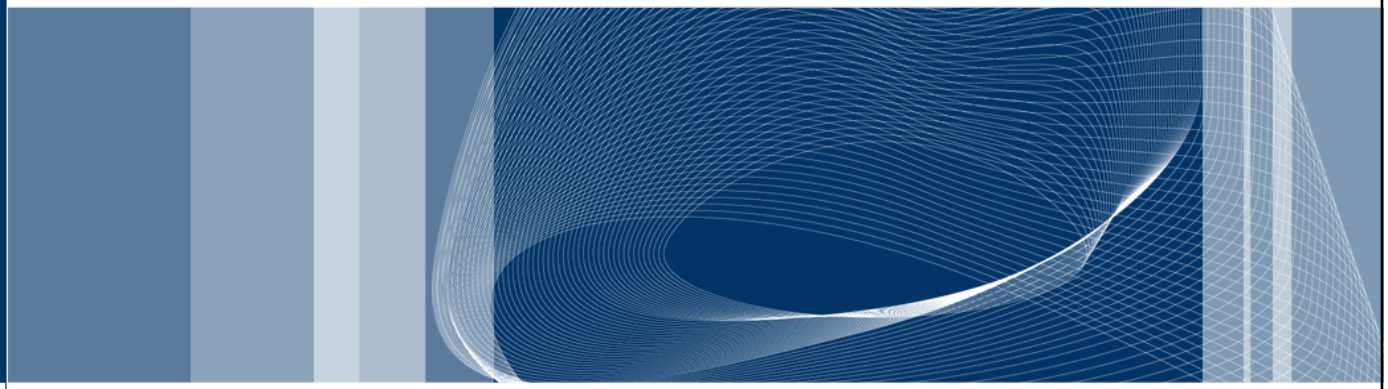
POLITECNICO DI MILANO



Representative-Based Clustering

Data Mining and Text Mining

Prof. Pierluca Lanzi

For each cluster there is a point (or a parameter vector) that summarizes it

Representative-Based Algorithms

- Given a dataset of N instances, and a desired number of clusters k, this class of algorithms generates a partition C of N in k clusters $\{C_1, C_2, \ldots, C_k\}$
- For each cluster there is a point that summarizes the cluster
- The common choice being the mean of the points in the cluster

$$\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j$$

where $n_i = |C_i|$ and $\boldsymbol{\mu}_i$ is the centroid

Brute-force Approach

Generate all the possible clustering $C = \{C_1, C_2, \ldots, C_k\}$

Then, select the best one

Unfortunately, there are $O(k^N/k!)$ possible partitions

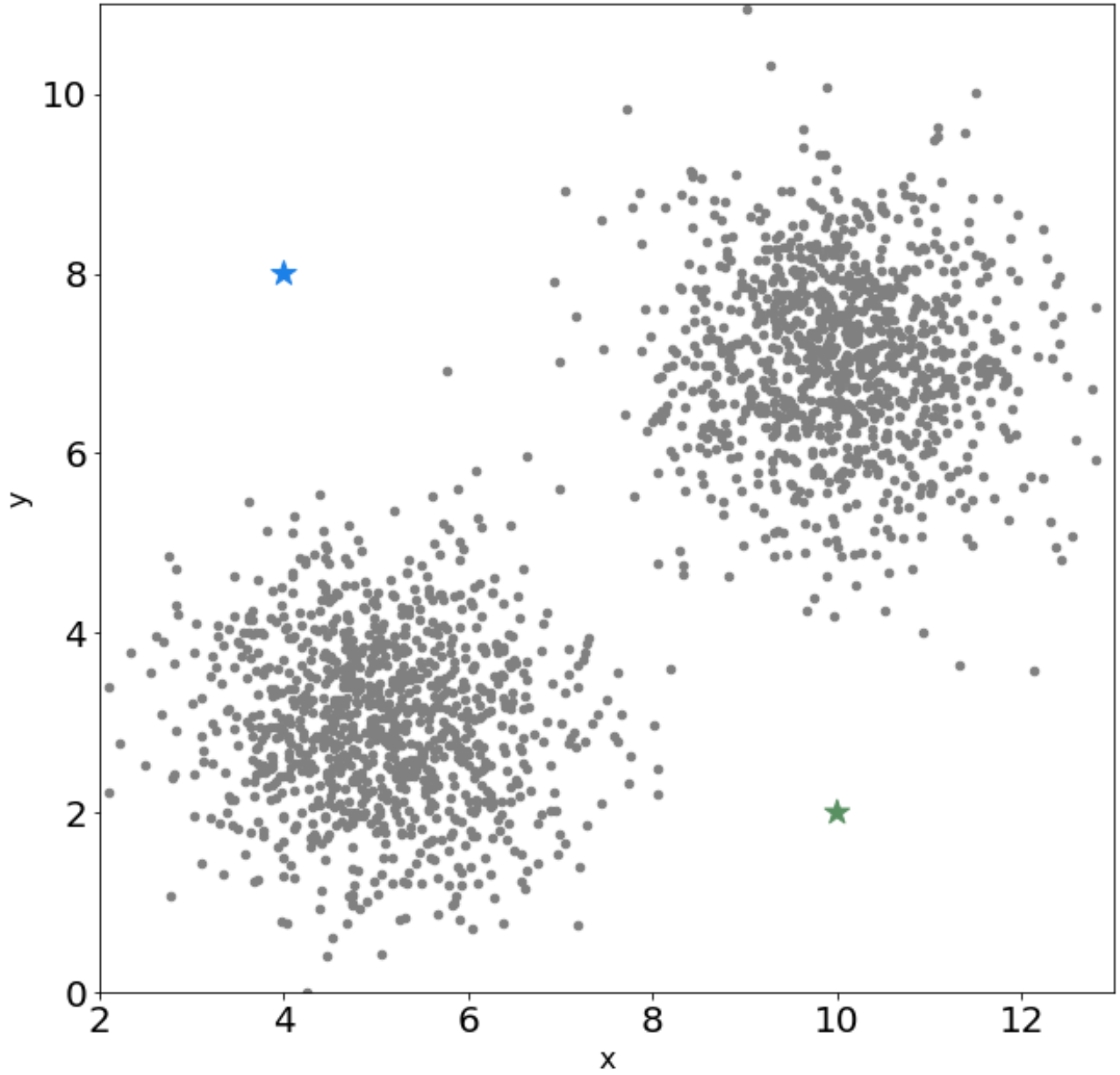
K-Means algorithm

- Greedy iterative approach to find a clustering that minimizes the SSE objective:

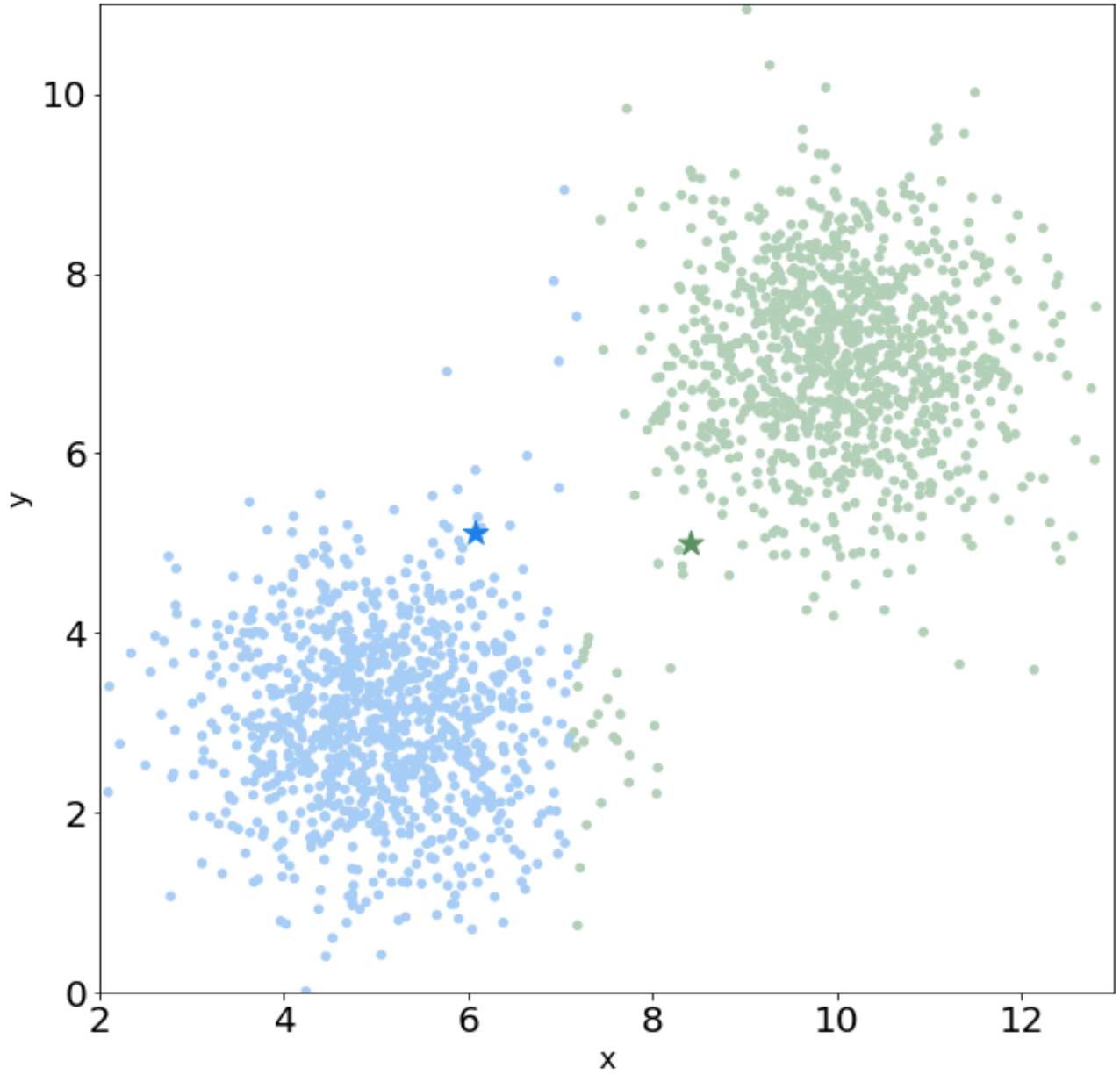
$$\text{SSE}(C) = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$

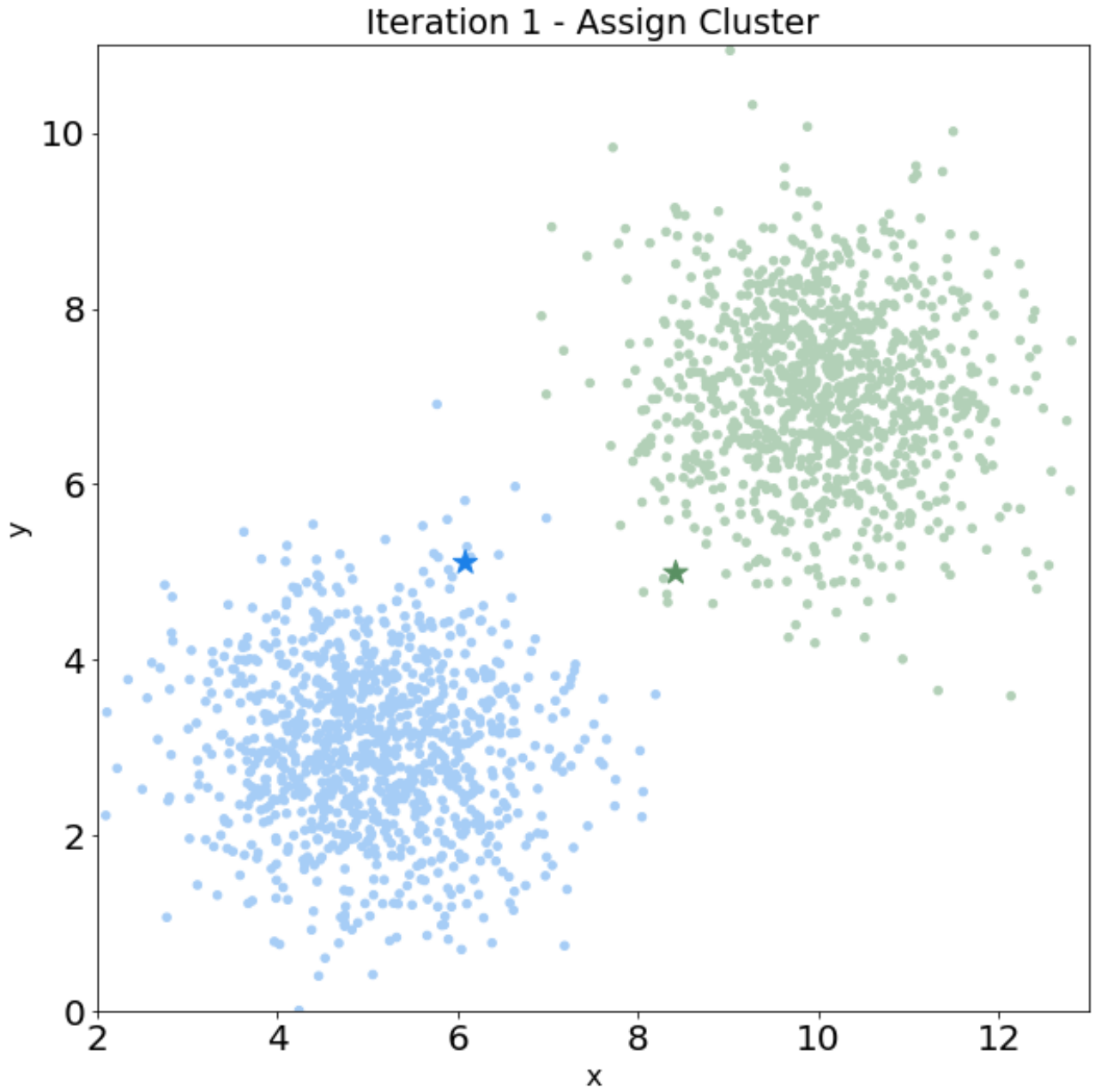
- The goal of the clustering process is thus to find

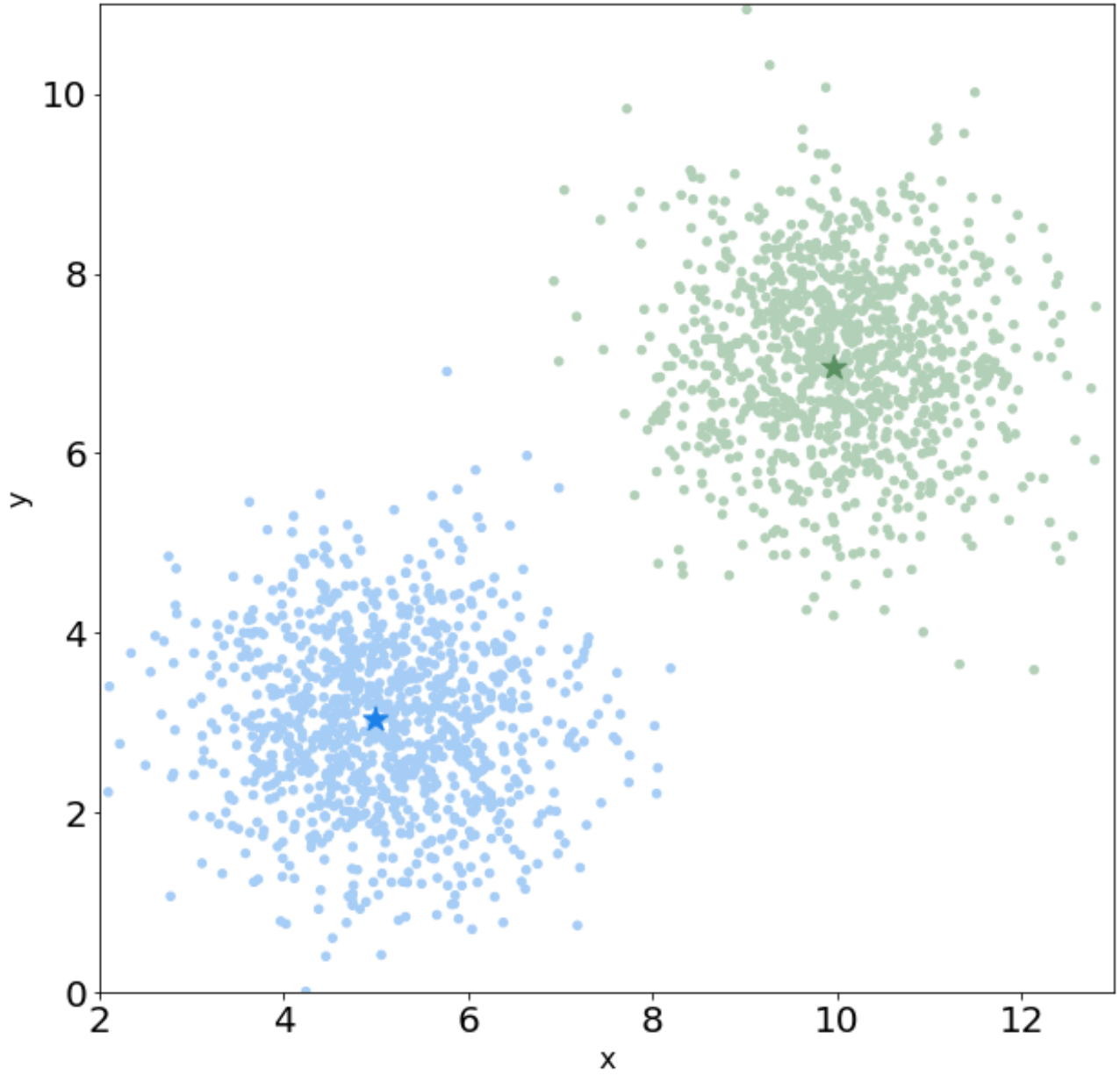
$$C^\star = \arg\min_C \text{SSE}(C)$$

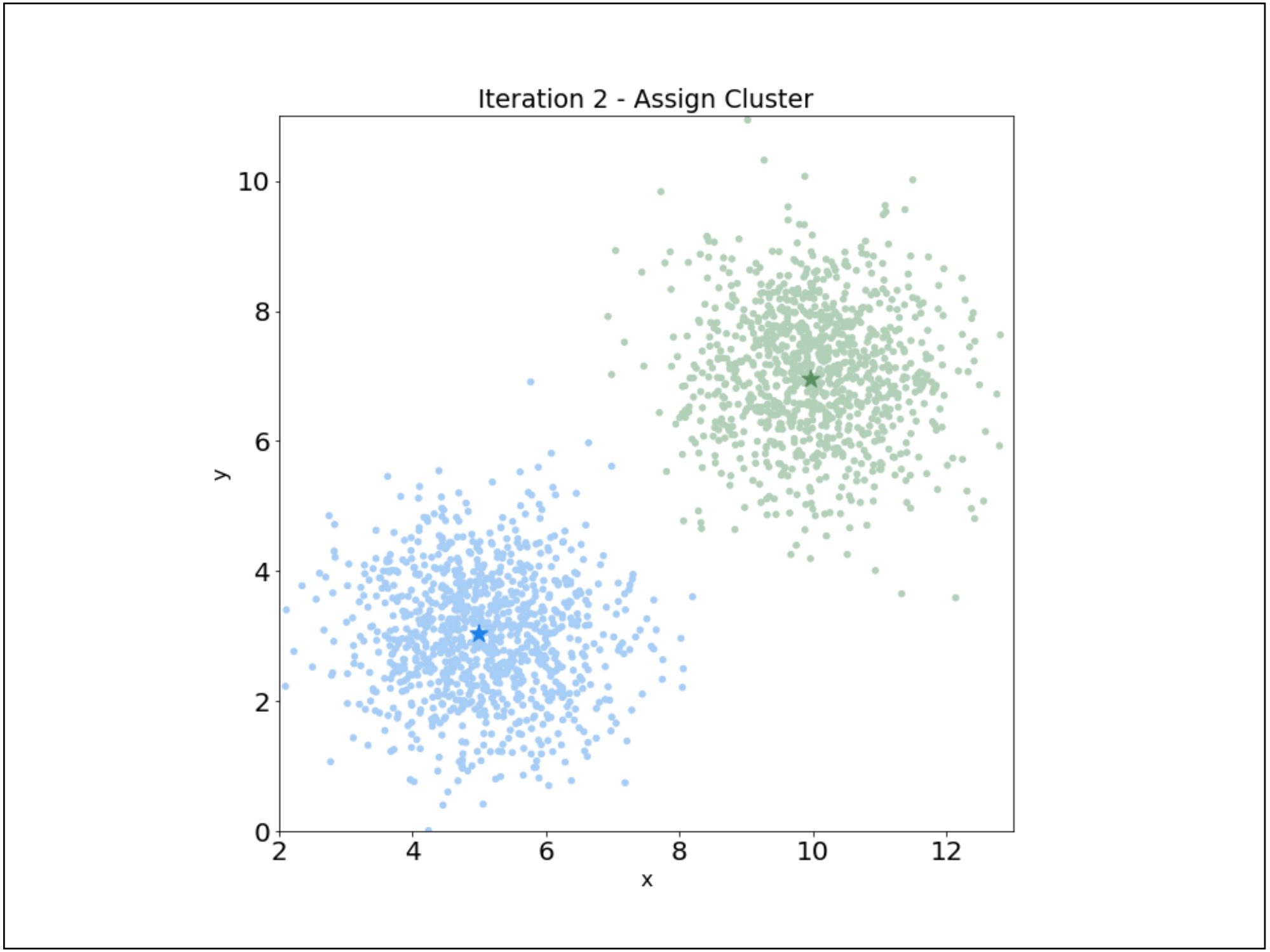

10
8
6
4
2
0
y
2
4
6
8
10
12
x

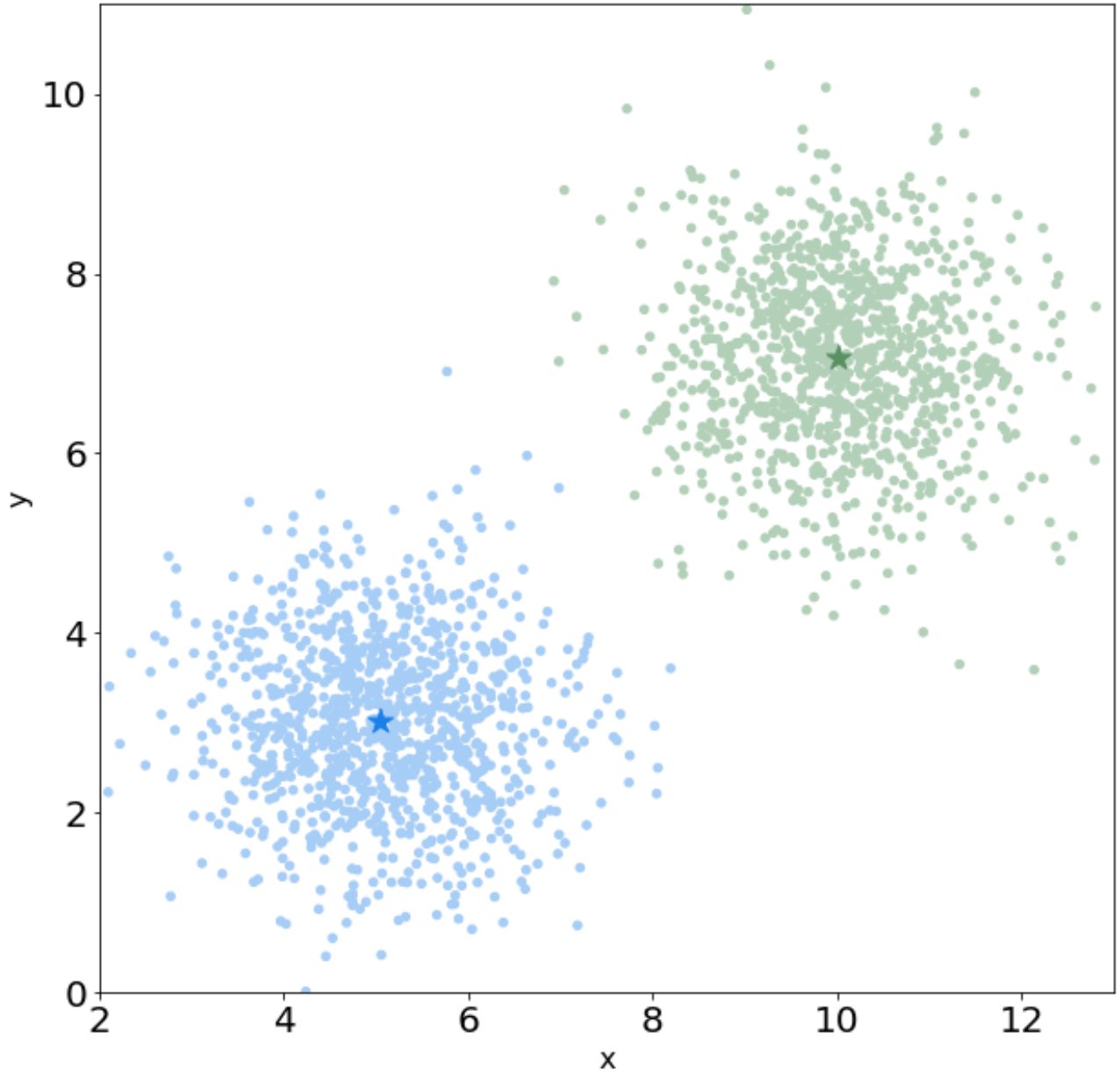

Iteration 0 - Assign Cluster
10
8
6
4
2
0
y
2
4
6
8
10
12
x


10
8
6
4
2
0
y
2
4
6
8
10
12
x


Iteration 1 - Assign Cluster
10
8
6
4
2
0
y
2
4
6
8
10
12
x


10
8
6
4
2
0
y
2
4
6
8
10
12
x


Iteration 2 - Assign Cluster
10
8
6
4
2
0
y
2
4
6
8
10
12
x


10
8
6
4
2
0
y
2
4
6
8
10
12
x

ALGORITHM 13.1. K-means Algorithm

K-MEANS $(\mathbf{D}, k, \epsilon)$:

1 $t = 0$
2 Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$
3 **repeat**
4 $\quad t \leftarrow t + 1$
5 $\quad C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$
$\quad$ `// Cluster Assignment Step`
6 $\quad$ **foreach** $\mathbf{x}_j \in \mathbf{D}$ **do**
7 $\quad\quad j^* \leftarrow \arg\min_i \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_i^{t-1} \right\|^2 \right\}$ `// Assign` $\mathbf{x}_j$ `to closest centroid`
8 $\quad\quad C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$
$\quad$ `// Centroid Update Step`
9 $\quad$ **foreach** $i = 1$ *to* k **do**
10 $\quad\quad \boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$
11 **until** $\sum_{i=1}^k \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$

"Data Mining and Analysis" by Zaki & Meira - Chapter 13
http://www.dataminingbook.info

k-Means Algorithm

- Most widely known representative-based algorithm
- Assumes an Euclidean space but sometimes it can be extended to the non-Euclidean case
- Employs a greedy iterative approaches that minimizes the SSE objective. Accordingly it can converge to a local optimal instead of a globally optimal clustering.

Computational Complexity

- Cluster assignment takes O(nkd) time since, for each of the n points, it computes its distance to each of the k clusters, which takes d operations in d dimensions
- The centroid re-computation step takes O(nd) time because it adds at total of n d-dimensional points
- Assuming that there are t iterations, the total time for K-means is given as O(tnkd).
- In terms of the I/O cost it requires O(t) full database scans, because we have to read the entire database in each iteration.

Centroid initialization

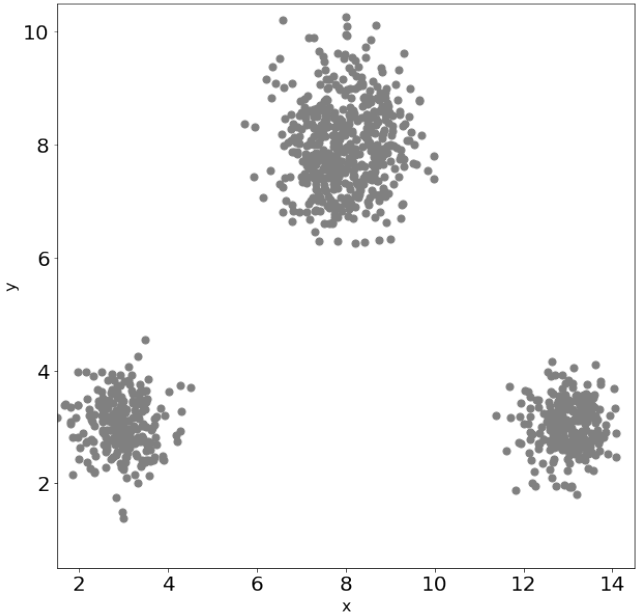
Centroid Initialization

- Solution 1
  - Pick points that are as far away from one another as possible.
- Solution 2

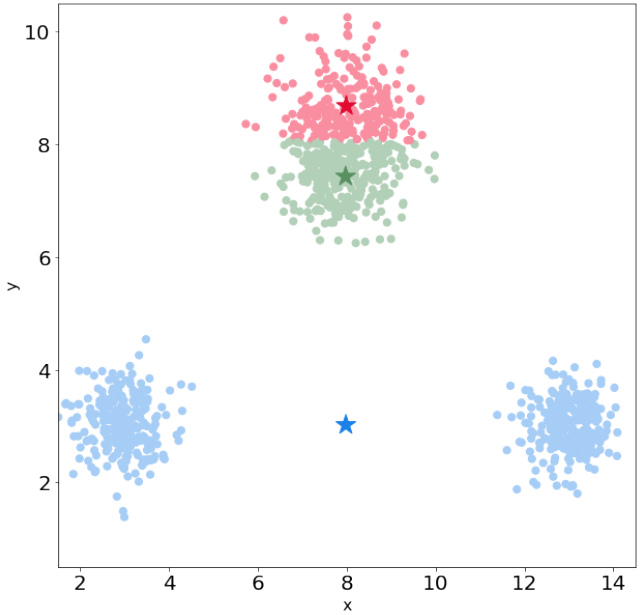
```
Pick the first point at random;
WHILE there are fewer than k points DO
    Add the point whose minimum distance
    from the selected points is as large as
    possible;
END;
```

- Solution 3
  - Cluster a sample of the data, perhaps hierarchically, so there are k clusters. Pick a point from each cluster, perhaps that point closest to the centroid of the cluster.

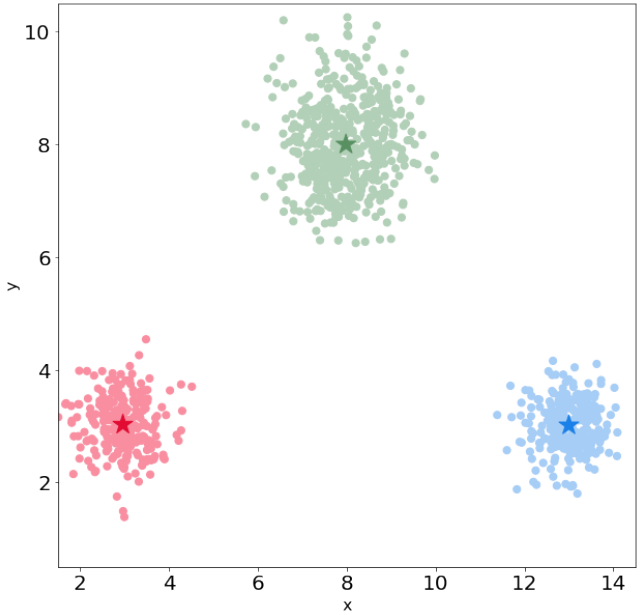
Initial centroids matters!



