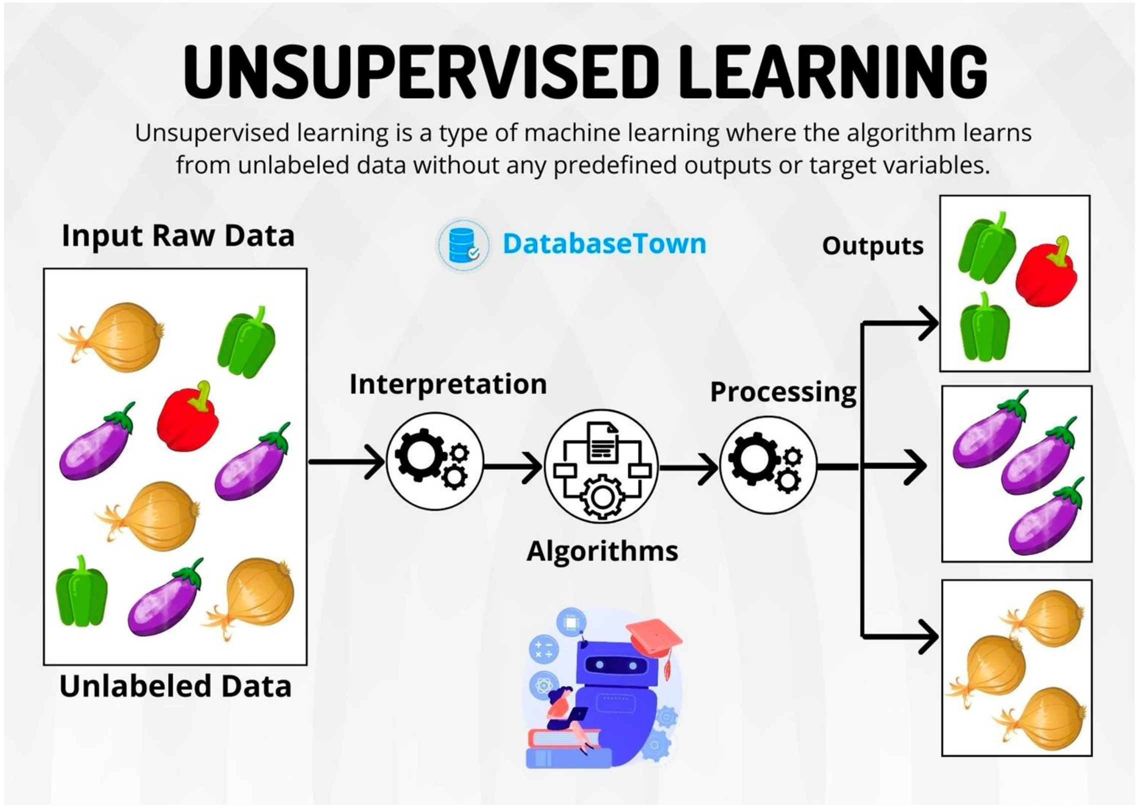
Unsupervised learning: clustering, mixture models and other alternatives

BIOS 7747 Flipped Classroom 3

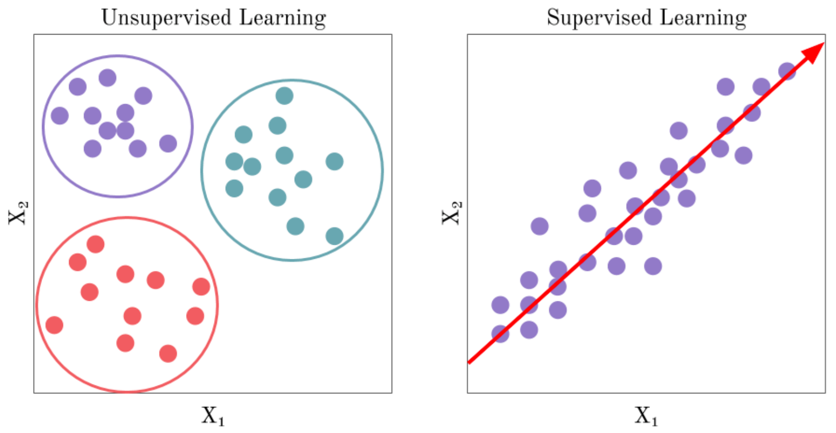
Presented by Jacob Krol, Jade Young, Karla Vela Lopez, Keenan Manpearl, and Kewalin Samart

What is unsupervised learning?


UNSUPERVISED LEARNING
Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data without any predefined outputs or target variables.
Input Raw Data
DatabaseTown
Outputs
Interpretation
Processing
Algorithms
Unlabeled Data

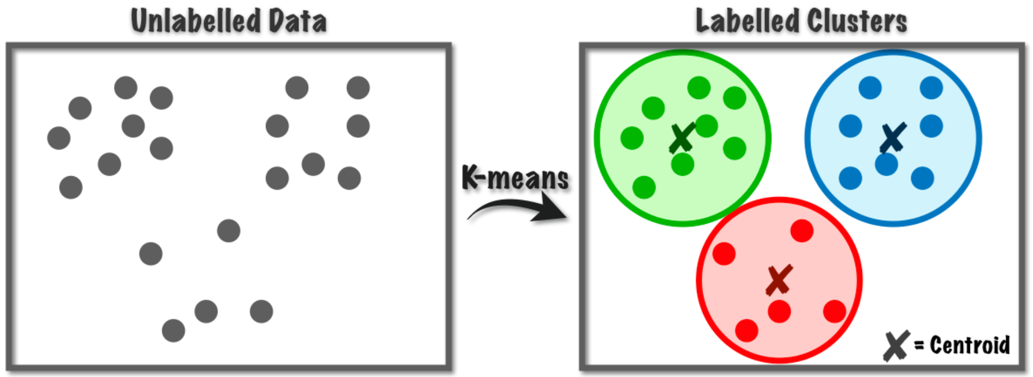
What is clustering?

- Clustering is based on dividing the data into several groups that are similar to one another.
- These algorithms help find hidden patterns in the data.



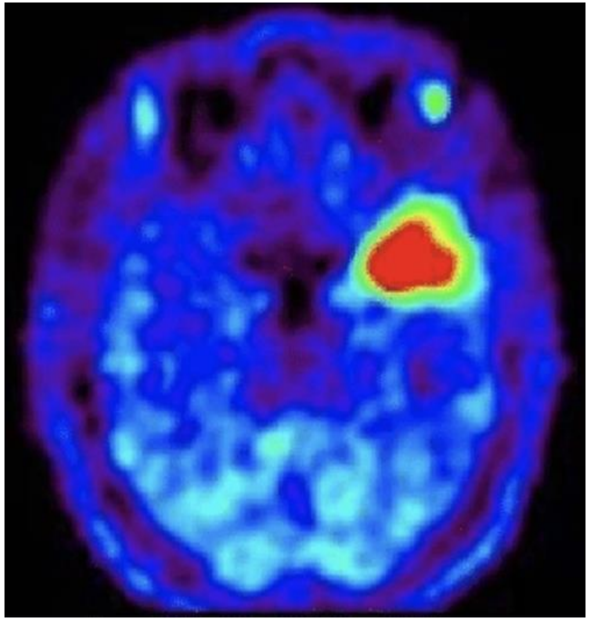
What is K-means Clustering?

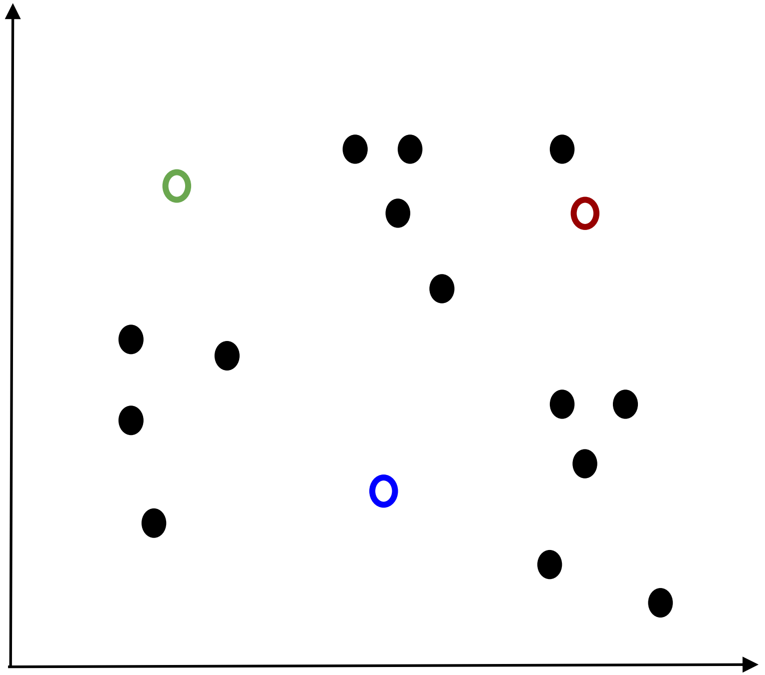
- K-means is a centroid-based clustering algorithm.
- This clustering process results in clustering data points based on similar features.
- The goal is to identify the K number of groups in the dataset.[1]



Example time!

Image Segmentation of Brain tumors





Step 1: Choose K
K=3

Step 2: Initialize centroids

- Important to initialize centroids as close to the optimal cluster centers as possible.

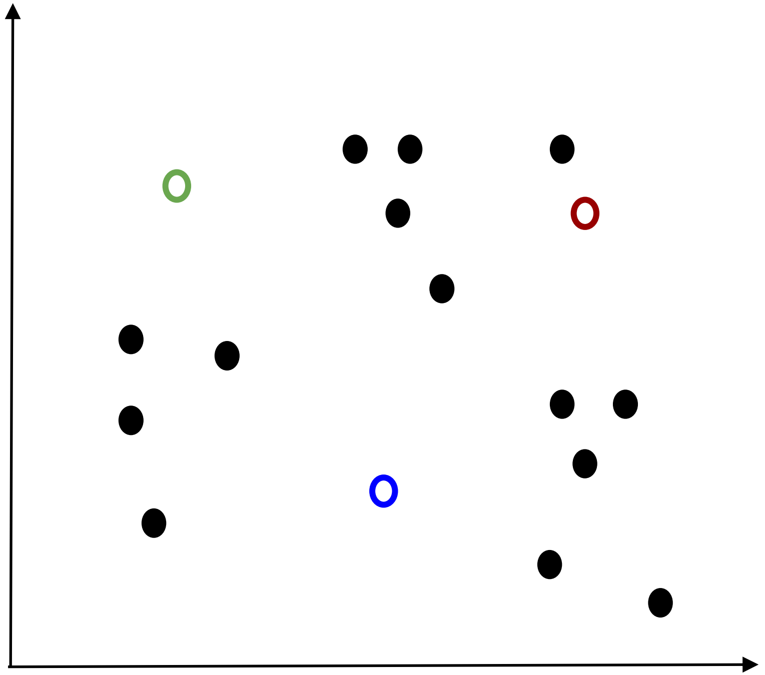
Centroid initialization methods

- Random Data Points
  - As the dataset's complexity grows, suboptimal initialization can result in non-optimal solutions and time-consuming optimization.

- Naive sharding
  - Creating a composite value by summing all attributes for each data point, sorting the data, dividing it into "shards," find the average of each shard, and using the average values of attributes within each shard as the initial centroids
  - Resulting in faster and often more effective clustering.

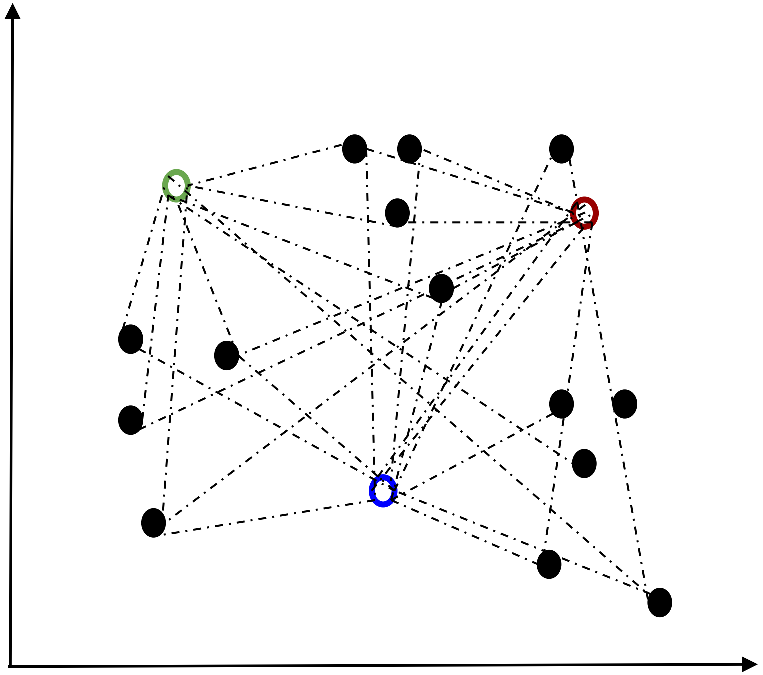
Centroid initialization methods

- K-means++
  - Starts by randomly selecting the first centroid and then chooses the subsequent centroids based on their maximum squared distance from existing centroids, spreading them apart to improve clustering.



Step 1: Choose K
K=3

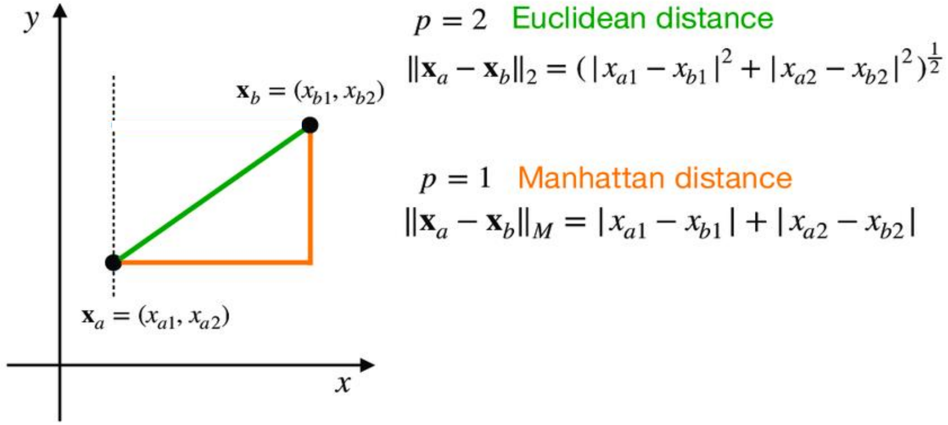
Step 2: Initialize centroids



Step 3A: Calculate the distance between every data point to each centroid.

Manhattan distance

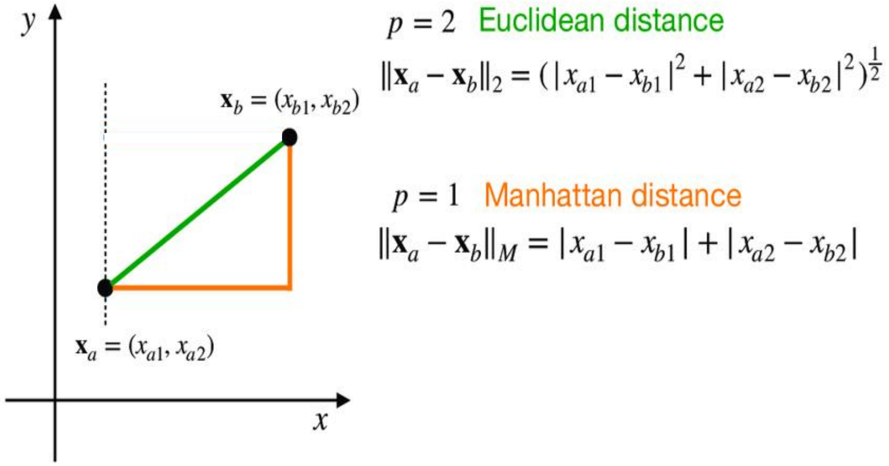
- Manhattan distance is preferred when the dimension of the data increases.
- It’s also ideal for binary and discrete features.
- NOTE: Distance tends to be larger than Euclidean distance since it’s not based on shortest path.

