

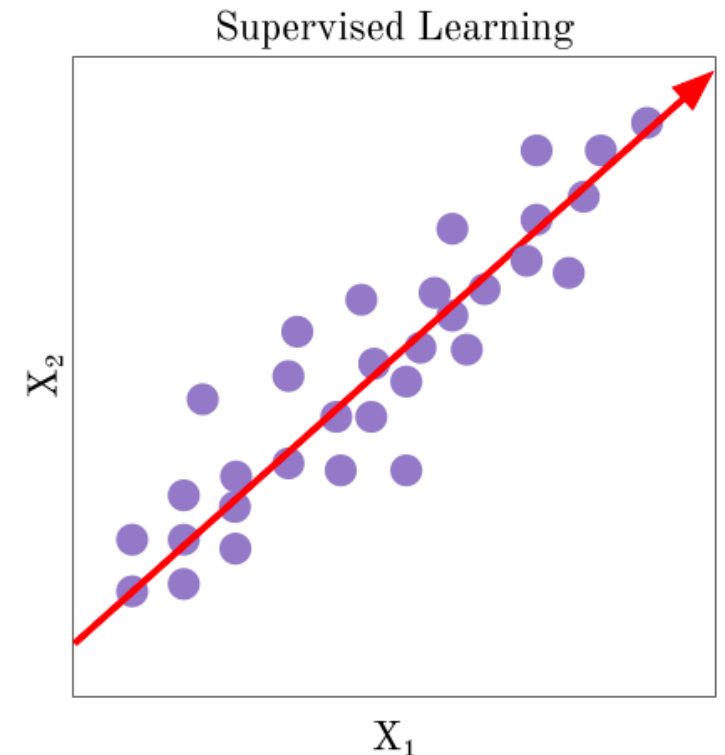
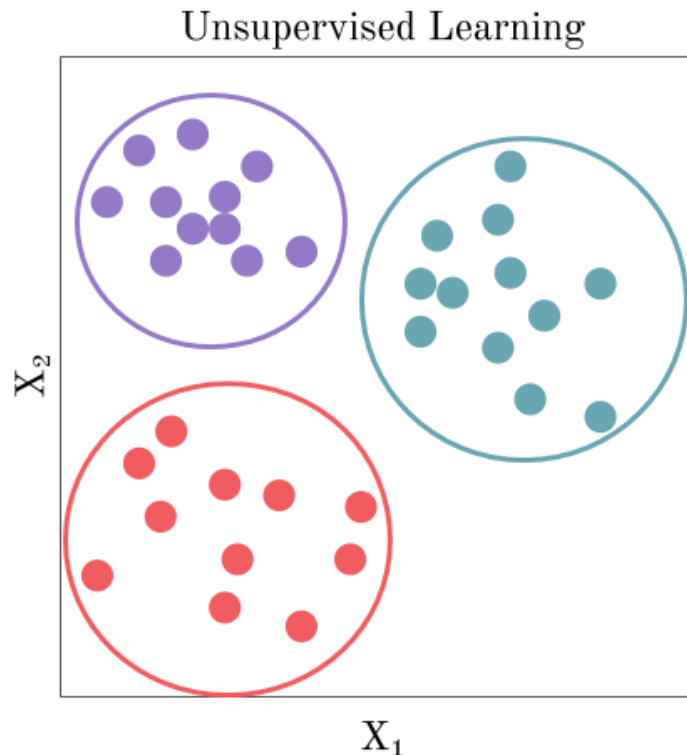
# Unsupervised learning: Clustering

Thursday, October 20, 2022

Ray Pomponio, Trent Hawkins,  
Lucia Guatney, Brenton Graham,  
Dave Johnson

# Unsupervised learning works with *un*-labeled data.

- Labeled data have known, "ground truth" outcome measures
- *Un*-labeled data are lacking known outcomes
- We believe we can group data in a systematic way



# Clustering

Hard Clustering: Results in cluster assignment

Soft Clustering: Results in a vector of probabilities of belonging to a cluster

Hard Clustering	Soft Clustering
K - Means	Mixture Models
Hierarchical	
DBSCAN	
OPTICS	

# Mixture Models: A Batty Motivation

Let's say we have three kinds of bats...



Townsend's Big-eared Bat  
*Corynorhinus townsendii*

<https://www.flickr.com/photos/usfwsqh/6009182505>



Little Brown Bat  
*Myotis lucifugus*

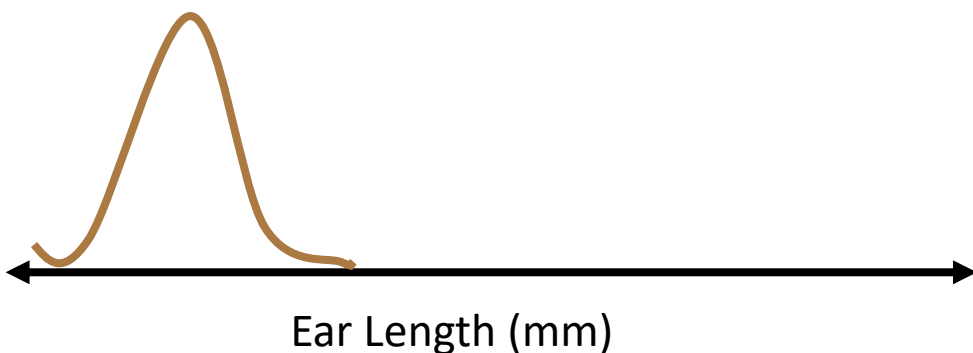
[https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/Little\\_Brown\\_Myotis\\_%28cropped%29.JPG/800px-Little\\_Brown\\_Myotis\\_%28cropped%29.JPG](https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/Little_Brown_Myotis_%28cropped%29.JPG/800px-Little_Brown_Myotis_%28cropped%29.JPG)



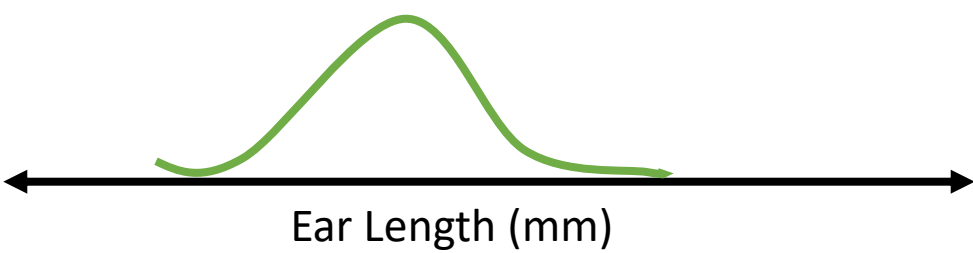
Big Brown Bat  
*Eptesicus fuscus*

[https://upload.wikimedia.org/wikipedia/commons/thumb/6/6c/Big\\_brown\\_bat\\_crawl.png/1024px-Big\\_brown\\_bat\\_crawl.png](https://upload.wikimedia.org/wikipedia/commons/thumb/6/6c/Big_brown_bat_crawl.png/1024px-Big_brown_bat_crawl.png)

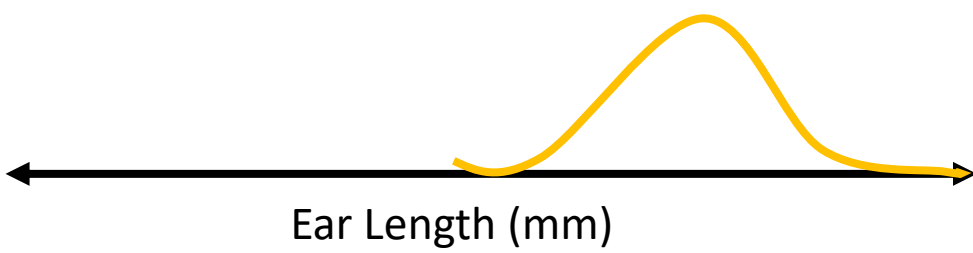
One way to classify them... Ear Length!



$$\sim N(\mu_l, \sigma_l^2)$$

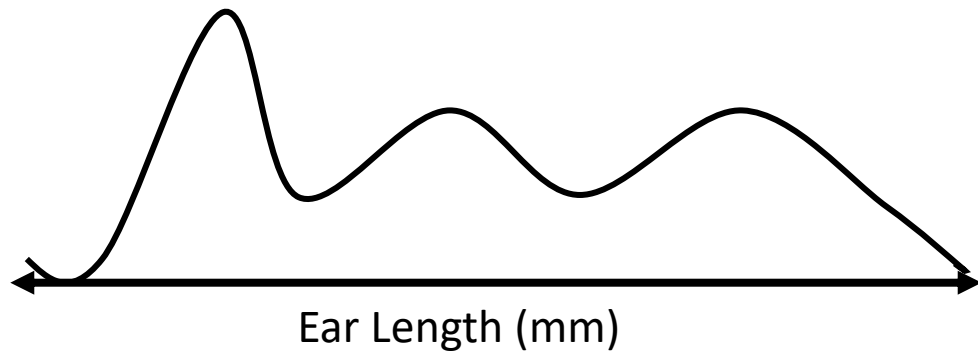


$$\sim N(\mu_b, \sigma_b^2)$$



$$\sim N(\mu_t, \sigma_t^2)$$

But the data we have looks more like...



$$\sim N(\mu_l, \sigma_l^2)$$



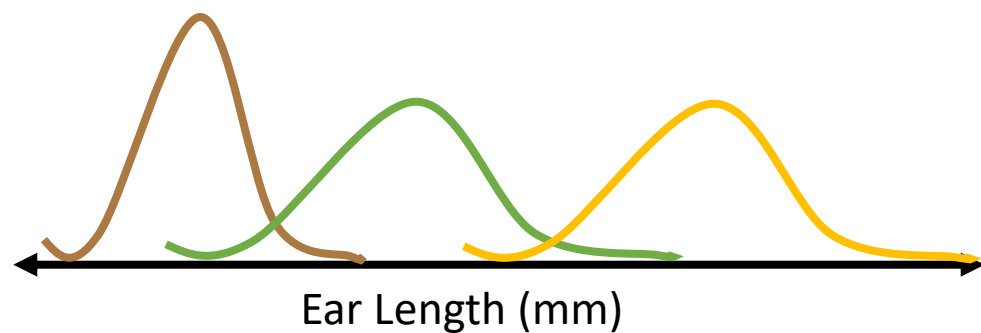
$$\sim N(\mu_b, \sigma_b^2)$$



$$\sim N(\mu_t, \sigma_t^2)$$



We can average over these distributions



But that assumes that each class has the same probability of occurring...



$$\sim N(\mu_l, \sigma_l^2)$$

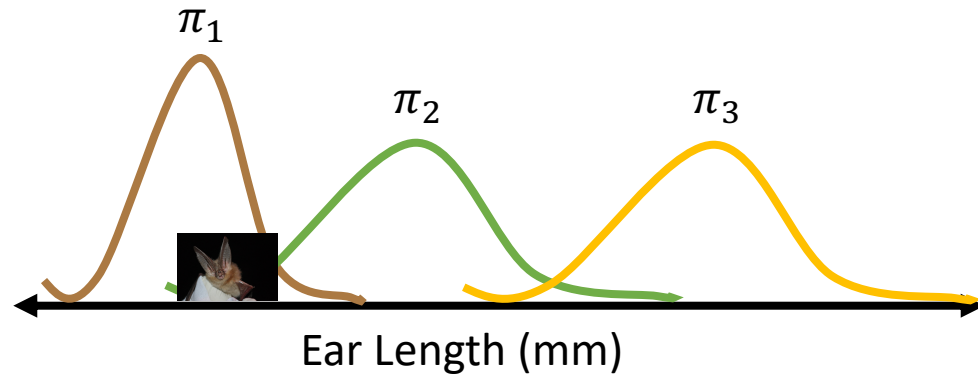


$$\sim N(\mu_b, \sigma_b^2)$$



$$\sim N(\mu_t, \sigma_t^2)$$

So we can add a weight,  $\pi_k$ , to each of our distributions where  $k$  = the total number of classes.



$$\boldsymbol{\pi} = [\pi_1 \ \pi_2 \ \pi_3]$$

The sum over  $\pi_k$  must be equal to 1



$$\sim N(\mu_l, \sigma_l^2)$$



$$\sim N(\mu_b, \sigma_b^2)$$



$$\sim N(\mu_t, \sigma_t^2)$$



# Expanding to Higher Dimensions

In two dimensions:

Ear Length (mm)

Weight (g)



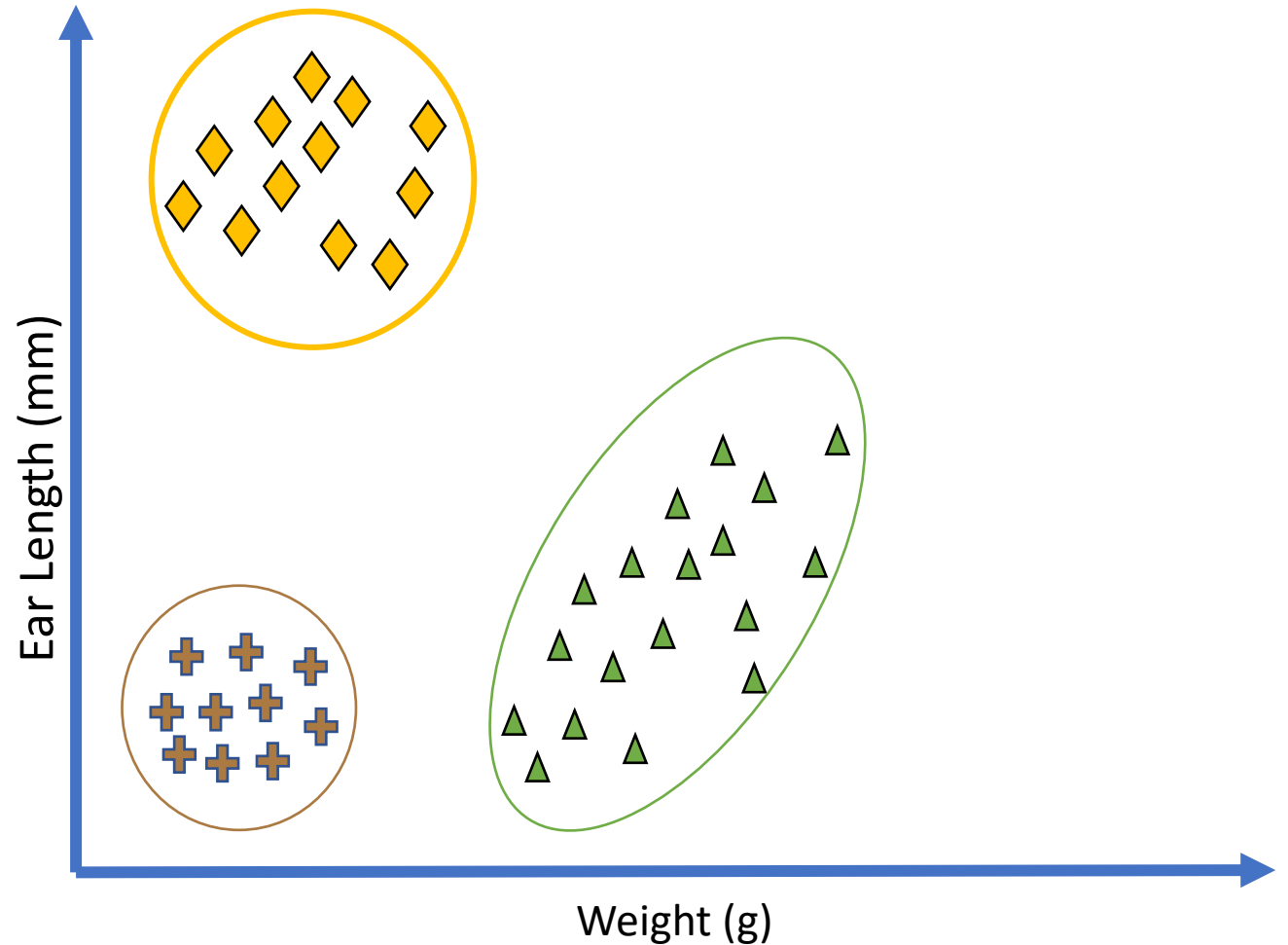
$$\sim MVN(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)$$



$$\sim MVN(\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$$



$$\sim MVN(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$



# Expanding to Higher Dimensions

In two dimensions:



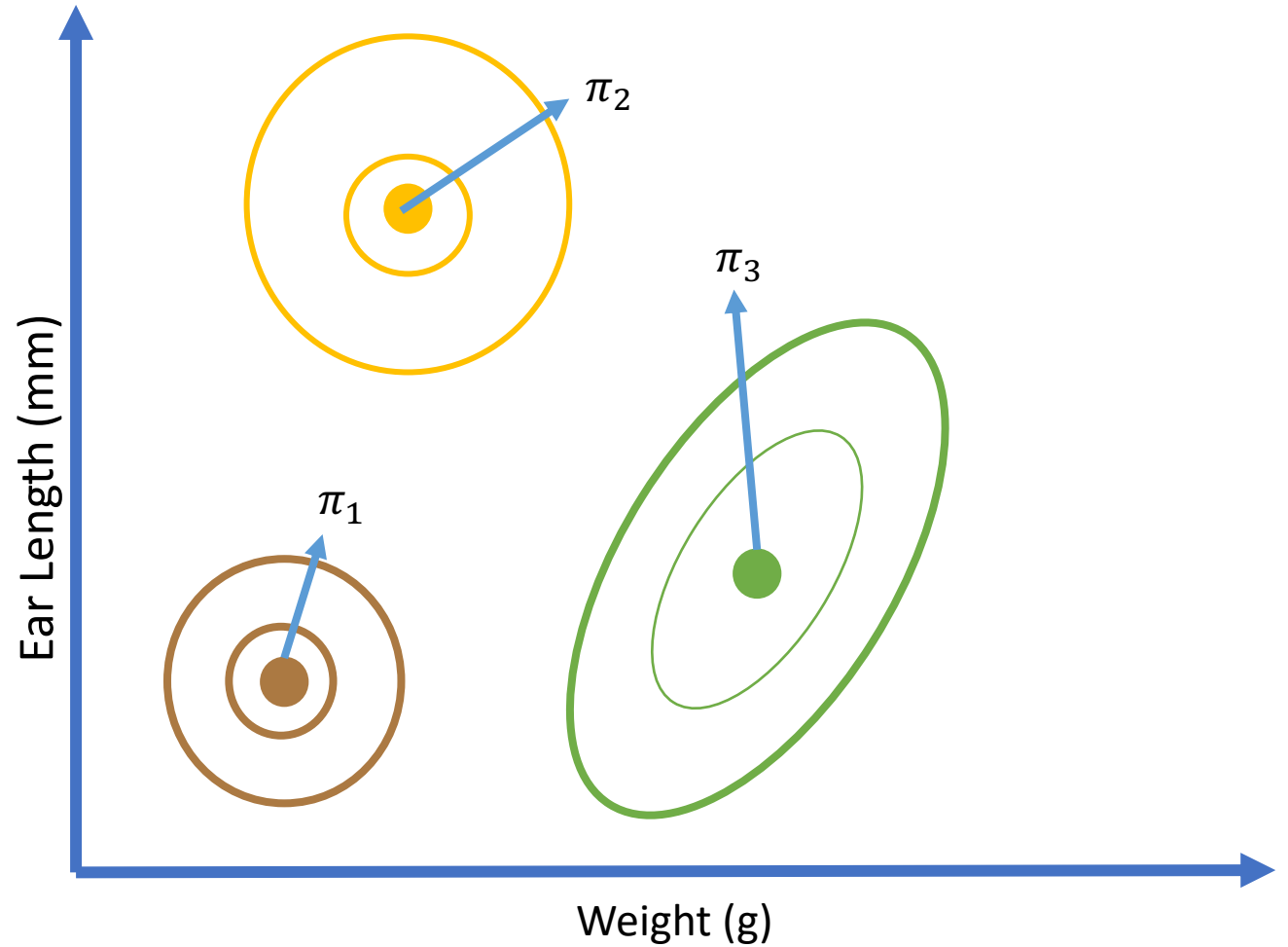
$$\sim MVN(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)$$



$$\sim MVN(\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$$



$$\sim MVN(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$



# What if we only know the parameters of our distribution?

1. Calculate the Likelihood of our data

$$p(\mathbf{X} \mid \pi_k, \mu_k, \Sigma_k)$$

2. Define an indicator variable that will allow us to calculate a probability for each class

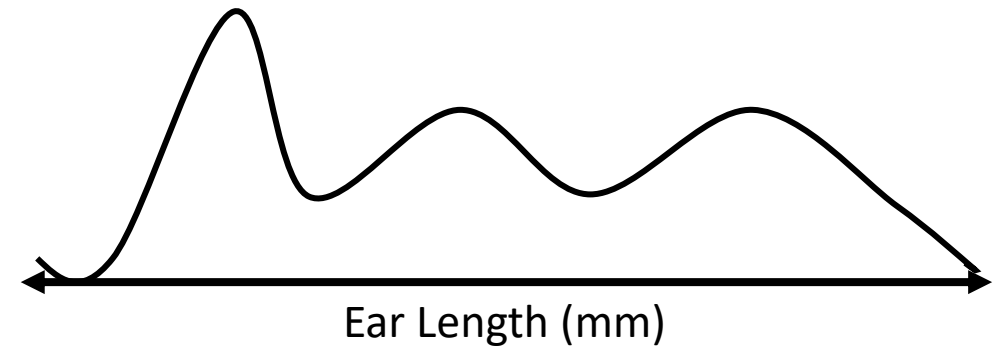
$$Z_{nk} = \begin{cases} 1 & \text{If } x_n \text{ in class } k \\ 0 & \text{Else} \end{cases}$$

3. Find the probability that an observation comes from class K using Baye's Theorem

$$\gamma(Z_{nk}) = p(Z_{nk} = 1 \mid x_n)$$



The data we have...