Original Points

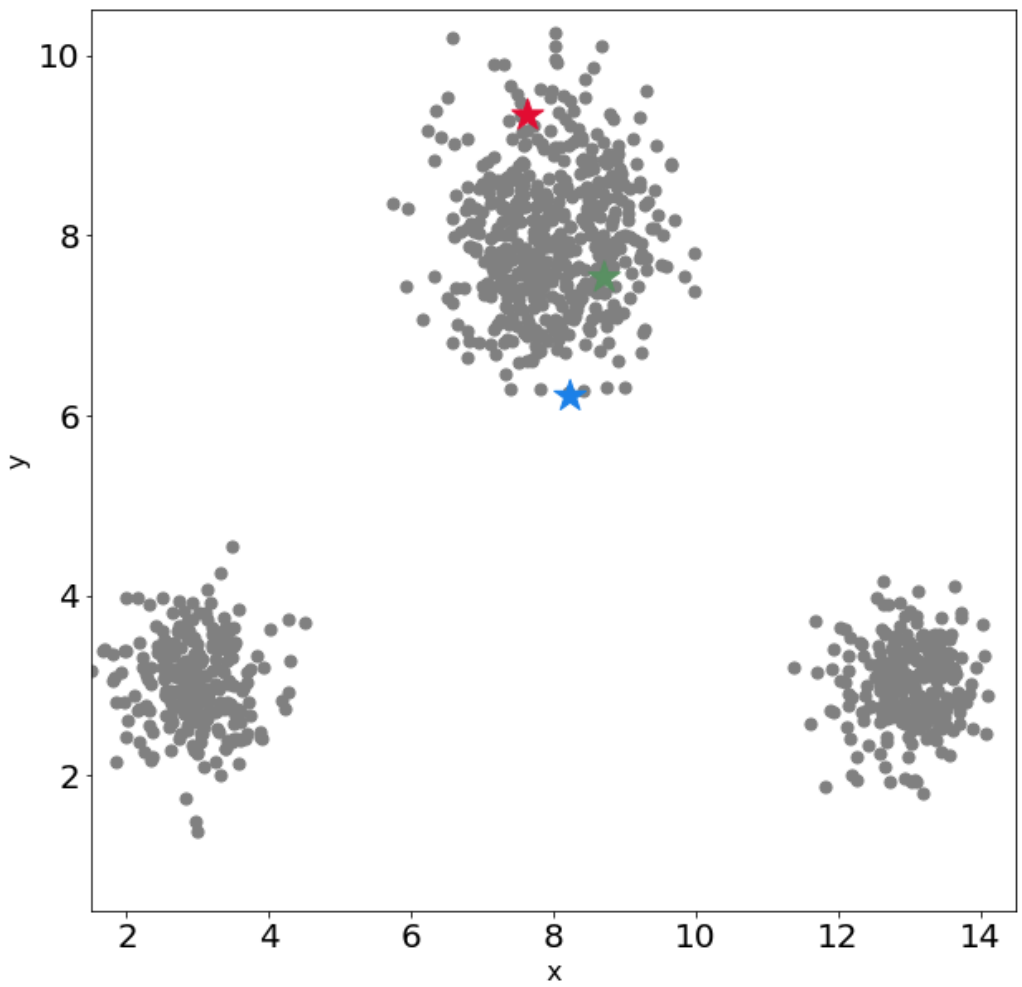


Sub-optimal Clustering

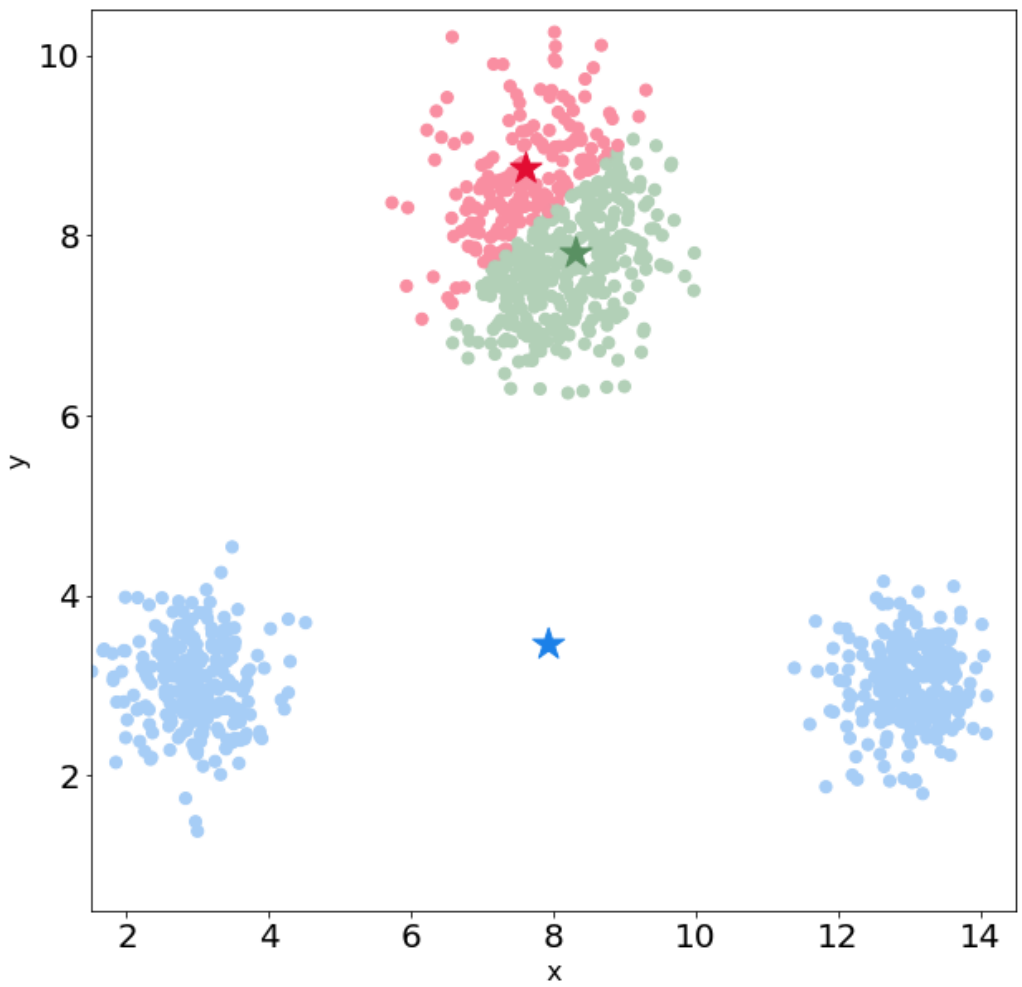


Optimal Clustering

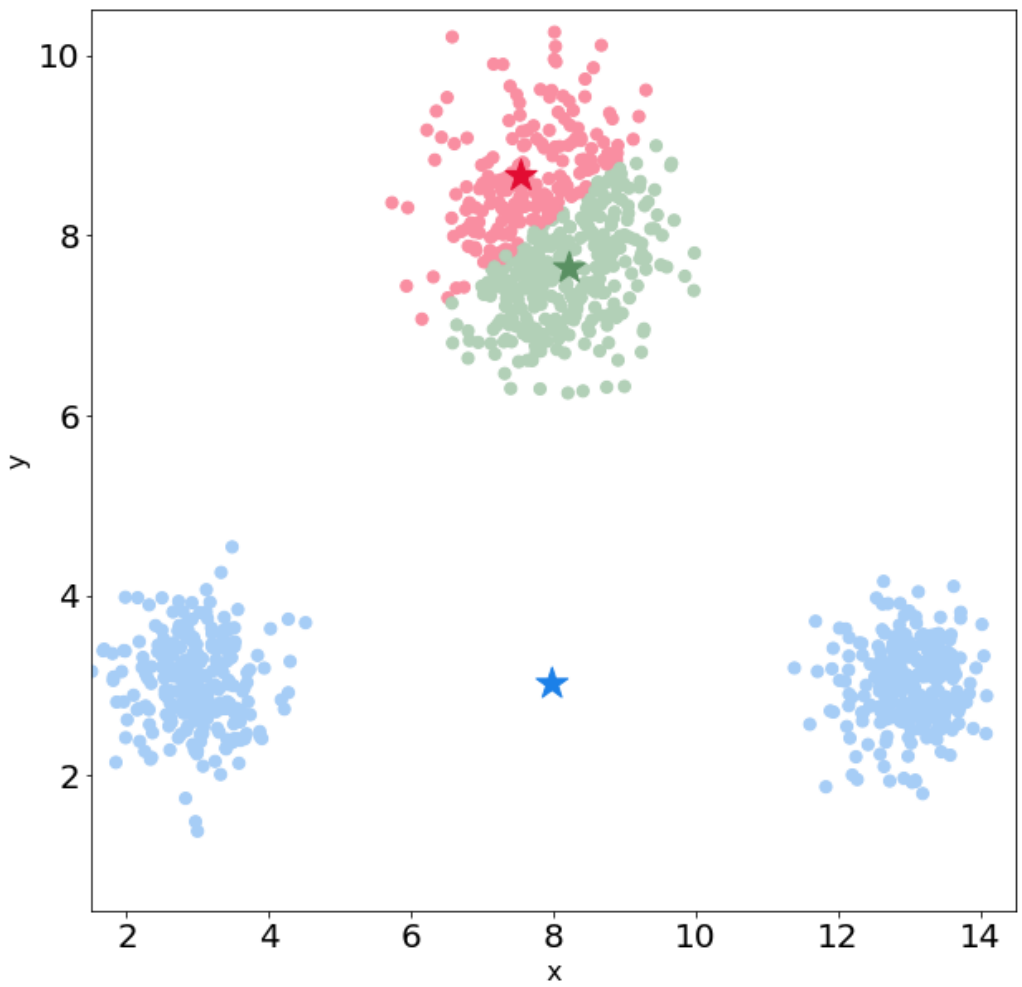
Run #1



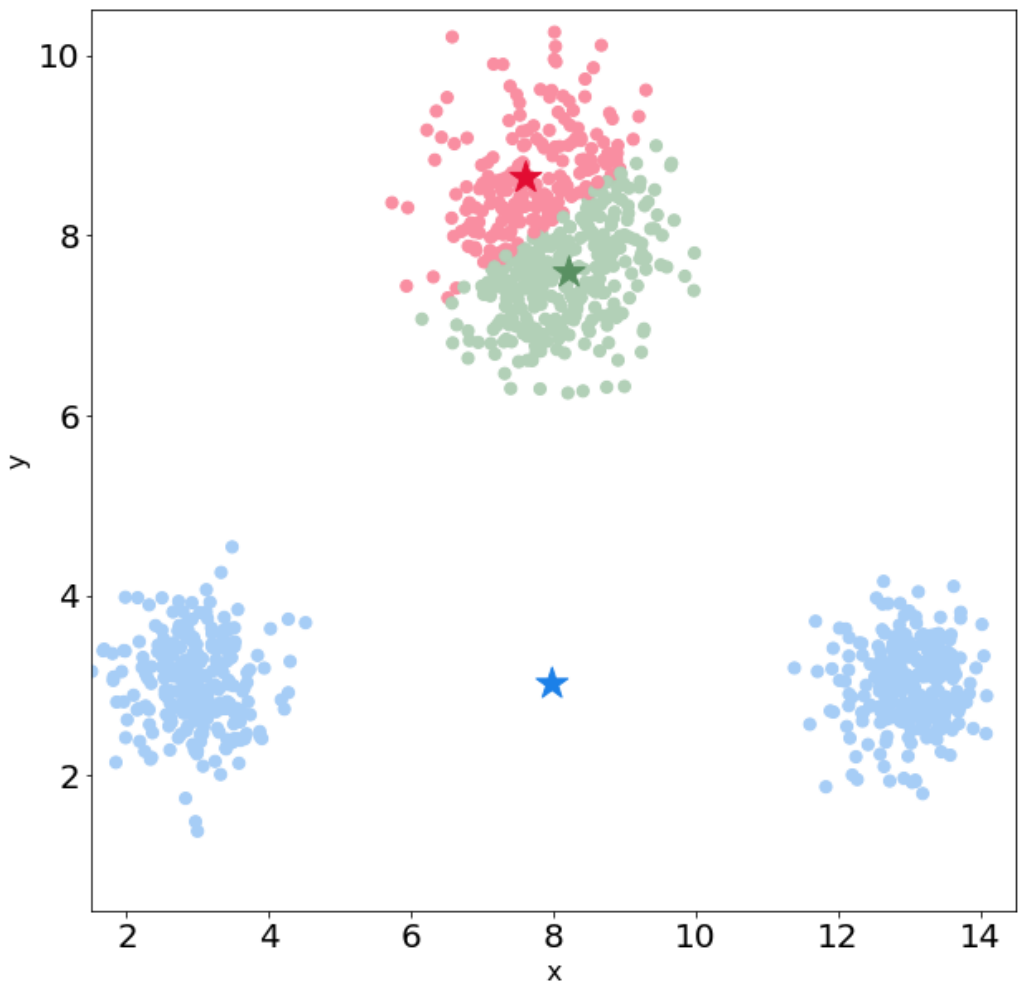
Run #1 – Start



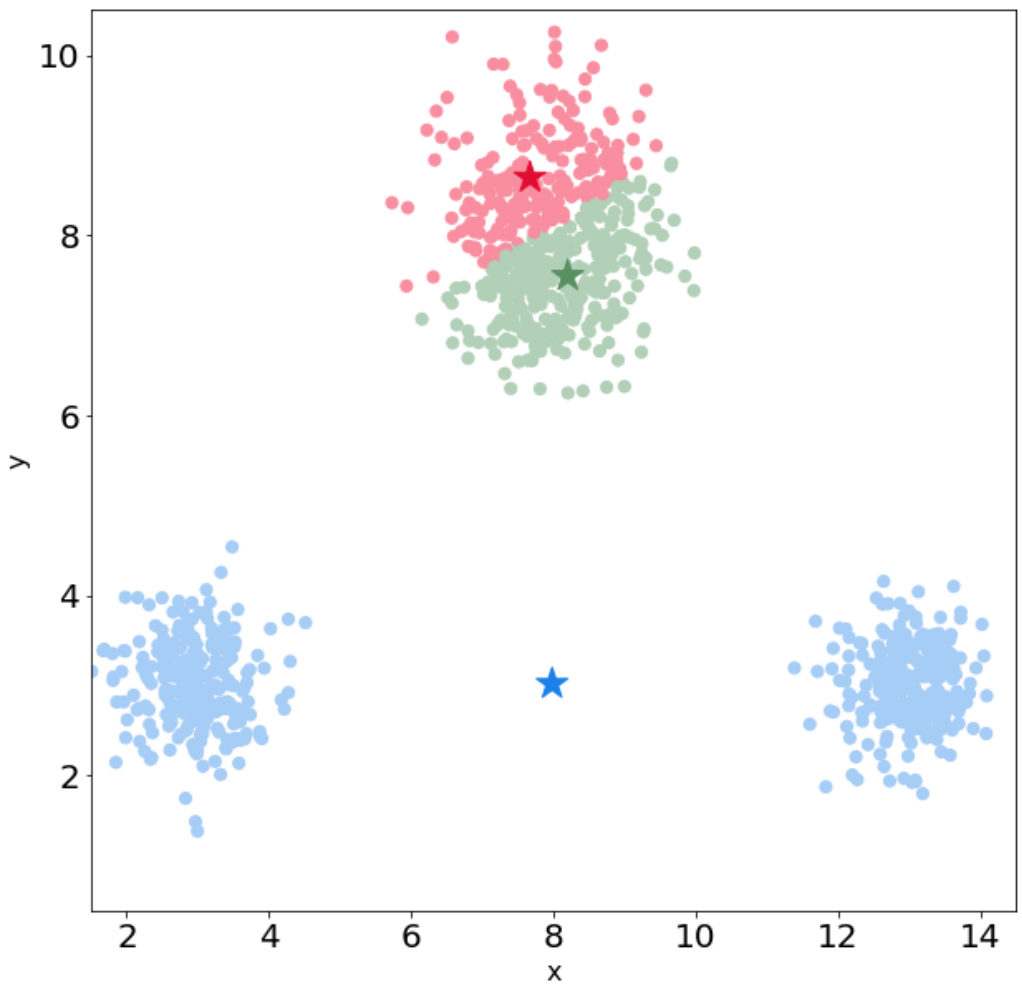
Run #1 – Step 1



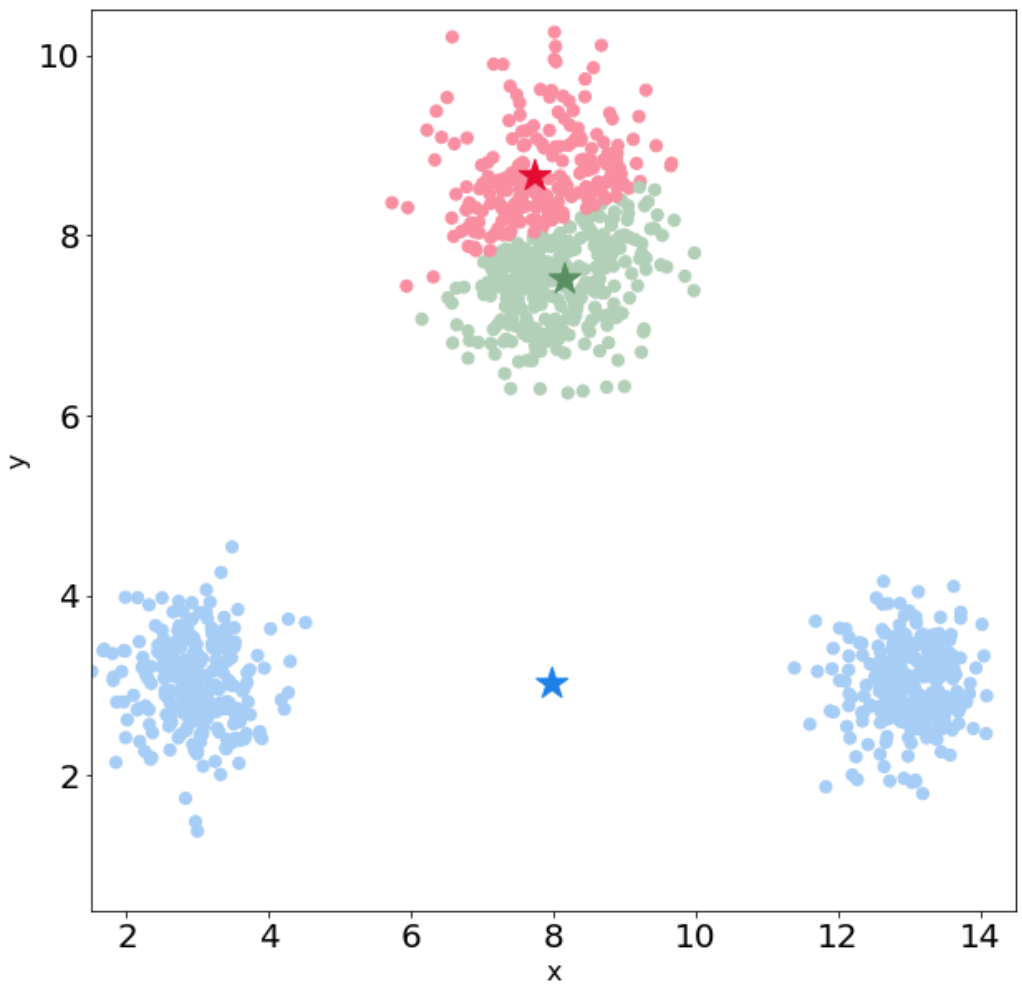
Run #1 – Step 2



Run #1 – Step 3

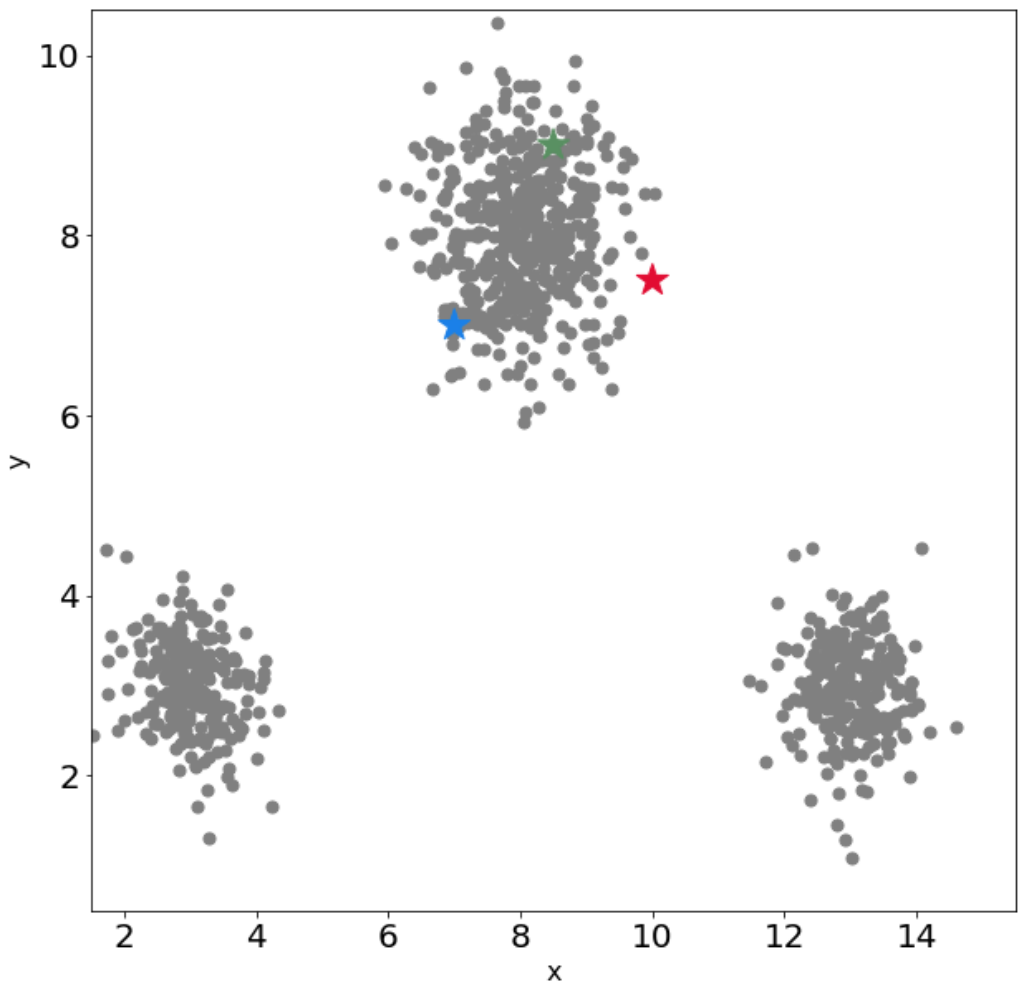


Run #1 – Step 4

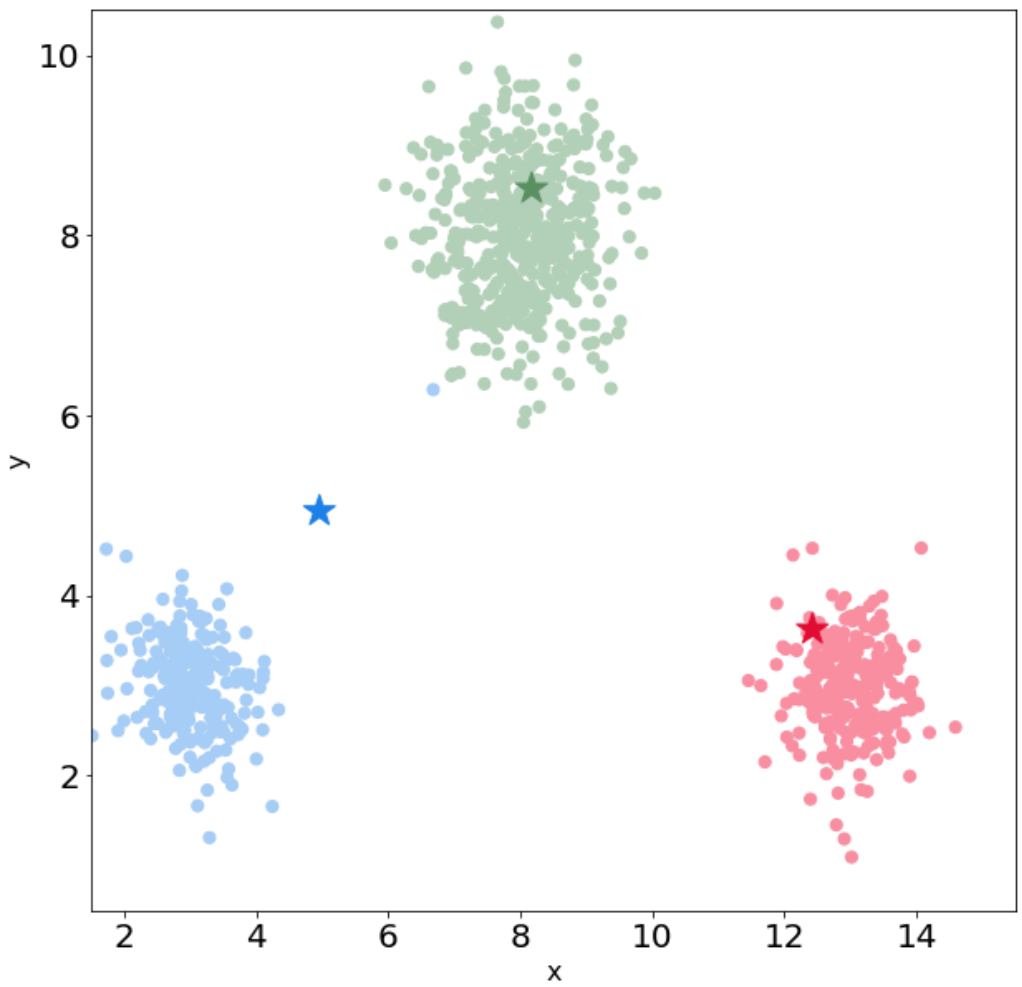


Run #1 – Step 5

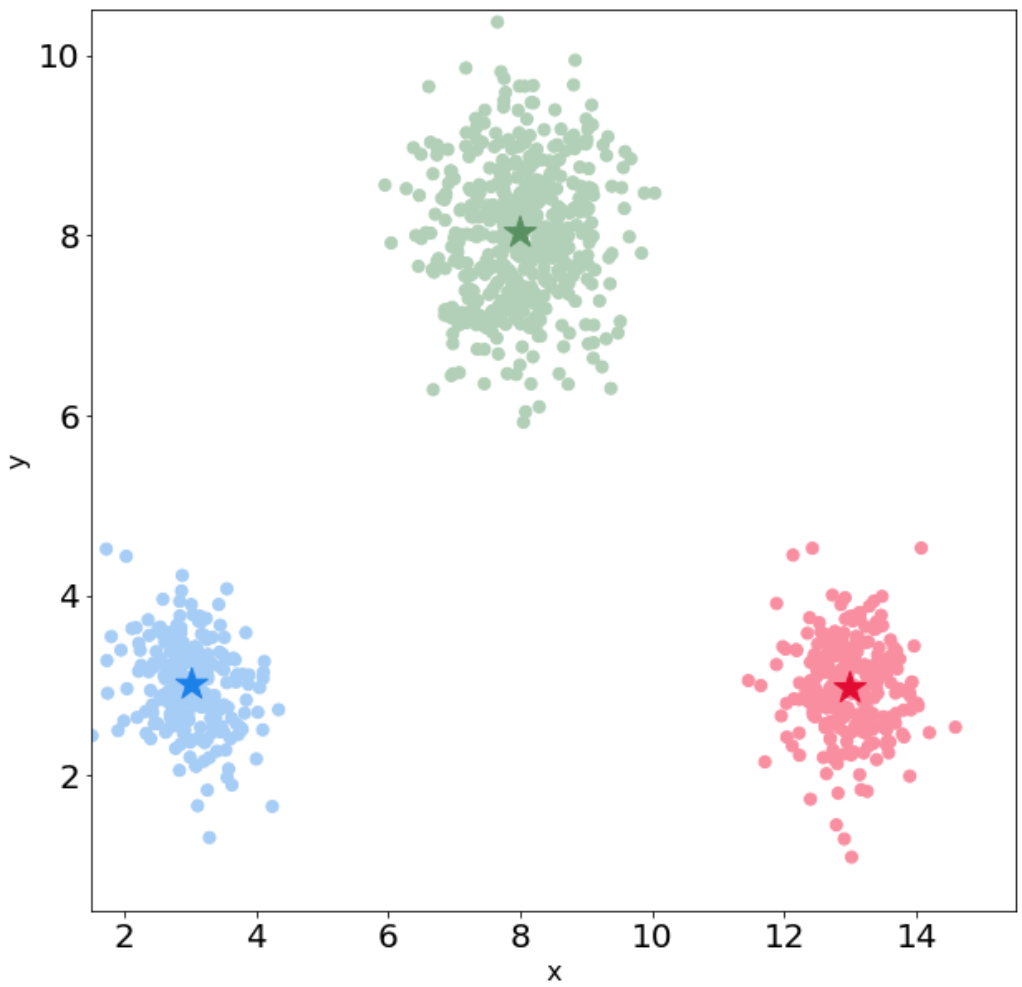
Run #2



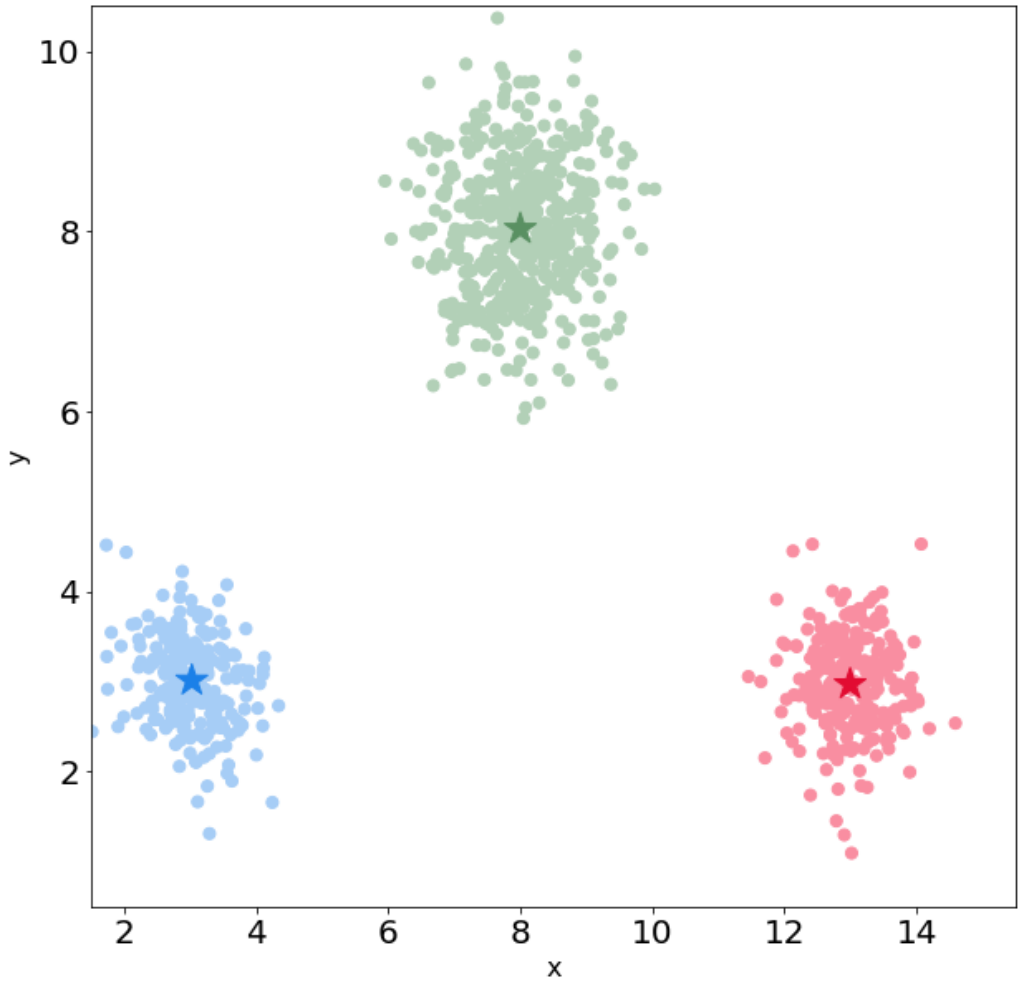
Run #2 – Start



Run #2 – Step 1



Run #2 – Step 2



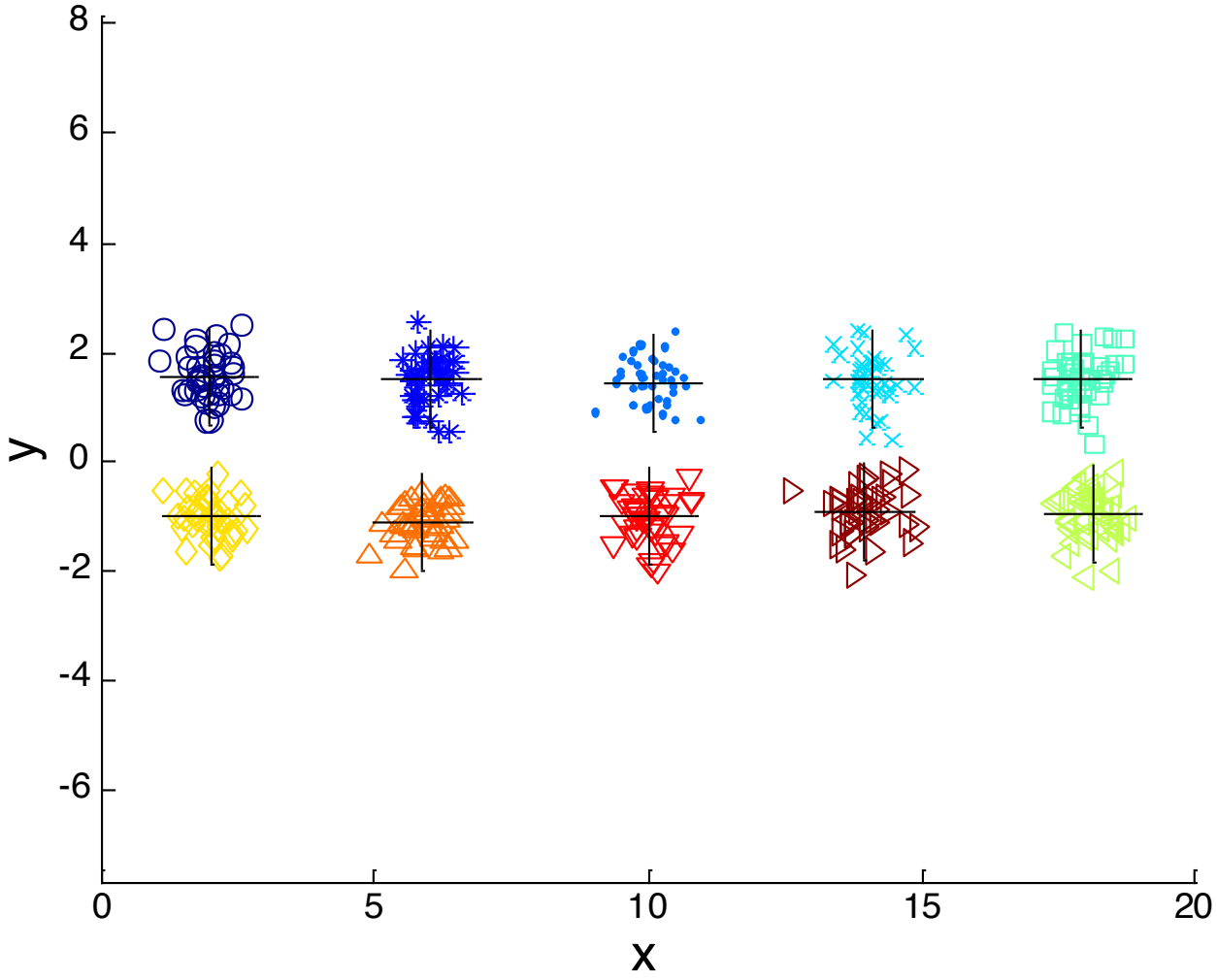
Run #2 – Step 3

Why Selecting the Best Initial Centroids is Difficult?