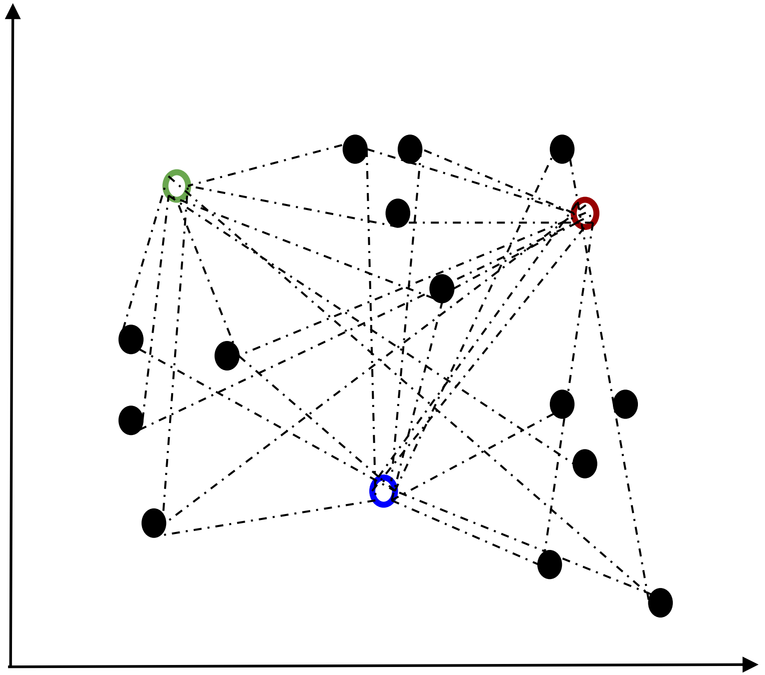


y
x
$\mathbf{x}_b = (x_{b1}, x_{b2})$
$\mathbf{x}_a = (x_{a1}, x_{a2})$
$p = 2$ Euclidean distance
$\|\mathbf{x}_a - \mathbf{x}_b\|_2 = (|x_{a1} - x_{b1}|^2 + |x_{a2} - x_{b2}|^2)^{\frac{1}{2}}$
$p = 1$ Manhattan distance
$\|\mathbf{x}_a - \mathbf{x}_b\|_M = |x_{a1} - x_{b1}| + |x_{a2} - x_{b2}|$

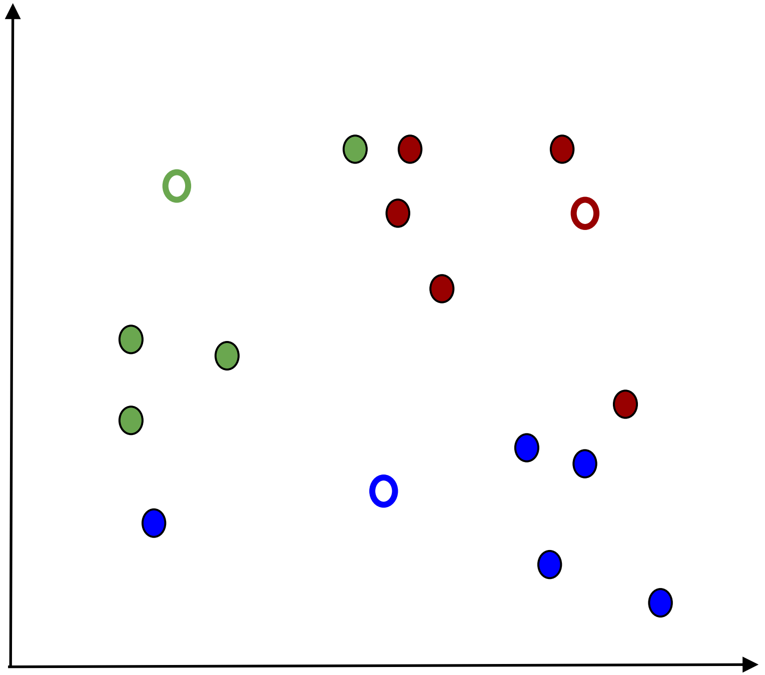
Euclidean distance

- Euclidean distance is preferred when you have low-dimensional data and the magnitude of the vectors is important to be measured.
- Easy to implement and the most commonly applied in machine learning.

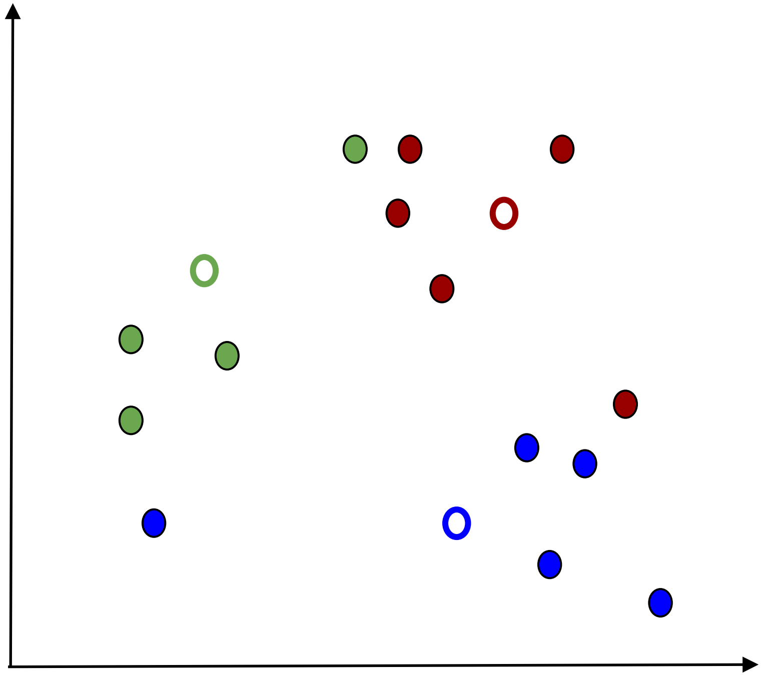

y
$\mathbf{x}_b = (x_{b1}, x_{b2})$
$\mathbf{x}_a = (x_{a1}, x_{a2})$
x
$p = 2$ Euclidean distance
$\|\mathbf{x}_a - \mathbf{x}_b\|_2 = (|x_{a1} - x_{b1}|^2 + |x_{a2} - x_{b2}|^2)^{\frac{1}{2}}$
$p = 1$ Manhattan distance
$\|\mathbf{x}_a - \mathbf{x}_b\|_M = |x_{a1} - x_{b1}| + |x_{a2} - x_{b2}|$



Step 3A: Calculate the distance between every data point to each centroid.



Step 3B: Assign the data points to the closest centroid.

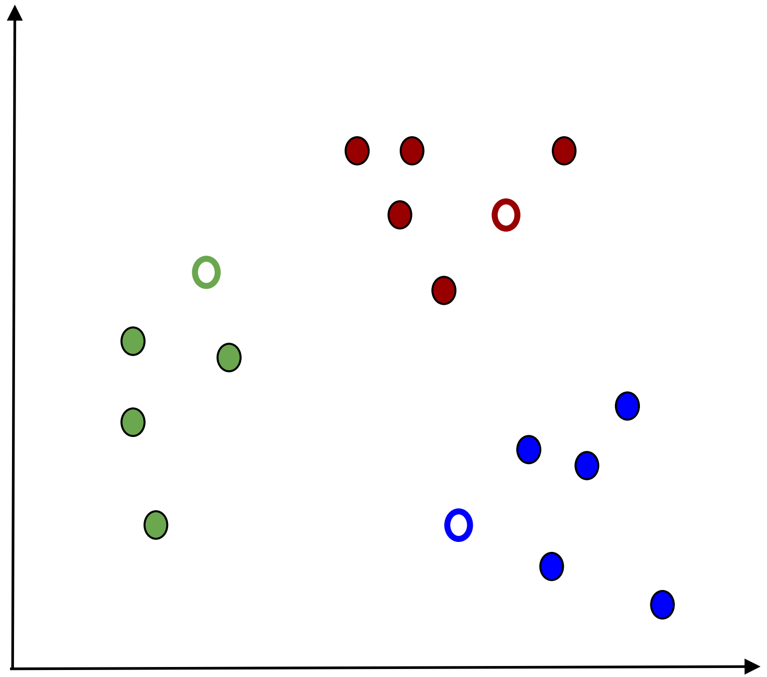


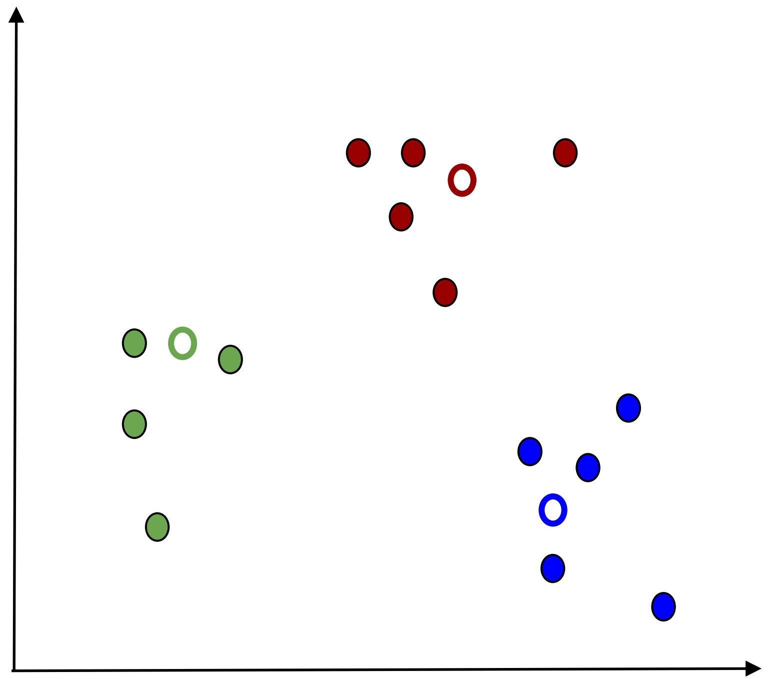
Step 4: Re-initialize centroids by calculating the average of all data points of that cluster.

- The average of the cluster is now the new position of the centroid.

$$C i = \frac{1}{|N i|} \sum xi$$

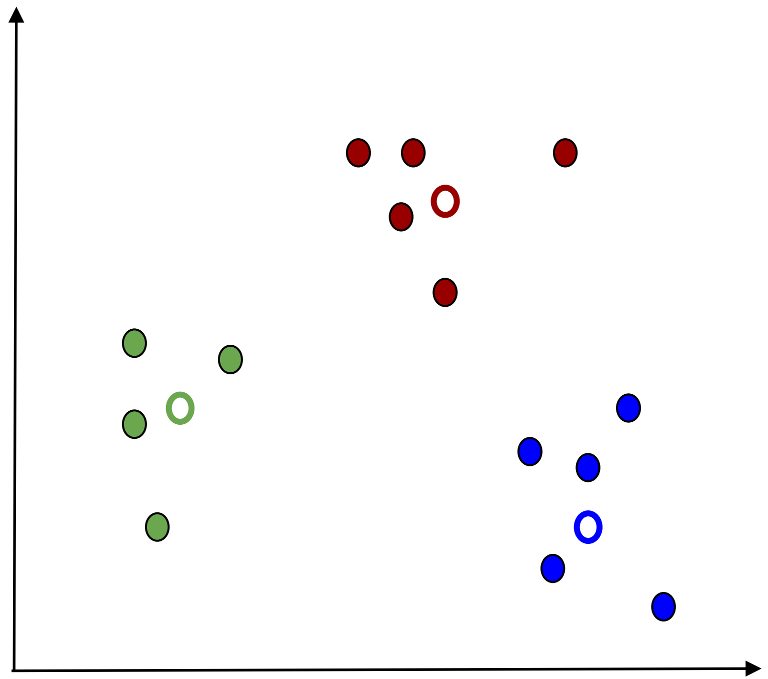
Repeat Steps 3 & 4





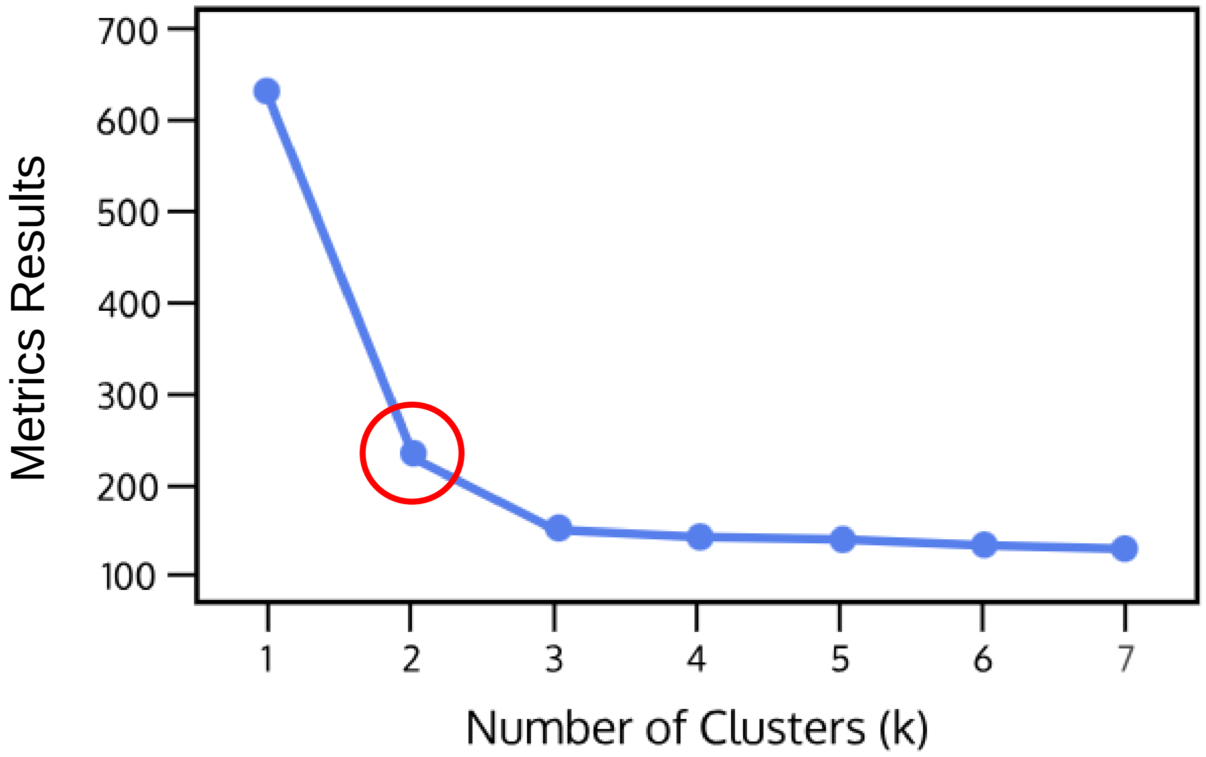
Stop until one of the following criteria is met:

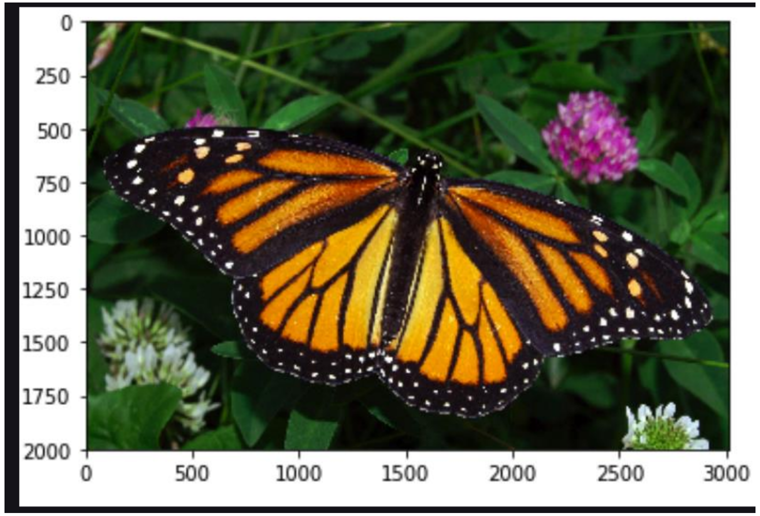
1. The data points assigned to specific cluster remain the same even after multiple iterations.
2. Centroids do not change.
3. All data points fall within a pre-established minimun distance from a centroid.
4. Fixed number of iterations have been reached.



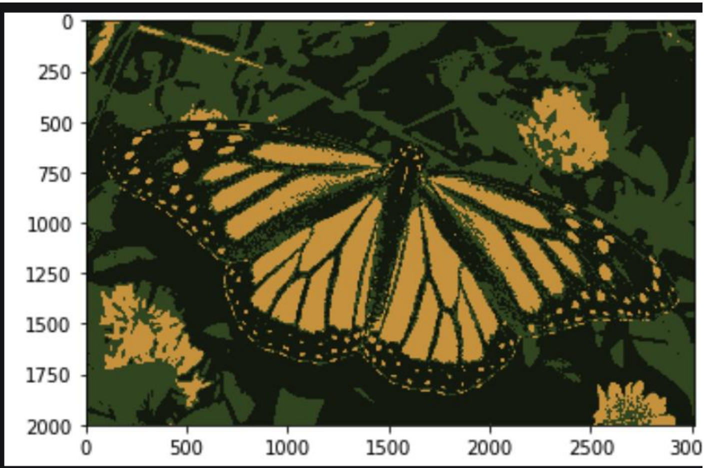
Once your predefined stopping criteria has been met, the algorithm has defined the 3 clusters.

Elbow Method

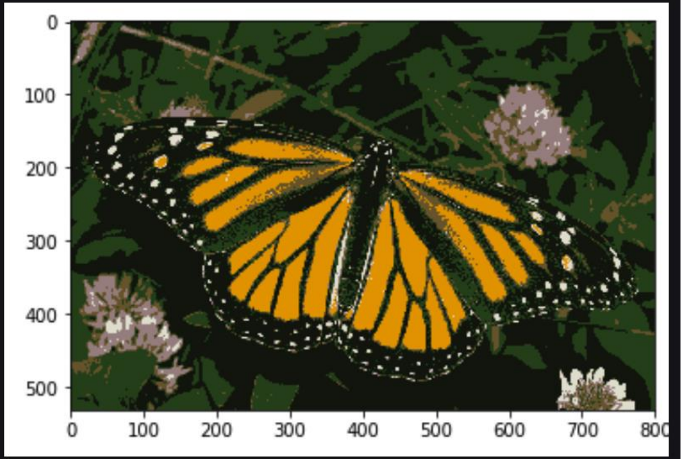




Original Image



Above K=3, below K=6



What performance metrics can we use?

1. **Inertia**: how tightly a dataset was clustered.

2. **Silhouette score**: the goodness of a clustering technique.

3. **Rand Index**: compare the similarity of results between two different clustering methods.

Inertia: how tightly points are to one another in cluster

μj= mean of the samples in the cluster (also the center of the cluster)

C= distinct cluster

N= number of samples

No set range for the score.
A good model is one with low inertia.

