

COMP6237 Data Mining

Lecture 2: Discovering Groups

Zhiwu Huang

Zhiwu.Huang@soton.ac.uk

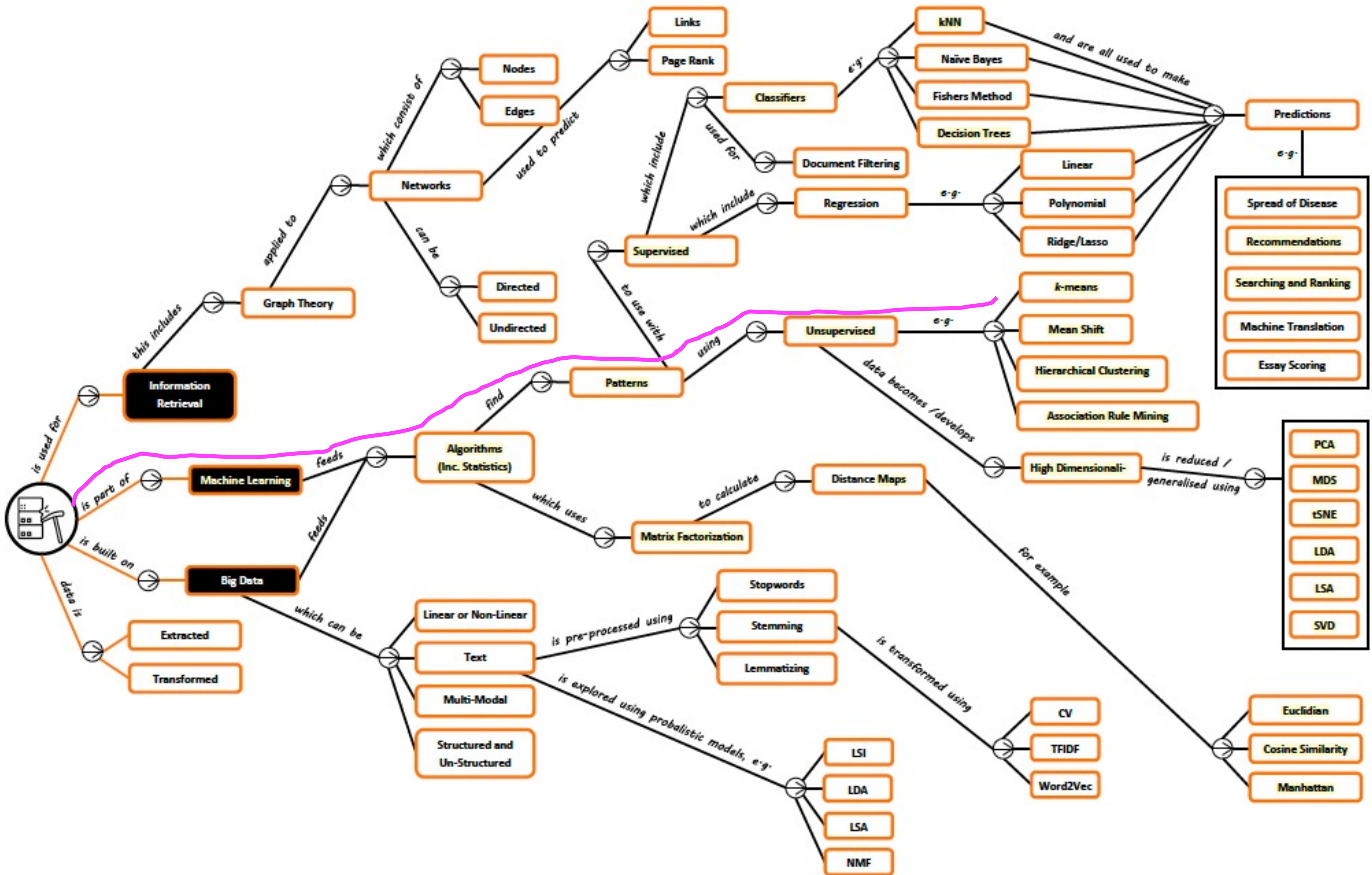
Lecturer (Assistant Professor) @ VLC of ECS
University of Southampton

Lecture slides available here:

<http://comp6237.ecs.soton.ac.uk/jon.html>

(Thanks to Prof. Jonathon Hare and Dr. Jo Grundy for providing the lecture materials used to develop the slides.)

Discovering Groups – Roadmap



Discovering Groups – Textbook

CHAPTER 3

Discovering Groups

- ▶ Programming Collective Intelligence: Building Smart Web 2.0 Applications *T. Segaran.*

Chapter 7

Clustering

Clustering is the process of examining a collection of “points,” and grouping the points into “clusters” according to some distance measure. The goal is that points in the same cluster have a small distance from one another, while points in different clusters are at a large distance from one another. A suggestion of

- ▶ Mining of Massive Datasets J. Leskovec *et al*

Discovering Groups – Overview (1/3)



*Clustering algorithms aim to group data **without any prior knowledge of what the groups should look like or contain.***

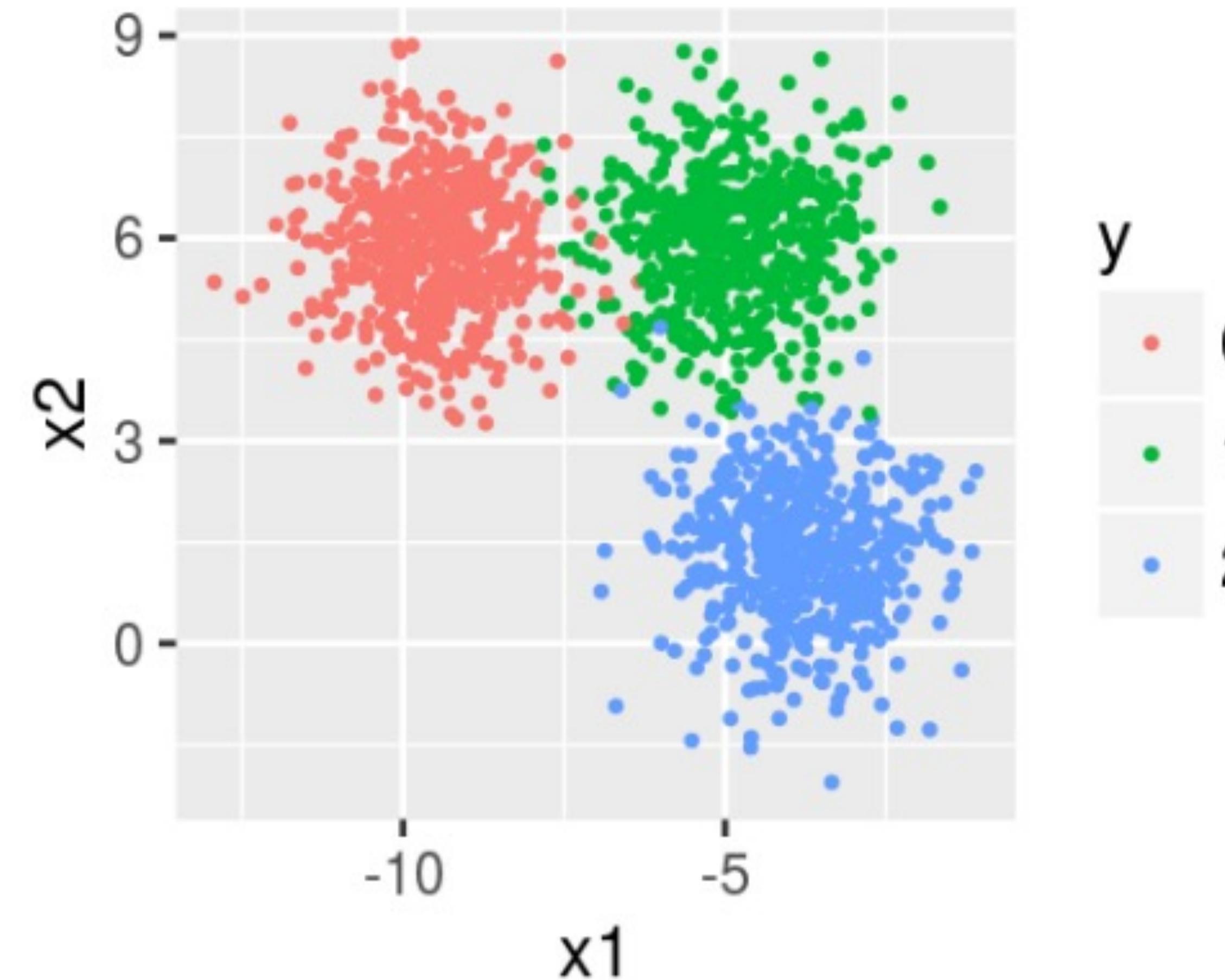
*In terms of feature vectors, **items with similar vectors should be grouped together** by a clustering algorithm.*



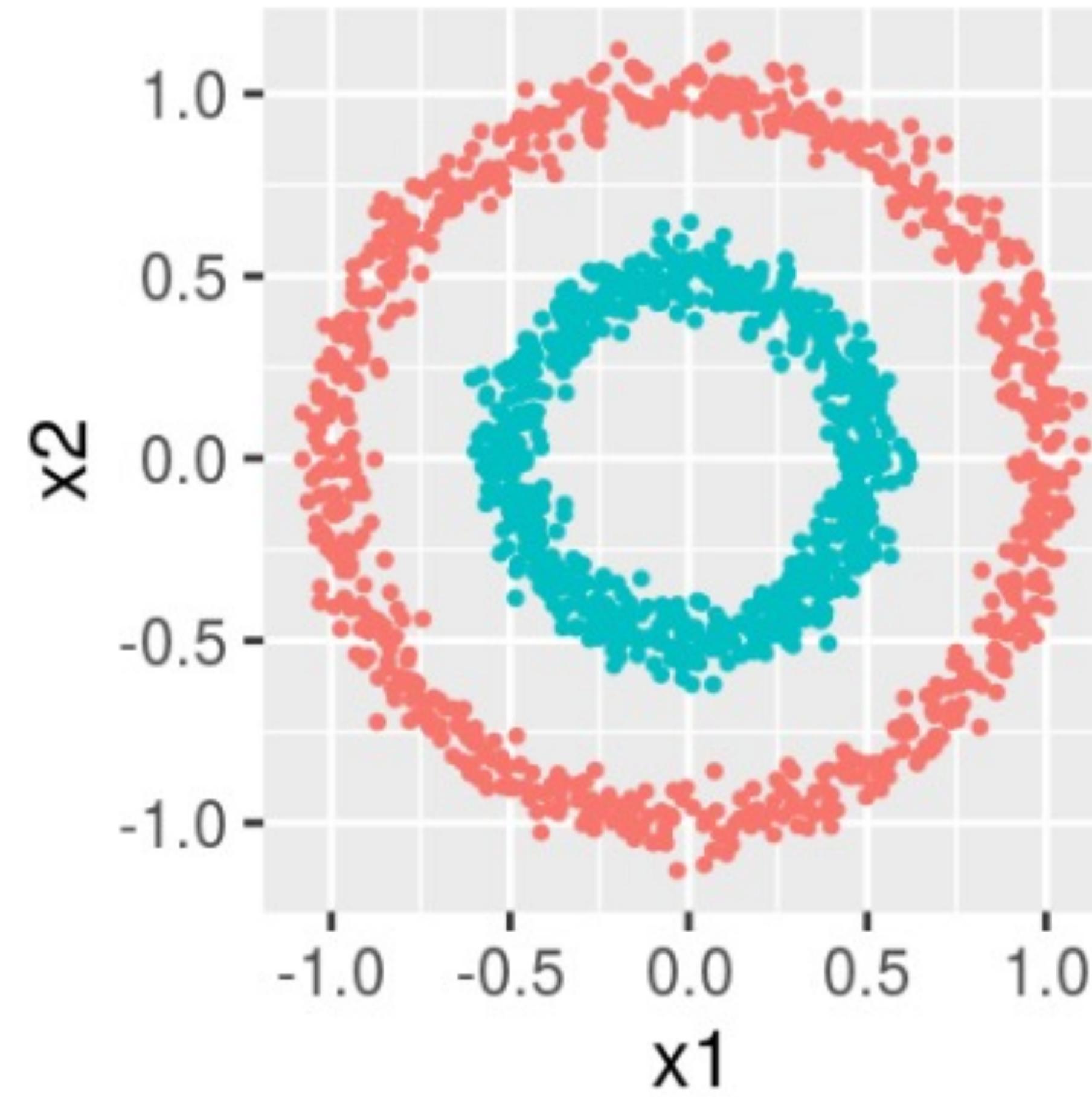
Some clustering algorithms create overlapping groups; others assign an item to a single group.

Discovering Groups – Overview (2/3)