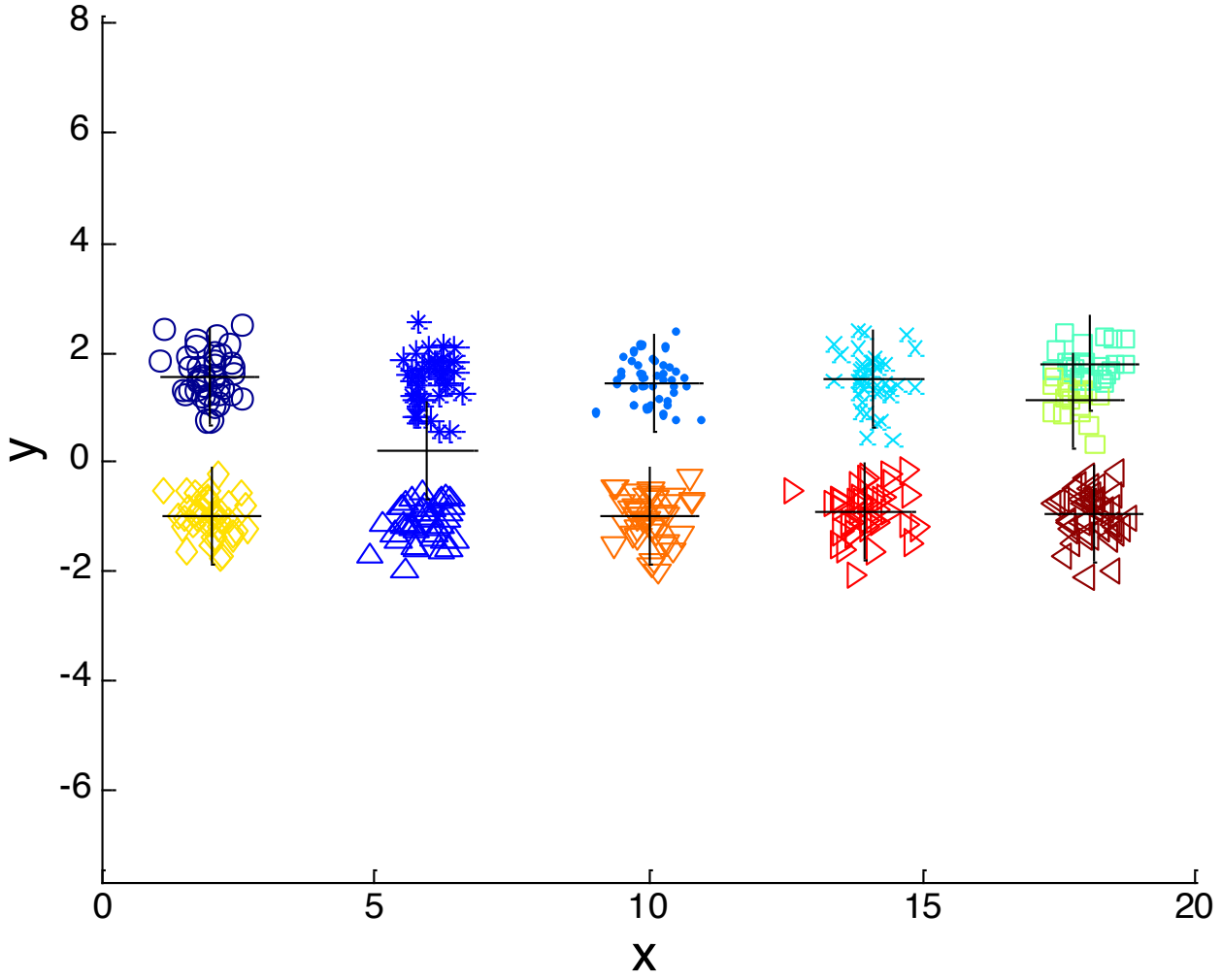
- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
- Chance is relatively small when K is large
- If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if K = 10, then probability = 10!/1010 = 0.00036
- Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
- Consider an example of five pairs of clusters



Starting with two initial centroids in one cluster of each pair of clusters

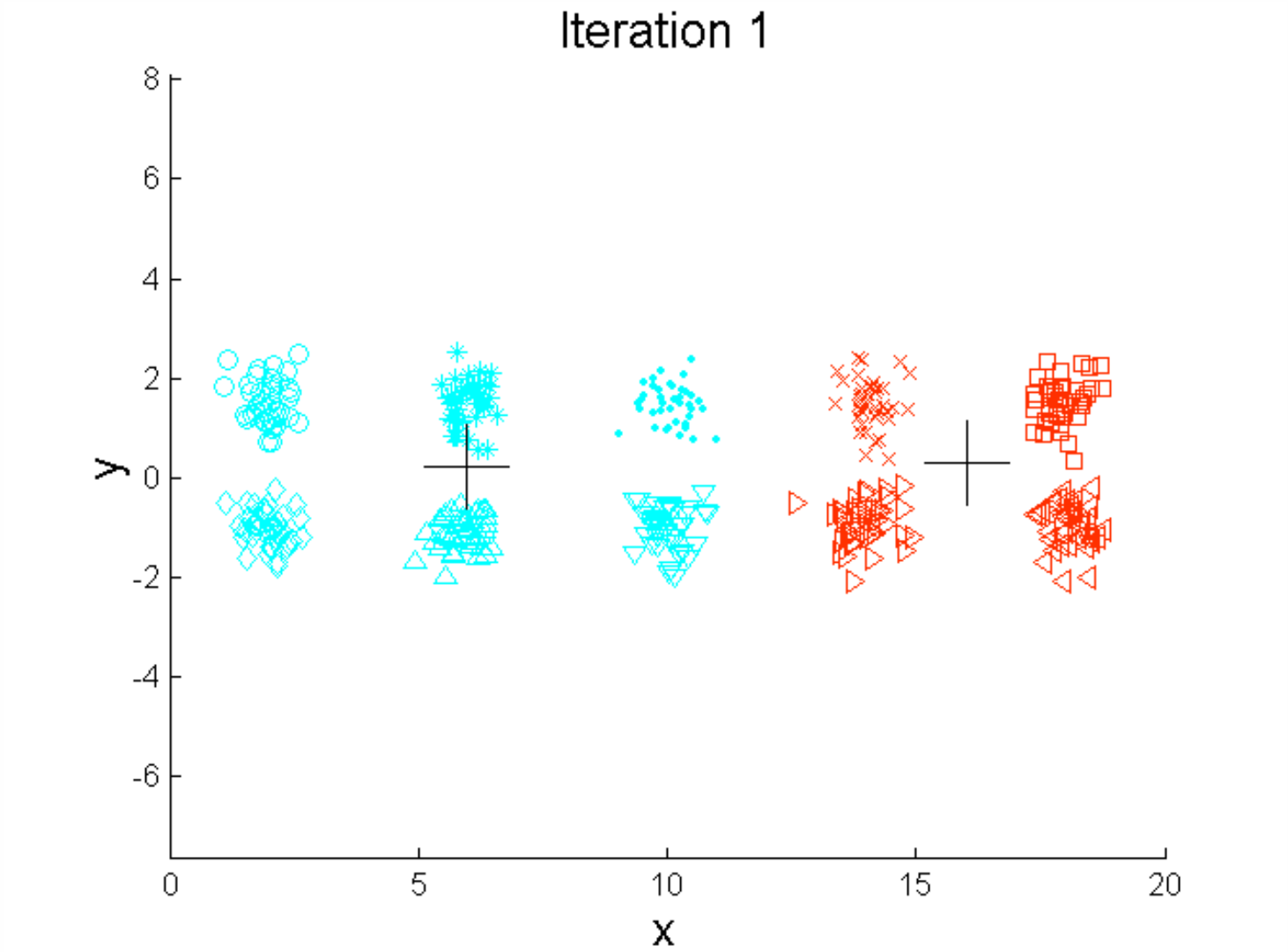


Starting with some pairs of clusters having three initial centroids, while other have only one.

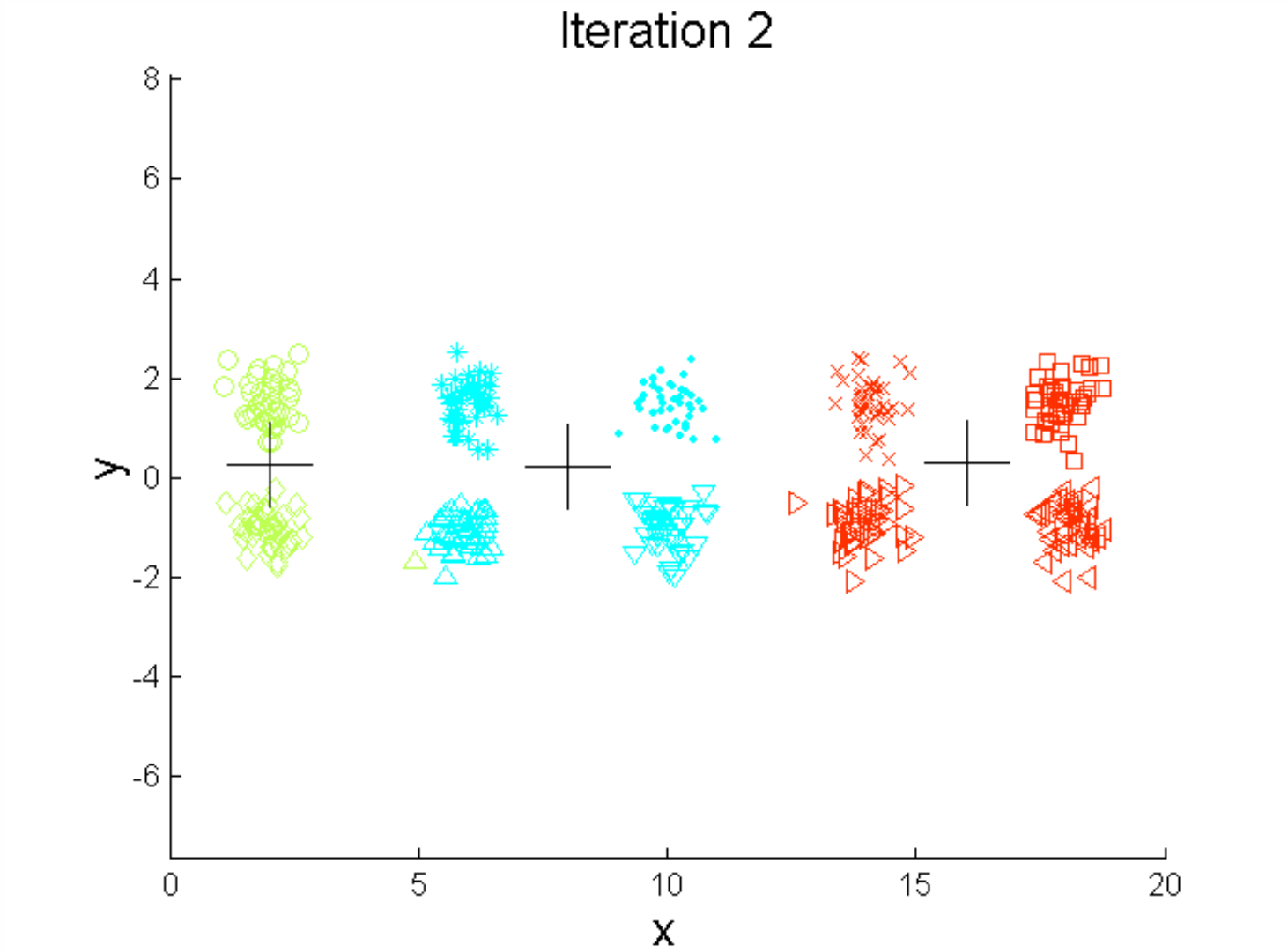
Algorithm 3 Bisecting K-means Algorithm.

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to $number\_of\_iterations$ **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

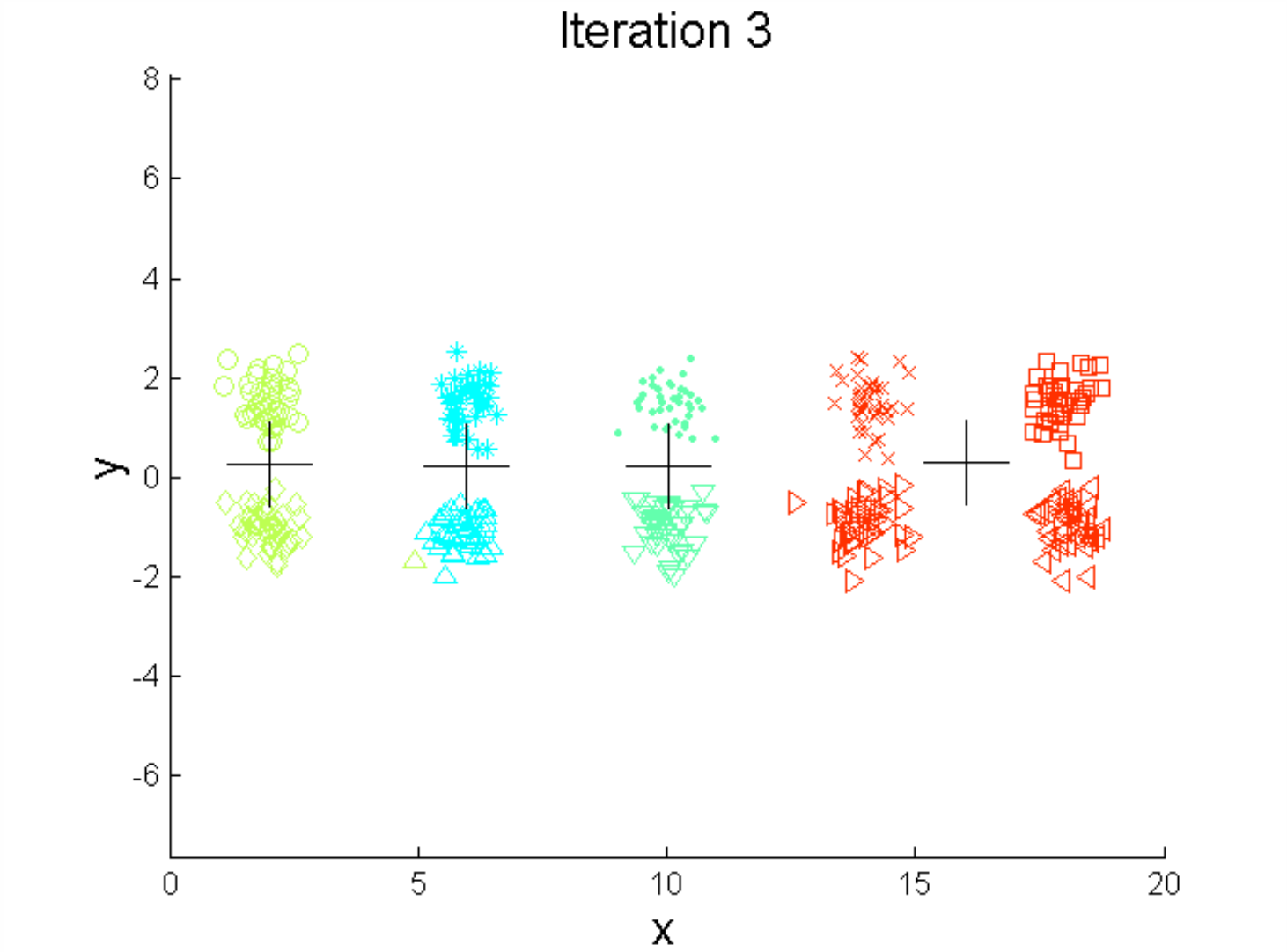
Bisecting K-means Example



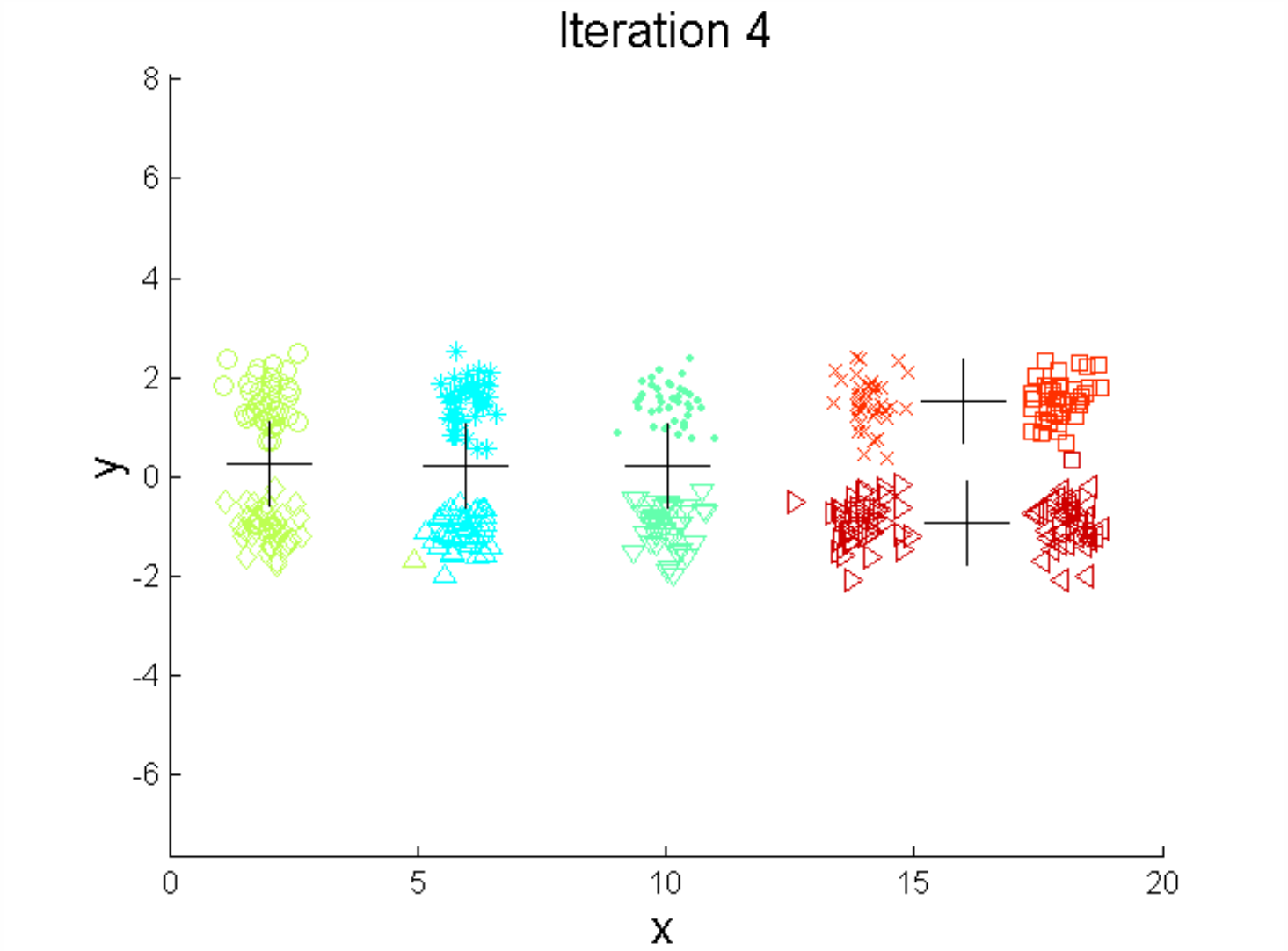
Bisecting K-means Example



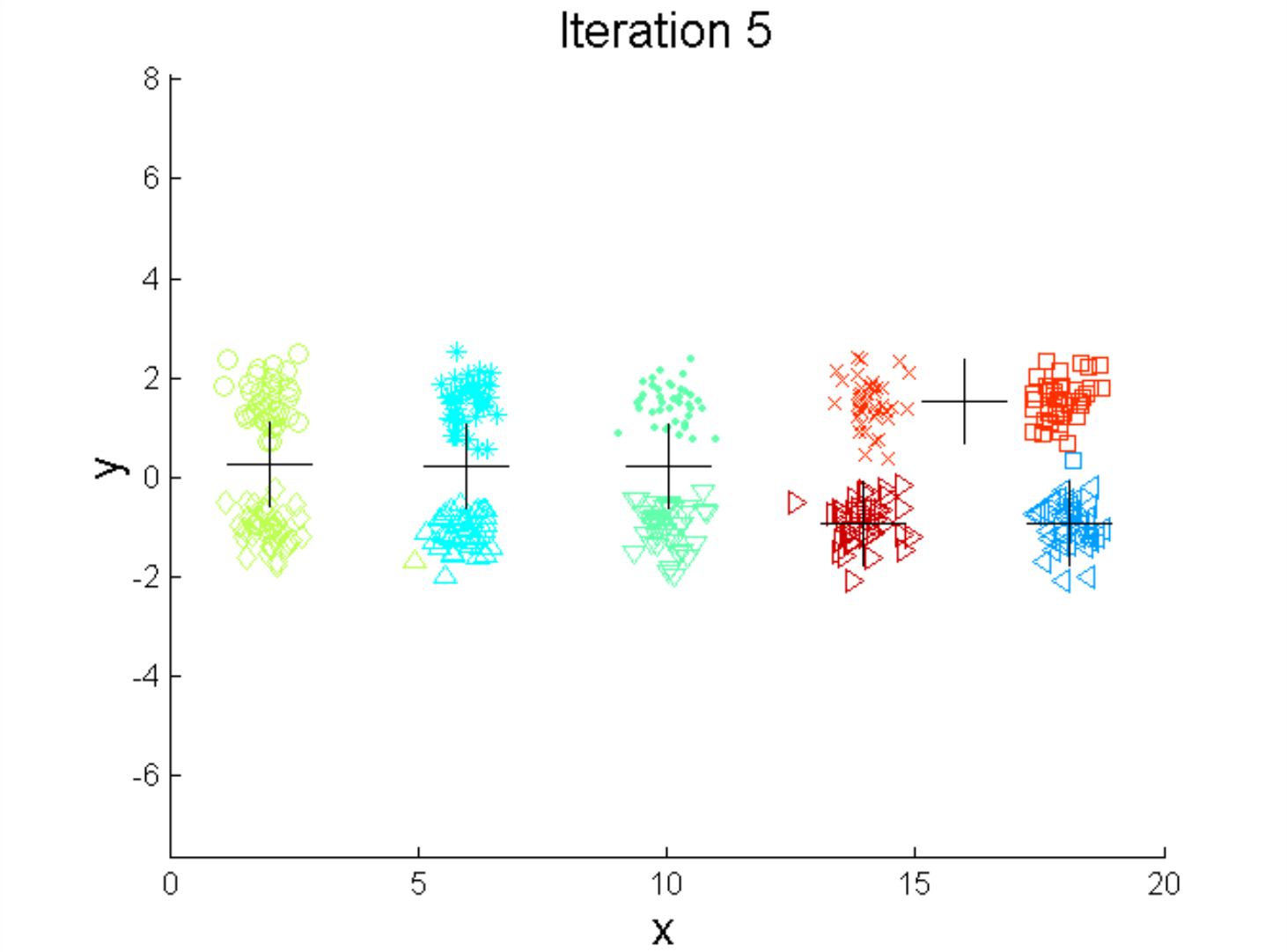
Bisecting K-means Example



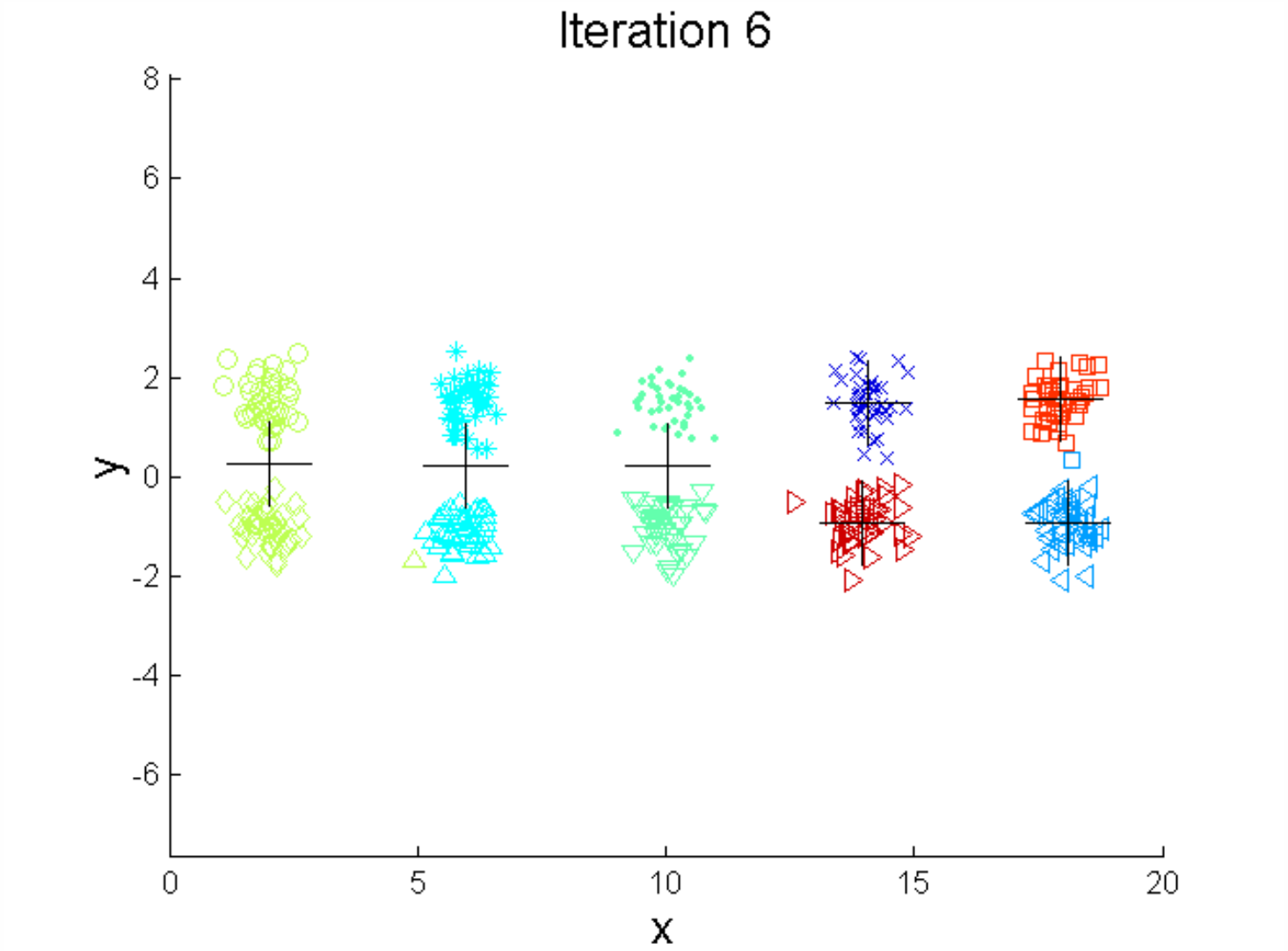
Bisecting K-means Example



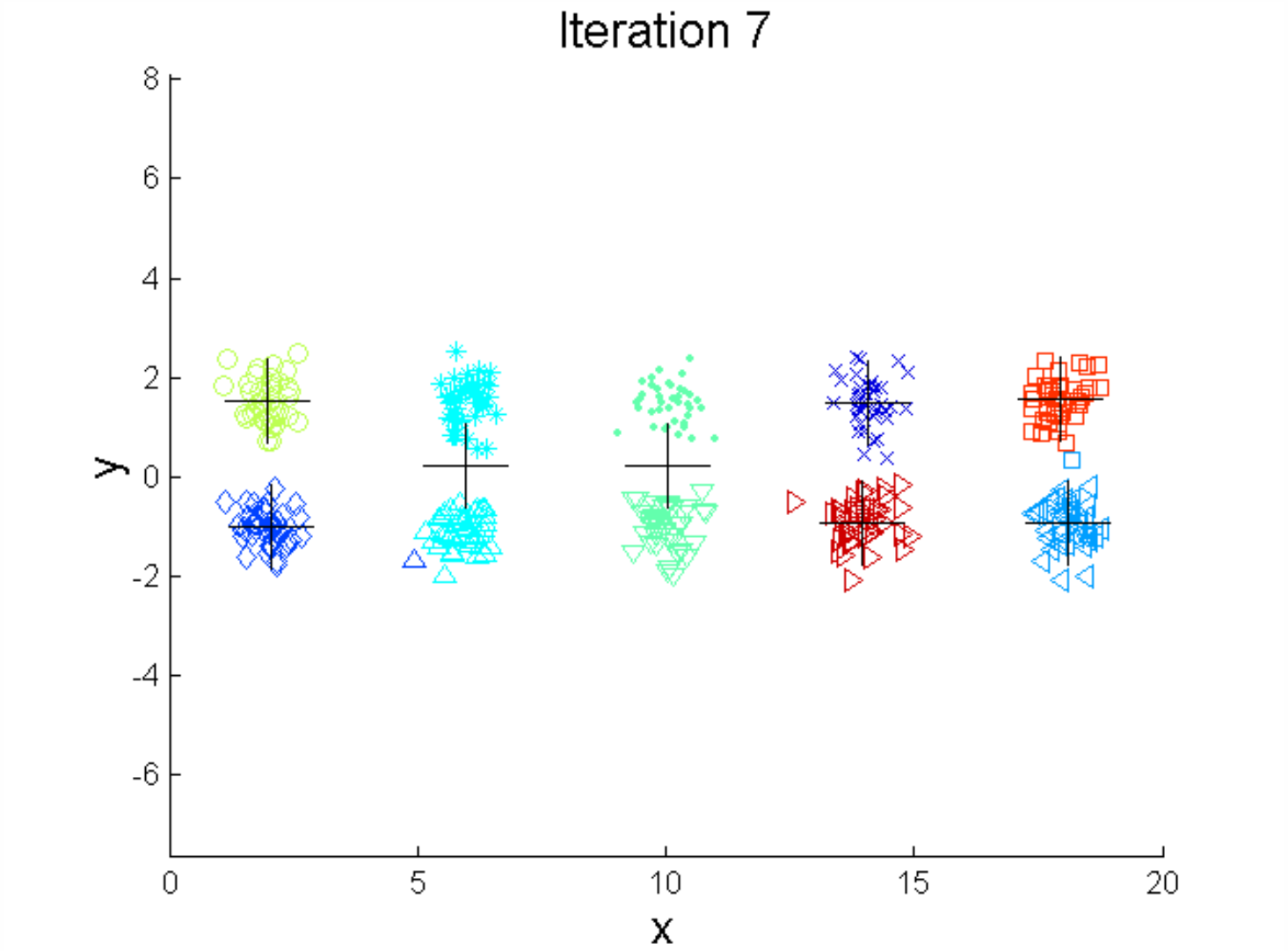
Bisecting K-means Example



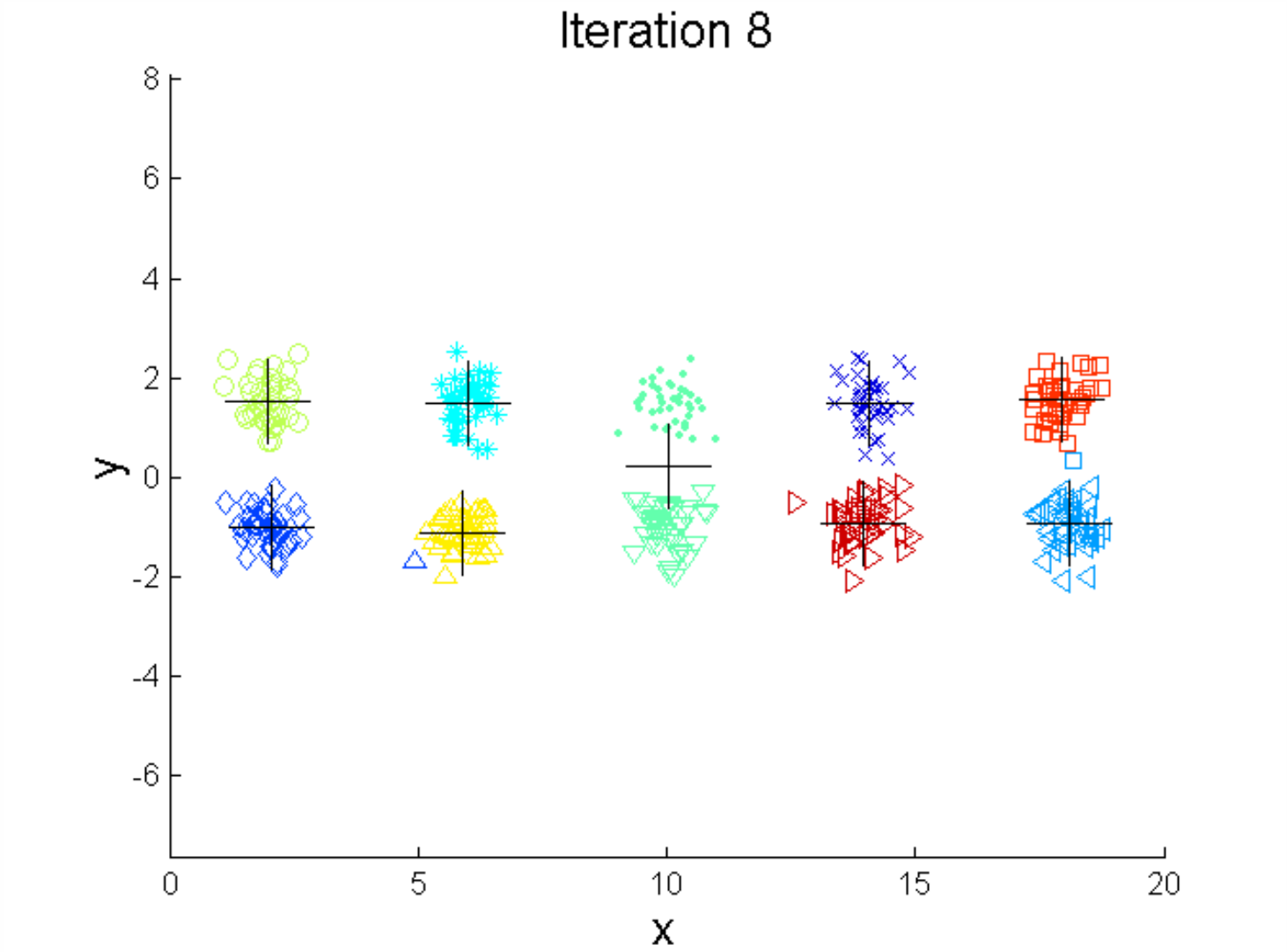
Bisecting K-means Example



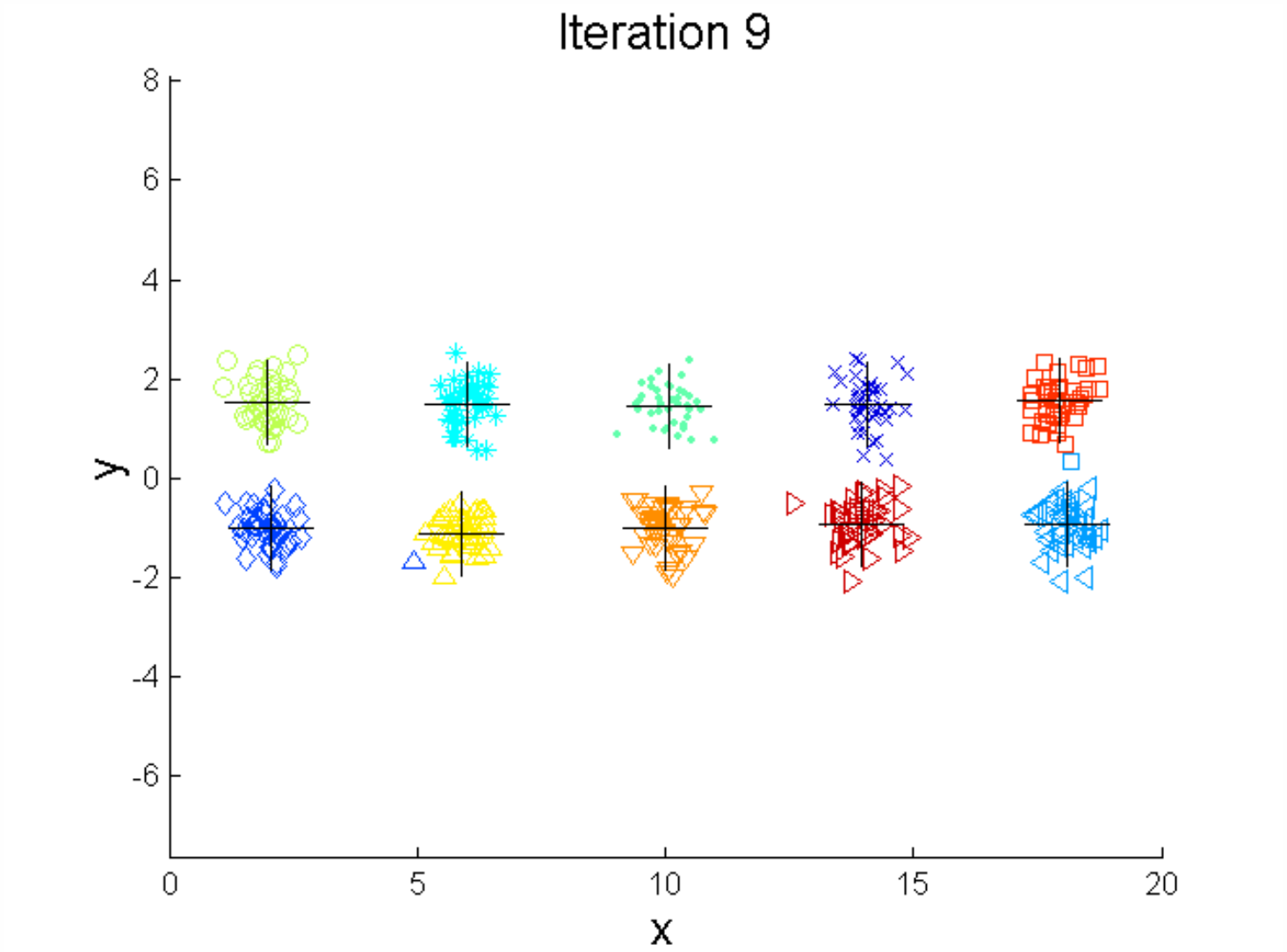
Bisecting K-means Example



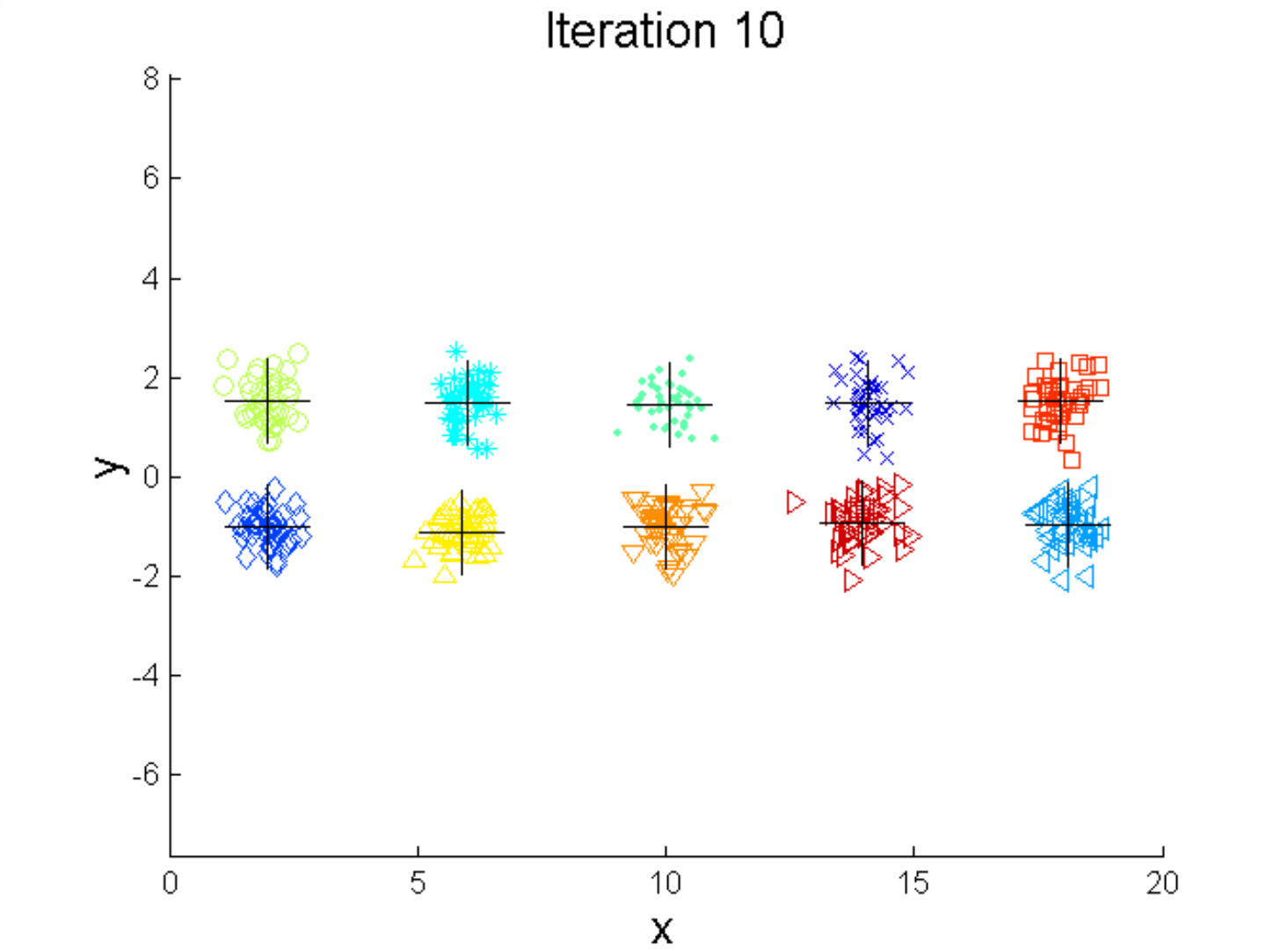
Bisecting K-means Example



Bisecting K-means Example



Bisecting K-means Example



Dealing with the Initial Centroids Issue

- Multiple runs, helps, but probability is not on your side
- Sample and use another clustering method (hierarchical?) to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
- Postprocessing
- Bisecting K-means, not as susceptible to initialization issues