# What if we only know the parameters of our distribution?

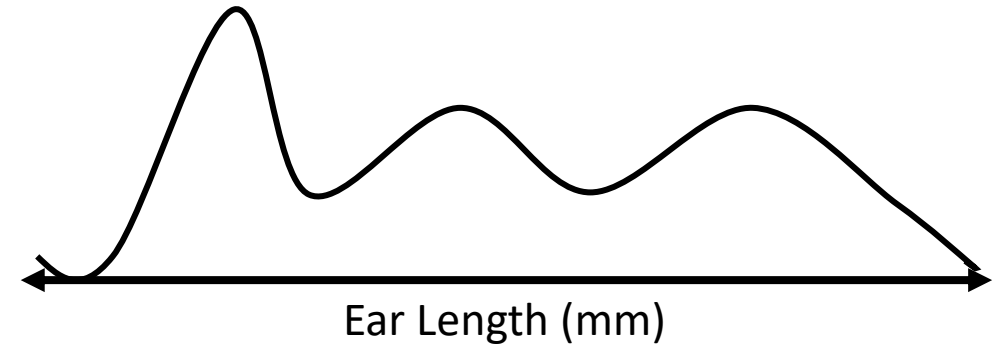
4. Calculate the MLE for each of our parameters

$$p(\pi_k | x_n, \mu_k, \Sigma_k) = f(\gamma(Z_{nk}))$$

$$p(\mu_k | x_n, \pi_k, \Sigma_k) = g(\gamma(Z_{nk}))$$

$$p(\Sigma_k | x_n, \mu_k, \pi_k) = h(\gamma(Z_{nk}))$$

The data we have...

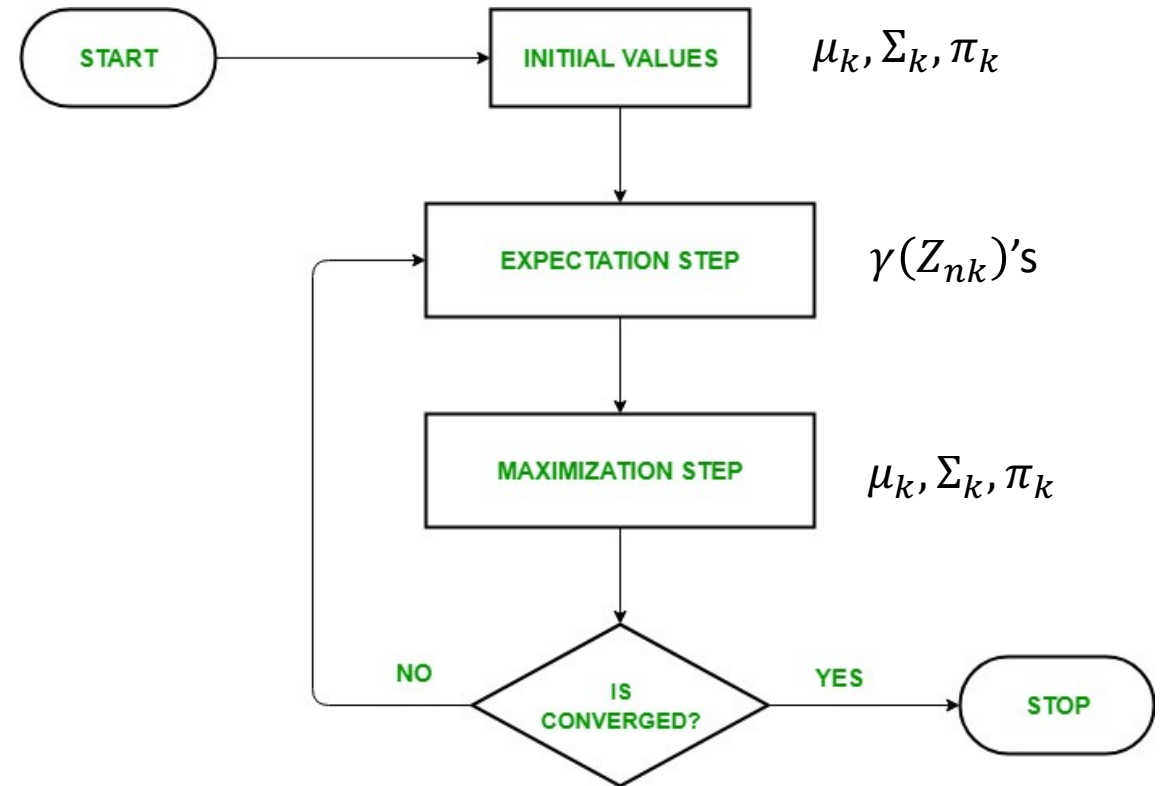


# Expectation – Maximization (EM) Algorithm

It turns out that each of the maximized functions can be expressed in terms of  $\gamma(Z_{nk})$

## Steps

1. Initialize  $\mu_k, \Sigma_k, \pi_k$
2. Compute  $\gamma(Z_{nk})$ 's
3. Update our parameters ( $\mu_k, \Sigma_k, \pi_k$ )
4. Repeat steps 2 & 3 until convergence





# Expectation (E-step)

Calculate the probability that observation  $x_n$  belongs to class  $k$

$$\gamma(Z_{nk}) = p(Z_{nk} = 1 \mid x_n) = \frac{\overset{\pi_k}{\uparrow} p(Z_{nk} = 1) \overset{N(x_n \mid \mu_k, \Sigma_k)}{\uparrow}}{\sum_{j=1}^K \underset{\pi_j}{\downarrow} p(Z_{nj} = 1) \underset{N(x_n \mid \mu_j, \Sigma_k)}{\downarrow}}$$

$\pi_k$  = weight of class  $k$        $\mu_k$  = mean of class  $k$        $\Sigma_k$  = covariance of class  $k$

# Maximization (M-Step)

Update all the weights and parameters using the weighted data calculated expectation

$$m_k = \sum_{\forall n} \gamma(Z_{nk}) \longrightarrow \text{Total Likelihood of Class K}$$

$$\pi_k = \frac{m_k}{m} \longrightarrow \text{Updated weight of class k}$$

$$\mu_k = \frac{1}{m_k} \sum_{\forall n} \gamma(Z_{nk}) x_n \longrightarrow \text{Updated weighted mean of class K}$$

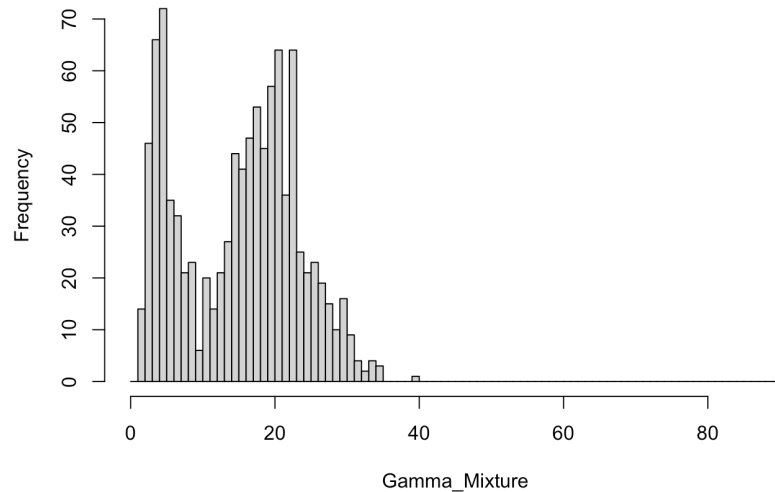
$$\Sigma_k = \frac{1}{m_k} \sum_{\forall n} \gamma(Z_{nk}) (x_n - \mu_k)^T (x_n - \mu_k) \longrightarrow \text{Updated weighted covariance of class K}$$

# Non-Gaussian Distributions

Our data does not have to be Gaussian...

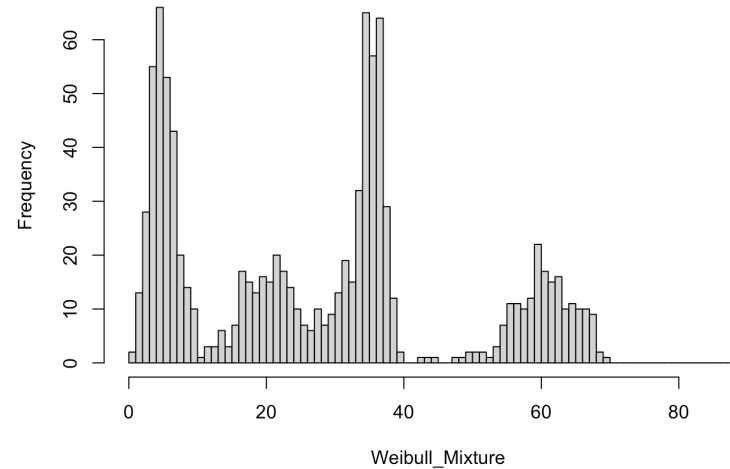
1. We need to know the parameters we want to maximize
2. The function needs to be differentiable (i.e. we can calculate maximum likelihood)

Gamma



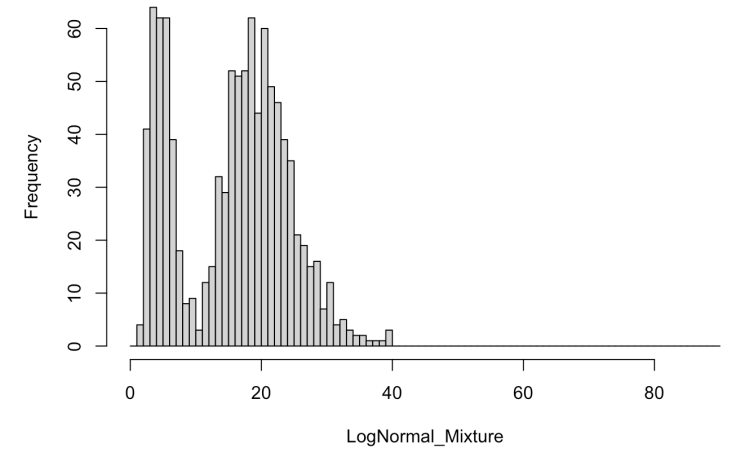
$\alpha, \beta$

Weibull



$\lambda, k$

Log-Normal



$\mu, \sigma^2$

# How many clusters (k) should we assume?

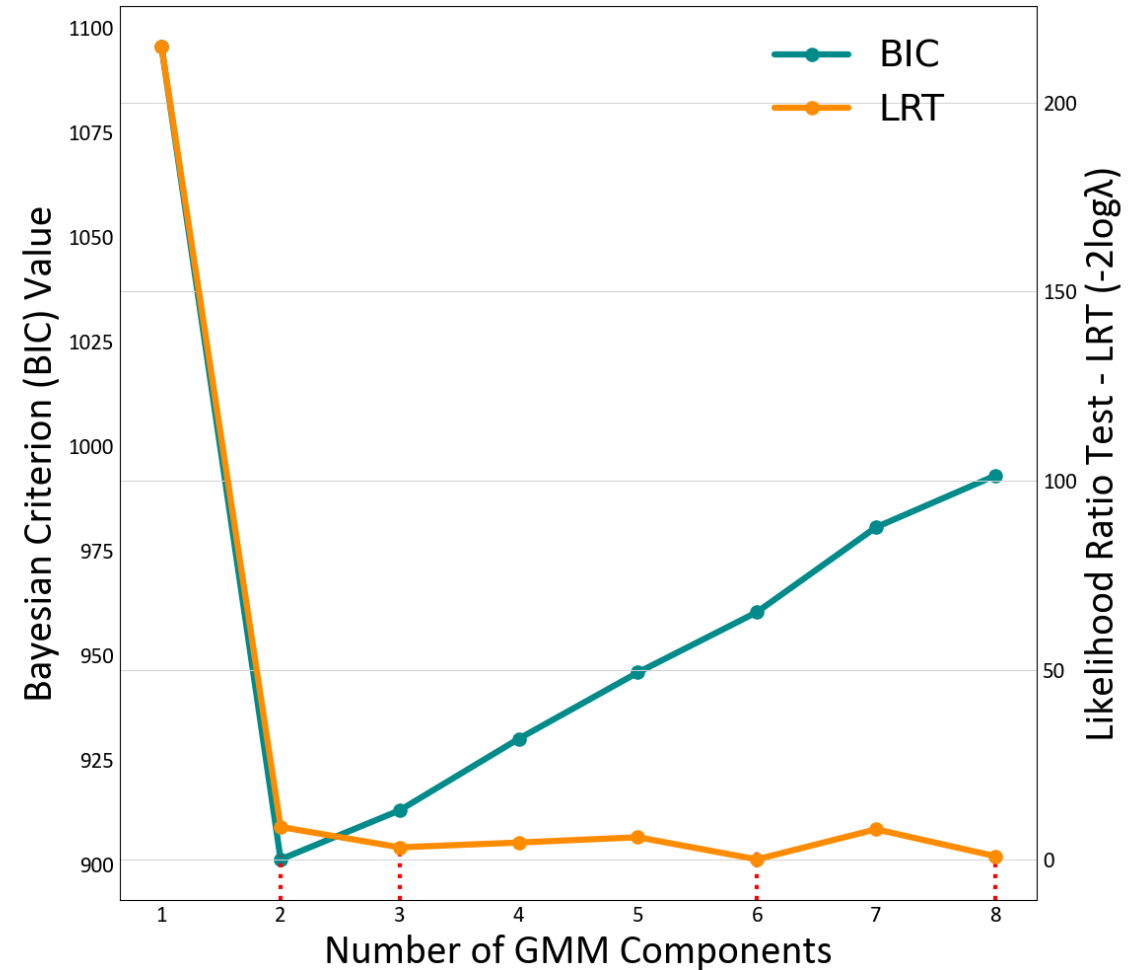
Many ways to evaluate K:

## 1. Bayesian Information Criterion (BIC)

- Favors parsimony
- May be overly conservative in some scenarios

## 2. Likelihood Ratio Test

- Similar to an 'elbow' plot
- Still susceptible to selecting a high number of components and overfitting



<https://geostatisticslessons.com/lessons/gmm>

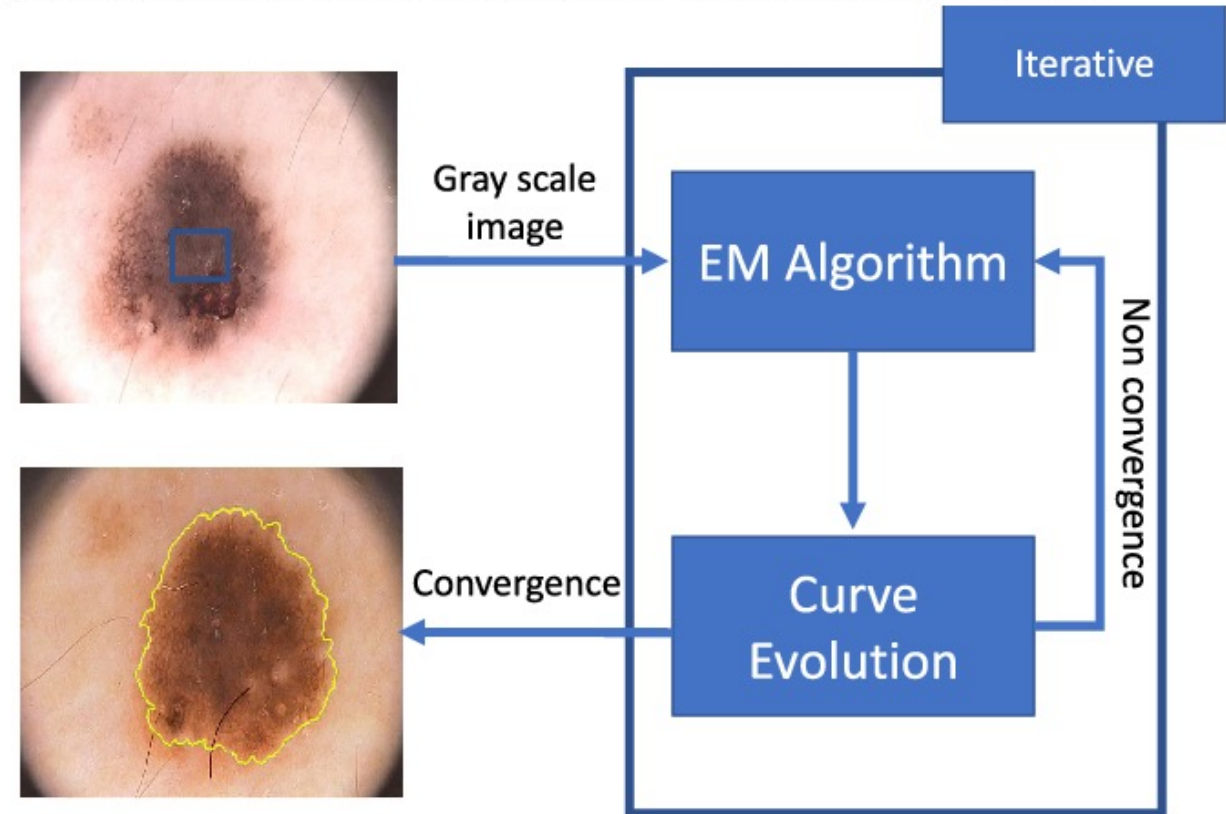
# Examples in Biomedicine

Medical imaging is one of the most common applications of GMMs:

1. Dermoscopy
2. MRI
3. Chromoendoscopy
4. Radiology
5. Etc...

## Gaussian Mixture Model Based Probabilistic Modeling of Images for Medical Image Segmentation

FARHAN RIAZ<sup>ID1</sup>, SAAD REHMAN<sup>ID1</sup>, MUHAMMAD AJMAL<sup>ID2</sup>, REHAN HAFIZ<sup>ID3</sup>,  
ALI HASSAN<sup>1</sup>, NAIF RADI ALJOHANI<sup>ID4</sup>, RAHEEL NAWAZ<sup>ID5</sup>, RUPERT YOUNG<sup>ID6</sup>,  
AND MIGUEL COIMBRA<sup>ID7</sup>