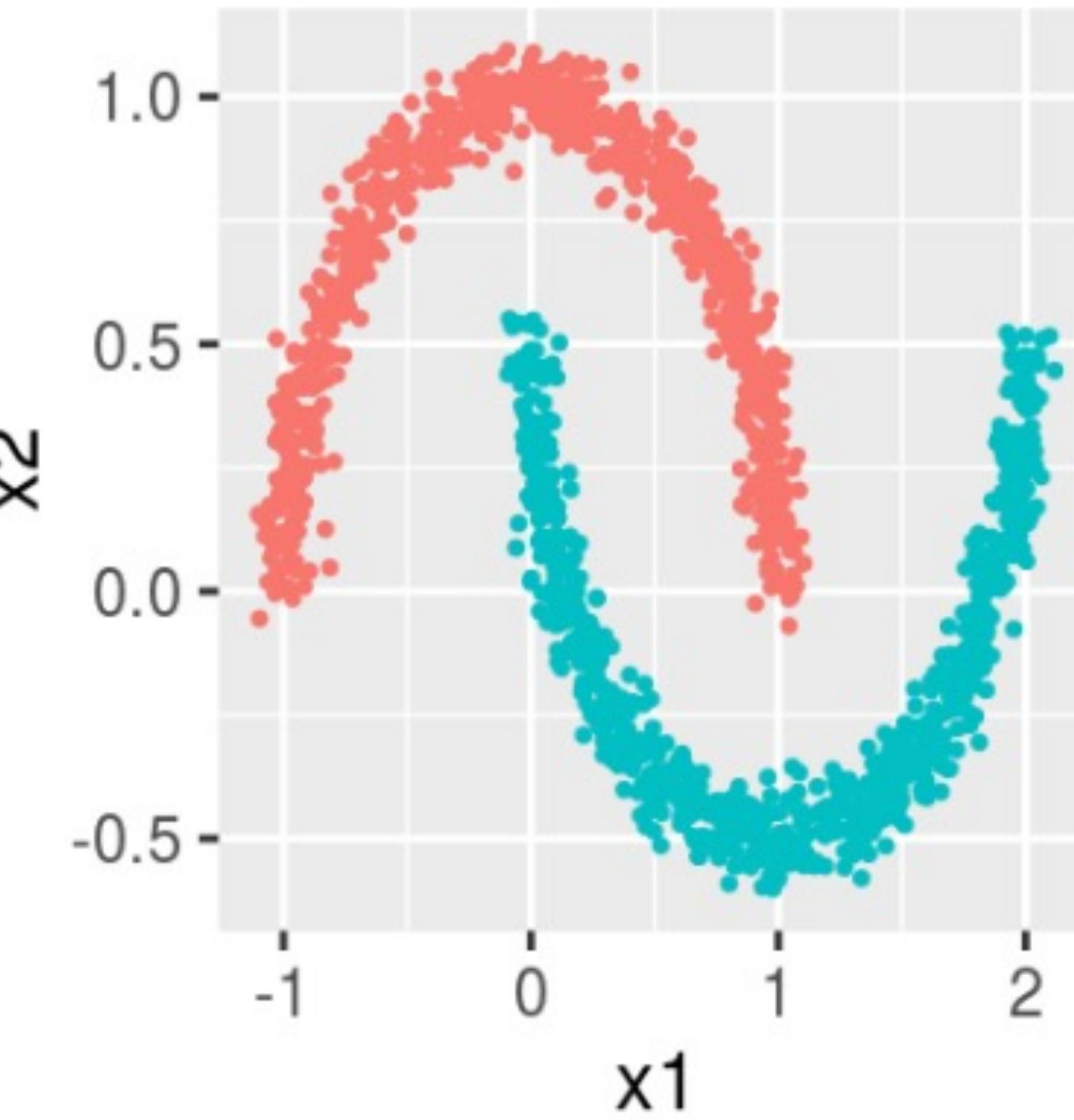
Blobs



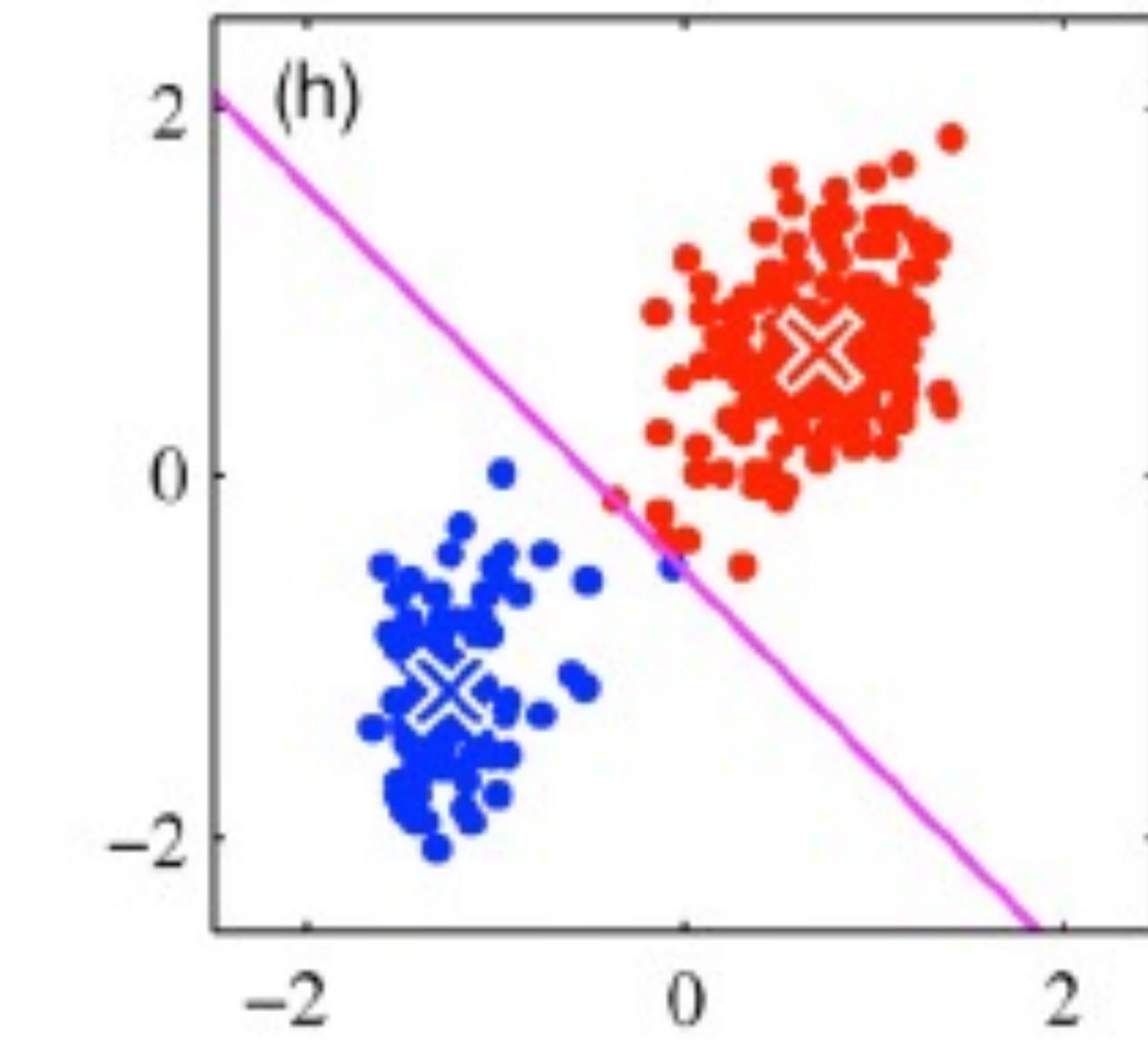
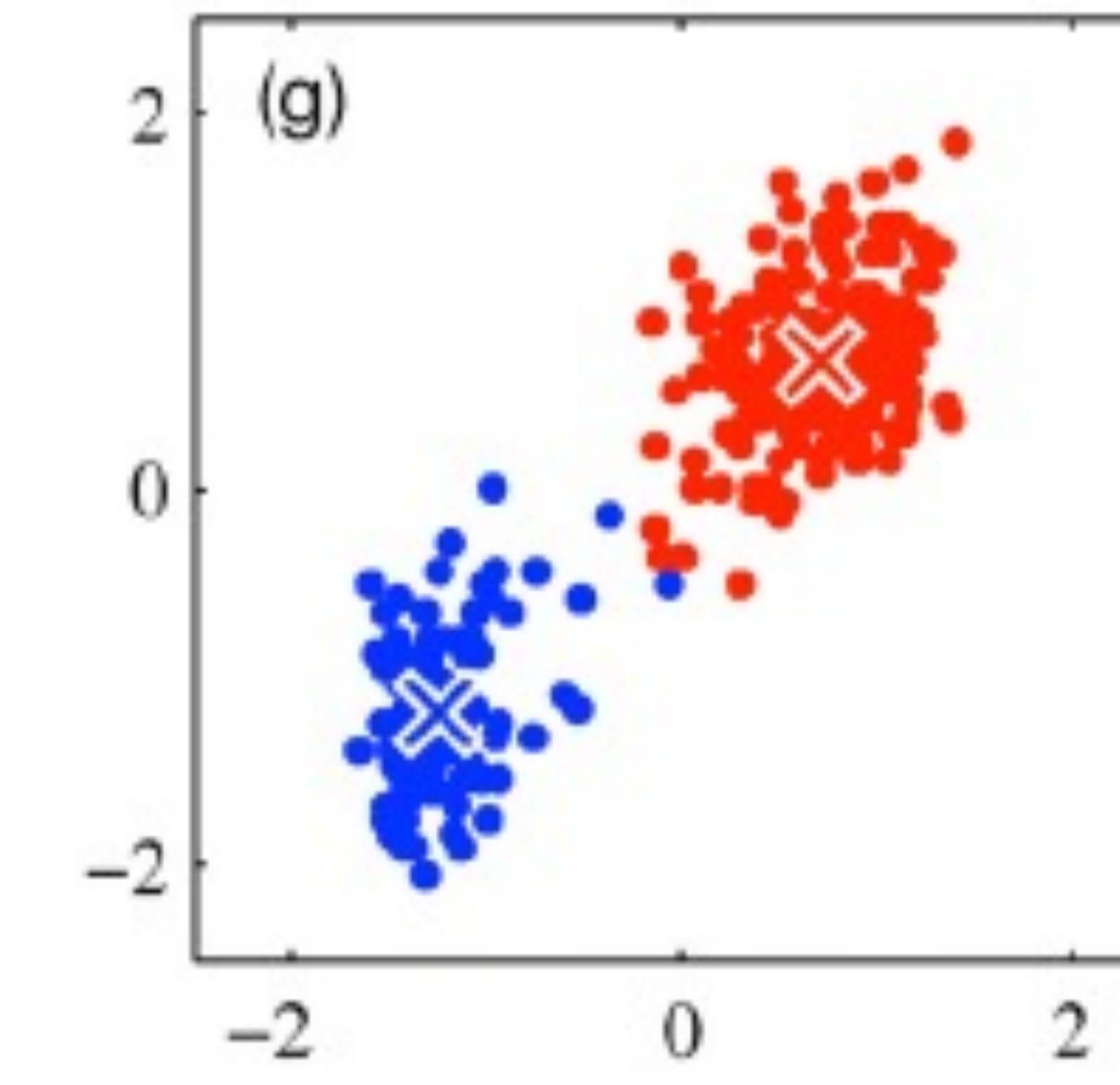
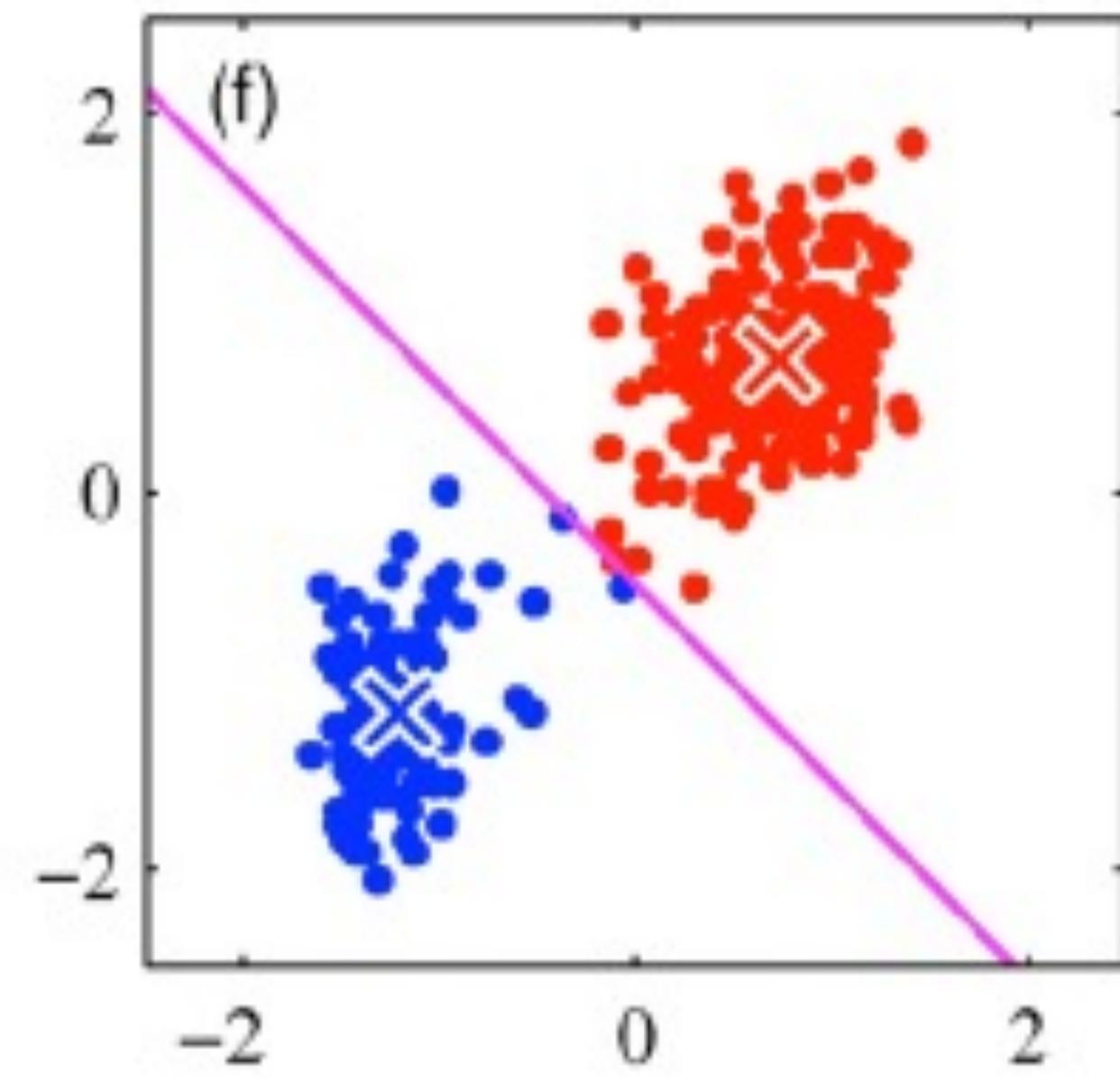
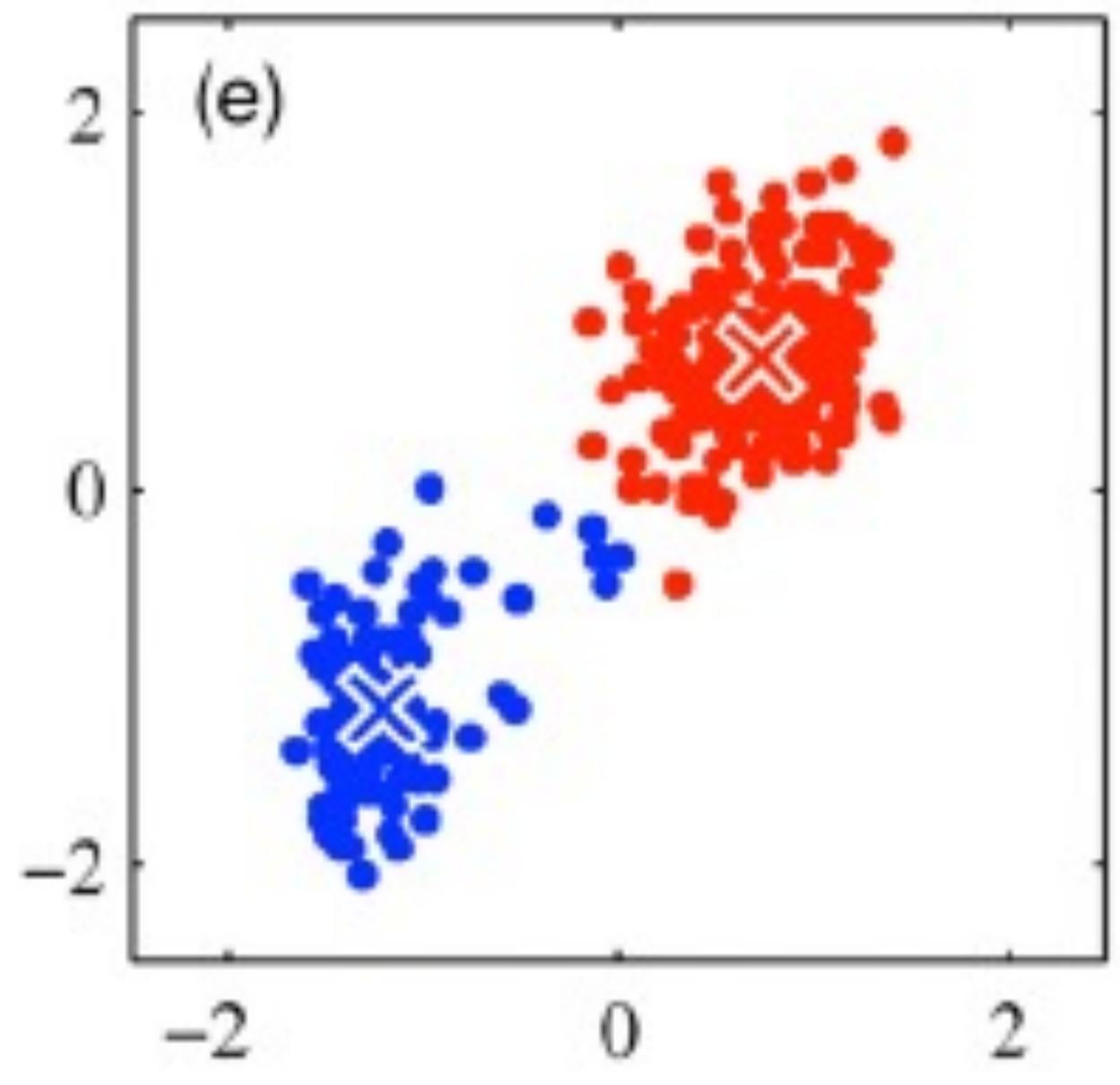
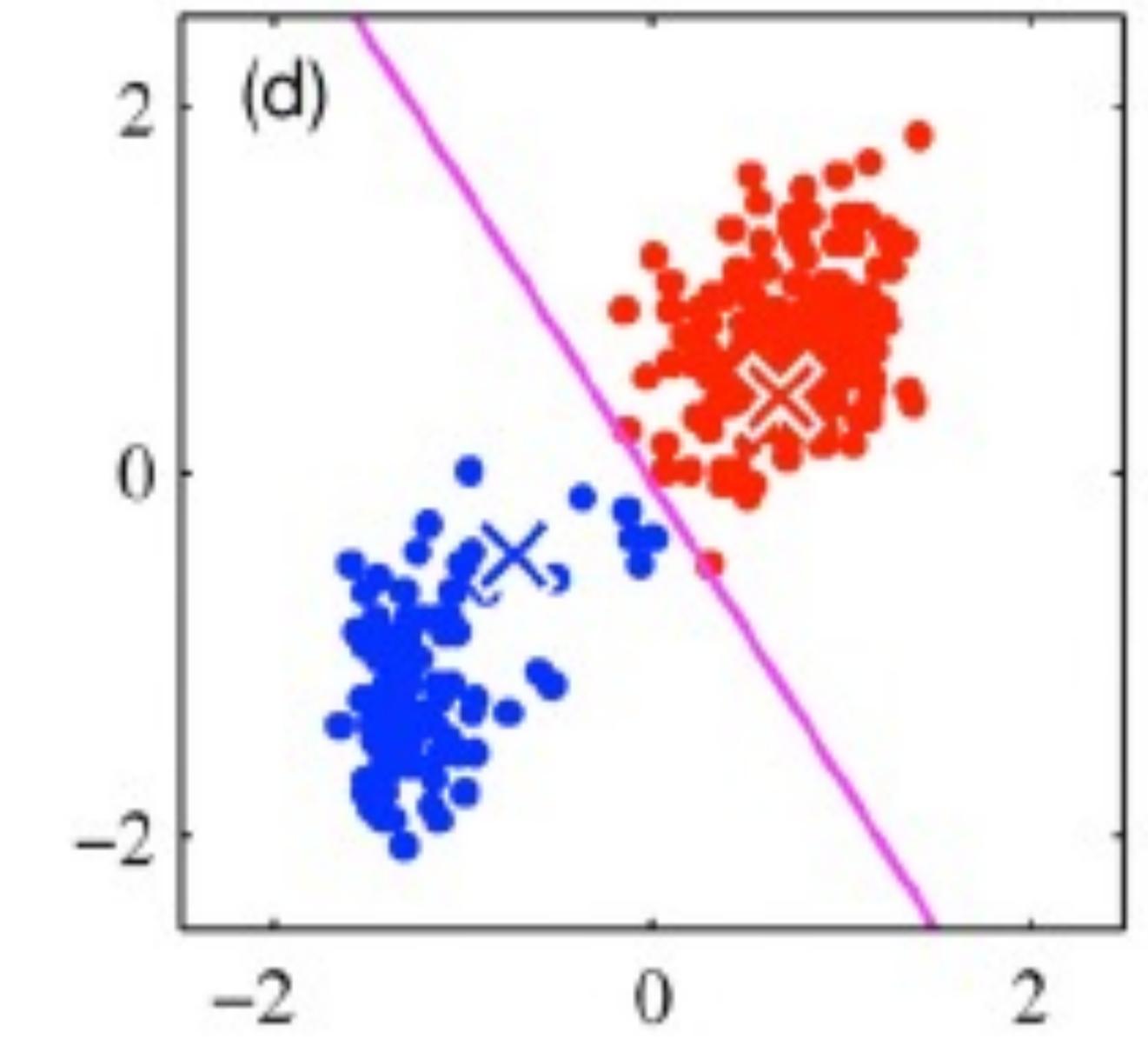
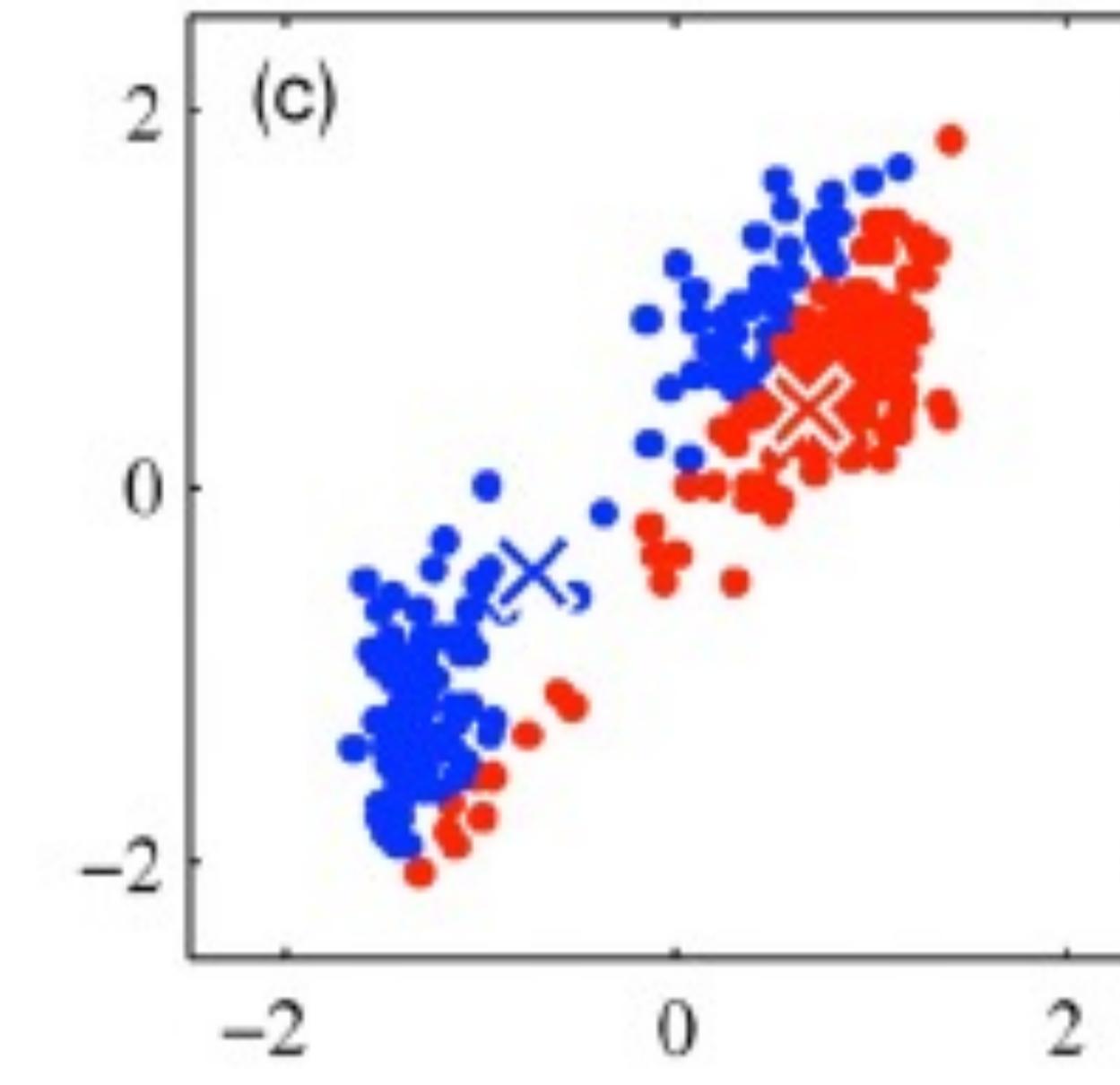
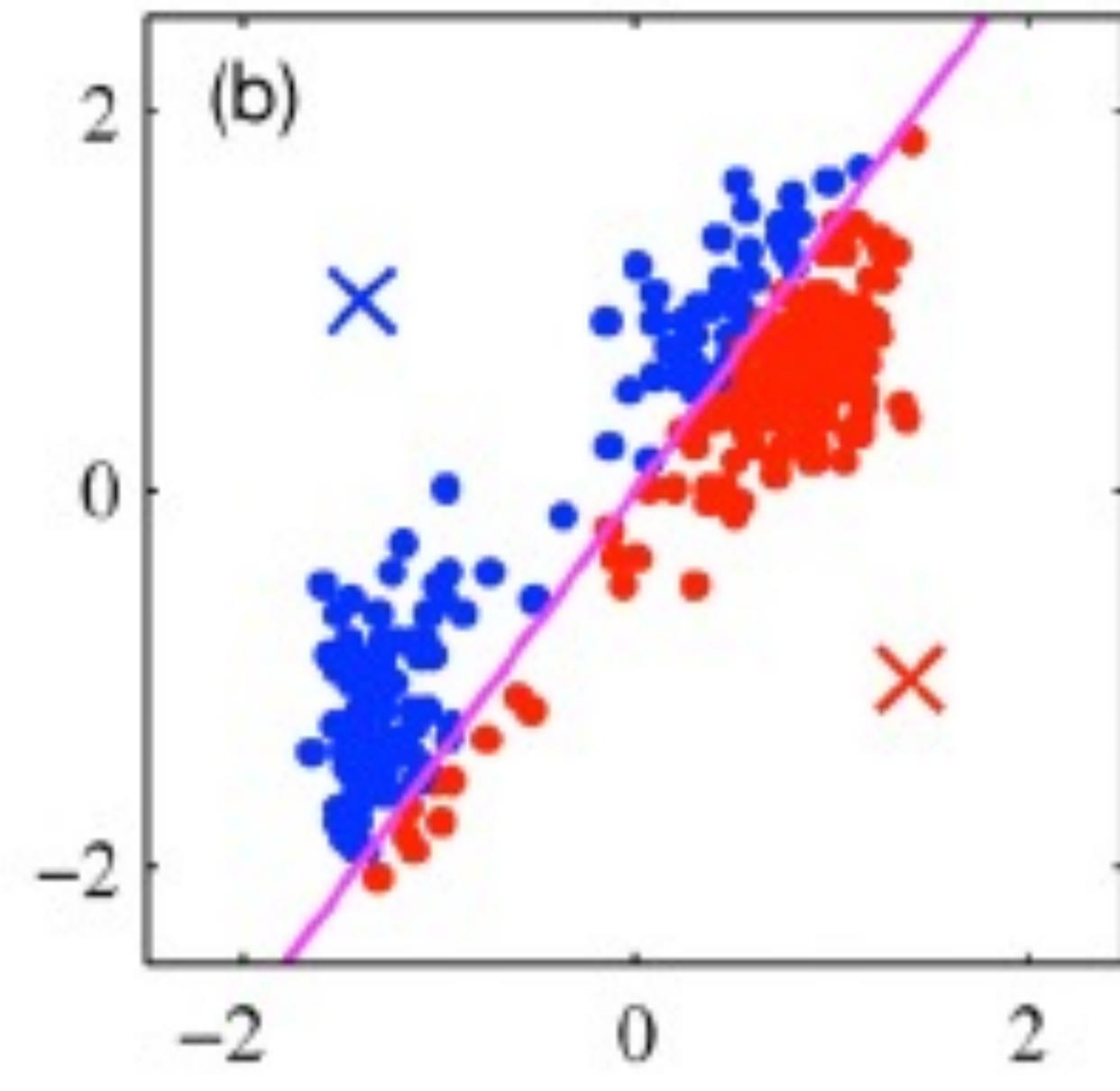
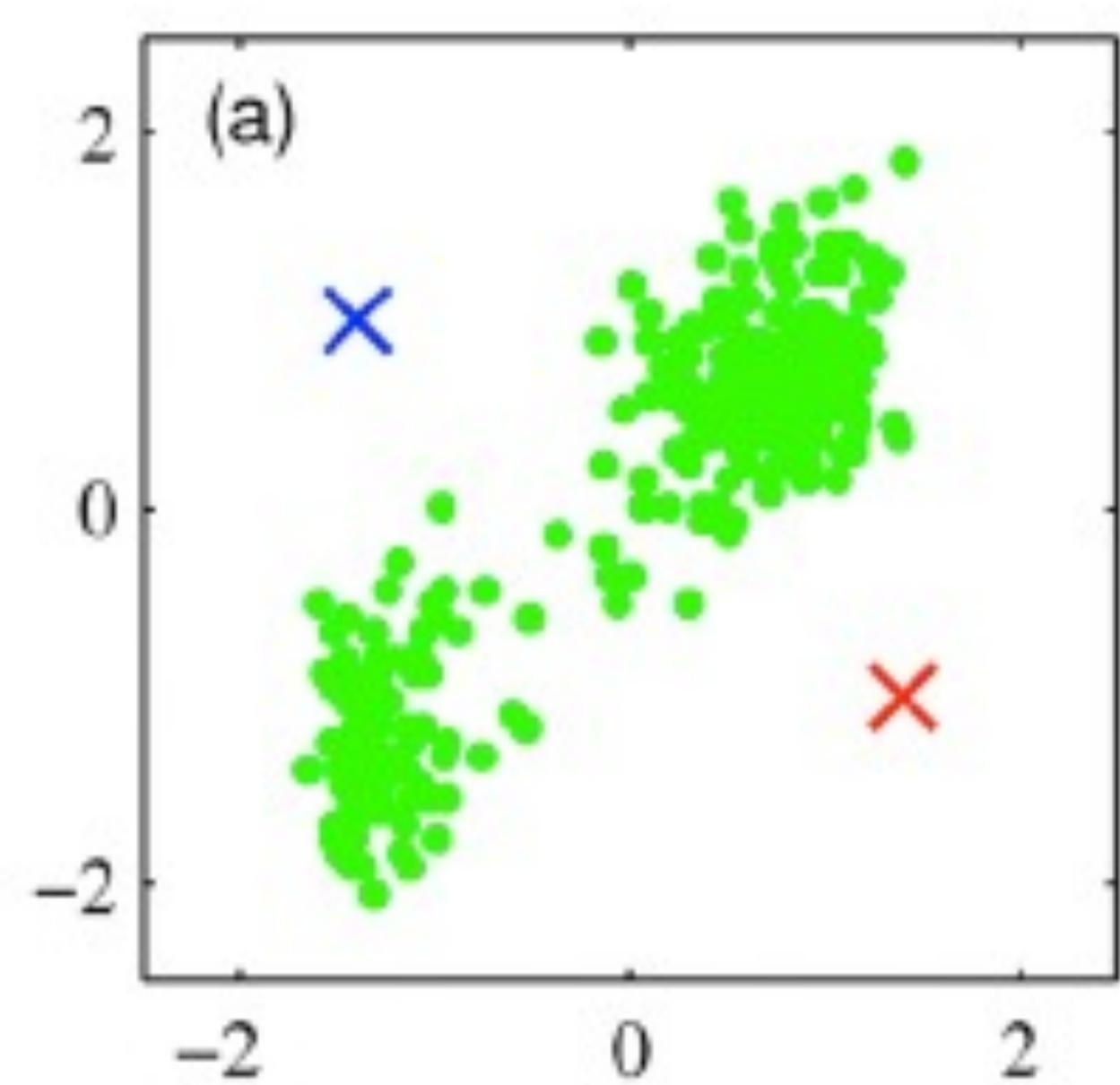
Circles