Extensions

Updating Centers Incrementally

- In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid
- An alternative is to update the centroids after each assignment (incremental approach)
  - Each assignment updates zero or two centroids
  - More expensive
  - Introduces an order dependency
  - Never get an empty cluster
  - Can use "weights" to change the impact

Pre-processing and Post-processing

- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split 'loose' clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are 'close' and that have relatively low SSE
  - These steps can be used during the clustering process

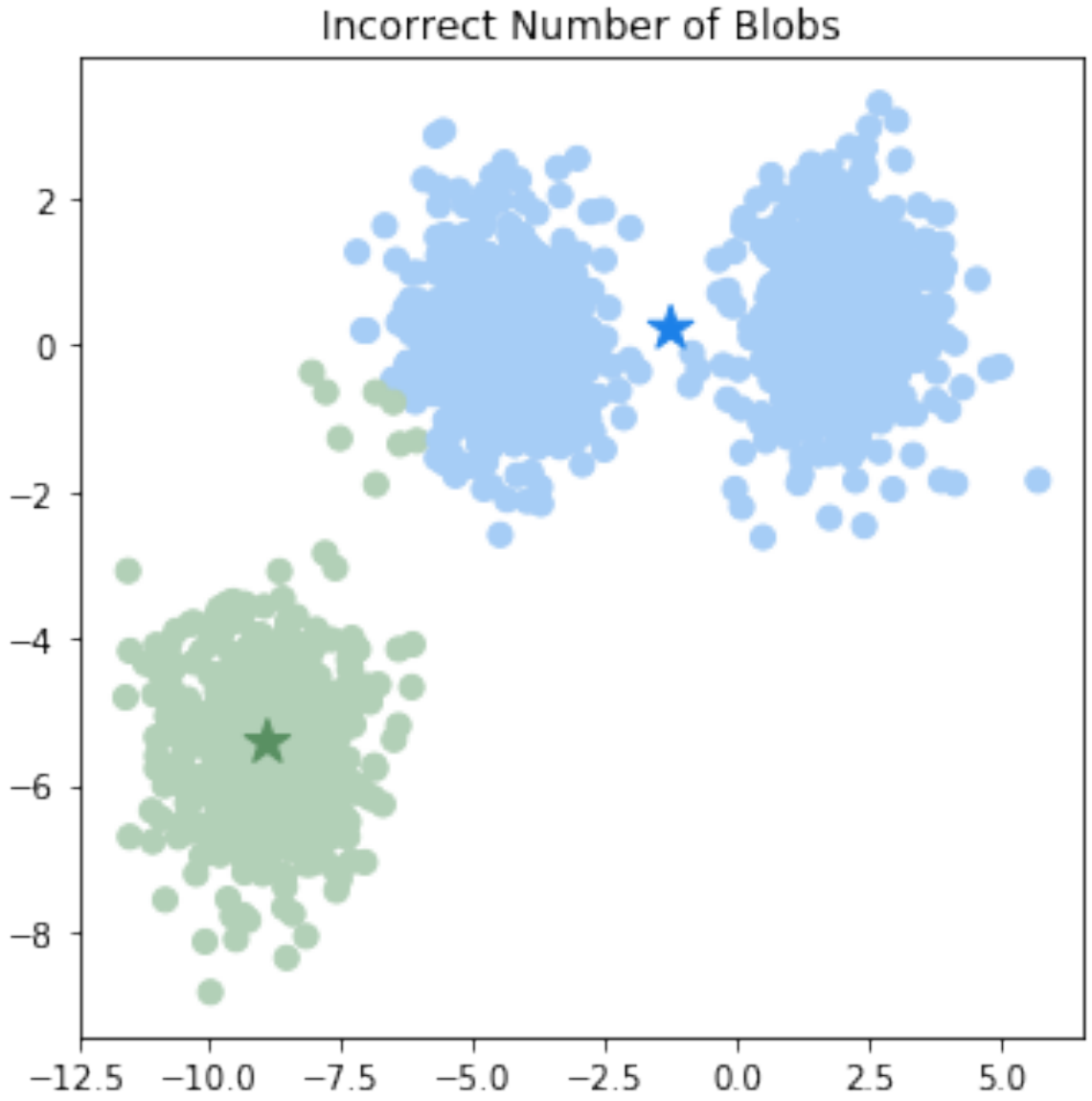
Other K-Means Variants

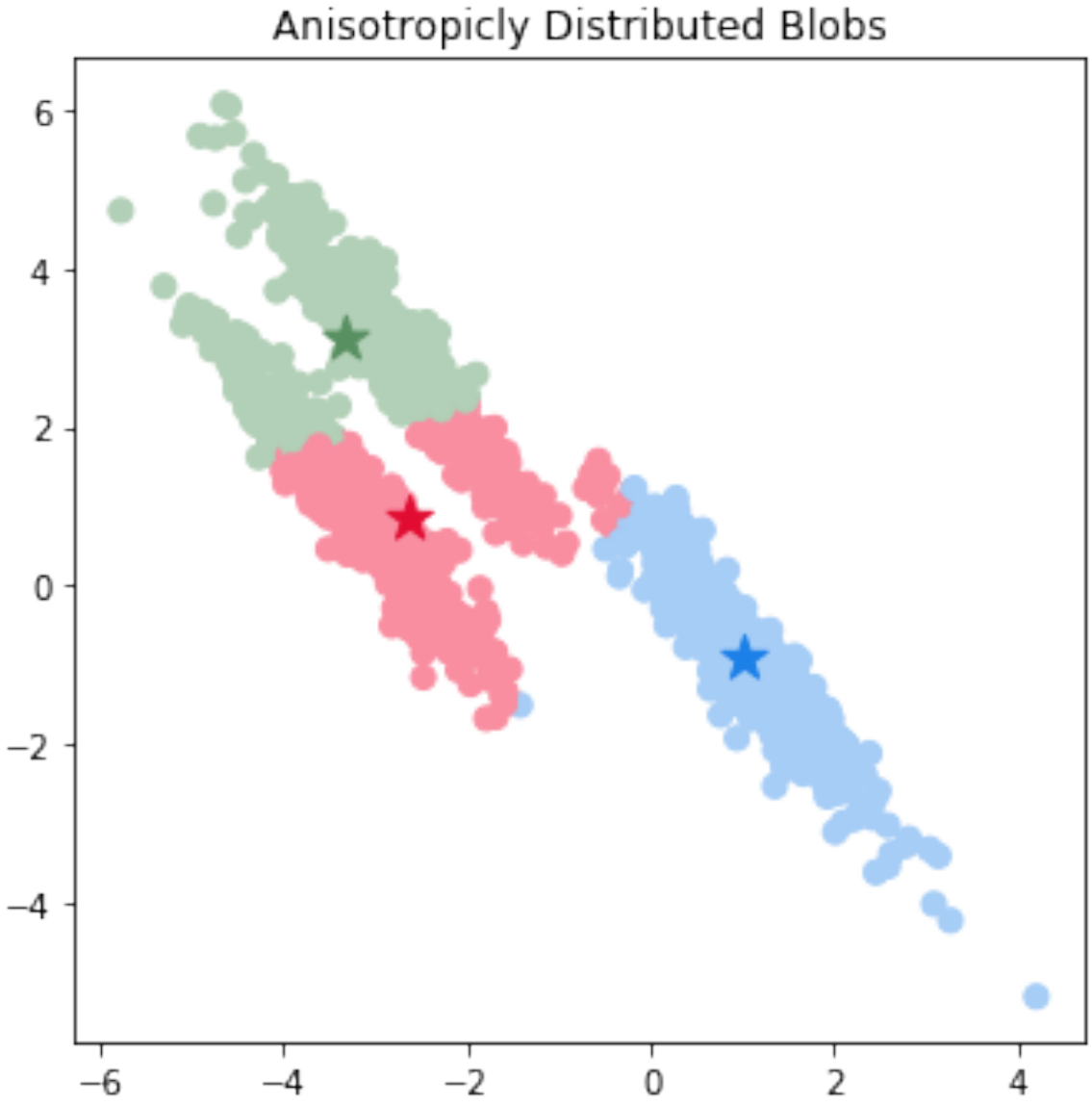
- A few variants of the k-means which differ in
  - Selection of the initial k means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: k-modes
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: k-prototype method

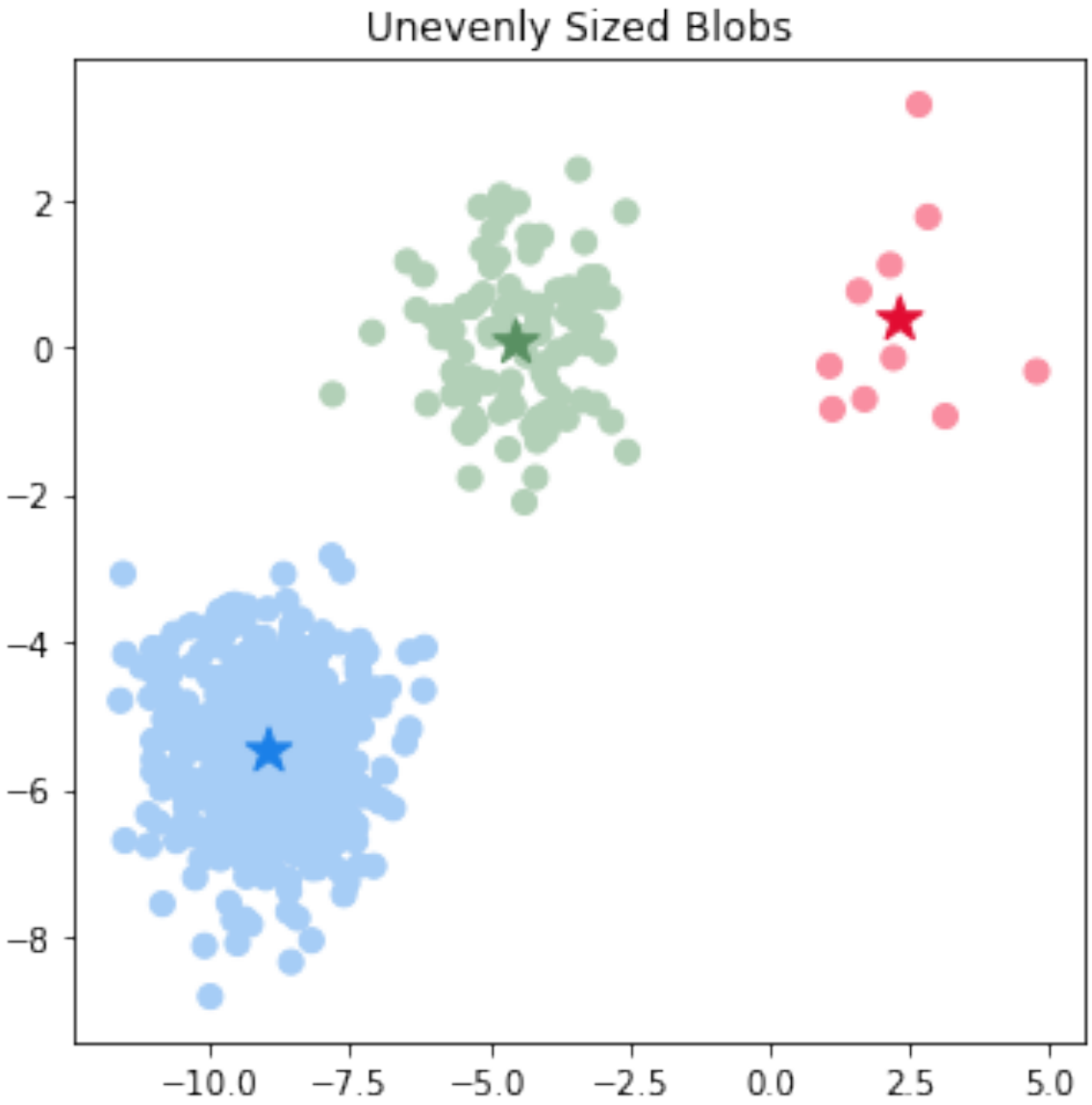
Limitations

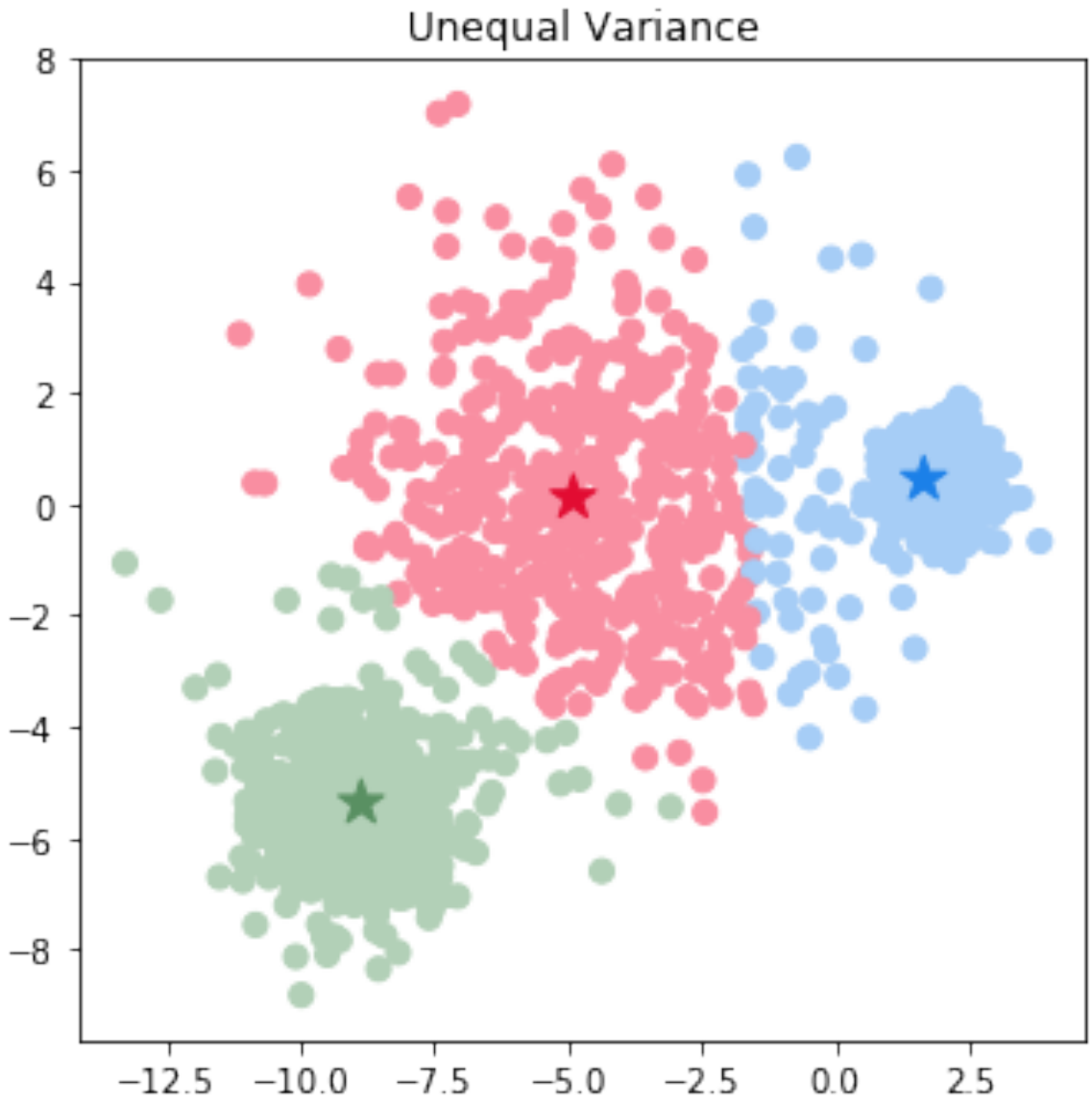
Limitations of K-means

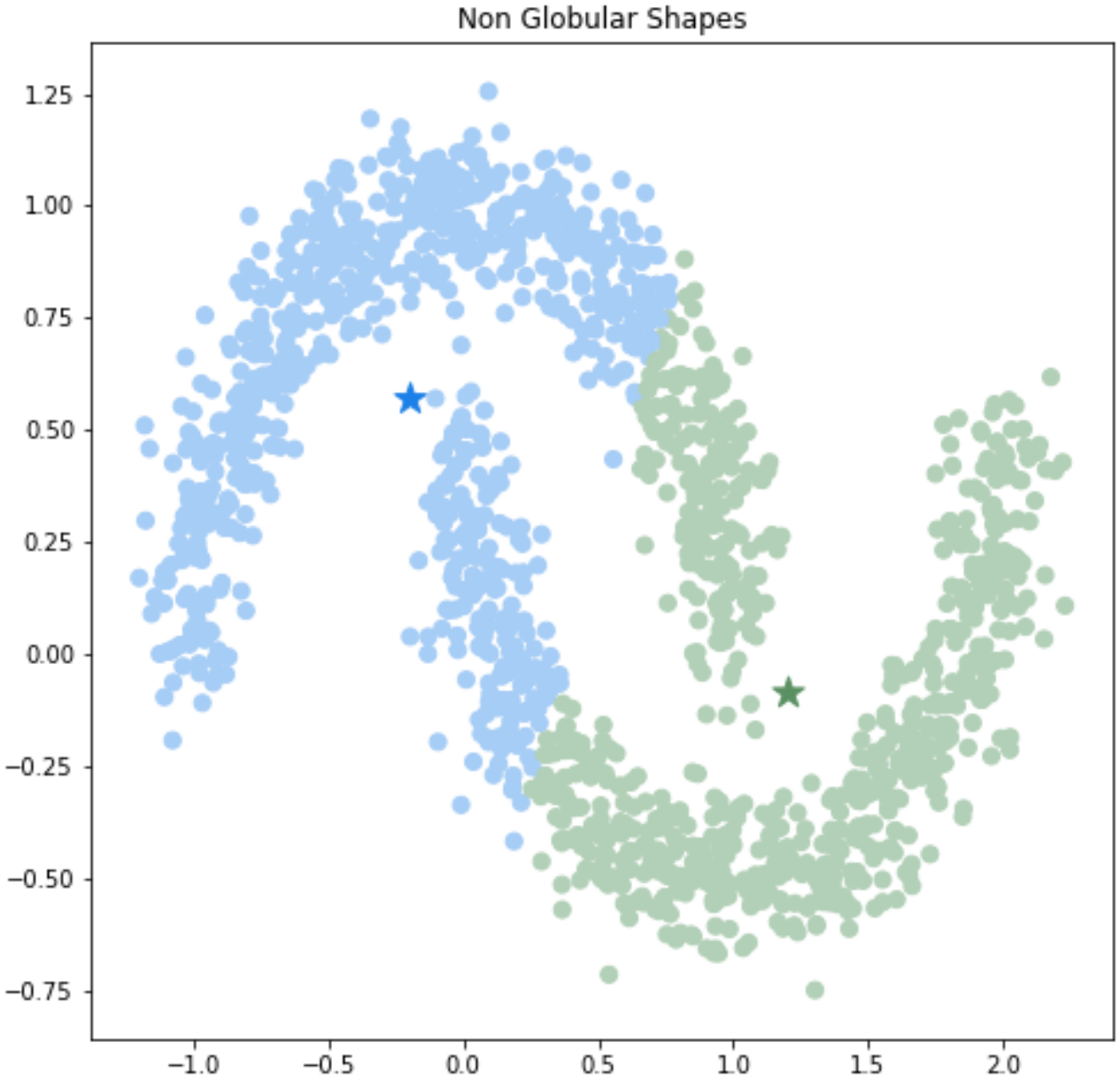
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has also problems when the data contains outliers.

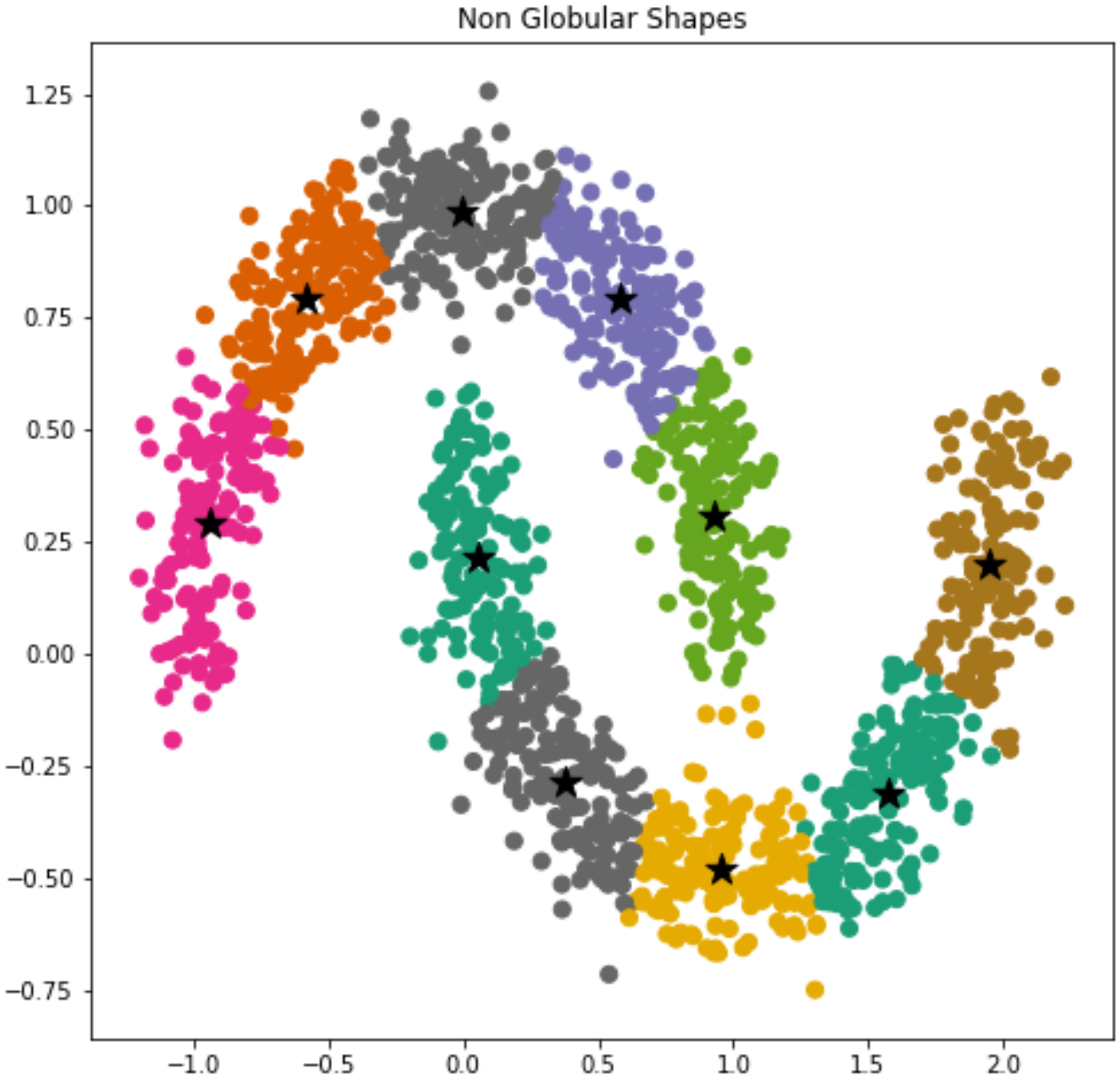

Incorrect Number of Blobs
2
0
−2
−4
−6
−8
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0


Anisotropicly Distributed Blobs
6
4
2
0
−2
−4
−6
−4
−2
0
2
4


Unevenly Sized Blobs
2
0
−2
−4
−6
−8
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0


Unequal Variance
8
6
4
2
0
−2
−4
−6
−8
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5


Non Globular Shapes
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



We can use higher values of k and then preprocess the resulting clusters and merge them when we deem it useful.

k-Means Clustering Summary

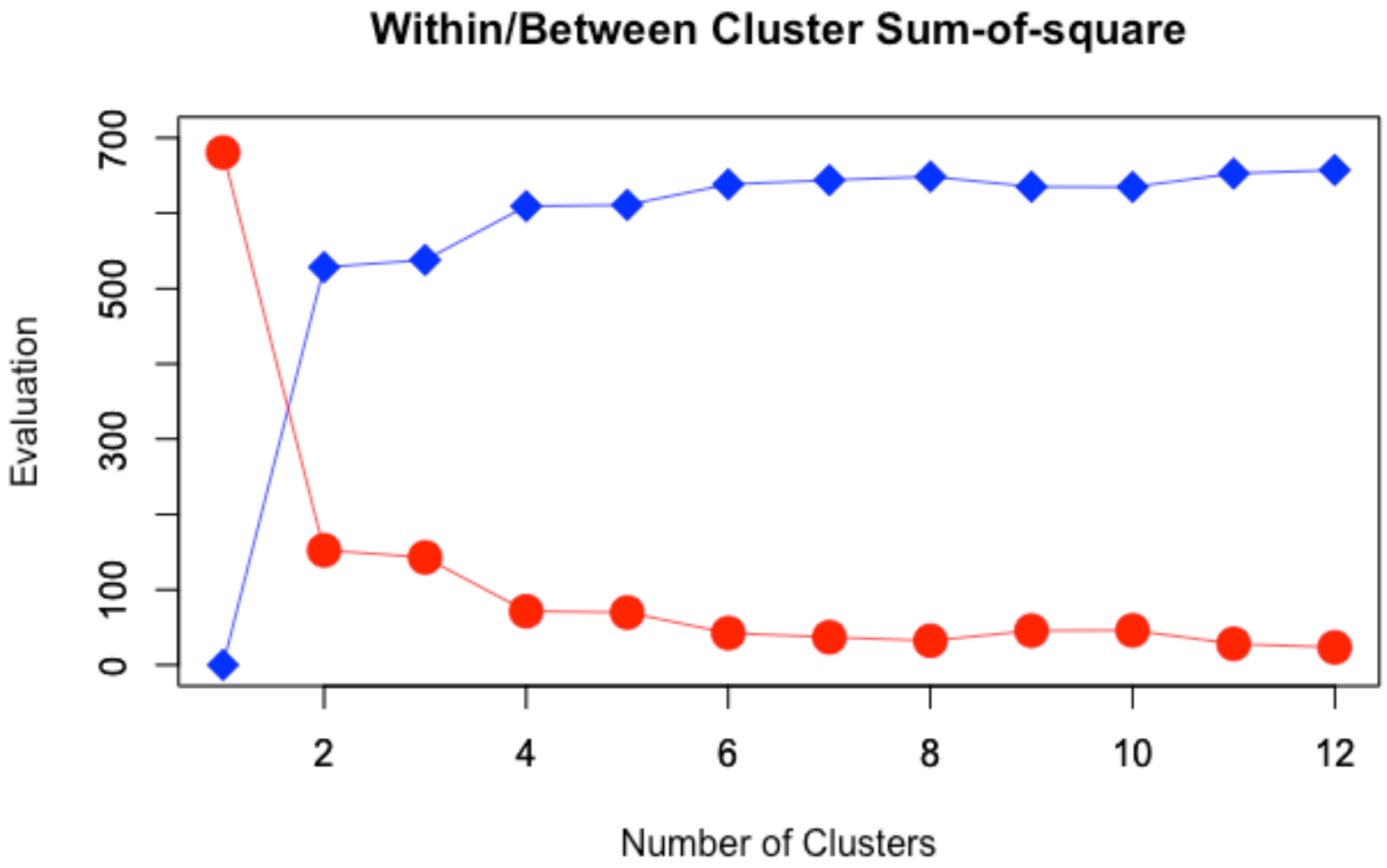
- Strength
  - Relatively efficient
  - Often terminates at a local optimum
  - The global optimum may be found using techniques such as: deterministic annealing and genetic algorithms
- Weakness
  - Applicable only when mean is defined, then what about categorical data?
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes
- Advantages
  - Simple, understandable
  - Items automatically assigned to clusters
- Disadvantages
  - Must pick number of clusters before hand
  - All items forced into a cluster
  - Too sensitive to outliers

Variants of the K-Means Method

- A few variants of the k-means which differ in
  - Selection of the initial k means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: k-modes
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: k-prototype method

How do we choose k?

