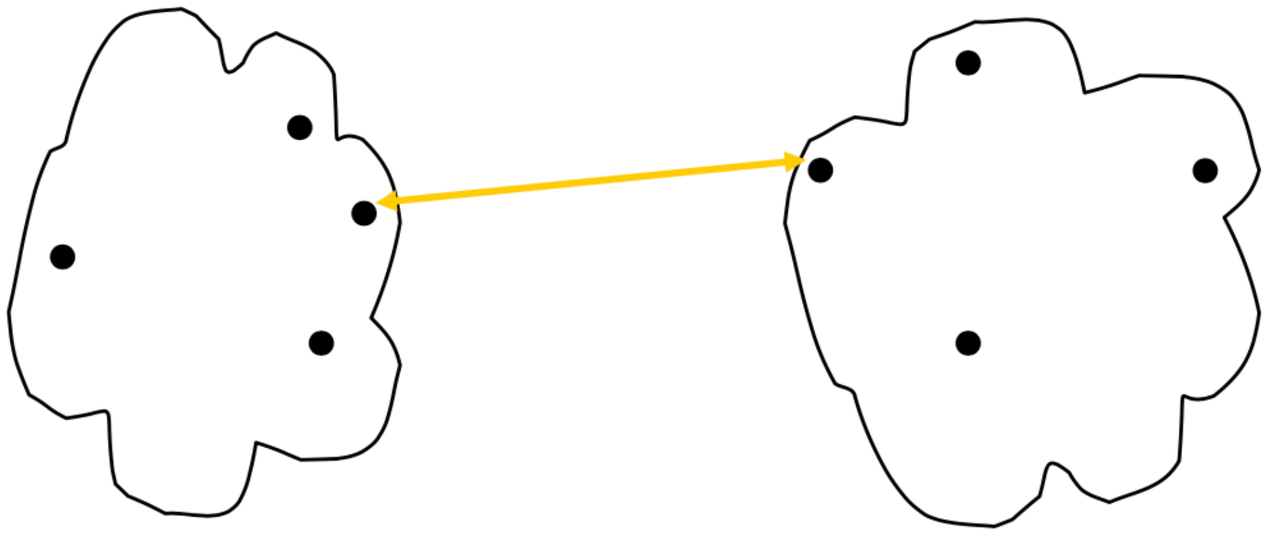
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

Proximity Matrix

Distance Calculation

Min



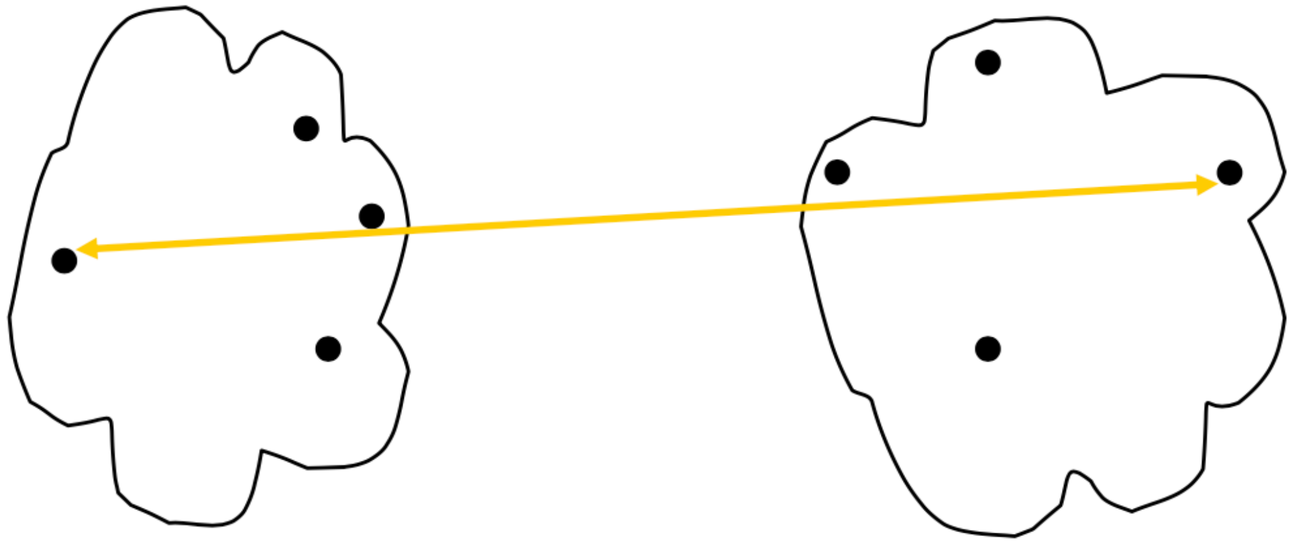
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

Proximity Matrix

Distance Calculation

Max



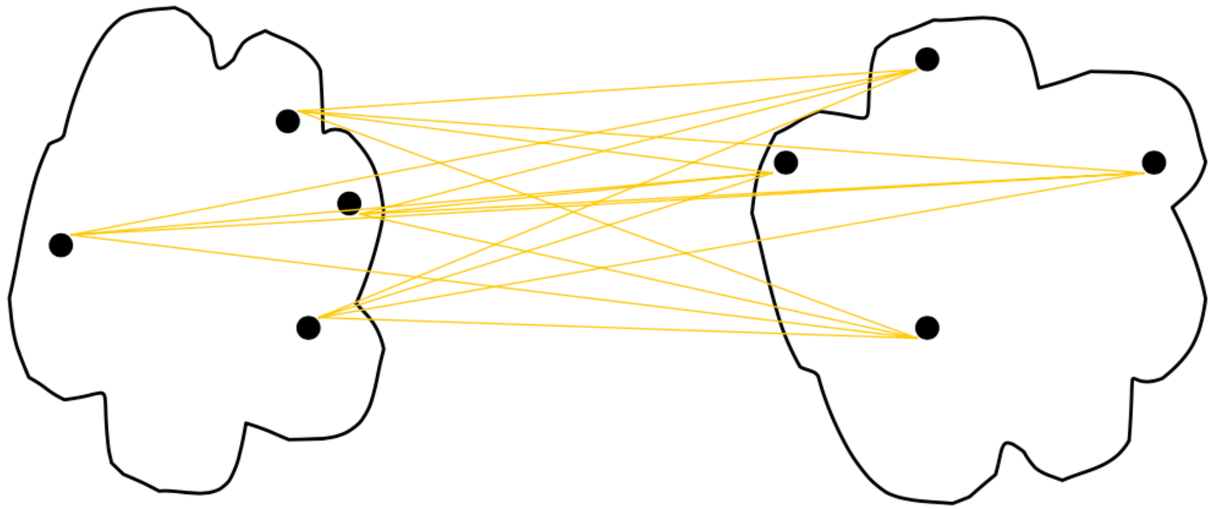
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

Proximity Matrix

Distance Calculation

Average



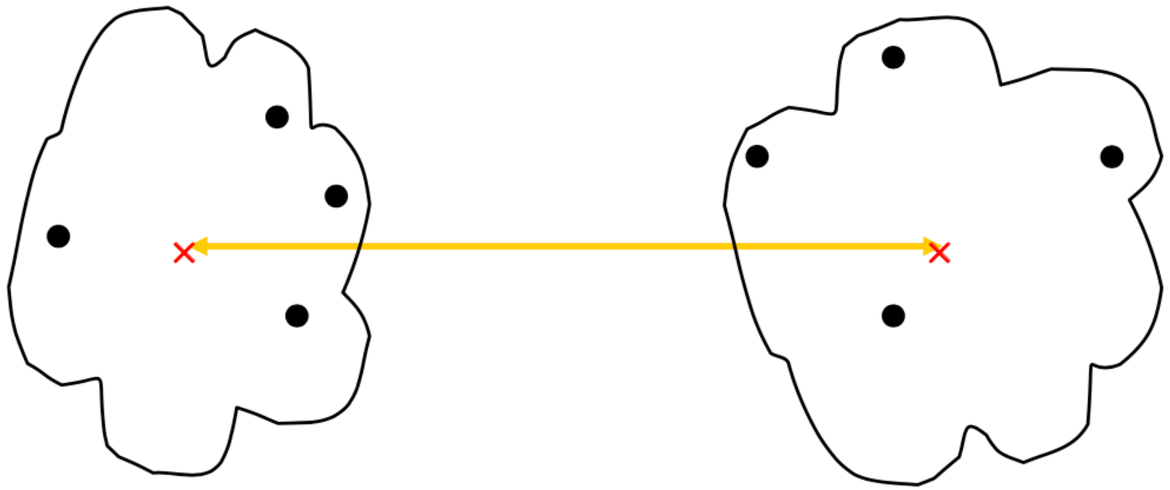
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