Moons



Discovering Groups – Overview (3/3)



Discovering Groups – Learning Outcomes

- **LO1:** Comprehend the key ideas and the essential mathematical formulations employed in clustering methods (exam).
 - ❖ E.g., how is sum of squared error (SSE) defined?
 - ❖ E.g., understand the pros and cons of the learned algorithms
- **LO2:** Compute the fundamental stages of learned clustering approaches (exam).
 - ❖ E.g., given a dataset and a distance metric, be prepared to follow the selected clustering algorithm to cluster the instances in the dataset
- **LO3:** Implement and evaluate the learned clustering algorithms using Python (course work)

Assessment hints: Multi-choice Questions (single answer: concepts, calculation etc)

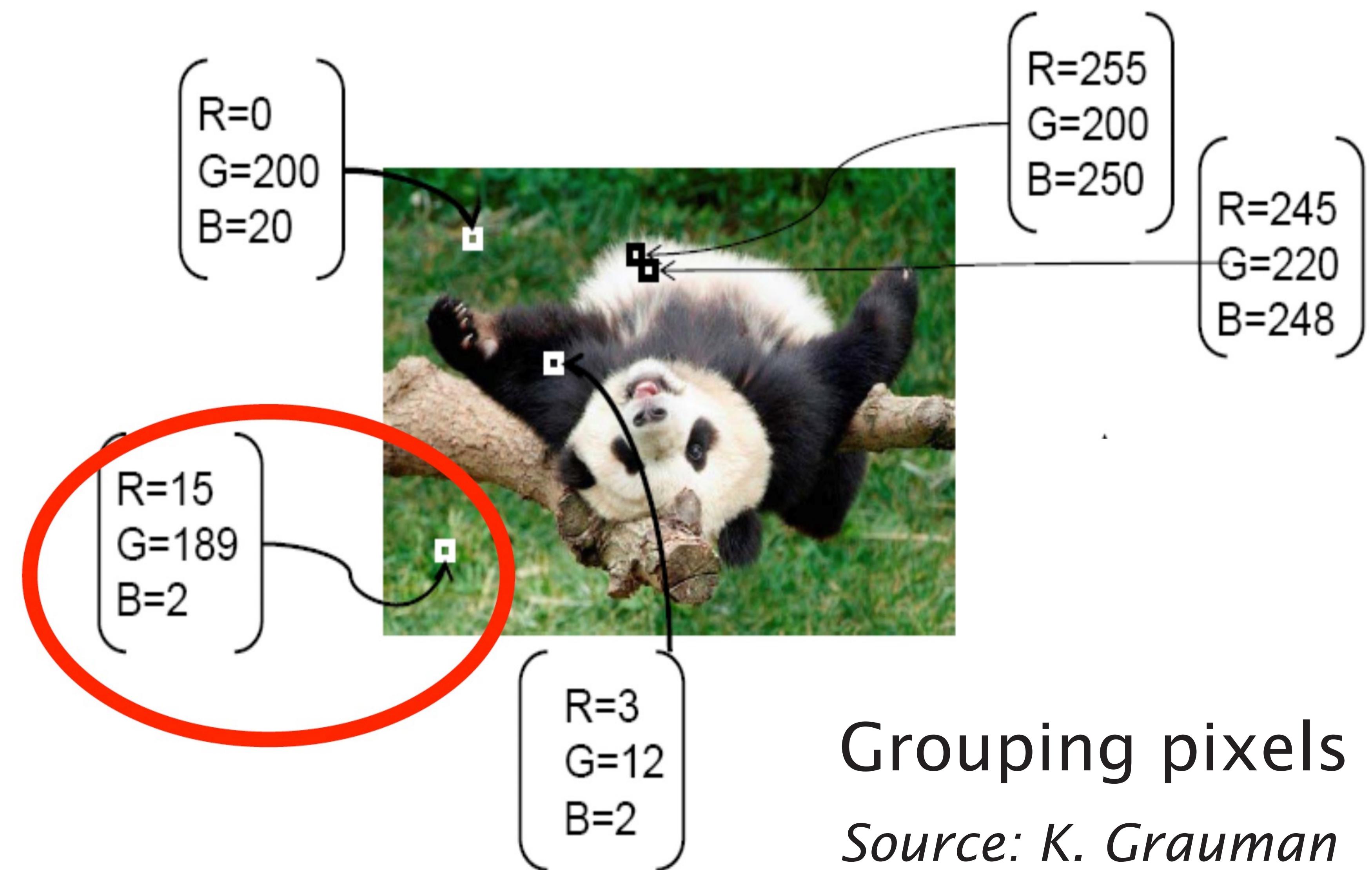
- *Textbook Exercises: textbooks (Programming + Mining)*
- *Other Exercises: <https://www-users.cse.umn.edu/~kumar001/dmbook/sol.pdf>*
- *ChatGPT or other AI-based techs*

Discovering Groups – Introduction

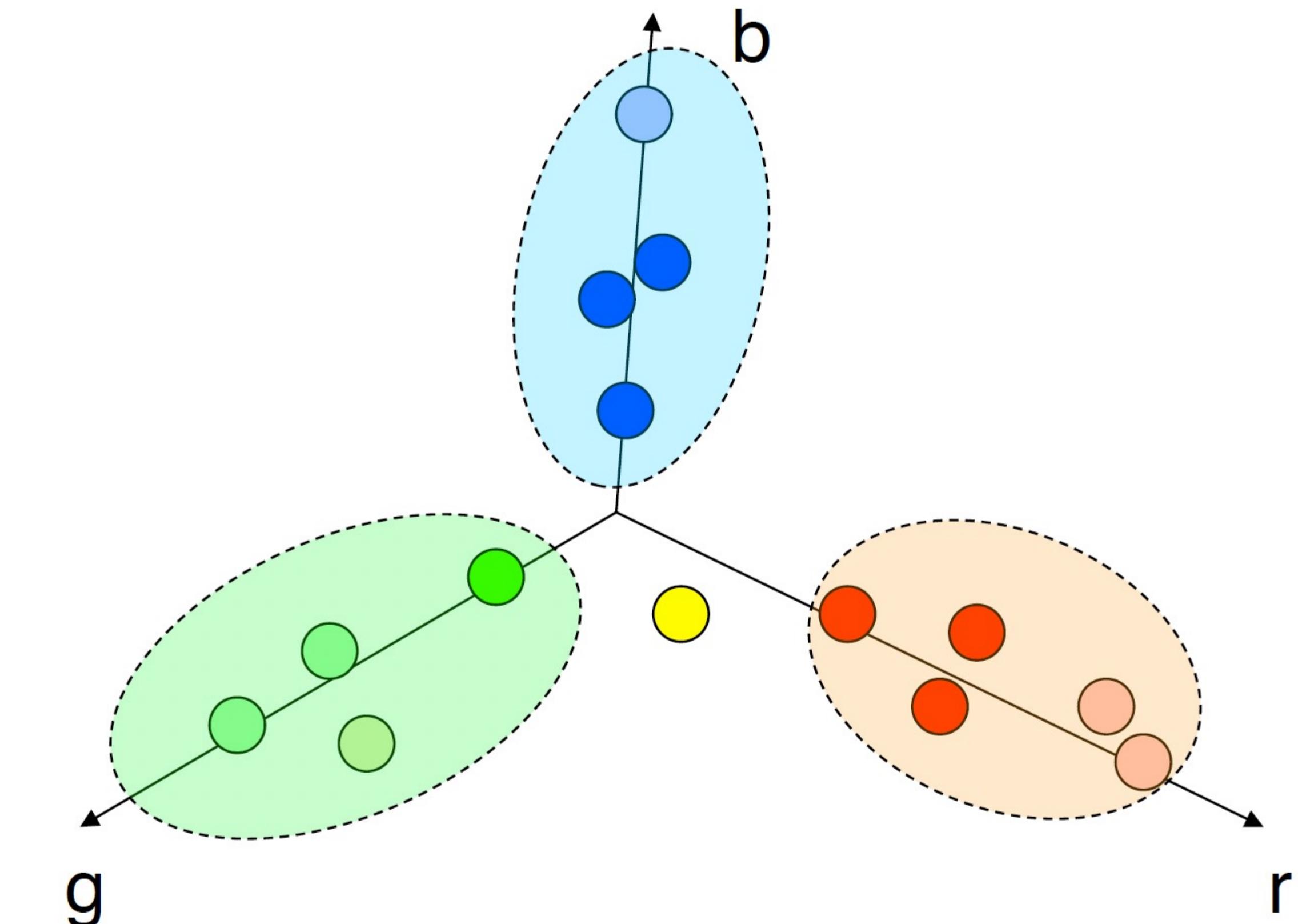
- Understanding large datasets is hard, especially if it has high dimensional features
- To help understand a dataset
 - Find similar data items
 - Find similar features

Discovering Groups – Clustering

- Grouping data, just using the feature vectors
 - Unsupervised: no group labels for training
 - Key idea: data with similar features grouped together
 - Can be
 - Soft (allow overlapping groups)
 - Hard (each item assigned to one group)



Feature space:
color value (3D)



Grouping pixels based on **color** similarity
Source: K. Grauman

Discovering Groups – K-means

- Given: data set $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, number of clusters K
- Goal: find cluster centers $\{\mu_1, \dots, \mu_K\}$ so that

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2$$

is minimal, where $r_{nk} = 1$ if \mathbf{x}_n is assigned to μ_k

- Idea: compute r_{nk} and μ_k iteratively Otherwise, $r_{n,k} = 0$

The used objective function is **Sum of Squared Error (SSE)**

Discovering Groups – K-means

Algorithm 1: K Means clustering

Data: X, K

initialise K centroids;

while positions of centroids change **do**

for each data point **do**

 | assign to nearest centroid

end

for each centroid **do**

 | move to average of assigned data points

end

end

return centroids, assignments;

A special case of Expectation Maximisation - why?

Discovering Groups - K-means

Algorithm 2: K Means clustering

Data: X, K

initialise K centroids;

while positions of centroids change do

for each data point do

assign to nearest centroid ;

associations E-step: Estimate the posterior probabilities...

end

for each centroid do

move to average of assigned data points ;

// Maximisation of likelihood M-step: Estimate new parameters

end

end

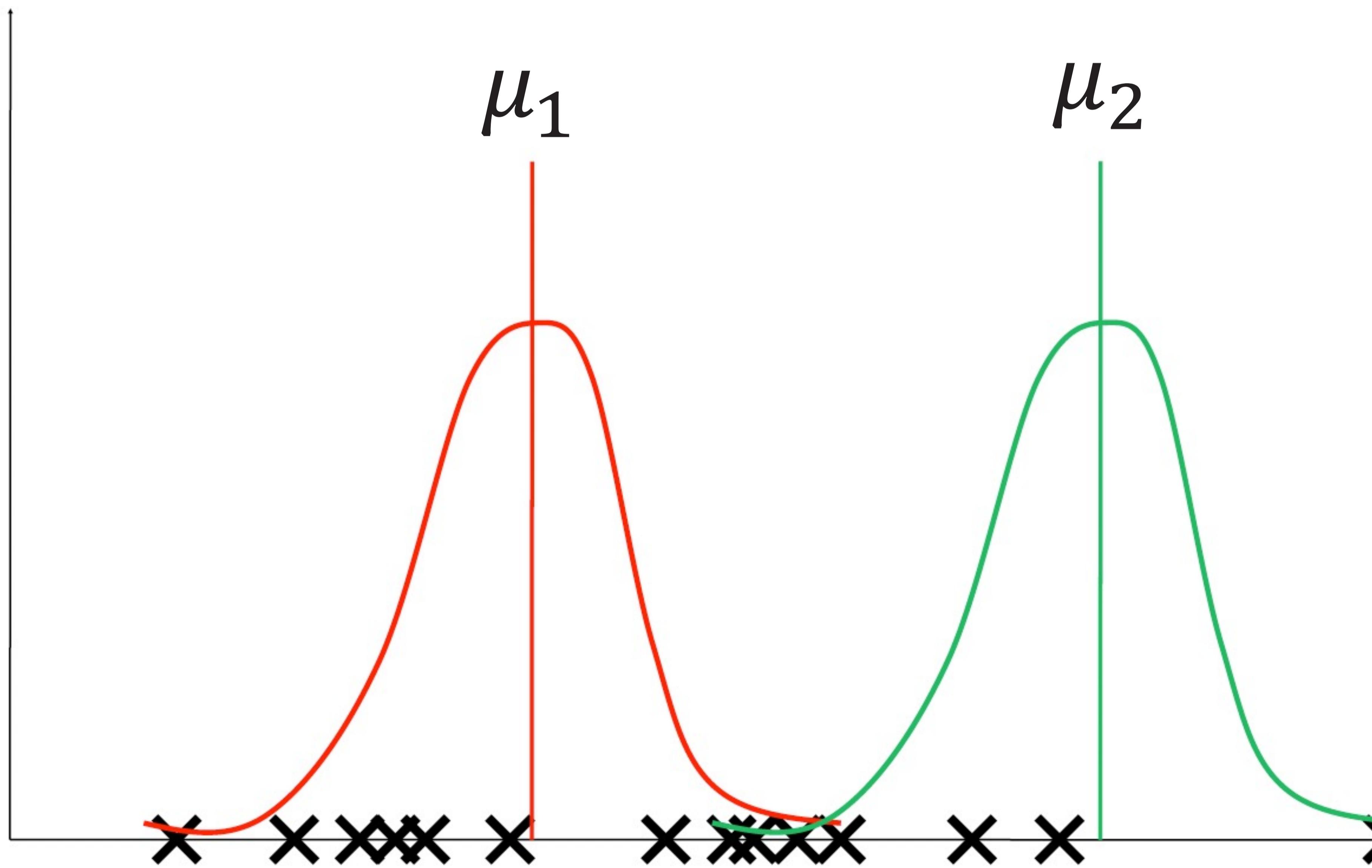
```
return centroids, assignments;
```

Assumes spherical clusters

Discovering Groups – K-means

1D Example

Initialize cluster means: $\{\mu_1, \dots, \mu_K\}$

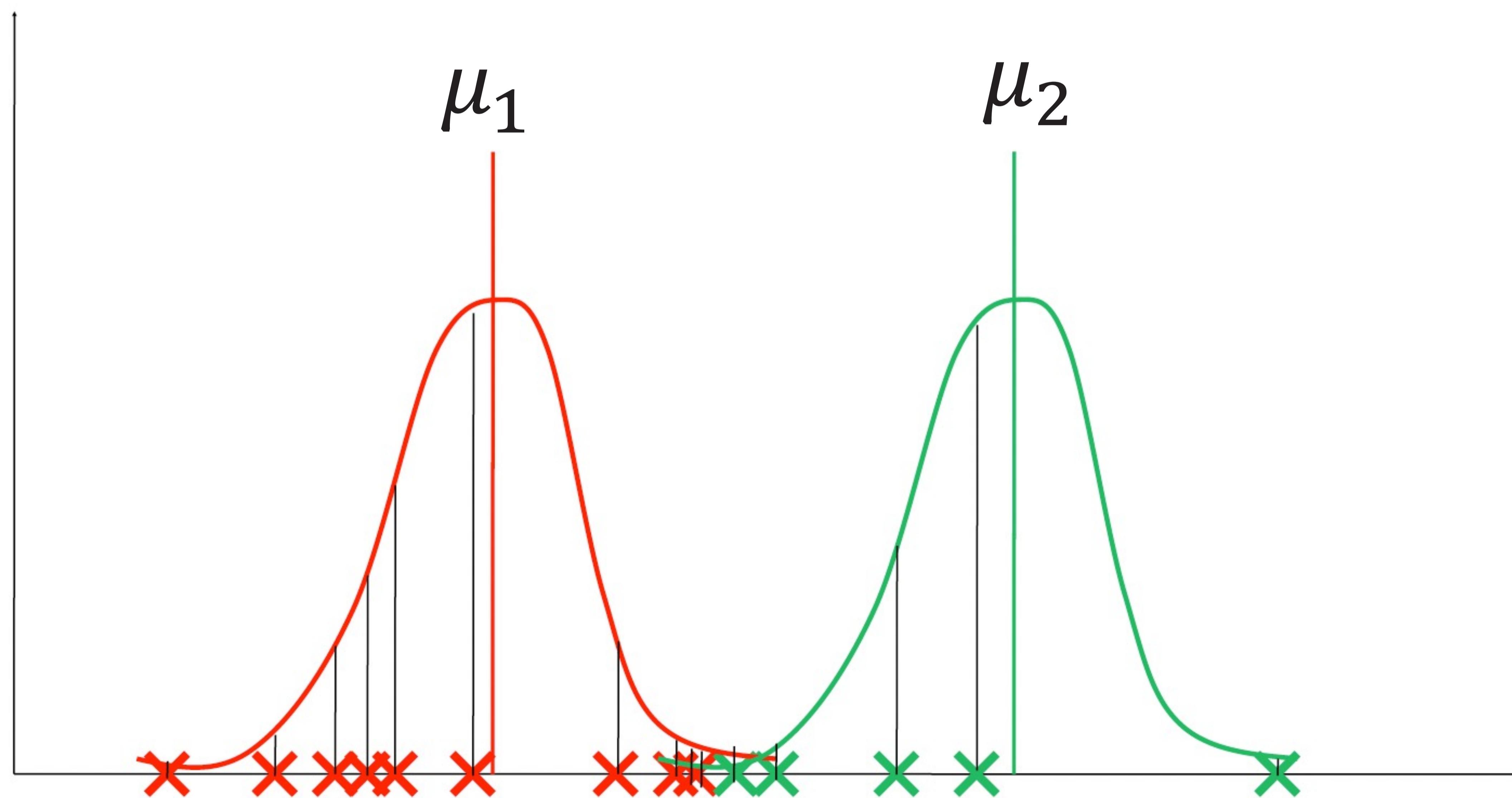


Discovering Groups – K-means

1D Example

Find optimal assignments:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\| \\ 0 & \text{otherwise} \end{cases}$$