- Low inertia = tighter cluster -> more distinct cluster

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

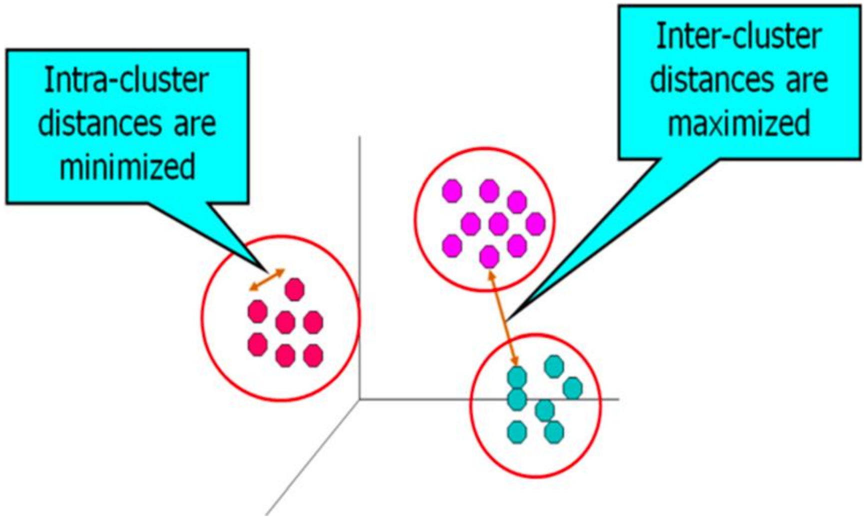
Silhouette Score: how tight the clusters are AND how spread out clusters are

Silhouette Score = (b-a)/max(a,b)

a= average **intra**-cluster distance
b= average **inter**-cluster distance

Score range

-1	0	1
Wrong clusters identified	Indifferent	Well distinguished


Intra-cluster distances are minimized
Inter-cluster distances are maximized

Rand Index: agreement score between clustering methods

a: The number of times a pair of elements belong to the same clusters across two clustering methods.

b: The number of times a pair of elements belong to different clusters across two clustering methods

n choose 2: The number of unordered pairs in a set of n elements.

$$R = \frac{a+b}{\binom{n}{2}}$$

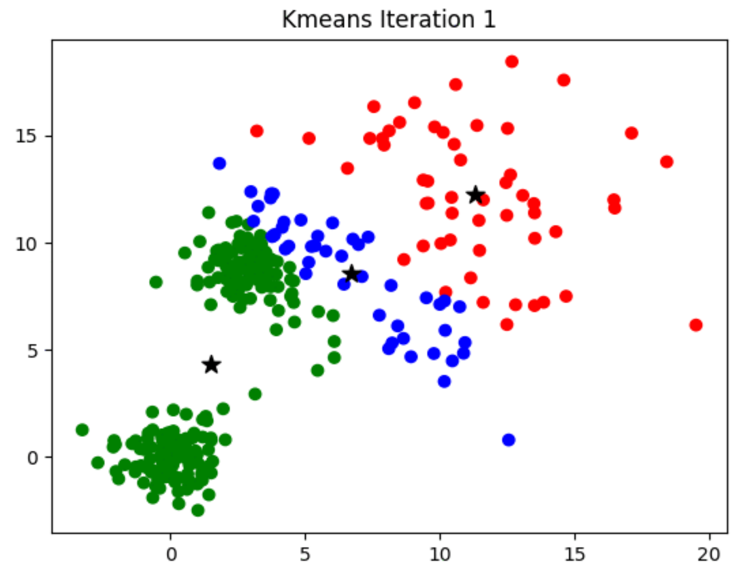
The Rand index score ranges from 0 and 1.

- **0:** Indicates that two clustering methods do not agree on the clustering of any pair of elements.

- **1:** Indicates that two clustering methods perfectly agree on the clustering of every pair of elements.

Summary

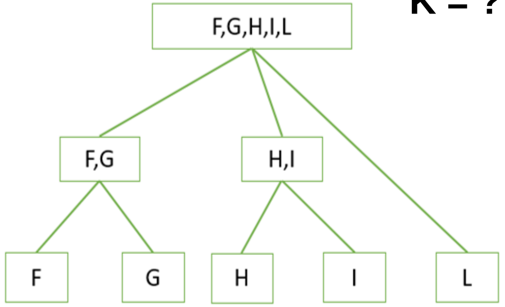
- K-means is an iterative centroid-based clustering algorithm which is easy to implement.
- There are 3 widely used initialization options and it’s important to choose the one that gets you to the closest optimal K.
- Elbow method is the most effective method to finding an optimal K.
- 3 main performance metrics used for K-means and other clustering algorithms:
  - Inertia
  - Silhouette Score
  - Rand Index (when another clustering can be used as a reference for performance evaluation)



Hierarchical Summary

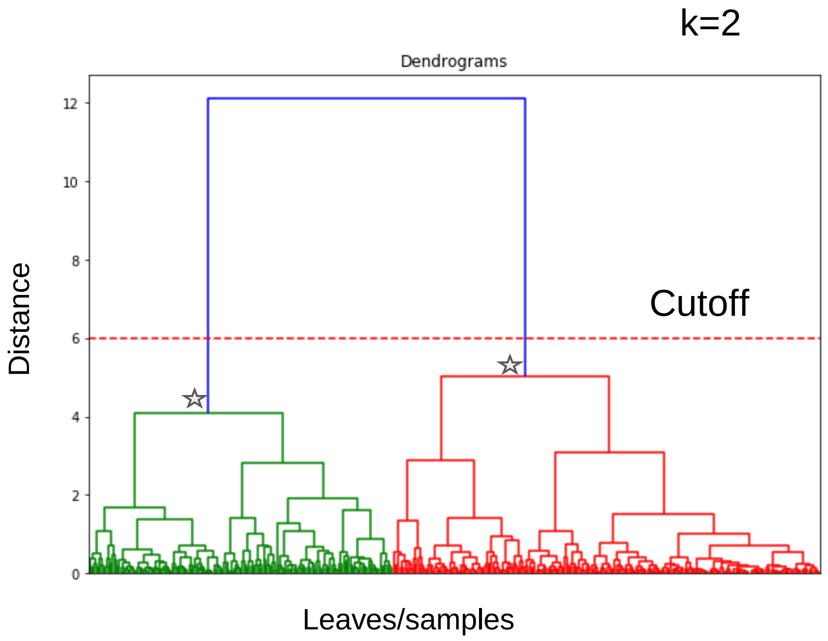
- **How is it different from K-means?**
  - No initial value of K required
- **Overview:**
  - Two types algorithms:
    - **Div**isive: start from root (k=1)
    - **Agg**lomerative: start with leaves (k=n)
  - Most algorithms are greedy
    - Best choice taken at each step (does not take into account context)
  - Dendrogram visualization
  - K is selected in post-analysis using cluster metrics, dendrogram analysis, or problem specific-context
- **Use-case**:
  - Exploratory:
    - Good visualization (particularly for smaller sample sizes)
    - No initial K required
  - Flexible:
    - Use various methods for cluster linkage comparisons
    - Use any distance metric for element wise comparisons

K = ?

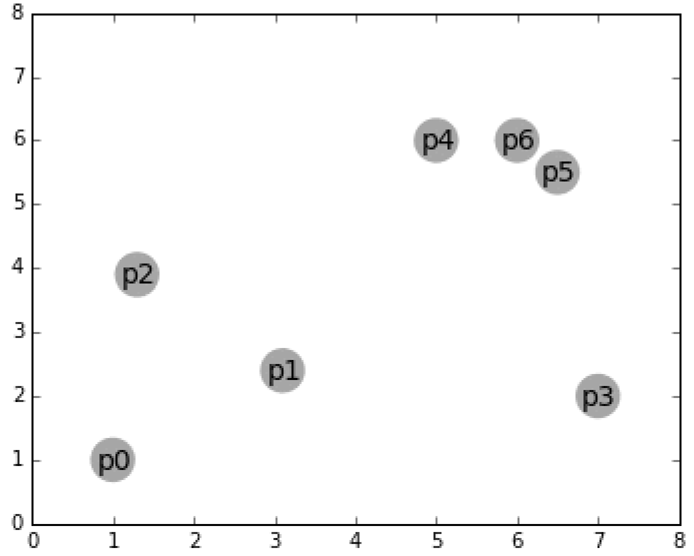


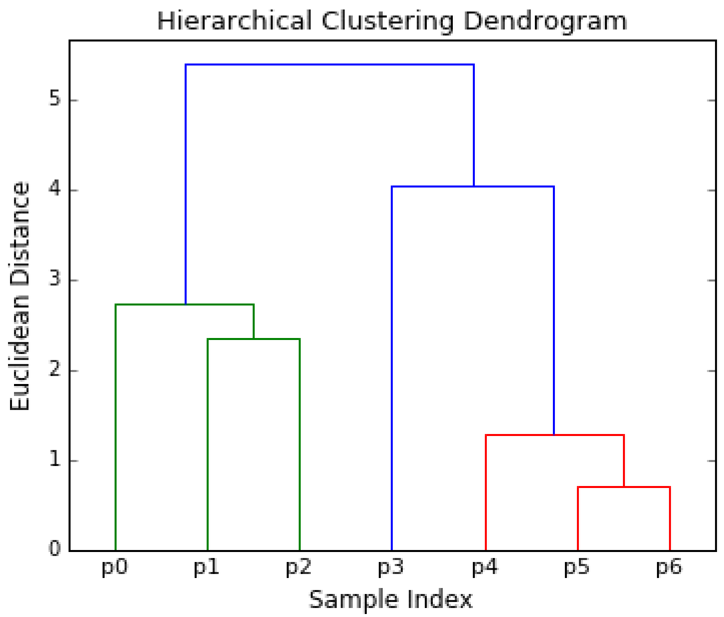
Hierarchical: Dendrogram

- **Dendrogram:** tree of clusters
  - **X-axis**: leaves/samples
  - **Y-axis**: distance between clusters (value depends on method)
  - **Nodes** of upside down U are **clusters**☆
  - **Cutoff** line is a distance threshold assigns final K clusters
    - K = number of vertical intersections with cutoff line
  - **Color** indicates cluster membership



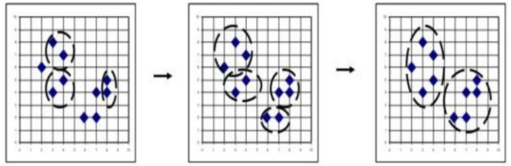
Hierarchical: Agglomerative

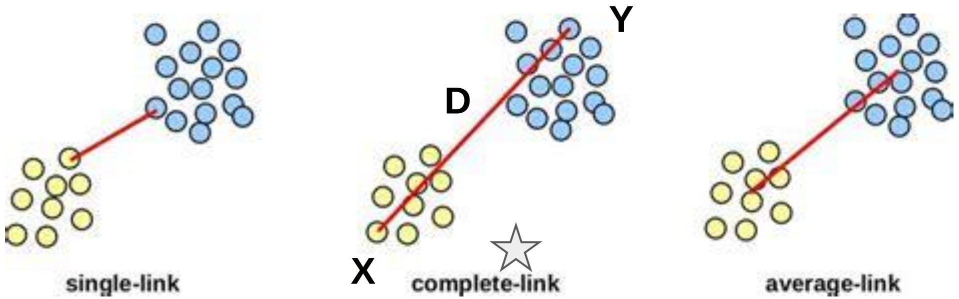




Hierarchical: complete-linkage (agglomerative) overview

1. Initialize each sample as a cluster (k = n)
2. A link (D) measures distance between each cluster
   a. For complete-linkage, the link equals the **maximum** of the pairwise distances within the two sets
      i. $D(X,Y) = \max_{x \in X, y \in Y} d(x,y)$
      ii. Represent as dissimilarity matrix
3. Merge clusters with minimum link value
4. Repeat 2,3 until k=1





Hierarchical: complete-linkage example

samples	A	B	C	D	E	F	G
A	0	0.5000	0.4286	1.0000	0.2500	0.6250	0.3750
B	0.5000	0	0.7143	0.8333	0.6667	0.2000	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.6667	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8000	0.8571
E	0.2500	0.6667	0.4286	1.0000	0	0.7778	0.3750
F	0.6250	0.2000	0.6667	0.8000	0.7778	0	0.7500
G	0.3750	0.7778	0.3333	0.8571	0.3750	0.7500	0

Dissimilarity matrix at k = n

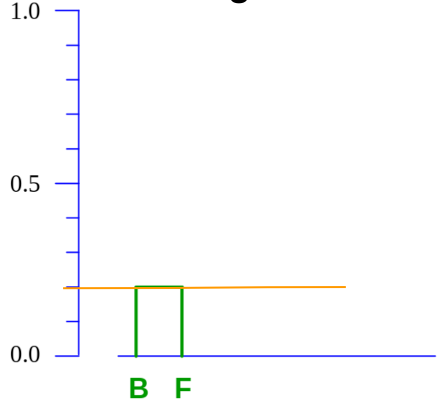
Minimum

	A	(B,F)	C	D	E	G
A	0	0.6250	0.4286	1.0000	0.2500	0.3750
(B,F)	0.6250	0	0.7143	0.8333	0.7778	0.7778
C	0.4286	0.7143	0	1.0000	0.4286	0.3333
D	1.0000	0.8333	1.0000	0	1.0000	0.8571
E	0.2500	0.7778	0.4286	1.0000	0	0.3750
G	0.3750	0.7778	0.3333	0.8571	0.3750	0

Dissimilarity matrix at k = n - 1

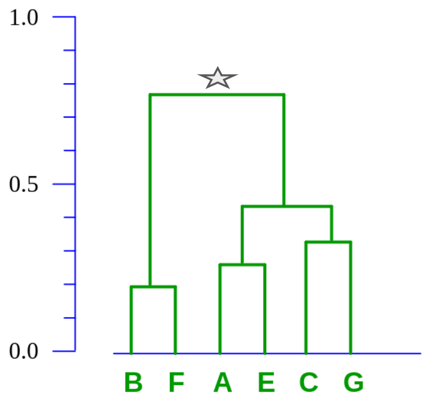
Dendrogram

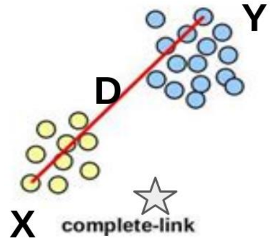
Complete-linkage distance



Hierarchical: complete-linkage example

samples	(A,E,C,G)	(B,F)	D
(A,E,C,G)	0	0.7778	1.0000
(B,F)	0.7778	0	0.8333
D	1.0000	0.8333	0

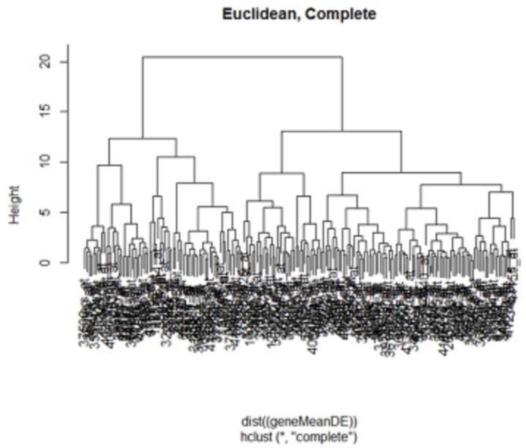


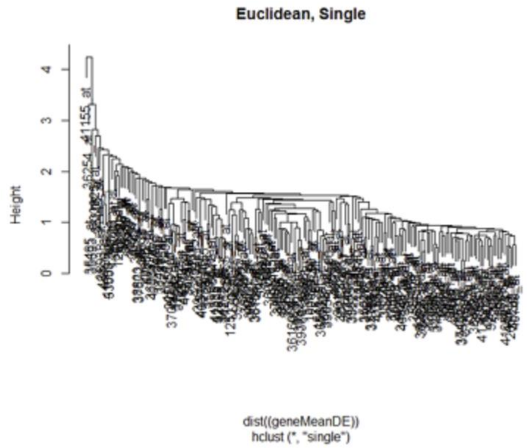


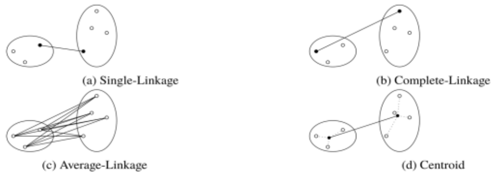
$$D(X,Y) = \max_{x \in X, y \in Y} d(x,y)$$

Hierarchical: linkage & distance methods affects outcome (biomedical examples)

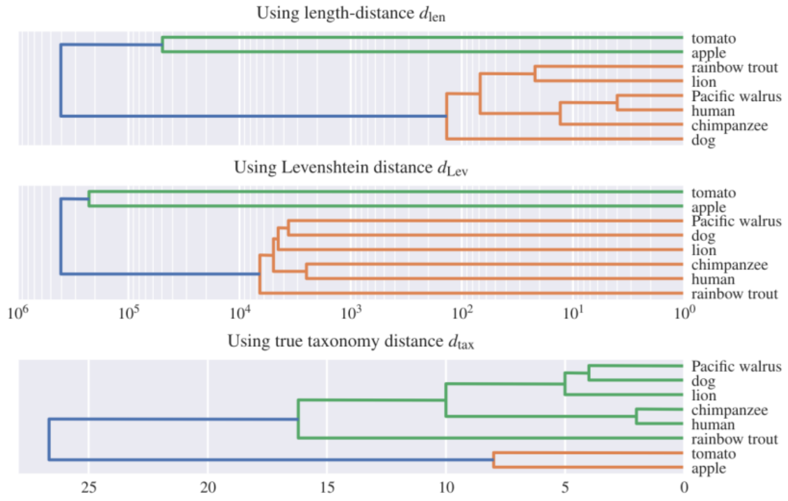
Different Linkage





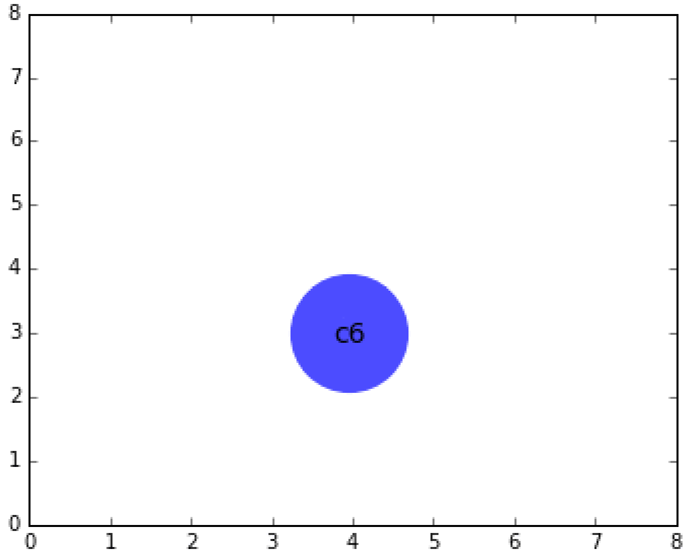


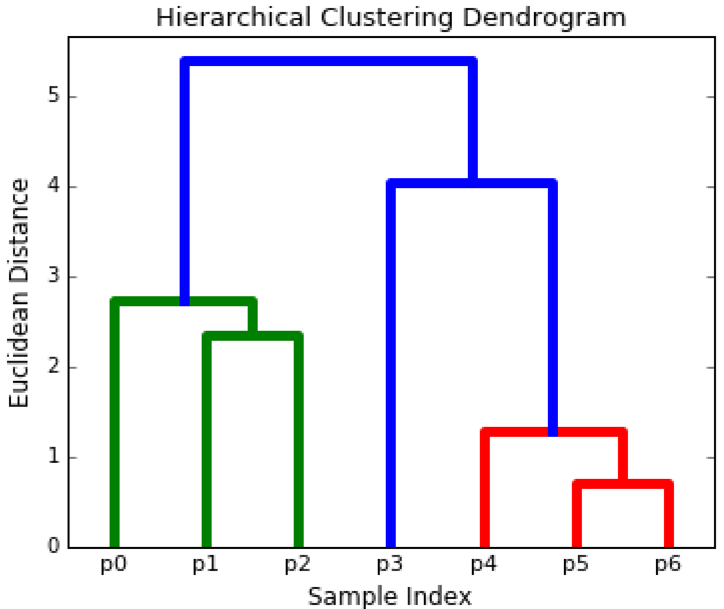
Different distance



Hierarchical: Divisive

- **Goal**: find the most dissimilar element of the cluster to make a new one
  - Inverse of agglomerative clustering
- Top-down approach: k = 1 -> k = n