Proximity Matrix

Distance Calculation

Centroid



	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

Steps to Perform Agglomerative Clustering

Step 1

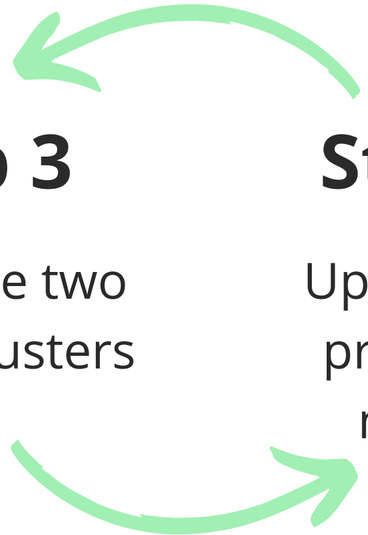
Compute the proximity matrix

Step 2

Let each data point be a cluster

Step 3

Merge the two closest clusters



Step 4

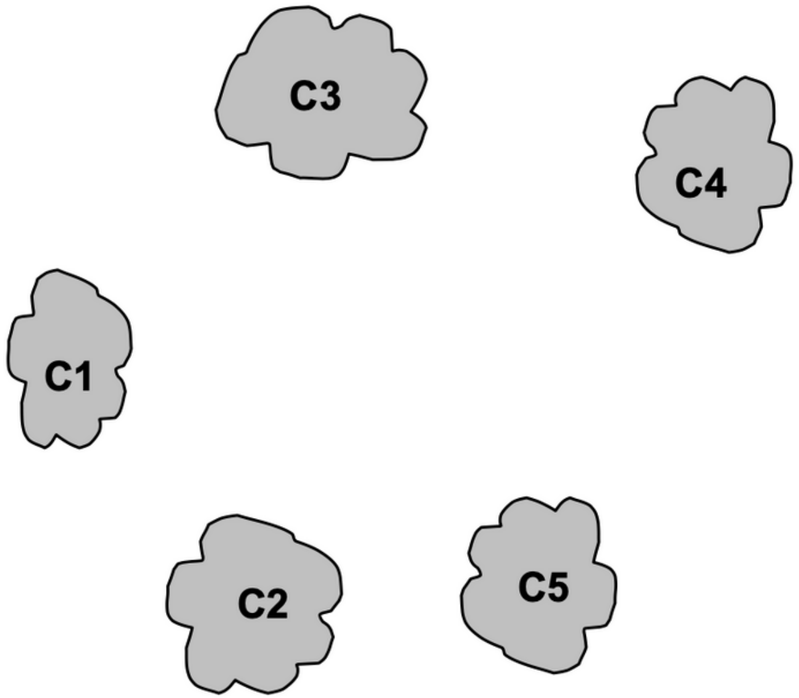
Update the proximity matrix

Until Step 5

Only a single cluster remains

Agglomerative

Steps

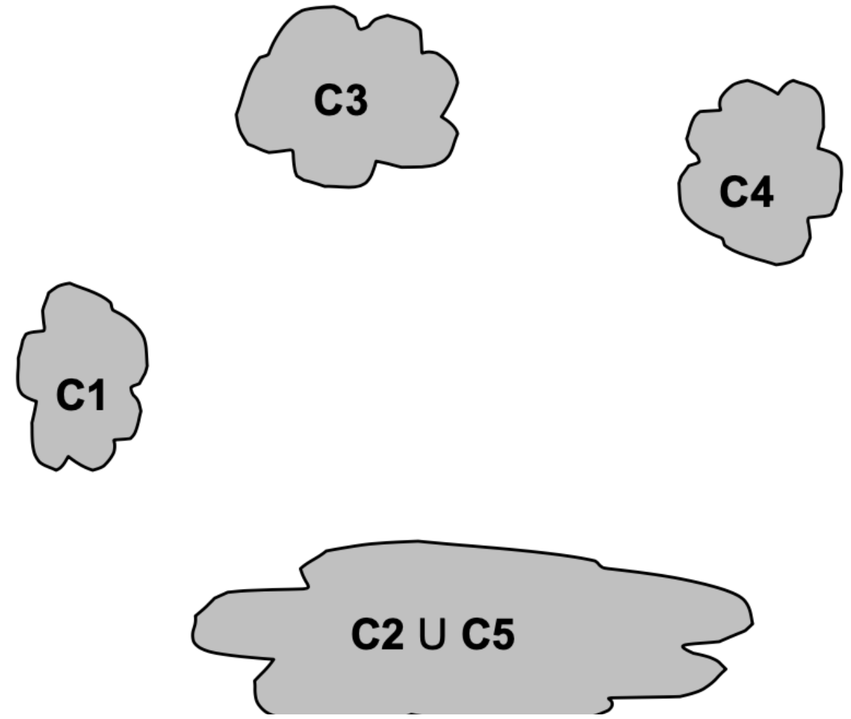


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

Agglomerative

Steps



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix

Agglomerative

Case Study Application

Agglomerative

Normal Numeric Features

```
[ ]  numeric_feats_copy = numeric_feats.copy()
```

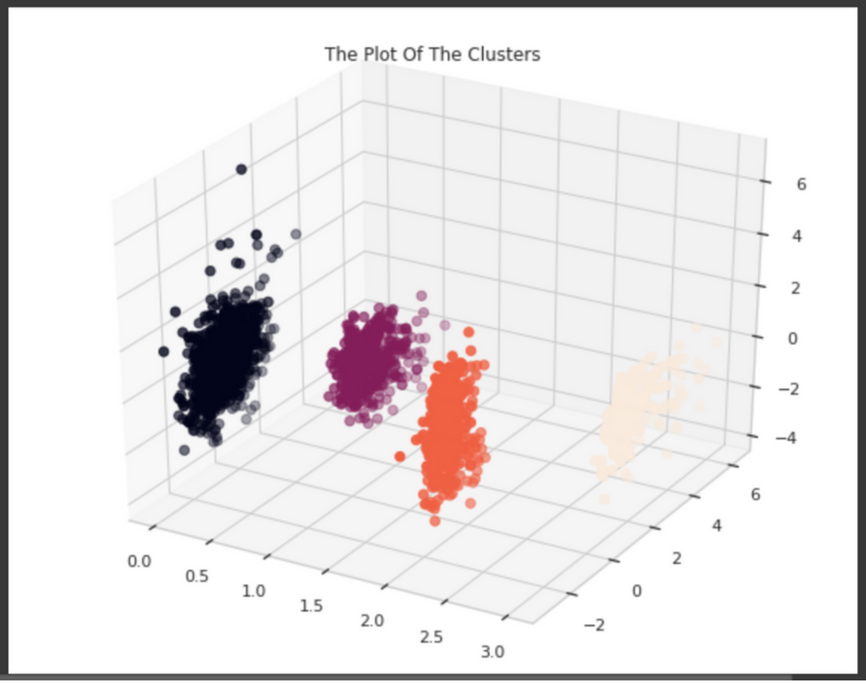
```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=4, affinity='euclidean', link
x= cluster.fit_predict(numeric_feats_copy)
x
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
[ ]  score = silhouette_score(numeric_feats_copy, x, metric='euclidean')
print('Silhouette Score: %.3f' % score)
```

```
Silhouette Score: 0.481
```

```
#Plotting the clusters
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds_copy["Label"], marker='o')
ax.set_title("The Plot Of The Clusters")
plt.show()
```



Silhouette Score

Definition

Silhouette score is used to evaluate the quality of clusters created using clustering algorithms in terms of how well samples are clustered with other samples that are similar to each other.

The value of Silhouette score varies from -1 to 1

- 1: Cluster is dense and well-separated than other clusters
- 0: Overlapping clusters with samples very close to the decision boundary of the neighbouring clusters
- -1: Indicate that the samples might have got assigned to the wrong clusters.

Silhouette Score

Algorithm

ai : is the intra cluster distance defined as the average distance to all other points in the cluster to which it's a part of

bi : is the inter cluster distance defined as the average distance to closest cluster of datapoint i except for that it's a part of

Overall Silhouette score for the complete dataset can be calculated as the mean of silhouette score for all data points in the dataset.

$$s_i = \frac{b_i - a_i}{max(b_i, a_i)}$$

Centroid-Based Clustering

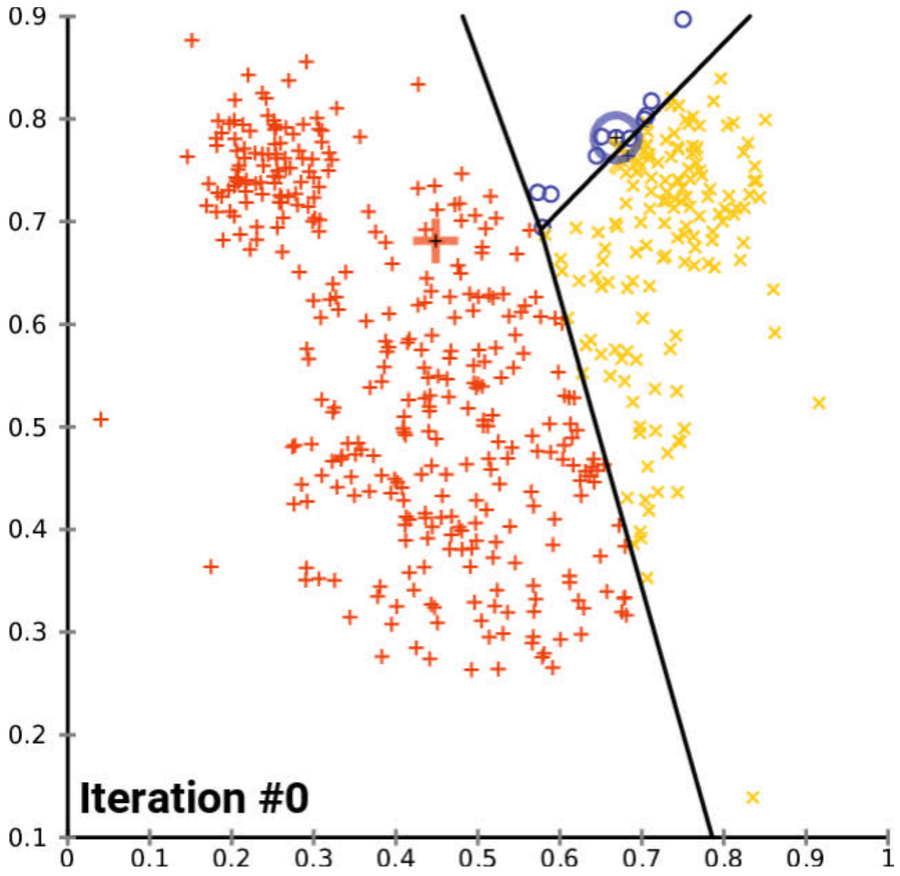
Centroid-Based Clustering

1- K Means

K-Means

Definition

- K-Means groups points into clusters by minimizing the distances between points and their cluster’s centroid .
- The centroid of a cluster is the mean of all the points in the cluster.



K-Means

Algorithm

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-Means

Parameters

- **K**: Chosen either randomly or if we have an idea about the nature of the dataset, we can set its value.

- **Centroids**: Initial centroids are often chosen randomly

Note: the value of K can be estimated by the Elbow Method

K-Means

Evaluation

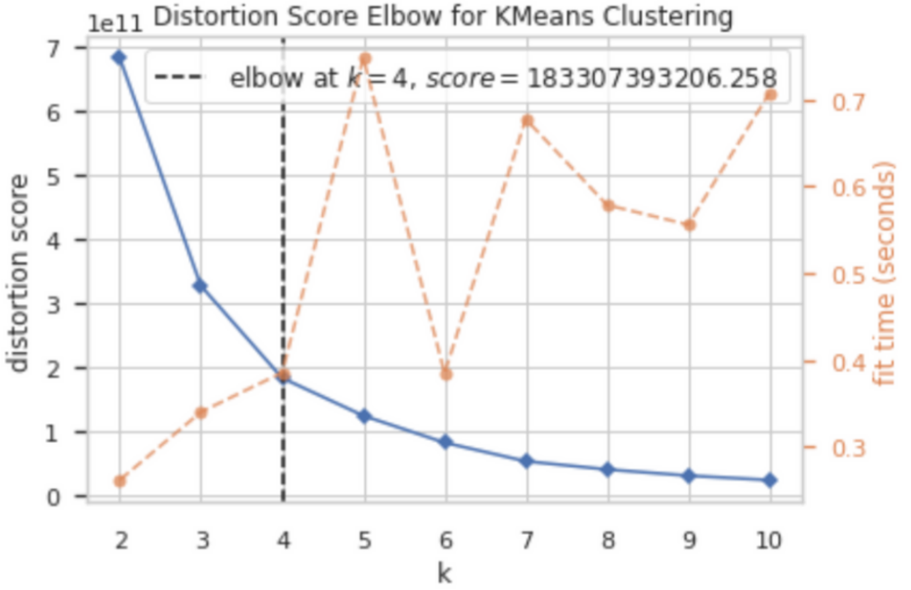
- Most common measure is Sum of Squared Error (SSE) also called Residual sum of squares (RSS)
- For each point, the error is the distance to the nearest cluster
- To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x) \qquad RSS = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

- **x** is a data point in cluster **Ci** and mi represents a certain cluster (C) and its centroid (m)
  - can show that mi corresponds to the center (mean) of the cluster
- One easy way to reduce SSE is to increase K (# of clusters)
  - k can increase untill each point is its own cluster; Error = 0