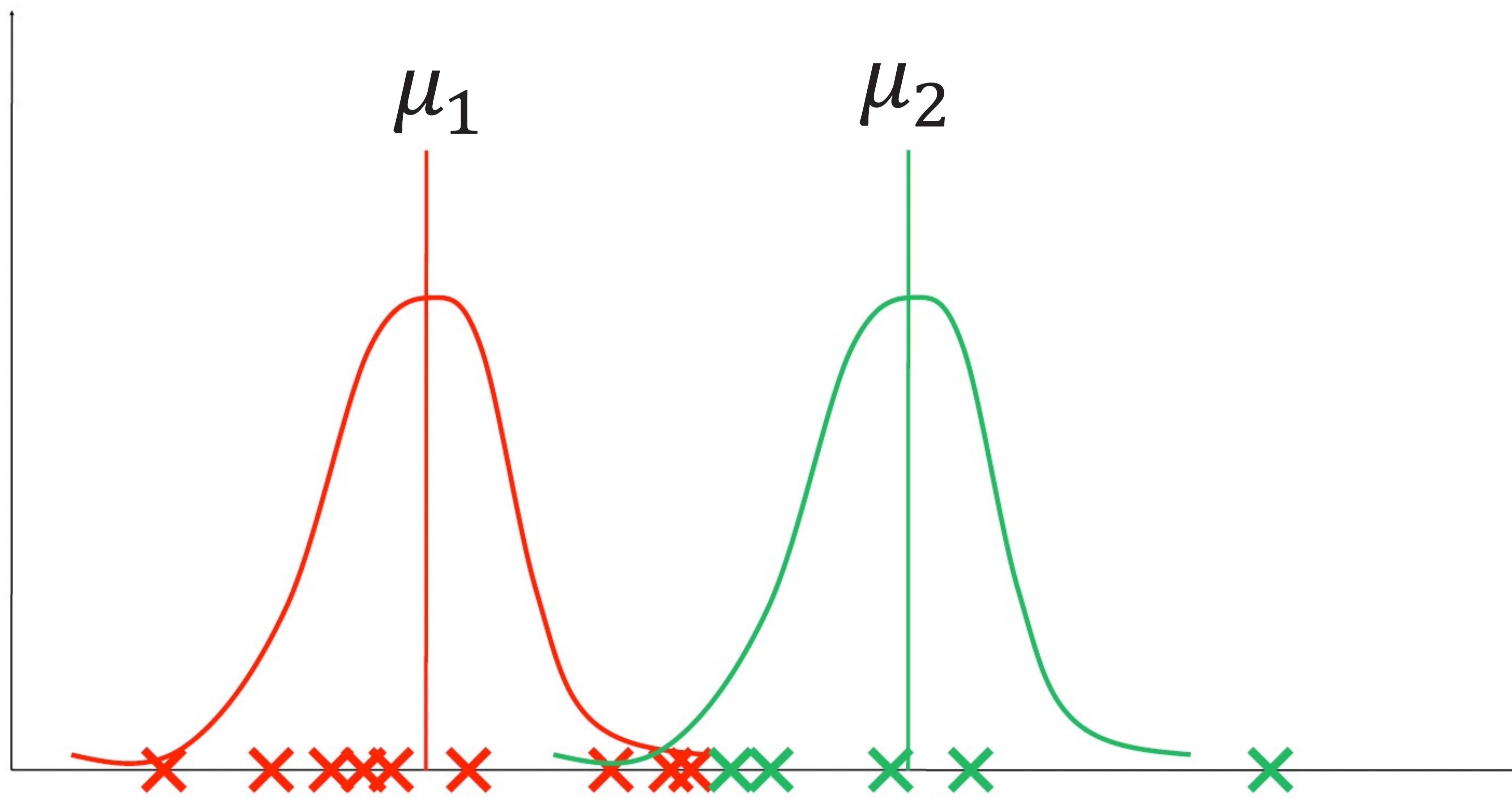
Discovering Groups – K-means

1D Example

Find new optimal means:

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \mu_k) \stackrel{!}{=} 0$$

$$\Rightarrow \mu_k = \frac{\sum_{n=1}^N r_{nk} \mathbf{x}_n}{\sum_{n=1}^N r_{nk}}$$

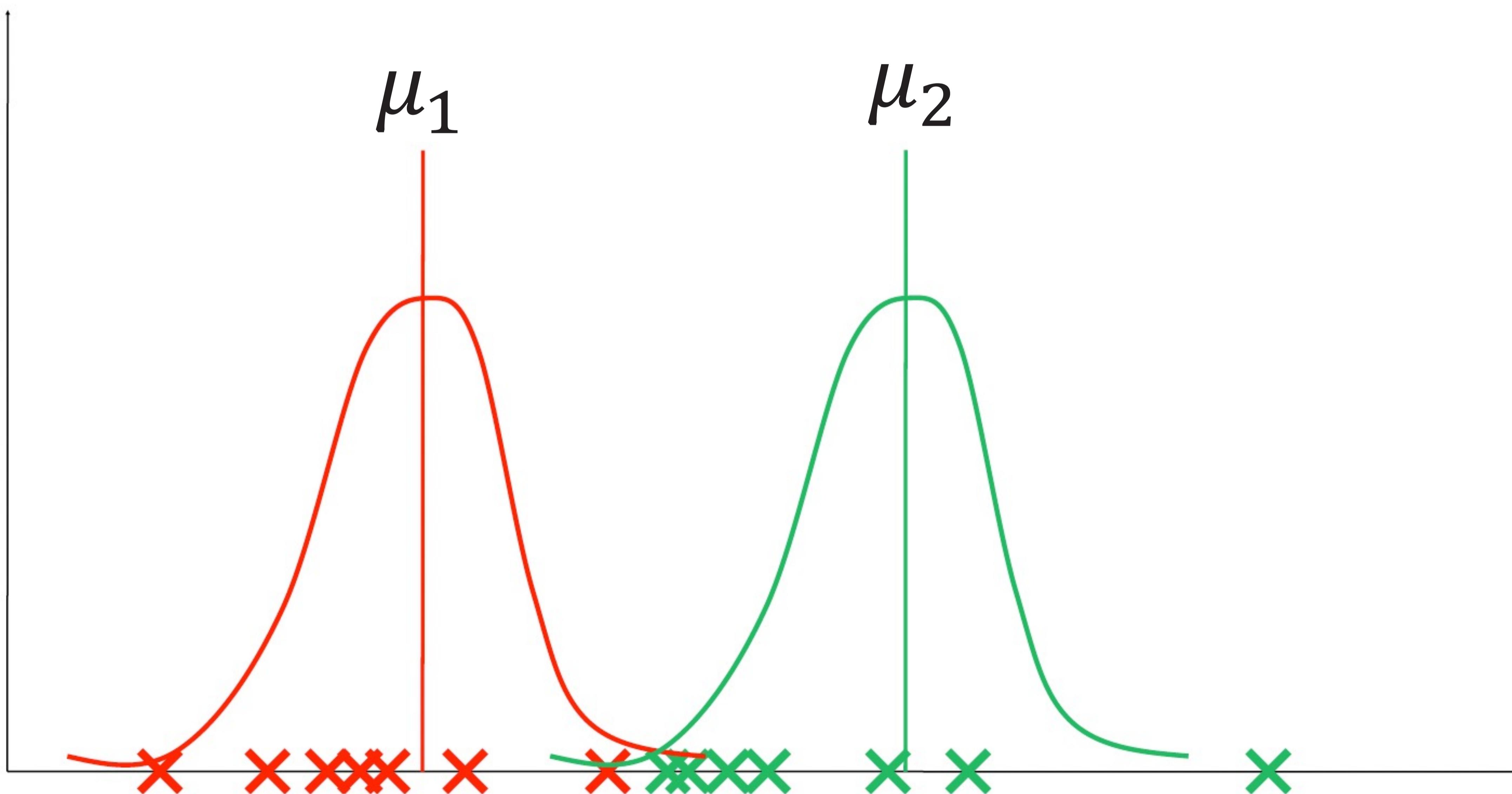


Discovering Groups – K-means

1D Example

Find new optimal assignments:

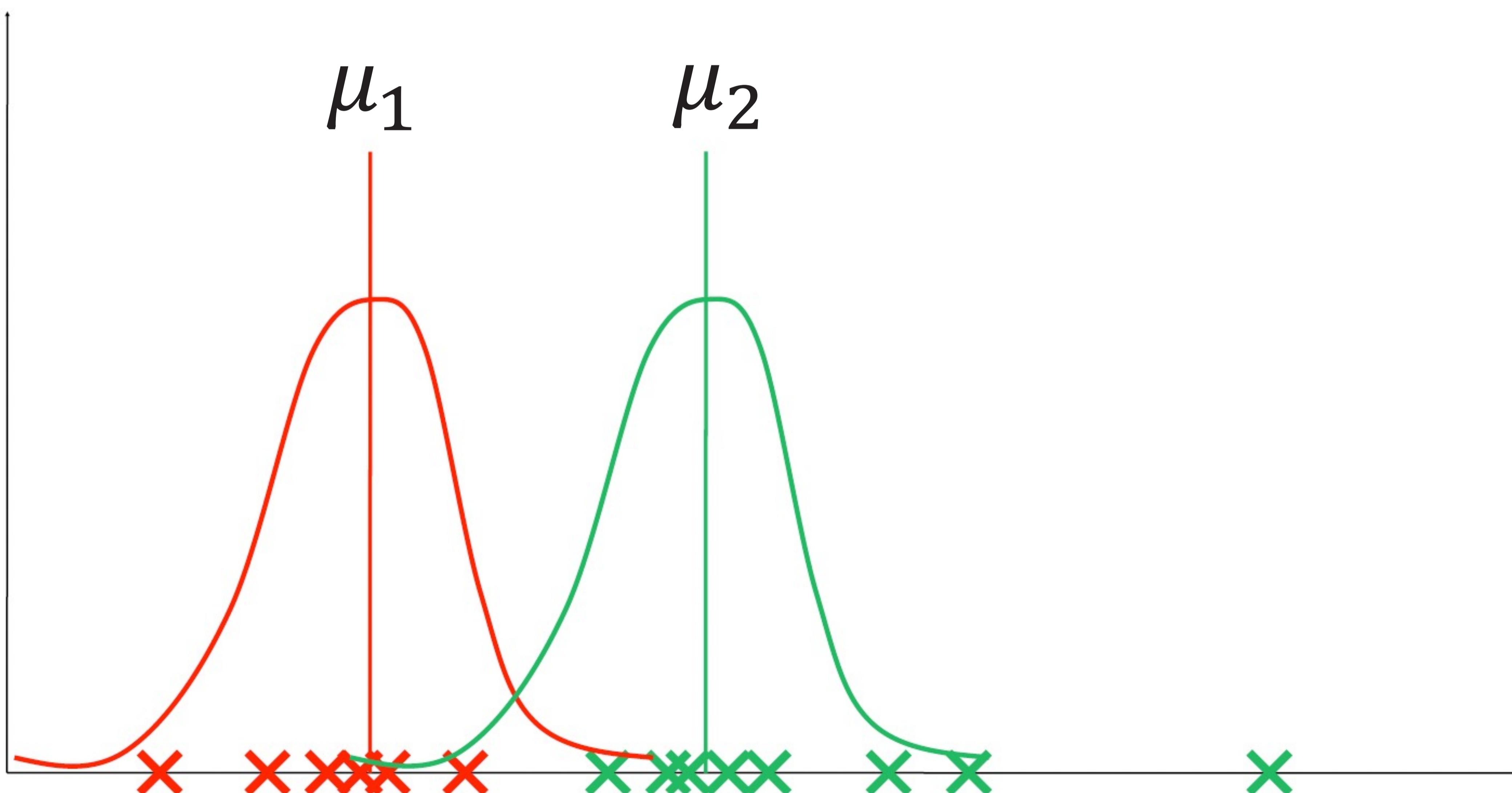
$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\| \\ 0 & \text{otherwise} \end{cases}$$



Discovering Groups – K-means

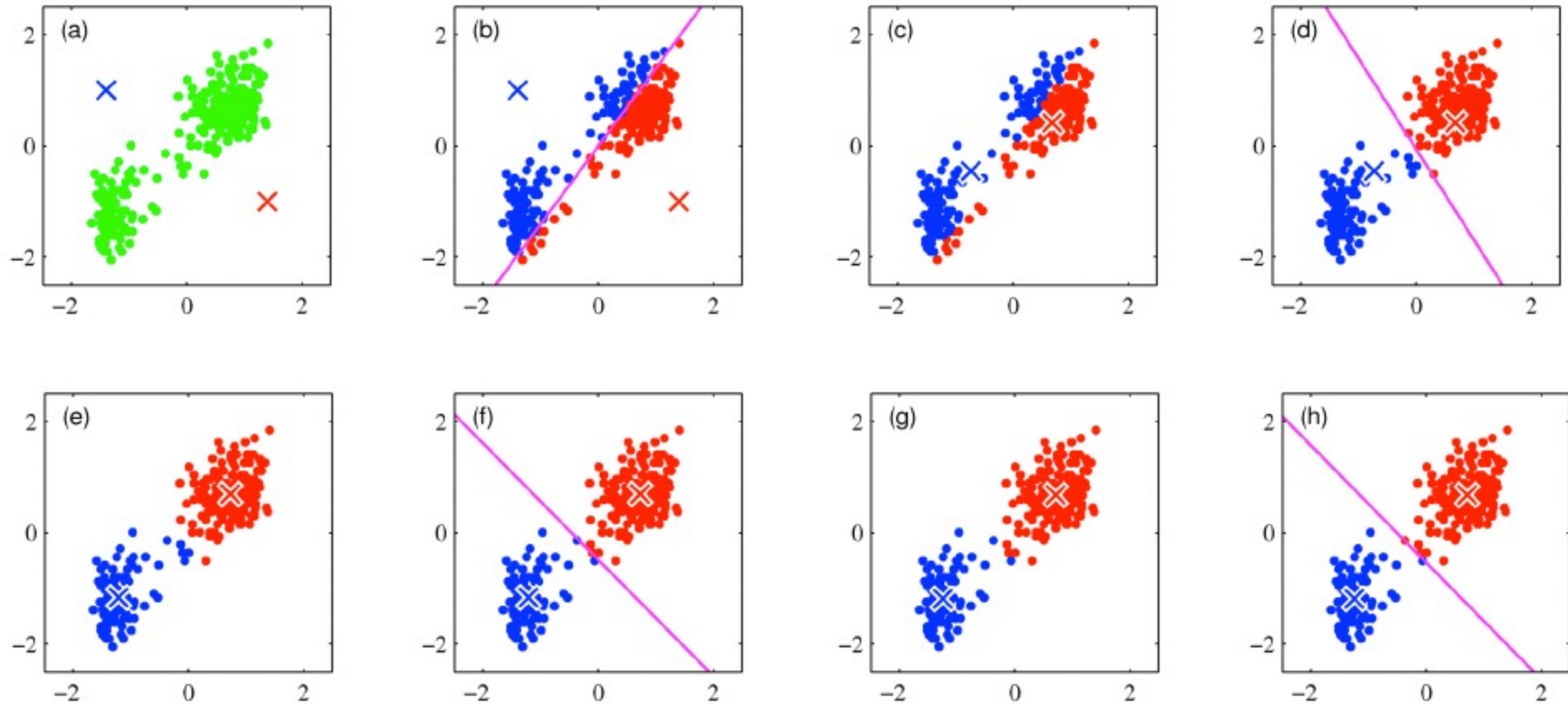
1D Example

Iterate these steps until means and assignments do not change any more



Discovering Groups – K-means

2D Example



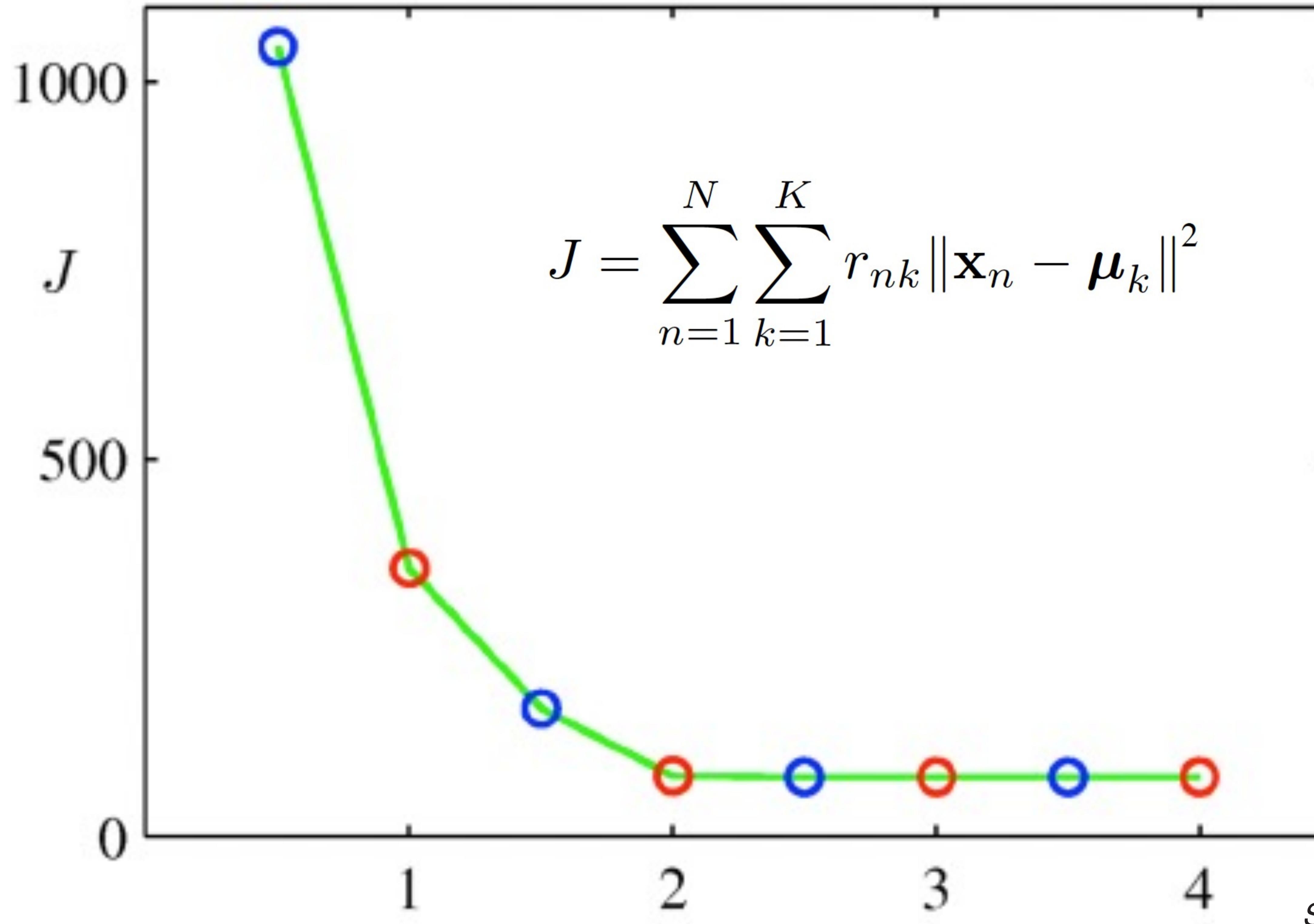
Source: D. Cremers

- Real data set
- Random initialization
- Magenta line is ‘decision boundary’

Source: D. Cremers

Discovering Groups – K-means

Sum of Squared Error (SSE) Curve



Source: D. Cremers

- After every step the cost function J is minimized
- Blue steps: update assignments
- Red steps: update means
- Convergence after 4 rounds

Discovering Groups – K-means

K-means can quickly and cheaply cluster data.

Problems?

- need to specify the cluster number k
- depends on good initial centroid guesses
- may converge on local minimum
- assumes spherical data (or ellipsoid-shaped clusters, or at best convex clusters)

Gaussian Mixture models (GMM) can work better, using a generalization of K-means (assuming each cluster is Gaussian), not discussed in this lecture.

Discovering Groups – DBSCAN

- Idea: uses the **local density** of points to determine the clusters, rather than using only the distance between points

$$N_\epsilon(\mathbf{x}) = B_d(\mathbf{x}, \epsilon) = \{\mathbf{y} \mid \delta(\mathbf{x}, \mathbf{y}) \leq \epsilon\}$$

where $\delta(\mathbf{x}, \mathbf{y})$ represents the distance between \mathbf{x} and \mathbf{y} , ϵ indicates **Max radius**, and \mathbf{x} is a **core point** if $|N_\epsilon(\mathbf{x})| \geq minpts$, where $minpts$ is a **Min number** that is user-defined local density or frequency threshold

- \mathbf{x} belongs to a density-based cluster when

$$|N_\epsilon(\mathbf{x})| \geq minpts \text{ or } \mathbf{x} \in N_\epsilon(\mathbf{z})$$

where \mathbf{z} is another data point, $minpts$ is a **Min number** that is user-defined local density or frequency threshold

Max radius is the limit on which to look for neighbours

Min number is the lower limit on what can be in a cluster

Discovering Groups – DBSCAN

Algorithm 3: DBSCAN

Data: X , eps , min_pts

initialise labels list as zeros, count list, core list;