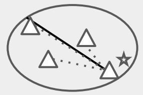


Hierarchical clustering: Divisive (DIANA)

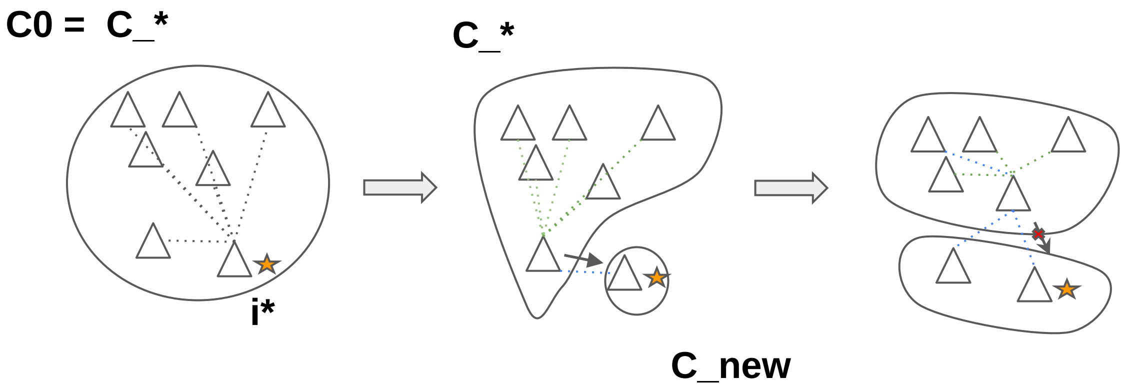
1. Let $C_0 = \{1 \dots n\}$ be the set of all n object indices and $\mathcal{C} = \{C_0\}$ the set of all formed clusters so far.
2. Iterate the following until $|\mathcal{C}| = n$:
    1. Find the current cluster with 2 or more objects that has the largest diameter: $C_* = \arg\max_{C \in \mathcal{C}} \max_{i_1, i_2 \in C} \delta(i_1, i_2)$
    2. Find the object in this cluster with the most dissimilarity to the rest of the cluster: $i^* = \arg\max_{i \in C_*} \frac{1}{|C_*| - 1} \sum_{j \in C_* \setminus \{i\}} \delta(i, j)$
    3. Pop i^* from its old cluster C_* and put it into a new *splinter group* $C_{\text{new}} = \{i^*\}$.
    4. As long as C_* isn't empty, keep migrating objects from C_* to add them to C_{new}. To choose which objects to migrate, don't just consider dissimilarity to C_*, but also adjust for dissimilarity to the splinter group: let $i^* = \arg\max_{i \in C} D(i)$ where we define $D(i) = \boxed{\frac{1}{|C_*| - 1} \sum_{j \in C_* \setminus \{i\}} \delta(i, j)} - \boxed{\frac{1}{|C_{\text{new}}|} \sum_{j \in C_{\text{new}}} \delta(i, j)}$, then either stop iterating when $D(i^*) < 0$, or migrate i^*.
    5. Add C_{new} to $\mathcal{C}$.

Plain english
Initialization: Start with one cluster
Repeat the following until the number of clusters equals the number of samples (divisive):

1. Find the cluster (C_*) with the largest, single, intra-cluster dissimilarity (diameter)
2. Choose the element (i*) with max average dissimilarity to other elements in C_*
3. Move i* from C_* to its own cluster (C_new)
4. For all elements in C_*, calculate **average dissimilarity** between C_* and C_new to decide whether to stay or leave



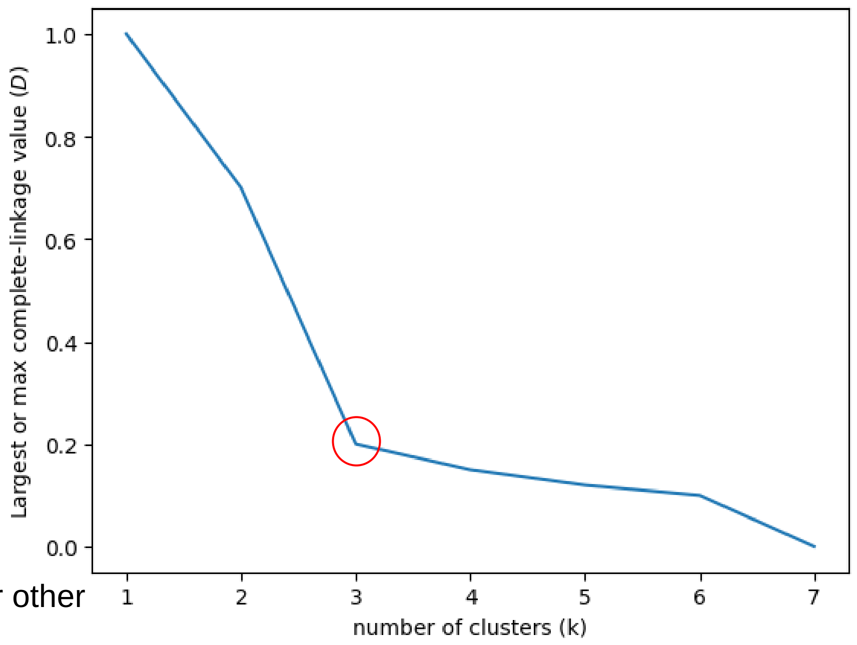
Hierarchical clustering: Divisive (DIANA)



For all elements in C_*:

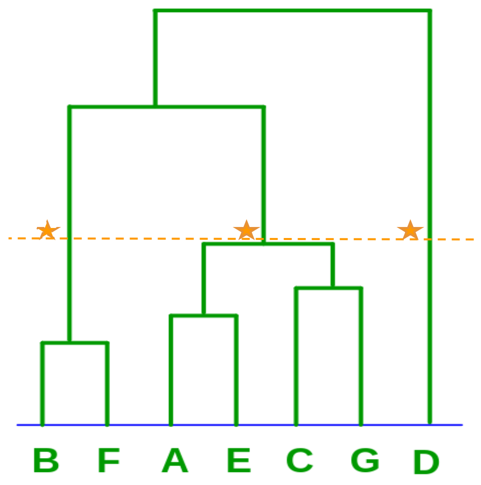
- Calculate average distance between **C_*** and **C_new** to decide whether to move
- Always move the element closest to the new set first

Hierarchical: choosing cutoff (K), Elbow method



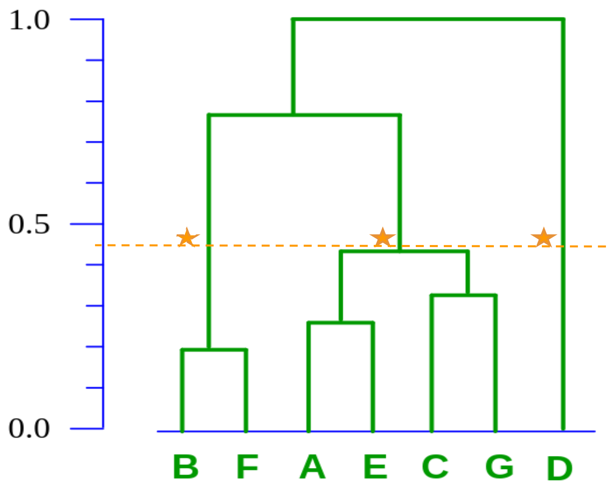
Inertia or other metrics

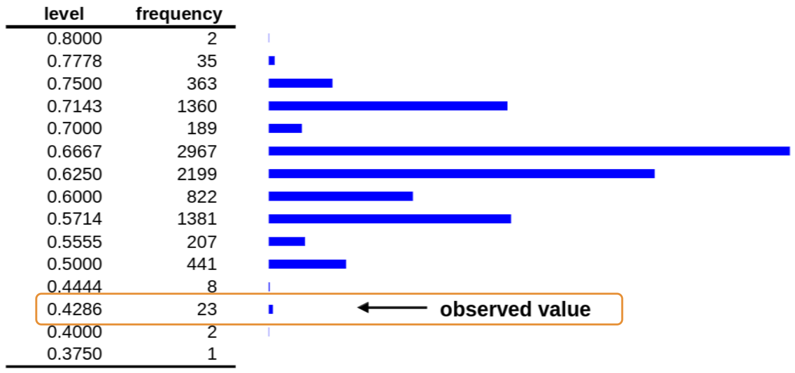
$$\sum_{i=1}^{N}(x_i - C_k)^2$$



Hierarchical: choosing cutoff (permutation hypothesis testing)

1. Choose a K of interest
2. Shuffle data
3. Generate distribution of distance cutoffs used to assign K
4. See if the distance cutoff for unshuffled data is significantly different from the randomized data





level	frequency
0.8000	2
0.7778	35
0.7500	363
0.7143	1360
0.7000	189
0.6667	2967
0.6250	2199
0.6000	822
0.5714	1381
0.5555	207
0.5000	441
0.4444	8
0.4286	23
0.4000	2
0.3750	1

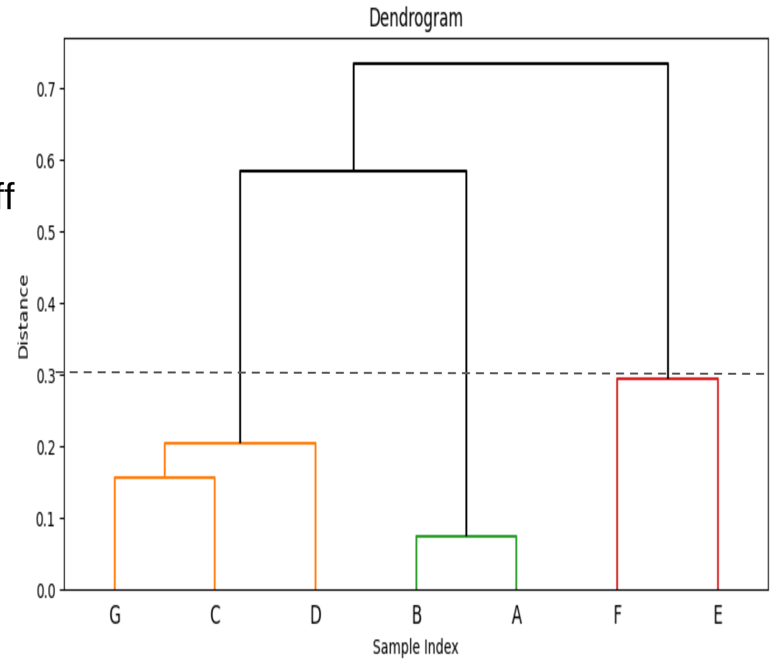
Hierarchical clustering: permutation testing for K=3 ex

	j1	j2
A	0	0.1
B	0	0.2
C	0	0.3
D	0	0.4
E	1	0.5
F	1	0.6
G	1	0.7

Shuffle

	j1	j2
A	1	0.5
B	1	0.6
C	0	0.1
D	0	0.4
E	0	0.2
F	1	0.3
G	0	0.7

Calculate cutoff at K



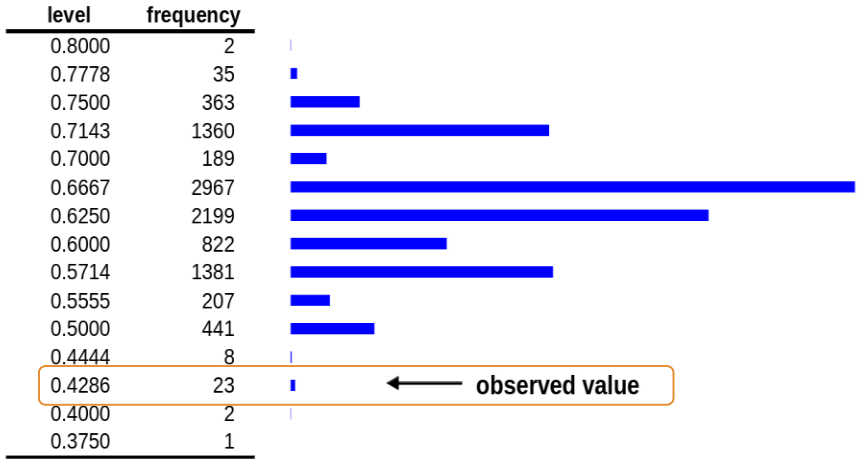
K = 3 when distance cutoff = 0.4

What distance cutoff will yield K clusters?

K = 3 when distance cutoff = 0.3

Hierarchical clustering: permutation testing for K=3 ex

- Result: give insight on how probable this distance cutoff is to be observed in the actual data versus randomized data
- Assumptions when randomizing:
  - When shuffling individual columns, assume within column values are fixed
  - When generating with distributions: assume a distribution that represents the column and generate values.



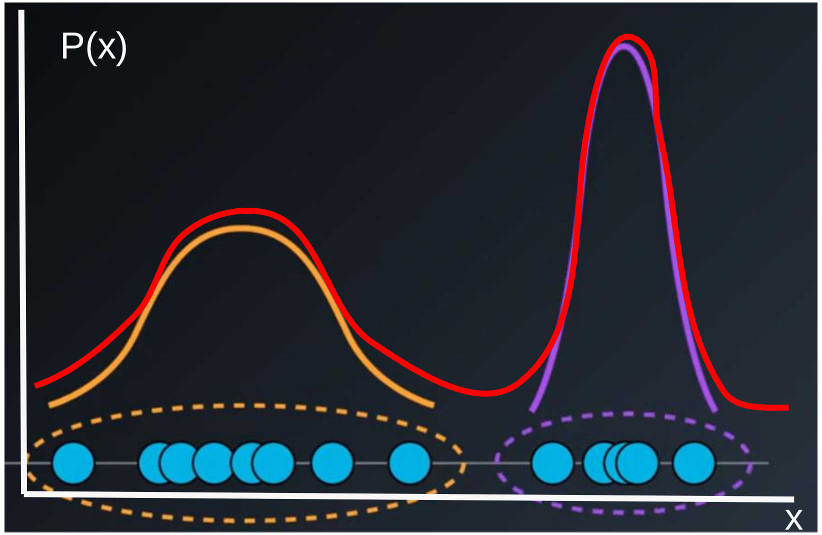
level	frequency
0.8000	2
0.7778	35
0.7500	363
0.7143	1360
0.7000	189
0.6667	2967
0.6250	2199
0.6000	822
0.5714	1381
0.5555	207
0.5000	441
0.4444	8
0.4286	23
0.4000	2
0.3750	1

Hierarchical: summary

- Two main categories: agglomerative & divisive
- Dendrograms visualize clustering process
- Typically, K or distance cutoff is chosen after the clustering is done:
  - Elbow method
  - Permutation testing
- Pros:
  - Visuals, flexibility
- Cons:
  - Computational complexity can be very high
  - Dendrogram is hard to interpret for large datasets
  - Flexibility

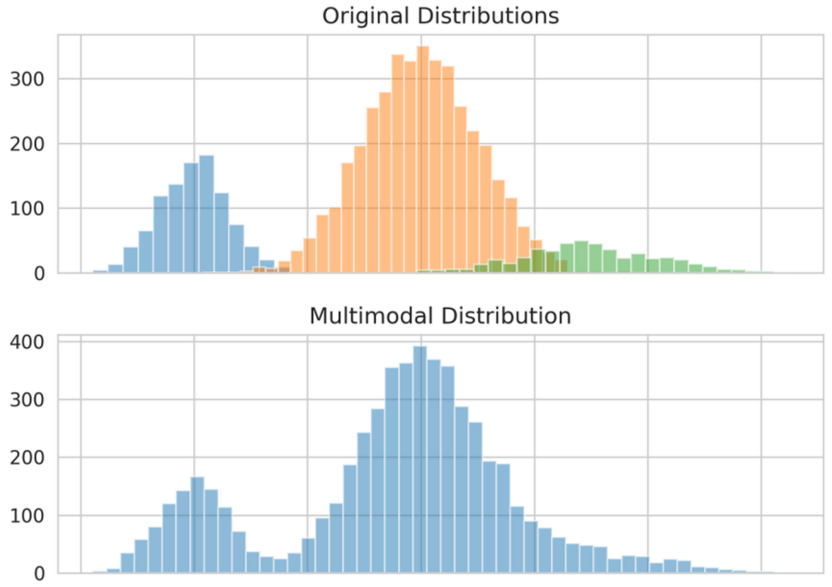
Mixture Models

- **Probabilistic model** for clustered data with real valued components
- Data generated from different random processes that are mixed to give us what we observe



Some assumptions made for mixture models

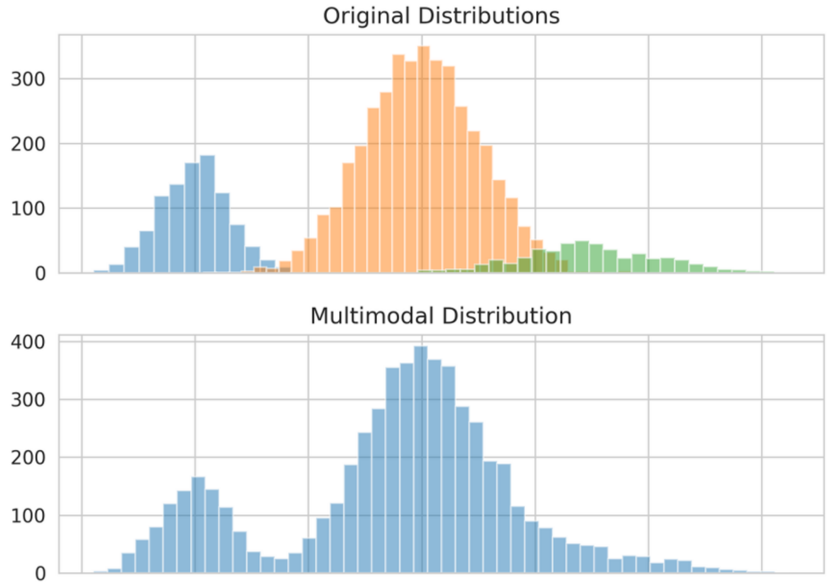
- How many models?
  - k
- What type of models?
  - Gaussian! Yay!



https://medium.com/@adev94/how-to-split-multimodal-distribution-with-gaussian-mixture-models-in-python-c87957553e4

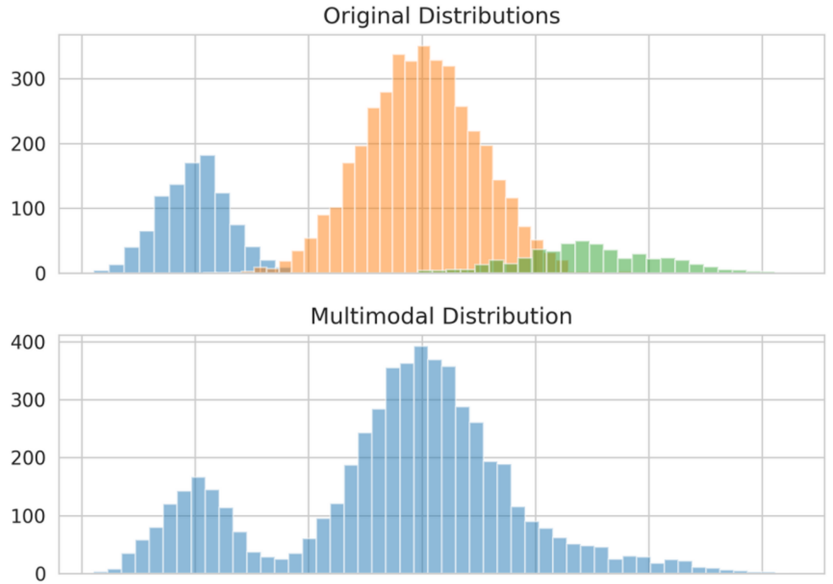
A superposition of gaussians?