K-Means

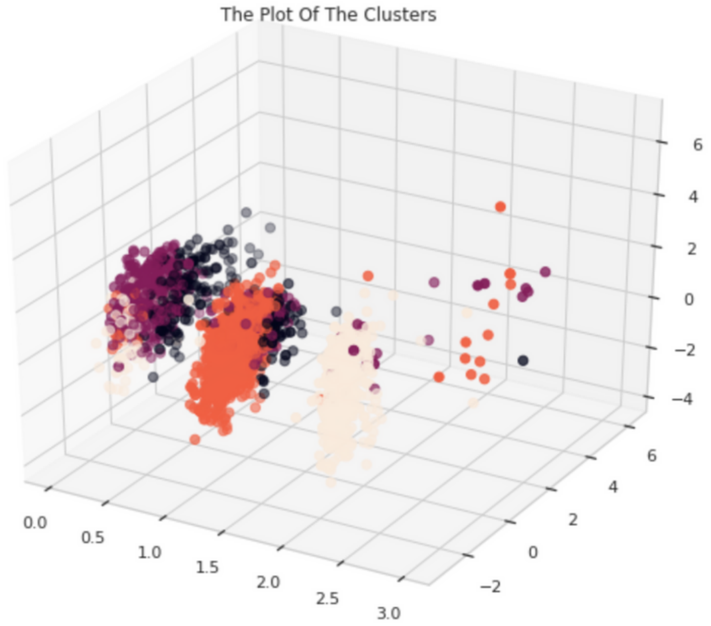
Elbow Method



- Used to determine the optimal number of clusters K using K-Means
- Plots the value of the cost/Error function produced by different values of k.
- If k increases, average Error will decrease
- Yet, the improvements in average error will decline as k increases.
- The value of k at which improvement in distortion declines the most is called The Elbow,
  - We should stop dividing the data into further clusters at this point

K-Means

Case Study Application



We Chose the number of clusters to be 4 based on the Elbow Method result above

```
[ ]  PCA_ds_copy= PCA_ds.copy()
```

```
[ ]  k_elbow = 4
     kmeans = KMeans(n_clusters=k_elbow)
```

```
[ ]  kmeans.fit(PCA_ds_copy)
```

```
     KMeans(n_clusters=4)
```

```
[ ]  kmeans.cluster_centers_
```

```
     array([[-1.78596065, -0.55013445, -0.00532644],
            [ 2.83316553, -0.823782  , -0.08750001],
            [-0.67835282,  2.20101085, -0.4514882 ],
            [ 1.06498061,  0.76316139,  0.3294378 ]])
```

```
[ ]  print(kmeans.labels_)
     print(len(kmeans.labels_))
```

```
     [3 0 3 ... 1 3 2]
     2240
```

```
from sklearn.metrics import silhouette_score
score = silhouette_score(PCA_ds_copy, kmeans.labels_, metric='euclidean')
print('Silhouette Score: %.3f' % score)
```

```
Silhouette Score: 0.438
```

Advantages

- Relatively simple to implement.
- Scales to large data sets.
- Guarantees convergence.

Disadvantages

- Being dependent on initial values.
- Centroids can be dragged by outliers.
- Outliers are forced to get into clusters
- Form Clusters of Convex shapes
  - Not good with clusters of irregular shapes

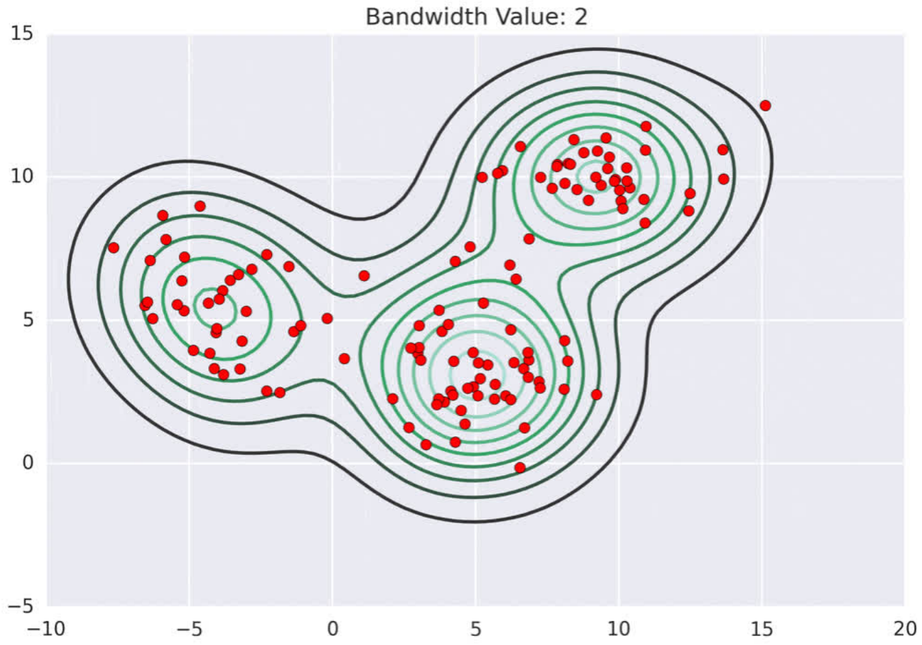
Centroid-Based Clustering

2- Mean-Shift

Mean-Shift

Definition

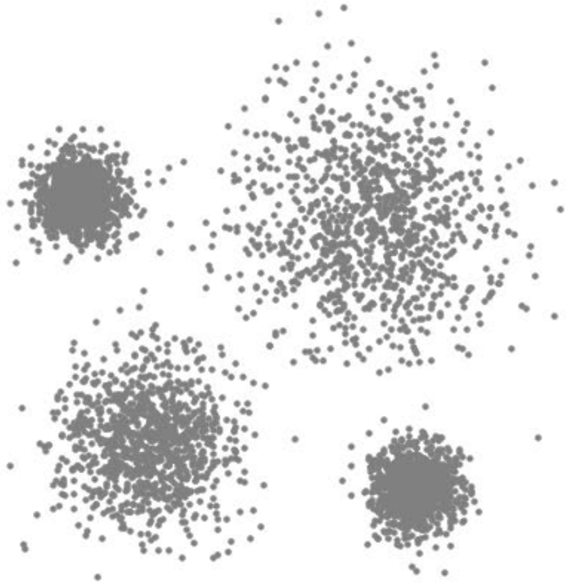
Meanshift is a centroid-based algorithm, where the **centroid of a cluster is the mode**. It assigns the data points to the clusters iteratively by shifting points towards the nearest mode (mode is the highest density of data points in the region, in the context of the Mean-shift).



- Also known as the Mode-seeking algorithm.
- Mean-shift algorithm has applications in the field of image processing and computer vision.

Mean-Shift

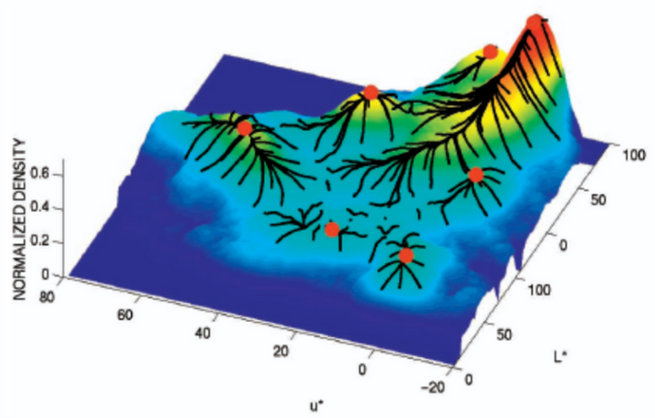
Algorithm



- Data points points that are close to each other, have higher densities (Hills)
- Goal: each point Climbs up the nearest Hill

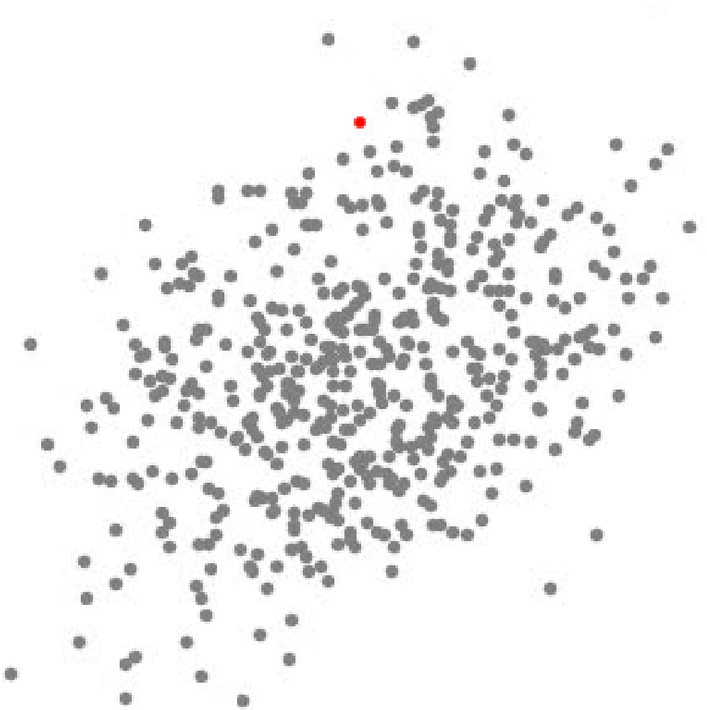
1. Start by each point as its own cluster
2. determine a value for the bandwidth
3. find local denstiy estimation within the bandwidth of the point
4. find the mode (region with highest density)
5. move the point to the mode.