Find neighbours for each point, Find core points;
 $|N_\epsilon(\mathbf{x})| \geq \text{minpts}$
 $\text{class} = 1$;

for each core point p **do**

 add neighbours(p) to queue;

while queue not empty **do**

 neighbours = next(queue);

for q in neighbours **do**

 set label($q = \text{class}$);

if label(q) is 'core' **then**

 add neighbours(q) to queue

end

end

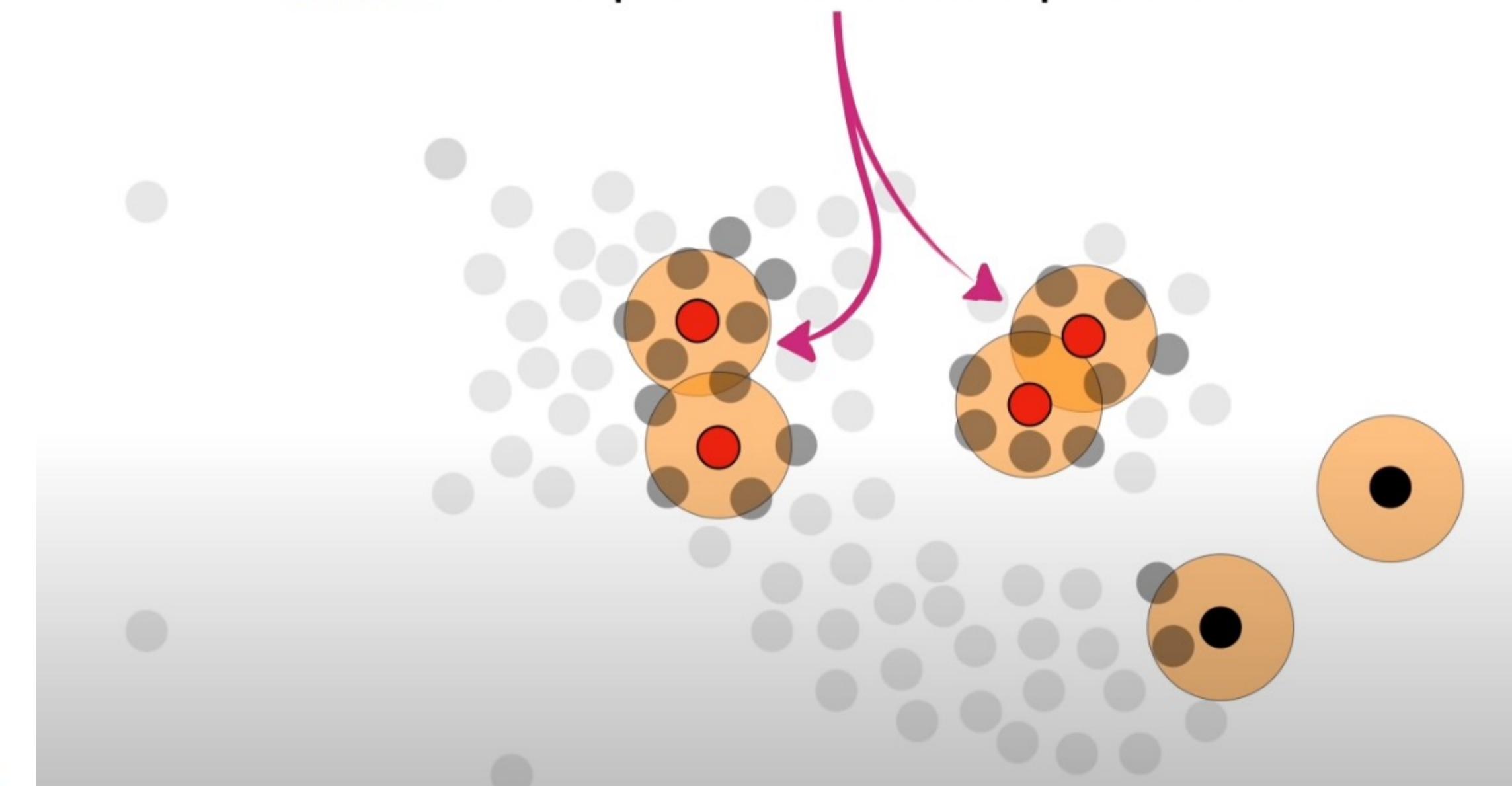
end

$\text{class} = \text{class} + 1$

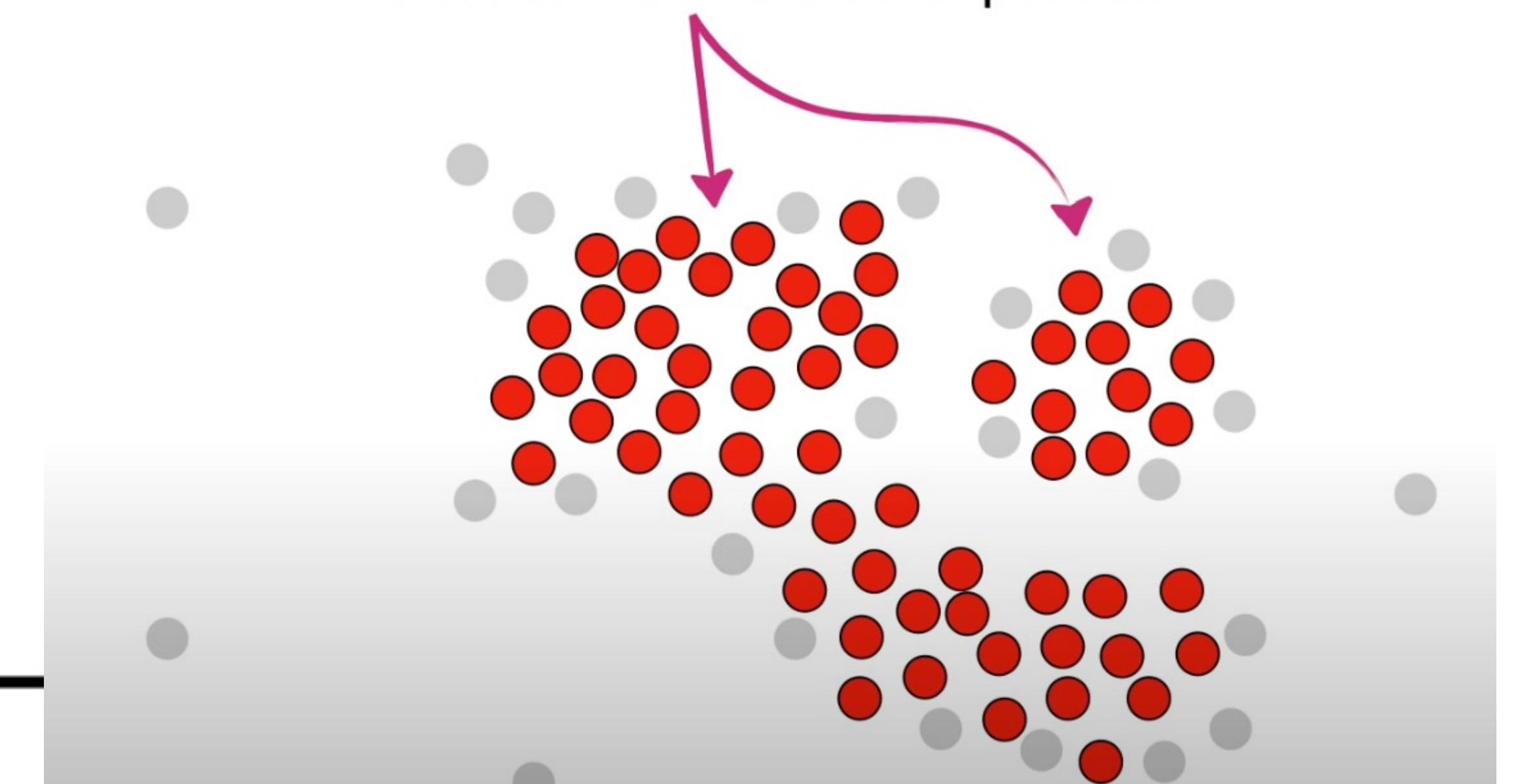
end

return labels;

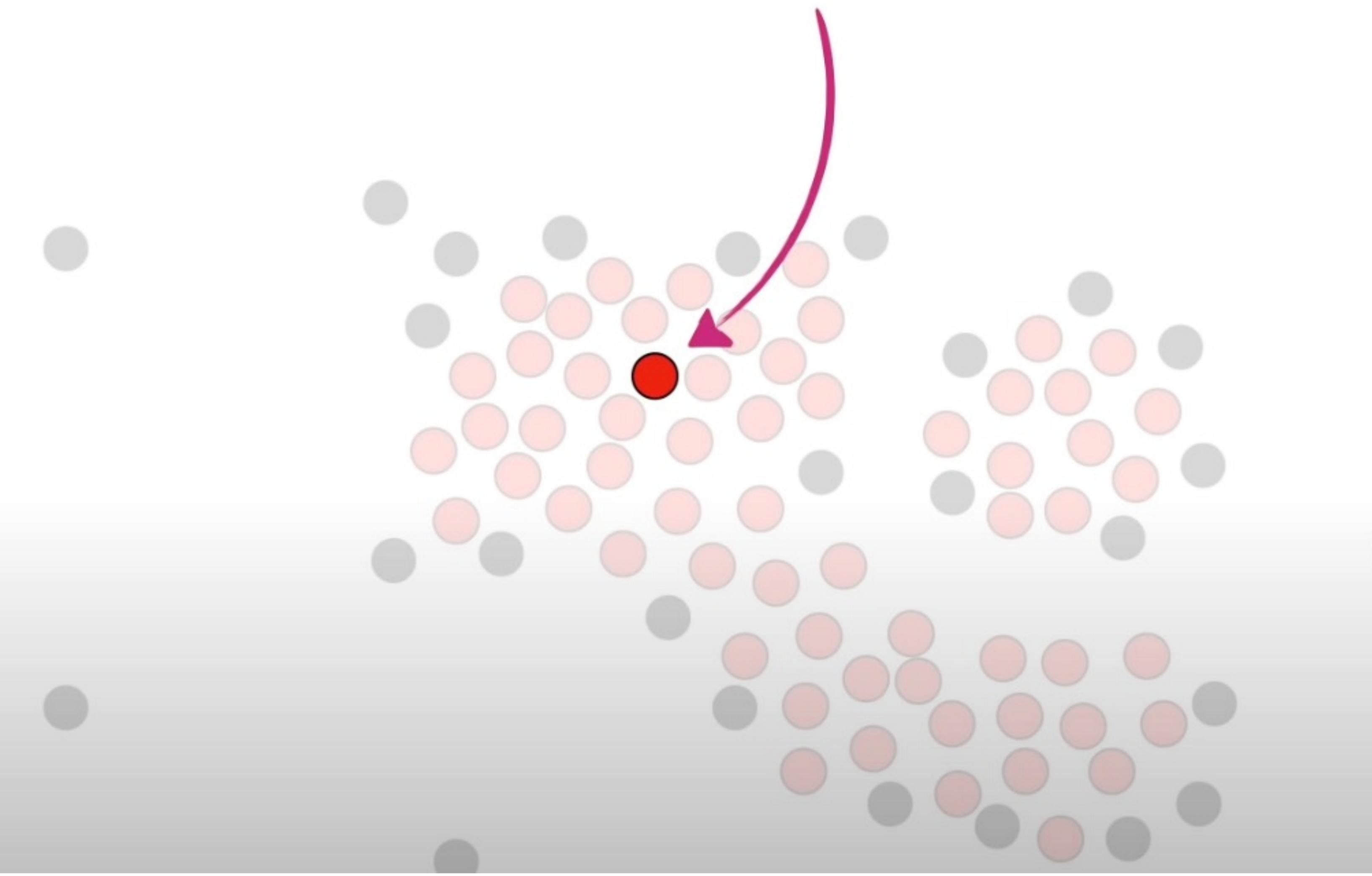
Anyway, these 4 points are some of the
Core Points, because their **orange circles** overlap at least 4 other points...



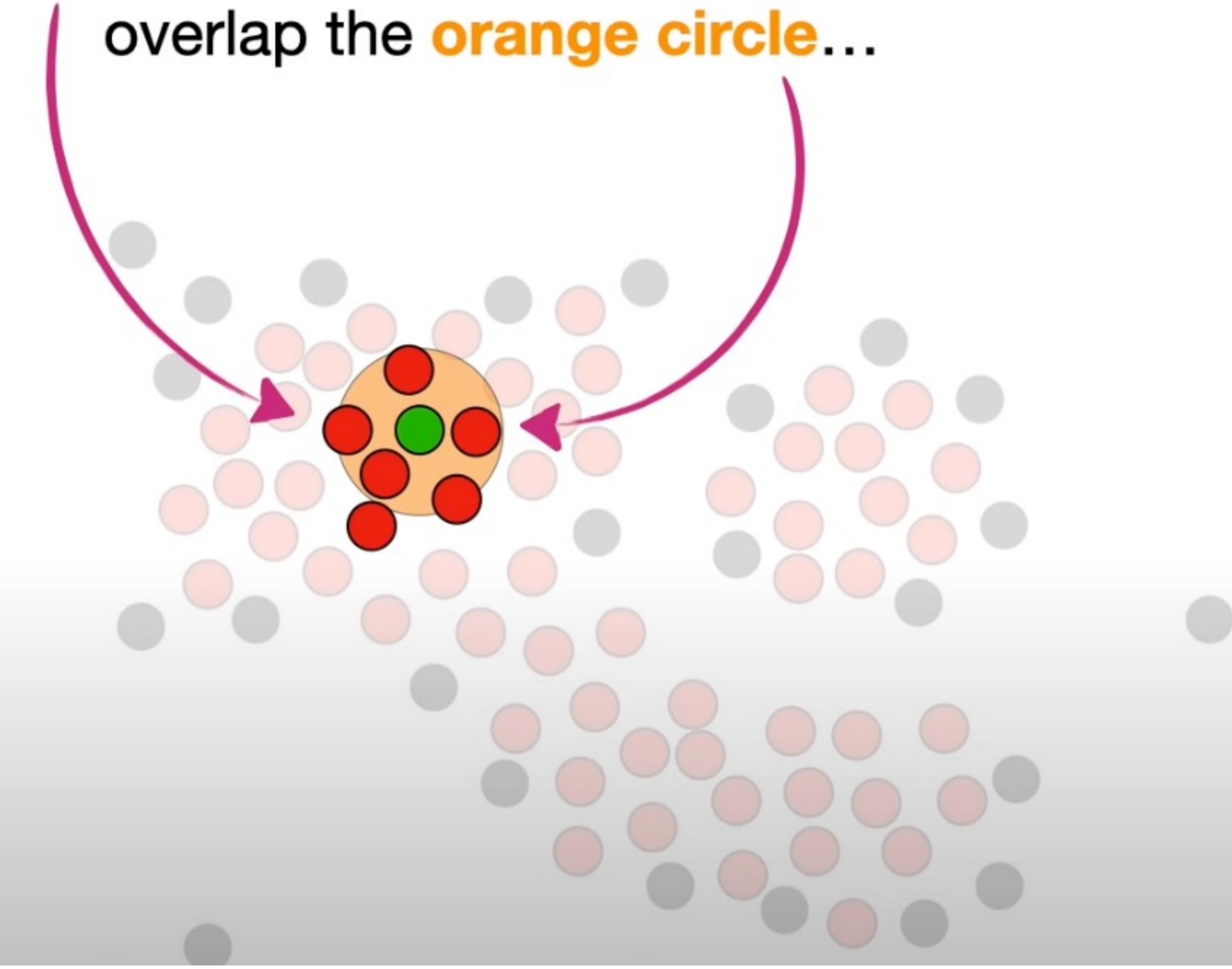
Ultimately, we can call all of these **red points Core Points** because they are all close to 4 or more other points...



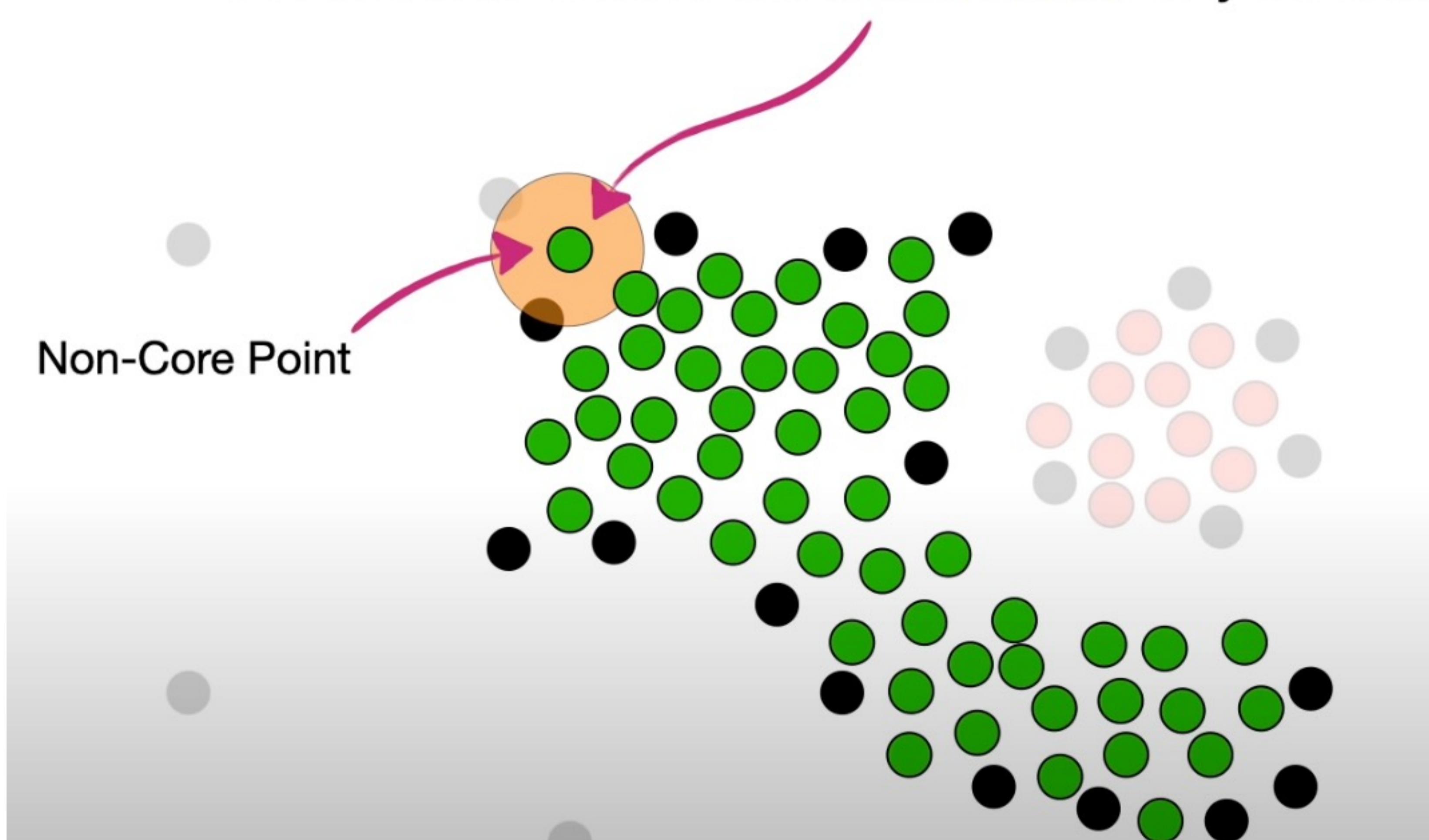
Now we randomly pick
a **Core Point**...



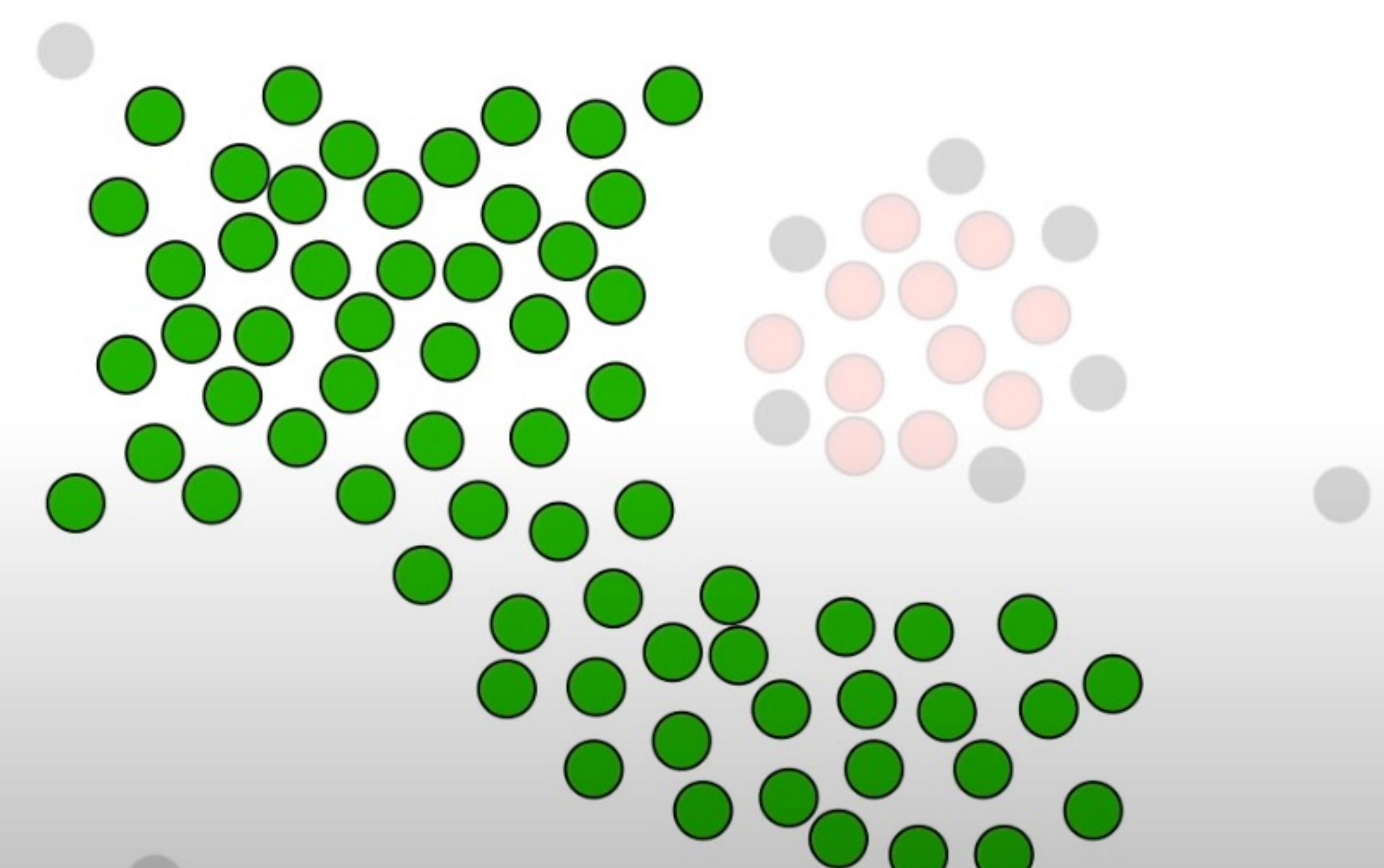
Next, the **Core Points** that are close
to the **first cluster**, meaning they
overlap the **orange circle**...



However, because this is not a **Core Point**, we do
not use it to extend the **first cluster** any further.



And now we are done
creating the **first cluster**.



Discovering Groups – DBSCAN

DBSCAN works well on any shape of data and is robust to outliers.

Problems?

- can struggle in high dimensions
- needs a distance parameter
- same parameter may not work for different cluster density
- need also minimum number specified

Discovering Groups – Hierarchical Clustering

Creates a **binary tree** that **recursively groups** pairs of similar items or clusters

Can be:

- Agglomerative (bottom up)
- Divisive (top down)

We will look at Agglomerative clustering. Needs a distance measure.

Discovering Groups – Hierarchical Clustering

Algorithm 4: Hierarchical Agglomerative Clustering

Data: N data points with feature vectors X_i $i = 1 \dots N$
 $numClusters = N$;
while $numClusters > 1$ **do**
 cluster1, cluster2 = FindClosestClusters();
 merge(cluster1, cluster2);
end

The distance between the clusters is evaluated using a linkage criterion.