- $P(X) = P_1(X) + P_2(X) + P_3(X)$



A superposition of gaussians? - No.

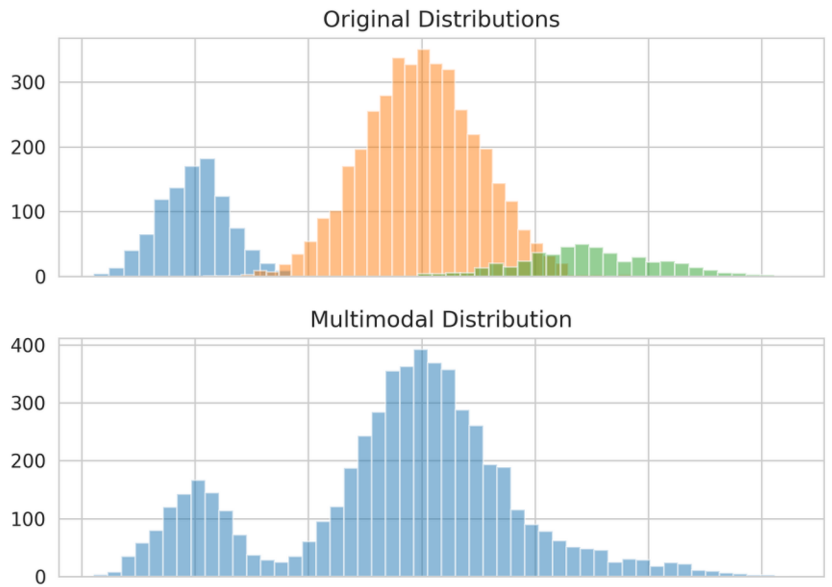
- $P(X) \neq P_1(X) + P_2(X) + P_3(X)$
  - AUC of probability density function must be equal to 1
  - $P_1(X) + P_2(X) + P_3(X) = 3$
- We can use **mixing coefficients** to satisfy the property of probability functions
  - $P(X) = \pi_1 P_1(X) + \pi_2 P_2(X) + \pi_3 P_3(X)$
    - $\pi_1 = 0.3$
    - $\pi_2 = 0.6$
    - $\pi_3 = 0.1$



Modeling with Gaussian Mixture Models (GMM): Equation and Parameters

$$P(x) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x|\mu_k, \Sigma_k)$$

- Where *P(x)* = overall probability distribution of observing data point x
- π_k = mixing coefficient for the k-th distribution (such that they sum to 1)
- μ_k = mean vector of k-th distribution
- Σ_k = covariance matrix of k-th component

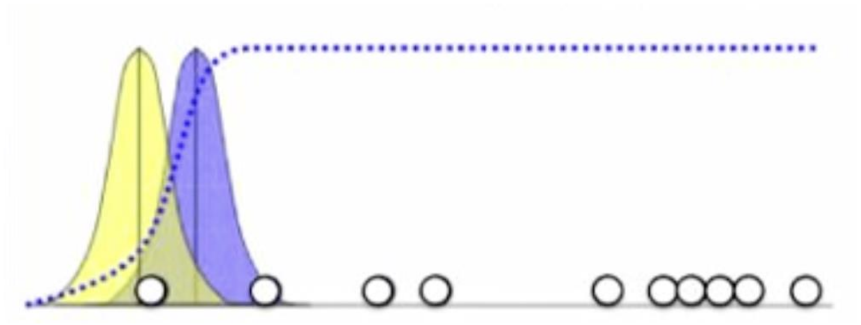


Expectation Maximization Algorithm

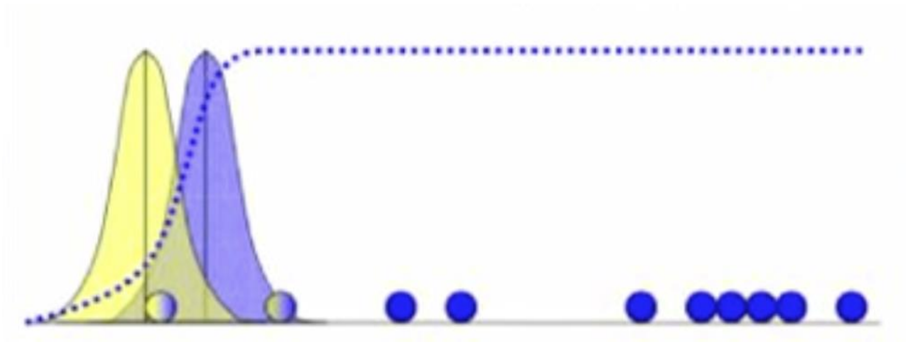
Procedure

1. Initialization of the parameters
2. Compute the cluster membership probabilities given the current parameters (**E**xpectation)
3. Update parameters to maximize the expected log likelihood (**M**aximization)
4. Repeat E and M steps until convergence

Step 1: Initialize parameters with random seed



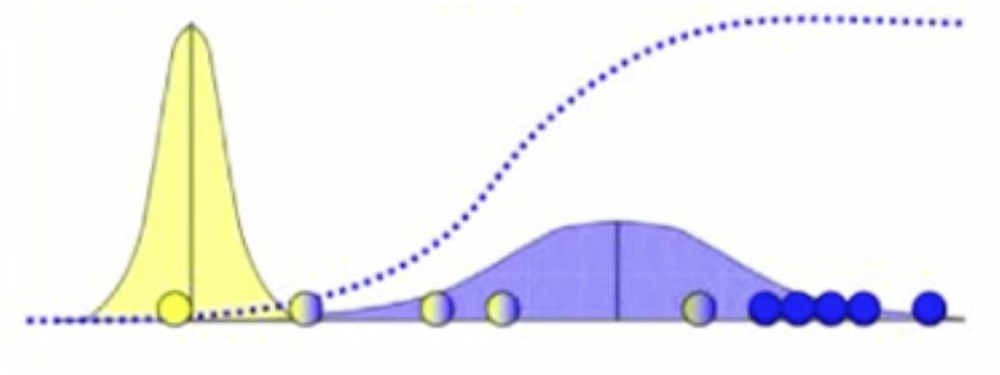
Step 2: **Expectation** step — compute soft class memberships



$$\tau_{ij} = P(z_i = j | x_{ij}, \pi, (\mu\ , \Sigma\ )).$$

- τ_{ij} = probability of point x_i belonging to cluster j given our guesses for parameters mean, covariance and mixing coefficient

Step 3: **Maximization** step — Re-estimate and update parameters based on our membership classification

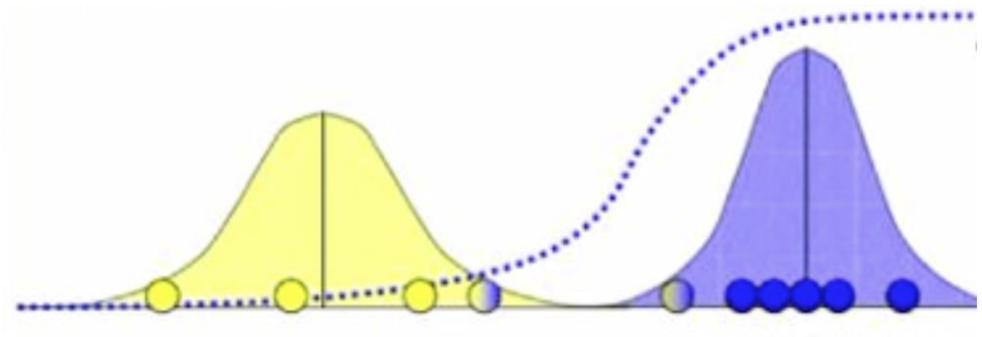


$$\pi_j = \frac{1}{n}\sum_{i=1}^{n} \tau_{ij},$$

$$\Sigma_j = \frac{\sum_{i=1}^{n} \tau_{ij}(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{n} \tau_{ij}}.$$

$$\mu_j = \frac{\sum_{i=1}^{n} \tau_{ij} x_i}{\sum_{i=1}^{n} \tau_{ij}},$$

Step 4: Repeat steps 2 and 3 until convergence

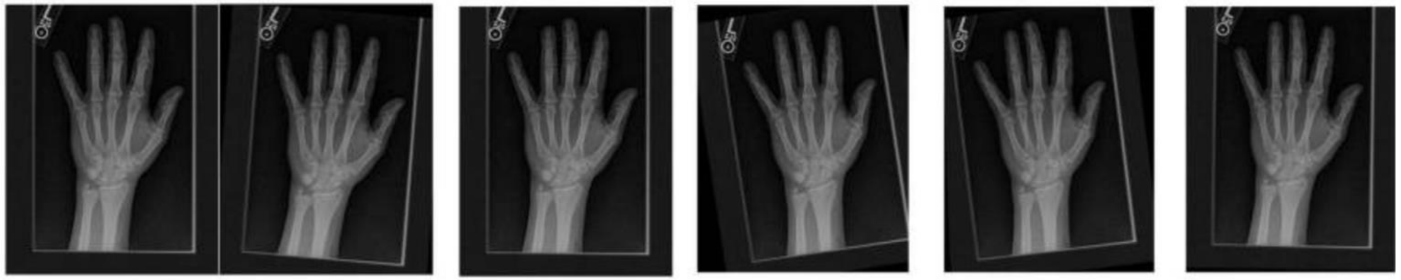


- Stopping criteria
  - Change in estimated parameters between iterations
  - Change in Log-Likelihood of the observed data with current parameters
  - Maximum number of iterations
- Note: EM is not guaranteed to find the global maximum
  - Make multiple initializations

If we have more information about the data, we can use other distributions!

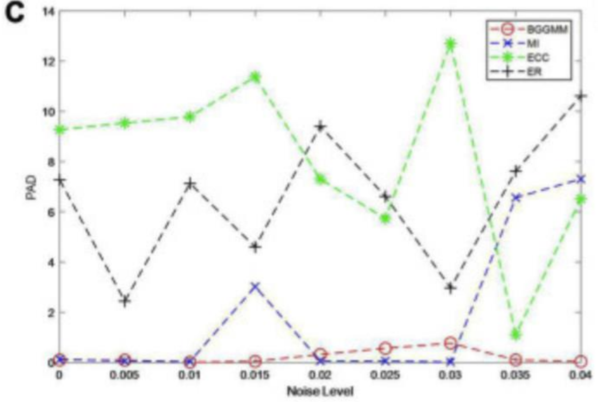
- Poisson Distribution - suitable for modeling count data
- Negative Binomial Distribution - useful for modeling overdispersed count data
- Log Normal Distribution
- Gamma Distribution

Example: Bounded generalized GMM for medical image registration



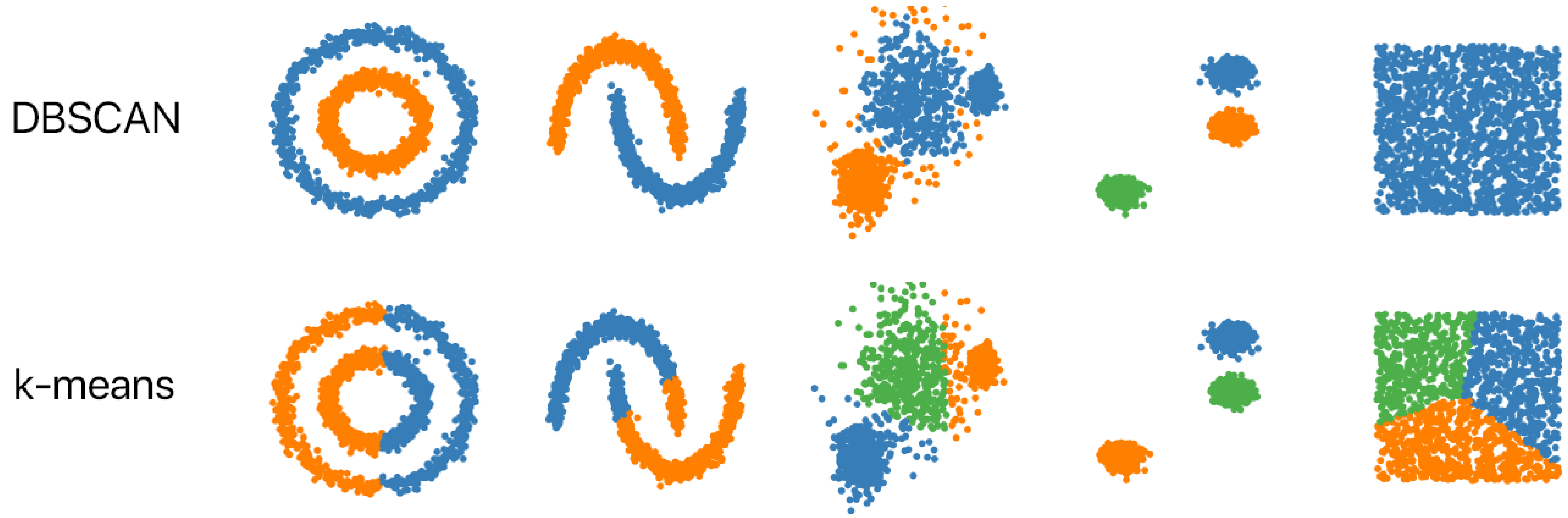
Initialization BGGMM ECC ER MI

C



Wang, J., et al. (2022). Medical Image Registration Algorithm Based on Bounded Generalized Gaussian Mixture Model. *Frontiers in Neuroscience*.