- Points that walk to the same hill are clustered together



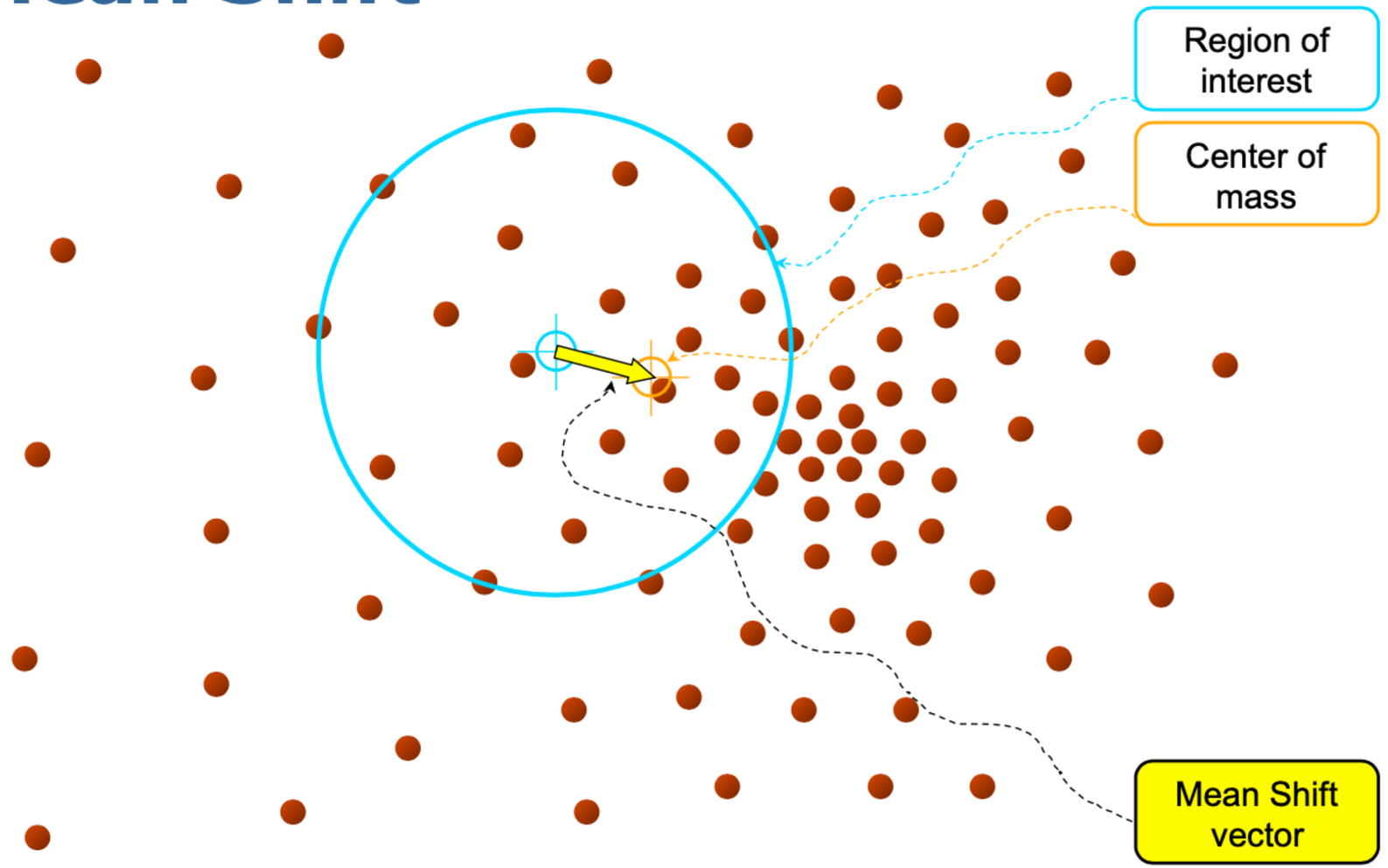
Mean-Shift

Algorithm



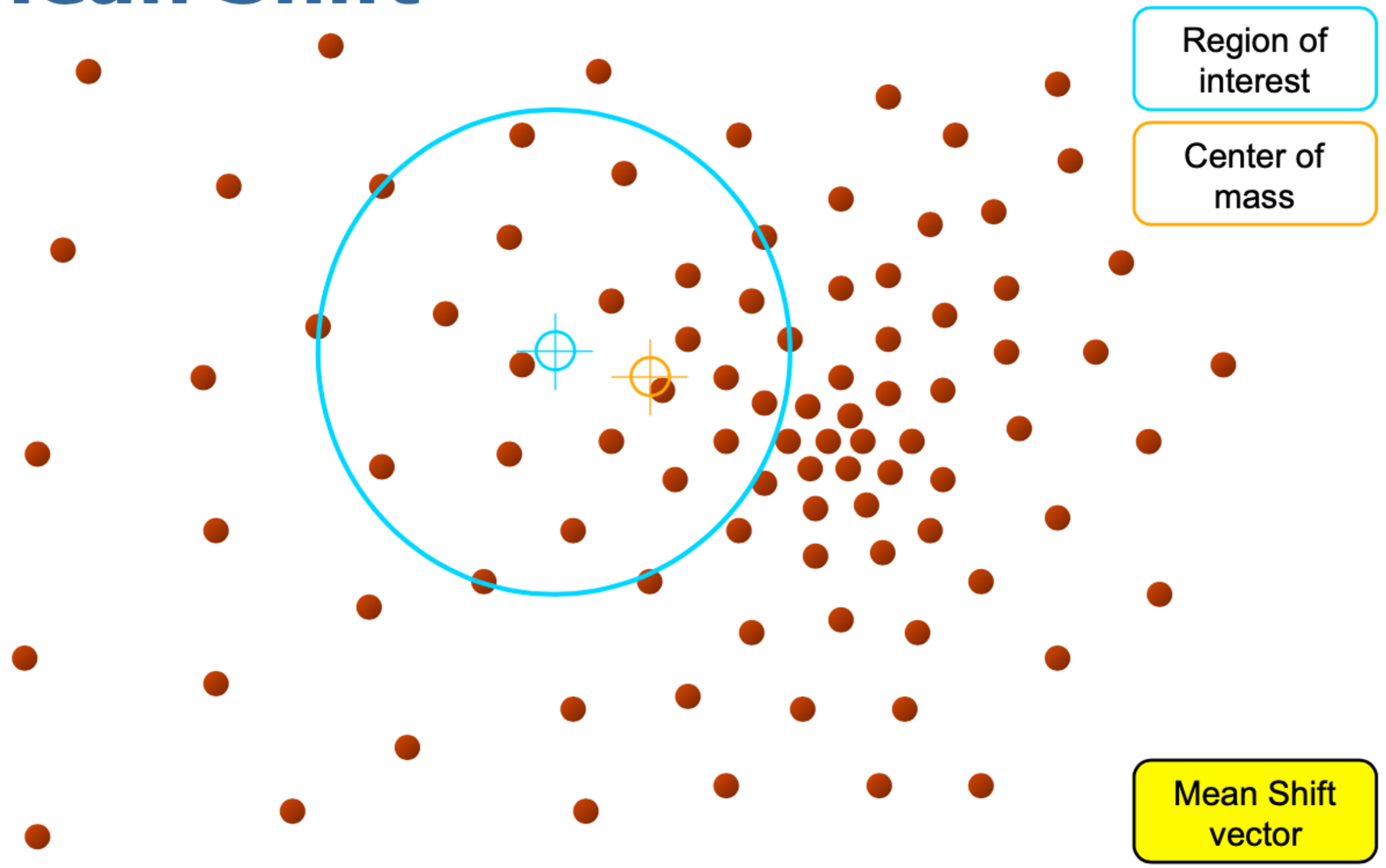
- Create a sliding window/cluster for each data-point
- Each of the sliding windows is shifted towards higher density regions by shifting their centroid (center of the sliding window) to the data-points' mean within the sliding window. This step will be repeated until no shift yields a higher density (number of points in the sliding window)
- Selection of sliding windows by deleting overlapping windows. When multiple sliding windows overlap, the window containing the most points is preserved, and the others are deleted.
- Assigning the data points to the sliding window in which they reside.