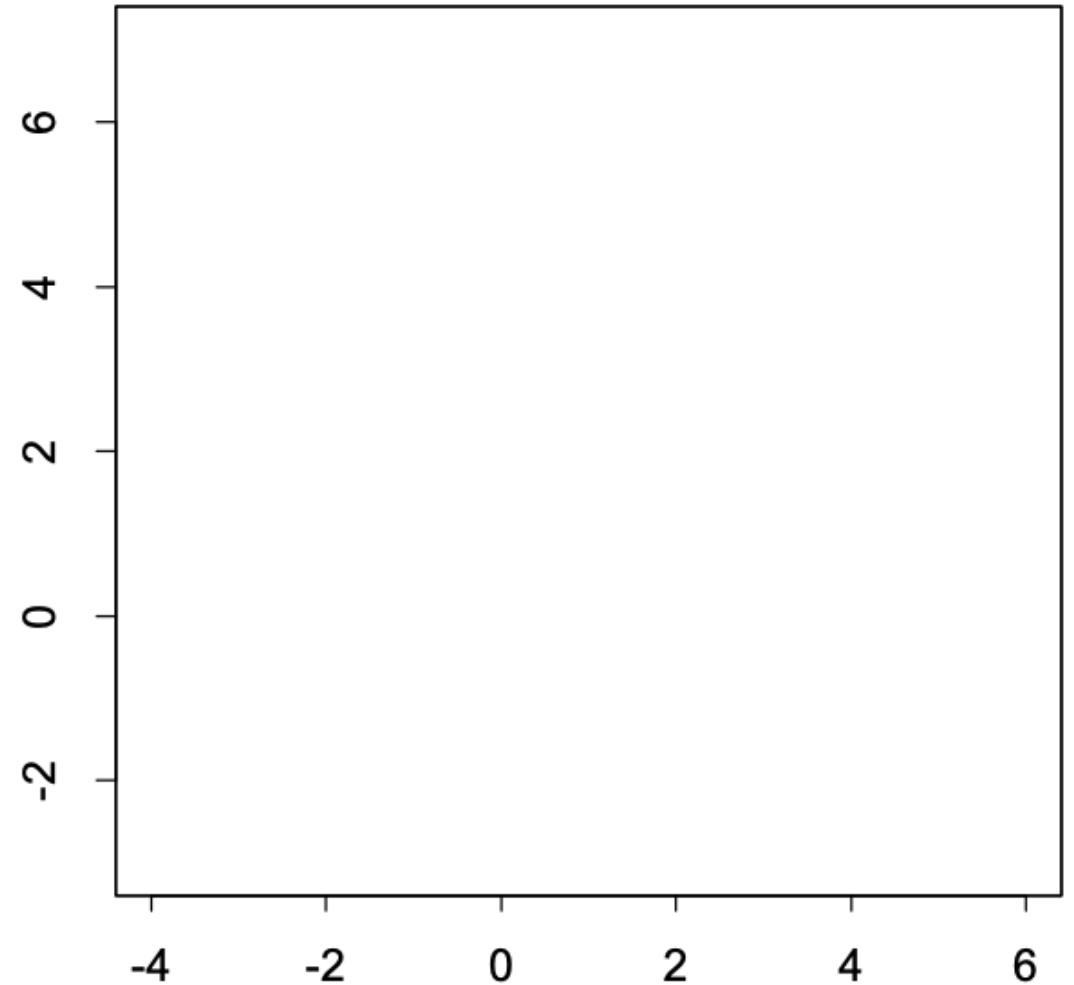


# Limitations

1. Does not scale well in higher dimensions
  - Complexity =  $O(NKD^3)$ 
    - N = Sample Size
    - K = Clusters
    - D = Dimensionality
2. The E-M algorithm is sensitive to starting values
  - Could converge to local max.
3. Sensitive to data transformations
4. Sensitive to outliers

# K-Means Clustering

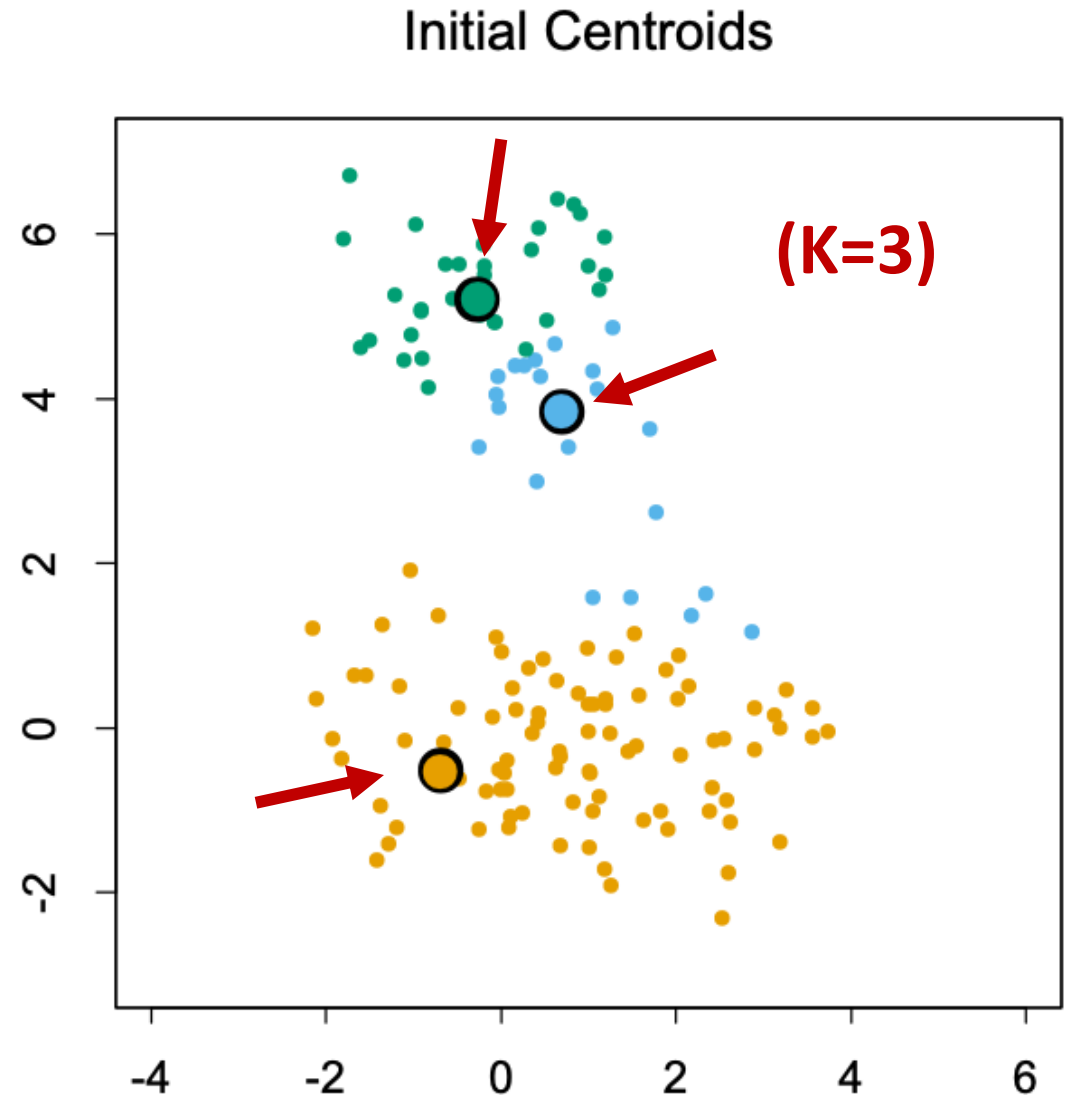
*Imagine we have two continuous features.*



# K-Means Clustering

*Imagine we have two continuous features.*

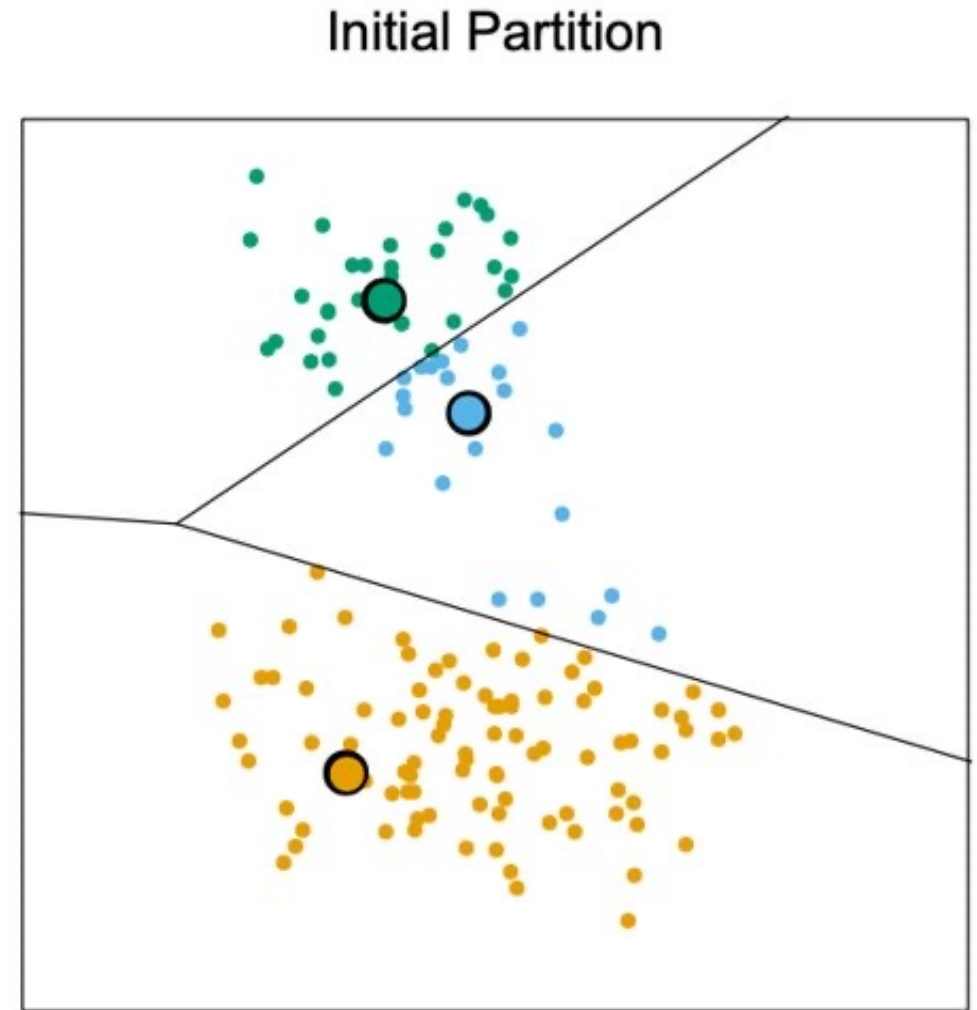
1. Start with initial set of cluster centers



# K-Means Clustering

*Imagine we have two continuous features.*

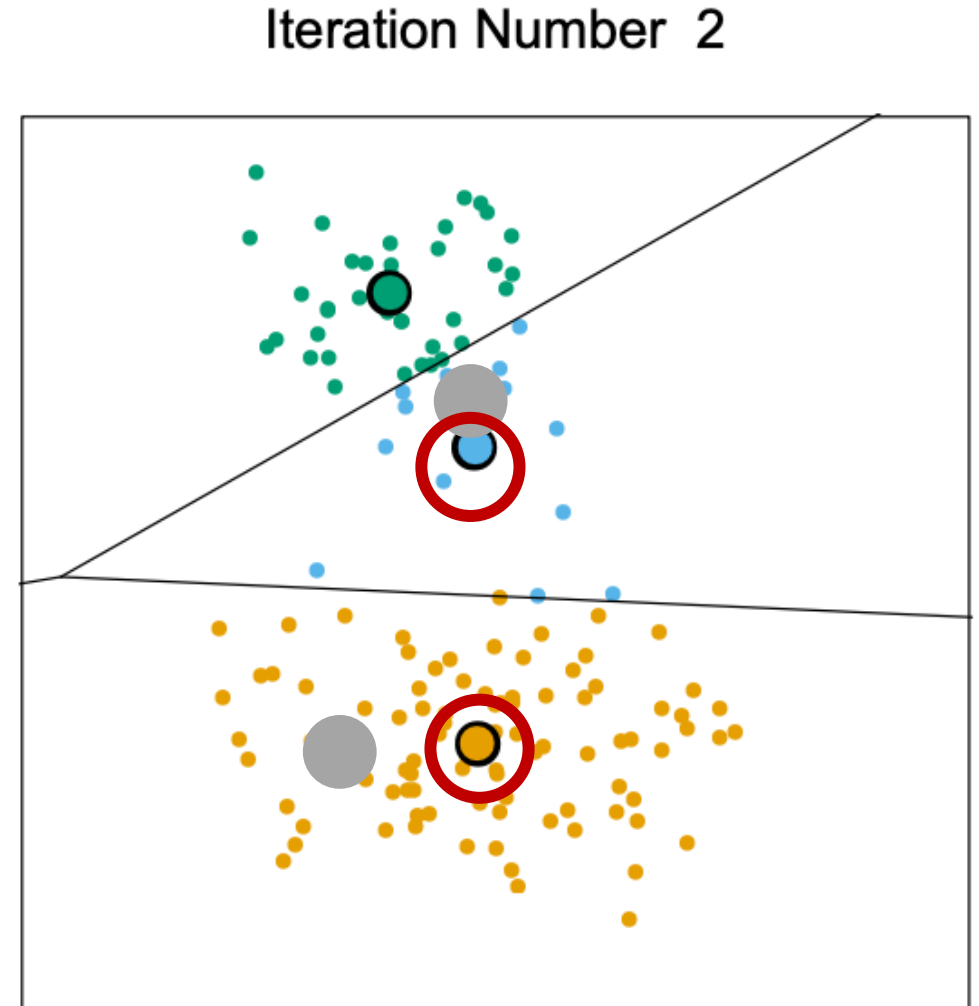
1. Start with initial set of cluster centers
2. Identify subset of data points closer to center  $k_i$  than any other  $k_{j \neq i}$



# K-Means Clustering

*Imagine we have two continuous features.*

1. Start with initial set of cluster centers
2. Identify subset of data points closer to center  $k_i$  than any other  $k_{j \neq i}$
3. Compute means of each feature within cluster  $i$
4. Reassign cluster center  $i$  to the feature-wise means

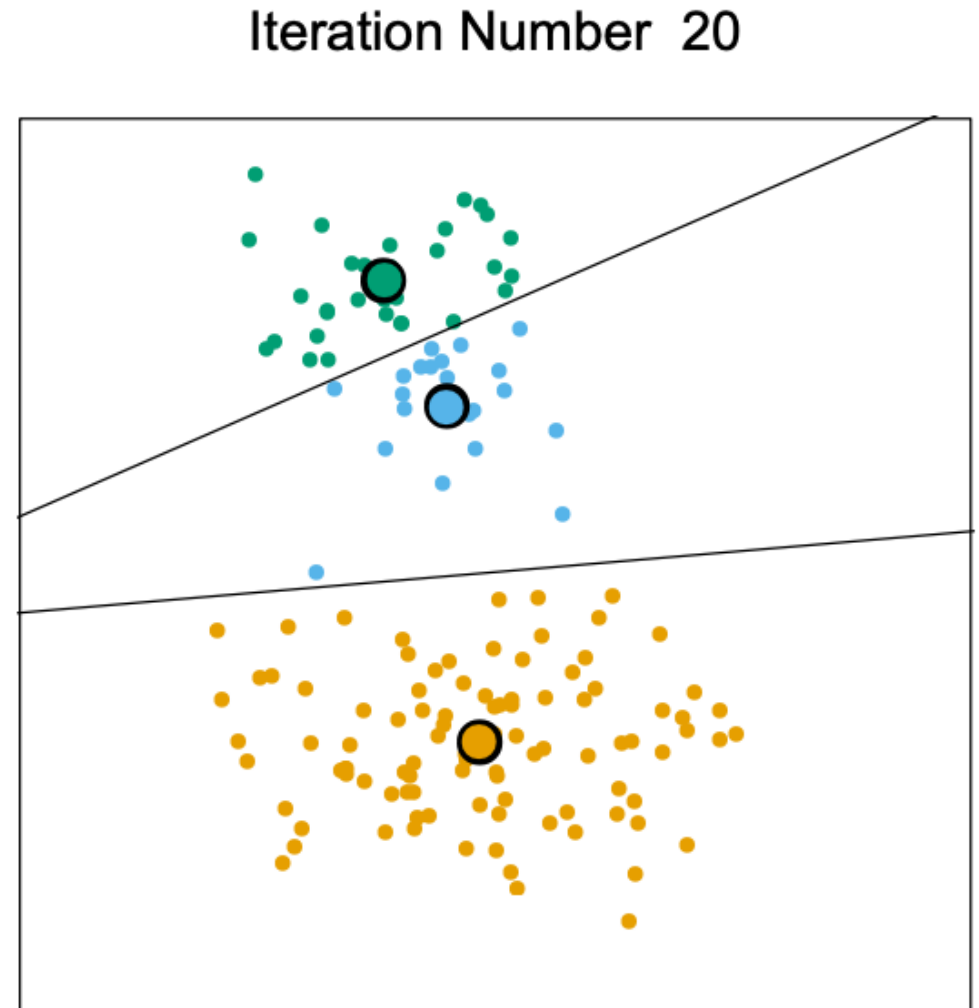




# K-Means Clustering

*Imagine we have two continuous features.*

1. Start with initial set of cluster centers
2. Identify subset of data points closer to center  $k_i$  than any other  $k_{j \neq i}$
3. Compute means of each feature within cluster  $i$
4. Reassign cluster center  $i$  to the feature-wise means
5. Repeat steps 2-4 until convergence



# Algorithm considerations

- Initialization of cluster centers
- Distance calculation
- Convergence criterion
- Selecting the optimal number of clusters

# Algorithm considerations

- **Initialization of cluster centers**
- Distance calculation
- Convergence criterion
- Selecting the optimal number of clusters
- A common approach is to pick  $K$  random data point to serve as initial cluster centers
- One needs to try different starting values as solutions may vary

# Algorithm considerations

- Initialization of cluster centers
- **Distance calculation**
- Convergence criterion
- Selecting the optimal number of clusters

- Euclidean distance

$$d(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2}$$

for  $i=1, \dots, n$  features

- Manhattan distance

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

for  $i=1, \dots, n$  features

# Algorithm considerations

- Initialization of cluster centers
- **Distance calculation**
- Convergence criterion
- Selecting the optimal number of clusters

## Pros/Cons

- Euclidean distance tend to become inflated in high-dimensional spaces
- Euclidean distance will exaggerate *discrepant* differences, i.e., larger distances in specific dimensions
- Manhattan distance is less influenced by outlier distances



# Algorithm considerations

- Initialization of cluster centers
- Distance calculation
- **Convergence criterion**
- Selecting the optimal number of clusters

- Scikit-learn default is the *Frobenius norm* of the difference in cluster centers ( $\text{tol}=0.0001$ )

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$$

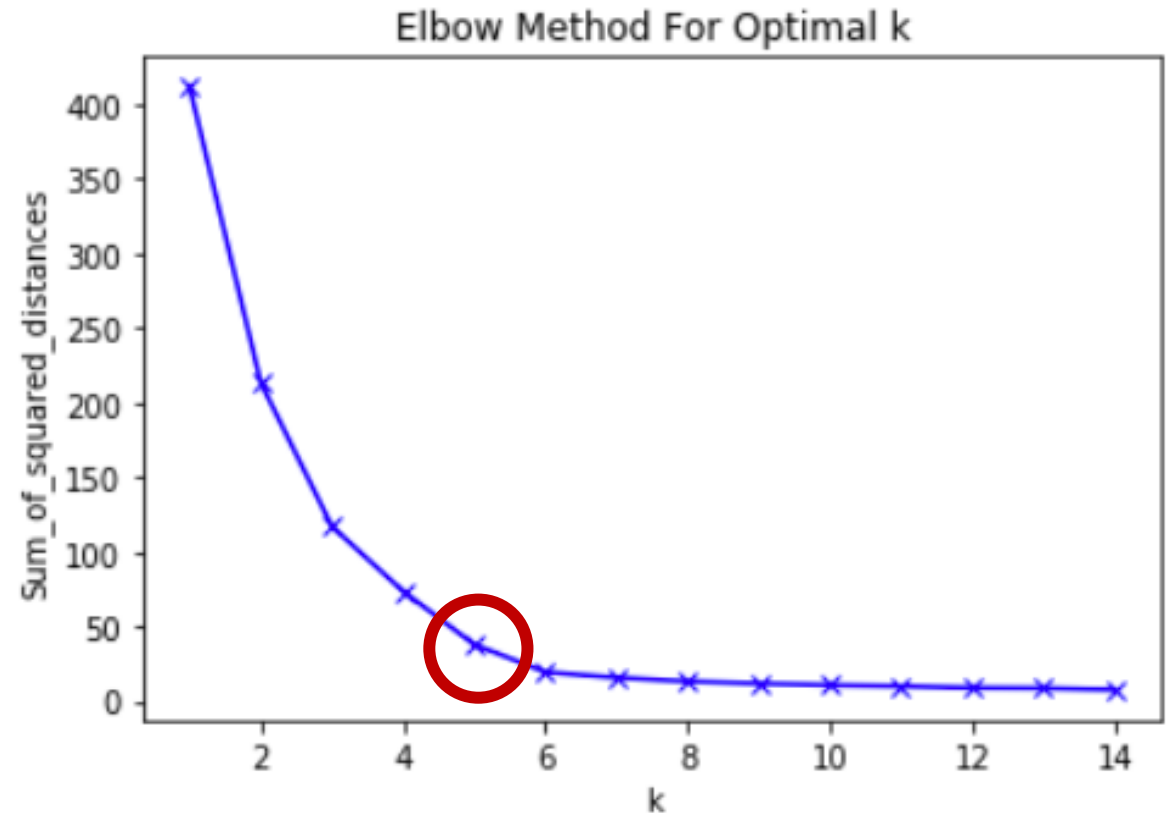
- Local minima may be encountered
- Hartigan-Wong algorithm searches for local solutions that improve the objective function, making it less prone to local minima.