DBSCAN (Density-based spatial clustering of applications with noise)

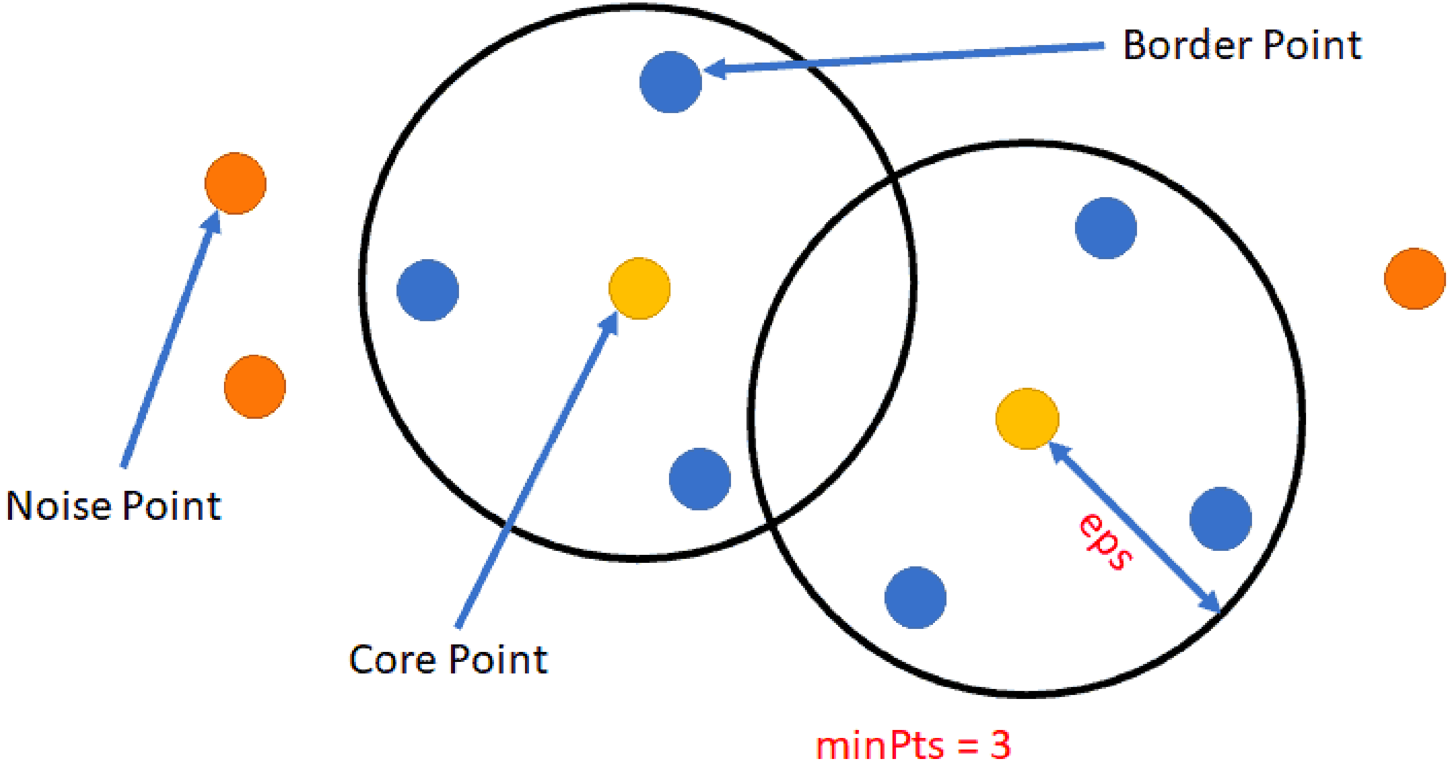


https://github.com/NSHipster/DBSCAN

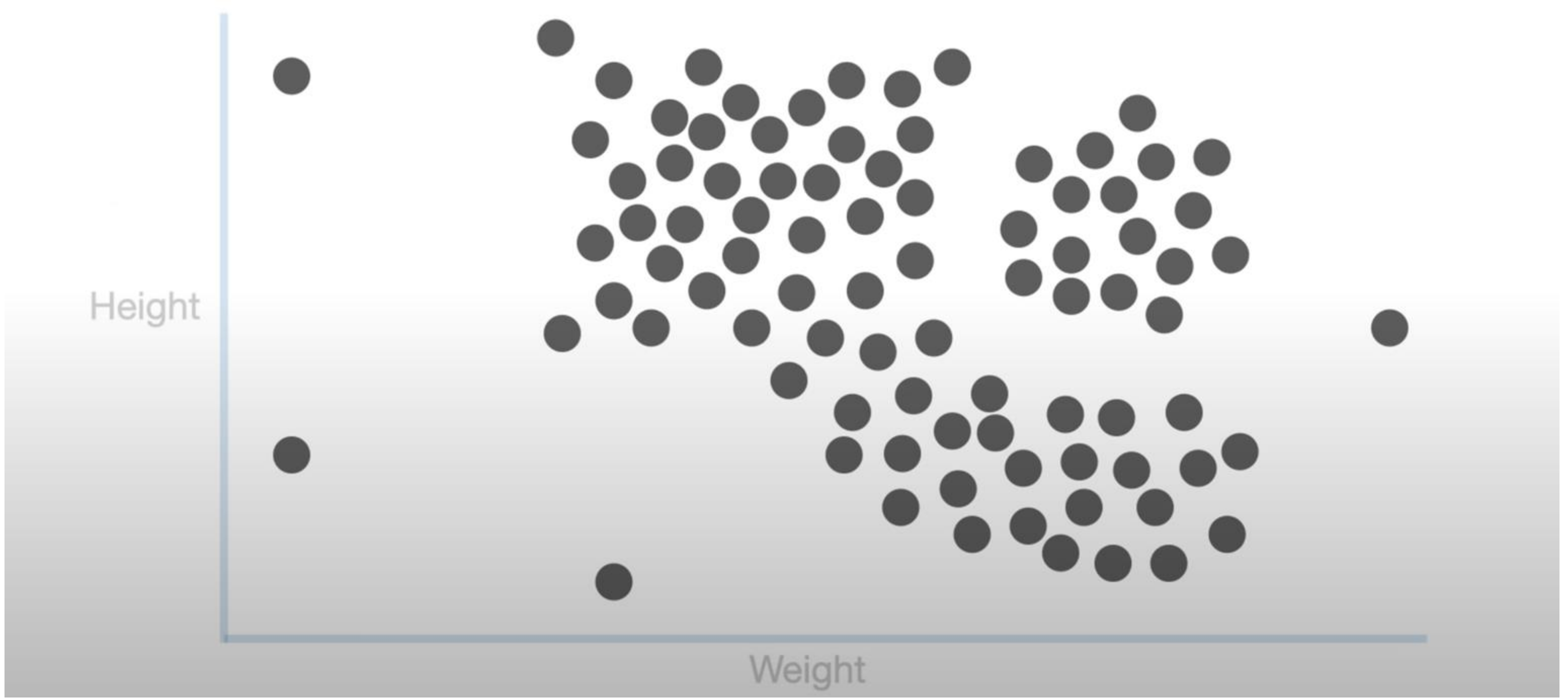
DBSCAN

- No assumptions about distributions
- Does not necessarily assign all objects to a cluster
- Clusters based on Euclidean density (usually)
  - Density of clusters must be significant
  - Does not scale well to high dimensional data
- Two hyperparameters:
  - Size of radius (ε/eps)
  - Minimum number of neighbors to be a core point (MinPts)

Terms

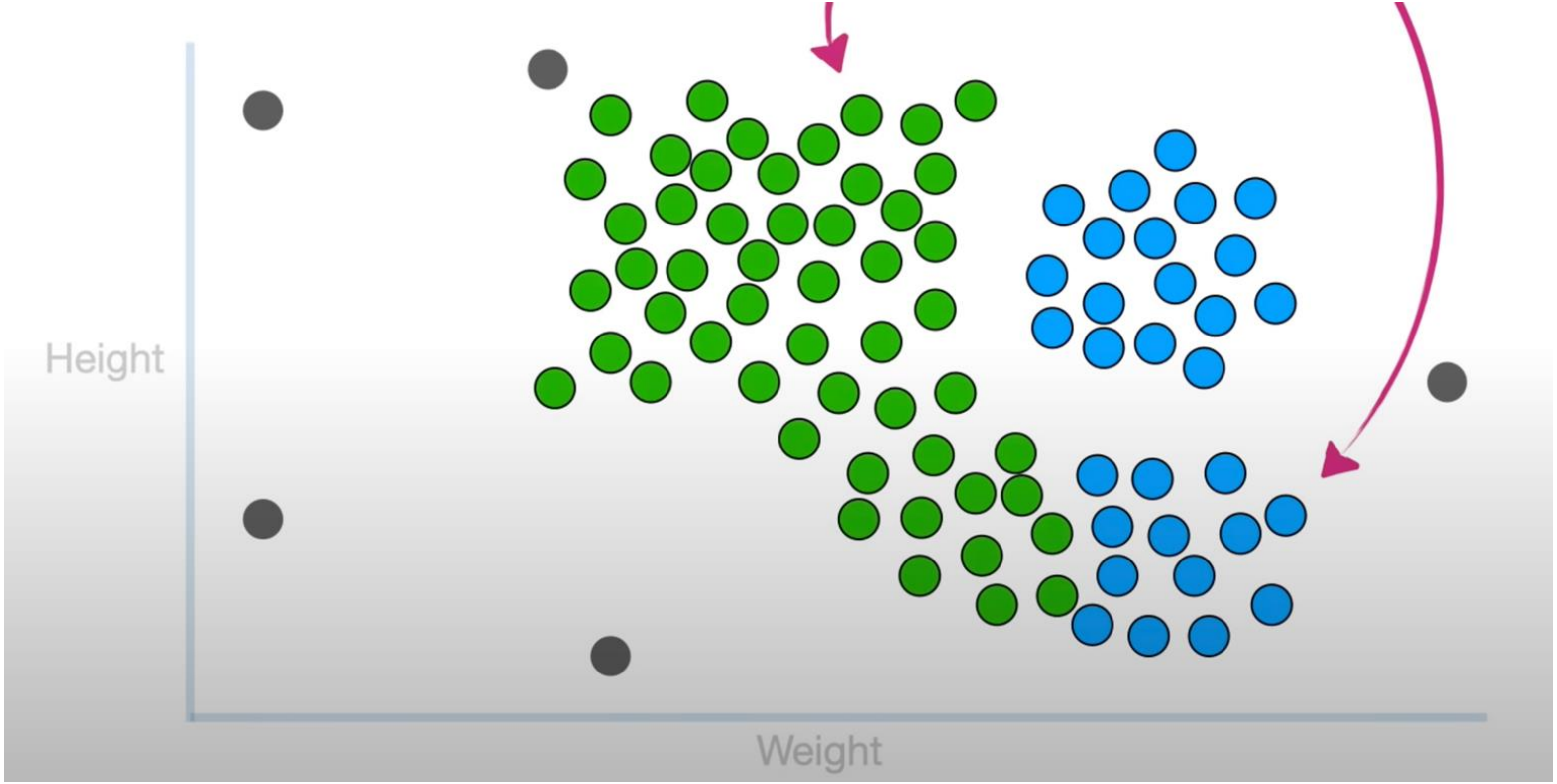


Example: Grouping people by height and weight

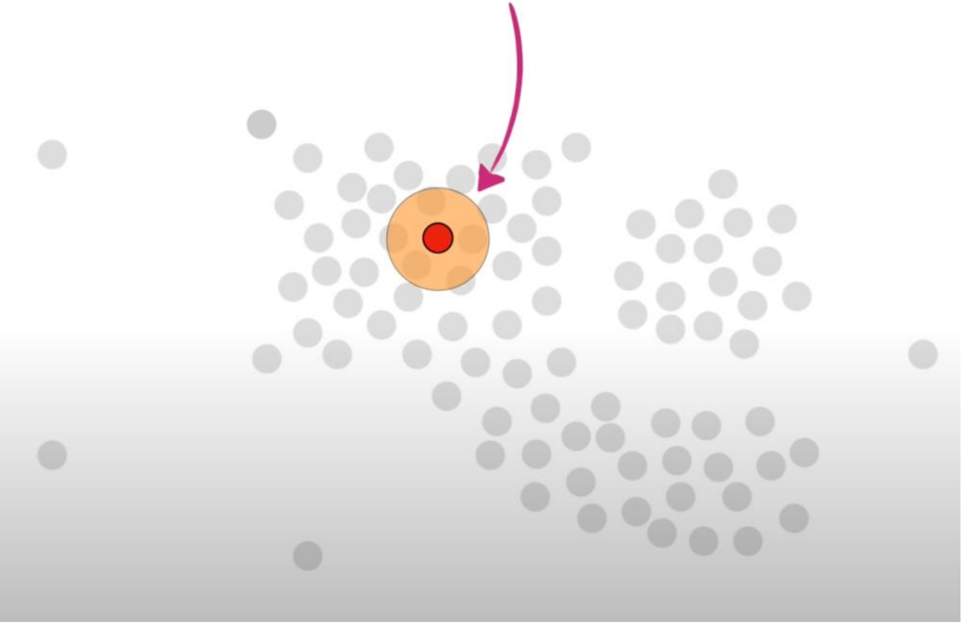


Example taken from StatQuest

K-means clustering

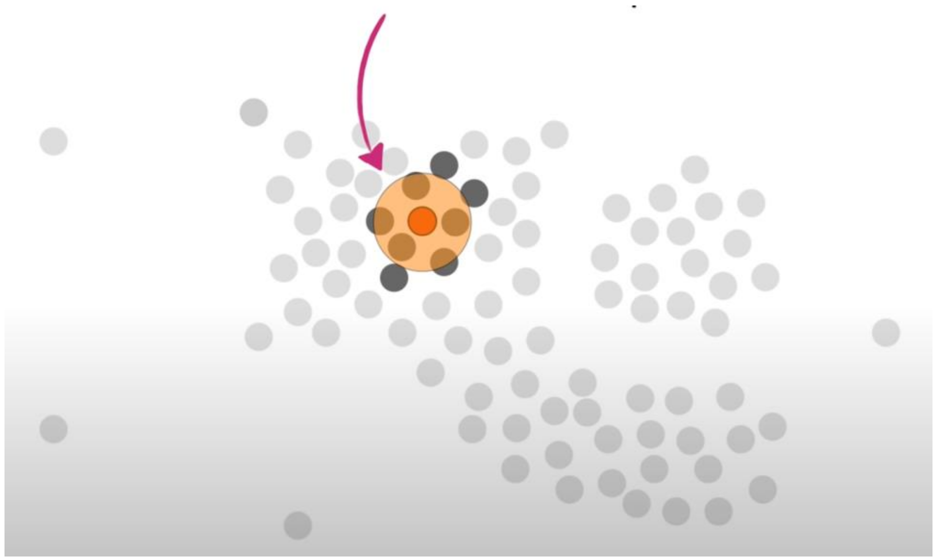


Start with a point and radius (ε)



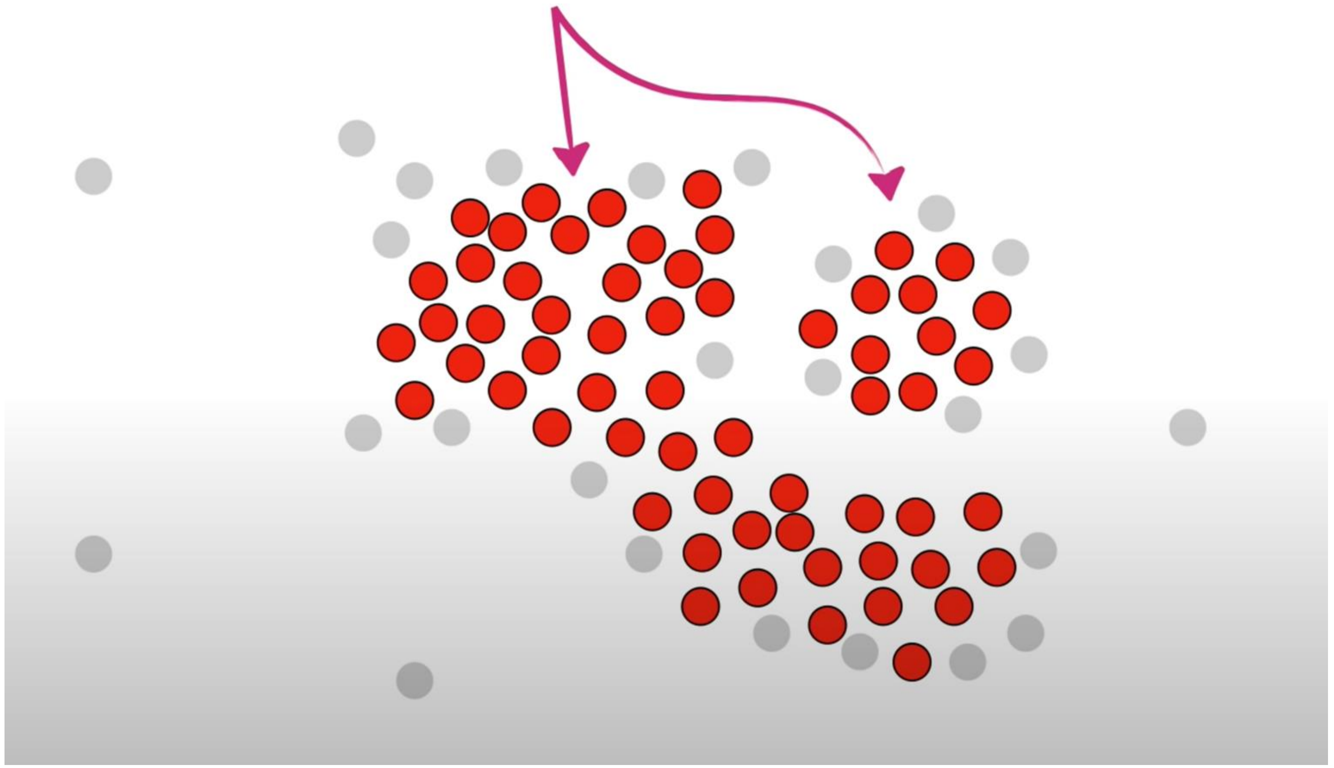
- How to pick ε?
  - Visual inspection/domain knowledge
  - k-NN distance/elbow method
  - Silhouette score
  - Iterative experimentation

Find the number of points that fall in the radius

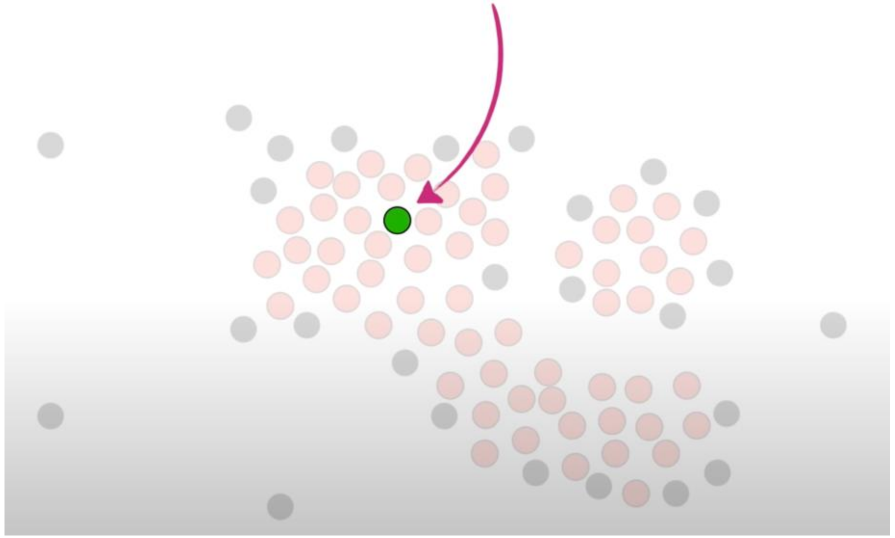


- p is a *core point* if at least MinPts fall partially within this radius
- p is a *directly reachable* or *border point* if it falls within the radius of a core point but is not a core point itself
- p is an *outlier* or *noise* if it is not reachable from any core points