Mean Shift



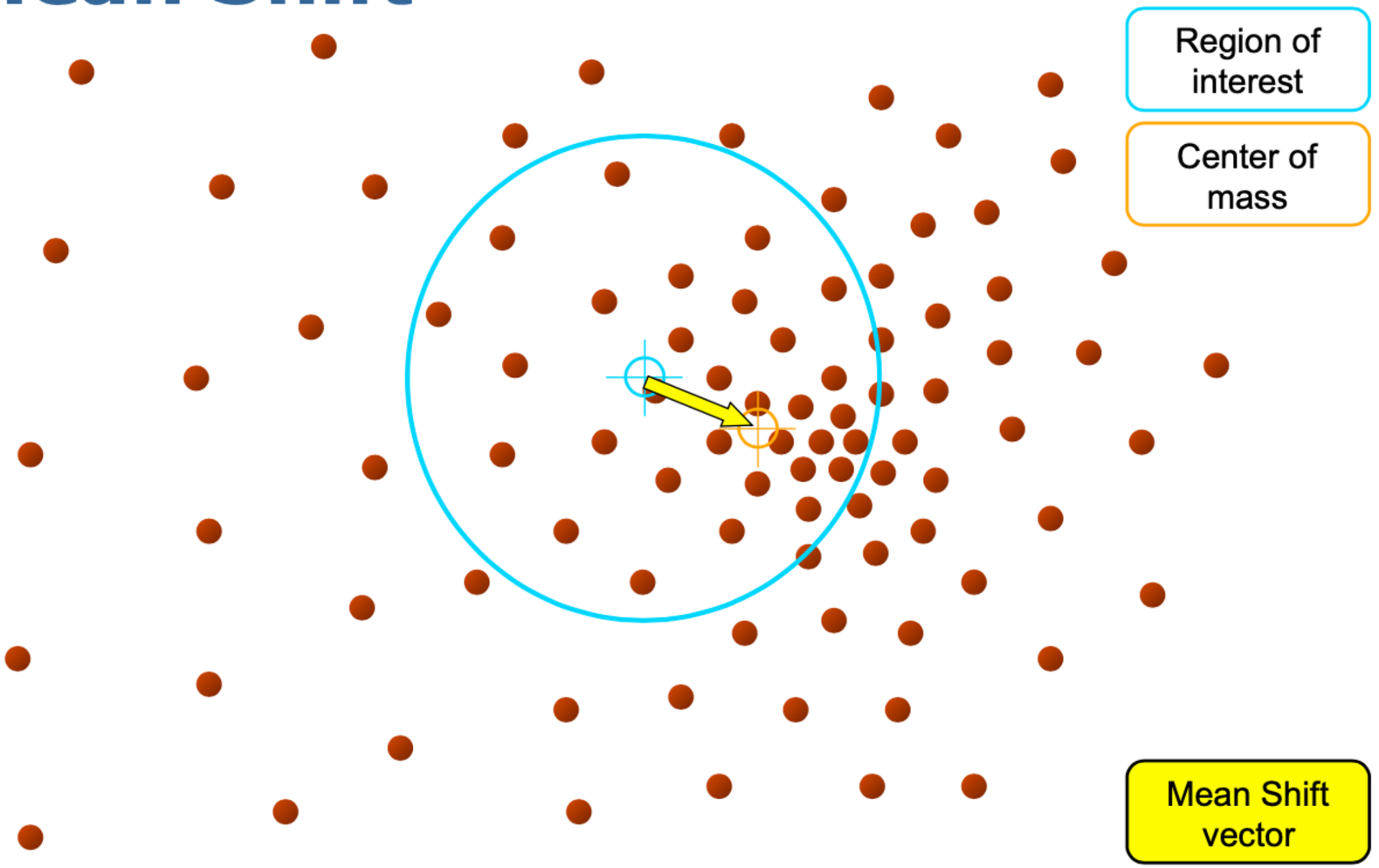
Slide courtesy: Ulkrainitz & Sarel

Mean Shift



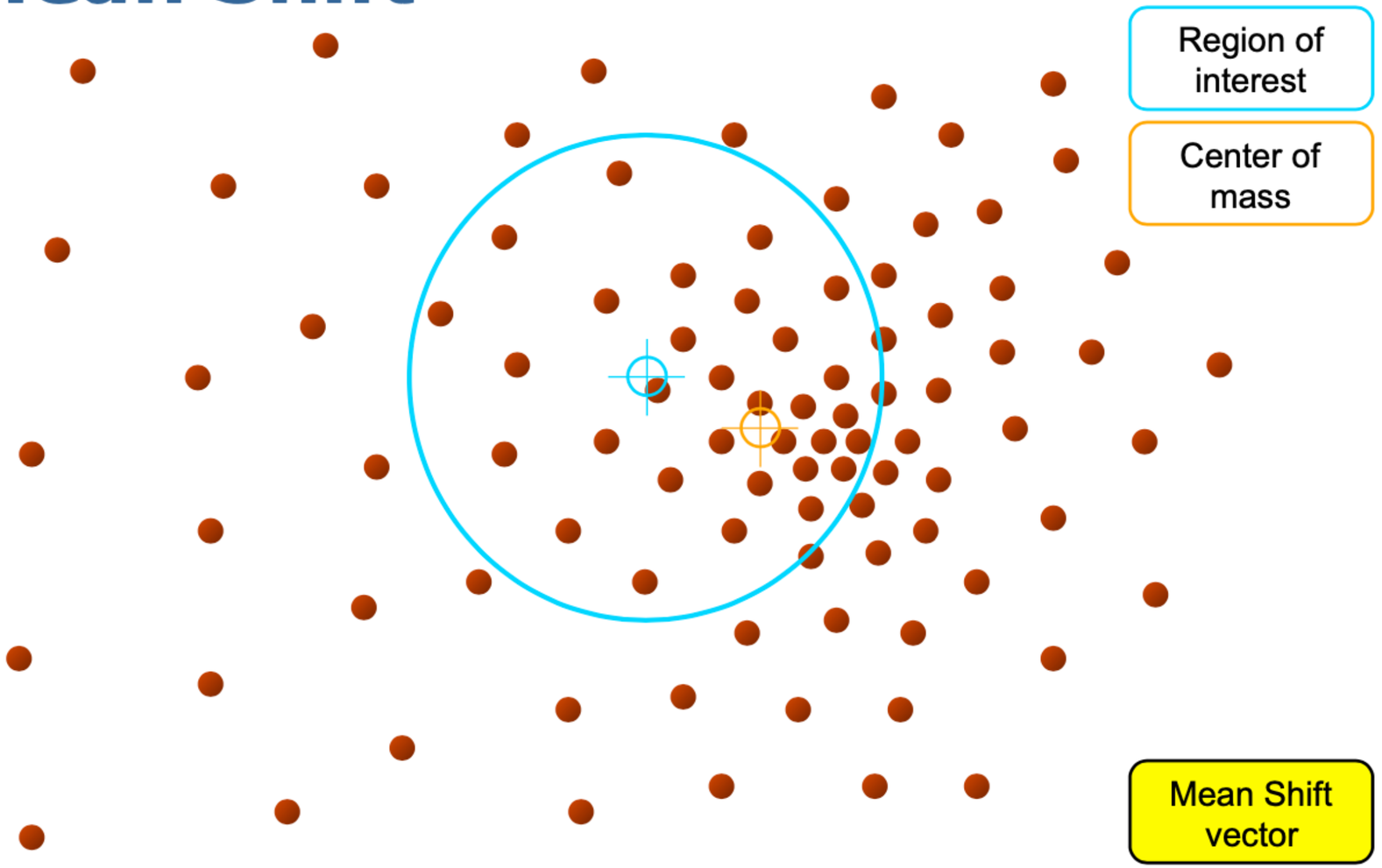
Slide courtesy: Ulkrainitz & Sarel

Mean Shift



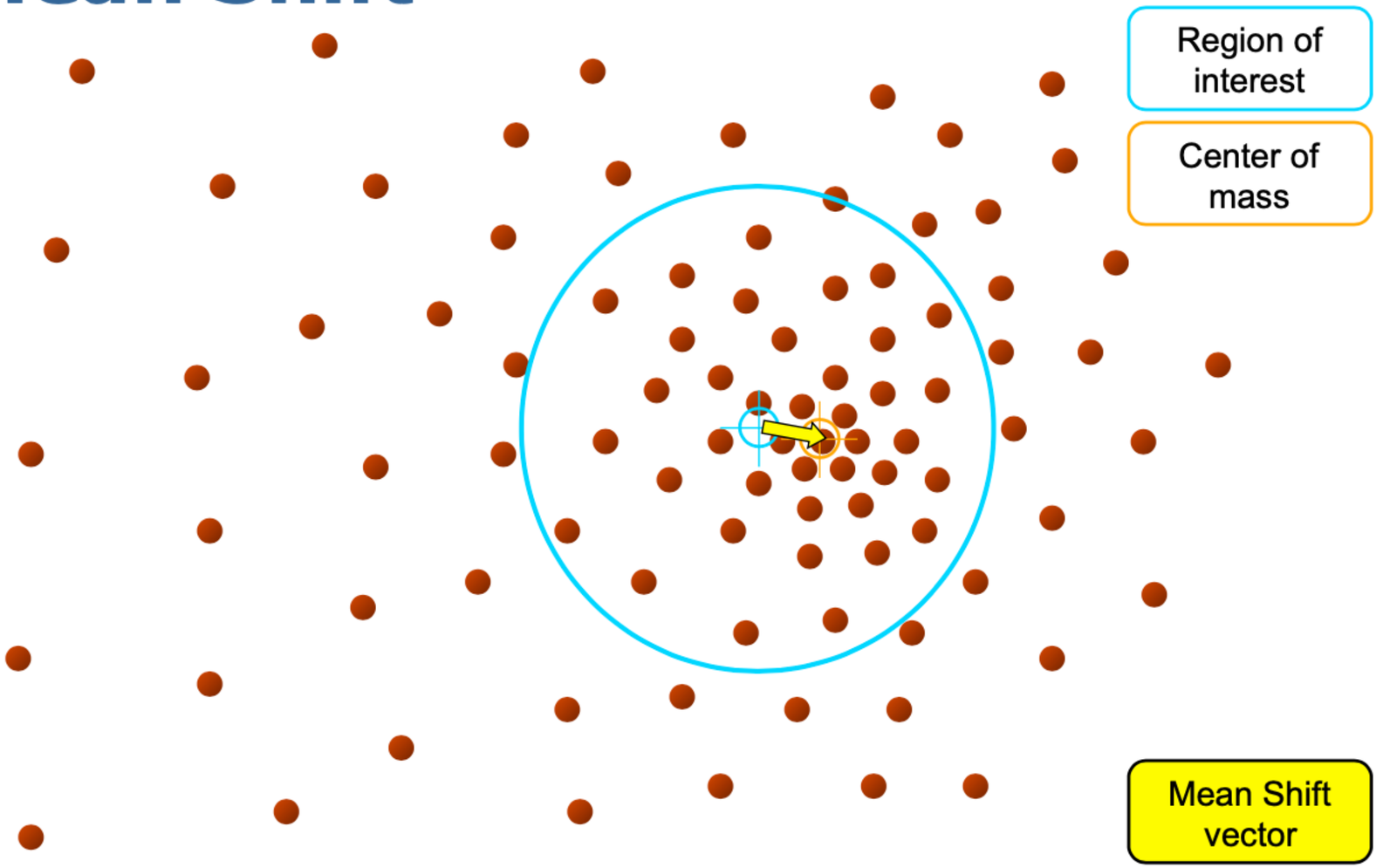
Slide courtesy: Ulkrainitz & Sarel

Mean Shift



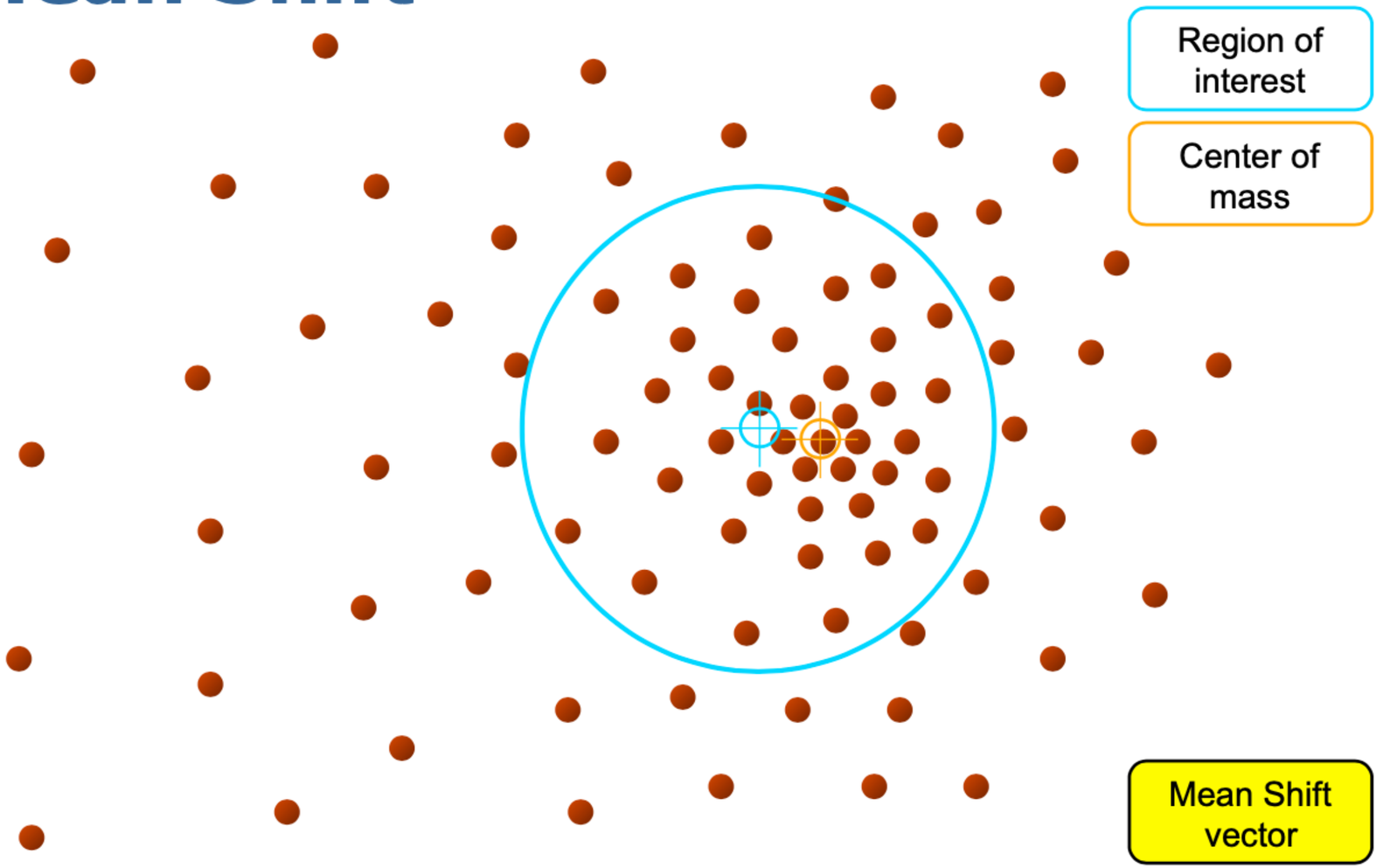
Slide courtesy: Ulkrainitz & Sarel

Mean Shift



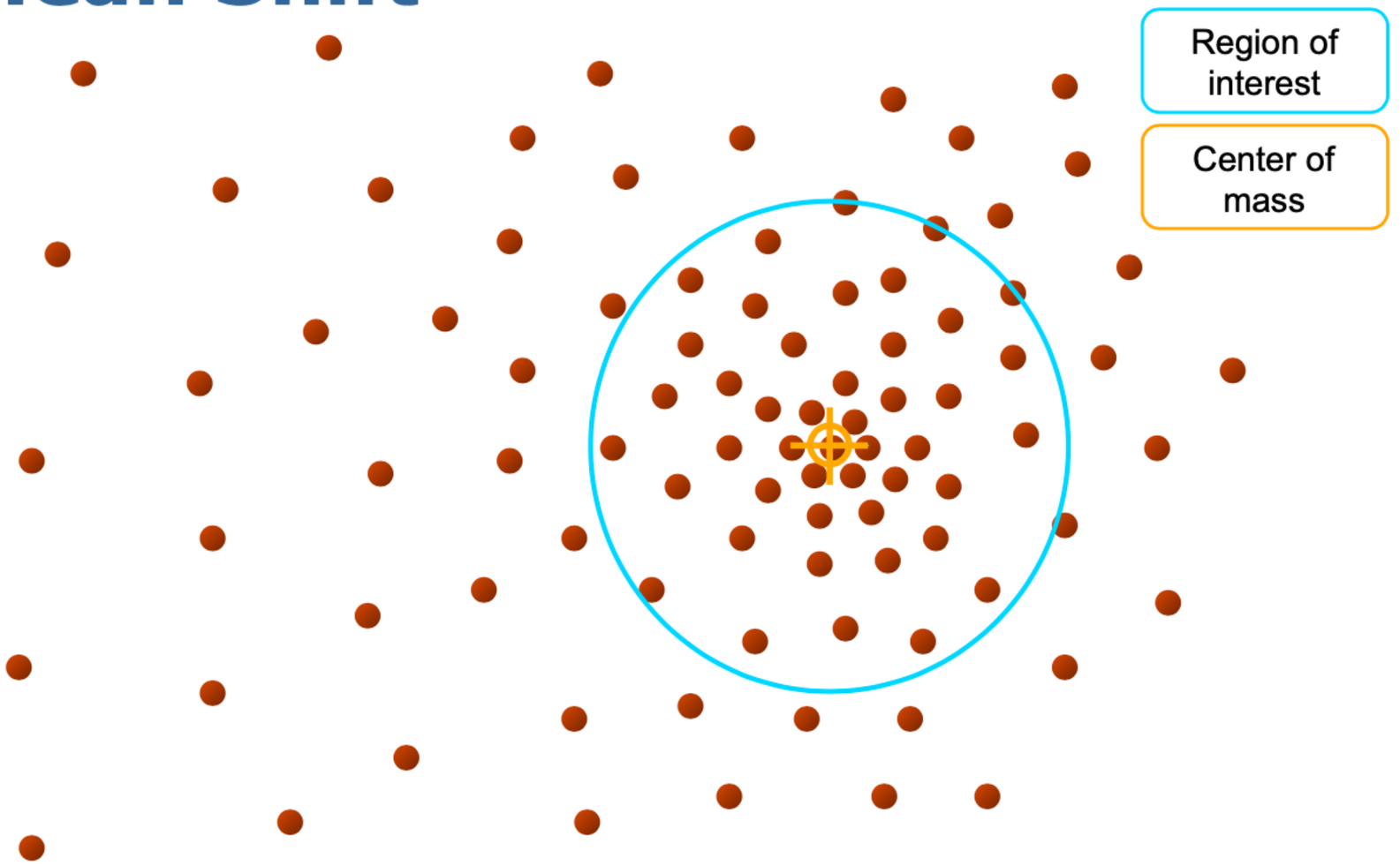
Slide courtesy: Ulkrainitz & Sarel

Mean Shift



Slide courtesy: Ulkrainitz & Sarel

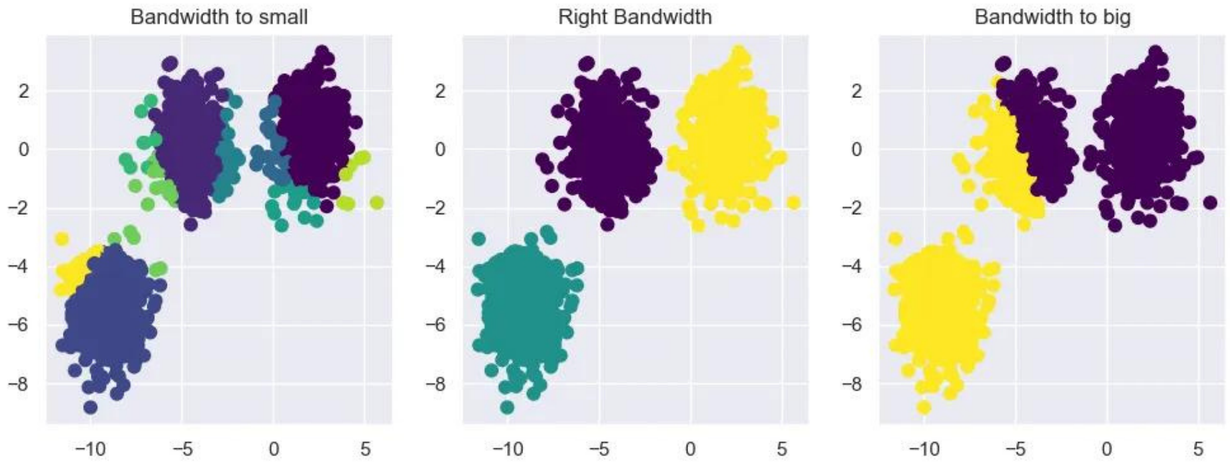
Mean Shift



Slide courtesy: Ulkrainitz & Sarel

Mean-Shift

How to Choose the Bandwidth



Depending on the bandwidth, the resulting clusters can look quite different. As an extreme case, imagine that we choose an extremely small bandwidth. This will result in each point having its own cluster. On the other hand, if we use a huge bandwidth, there will only be one cluster containing all the data-points.

Mean Shift

Case Study Application

```
from sklearn.cluster import MeanShift, estimate_bandwidth
# Compute clustering with MeanShift
# The following bandwidth can be automatically detected using
bandwidth = estimate_bandwidth(PCA_ds_copy, quantile=0.2, n_samples=500)

ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms.fit(PCA_ds_copy)
labels = ms.labels_