Elbow & Knee Analysis

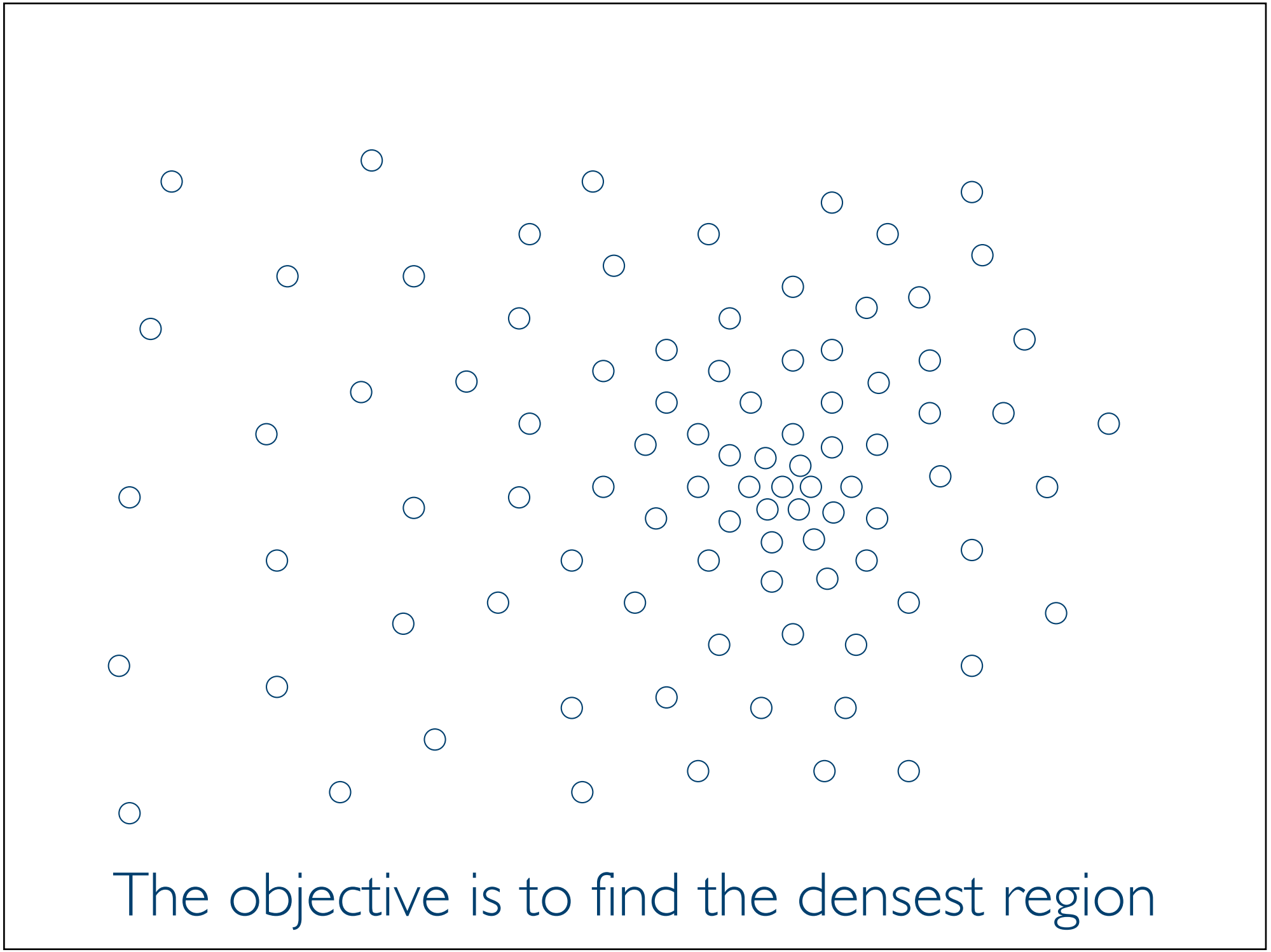


Mean Shift Clustering

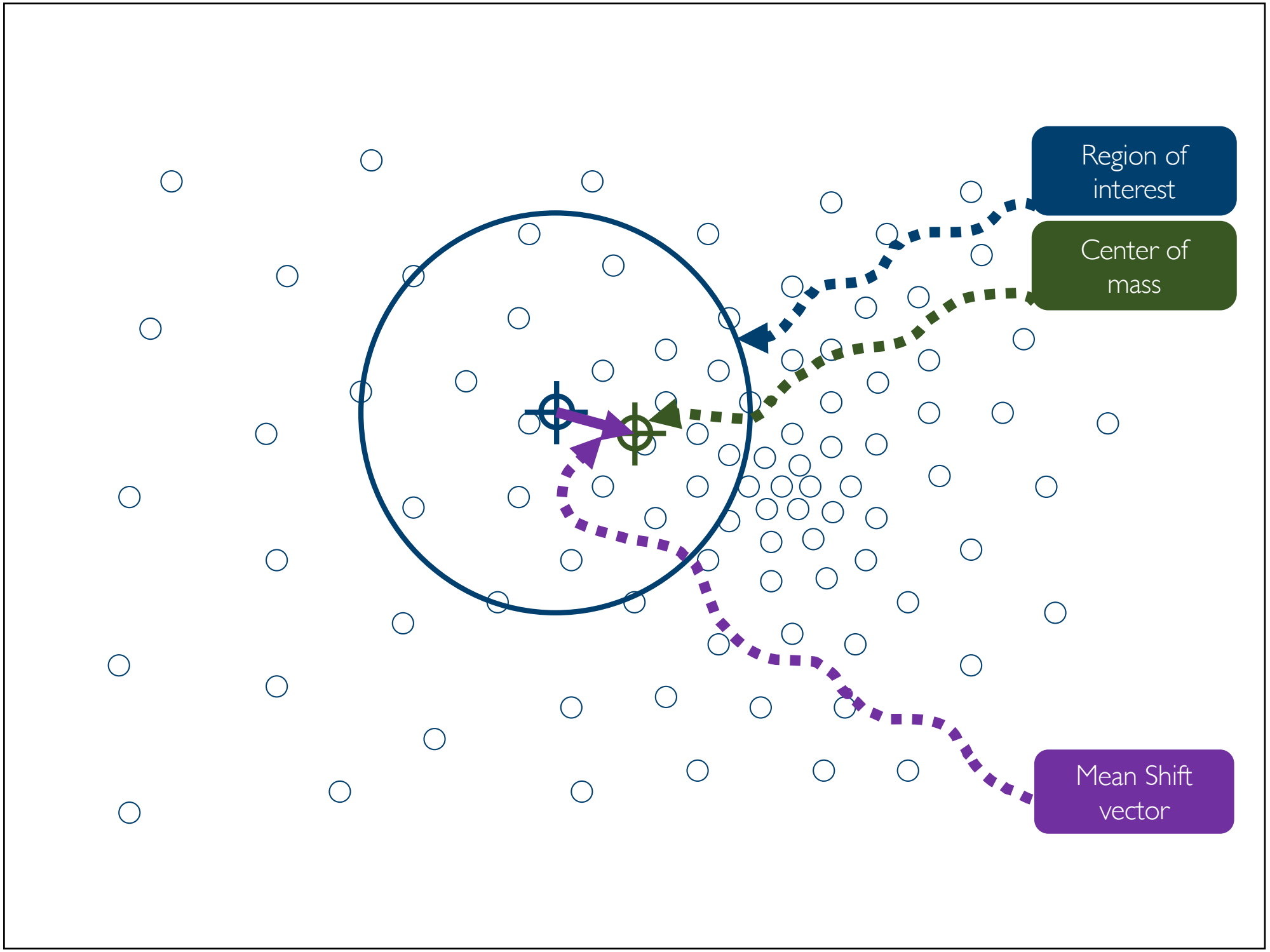
Mean shift is an iterative, nonparametric, and versatile algorithm

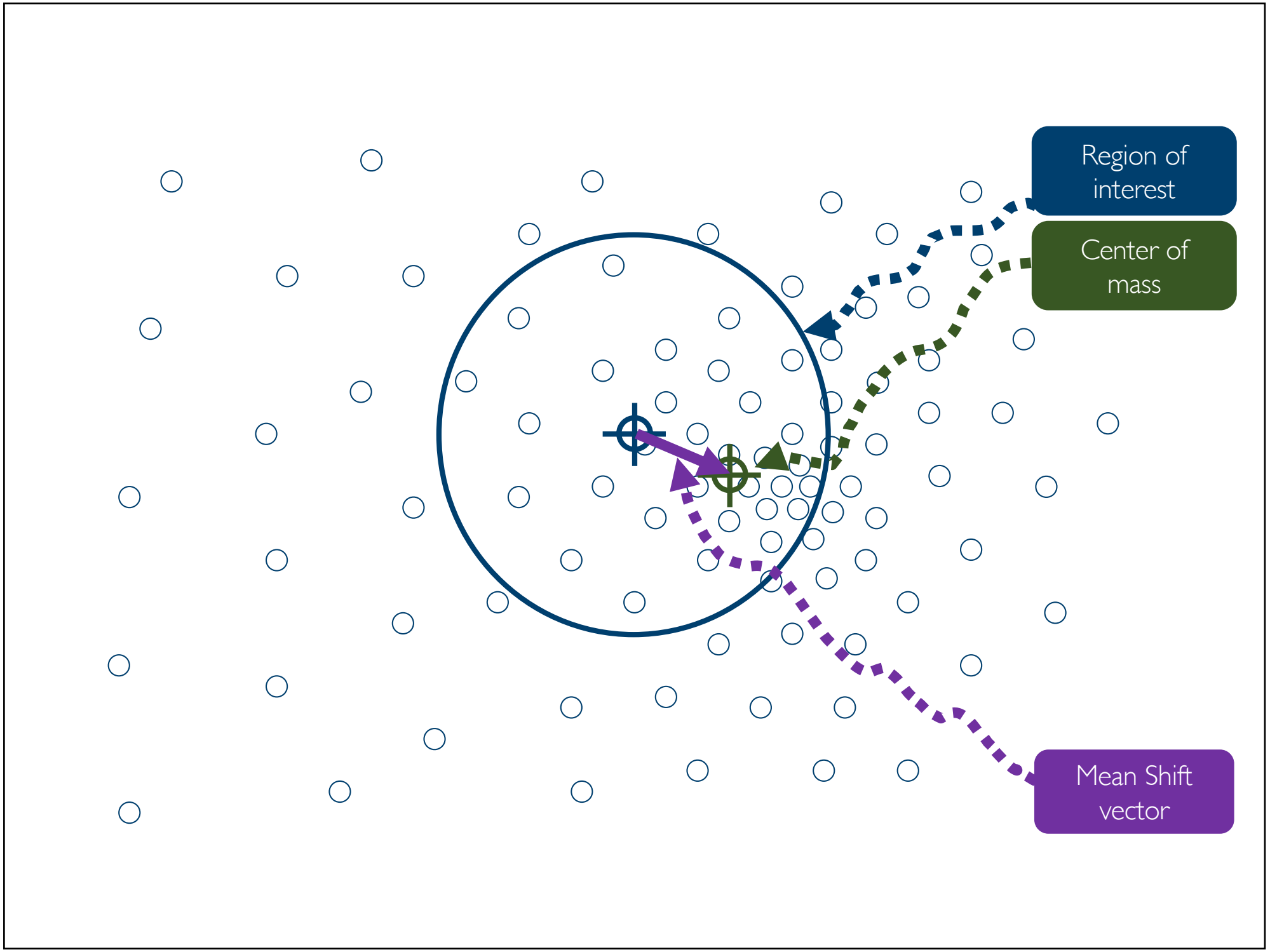
It searches for the mode (i.e., the point of highest density) of a data distribution

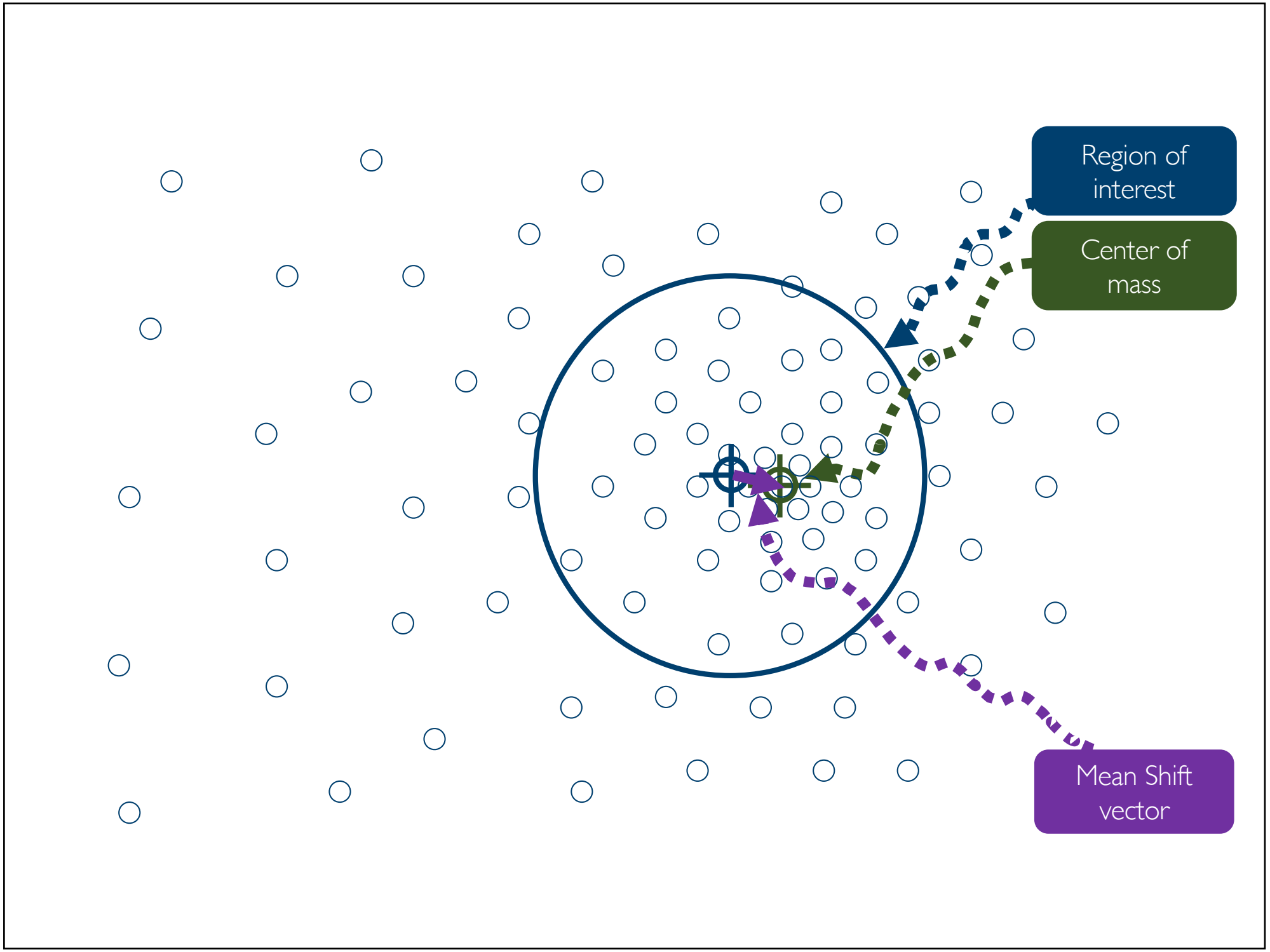
Underlying intuition

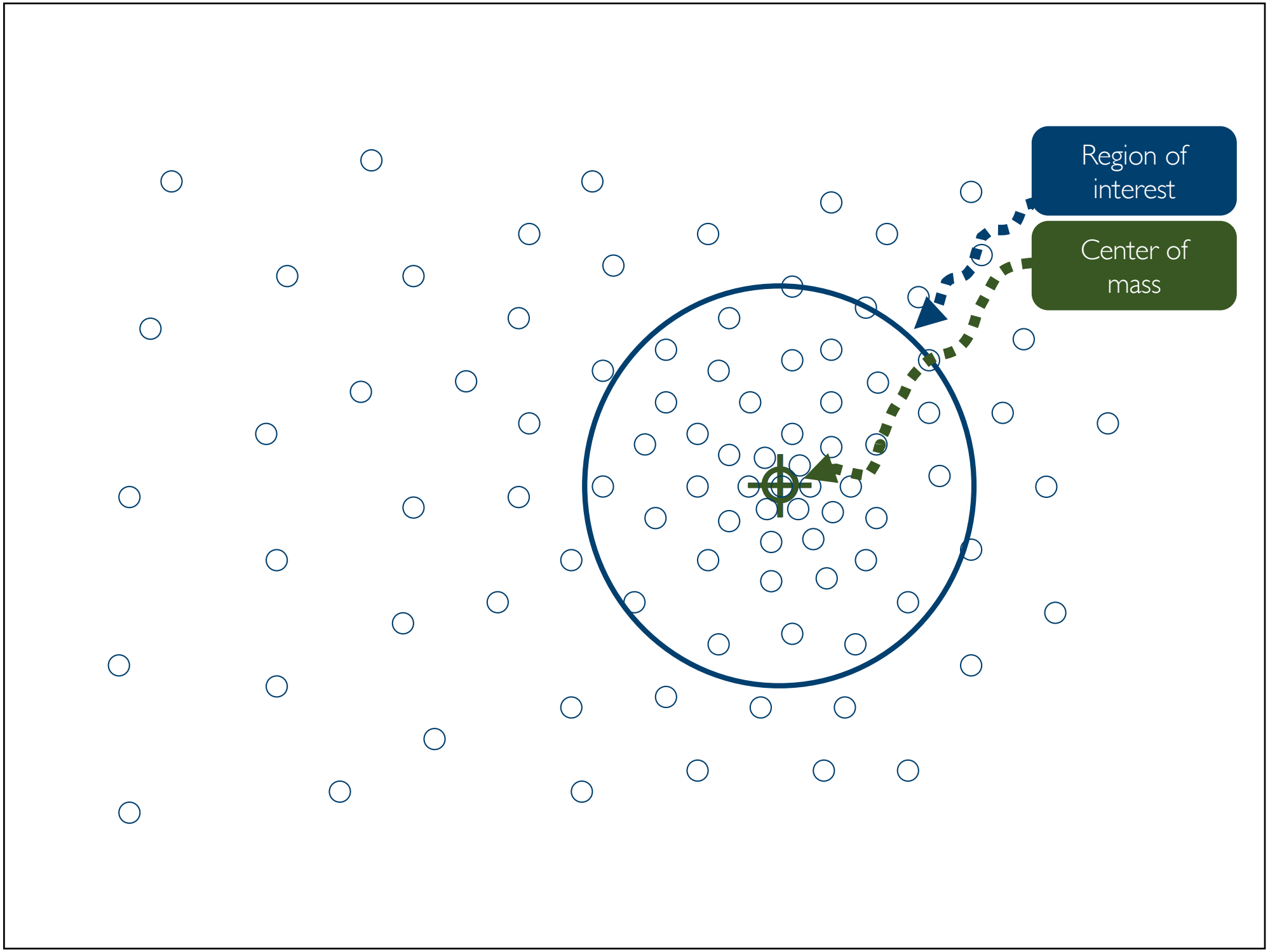


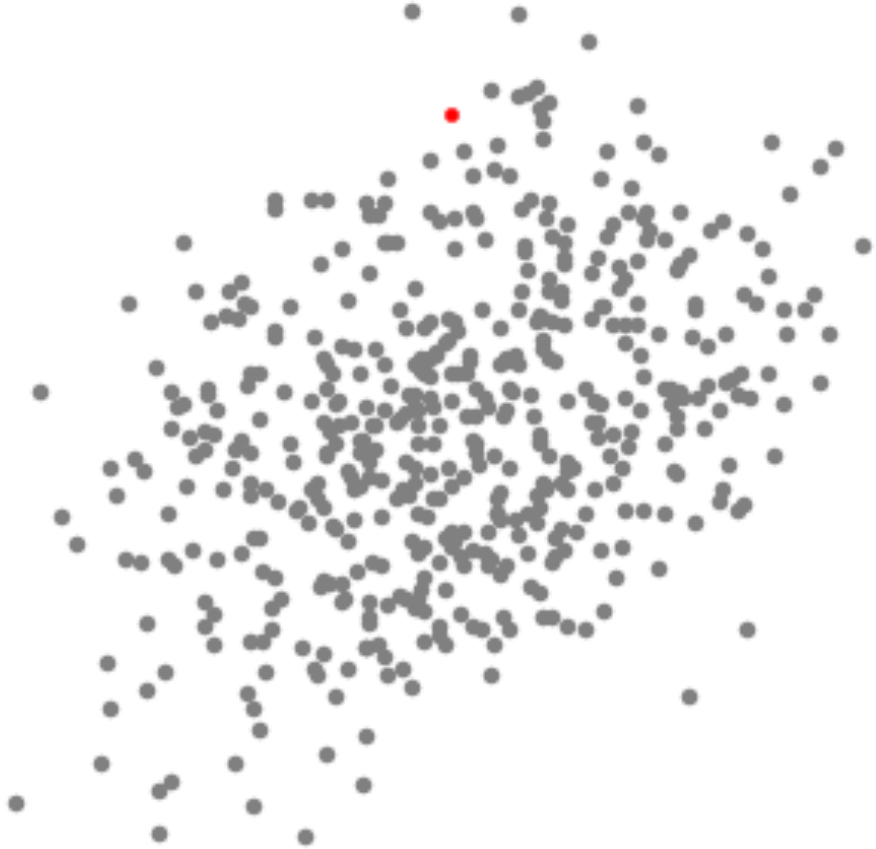
The objective is to find the densest region


Region of interest
Center of mass
Mean Shift vector


Region of interest
Center of mass
Mean Shift vector


Region of interest
Center of mass
Mean Shift vector


Region of interest
Center of mass



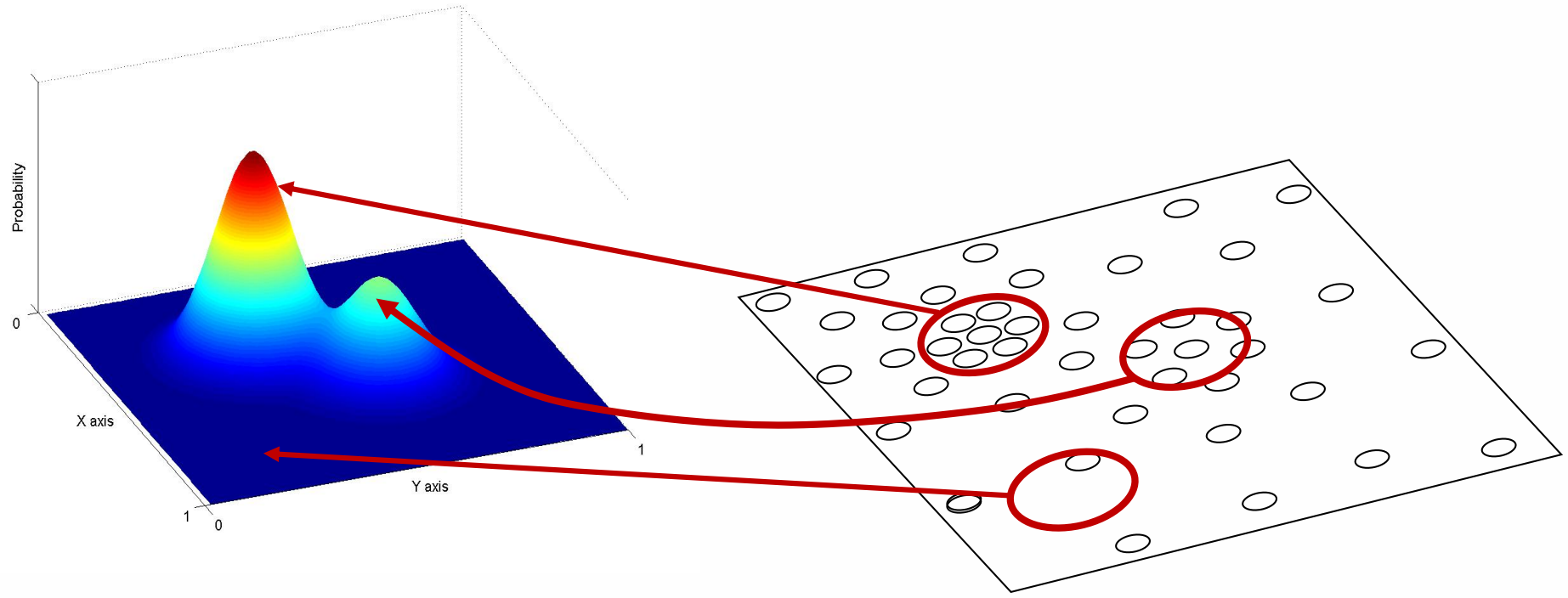
Clustering with Scikit with GIFs
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Mean Shift Algorithm Pseudocode

1. Choose a search window size (the bandwidth)
2. Choose the initial location of the search window
3. Compute the mean location in the search window
4. Center the search window at the location computed in Step 3
5. Repeat Steps 3 and 4 until convergence

Non-Parametric Density Estimation

- We assume that the data points are sampled from an underlying probability density function (PDF)



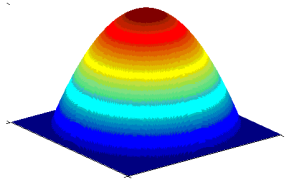
Assumed Underlying PDF

Real Data Samples

Kernel Density Estimation - Examples

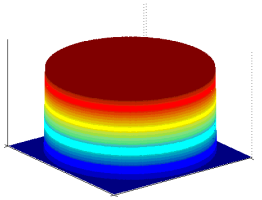
- Epanechnikov Kernel

$$K_E(x) = \begin{cases} c(1 - ||x||^2) & ||x|| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



- Uniform Kernel

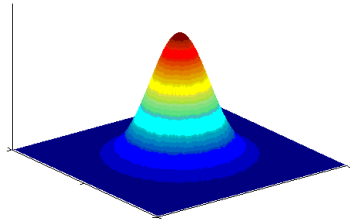
$$K_U(x) = \begin{cases} c & ||x|| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



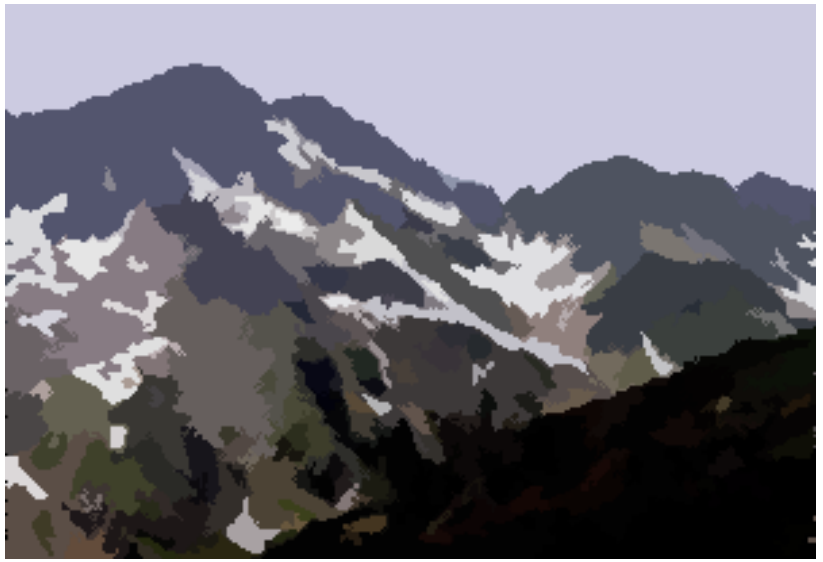
- Normal/Gaussian Kernel

$$K_N(x) = c \cdot e^{-0.5||x||^2}$$

$$K_G(\frac{x - x_i}{h}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}$$



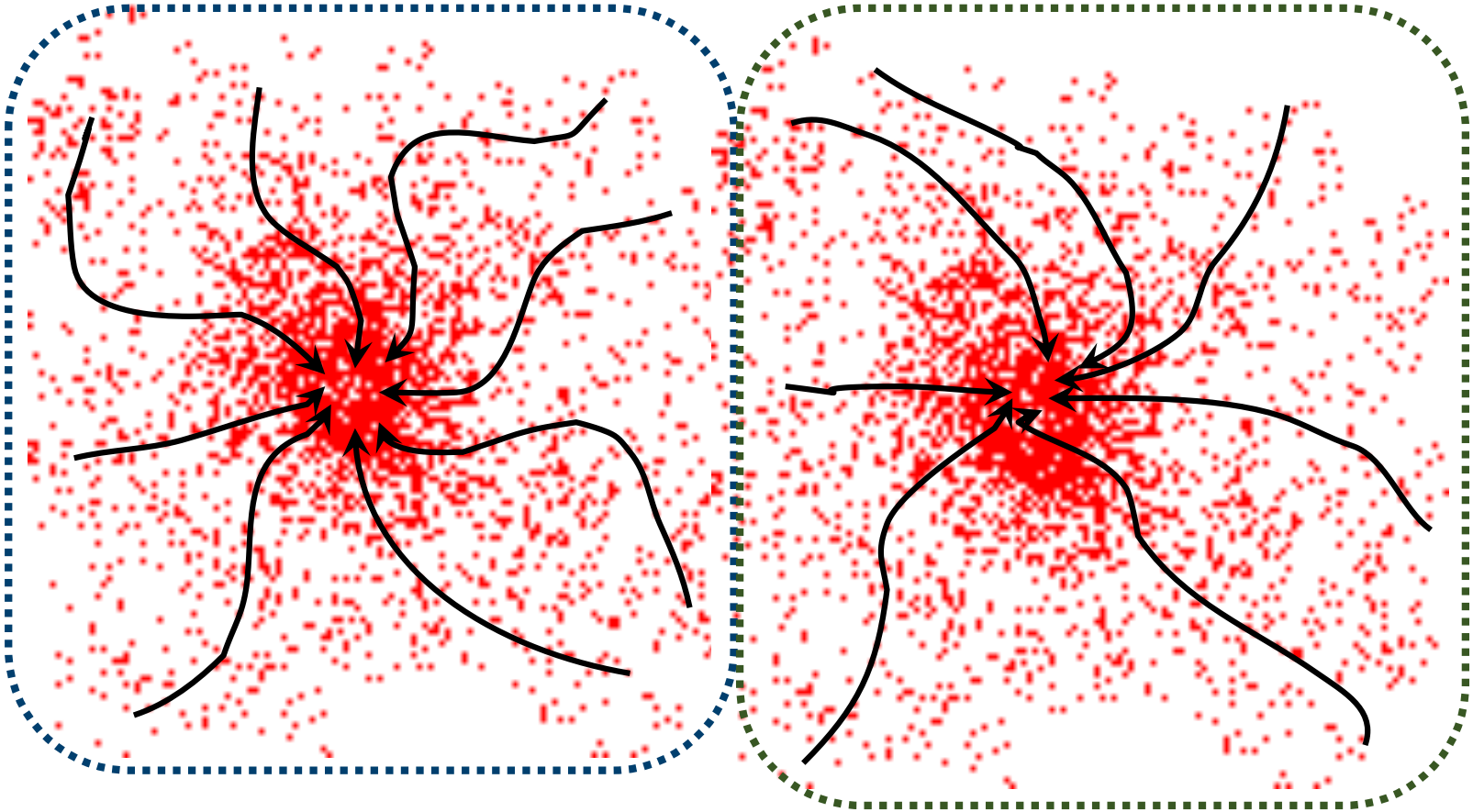
Mean Shift Segmentation

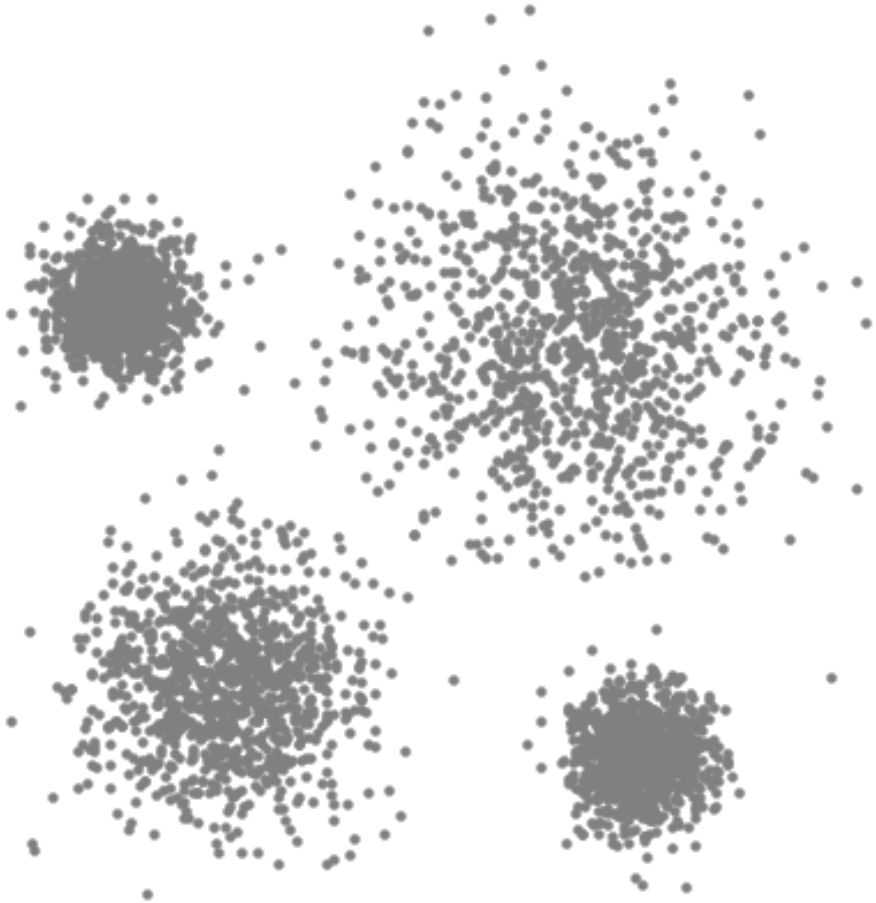


D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Mean Shift Clustering

Trajectories that lead to the same mode (attraction basin) should be in the same cluster





Clustering with Scikit with GIFs
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Mean Shift Clustering - Pseudocode

1. Choose kernel and bandwidth
2. For each point
   a. Center a window on that point
   b. Compute the mean of the data in the search window
   c. Center the search window at the new mean location
   d. Repeat (b,c) until convergence
3. Assign points that lead to nearby modes to the same cluster

K-Means & Mean Shift Clustering

- Number of clusters and shape
  - k-means assumes that the number of clusters is already known and the clusters are shaped spherically (or elliptically)
  - Mean shift does not assume anything about number of clusters. Instead, the number of modes give the number of clusters
  - Also, since mean shift is based on density estimation, it can handle arbitrarily shaped clusters.
- Initialization and outliers
  - k-means is sensitive to initialization
  - Mean shift is robust to initializations. Typically, mean shift is run for each point or sometimes points are selected uniformly from the feature space
  - Similarly, k-means is sensitive to outliers while Mean Shift is less sensitive.
- Time complexity
  - k-means is fast and has a time complexity O(knT) where k is the number of clusters, n is the number of points and T is the number of iterations.
  - Classic mean shift is computationally expensive with a time complexity $O(Tn^2)$.

Mean Shift Clustering

- Strength
  - Does not assume any prior shape on data clusters
  - Can handle arbitrary feature spaces
  - Only the window size h to choose
  - The window size h (or bandwidth) has a physical meaning
- Weaknesses
  - The window size h (or bandwidth) selection is not trivial
  - Inappropriate window size can cause modes to be merged, or generate additional "shallow" modes
  - Adaptive window size can help
  - Not suitable for high-dimensional features

Computational Complexity

- MeanShift clustering runs at $O(Tn^2)$, k-Means at $O(knT)$ where T is the number of iterations and n is the number of data points

- The functions to estimate the bandwidth scale particularly badly (see sklearn estimate_bandwidth documentation)

Expectation Maximization

Expectation-Maximization (EM) Clustering

- k-means assigns each point to only one cluster (hard assignment)
- The approach can be extended to consider soft assignment of points to clusters, so that each point has a probability of belonging to each cluster
- We assume that each cluster C_i is characterized by a multivariate normal distribution and thus identified by
  - The mean vector μ_i
  - The covariance matrix Σ_i
- A clustering is identified by a vector of parameter θ defined as

$$\theta = \{\mu_i \; \Sigma_i \; P(C_i)\}$$

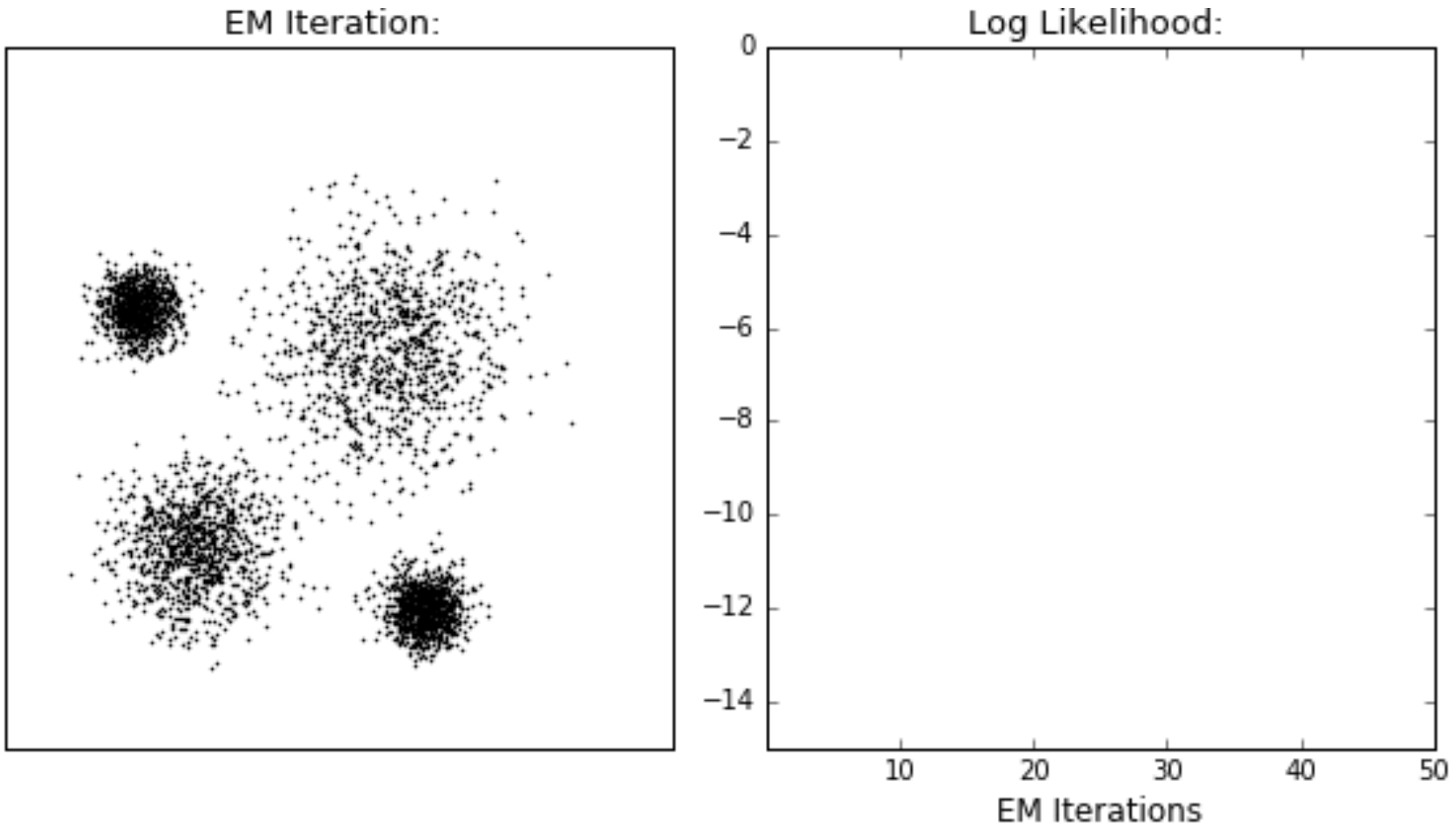
where $P(C_i)$ are the prior probability of all the clusters C_i which sum up to one

Expectation-Maximization (EM) Clustering

- The goal of maximum likelihood estimation (MLE) is to choose the parameters θ that maximize the likelihood, that is

$$\theta^{\star} = \arg\max_{\theta} P(D|\theta)$$

- General idea
  - Starts with an initial estimate of the parameter vector
  - Iteratively rescores the patterns against the mixture density produced by the parameter vector
  - The rescored patterns are used to update the parameter updates
  - Patterns belonging to the same cluster, if they are placed by their scores in a particular component



Clustering with Scikit with GIFs
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Example in One Dimension

EM in One Dimension - Initialization

- Consider a dataset consisting of a single attribute $X = \{x_1, \ldots, x_n\}$
- Clusters will be represented using univariate normal,

$$f_i(x) = f(x|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\}$$

- Each cluster will be represented by the parameters $\{\boldsymbol{\mu_i}\ \boldsymbol{\sigma_i}^2\ P(C_i)\}$
- We initialize the parameters as follows, the mean $\boldsymbol{\mu_i}$ is selected uniformly at random, $\boldsymbol{\sigma_i}^2$ is initialized to 1, the cluster $P(C_i)$ probability with 1/k (where k is the number of clusters).