Find all core points

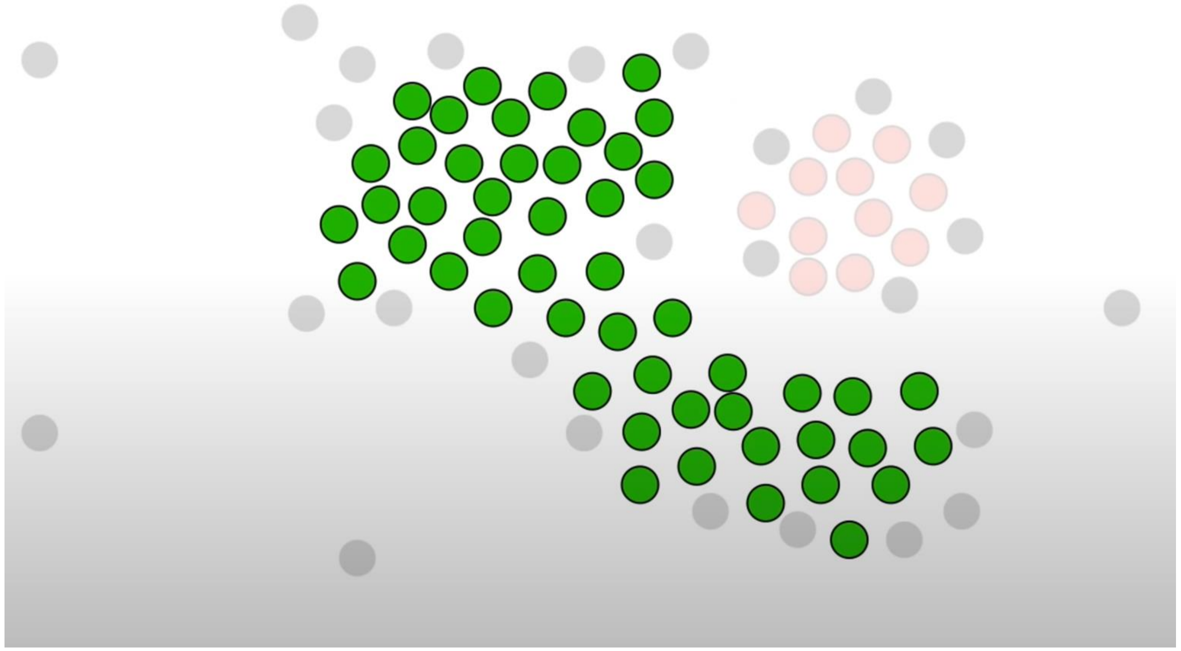


Randomly assign a core point to the first cluster

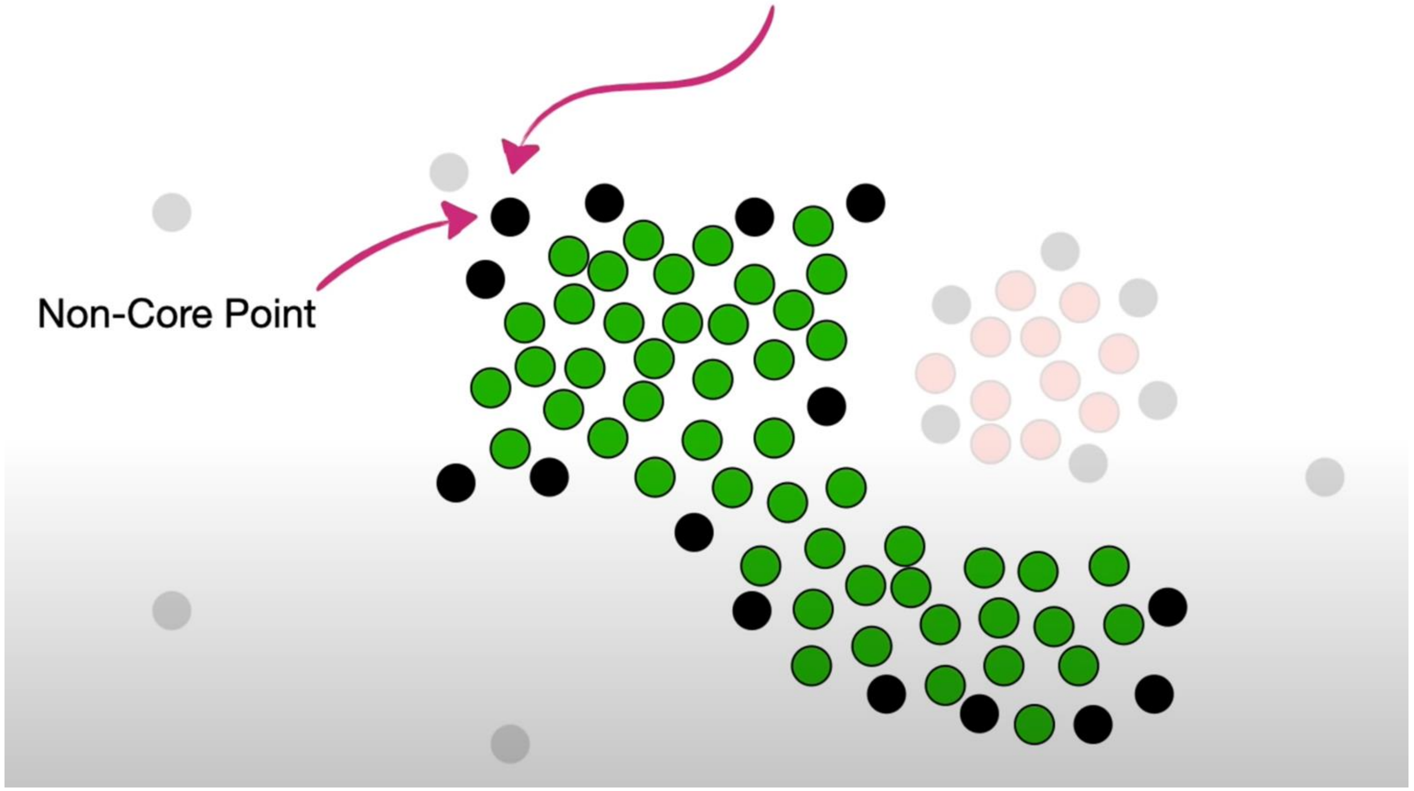


- Starting point can influence clusters
- Recommended to run multiple times

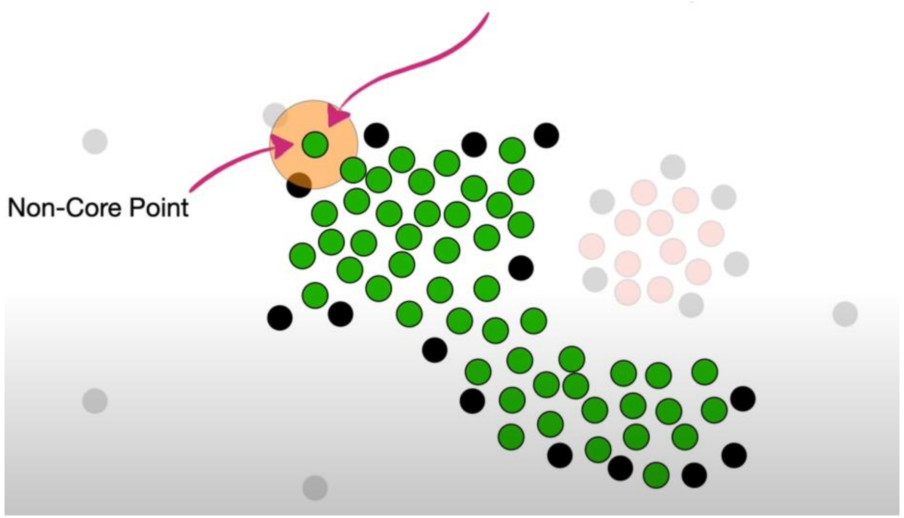
Expand cluster until no more core points can be added



Expand cluster to include all directly reachable points

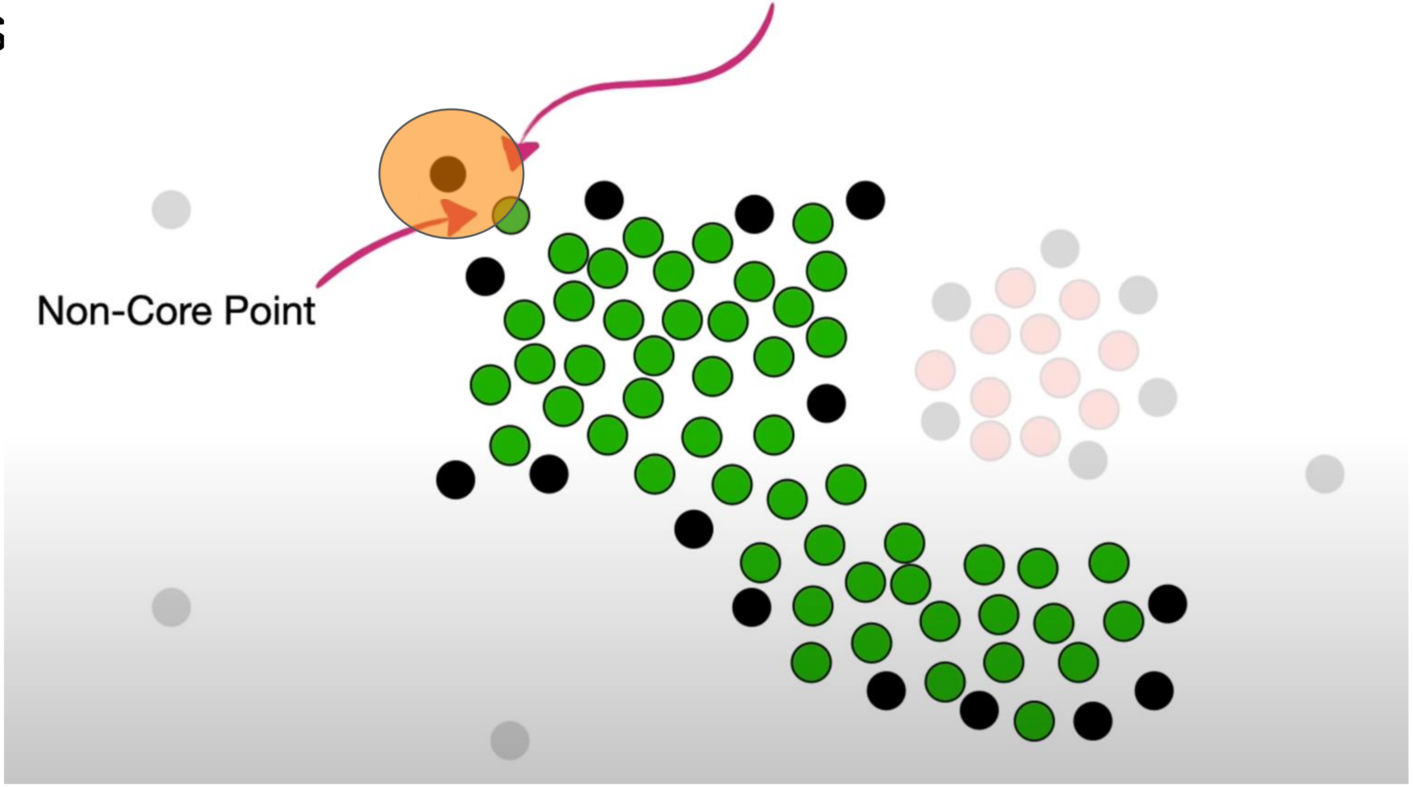


Expand cluster to include all directly reachable points

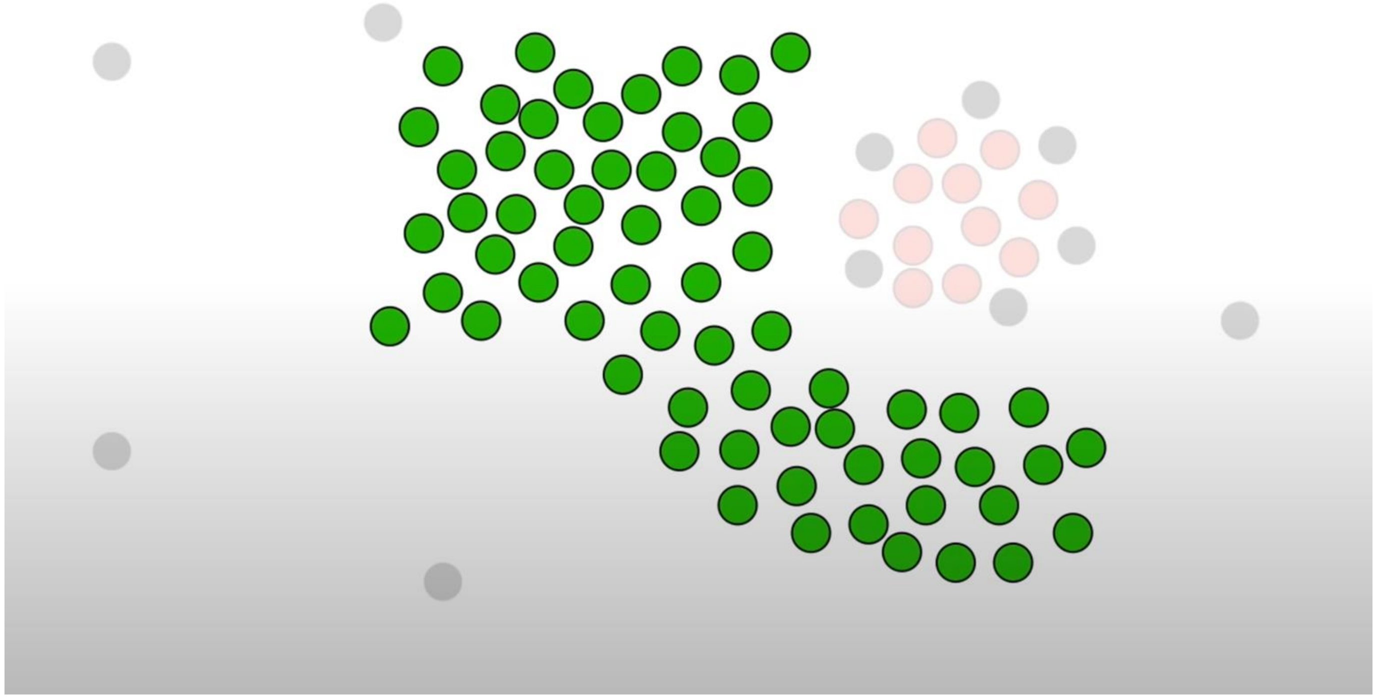


- Directly reachable points are added to the cluster as border points but **not** used to expand it

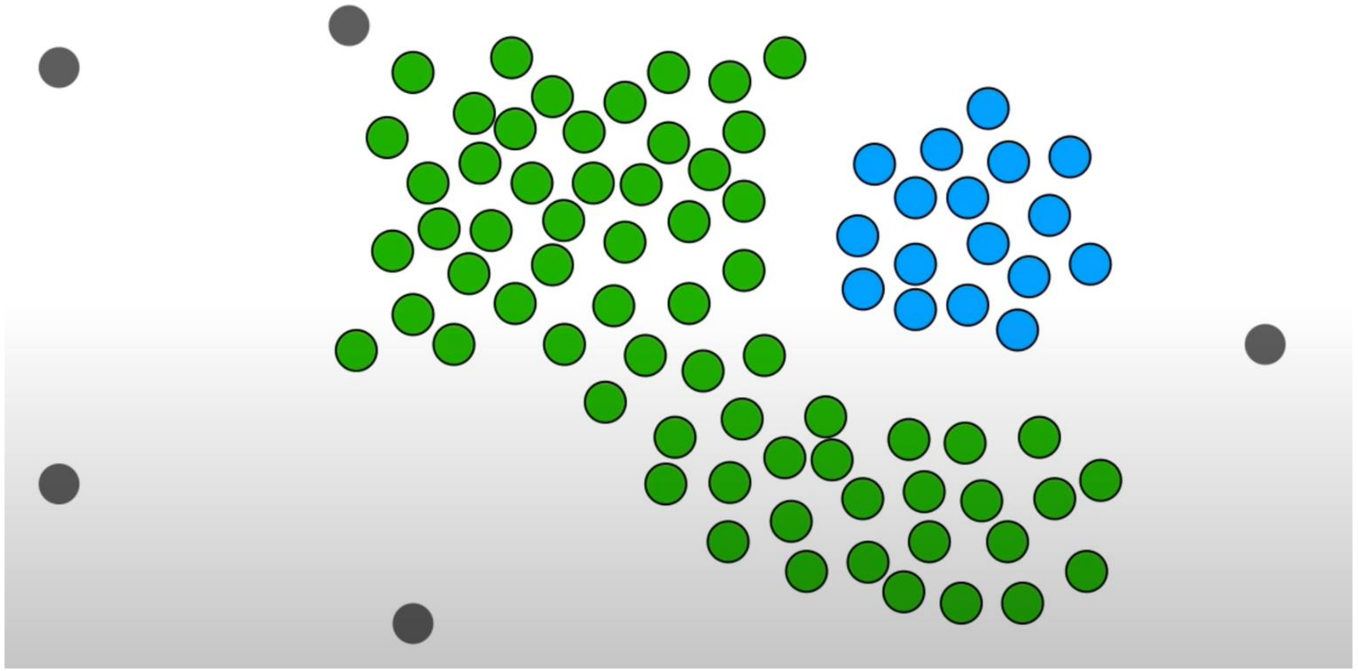
Expand cluster to include all points reachable from core points



Expand cluster to include all points reachable from core points



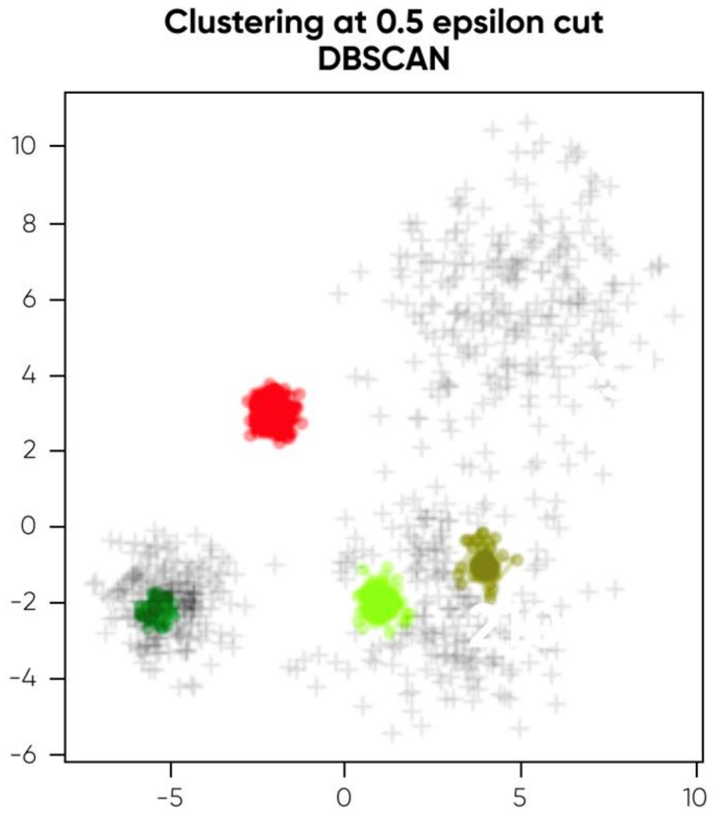
Repeat until all core points are part of a cluster