cluster_centers = ms.cluster_centers_

labels_unique = np.unique(labels)
n_clusters_ = len(labels_unique)

print("number of estimated clusters : %d" % n_clusters_)
```

```
number of estimated clusters : 6
```

```
[ ] score = silhouette_score(PCA_ds_copy, labels, metric='euclidean')
    print('Silhouette Score: %.3f' % score)
```

```
Silhouette Score: 0.366
```

Advantages

- It does not need to make any model assumption as like in K-means
- Models non-convex shape.
- It **only needs one parameter** named bandwidth which automatically determines the number of clusters.
- Outliers do not affect the centroids as K-means

Disadvantages

- Mean-shift algorithm does not work well in case of high dimension, where the number of clusters changes abruptly.

- It cannot differentiate between meaningful and meaningless modes.

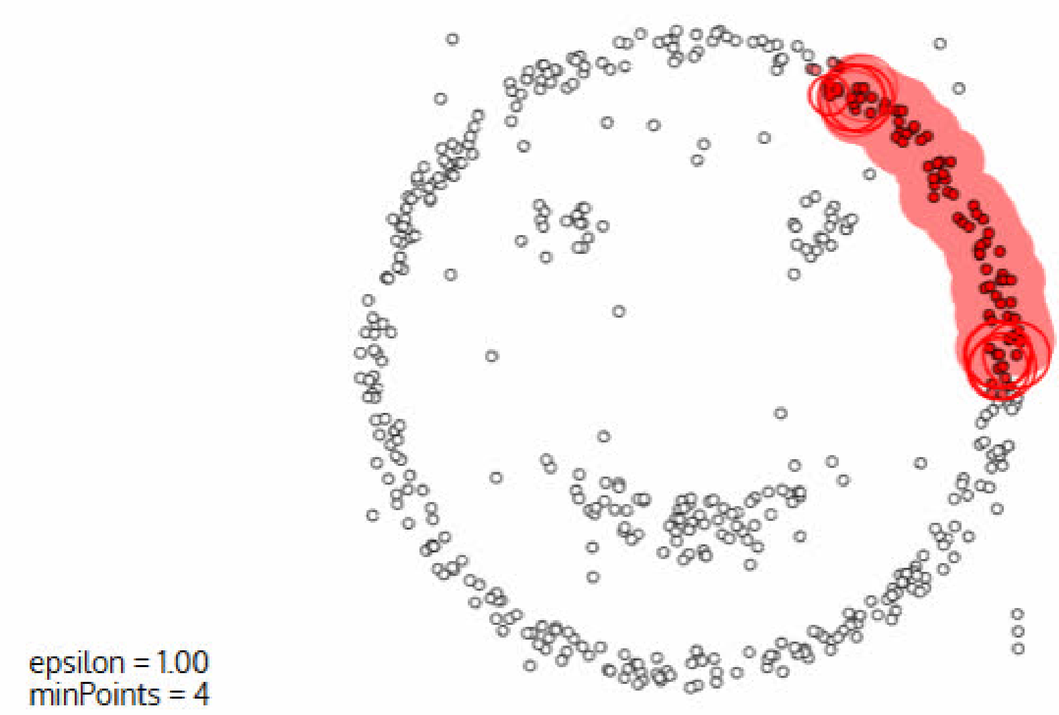
Density-Based Clustering

DBSCAN

Density-Based Clustering

Definition

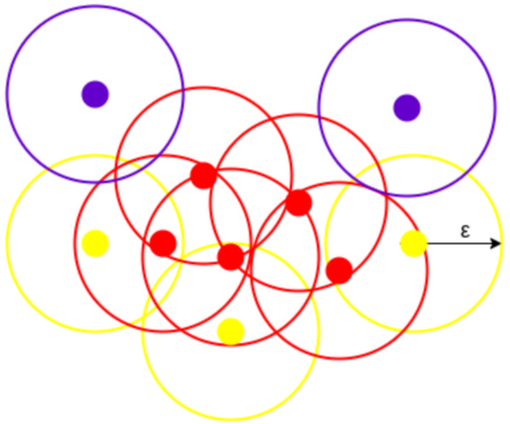
- DBSCAN is a density-based clustered algorithm similar to mean-shift, but with a couple of notable advantages.
- DBSCAN groups together points that are close to each other based on a distance measurement (usually Euclidean distance) and a minimum number of points. It also marks as outliers the points that are in low-density regions.