EM in One Dimension – Expectation Step

- For each cluster, 1 … k, we can use the current estimate of the parameters $\{\boldsymbol{\mu}_i\ \boldsymbol{\sigma}_i^2\ P(C_i)\}$ to compute the posterior probabilities,

$$P(C_i|x_j) = \frac{f(x_j|\mu_i, \sigma_i^2) \cdot P(C_i)}{\sum_{a=1}^{k} f(x_j|\mu_a, \sigma_a^2) \cdot P(C_a)}$$

- We denote with
  - $w_{i,j}$ the values $P(C_i|x_j)$
  - w_i the weight vector for cluster C_i over all the n points, that is, $(w_{i,1} \ \ldots \ w_{i,n})^T$

EM in One Dimension – Maximization Step (1)

- Given all the posterior probability values $w_{i,j}$ the maximization step computes the maximum likelihood estimates of the cluster parameters $\{\boldsymbol{\mu}_i\ \boldsymbol{\sigma}_i^2\ P(C_i)\}$

- The means are updated as,

$$\mu_i = \frac{\sum_{j=1}^{n} w_{ij} \cdot x_j}{\sum_{j=1}^{n} w_{ij}}$$

which can be rewritten as

$$\mu_i = \frac{\mathbf{w}_i^T X}{\mathbf{w}_i^T \mathbf{1}}$$

EM in One Dimension – Maximization Step (2)

- Similarly, for $\boldsymbol{\sigma}_i^2$

$$\sigma_i^2 = \frac{\sum_{j=1}^{n} w_{ij}(x_j - \mu_i)^2}{\sum_{j=1}^{n} w_{ij}} \quad \text{or} \quad \sigma_i^2 = \frac{\mathbf{w}_i^T \mathbf{Z}_i^s}{\mathbf{w}_i^T \mathbf{1}}$$

where $Z_i = (x_1 - \boldsymbol{\mu}_i, \ldots, x_n - \boldsymbol{\mu}_i)^T$

- Finally,

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{a=1}^{k}\sum_{j=1}^{n} w_{aj}} = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{j=1}^{n} 1} = \frac{\sum_{j=1}^{n} w_{ij}}{n}$$

- The expectation and maximization steps are repeated until convergence (e.g. until the means change a little between steps)

Example with one variable (Example 13.4 from the textbook)

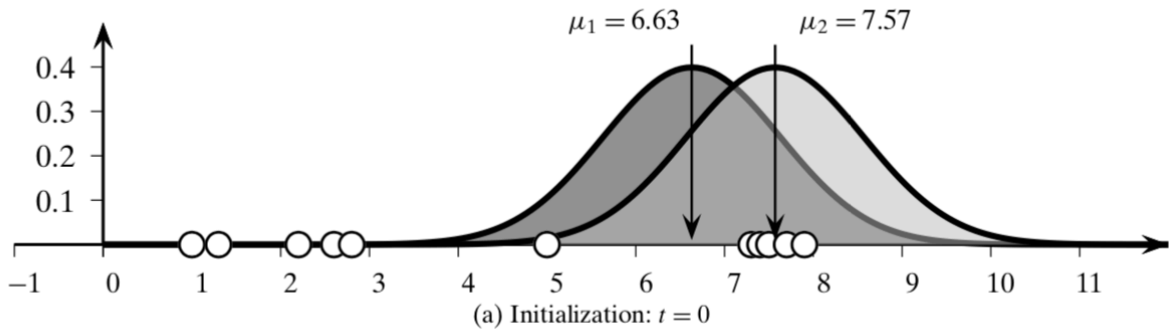
Example 13.4 from the Textbook (1)

- Data Points

$$x_1 = 1.0 \quad x_2 = 1.3 \quad x_3 = 2.2 \quad x_4 = 2.6 \quad x_5 = 2.8$$
$$x_6 = 5.0 \quad x_7 = 7.3 \quad x_8 = 7.4 \quad x_9 = 7.5 \quad x_{10} = 7.7 \quad x_{11} = 7.9$$

- Initialization of the parameters

$$\mu_1 = 6.63 \quad \sigma_1^2 = 1 \quad P(C_2) = 0.5$$
$$\mu_2 = 7.57 \quad \sigma_2^2 = 1 \quad P(C_2) = 0.5$$

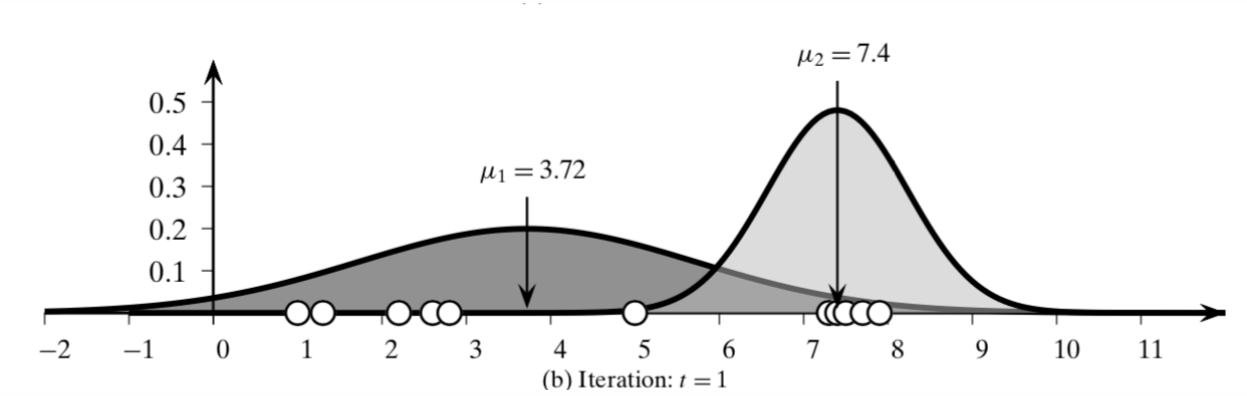


(a) Initialization: $t = 0$

Example 13.4 from the Textbook (2)

- After one expectation-maximitation iteraton the parameters are

$\mu_1 = 3.72$	$\sigma_1^2 = 6.13$	$P(C_1) = 0.71$
$\mu_2 = 7.4$	$\sigma_2^2 = 0.69$	$P(C_2) = 0.29$

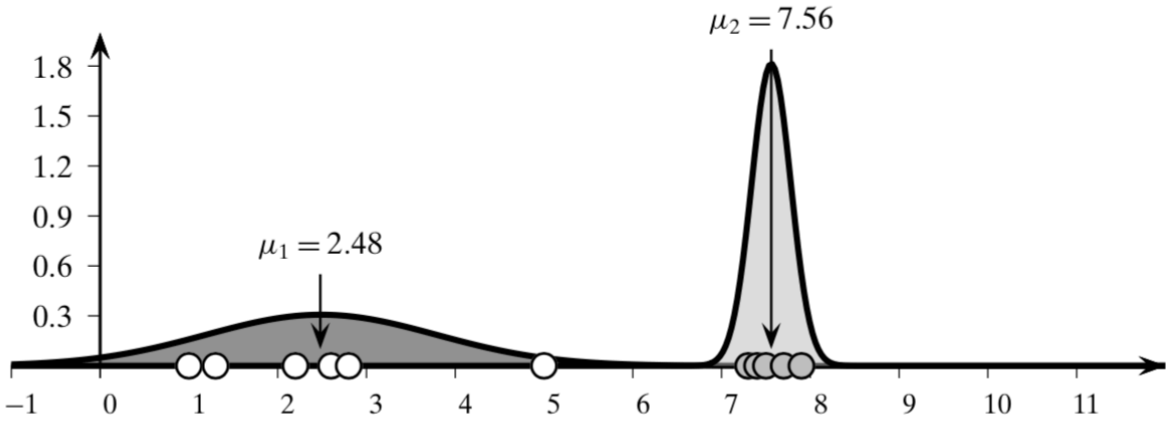


(b) Iteration: $t = 1$

Example 13.4 from the Textbook (3)

- The procedure converges after five iterations to

$$\mu_1 = 2.48 \qquad \sigma_1^2 = 1.69 \qquad P(C_1) = 0.55$$
$$\mu_2 = 7.56 \qquad \sigma_2^2 = 0.05 \qquad P(C_2) = 0.45$$



- EM Clustering returns probabilities, it does not assign points to clusters.
- In this example, we assigned elements to the cluster with the highest posterior probability

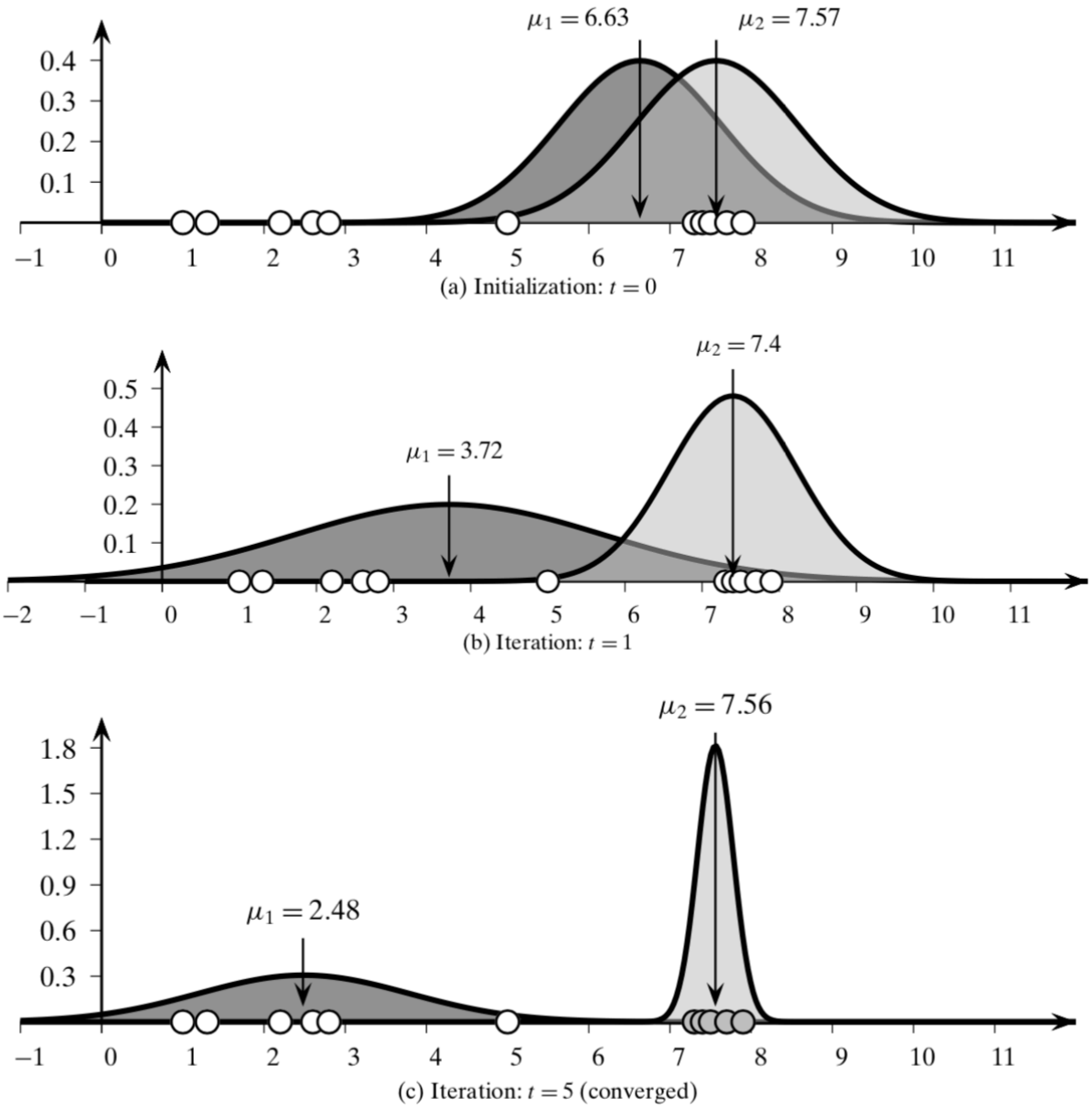


Figure 13.4. EM in one dimension.

Example from Chapter 13 of the textbook

Example with two variables

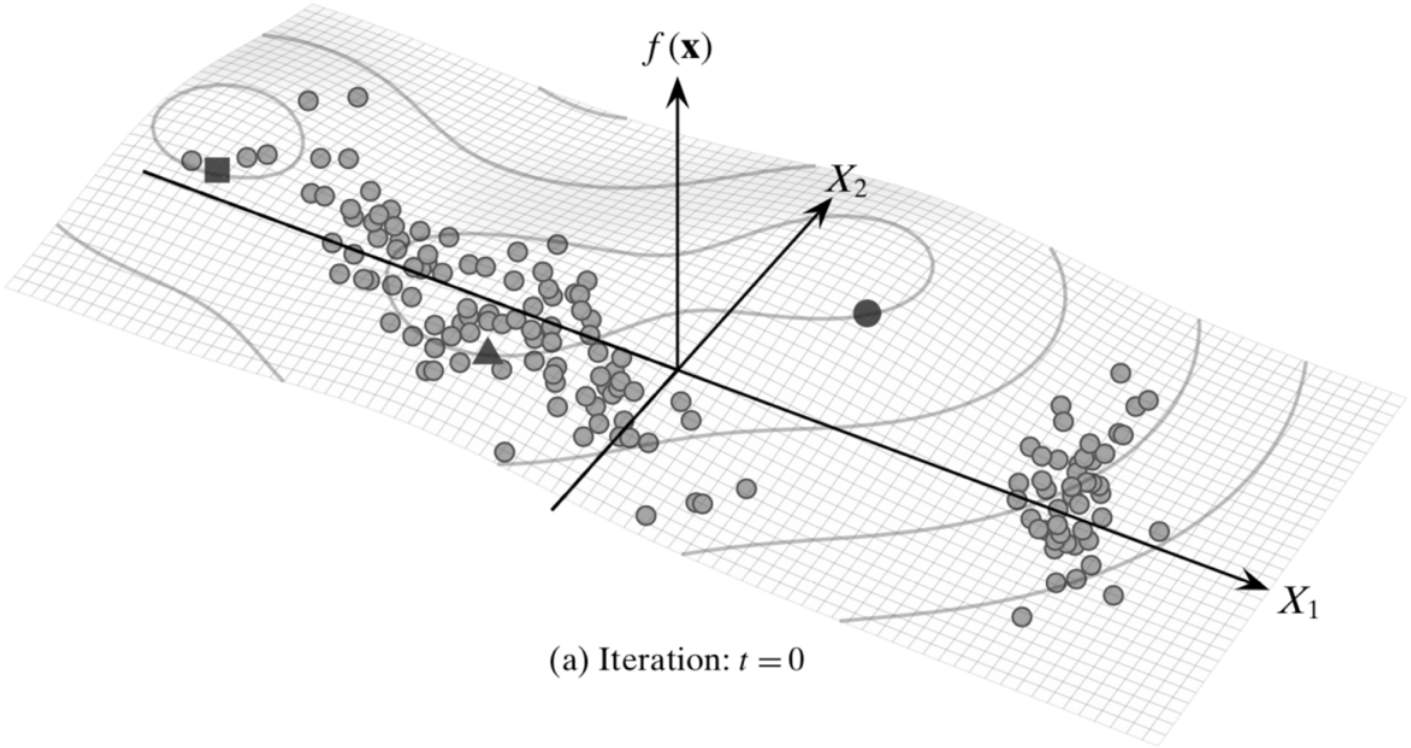


Figure 13.5 from Chapter 13 of the textbook

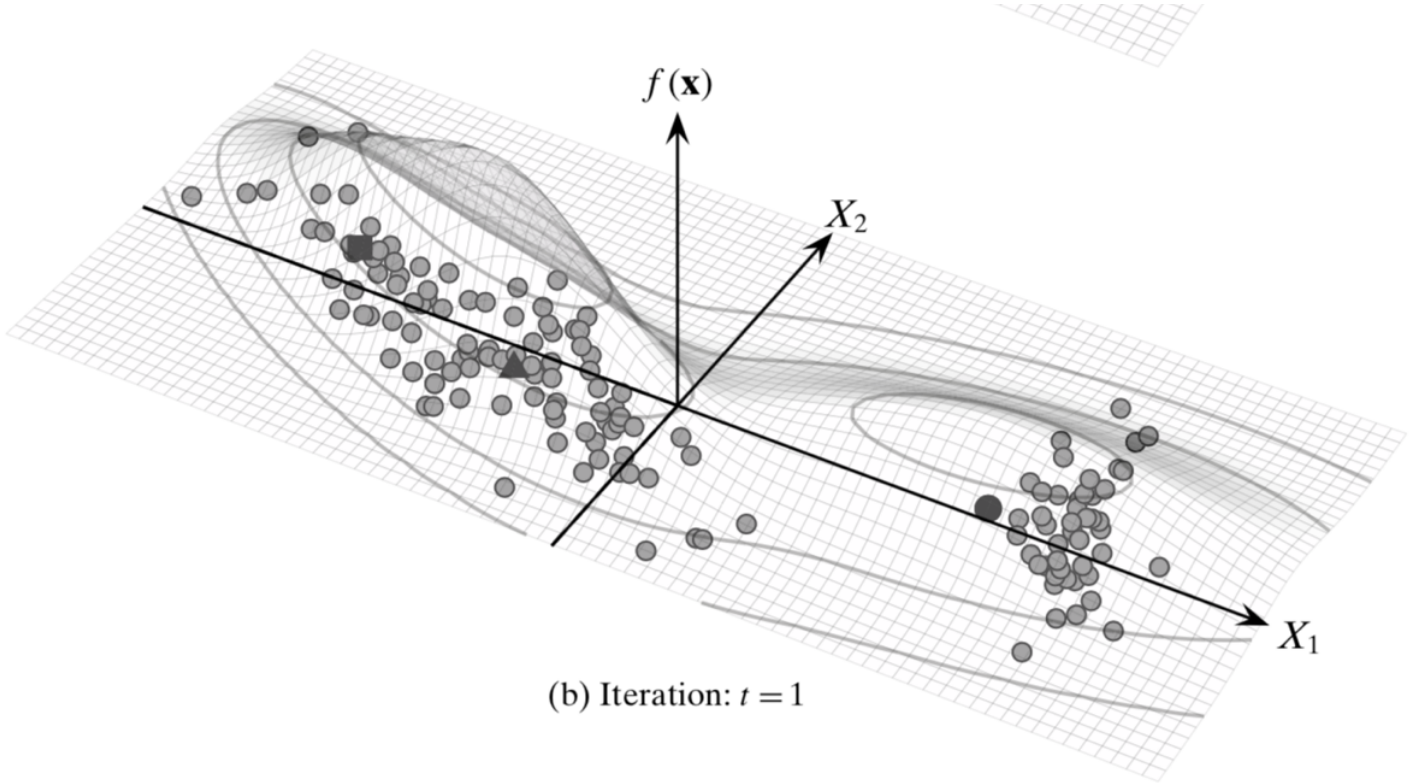


Figure 13.5 from Chapter 13 of the textbook

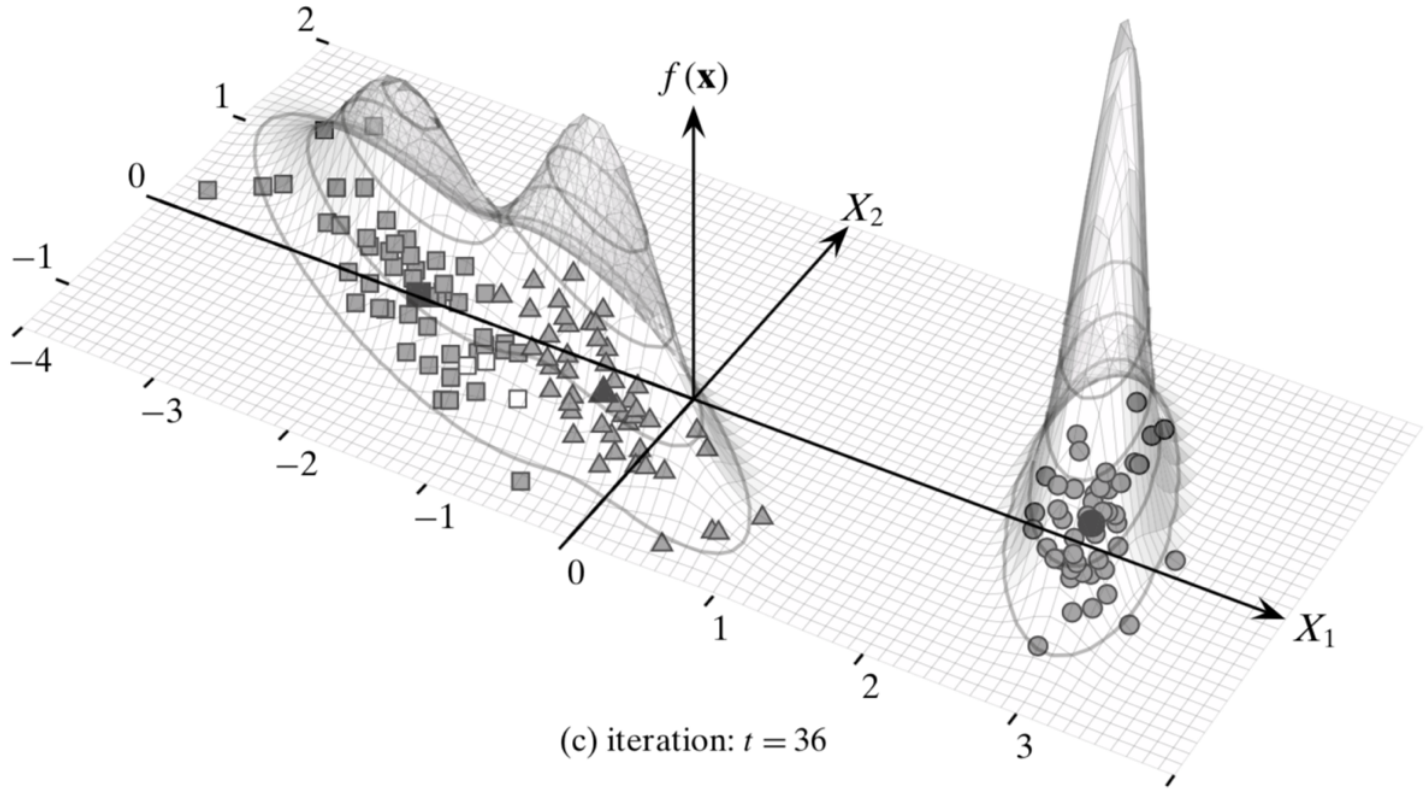


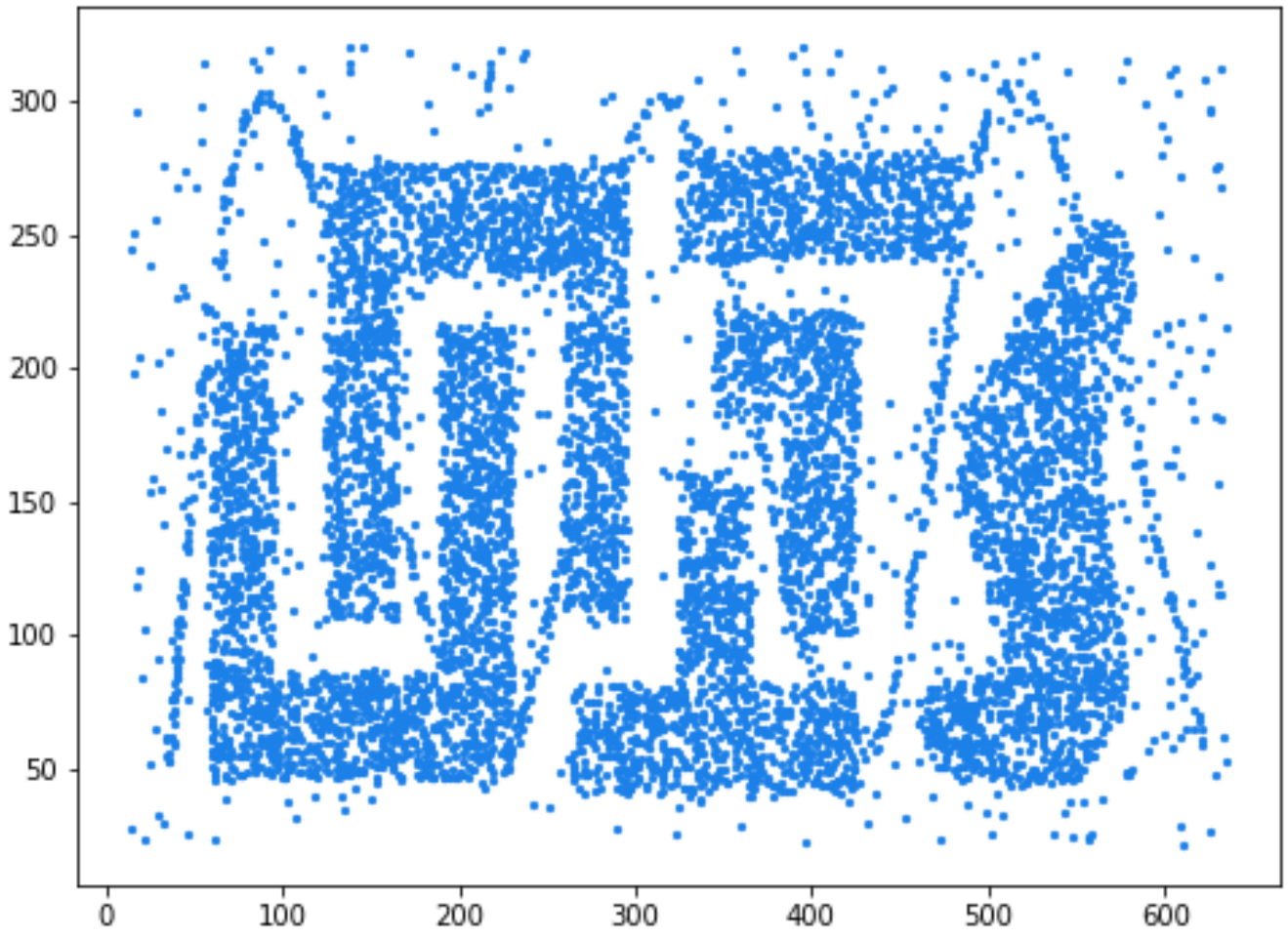
Figure 13.5 from Chapter 13 of the textbook

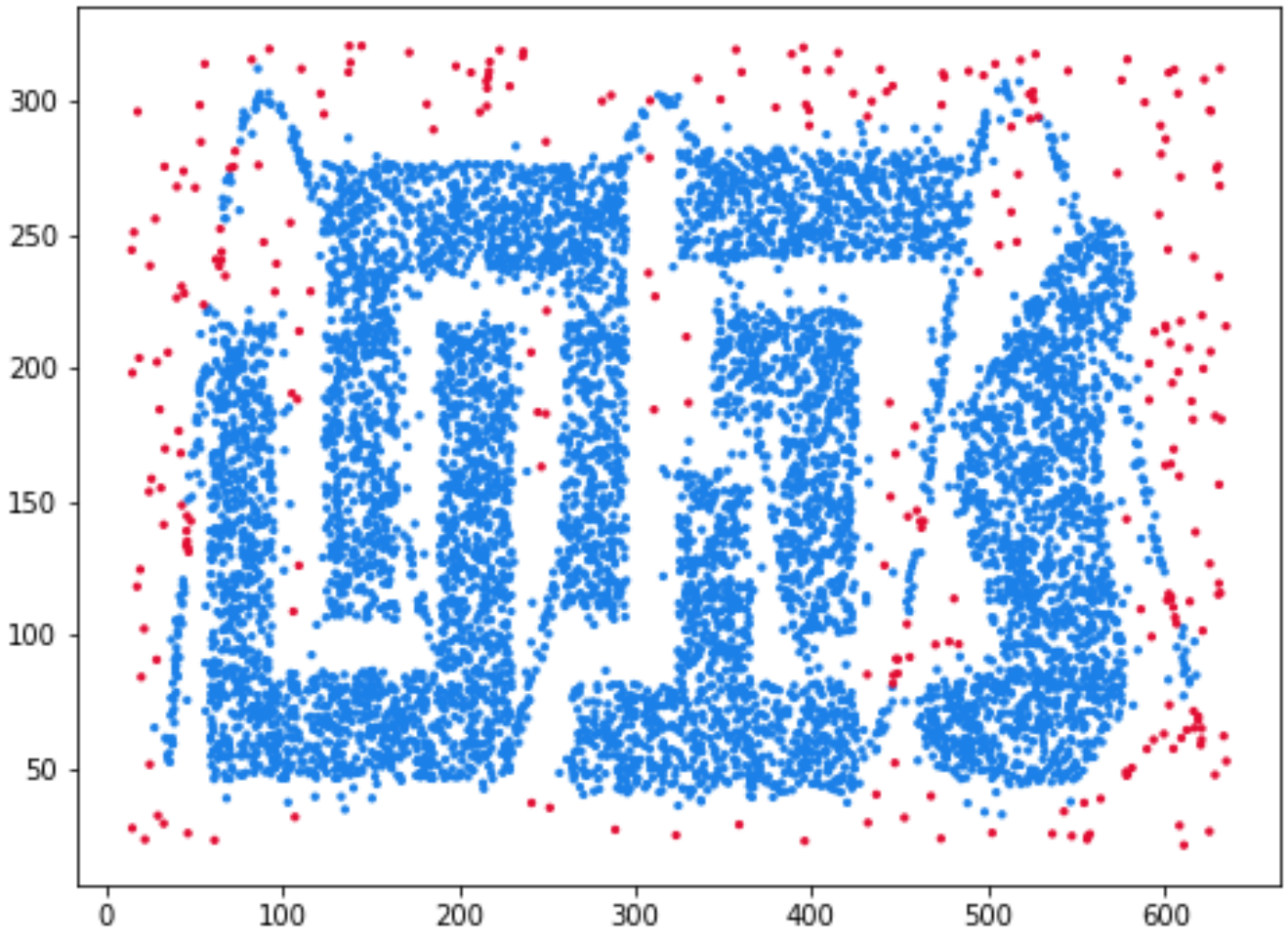
Beyond convex clusters

Representative-based clustering methods are suitable for finding ellipsoid-shaped clusters, or at best convex clusters

For nonconvex clusters, these methods have trouble finding the true clusters

Density-based methods can mine such nonconvex clusters


300
250
200
150
100
50
0
100
200
300
400
500
600


300
250
200
150
100
50
0
100
200
300
400
500
600

What is density-based clustering?

- Clustering based on density (local cluster criterion), such as density-connected points

- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition

DBSCAN: Basic Concepts

- The neighborhood within a radius ε of a given object is called the ε-neighborhood of the object

$$N_\epsilon(x) = \{y | \delta(x, y) \leq \epsilon\}$$

- Core Point
  - If the ε-neighborhood of an object contains at least **minpts** objects, then the object is a core object
- Border Point
  - If its ε-neighborhood does not contain at least **minpts,** but it is inside the neighborhood of a core point
- Noise Point
  - If its ε-neighborhood does not contain at least **minpts** and it does not belong to the neighborhood of a core point

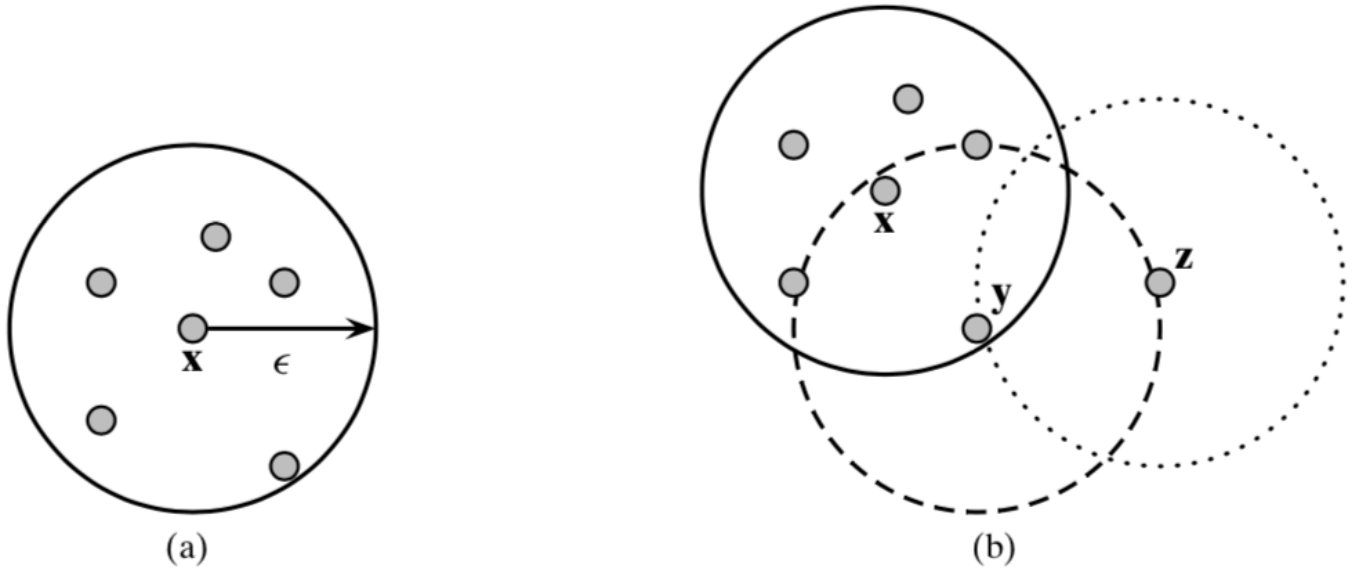


Figure 15.2. (a) Neighborhood of a point. (b) Core, border, and noise points.