# Algorithm considerations

- Initialization of cluster centers
- Distance calculation
- Convergence criterion
- **Selecting the optimal number of clusters**
- Use an "elbow method" to evaluate the smallest number of clusters that yields high coherence
- Coherence will be based on performance metrics (see final slides)
- The following slide illustrates an example with the sum of squared distances (aka *inertia*)

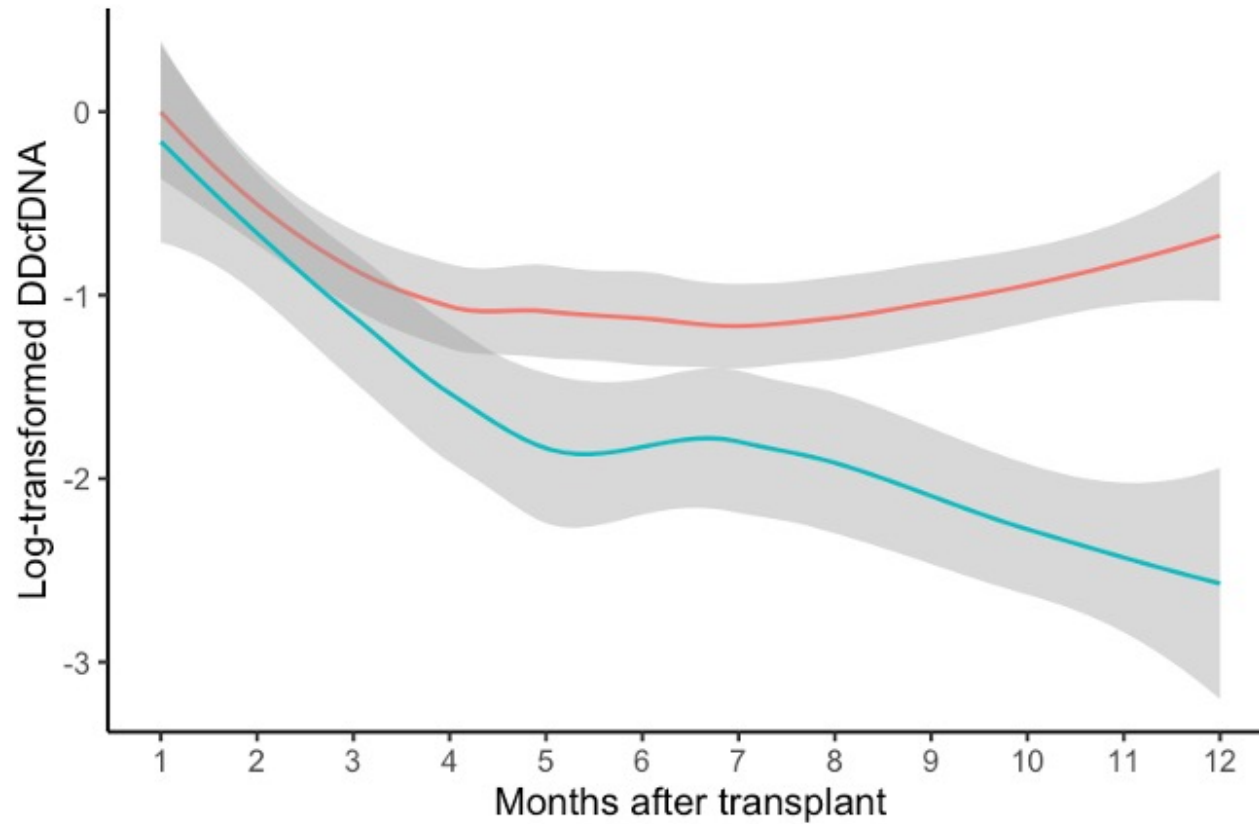
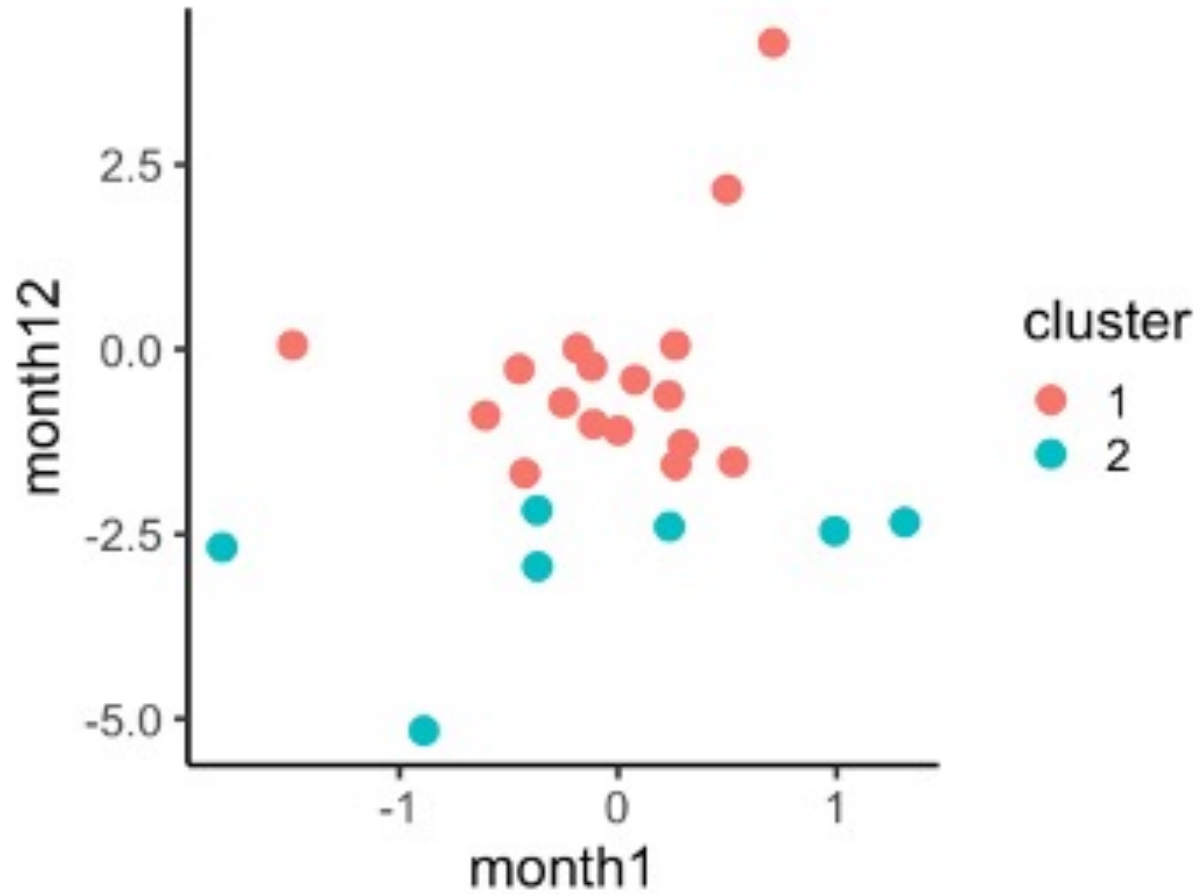
# Algorithm considerations

- Initialization of cluster centers
- Distance calculation
- Convergence criterion
- **Selecting the optimal number of clusters**

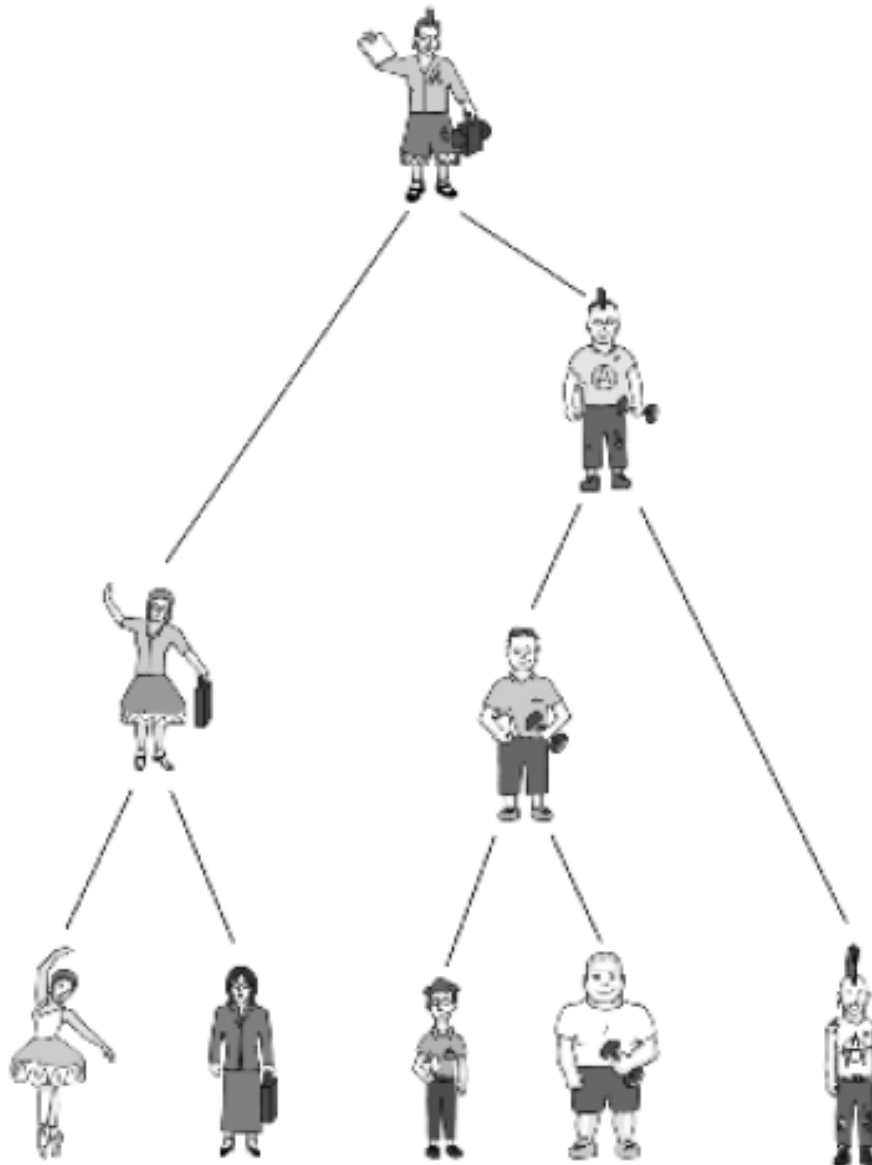


<https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f>

# Clustering lung-transplant recipients based on longitudinal biomarker values



# Hierarchical Clustering



This is a dendrogram

- Each leaf is a cluster of membership one
- Each branch point is a new cluster
- Branch lengths indicate distance

Image from Jones and Pevzner  
(2004) *An Introduction to  
Bioinformatics Algorithms*.  
Lyrics from Lavigne (2002) *Sk8er  
Boi*.

# Agglomerative vs. Divisive

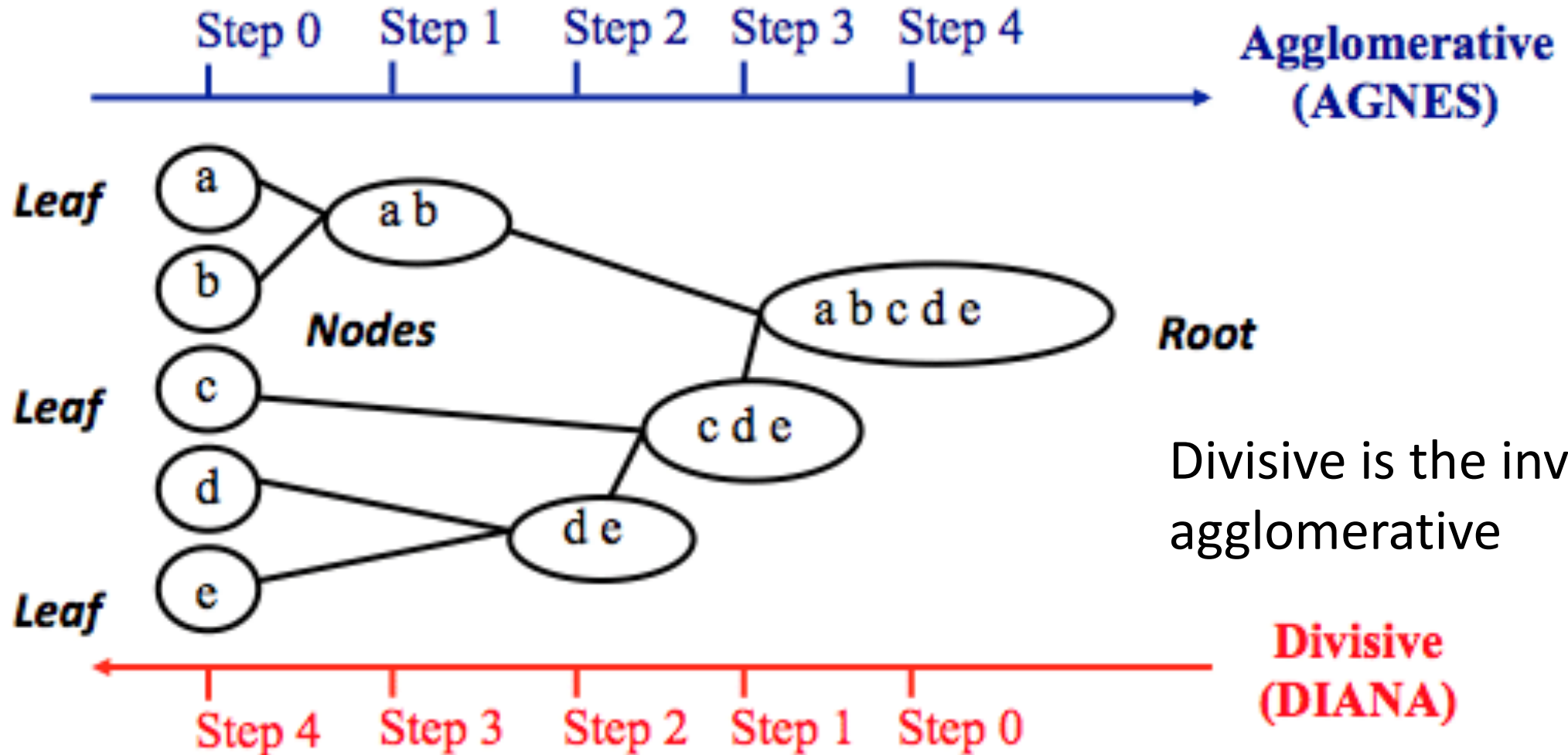
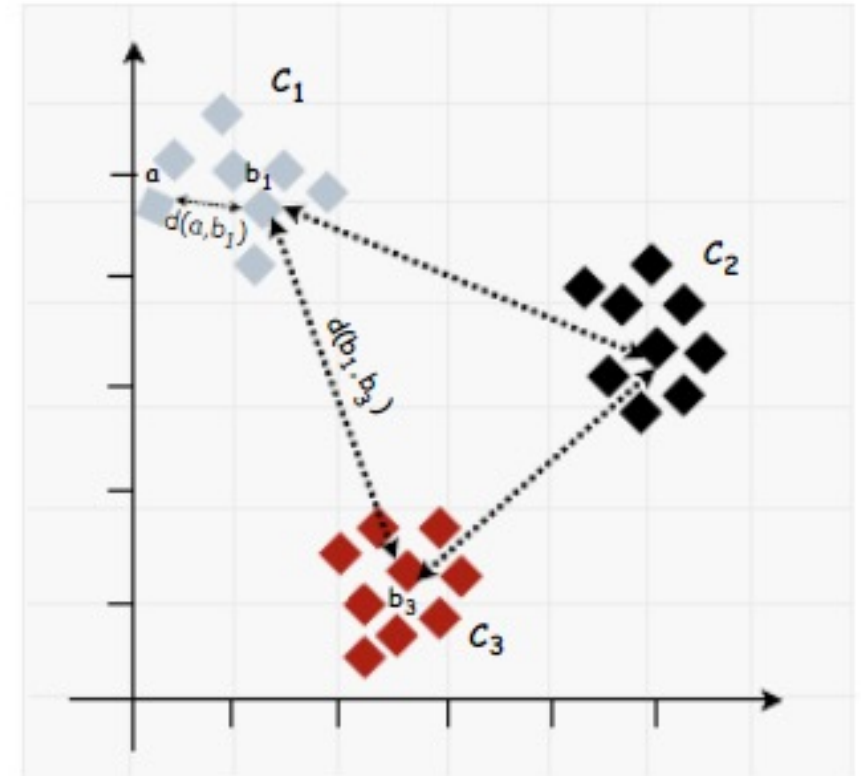


Image from Kamande et al. (2018). *International Journal of Computer Applications*.

# Agglomerative Algorithm

HIERARCHICALCLUSTERING( $\mathbf{d}, n$ )

- 1 Form  $n$  clusters, each with 1 element
- 2 Construct a graph  $T$  by assigning an isolated vertex to each cluster
- 3 **while** there is more than 1 cluster
- 4     Find the two closest clusters  $C_1$  and  $C_2$
- 5     Merge  $C_1$  and  $C_2$  into new cluster  $C$  with  $|C_1| + |C_2|$  elements
- 6     Compute distance from  $C$  to all other clusters
- 7     Add a new vertex  $C$  to  $T$  and connect to vertices  $C_1$  and  $C_2$
- 8     Remove rows and columns of  $\mathbf{d}$  corresponding to  $C_1$  and  $C_2$
- 9     Add a row and column to  $\mathbf{d}$  for the new cluster  $C$
- 10 **return**  $T$



Algorithm from Jones and Pevzner (2004) *An Introduction to Bioinformatics Algorithms*. Image from Aljarah et al. (2021) *Evolutionary Data Clustering: Algorithms and Applications*.

# Distance

		Samples:	
		#1	#2
Gene 1		1.6	0.5
Gene 2		-0.5	-1.9

Here is an example of calculating distance from gene expression data for use in hierarchical clustering. You can use any distance metric whatsoever. Let your imagination run wild!

The Euclidean distance  
between Genes 1 and 2.



$$\sqrt{(\text{difference in sample \#1})^2 + (\text{difference in sample \#2})^2}$$

Image from Starmer *StatQuest*.