Density-Based Clustering

Paramaters

- **MinPoints**: This refers to the minimum number of points needed to construct a cluster. We consider MinPoints as a threshold for considering a cluster as a cluster. A cluster is only recognized if the number of points is greater than or equal to the MinPts.
- **Epsilon (Eps)**: This is the least distance required for two points to be termed as a neighbor.
    - If the distance between two points is utmost Eps, then we consider the two points to be neighbors.



Density-Based Clustering

Algorithm

1. DBSCAN begins with an arbitrary starting data point that has not been visited. The neighborhood of this point is extracted using a distance epsilon ε (All points which are within the ε distance are neighborhood points).
2. If there are a sufficient number of points within this neighborhood then the clustering process starts and the current data point becomes the first point in the new cluster. Otherwise, the point will be labeled as noise. In both cases that point is marked as “visited”.
3. For this first point in the new cluster, the points within its ε distance neighborhood also become part of the same cluster. This procedure of making all points in the ε neighborhood belong to the same cluster is then repeated for all of the new points that have been just added to the cluster group.
4. Repeat 2 and 3 until all points in the cluster are determined
    - i.e all points within the ε neighborhood of the cluster have been visited and labeled.
5. When done with the current cluster, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise. This process repeats until all points are marked as visited

Density-Based Clustering

How to choose parameter values

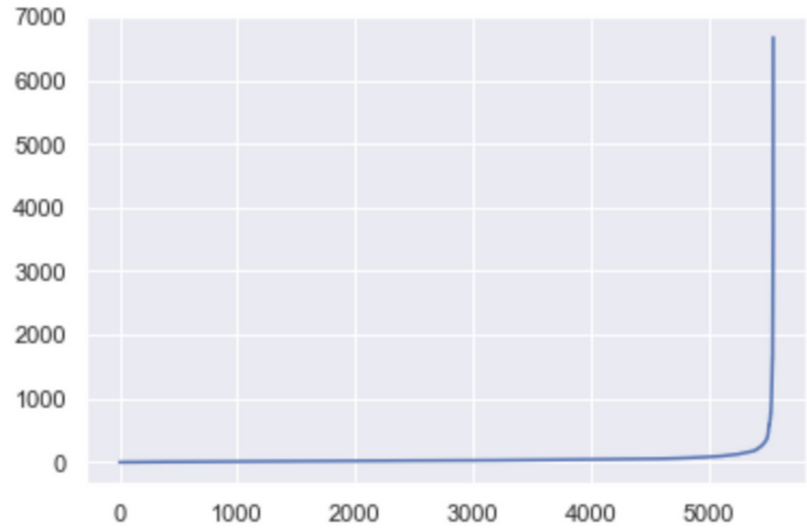
- **minPoints:** Larger values are usually better for large data sets or data sets with noise and will form more significant clusters. The minimum value for the minPoints must be 3, but the larger the data set, the larger the minPoints value that should be chosen.
  - Generally, MinPts should be **greater than** or equal to **the dimensionality of the data** set
  - If your data has more than 2 dimensions, **choose MinPts = 2*dim**

- **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.
  - generally, small eps values are preferable.
  - chosen based on how point of the dataset are distant from each other (k-distance graph)

Density-Based Clustering

K-distance Graph

- To determine the optimal ε value we calculate the average distance between each point and its k nearest neighbors.
  - Where k = the MinPts value you selected.
- The average k-distances are then plotted in ascending order on a k-distance graph.
- You'll find the optimal value for ε at the point of maximum curvature (Elbow).



Density-Based Clustering

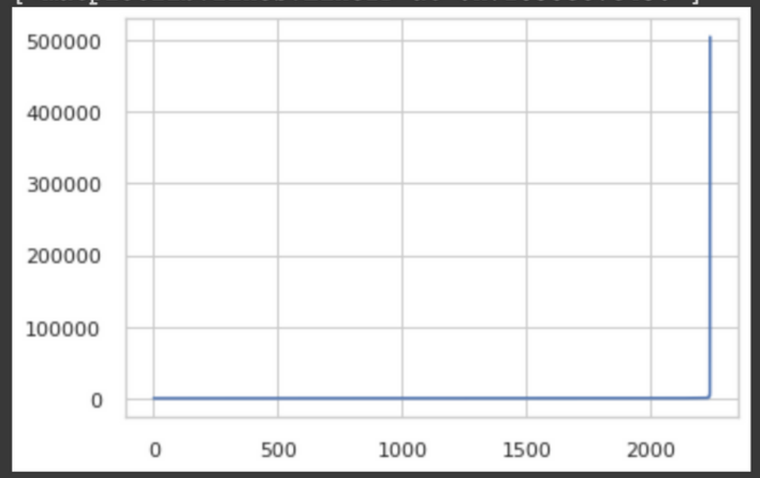
Case Study Application

```
[ ]  from sklearn.neighbors import NearestNeighbors
     from matplotlib import pyplot as plt

     neighbors = NearestNeighbors(n_neighbors=24)
     neighbors_fit = neighbors.fit(numeric_feats_copy)
     distances, indices = neighbors_fit.kneighbors(numeric_feats_copy)
```

```
     distances = np.sort(distances, axis=0)
     distances = distances[:,1]
     plt.plot(distances)
```

```
[<matplotlib.lines.Line2D at 0x7f853ec7e450>]
```




```
[ ]  from sklearn.cluster import DBSCAN
     # cluster the data into five clusters
     dbscan = DBSCAN(eps = 2050, min_samples = 24).fit(numeric_feats_copy) # fi
     labels = dbscan.labels_ # getting the labels
```

```
[ ]  score = silhouette_score(numeric_feats_copy, labels, metric='euclidean')
     print('Silhouette Score: %.3f' % score)
```

```
Silhouette Score: 0.617
```

Advantages

- It does not require a pre-set number of clusters at all
- It also identifies outliers as noises
  - unlike mean-shift which simply throws them into a cluster even if the data point is very different.
- It can find Convex/arbitrarily shaped clusters quite well.

Disadvantages

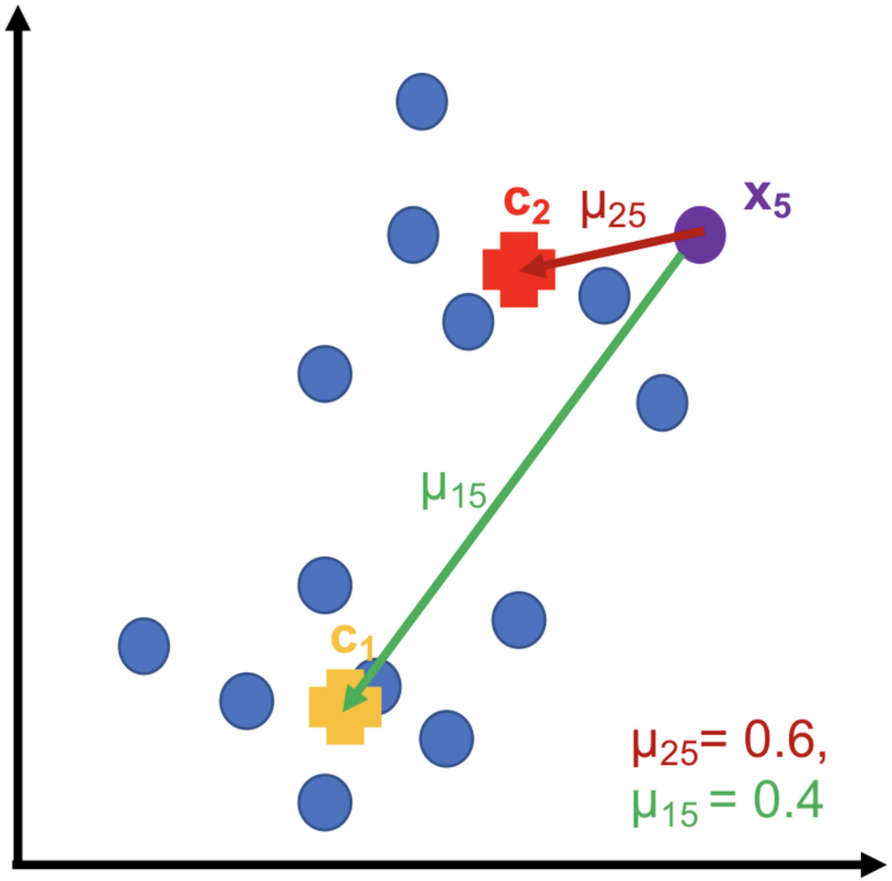
- DBSCAN doesn’t perform as well as others when the clusters are of varying density
- This is because the the best values of the epsilon(ε) and minPoints will vary from cluster to cluster.
- This drawback also occurs with very high-dimensional because ε becomes challenging to estimate.

Fuzzy Clustering

Fuzzy C-Means

Definition

"Fuzzy" means "**not sure**", which indicates that it's a **soft clustering method**. "C-means" means c cluster centers, which only replaces the "K" in "K-means" with a "C" to make it look different.



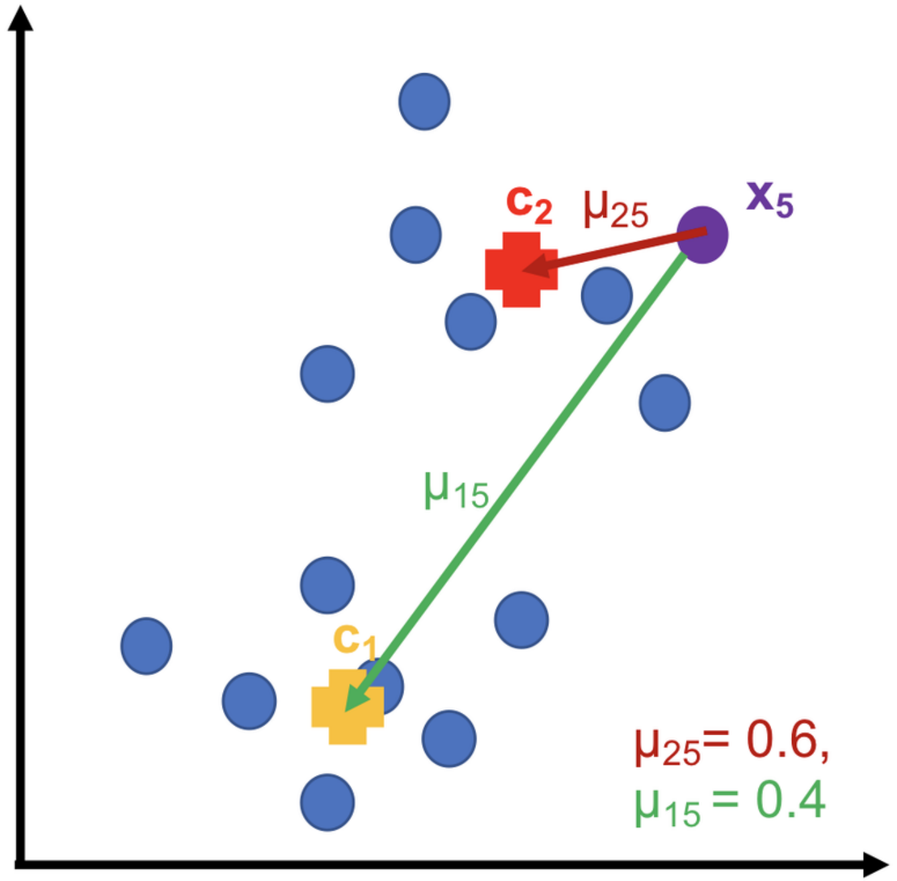
Fuzzy C-Means

Understand the Parameters

- μ_ij - membership value, is the probability that the **jth data point** belongs to the **ith cluster** and it is constrained to that the sum of μ_ij over C cluster centers is 1 for every data point j

- c_i - the center of the ith cluster (the same dimension with X).