If each merge is recorded, a binary tree structure linking the clusters can be formed.

This gives a **dendrogram**

Discovering Groups – Hierarchical Clustering

Linkage criterion: A measure of dissimilarity between clusters

Centroid Based:

- Dissimilarity is equal to distance between centroids
- Needs numeric feature vectors

Distance-Based:

- Dissimilarity is a function of distance between items in clusters
- Only needs precomputed measure of similarity between items

We could compute a distance matrix between points

Discovering Groups – Hierarchical Clustering

Centroid based linkage:

- WPGMC: Weighted Pair Group Method with Centroids.
When two clusters are combined into a new cluster, the average of the two centroids is the new centroid
- UPGMC: Unweighted Pair Group Method with Centroids.
When two clusters are combined into a new cluster, the new centroid is recalculated based on the positions of the items

Discovering Groups – Hierarchical Clustering

Distance based linkage:

- ▶ **Minimum, or single-linkage clustering** Distance between two closest members

$$\min d(a, b) : a \in A, b \in B$$

Produces long, thin clusters

- ▶ **Maximum, or complete-linkage clustering** Distance between two most distant members

$$\max d(a, b) : a \in A, b \in B$$

Finds compact clusters, approximately equal diameter

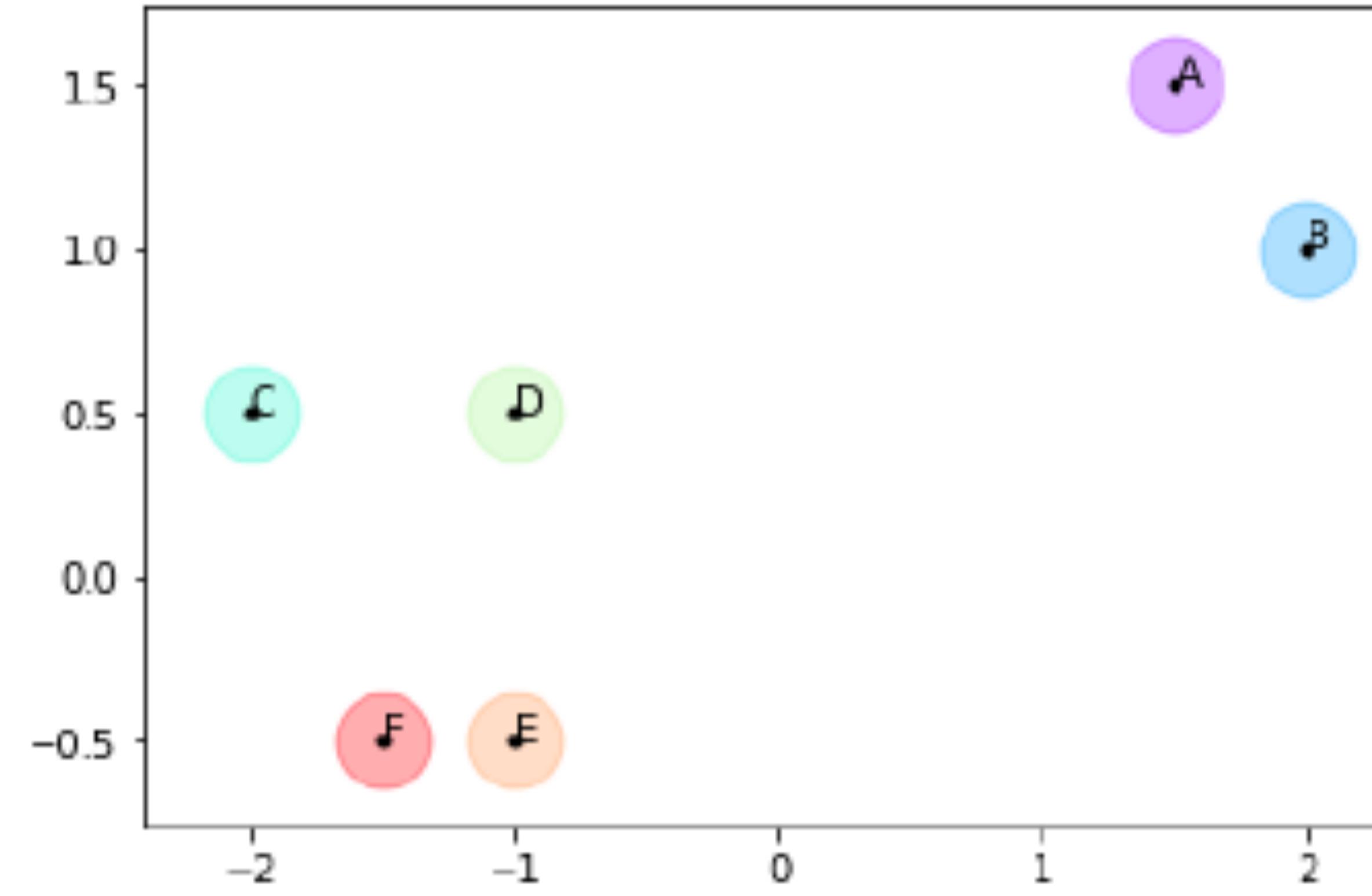
- ▶ **Mean or Average Linkage Clustering (UPGMA: Unweighted Pairwise Group Method with Arithmetic Mean):**

$$\frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$$

Discovering Groups – Agglomerative Clustering

With sample data:

$$X = \begin{bmatrix} 1.5 & 1.5 \\ 2.0 & 1.0 \\ 2.0 & 0.5 \\ -1.0 & 0.5 \\ -1.5 & -0.5 \\ -1 & 0.5 \end{bmatrix}$$



Distance matrix: *demo distances, not ground truth*

	A	B	C	D	E	F
A	0	0.7	2.7	1.8	...	
B	0.7	0	...			
C	2.7		0	...		
D	1.8			0	...	
F	:				0	...

Discovering Groups – Agglomerative Clustering

Algorithm 4: Hierarchical Agglomerative Clustering

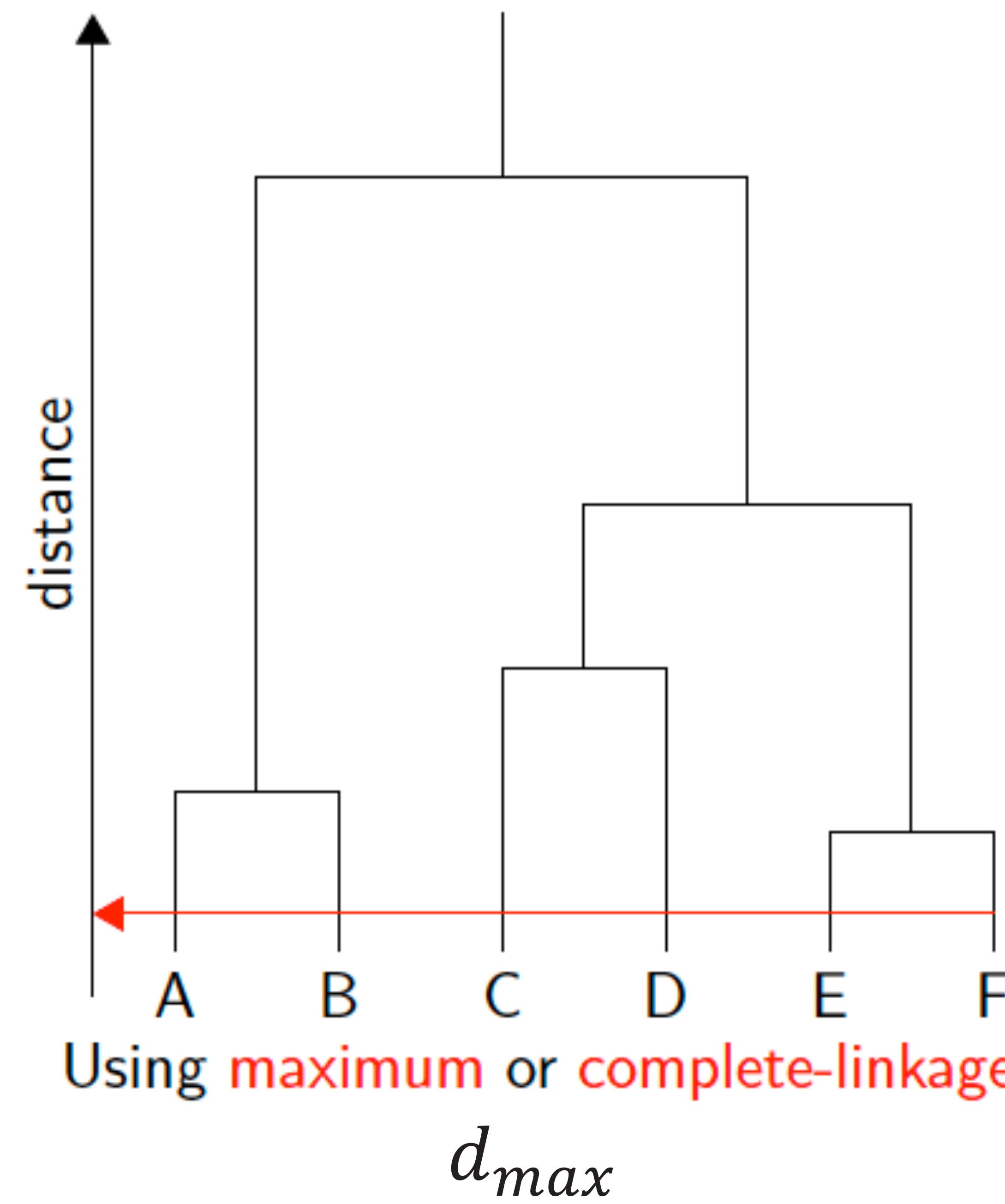
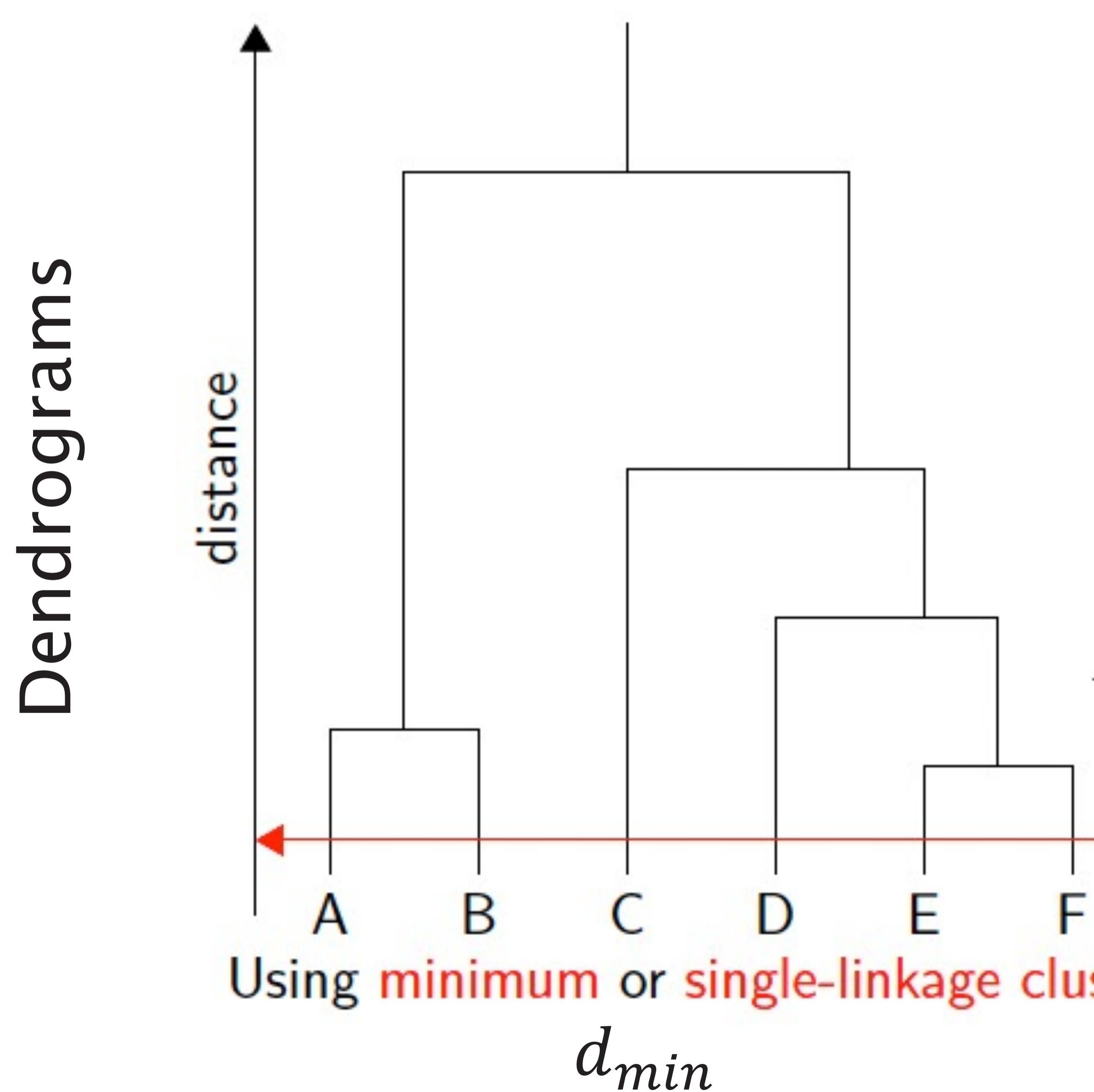
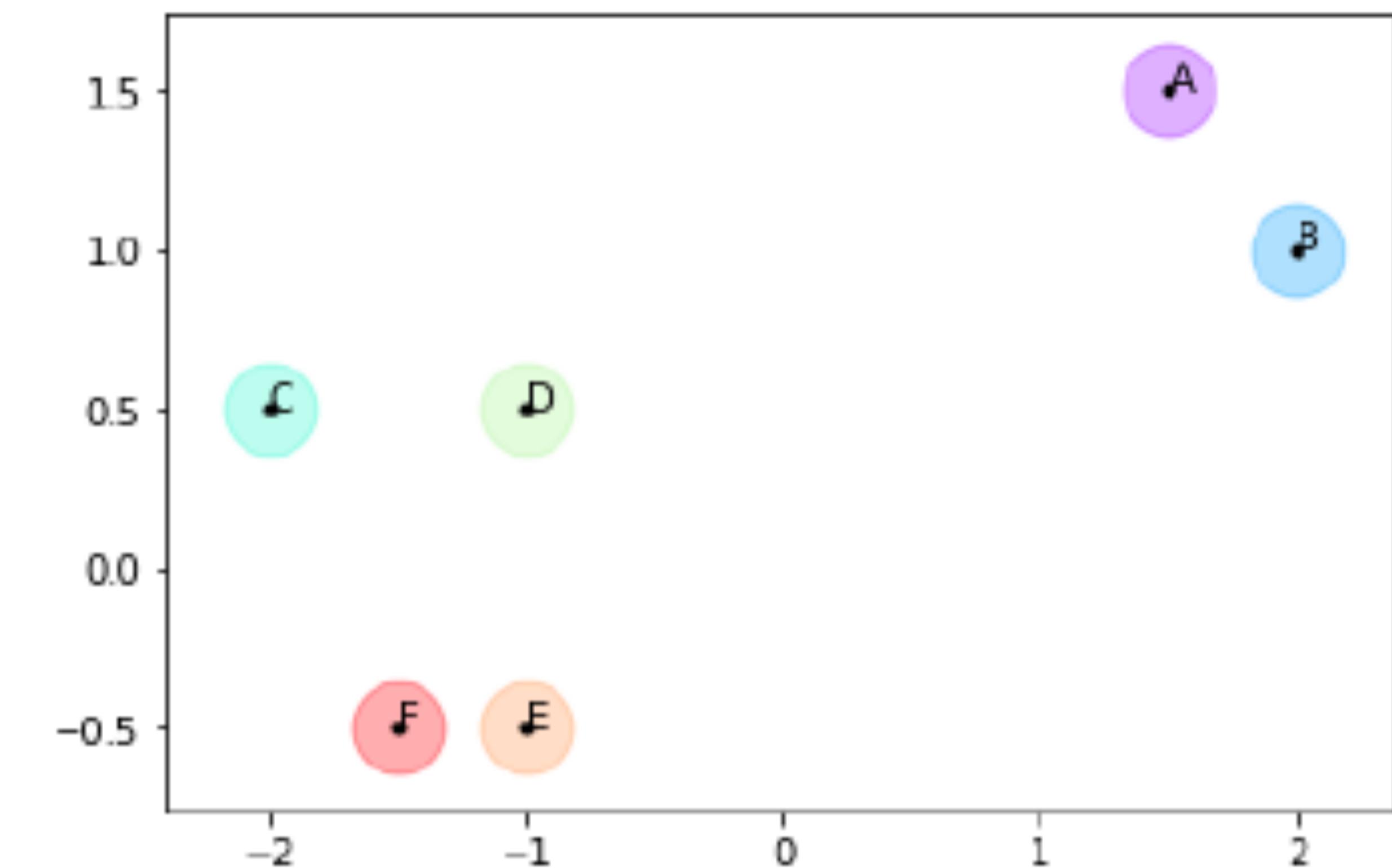
Data: N data points with feature vectors X_i $i = 1 \dots N$

$numClusters = N$;

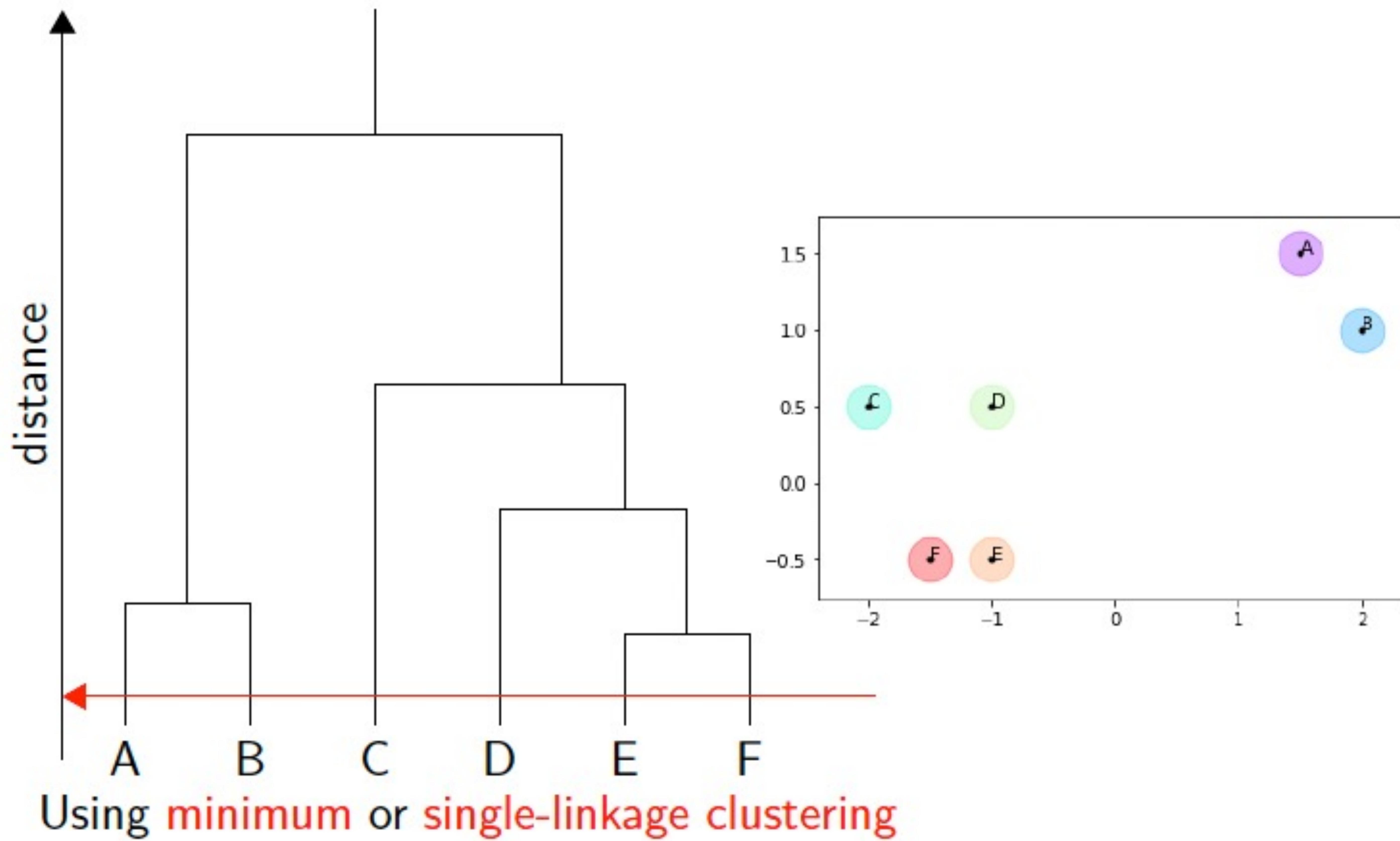
while $numClusters > 1$ **do**

- | cluster1, cluster2 = FindClosestClusters();
- | merge(cluster1, cluster2);