# Intercluster Distance

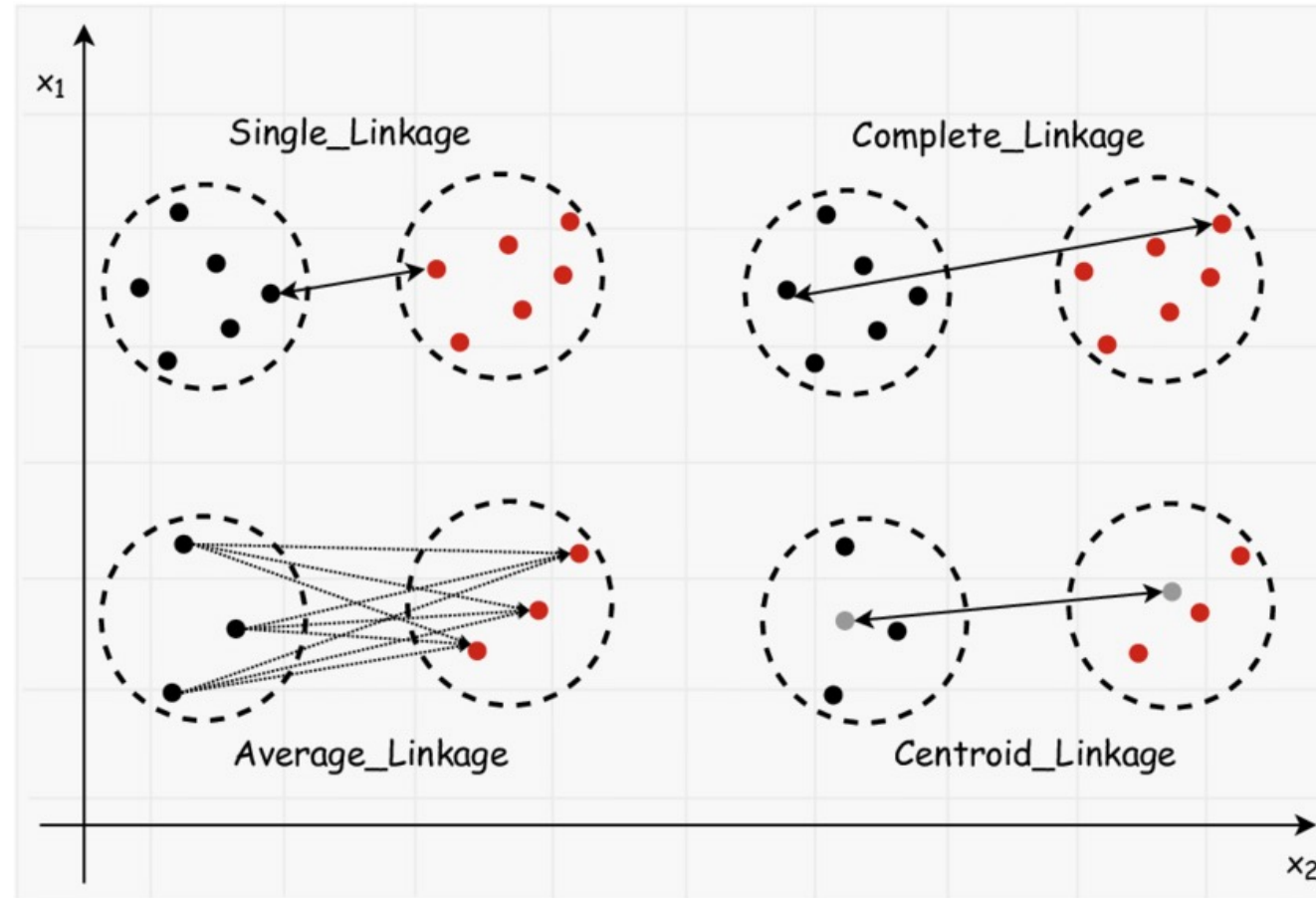


Image from Aljarah et al. (2021) *Evolutionary Data Clustering: Algorithms and Applications*.

# Applications

Hierarchical clustering on gene expression data has separated healthy and disease state patients

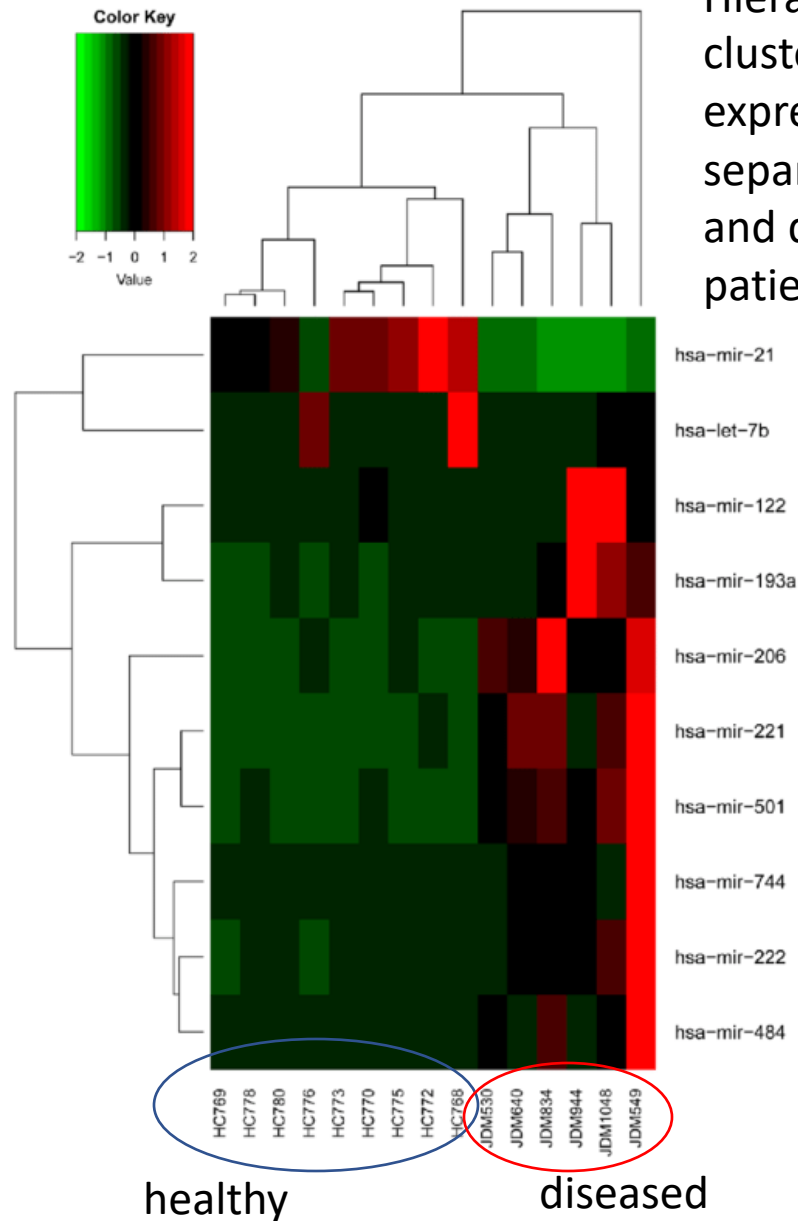


Image from Jiang et al. (2019) *Pediatric Rheumatology*.

Phylogeny is the oldest biological use of hierarchical clustering. Here they also used metabolic distance. Let your imagination run wild!

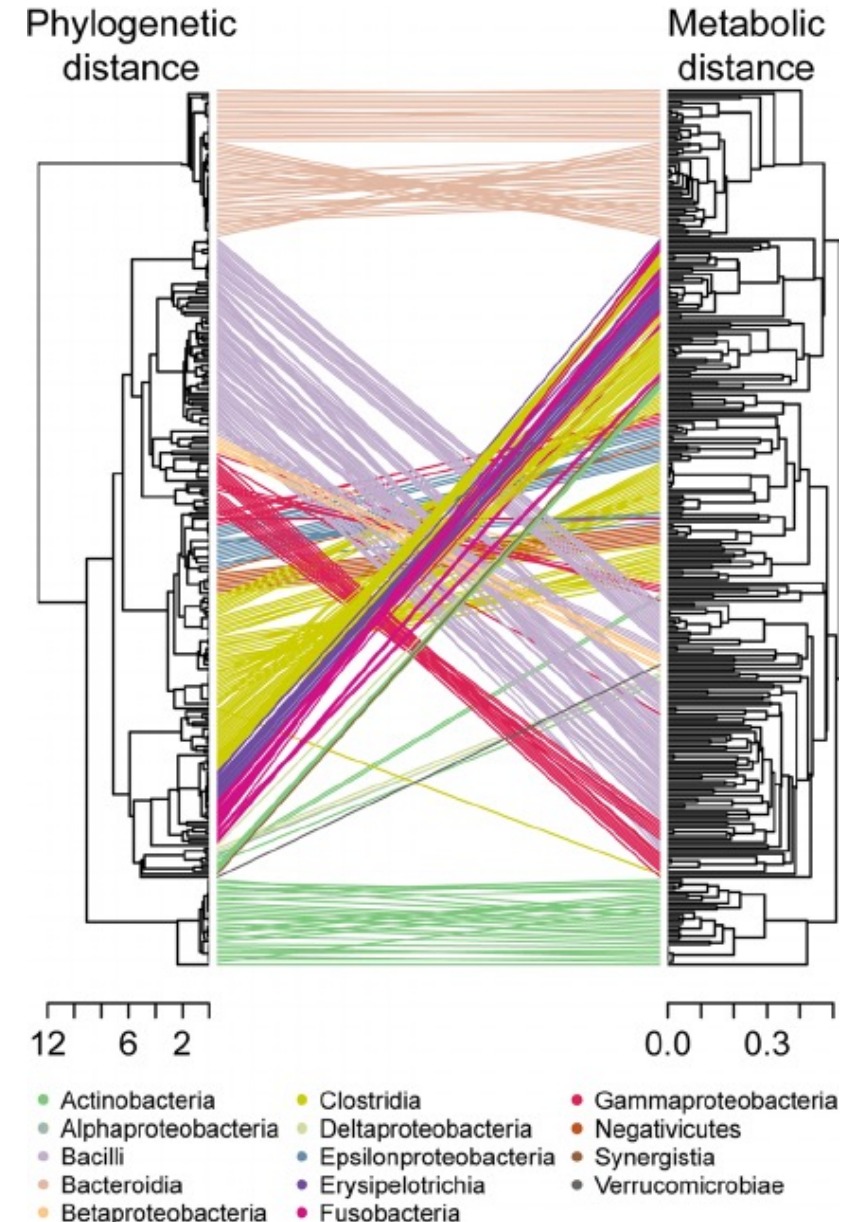


Image from Bauer et al. 2015

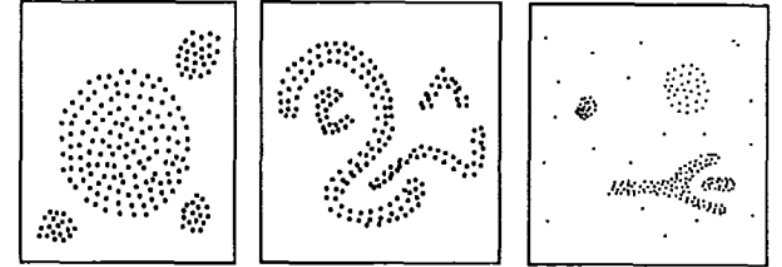
# DBSCAN

## Density-Based Spatial Clustering of Applications with Noise

- Intuitive idea of "clusters" and "noise"
- The main reason we can identify clusters is...
  1. The density within a "cluster" is considerably higher than the density outside of a cluster
  2. The density within a "noise" area is considerably lower than the density in any cluster
- The inventors aimed to formalize this notion of "clusters" and "noise" by creating DBSCAN

### 3. A Density Based Notion of Clusters

When looking at the sample sets of points depicted in figure 1, we can easily and unambiguously detect clusters of points and noise points not belonging to any of those clusters.



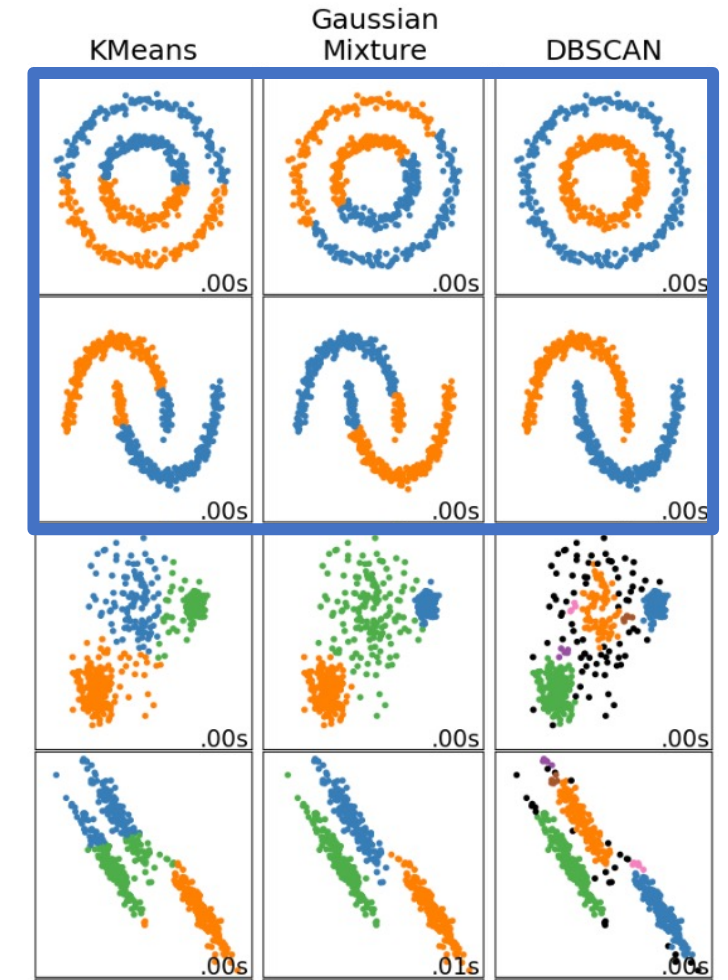
# DBSCAN Motivation

## K-means and Mixture Model Limitations

- Sensitive to data structure, esp. if clusters have arbitrary shapes
- Problems with outliers (e.g. detection, clustering)

## DBSCAN

- Discovery of clusters with arbitrary shape
  - e.g. Spherical, Drawn-out, Linear, Elongated, etc.
- Resistant to outliers



Resource: [Scikit-learn](https://scikit-learn.org/)

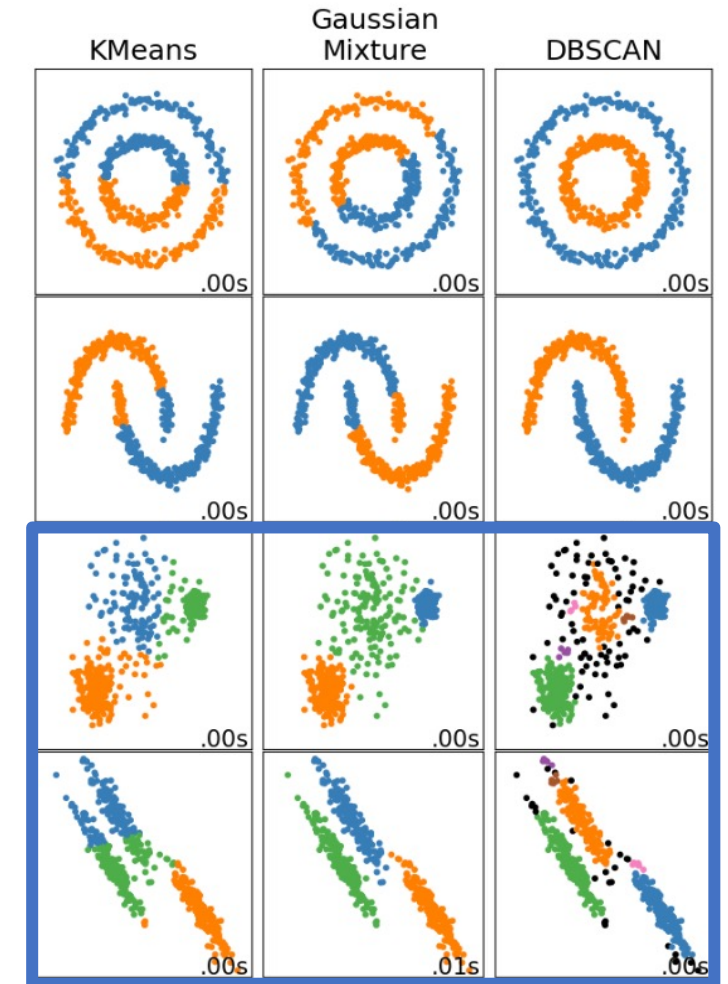
# DBSCAN Motivation

## K-means and Mixture Model Limitations

- Sensitive to data structure, esp. if clusters have arbitrary shapes
- **Problems with outliers (e.g. detection, clustering)**

## DBSCAN

- Discovery of clusters with arbitrary shape
  - e.g. Spherical, Drawn-out, Linear, Elongated, etc.
- **Resistant to outliers**



Resource: [Scikit-learn](https://scikit-learn.org/)

# Defining Density

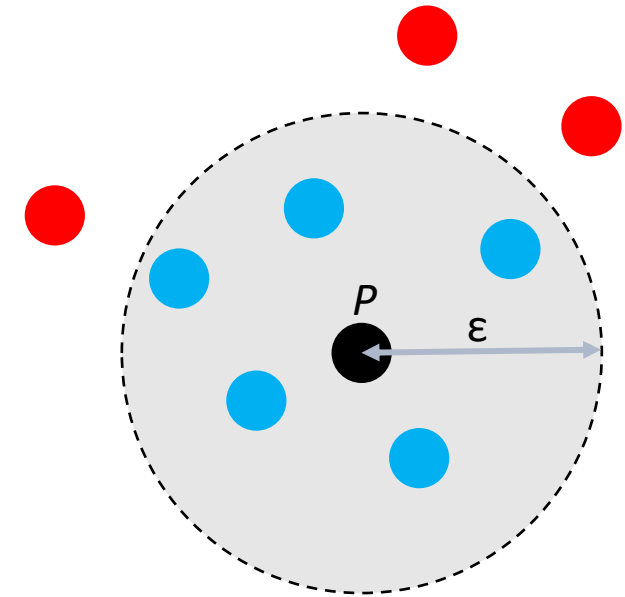
## DBSCAN Parameters

Eps ( $\epsilon$ )

MinPts

### $\epsilon$ -Neighborhood of a Point $P$

- The  $\epsilon$ -Neighborhood of  $P$  describes the space within distance  $\epsilon$  of  $P$
- An example in 2D-space
  - The  $\epsilon$ -neighborhood of  $P$  would be the space within a circle of radius  $\epsilon$  around  $P$



$\epsilon$ -Neighborhood of  $P$

# Defining Density

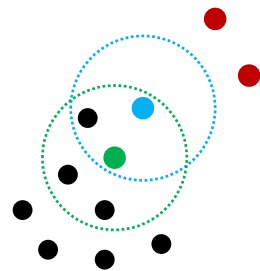
## DBSCAN Parameters

Eps ( $\epsilon$ )

MinPts

## Defining Point Types with *MinPts* & $\epsilon$ -Neighborhood

Point Type	Assigned to Cluster	Contains $\geq \text{MinPts}$ in $\epsilon$ -Neighborhood
Core Point	Yes	Yes
Border Point	Yes	No
Noise (Outlier)	No	No



$\text{MinPts} = 4$

- Assigned to Cluster
- Outliers
  - Core Point:  $\geq \text{MinPts}$  in  $N_\epsilon$
  - Border Point:  $< \text{MinPts}$  in  $N_\epsilon$
  - Other Points In Cluster
  - Neighborhood wrt  $\epsilon$