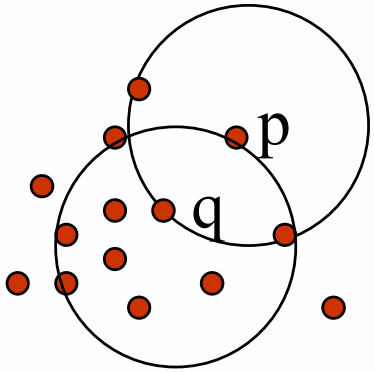
Core, border and noise points when minpts is 6
"Data Mining and Analysis" by M.J. Zaki & W. Meira

DBSCAN: Basic Concepts

- Directly density reachable
  - An object x is directly density-reachable from object y if x is within the ε-neighborhood of y and y is a core object
- Density Reachable
  - An object x is density-reachable from object y if there is a chain of objects $x_1, \ldots, x_n$ where $x_1 = x$ and $x_n = y$ such that x_{i+1} is directly density reachable from x_i
- Density Connected
  - An object p is density-connected to q with respect to ε and MinPts if there is an object o such that both p and q are density reachable from o
- Density-Based Cluster
  - A density-based cluster is defined as a maximal set of density connected points.

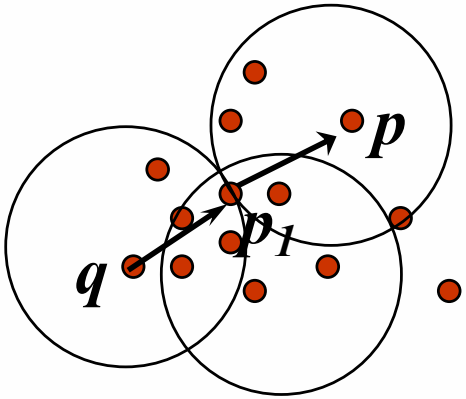
Density-Reachable & Density-Connected

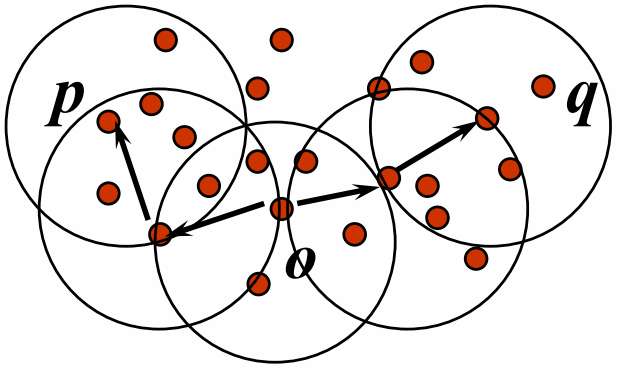
Directly density-reachable



MinPts = 5

Density-reachable





Density-connected

DBSCAN: Basic Concepts

- Density corresponds to have at least minpts points within a specified radius ε
- A border point has fewer than minpt within ε, but is in the neighborhood of a core point
- A noise point is any point that is not a core point nor a border point

ALGORITHM 15.1. Density-based Clustering Algorithm

DBSCAN (D, ϵ, $minpts$):

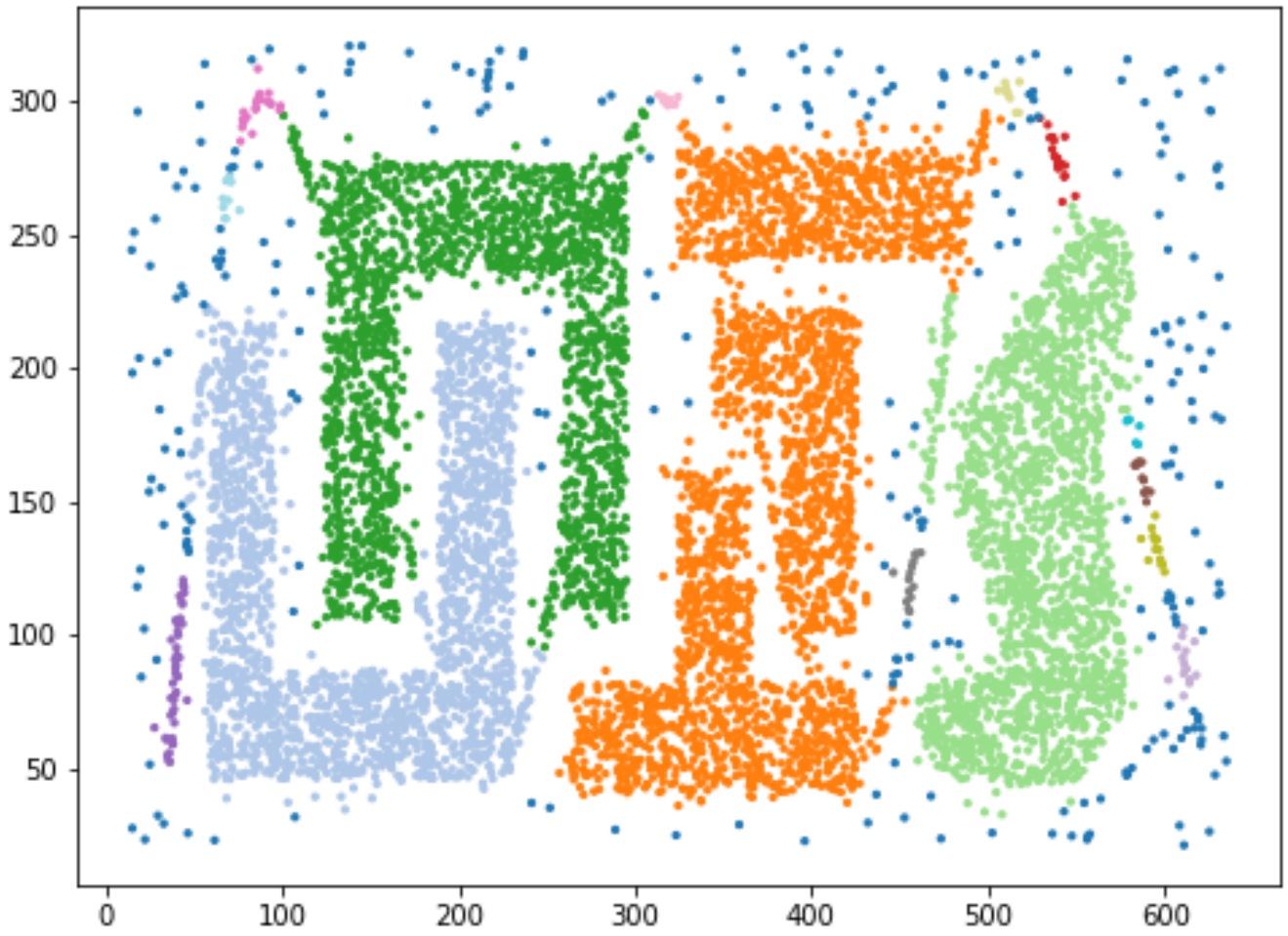
```
1  Core ← ∅
2  foreach x_i ∈ D do // Find the core points
3      Compute N_ε(x_i)
4      id(x_i) ← ∅ // cluster id for x_i
5      if N_ε(x_i) ≥ minpts then Core ← Core ∪ {x_i}
6  k ← 0 // cluster id
7  foreach x_i ∈ Core, such that id(x_i) = ∅ do
8      k ← k + 1
9      id(x_i) ← k // assign x_i to cluster id k
10     DENSITYCONNECTED (x_i, k)
11 C ← {C_i}_{i=1}^k, where C_i ← {x ∈ D | id(x) = i}
12 Noise ← {x ∈ D | id(x) = ∅}
13 Border ← D \ {Core ∪ Noise}
14 return C, Core, Border, Noise
```

DENSITYCONNECTED (x, k):

```
15 foreach y ∈ N_ε(x) do
16     id(y) ← k // assign y to cluster id k
17     if y ∈ Core then DENSITYCONNECTED (y, k)
```

Complexity of DBScan

- DBSCAN needs to compute the ε-neighborhood for each point
- If the dimensionality is not too high this can be done efficiently using a spatial index structure in O(n log n)
- If the dimensionality is high, it takes $O(n^2)$ to compute the neighborhood for each point.
- Once the ε-neighborhood has been computed the algorithm needs only a single pass over all the points to find the density connected clusters.
- The overall complexity of DBSCAN is between O(n log n) and in the worst-case.


300
250
200
150
100
50
0
100
200
300
400
500
600

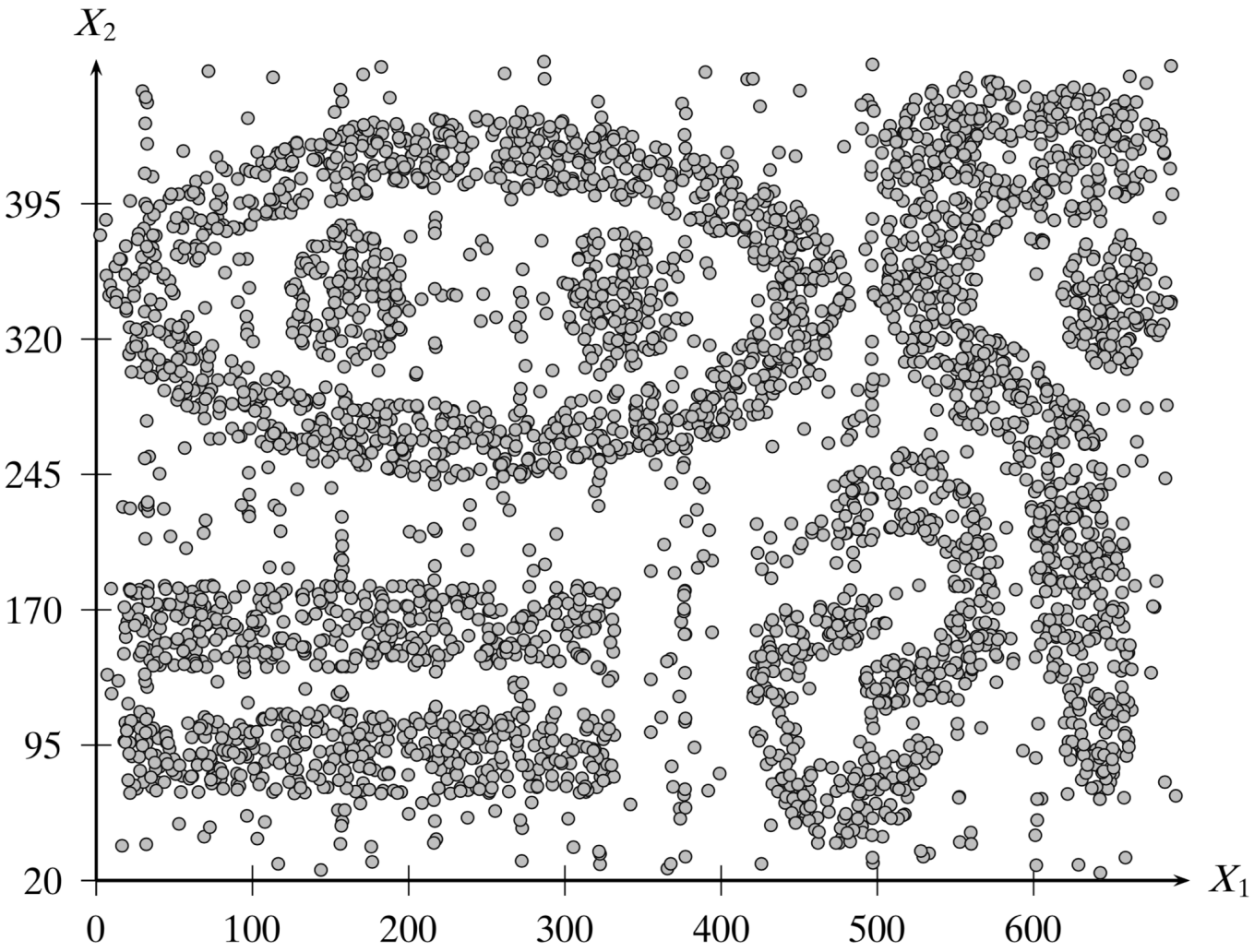


Figure 15.1 from Chapter 15 of the textbook "Data Mining and Analysis" by M.J. Zaki & W. Meira

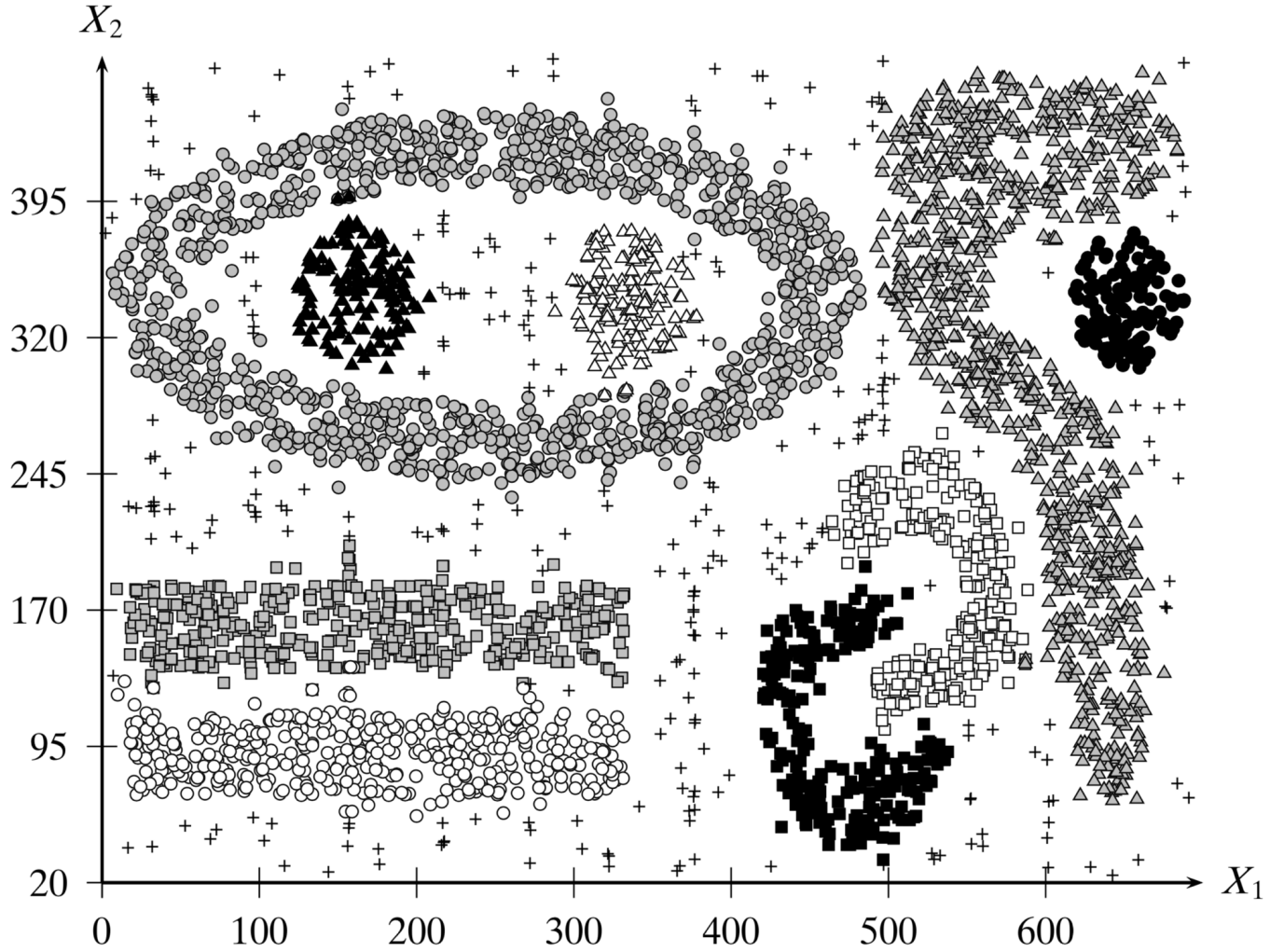


Figure 15.3 from Chapter 15 of the textbook "Data Mining and Analysis" by M.J. Zaki & W. Meira

Let's play with DBSCAN

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBScan fails when applied
to data of varying density

HDBSCAN

HDBSCAN

- Extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters
- It defines the "core distance" of a point as the minimum distance of a point to its k-th nearest neighbor or $core_k(x)$
- Thus, points in high density areas will have a low core distance, while point in low density areas will have a high core distance
- This provides a local inexpensive estimate of density

Mutual Reachability Distance

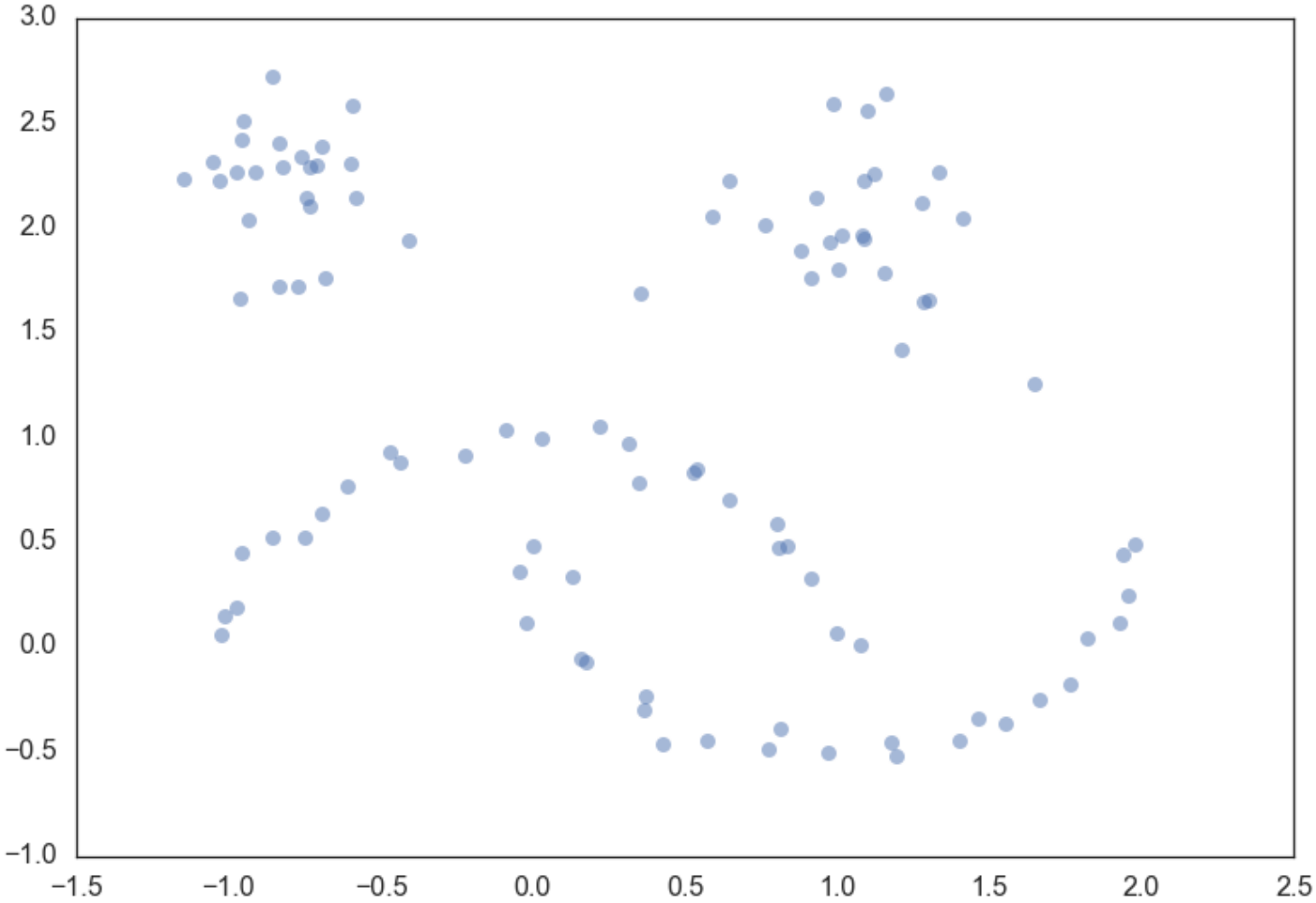
- This new distance metric is defined as,

$$d_{mreach-k}(a,b) = \max\{core_k(a), core_k(b), d(a,b)\}$$

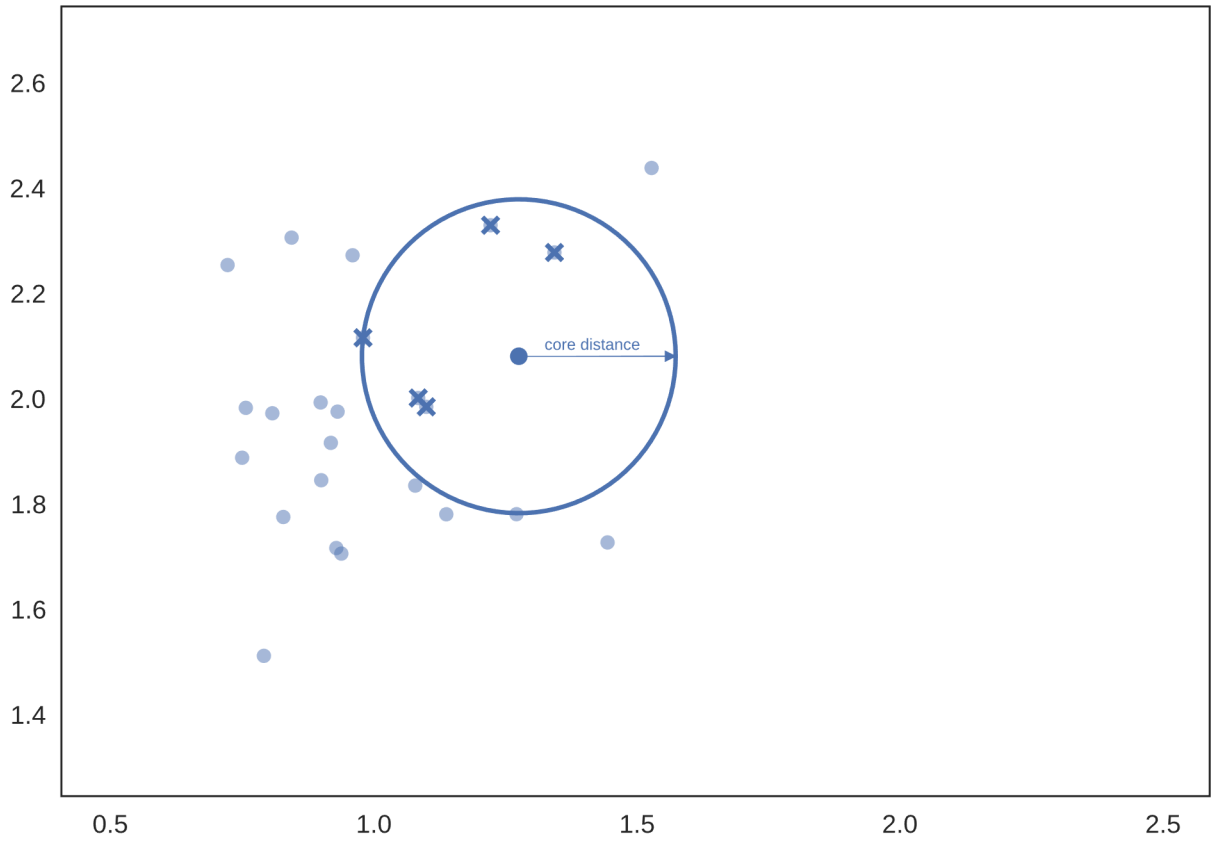
  using the core distance and the original distance metrics d

- This distance keeps dense points with lower core distance at the same distance while pushing away sparser points (at least at their core distance from any other point).

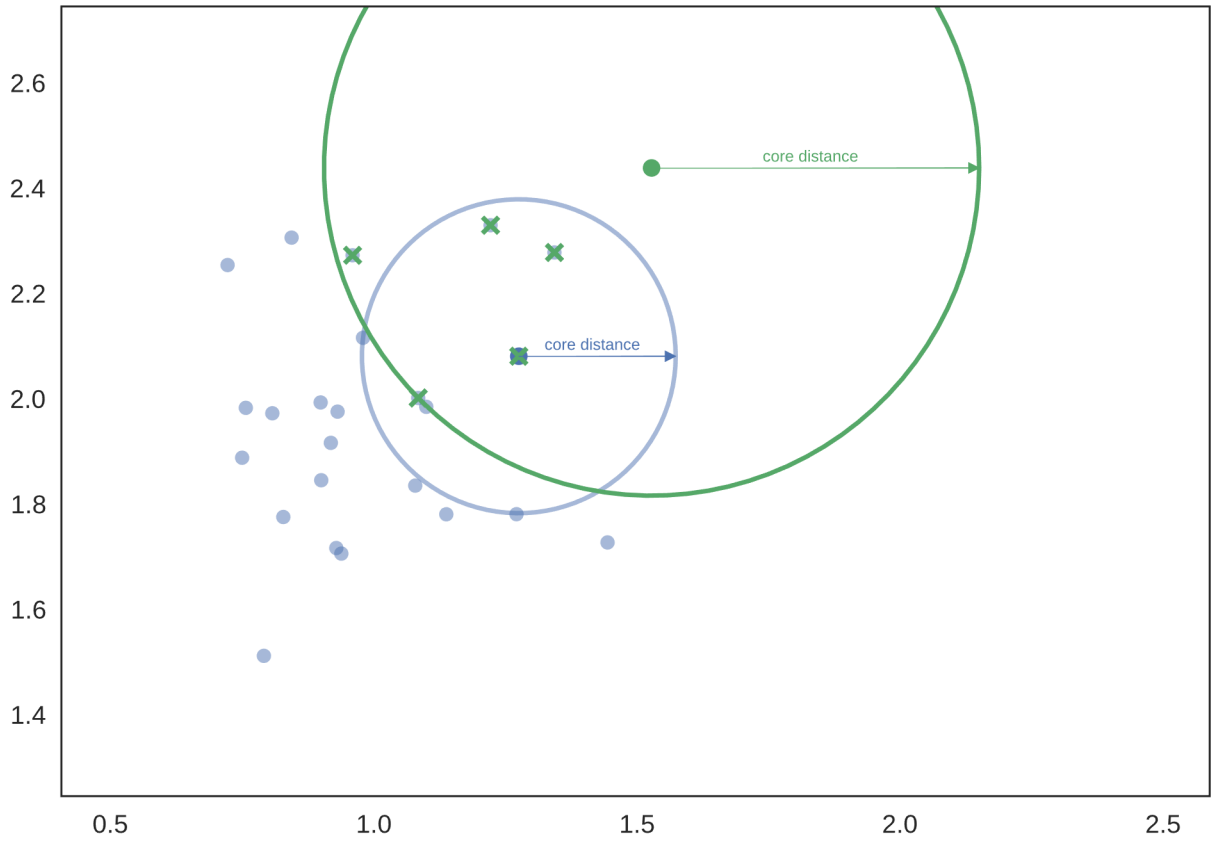
- Note that as with DBSCAN high values of k focus the algorithm toward very dense region



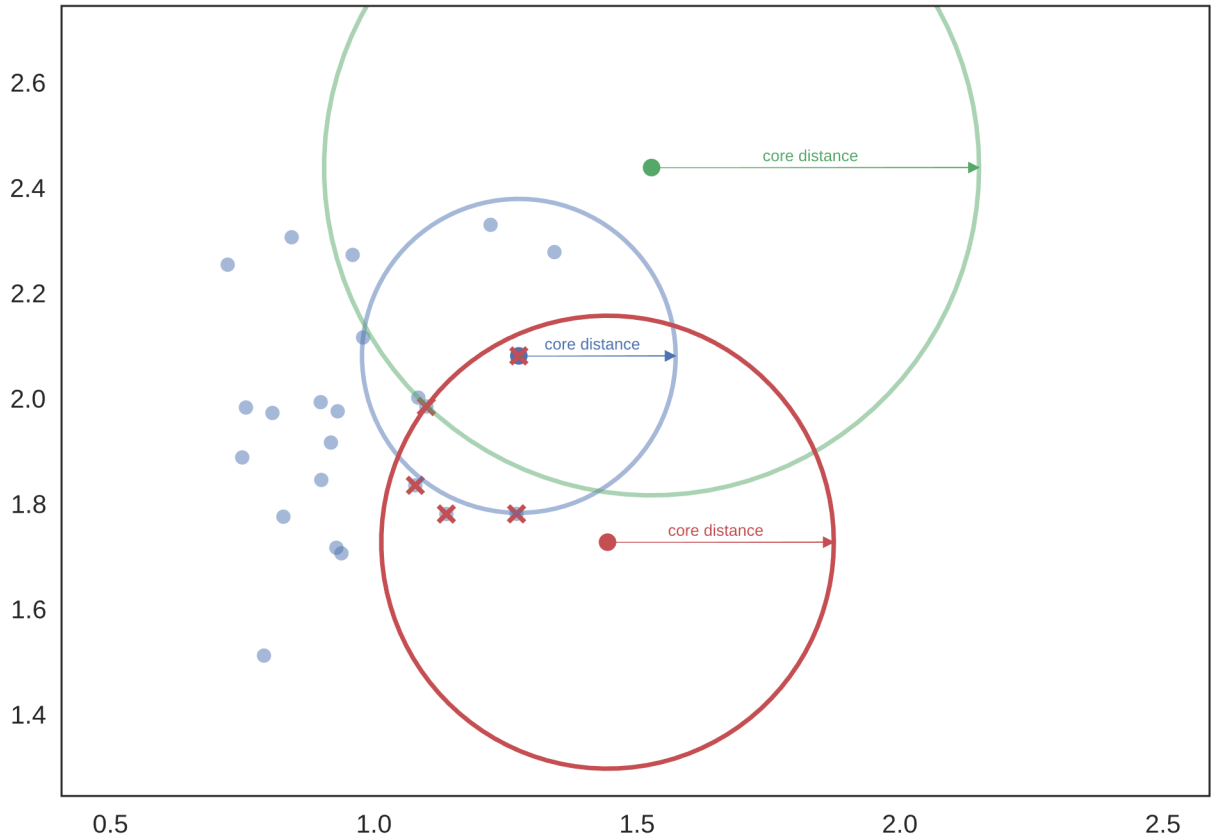
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html



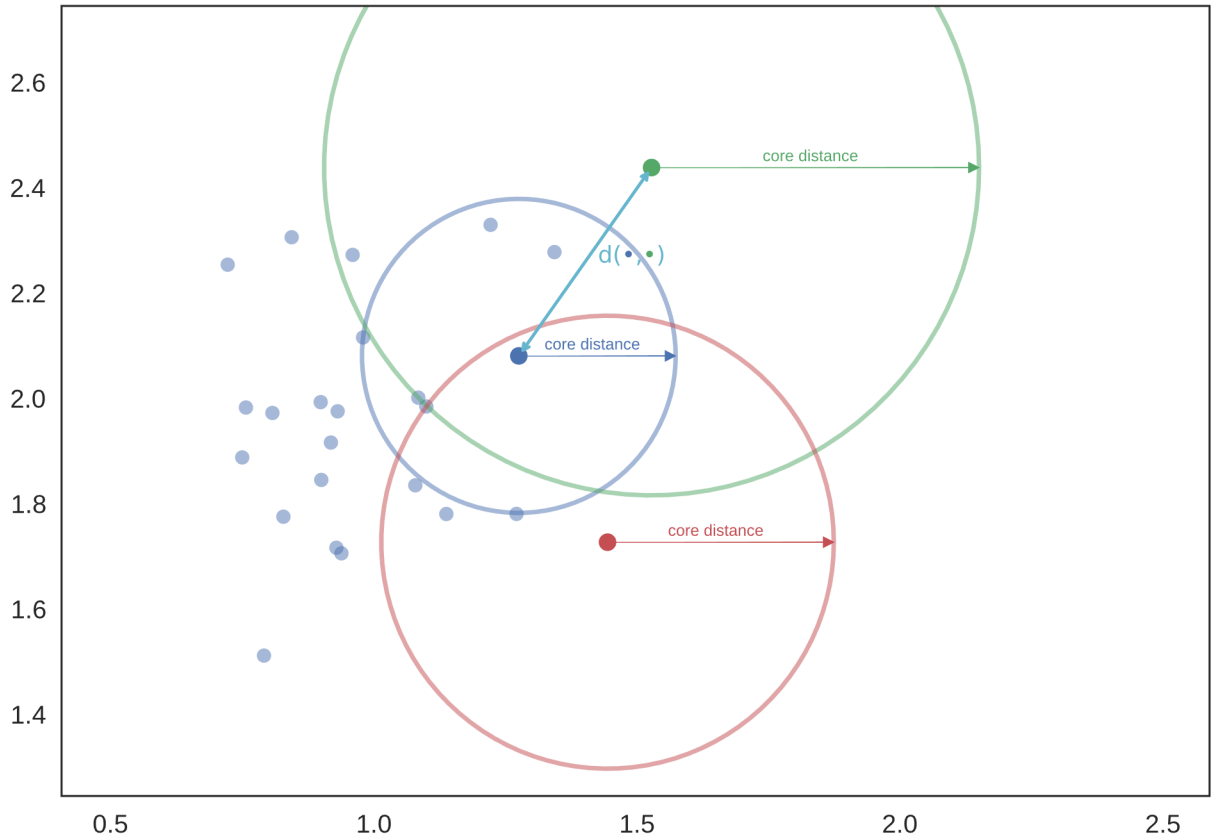
Let's consider a k equal to 5 and plot the core distance for one point.
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html