- m (hyperparameter) - the **fuzzifier**, which controls how fuzzy the cluster boundary should be. (m = 2)



Fuzzy

Objective Function

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^{m} \|x_j - c_i\|^2$$

The objective function of FCM. (Image by author)

Fuzzy

Full FCM Algorithm

1. Initialize the matrix of $\boldsymbol{\mu}_{ij}$, **U**
2. Update $\mathbf{c}_i$:

$$c_i = \frac{\sum_{j=1}^{N} \mu_{ij}^m x_j}{\sum_{j=1}^{N} \mu_{ij}^m}$$

3. Update $\boldsymbol{\mu}_{ij}$:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_j - c_i\|}{\|x_j - c_k\|}\right)^{\frac{2}{m-1}}}$$

4. If the change of U between two iterations is very small (predefined cutoff), then stop; otherwise, go to step #2.

Algorithm of FCM (Image by author)

Fuzzy Clustering

Case Study Application

```
[ ]  import numpy as np
     from fcmeans import FCM
     numpy_array = numeric_feats_copy.to_numpy()
     model = FCM(n_clusters=5)
     model.fit(numpy_array) ## X, numpy array. rows:samples columns:features
```

```
[ ]  centers = model.centers
     labels = model.predict(numpy_array)
```

```
[ ]  labels
```

```
     array([3, 3, 4, ..., 3, 4, 3])
```

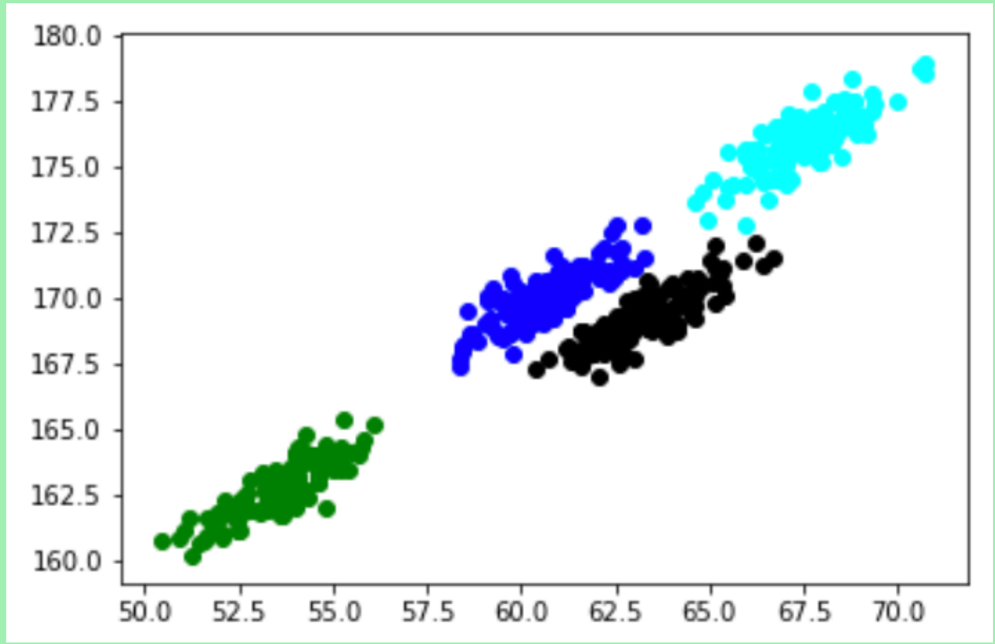
```
[ ]  score = silhouette_score(numeric_feats_copy, labels, metric='euclidean')
     print('Silhouette Score: %.3f' % score)
```

```
     Silhouette Score: 0.514
```

Distribution-Based Clustering

Distribution-Based Clustering

- Gaussian Mixture Models (GMMs) assume that there are a certain number of **Gaussian distributions**, and each of these distributions represent a cluster. Hence, a Gaussian Mixture Model tends to **group** the data points **belonging to a single distribution together**.
- Gaussian Mixture Models are **probabilistic models** and use the **soft clustering** approach for distributing the points in different clusters



E-step:

For each point x_i, calculate the probability that it belongs to cluster/distribution $c_1, c_2, \ldots c_k$. This is done using the below formula:

$$r_{ic} = \frac{\text{Probability Xi belongs to c}}{\text{Sum of probability Xi belongs to } c_1, c_2, .. c_k} = \frac{\pi_c \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i \; ; \; \mu_{c'}, \Sigma_{c'})}$$

This value will be high when the point is assigned to the right cluster and lower otherwise.

M-step:

Post the E-step, we go back and update the Π, μ and Σ values. These are updated in the following manner:

1. The new density is defined by the ratio of the number of points in the cluster and the total number of points:

$$\Pi = \frac{\text{Number of points assigned to cluster}}{\text{Total number of points}}$$

2. The mean and the covariance matrix are updated based on the values assigned to the distribution, in proportion with the probability values for the data point. Hence, a data point that has a higher probability of being a part of that distribution will contribute a larger portion:

$$\boldsymbol{\mu} = \frac{1}{\text{Number of points assigned to cluster}} \sum_i r_{ic} x_i$$

$$\Sigma_c = \frac{1}{\text{Number of points assigned to cluster}} \sum_i r_{ic} (x_i - \boldsymbol{\mu}_c)^T (x_i - \boldsymbol{\mu}_c)$$

Distribution-Based Clustering

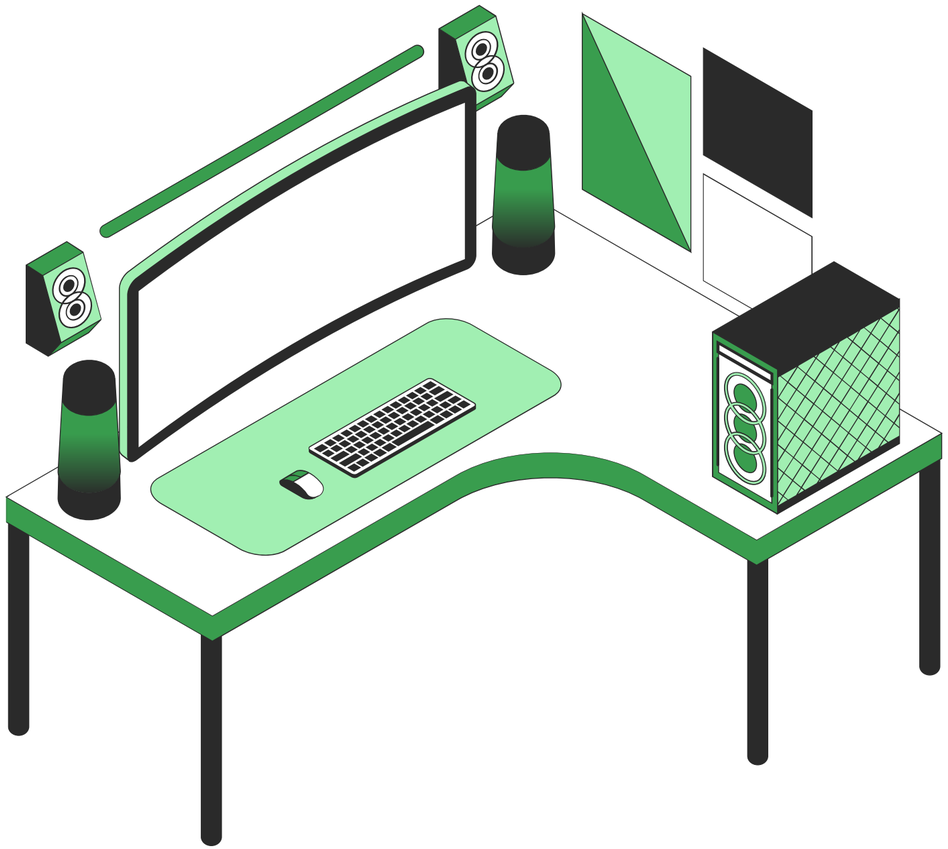
Case Study Application

```
[ ]  # training gaussian mixture model
     from sklearn.mixture import GaussianMixture
     gmm = GaussianMixture(n_components=4)
     gmm.fit(PCA_ds_copy)
     #predictions from gmm
     labels = gmm.predict(PCA_ds_copy)
     labels
```

```
array([1, 0, 1, ..., 1, 2, 2])
```

```
score = silhouette_score(PCA_ds_copy, labels, metric='euclidean')
print('Silhouette Score: %.3f' % score)
```

```
Silhouette Score: 0.254
```



Do you have any questions?

We hope you learned something new.

REFERENCES

https://www.analyticssteps.com/blogs/what-hierarchical-clustering-machine-learning
https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_clustering_algorithms_hierarchical.htm
https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering/
https://www.youtube.com/watch?v=-ETQ97mXXF0
https://www.researchgate.net/post/How-can-I-test-the-performance-of-a-clustering-algorithm
https://www.kaggle.com/general/19741
https://en.wikipedia.org/wiki/Cluster_analysis#Centroid-based_clustering
https://www.qualtrics.com/experience-management/research/cluster-analysis/
https://www.linkedin.com/learning/data-science-foundations-data-mining/clustering-data?autoSkip=true&autoplay=true&resume=false&u=57686545
https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/
https://www.geeksforgeeks.org/ml-hierarchical-clustering-agglomerative-and-divisive-clustering/
https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/?ref=lbp
https://developers.google.com/machine-learning/clustering
https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68
https://www.tutorialspoint.com/machine_learning_with_python/clustering_algorithms_mean_shift_algorithm.htm
https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68
https://analyticsindiamag.com/hands-on-tutorial-on-mean-shift-clustering-algorithm/
https://ml-explained.com/blog/mean-shift-explained
https://www.youtube.com/watch?v=3ERPpzrDkVg
https://www.youtube.com/watch?v=0h3bLXme46I

REFERENCES

https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/
https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/?ref=lbp
https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/#h2_3
https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/
https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam
https://vitalflux.com/kmeans-silhouette-score-explained-with-python-example/
https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning
https://www.guru99.com/supervised-vs-unsupervised-learning.html
https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms/
https://www.upgrad.com/blog/clustering-and-types-of-clustering-methods/
https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68
https://www.analyticssteps.com/blogs/5-clustering-methods-and-applications
https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/
https://www.youtube.com/watch?v=-ETQ97mXXF0
https://uc-r.github.io/hc_clustering
https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_clustering_algorithms_hierarchical.htm
https://www.analyticssteps.com/blogs/what-hierarchical-clustering-machine-learning
https://towardsdatascience.com/fuzzy-c-means-clustering-with-python-f4908c714081