# DBSCAN Algorithm

## Example Parameters

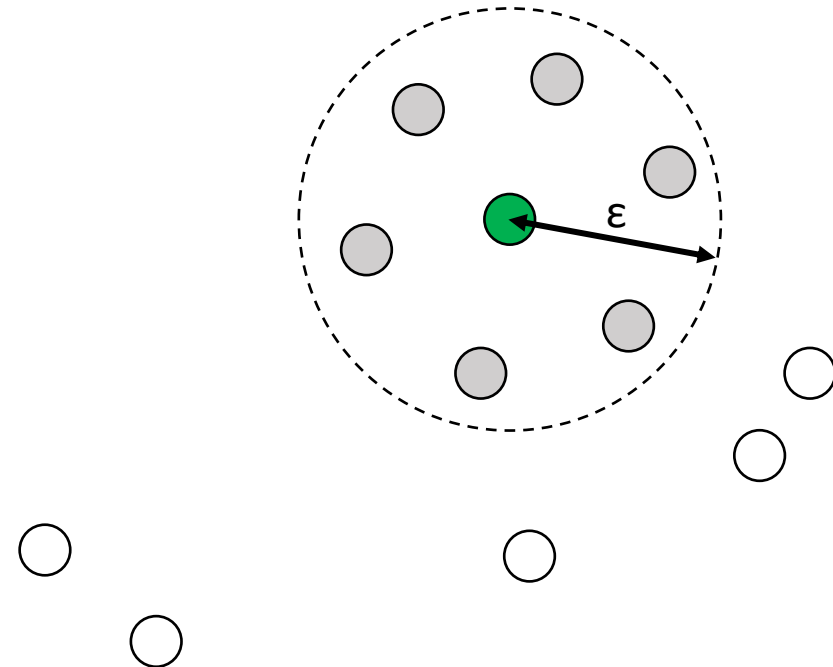
Eps ( $\epsilon$ )

MinPts: 4

Start at a random point ● and get its neighborhood information using  $\epsilon$

○ points fall within the  $\epsilon$ -neighborhood of ●

Since  $\geq \text{MinPts}$  are contained within the  $\epsilon$ -neighborhood, we begin cluster formation






# DBSCAN Algorithm

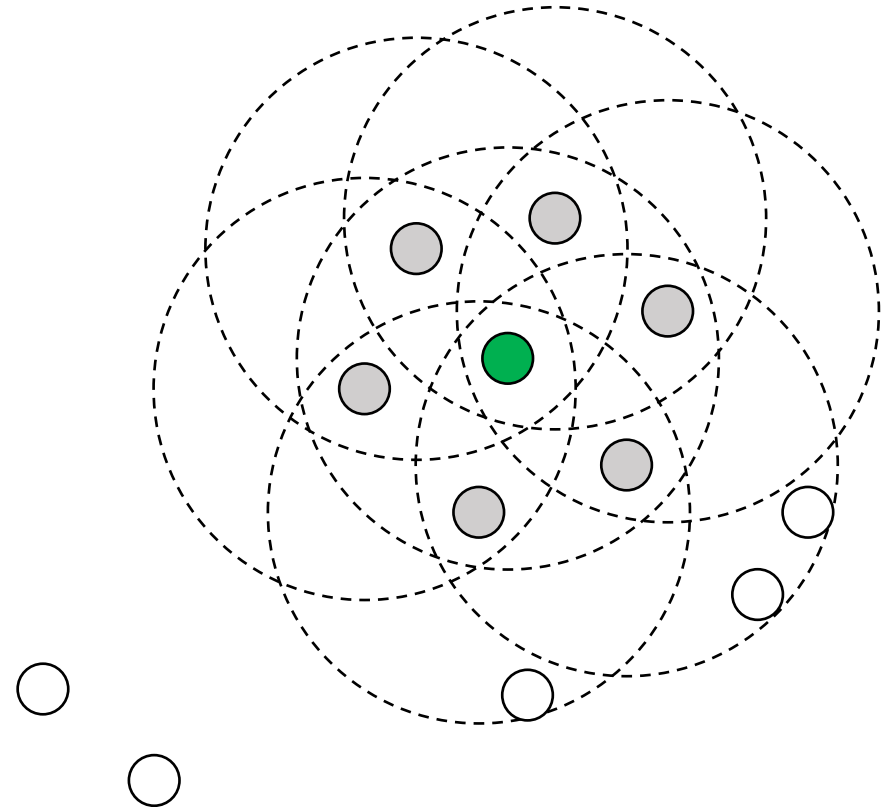
## Example Parameters

Eps ( $\epsilon$ )

MinPts: 4

Each point within the cluster then extends it's own  $\epsilon$ -neighborhood

In this example, all  points are determined to be **core points** since each point contains  $\geq \text{MinPts}$  in it's respective  $\epsilon$ -neighborhood





# DBSCAN Algorithm

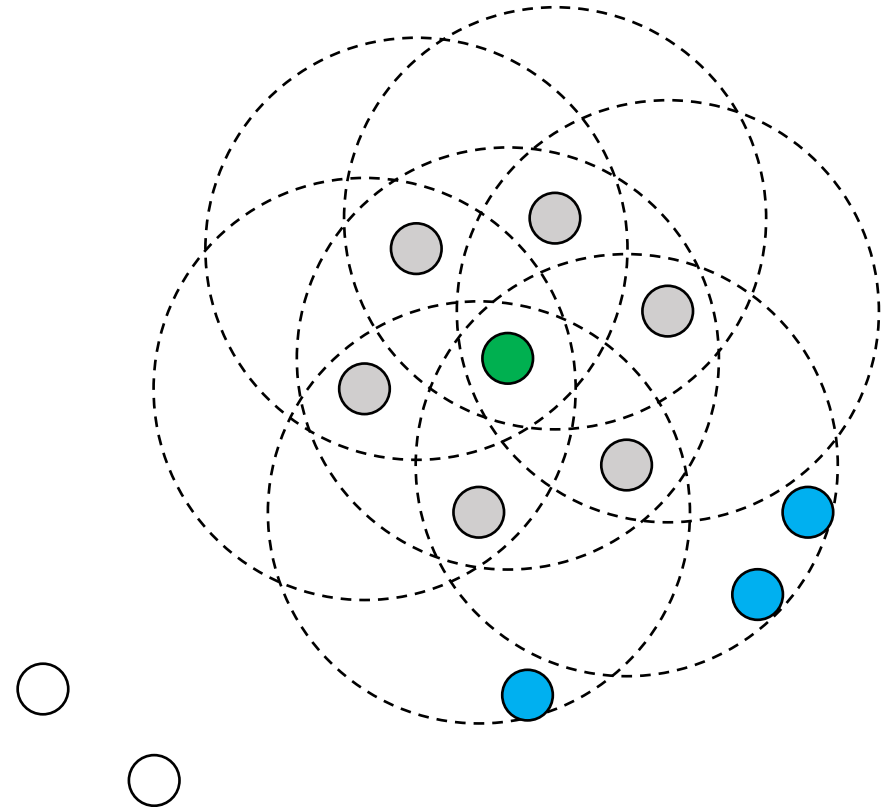
## Example Parameters

Eps ( $\epsilon$ )

MinPts: 4

Since all  points are **core points**, all new points found within their  $\epsilon$ -neighborhoods are added to the cluster

In this example,  are added to the forming cluster



# DBSCAN Algorithm

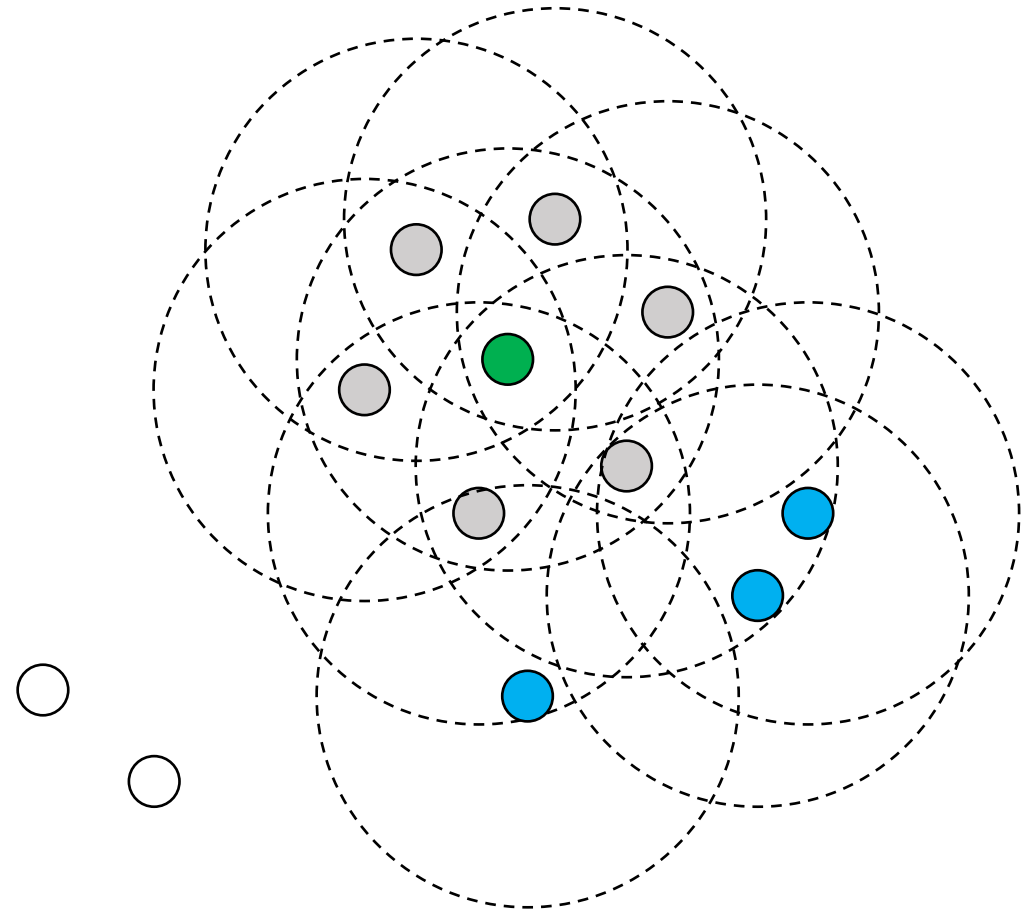
## Example Parameters

Eps ( $\epsilon$ )

MinPts: 4

Perimeters are expanded for each new member (●)

All of the ● points are found to be **border points**, since  $< MinPts$  are found within their  $\epsilon$ -neighborhoods



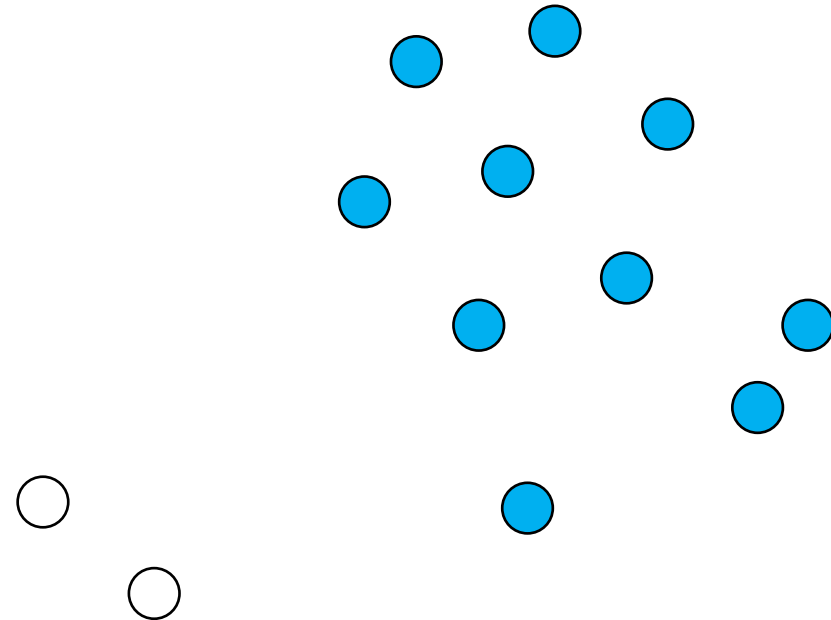
# DBSCAN Algorithm

## Example Parameters

Eps ( $\epsilon$ )

MinPts: 4

As such, the forming cluster is finalized



# DBSCAN Algorithm

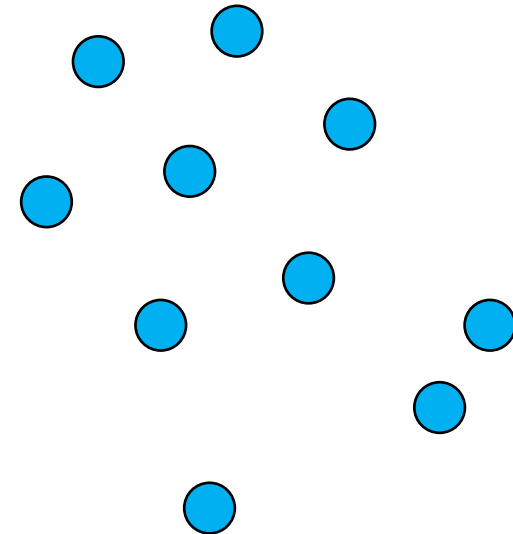
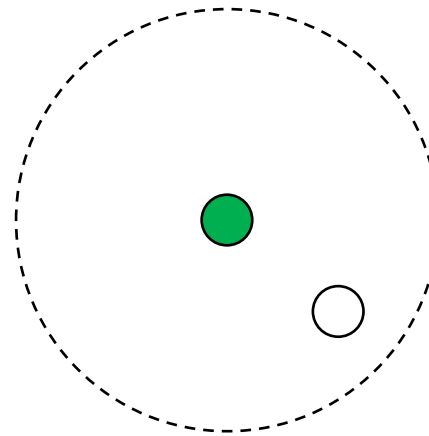
## Example Parameters

Eps ( $\epsilon$ )

MinPts: 4

The process is restarted, randomly selecting a new point (●) that has not yet been visited

This point is not determined to be a core point, so this point is labeled as noise



# DBSCAN Algorithm

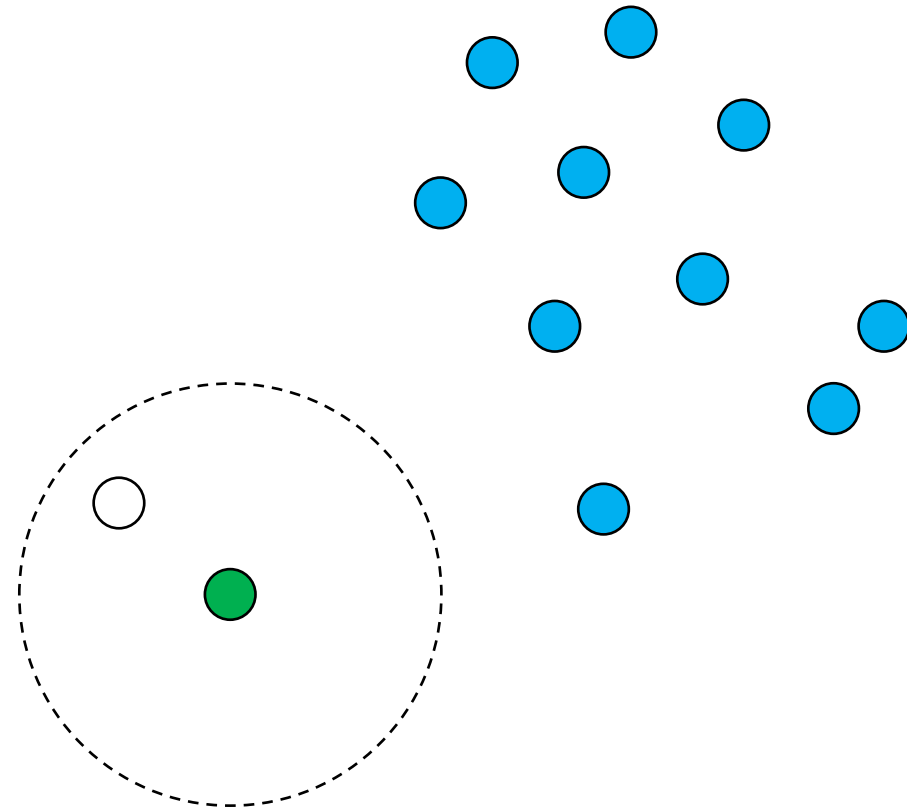
## Example Parameters

Eps ( $\epsilon$ )

MinPts: 4

**AGAIN**, the process is restarted, randomly selecting a new point (●) that has not yet been visited

This point is also not determined to be a core point, so this point is labeled as noise



# DBSCAN Algorithm

## Example Parameters

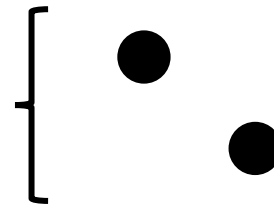
Eps ( $\epsilon$ )

MinPts: 4

At this point, all points have been visited and the process ends

All points that were not determined to be a part of a cluster are officially labeled as **noise** (●)

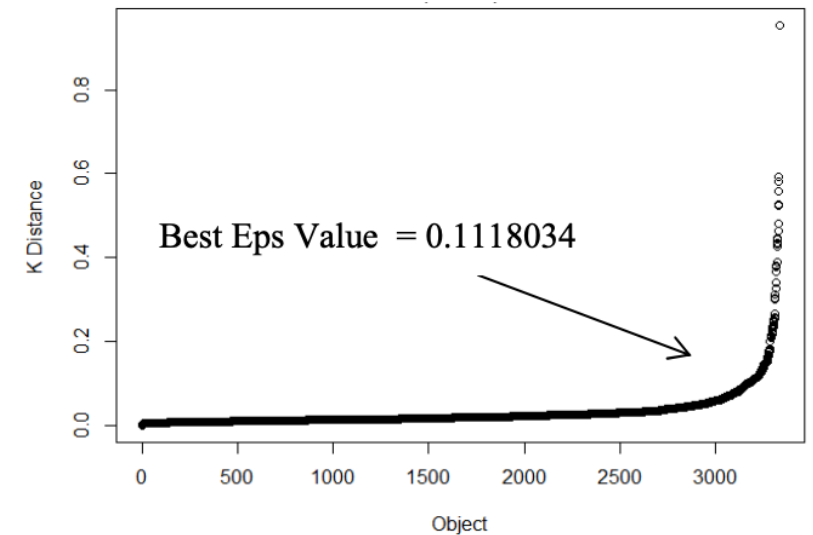
"Noise" points that are  
not assigned to a cluster



# DBSCAN $\epsilon$ Selection

**K-Distance Plot Approach:** An elbow point detection method for finding an *optimal*  $\epsilon$

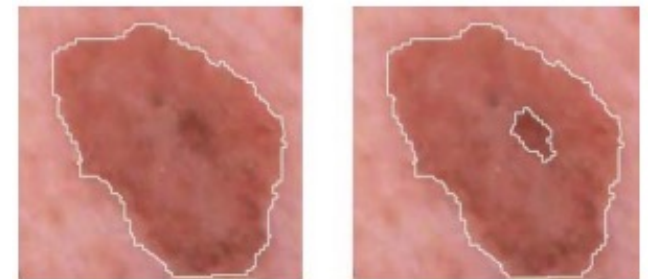
1. Calculate the distance of each point to the nearest  $k$  neighbors (using k-NN), where  $k = \text{MinPts}$
2. Sort distances in ascending order and plot
3. Optimal  $\epsilon$  corresponds to maximum point of curvature (elbow)





# DBSCAN: Real-world example

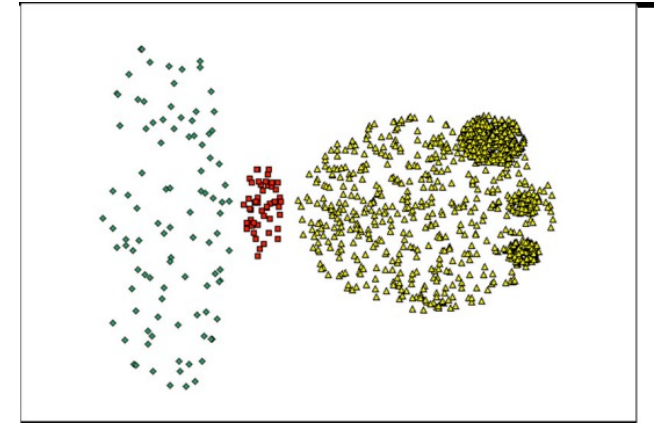
- DBSCAN for color image segmentation
- Authors used DBSCAN for the "identification of significant color regions in images of skin lesions"
- Note that the DBSCAN is used within their pipeline in conjunction with other steps that allow for segmentation (i.e. it is not as simple as running DBSCAN on unprocessed image data)
- M. E. Celebi, Y. A. Aslandogan and P. R. Bergstresser, "Mining biomedical images with density-based clustering," International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, 2005, pp. 163-168 Vol. 1, [doi: 10.1109/ITCC.2005.196](https://doi.org/10.1109/ITCC.2005.196).