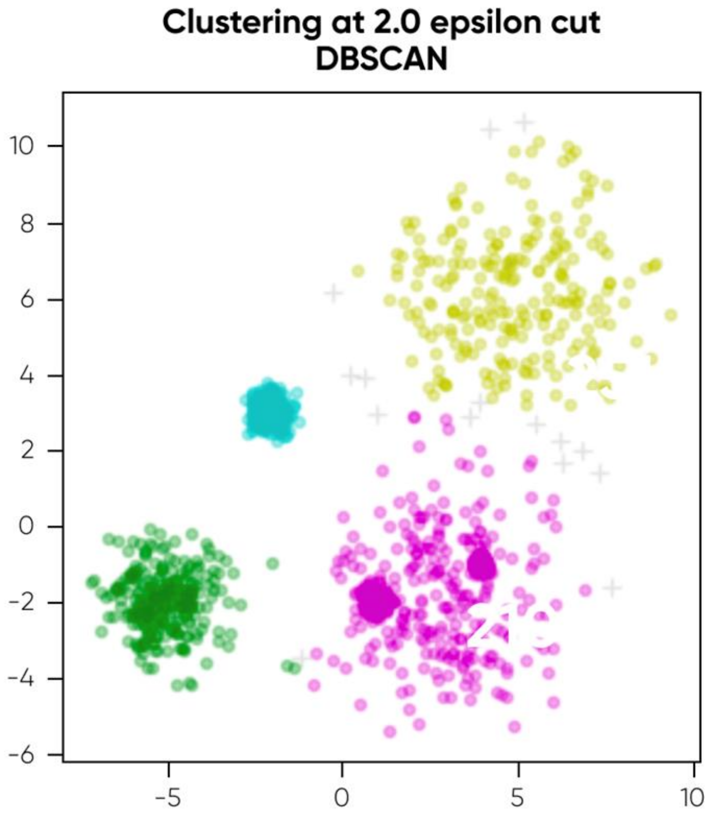


Biases associated with cluster ordering

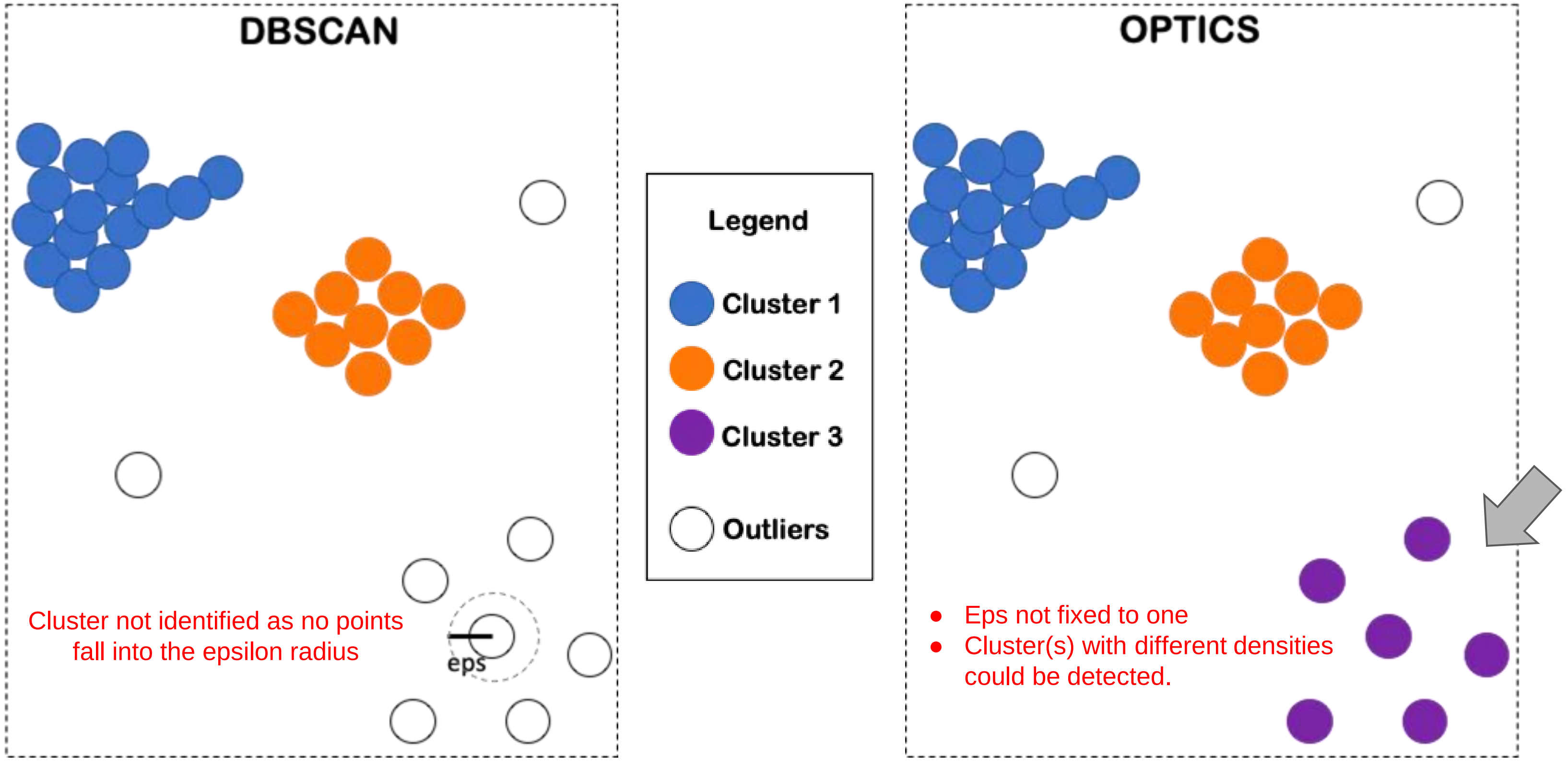
- Core points are not affected by random initialization like with k-means
- Border points will be assigned to the cluster they are first discovered from

DBSCAN is sensitive to the choice of the Eps and MinPts parameters.





DBSCAN does not work well with clusters of **varying densities**.



Cluster not identified as no points fall into the epsilon radius

- Eps not fixed to one
- Cluster(s) with different densities could be detected.

Sensitive to parameter settings — a fixed epsilon

With a broad range of parameter settings

OPTICS overview

- **OPTICS**: Ordering Points to Identify The Clustering Structure
  - Similar to DBSCAN, but it **allows for a range of values for the 'eps' parameter** instead of just one.
  - **The only difference is that we do not assign cluster memberships**.
    - Instead, we store the order in which the objects are processed and the information which would be used by an extended DBSCAN algorithm to assign cluster memberships (if this were at all possible for an infinite number of parameters).
    - **Group points into clusters based on the order of special distances Called "reachability distances" capturing how close and related the points are in the dataset.**

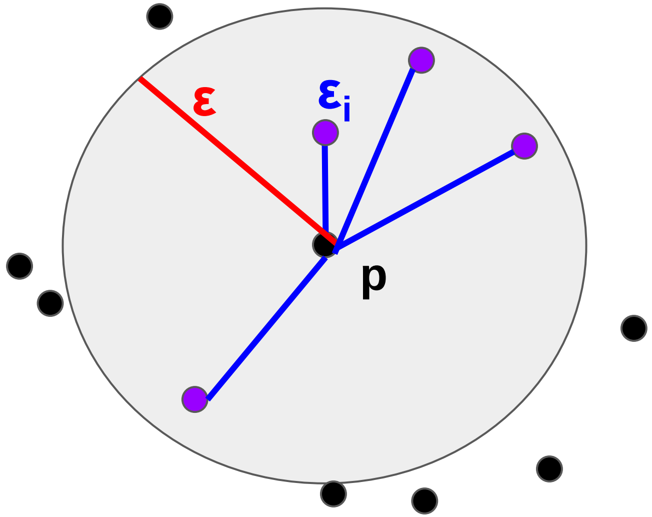
Ankerst et. al., 1999

OPTICS hyperparameters

(1) **An infinite number of distance parameters ε_i** which are smaller than a "**generating distance**" **ε (i.e. $0 < \varepsilon_i <= \varepsilon$)**.
  - epsilon (**ε**) is the maximum search distance.

(1) **MinPts**: Minimum number of neighbors for a point to be a core point.

How do we identify a core point?
i.e., a potential center point
of a cluster.

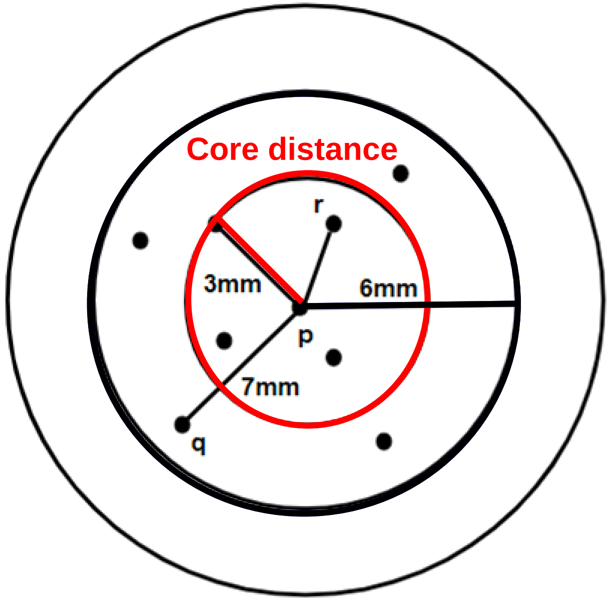


MinPts = 3

$N_\varepsilon(p) = 5$ >= MinPts
Then p is a core point.

Ankerst et. al., 1999

Keywords: Core distance and reachability distance



Core Distance

- The minimum value of radius that holds the MinPts objects from a core point.

Eps = 6mm

MinPts = 5

p is a core point as there are 9 (> 5) points present within the radius of eps (6).

Core_Distance(p) = 3mm

The distance from the core point p to the 5th points as extending the radius from p.

Reachability_Distance(q,p) = 7mm

Reachability_Distance(r,p) = 3mm

Reachability Distance

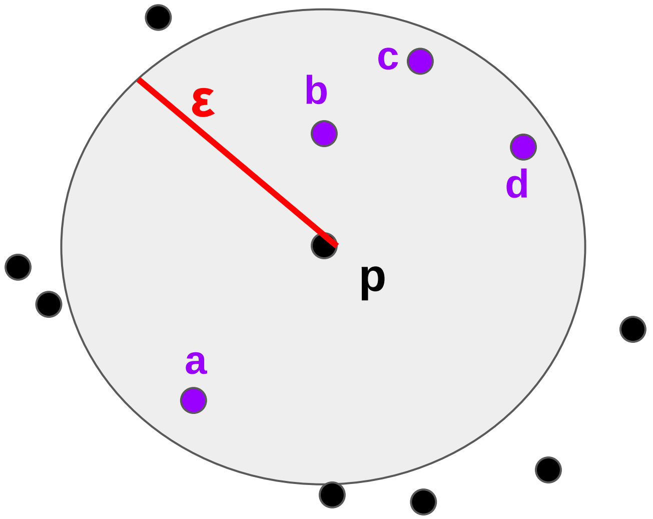
- How easy it is to reach a point from a core point **p**.
- **Take the maximum!**
  The reachability distance(p,q) = max(core_distance(p), distance(q,p))

Algorithmic steps: Identifying the density-connected points

(0) **Identify <u>all core points</u>** in the dataset w.r.t. predefined **ε** and **MinPts.**

(1) For each core point in the dataset, **identify its <u>k-nearest neighbors</u>** (for all k points within the radius of predefined **Ɛ**).

Example



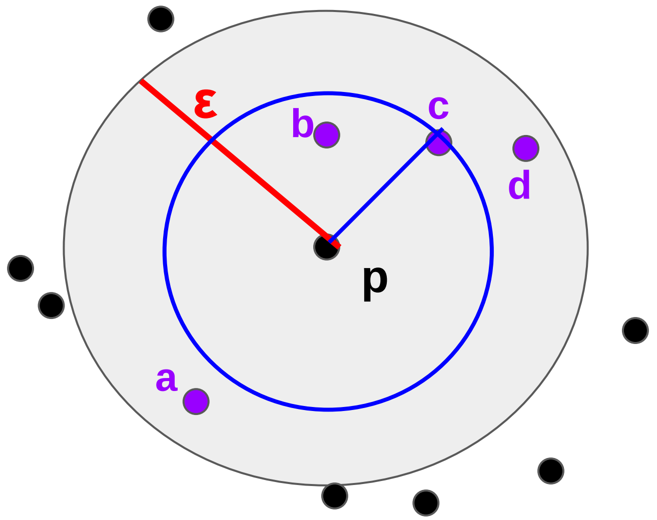
Let p be a core point, then

k-nearest neighbors of p, $N_{k(p)}$ = {a, b, c, d}

Algorithmic steps: Identifying the density-connected points

(2) Iterate through all the core points, starting with **an arbitrary core point p,**

calculate the reachability distance from p to all the k-nearest neighbors of p



Let p be a core point, then

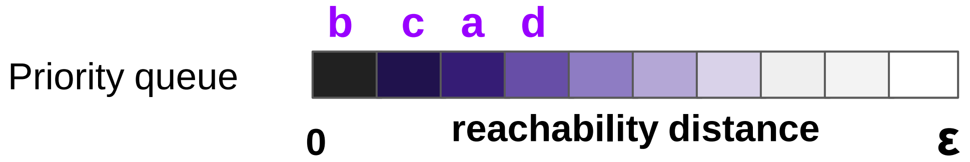
k-nearest neighbors of p, $N_{k(p)}$ = {a, b, c, d}

MinPts = 3, core distance = pc

Reachability distances (RD):

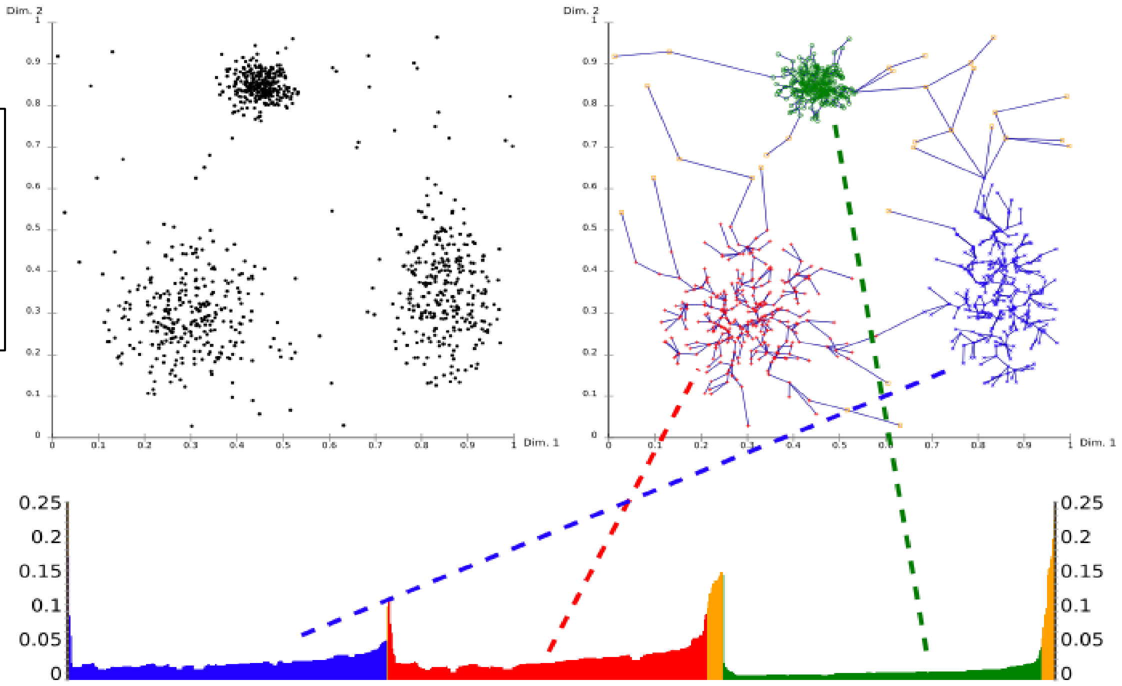
RD_{kn} = {b: pc, c: pc, a: pa, d: pd}

(3) **Order the points** based on their reachability distance using *a priority queue* and create **the reachability plot (shorter RD, high priority – come first).**



Extracting clusters from the reachability plot

Points with similar reachability distances and are close to each other are likely to be in the same cluster.



Reachability distance

The ordering of the points as processed by OPTICS

*** Since points belonging to a cluster have a low reachability distance to their nearest neighbor, the clusters show up as valleys in the reachability plot. The deeper the valley, the denser the cluster.*

Computational implications of using OPTICS vs. DBSCAN

- More memory cost
- Using a range of epsilons, not fixed to one value
- Varying density
- Cluster extraction requires more manual interpretation and decision making
- Higher runtime complexity due to the use of priority queue
  - ~ 1.6x DBSCAN

Ankerst et. al., 1999

Summary

	Pros	**Cons**
K-means clustering	Fast, easily scalable to large datasets	Sensitive to initial cluster centers/number, assumes clusters are spherical
Hierarchical clustering	No need to specify cluster number	Specific use cases, does not scale well (computationally and visually)
Mixture Models	Flexible, can handle overlapping clusters	Sensitive to initial parameters, may converge to local optima
DBSCAN	Doesn't assume any particular shape or size of clusters, robust to outliers	Struggles with clusters of varying densities
OPTICS	Can find clusters of varying densities	Can be slow on very large datasets, more parameter tuning required