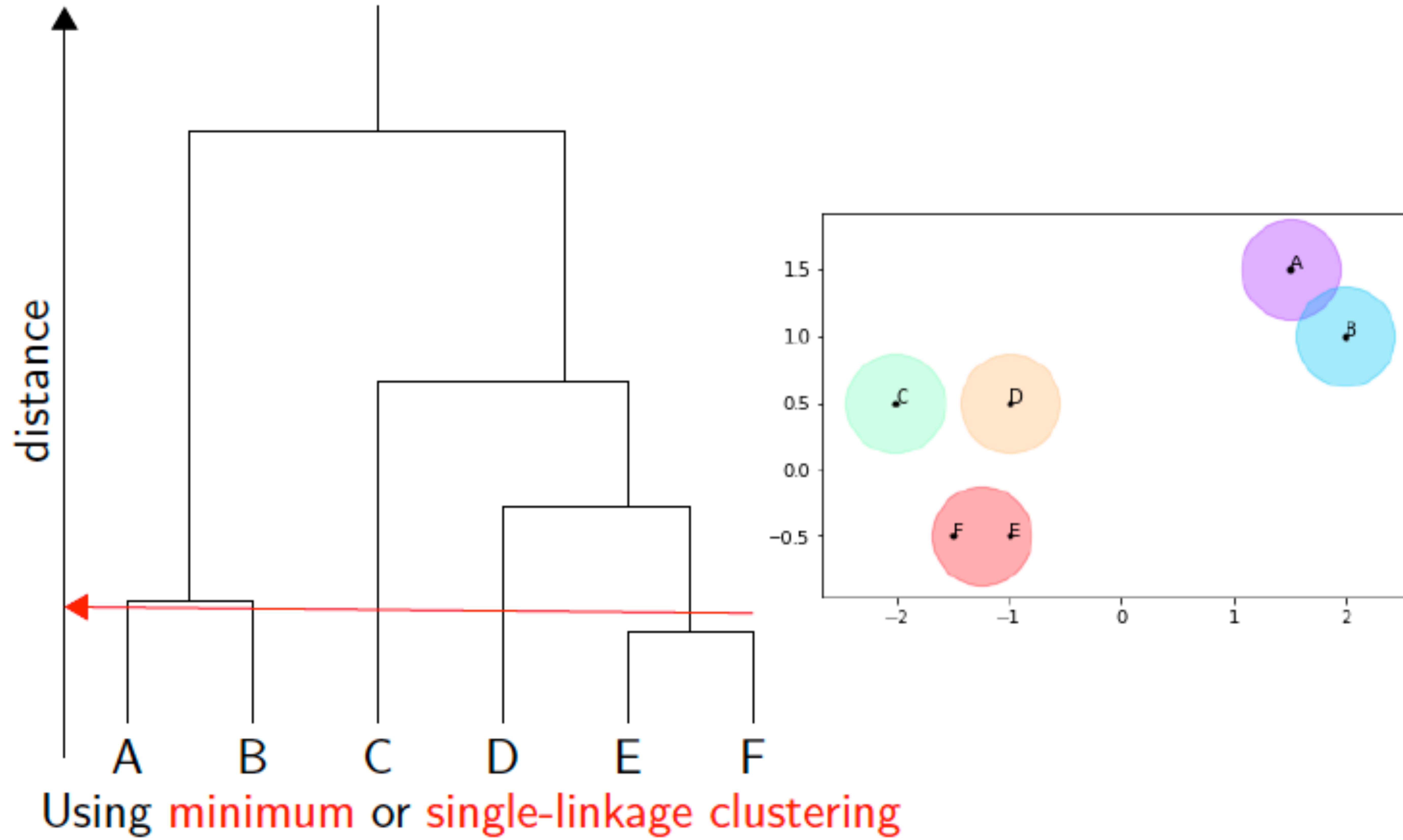
end



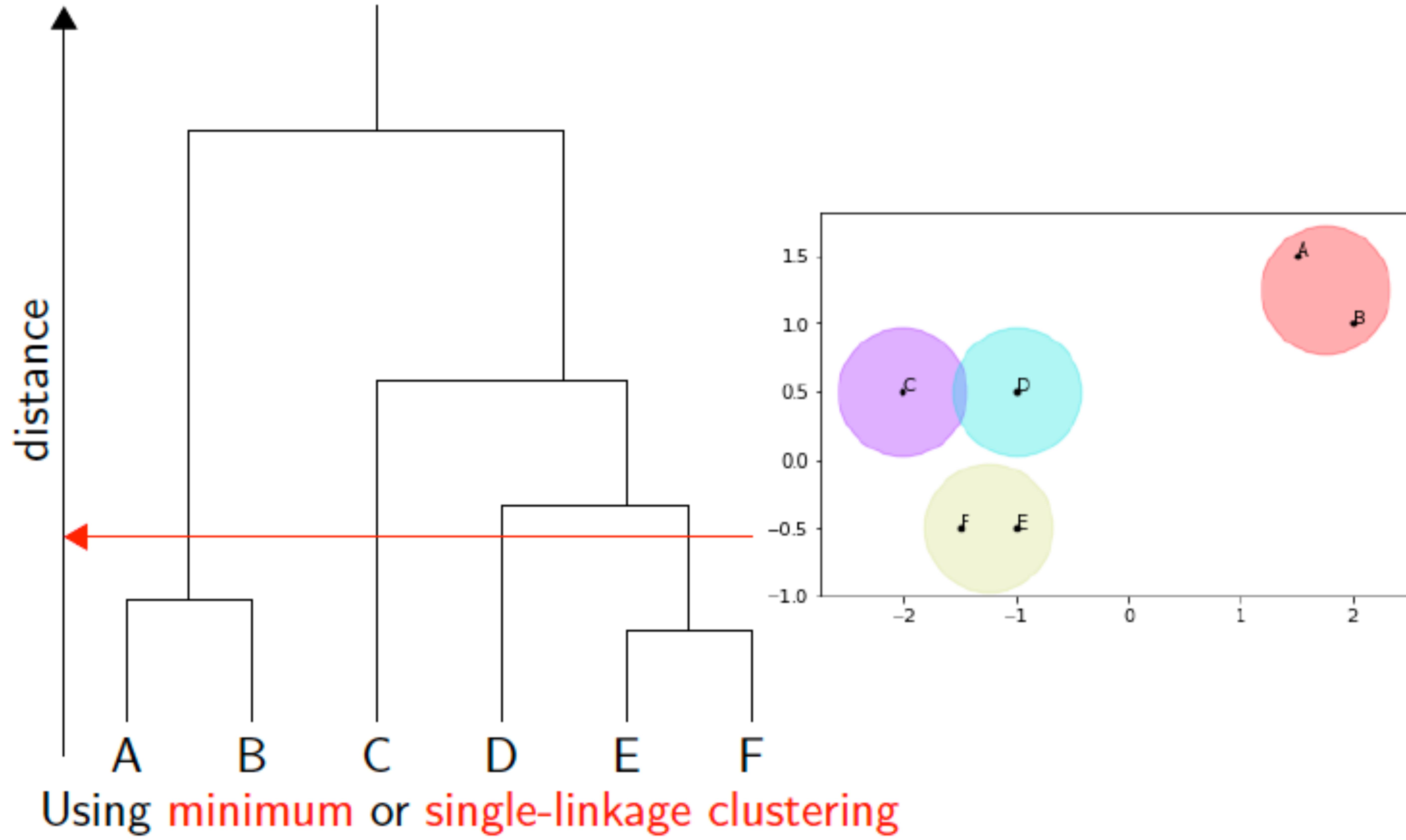
Discovering Groups – Agglomerative Clustering



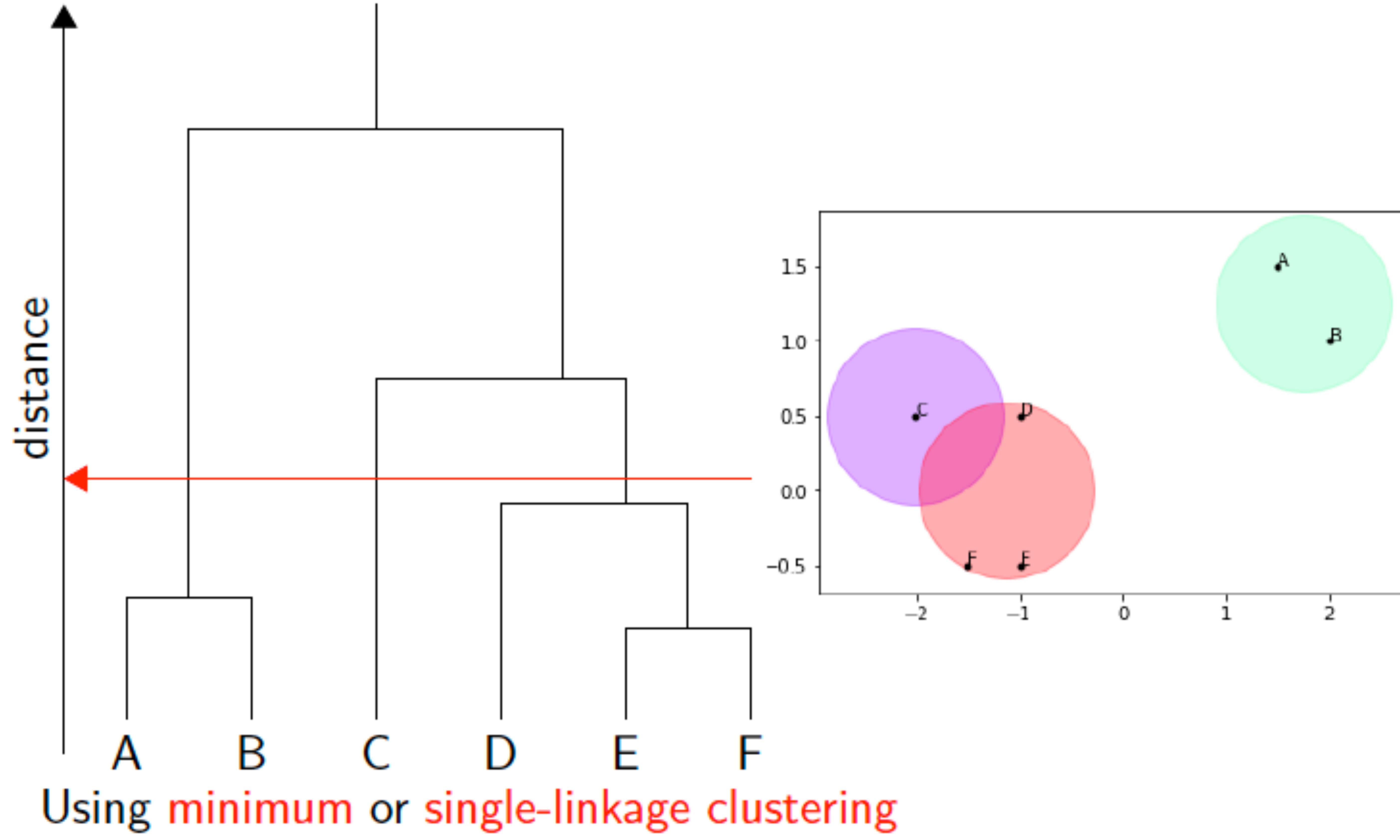
Discovering Groups – Agglomerative Clustering



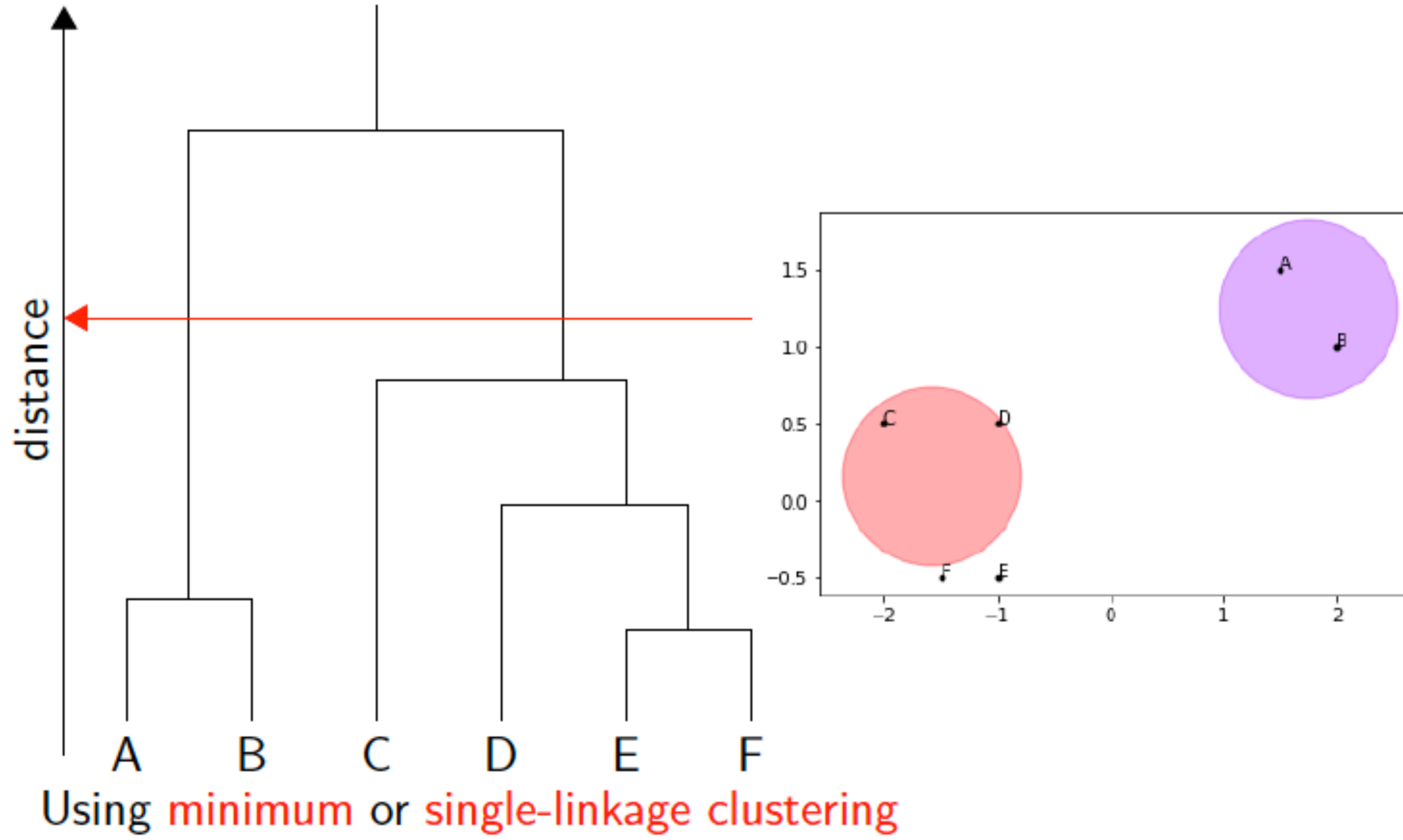
Discovering Groups – Agglomerative Clustering



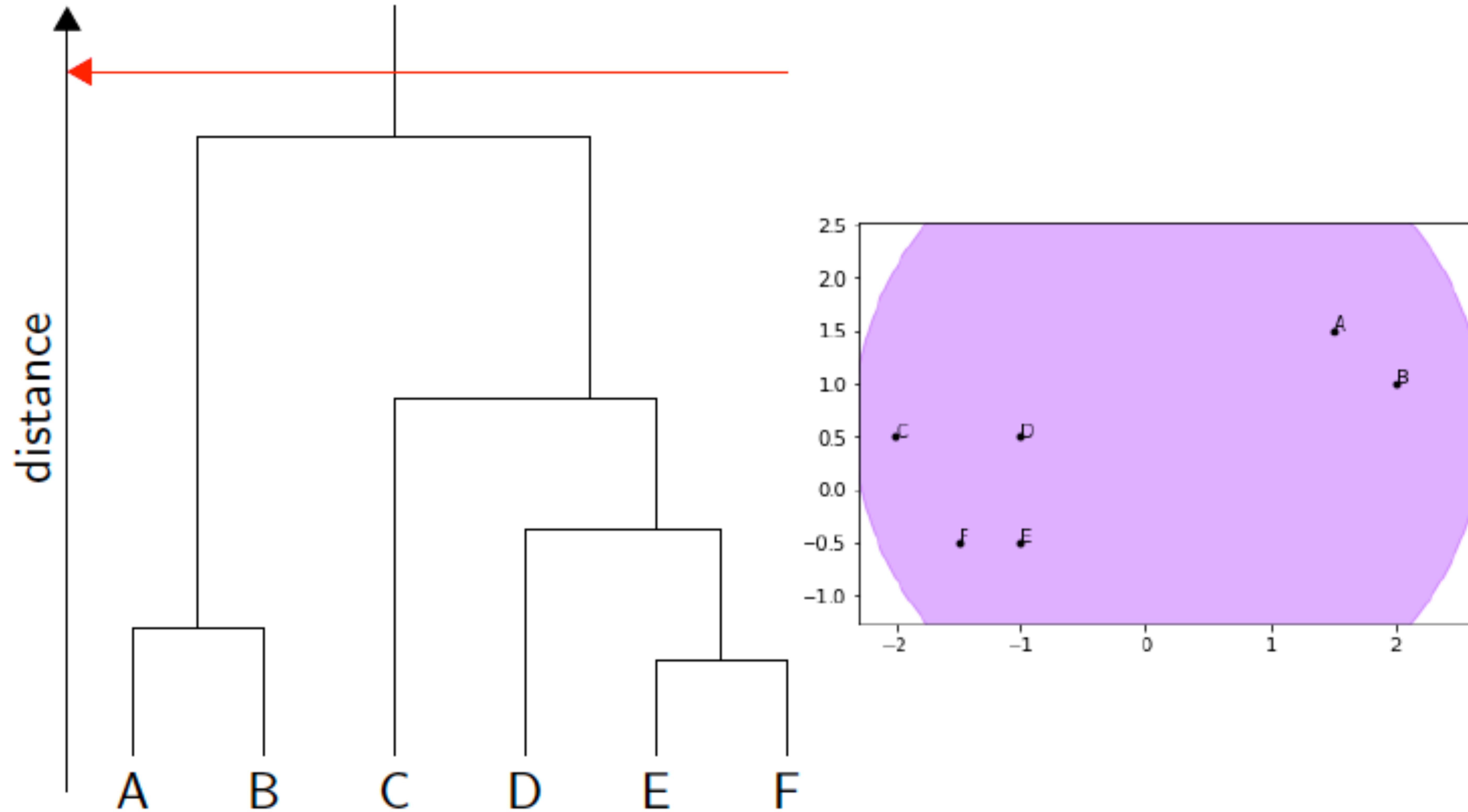
Discovering Groups – Agglomerative Clustering



Discovering Groups – Agglomerative Clustering

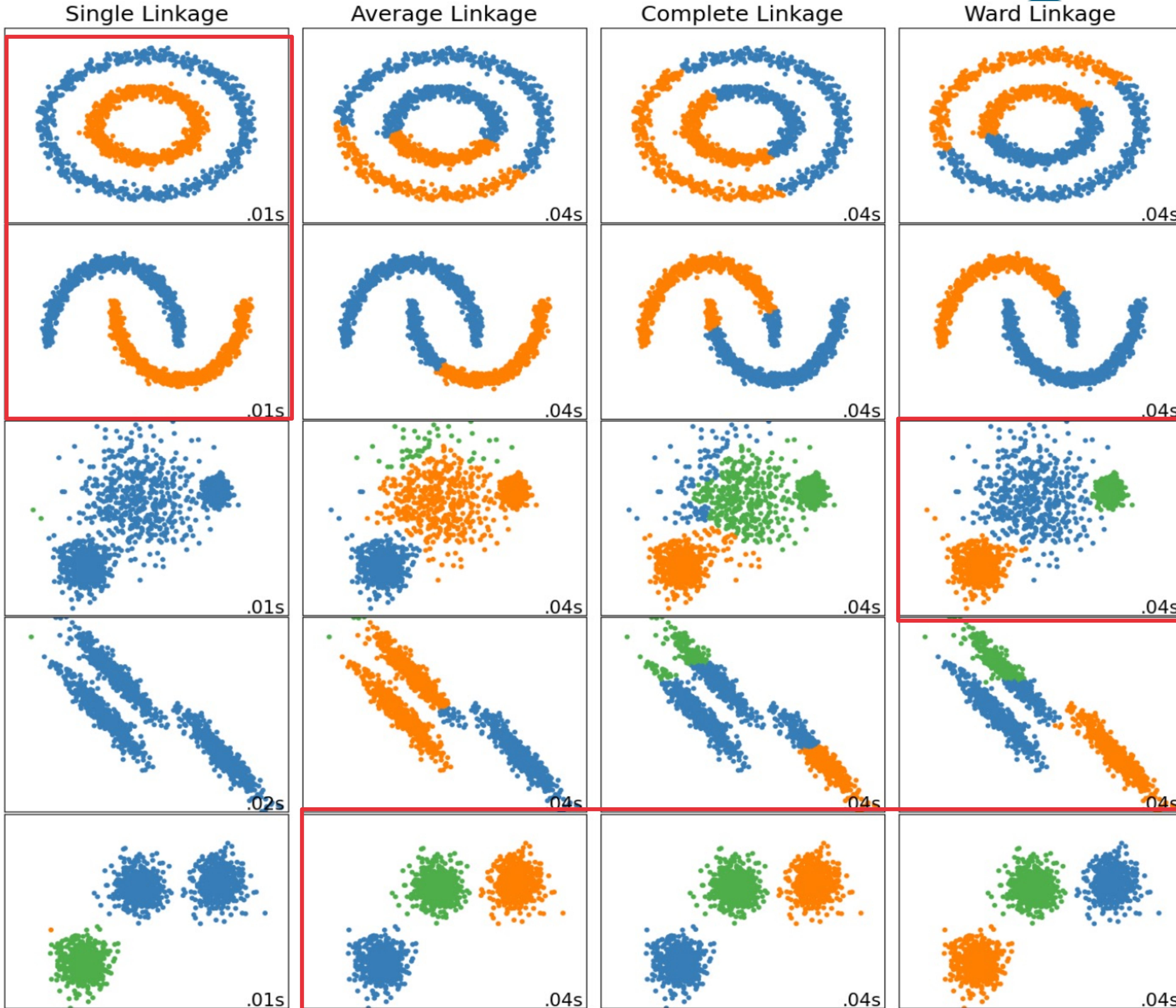


Discovering Groups – Agglomerative Clustering



Using minimum or single-linkage clustering

Source: [https://scikit-learn.org/stable/auto_ex.../plot-linkage-comparison.html#sphx-glr-auto-examples-cluster-plot-linkage-comparison-h](https://scikit-learn.org/stable/auto_examples/cluster/plot_linkage_comparison.html#sphx-glr-auto-examples-cluster-plot-linkage-comparison-h)



Discovering Groups – Hierarchical Clustering

Pros:

- No need to pre-specify cluster numbers; cut the dendrogram at the desired level for the clusters.
- Dendograms easily summarize data into a hierarchy, facilitating cluster examination and interpretation.