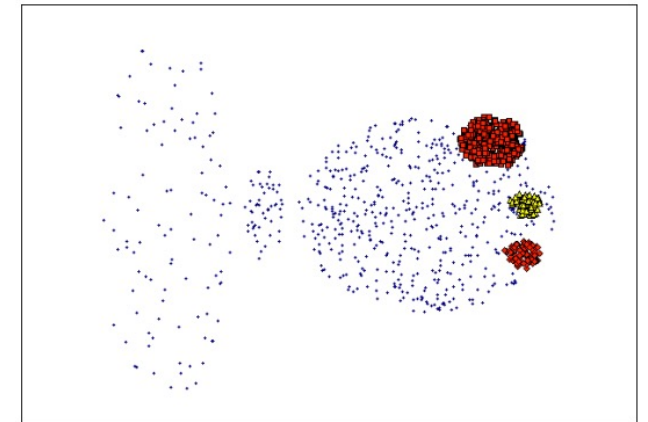
**Figure 4. a) After the 1st iter. b) Final segmentation result**

# DBSCAN Limitations

- DBSCAN performs poorly when there are large differences in densities
  - Sometimes the Eps-MinPts combinations cannot be chosen appropriately for all clusters
- *Enter OPTICS...*



(MinPts=4, Eps=9.75).

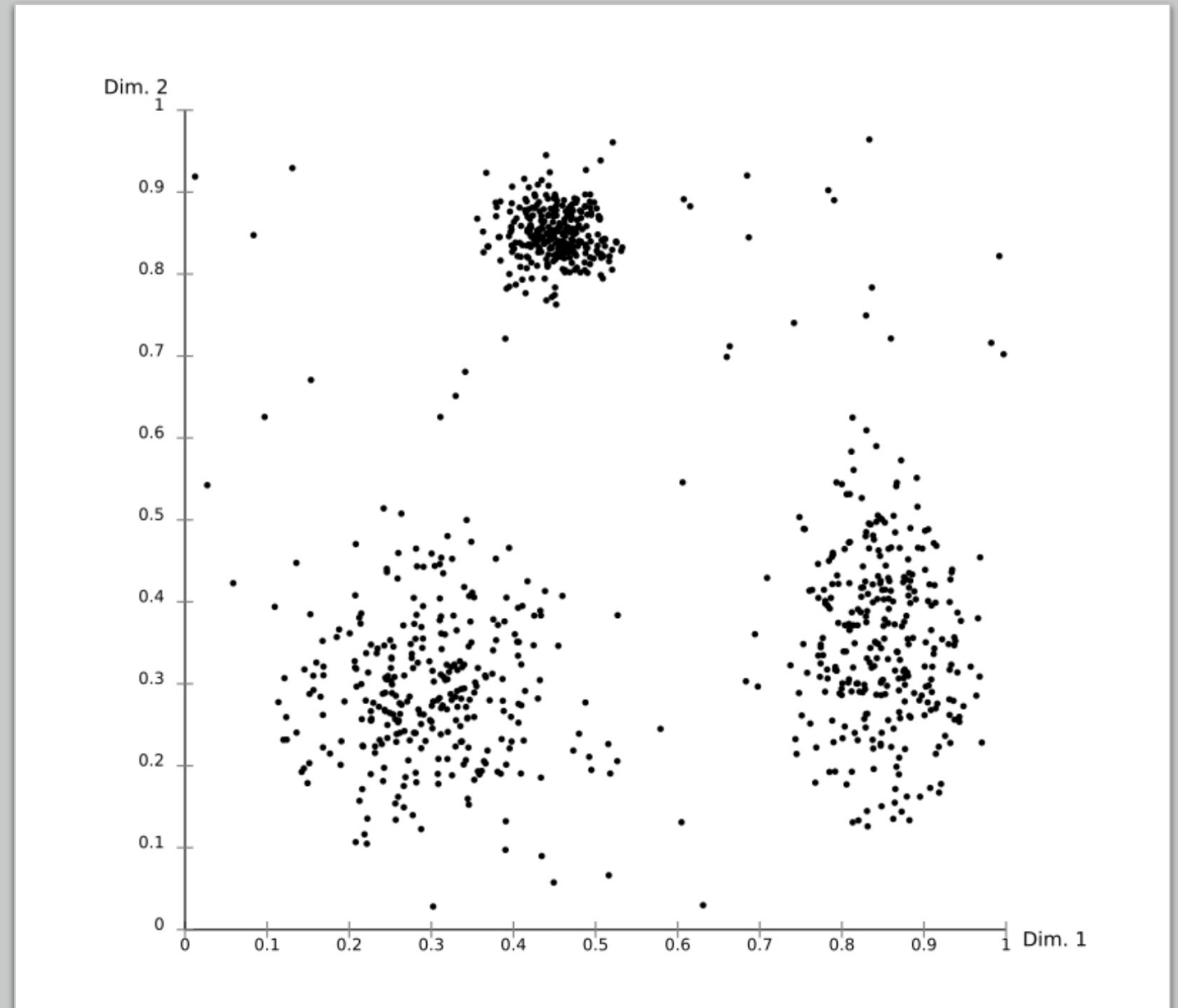


(MinPts=4, Eps=9.92)

# OPTICS

# OPTICS: motivation

- But what if clusters have *variable* densities? Or clusters within clusters?
- DBSCAN may separate sparse clusters into more clusters than we want, or too few (i.e. outcome is sensitive to  $\epsilon$ ).



# OPTICS: background

- **Ordering Points To Identify the Clustering Structure**
- Invented by the same authors of DBSCAN; outlined in a 1999 paper: [10.1.1.129.6542](#)
  - Mihael Ankerst; Markus M. Breunig; Hans-Peter Kriegel; Jörg Sander (1999). OPTICS: Ordering Points To Identify the Clustering Structure. ACM SIGMOD International Conference on Management of Data. ACM Press: 49–60.

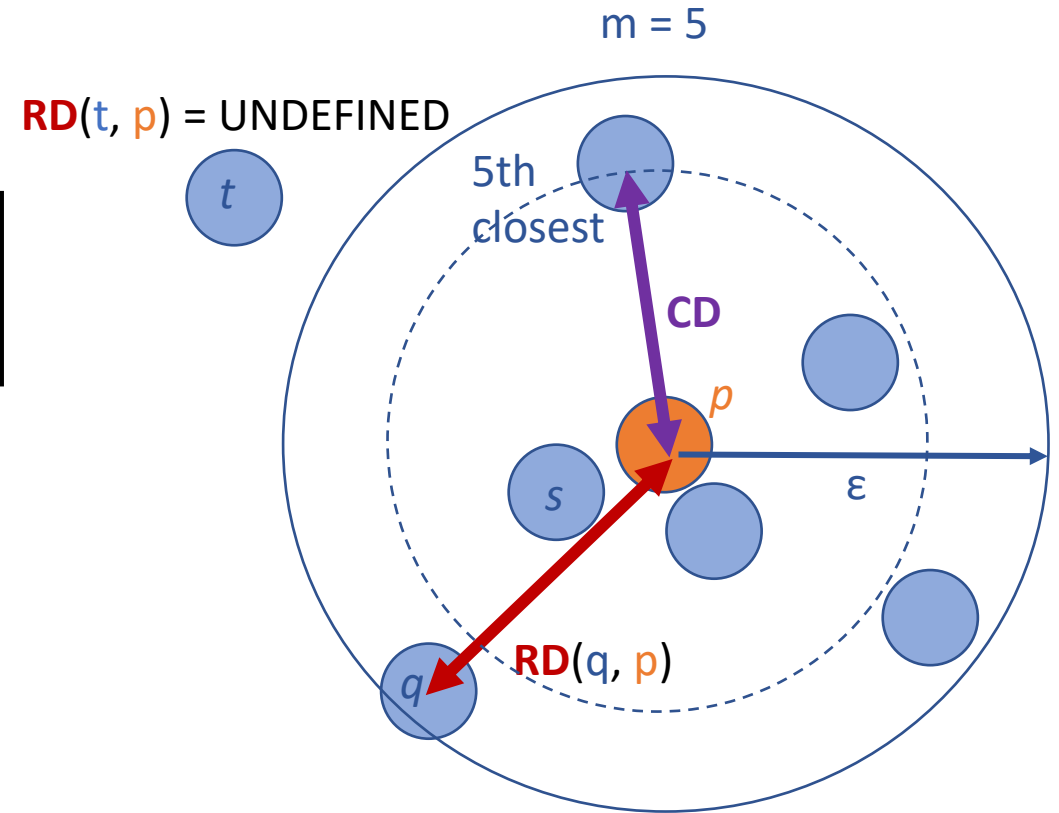
# OPTICS: Core Distance and Reachability Distance

The **core distance** (CD) of a core point  $p$ :

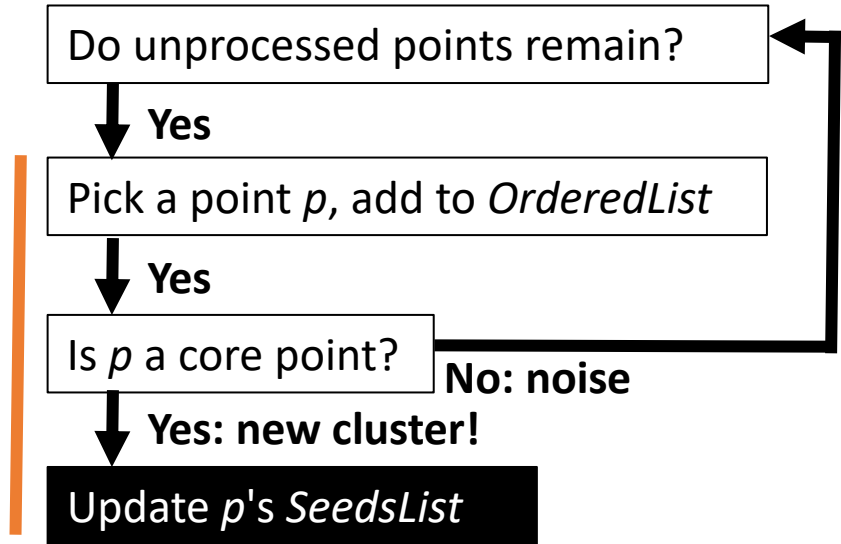
= the  $m$ th-smallest distance within a core point's  $\epsilon$ -radius neighborhood

The **reachability distance** (RD) between  $p$  and any point  $q$ :

$$= \max[ \text{dist}(p, q), \text{CD}(p) ]$$

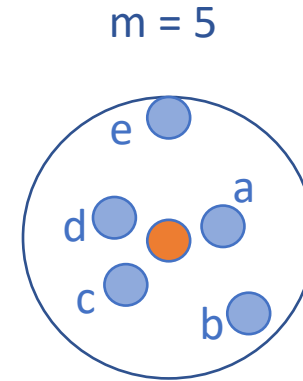


# OPTICS: algorithm



## Update *SeedsList* step

Order the unprocessed points in this point's neighborhood by **RD** in its *SeedsList*

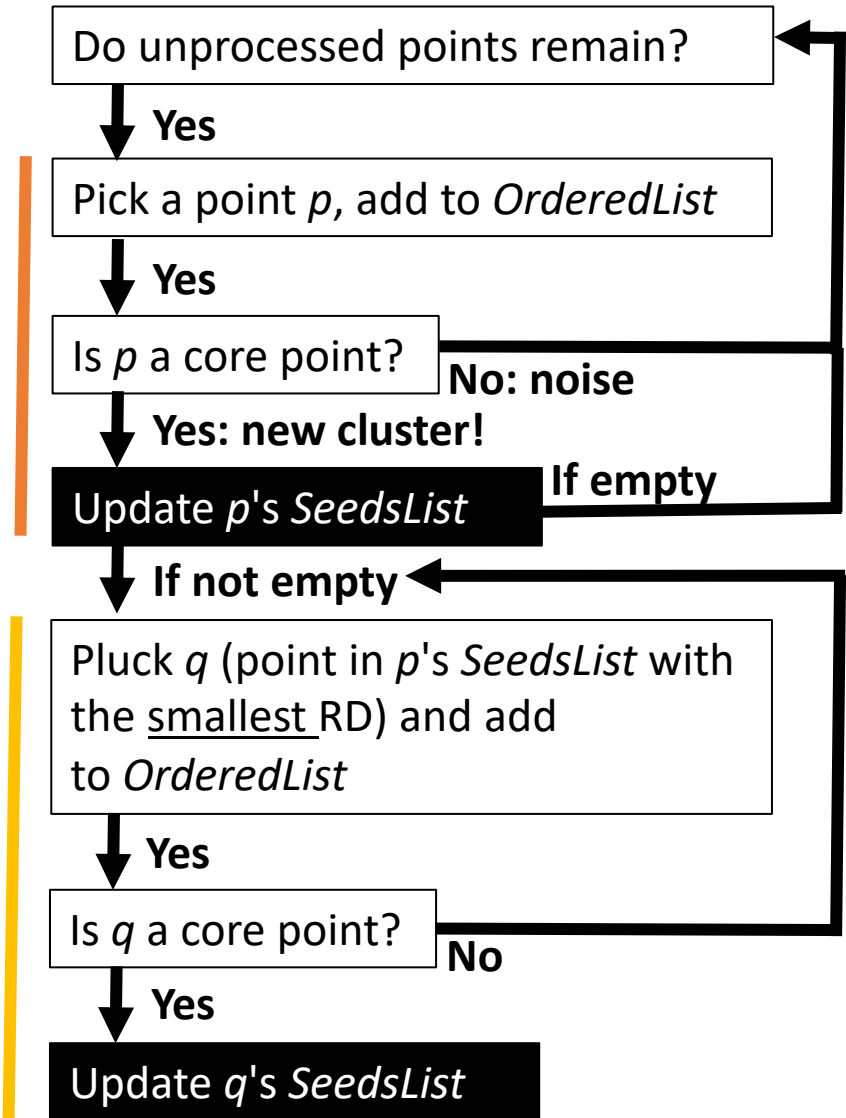


***OrderedList:***  
1.  $p$

## ***p: SeedsList***

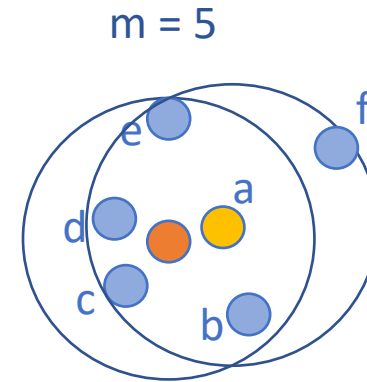
1. a, b, c, d (tie; RD = CD)
2. e

# OPTICS: algorithm



## Update *SeedsList* step

Order the unprocessed points in this point's neighborhood by **RD** in its *SeedsList*



**OrderedList:**

1.  $p$
2.  $q$

**$p$ : *SeedsList***

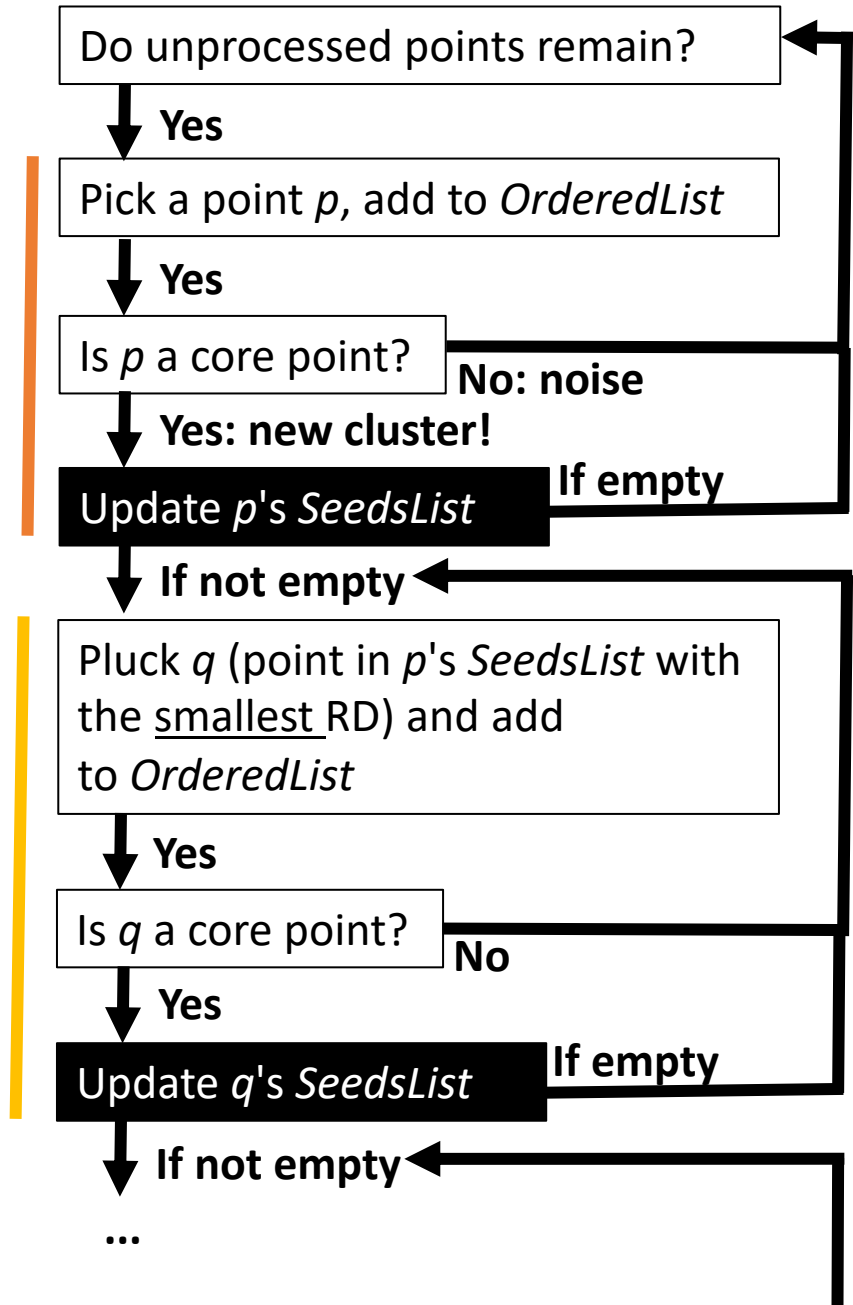
1.  $a, b, c, d$  (tie;  $RD = CD$ )
2.  $e$

**$q(=a)$ : *SeedsList***

1.  $f$

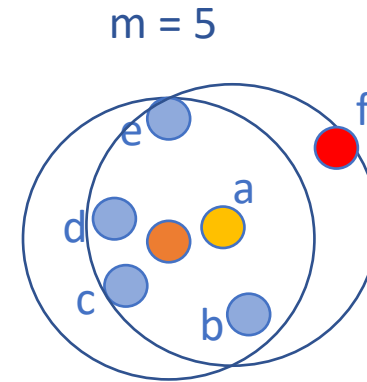


# OPTICS: algorithm



## Update *SeedsList* step

Order the unprocessed points in this point's neighborhood by **RD** in its *SeedsList*