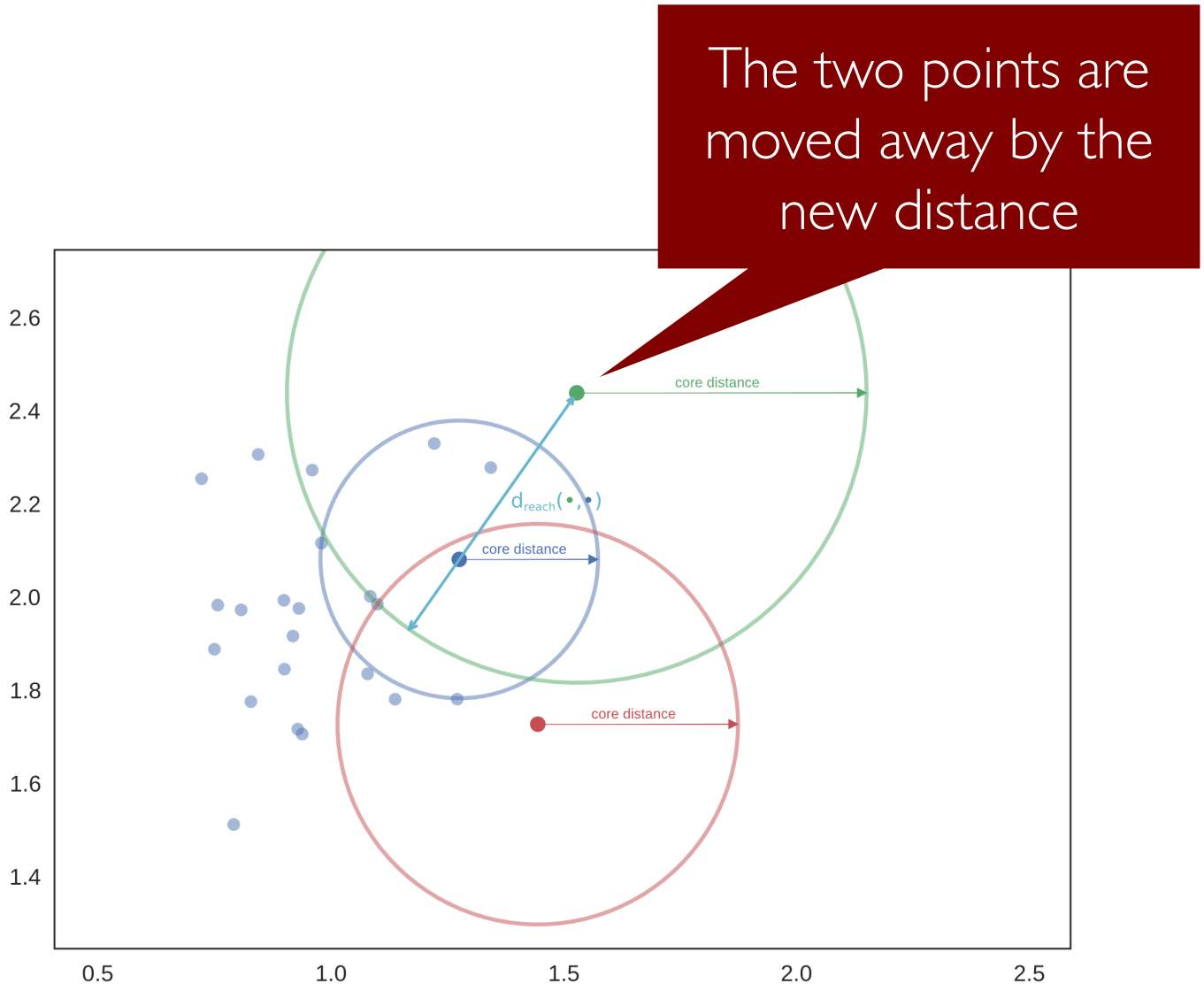
This point is in a lower density area and thus it has a higher core distance.
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html



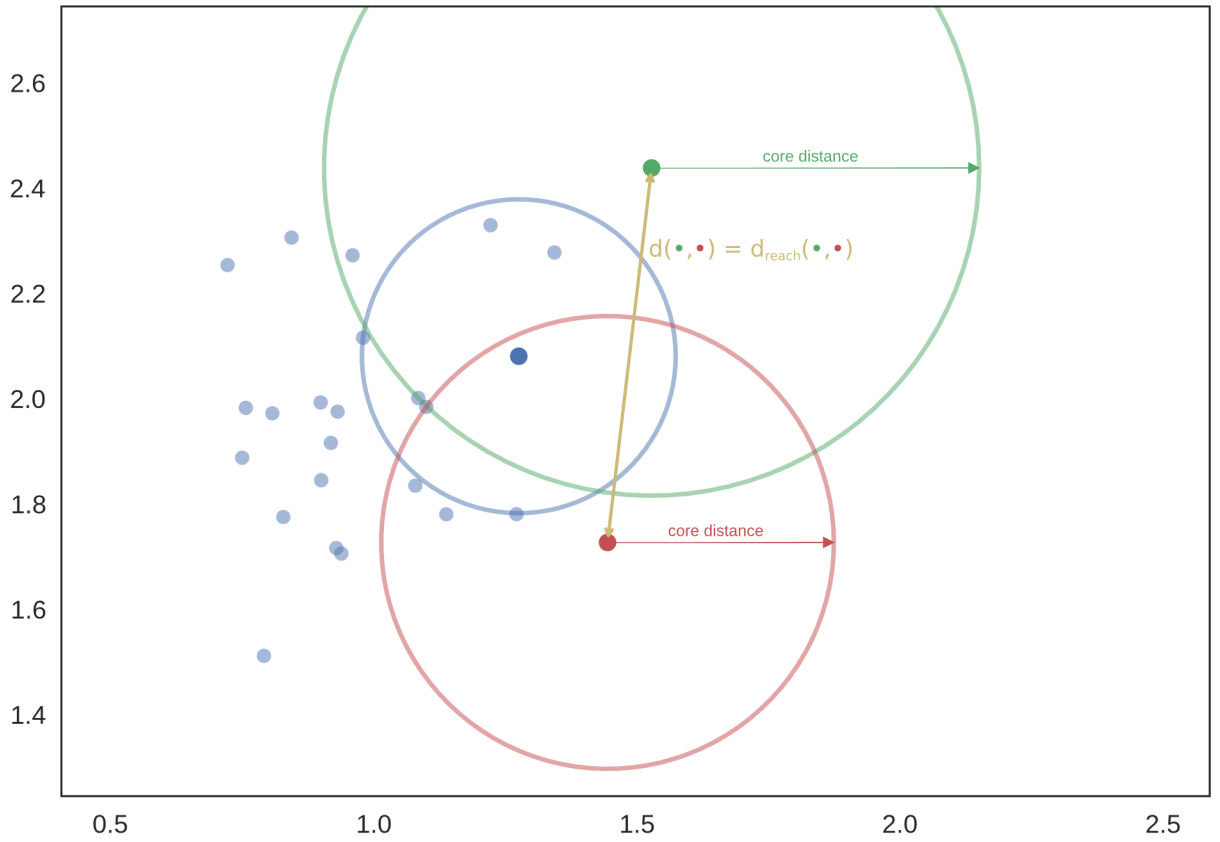
And let's add another point with its core distance
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html



To compute the mutual reachability distance between blue and green points we draw d(.,.) which crosses the blue circle but not the green one. So the green core distance is the largest value.



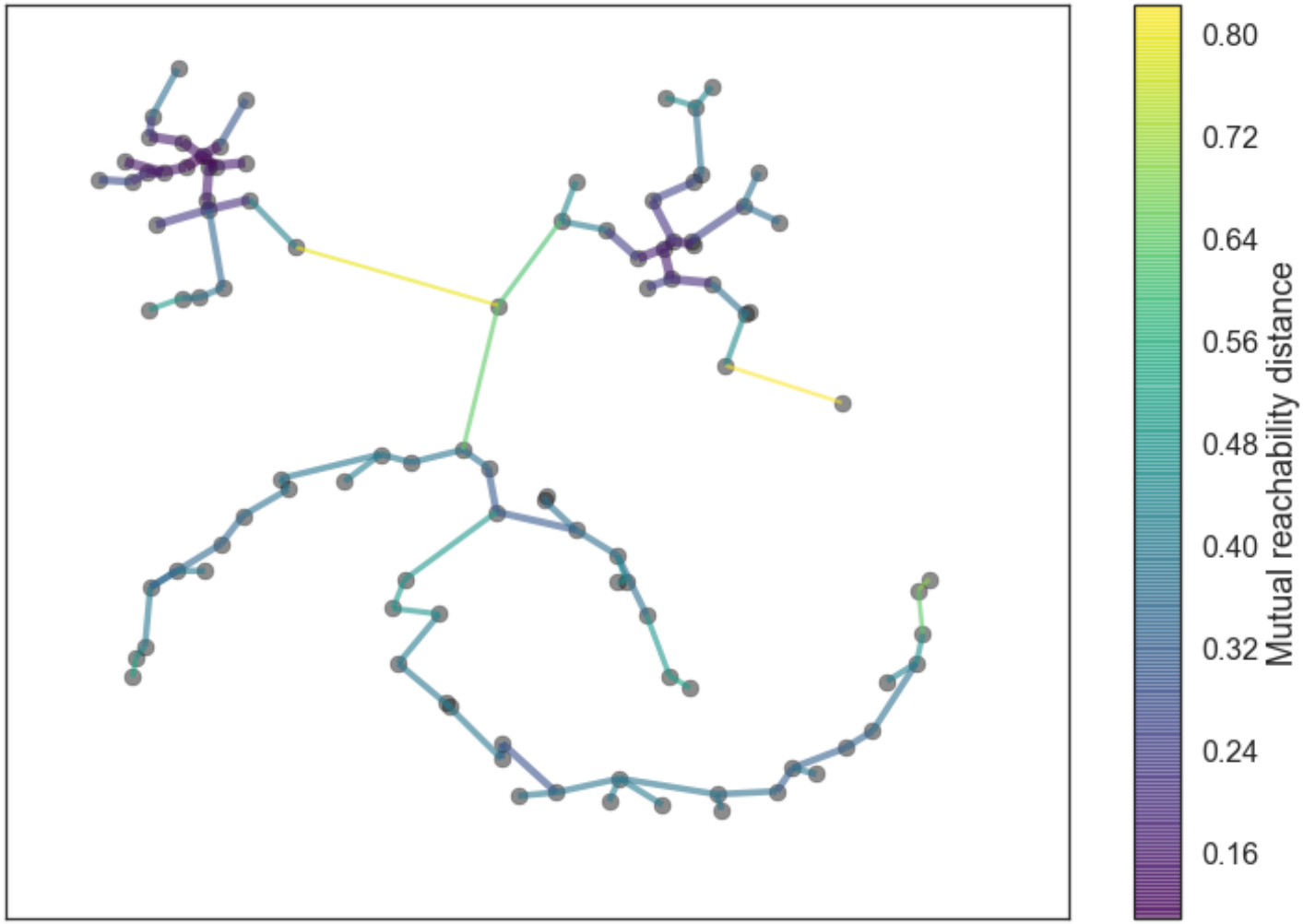
To compute the mutual reachability distance between blue and green points we draw d(.,.) which crosses the blue circle but not the green one. So the green core distance is the largest value.



However, the mutal reachability distance between red and green is the original distance d(. , .) which crosses both the red and green circle.

Next, Build the Minimum Spanning Tree using Prim's Algorithm

- Consider the data as a weighted graph
- Data points are vertices
- An edge between any two points with weight equal to the mutual reachability distance of those points
- Build the spanning tree one edge at a time by adding the lowest weight edge that connects the current tree to a vertex not yet in the tree

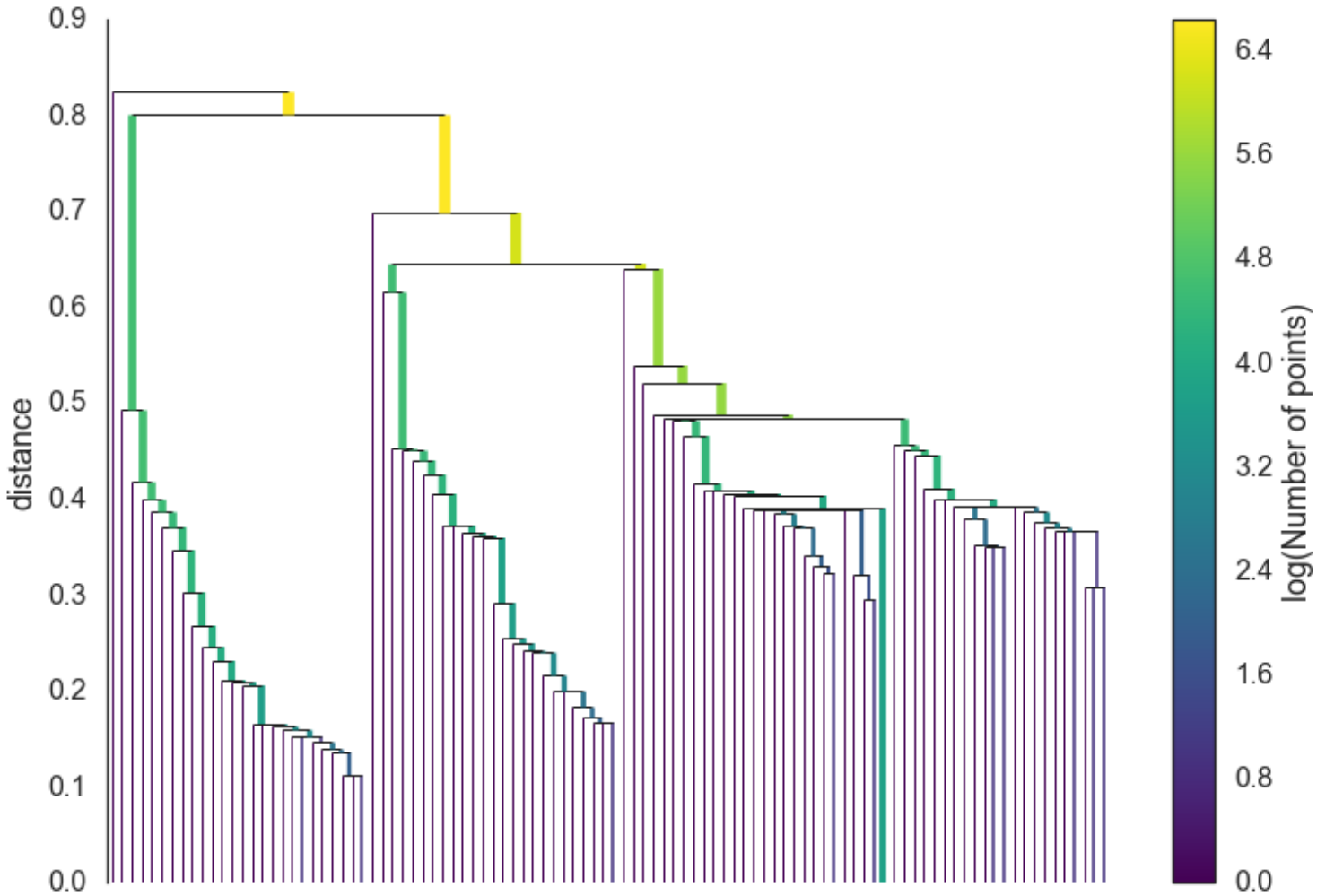


https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

HDBSCAN converts the minimum spanning tree into a hierarchy of connected components

It sorts the edges of the tree by distance (in increasing order)

Then iterate creating a new merged cluster for each edge.

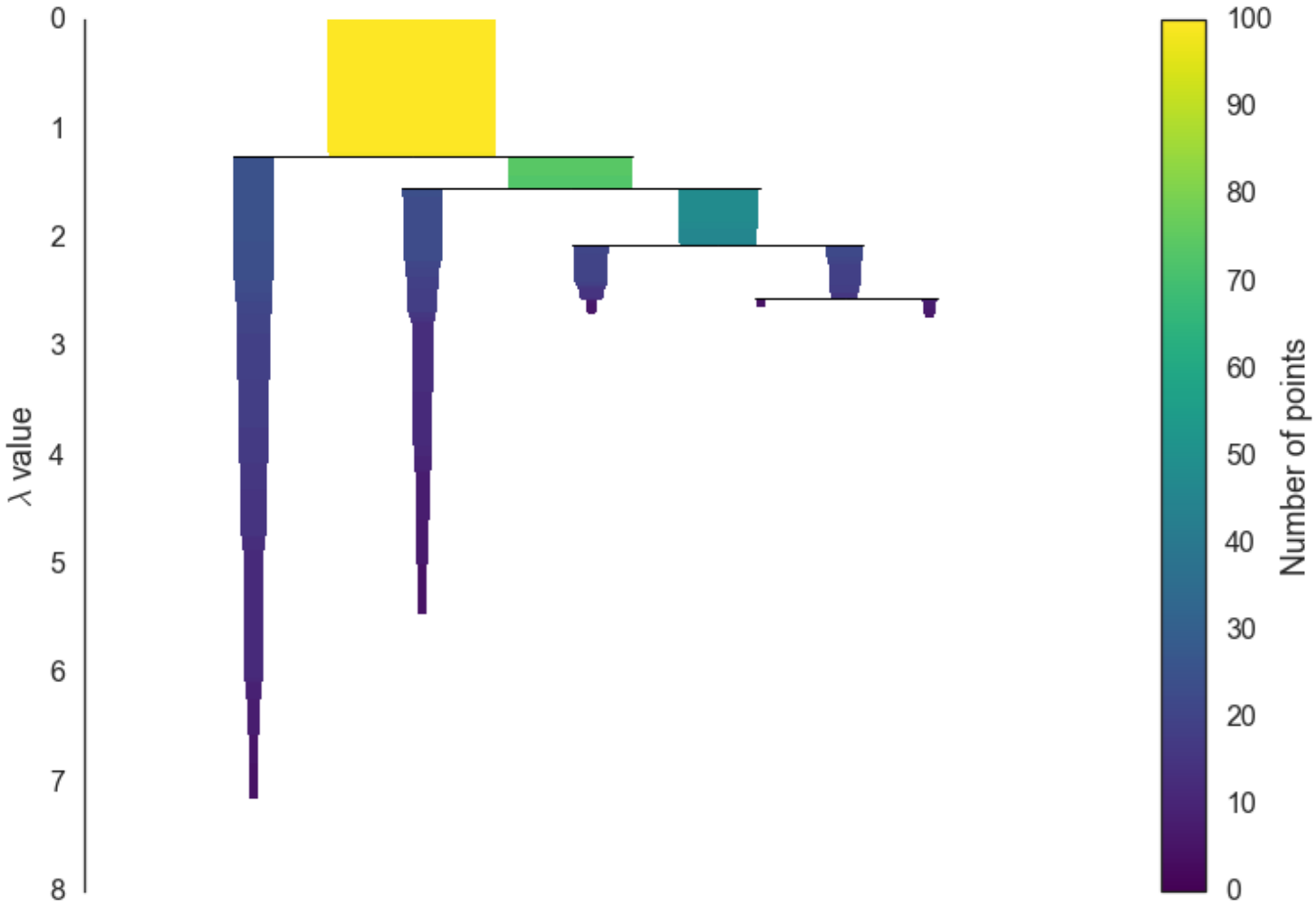


https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

We have a hierarchy now
we want a set of flat clusters

Condensing the Tree

- We introduce the notion of "minimum cluster size" threshold
- We navigate the hierarchy and from the top and at each split we check the size of the merged clusters
- If a clusters has fewer points that the threshold, then the smaller clusters is eliminated (the points have "fallen out of the cluster")
- Otherwise both clusters are maintained.
- Note that, which a point p 'fell out of the cluster' and at what distance value that happened is stored as $\lambda_p = 1/\text{distance}$
- At the end, a much smaller tree with a small number of nodes remains with node has data about how the size of the cluster at that node decreases over varying distance

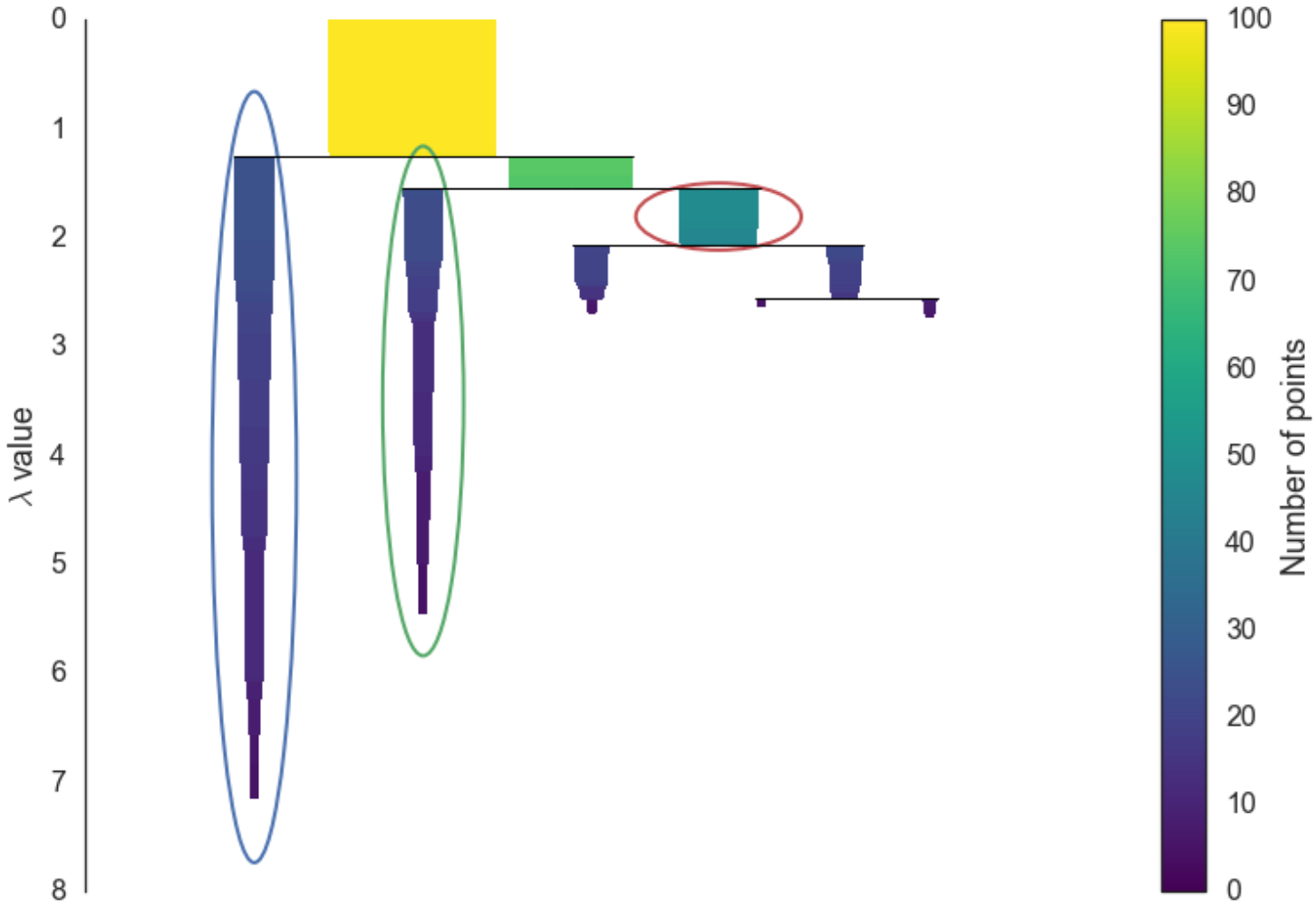


Condensed tree (λ = 1/distance)
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

Extracting the Cluster

- Given the simplified dendrogram we want to select clusters that persist and have a longer lifetime since short lived clusters probably represent artifacts

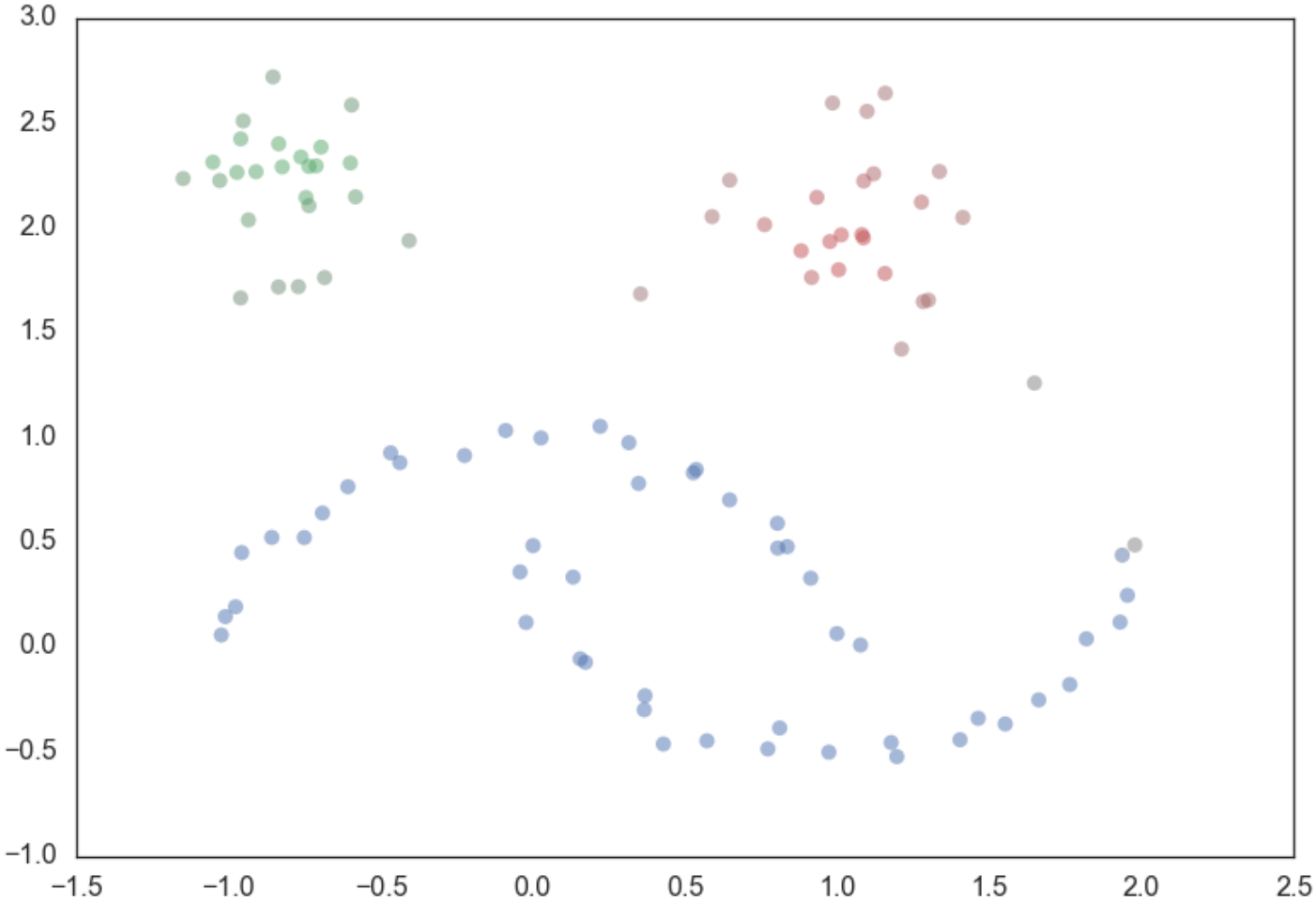
- We may say that we want to choose those clusters that have the greatest area of ink in the plot. Also, we cannot select any a descendant of a cluster we chose.



By applying the two principles (larger total ink area and no descendents) we extract three clusters
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

Finally, we can assign to each point a probability to belong to a cluster based on its λ_p

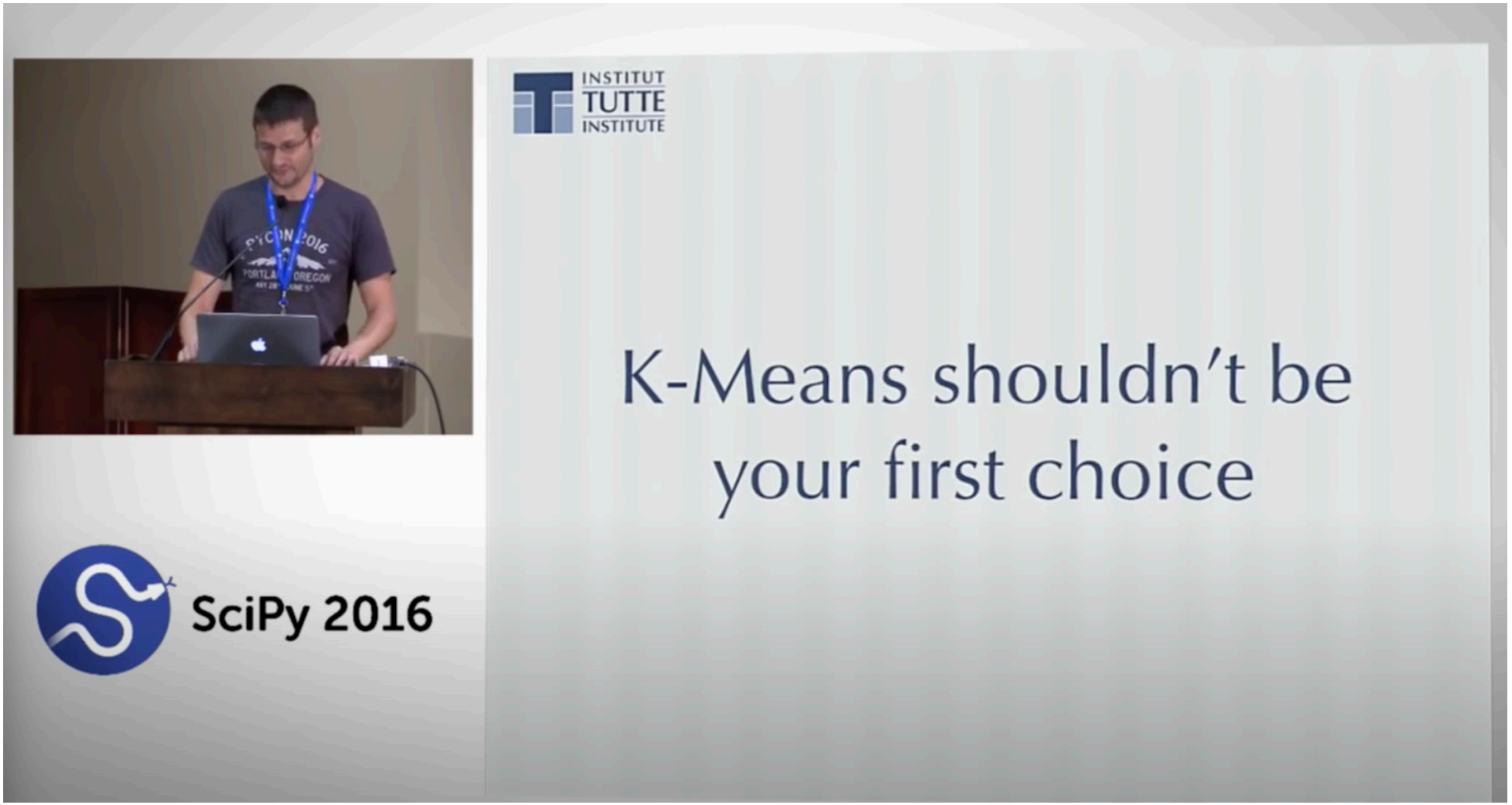
HDBSCAN complexity is comparable and lower than DBSCAN



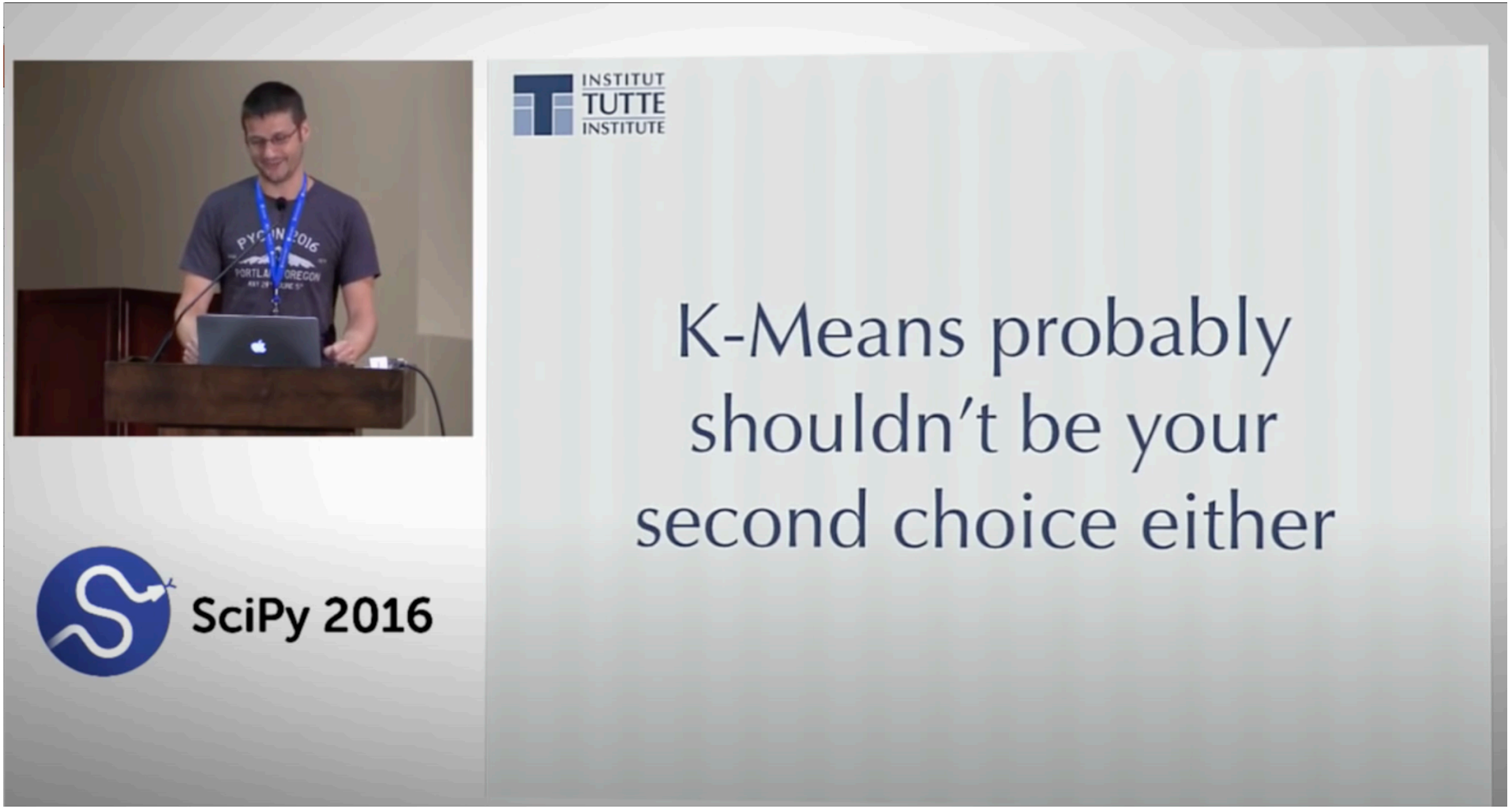
Finally, we can assign to each point a probability to belong to a cluster based on its λ_p
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

These are only the basic ones

- Birch and Chameleon are other hierarchical clustering algorithms
- There are also probabilistic hierarchical clustering algorithms
- OPTICS, and DENCLU are interesting density-based algorithms
- STING and CLIQUE are grid-based methods, where CLIQUE is also a subspace clustering algorithm
- …



https://www.youtube.com/watch?v=AgPQ76RIi6A

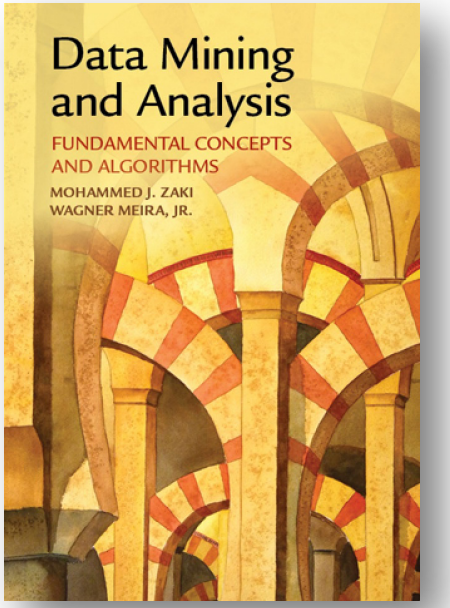


https://www.youtube.com/watch?v=AgPQ76RIi6A

https://www.youtube.com/watch?v=AgPQ76RIi6A

Study Material

- "Data Mining and Analysis" by Zaki & Meira
  - Chapter 13
  - Chapter 15



- http://www.dataminingbook.info

- https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html