***OrderedList:***

1.  $p$
2.  $q$

**$p$ : *SeedsList***

1.  $a, b, c, d$  (tie;  $RD = CD$ )
2.  $e$

**$q(=a)$ : *SeedsList***

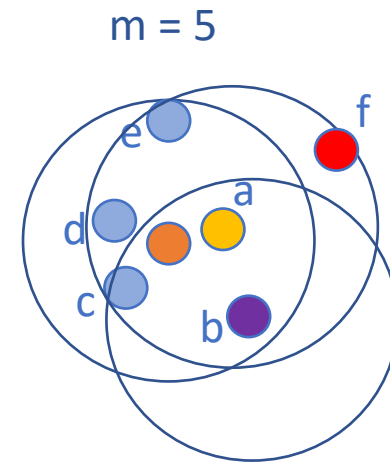
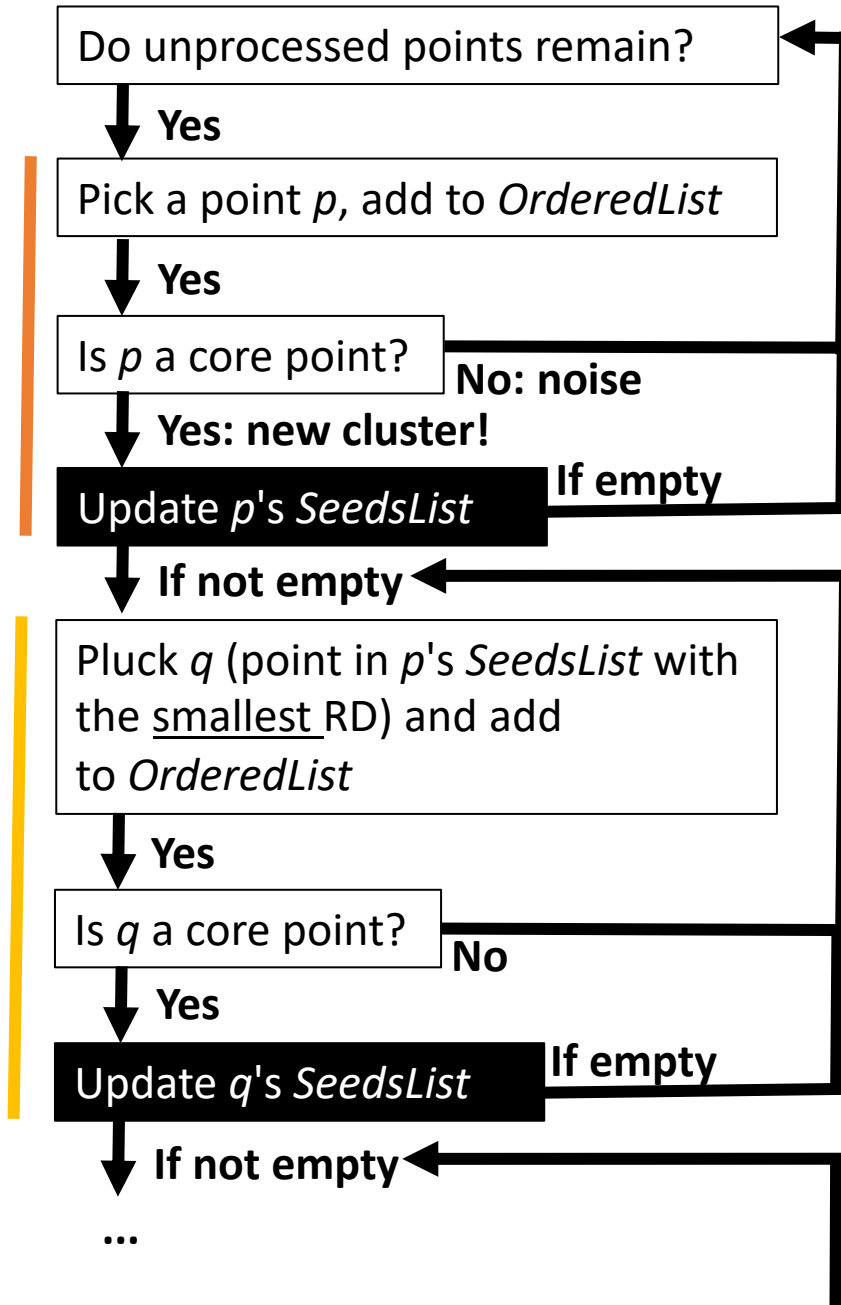
1.  $f$

-->  $f$  is not a core point

--> no more pts in  $q$ 's *Seedslist*

--> select  $b$  from  $p$ 's *Seedslist*, and iterate

# OPTICS: algorithm



***OrderedList:***

1.  $p$
2.  $q$
3. ...

## Update *SeedsList* step

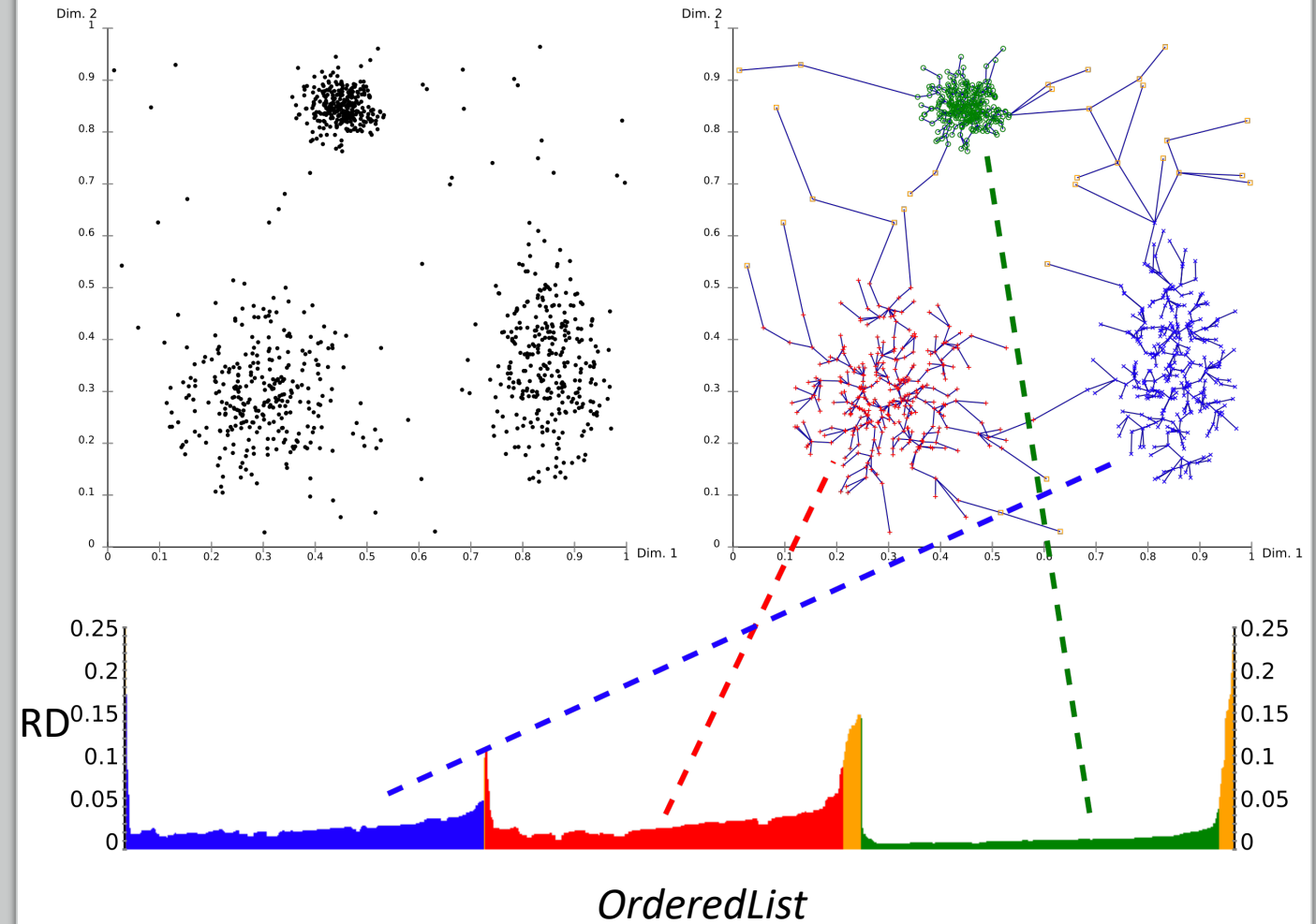
Order the unprocessed points in this point's neighborhood by **RD** in its *SeedsList*

Iteratively extends until the most extensive density-reachable point is found, then all other points in its parent's core point's *SeedsLists* are processed, then all points in its grandparent's *SeedsLists* are processed — and so on — until all points in the d-r cluster are marked with their smallest RD.

Exhausting a d-r cluster returns to the top of the algorithm to process a new cluster (if there is another one).

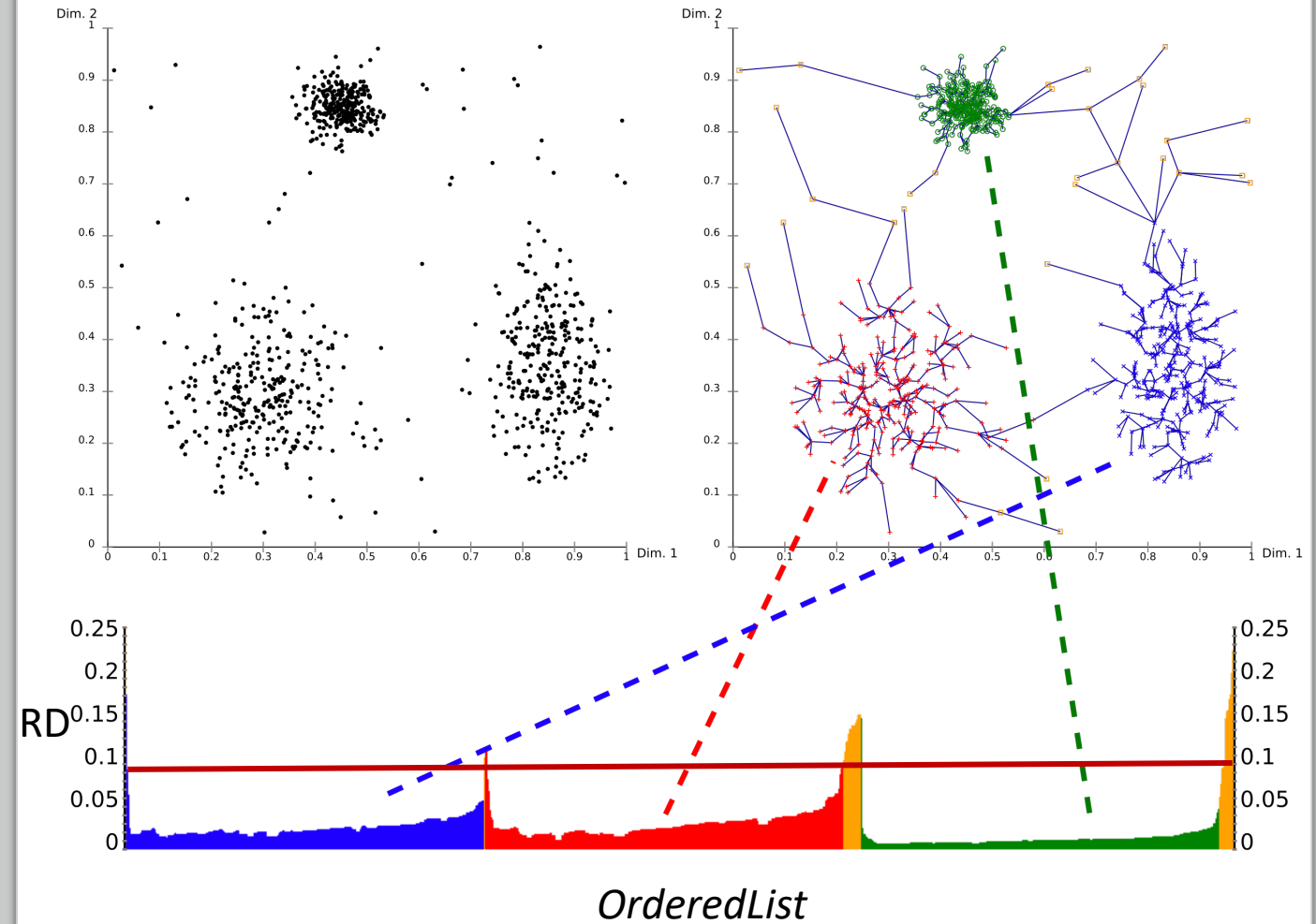
# OPTICS: output

- Sorts data points into a reachability plot, with every point ordered into hierarchical clusters on the x-axis, and reachability distance to the nearest neighbor on the y-axis
- Can extract ANY density-based clustering wrt  $\epsilon$  and  $m$  or denser!



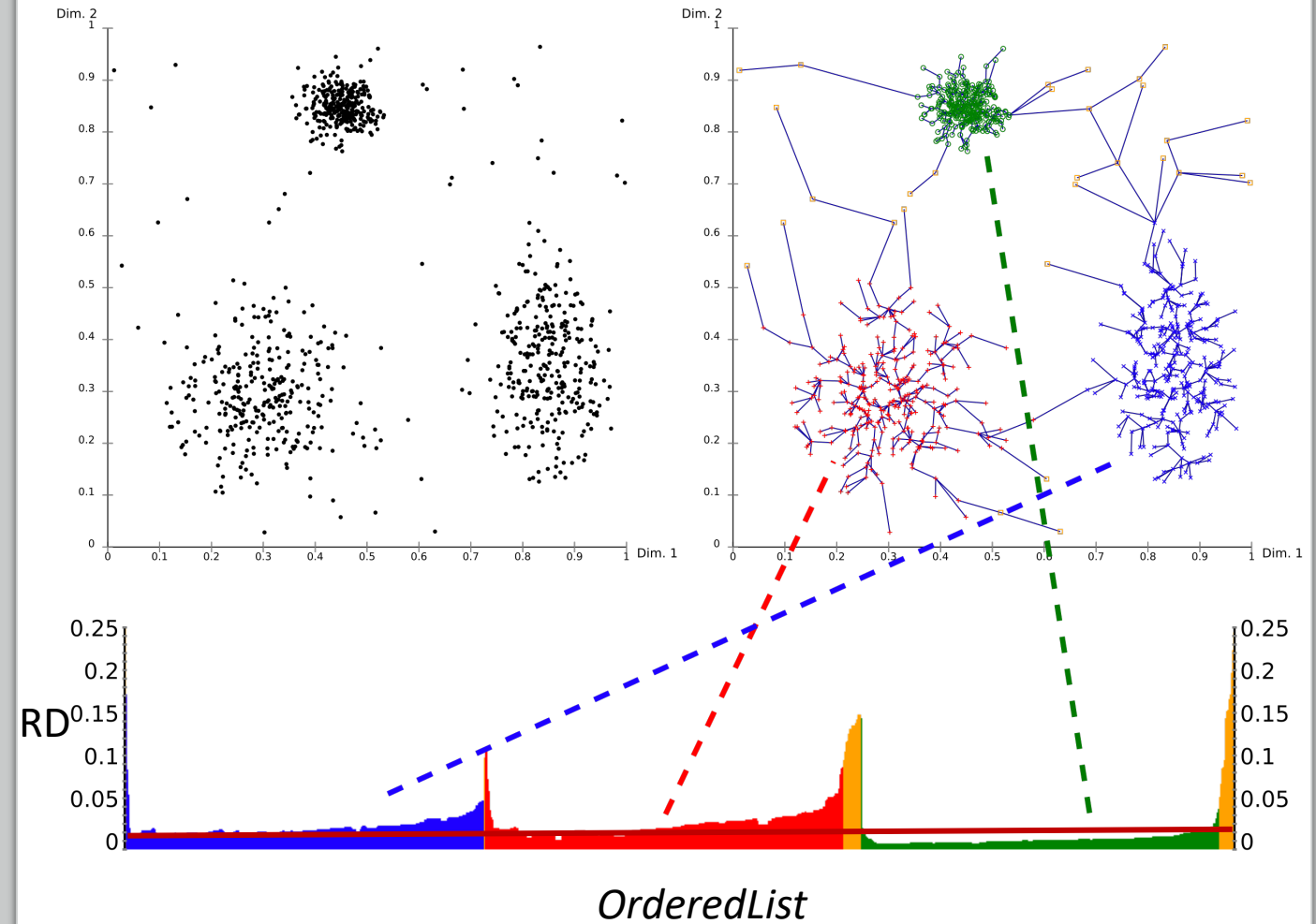
# OPTICS: output

- Sorts data points into a reachability plot, with every point ordered into hierarchical clusters on the x-axis, and reachability distance to the nearest neighbor on the y-axis
- Can extract ANY density-based clustering wrt  $\epsilon$  and  $m$  or denser!
- Exact boundaries of clusters can be set manually, by choosing an RD cutoff, or by algorithms to detect peaks vs. valleys



# OPTICS: output

- Sorts data points into a reachability plot, with every point ordered into hierarchical clusters on the x-axis, and reachability distance to the nearest neighbor on the y-axis
- Can extract ANY density-based clustering wrt  $\epsilon$  and  $m$ !
- Exact boundaries of clusters can be set manually, by choosing an RD cutoff, or by algorithms to detect peaks vs. valleys



# OPTICS vs. DBSCAN: Computational implications

- Like DBSCAN, each point is processed once.
- If  $\epsilon$  is too small, there's no tractable density cutoff to get CDs and RDs; but if we made  $\epsilon$  large enough to ensure this never happens, every neighborhood contains every point, and computation is quadratic, e.g.  $O(n^2)$ . Selecting a reasonable  $\epsilon$  and speeds up the algorithm.
- The original paper reports a constant 1.6x runtime compared to DBSCAN *given spatially indexed data querying*; without one,  $\epsilon$ -neighborhood queries for each point have to re-query the entire dataset sequentially.
  - Get  $O(n \log(n))$ .
  - If points are further arranged in a grid, get  $O(n)$ .

# OPTICS: real-world example

- OPTICS was tested for its utility to identify distinct clusters of high-cost Medicare patients
  - Yan J, et al. Applying Machine Learning Algorithms to Segment High-Cost Patient Populations. J Gen Intern Med. 2019 Feb;34(2):211-217. doi: 10.1007/s11606-018-4760-8
  - Set **MinPts** at 1% of the study population (62 patients)
  - "We varied  $\epsilon$  such that we extracted all of the clustering solutions that produced between five and ten clusters. Then, we then calculated the average silhouette width for each solution and selected the solution with the highest average silhouette width."
  - "OPTICS [was best because it] allows for “noise” points (i.e., observations not assigned to any cluster), whereas connectivity- and centroid-based clustering algorithms force all observations into clusters. By forcing outlier observations into subgroups, the resultant subgroups become more heterogeneous."

Method name	Parameters	Computation	Use cases	Metric(s) used
Gaussian Mixture Modeling	Prob. of each cluster ( $\pi_k$ ) Parameters of each cluster Number of clusters (k)	$O(NKD^3)$ D = Dimensionality N = Sample Size K = Clusters	Statistical modeling Probabilistic clustering	Mahalanobis distance
K-Means	Number of clusters	$N \times p$ distance operations in each iteration	General-purpose Small number of clusters	Euclidean or Manhattan distance
Hierarchical clustering	Number of clusters, distance (inter and intracluster)	$O(n^3)$ , $O(n^2 \log n)$ depends on implementation	Lots of clusters, distance flexible	Any distance
DBSCAN	$\epsilon$ , m	$O(N^2)$	Weird shaped clusters, outliers	Pairwise distance, usually Euclidean
OPTICS	$\epsilon$ , m	1.6 x DBSCAN; $O(n \log(n))$ or $O(n)$	General-purpose High-d datasets	Any distance; calculates CD and RD; sorts cluster points by RD



# Clustering performance evaluation

- Inertia: within-cluster sum-of-squares

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

**X**: observed data point

**C**: set of cluster means

**$\mu$** : individual cluster mean

Reference: <https://scikit-learn.org/stable/modules/clustering.html>

- Silhouette coefficient (calculated for every sample)

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

**s** will be between -1 (incorrect clustering) and +1 (perfect clustering);

Note that **s**=0 suggests decision boundary points

**a**: mean distance between sample and all other points in cluster

**b**: mean distance between sample and all other points in *next nearest cluster*

Reference: <https://scikit-learn.org/stable/modules/clustering.html>

- Rand index (next slide)

# Rand Index

$$R = \frac{a + b}{a + b + c + d}$$

- Measures agreement between two clustering methods applied to the same data
- Intuitively similar to accuracy:
  - Where agreements =  $a + b$ ; disagreements =  $c + d$
  - Bounded between  $[0, 1]$
- Pros
  - Interpretable
  - Can compare to 'ground truth'
- Cons
  - Ground truth rarely available
  - Accuracy measures don't capture when false positives or false negatives may be worse

		Clustering 2	
		Same assignment	Different assignment
Clustering 1	Same assignment	<b>a</b>	<b>c</b>
	Different assignment	<b>d</b>	<b>b</b>