Cons:

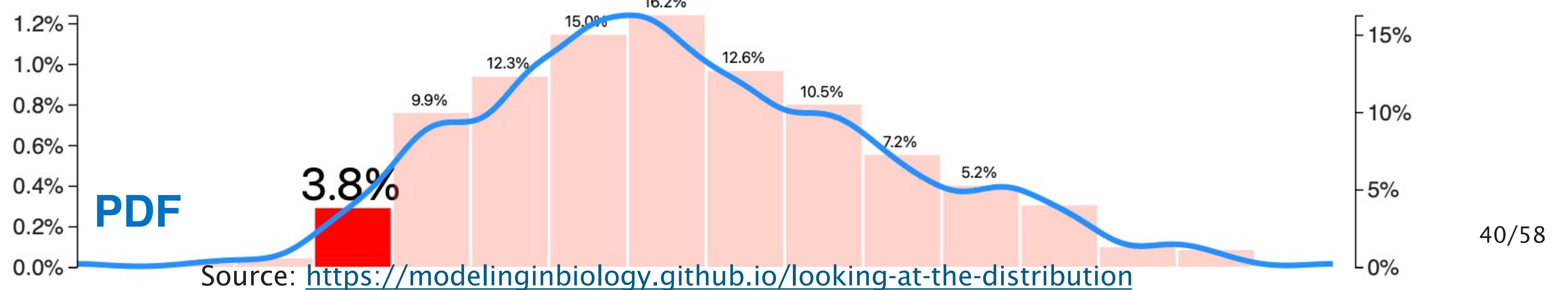
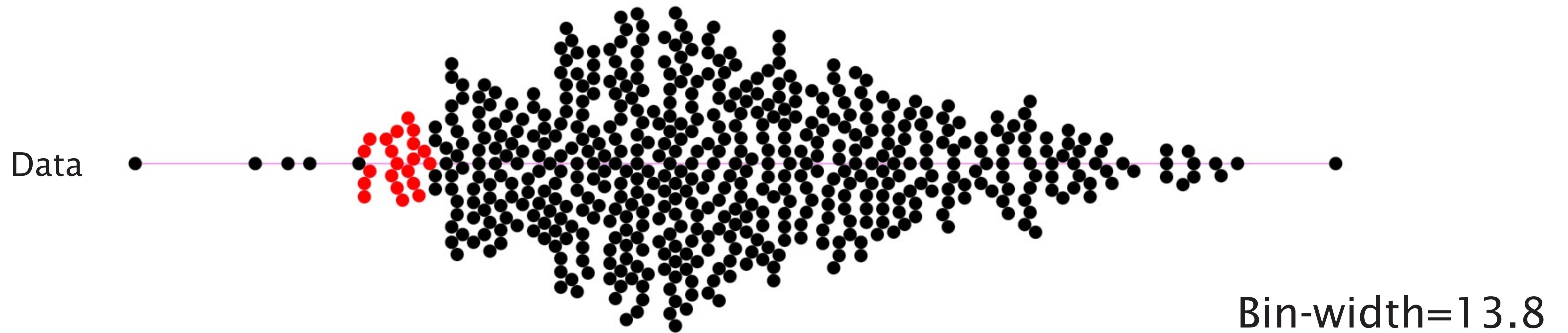
- Needs a threshold to determine the number of clusters
- Non-trivial to select the best linkage method

Discovering Groups – Mean Shift Clustering

Idea: The mean shift algorithm seeks a mode or local maximum of **density of a given distribution (PDF)**!

How can we estimate the PDF?

Could use a histogram, need to guess number of bins (or bin width)

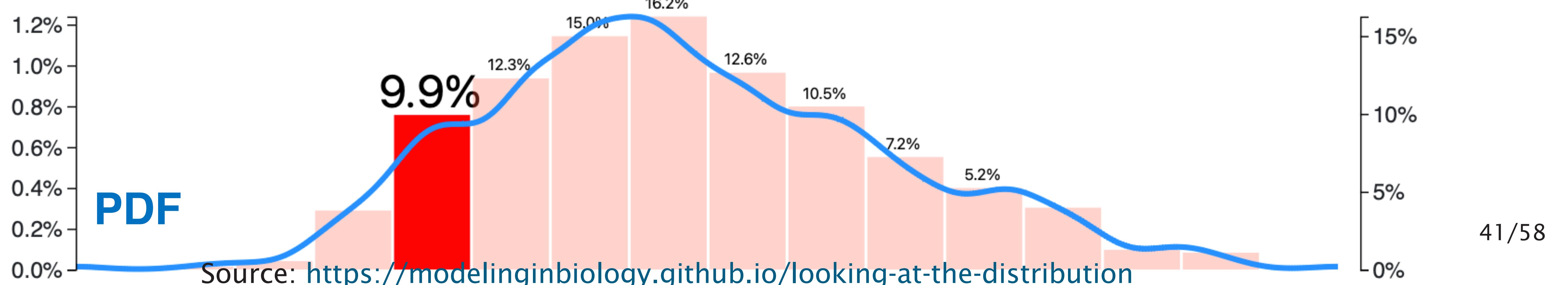
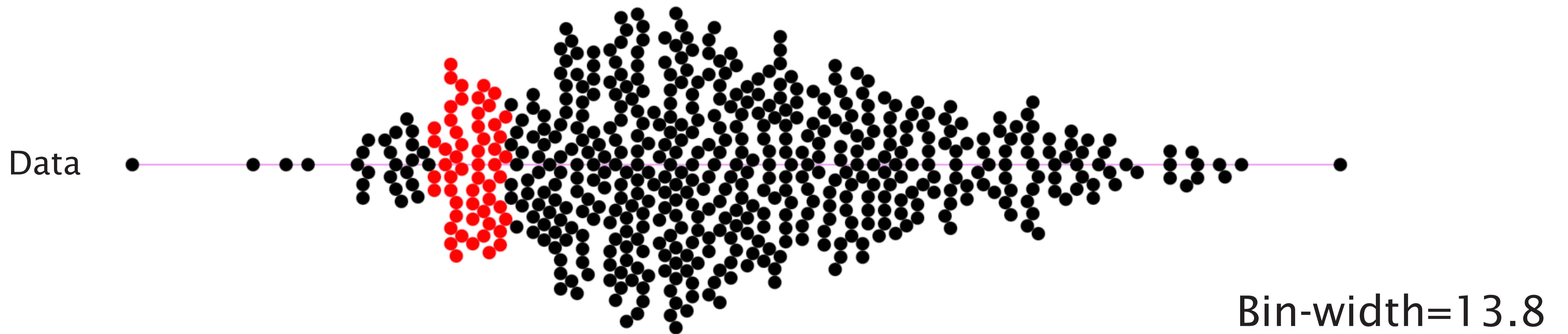


Discovering Groups – Mean Shift Clustering

Idea: The mean shift algorithm seeks a mode or local maximum of **density of a given distribution (PDF)**!

How can we estimate the PDF?

Could use a histogram, need to guess number of bins (or bin width)

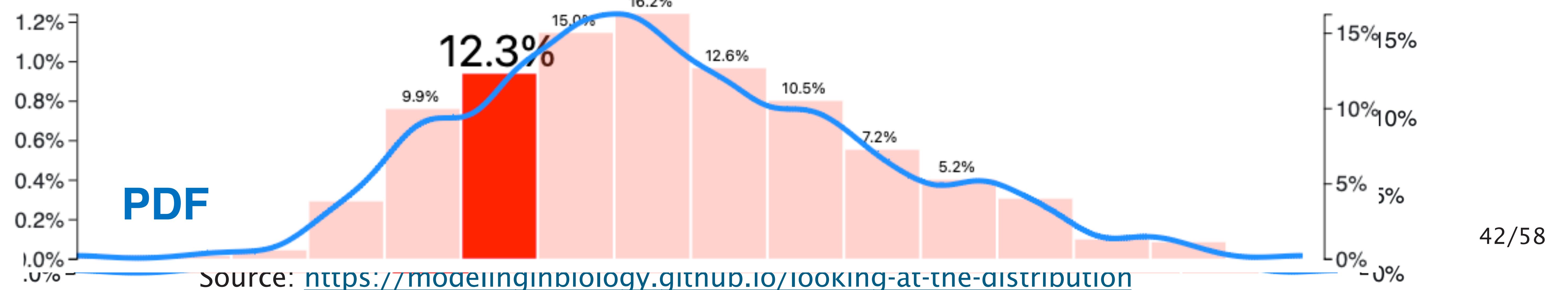
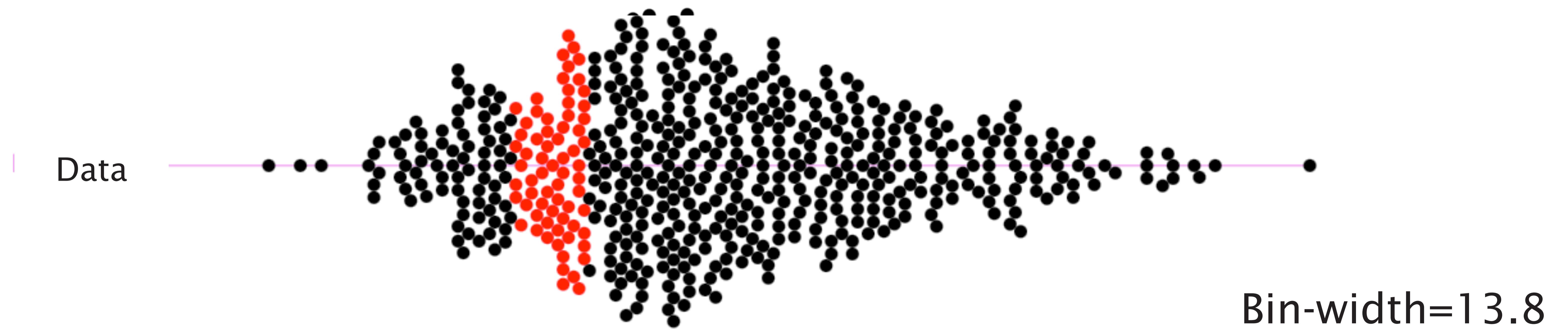


Discovering Groups – Mean Shift Clustering

Idea: The mean shift algorithm seeks a mode or local maximum of **density of a given distribution (PDF)**!

How can we estimate the PDF?

Could use a histogram, need to guess number of bins (or bin width)

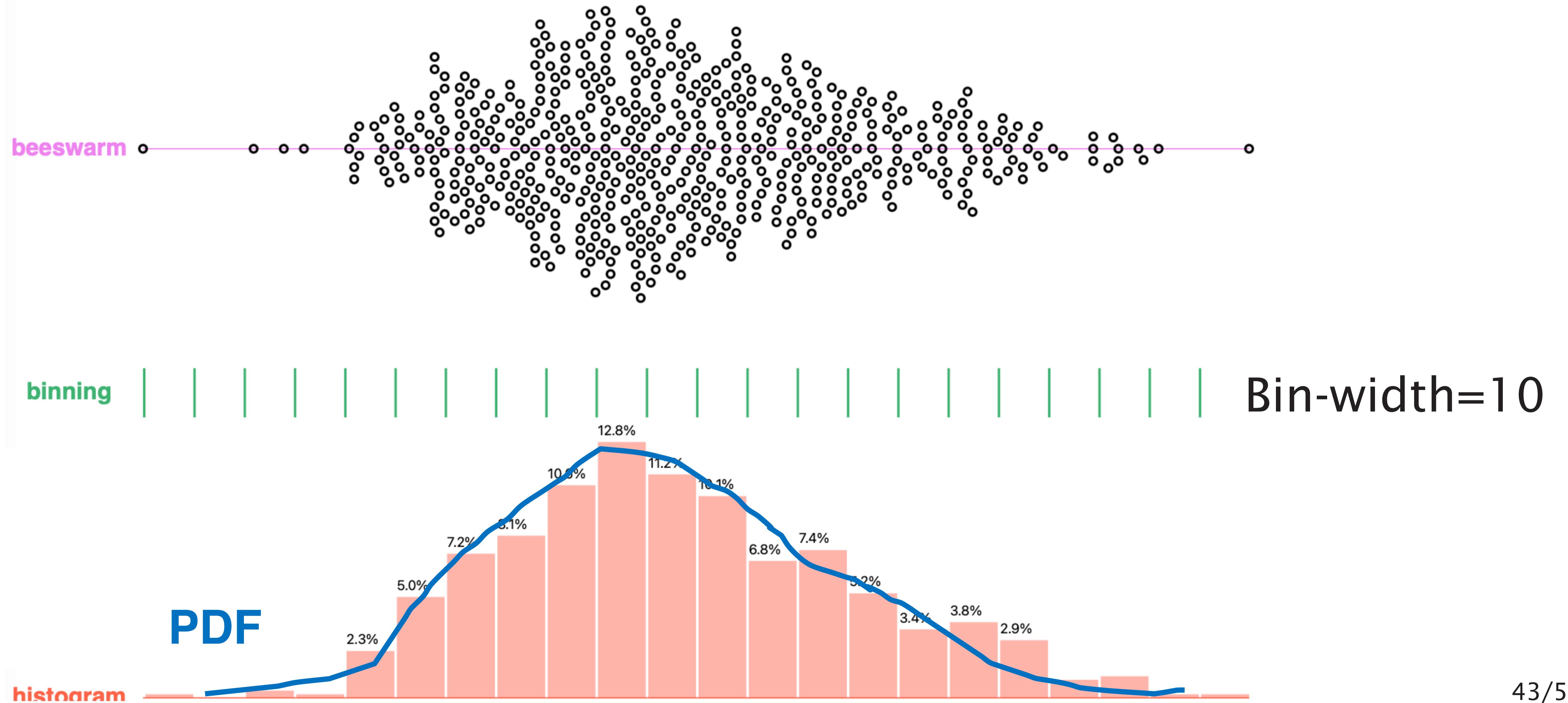


Discovering Groups – Mean Shift Clustering

Idea: The mean shift algorithm seeks a mode or local maximum of **density of a given distribution (PDF)!**

How can we estimate the PDF?

Could use a histogram, need to guess number of bins (or bin width)

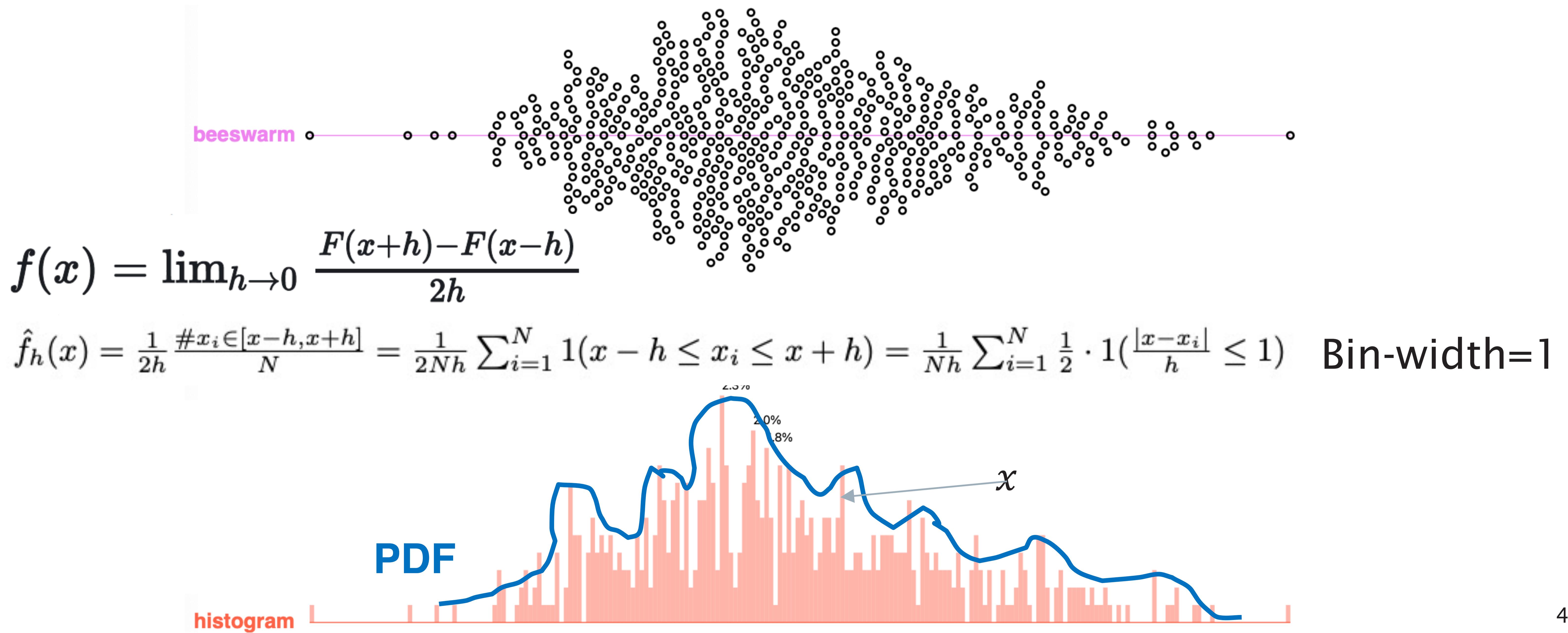


Discovering Groups – Mean Shift Clustering

Idea: The mean shift algorithm seeks a mode or local maximum of **density of a given distribution (PDF)**!

How can we estimate the PDF?

Could use a histogram, need to guess number of bins (or bin width)



Discovering Groups – Mean Shift Clustering

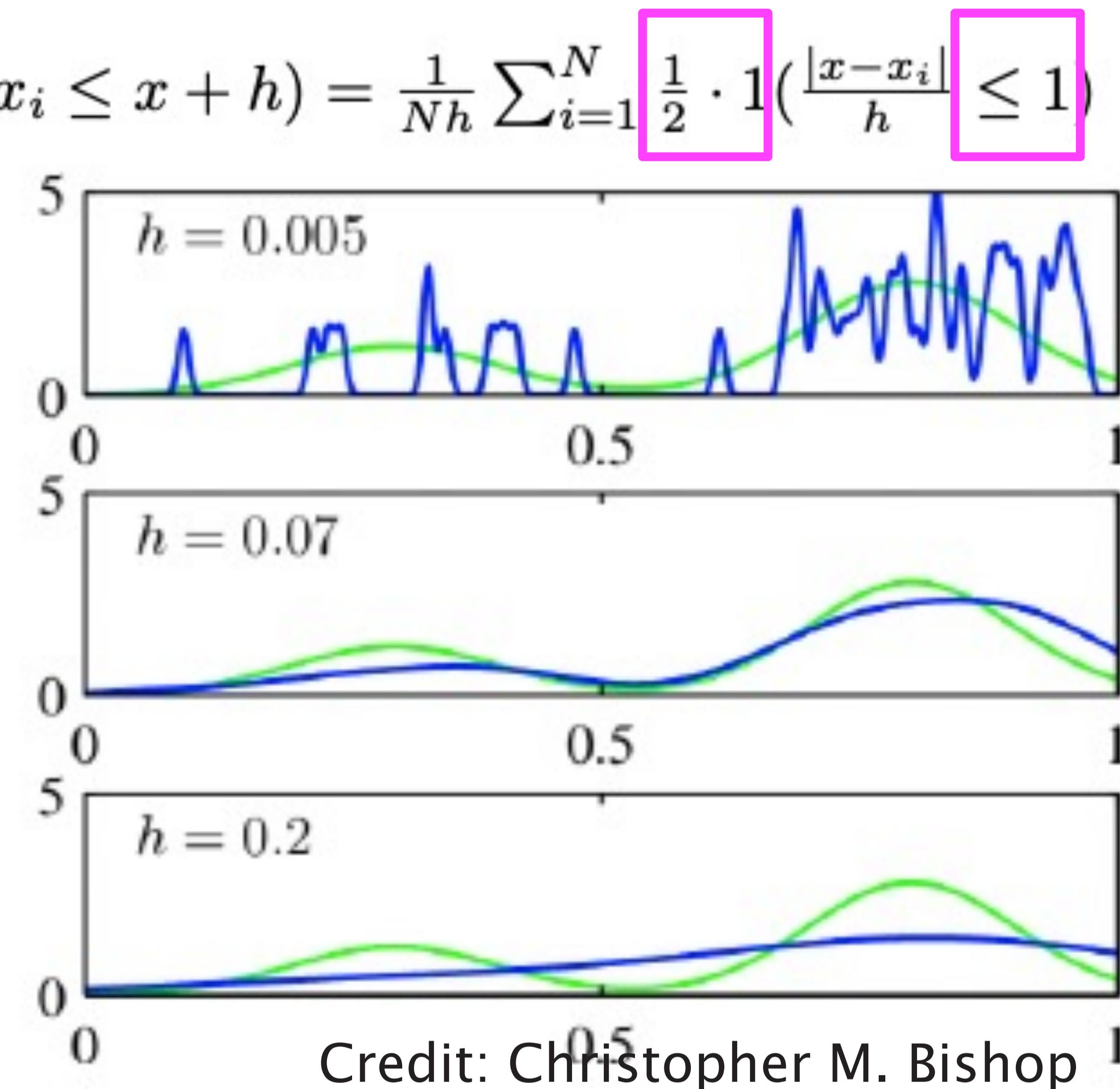
Kernel Density Estimation (aka Parzen Window)

Gives a smooth continuous estimate

$$\hat{f}_h(x) = \frac{1}{2h} \frac{\#\{x_i \in [x-h, x+h]\}}{N} = \frac{1}{2Nh} \sum_{i=1}^N \mathbf{1}(x-h \leq x_i \leq x+h) = \frac{1}{Nh} \sum_{i=1}^N \frac{1}{2} \cdot \mathbf{1}\left(\frac{|x-x_i|}{h} \leq 1\right)$$

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

Where nh is the number of items, d is the dimensionality of the feature space, K is the kernel function, x is an arbitrary position in feature space, h is the kernel bandwidth

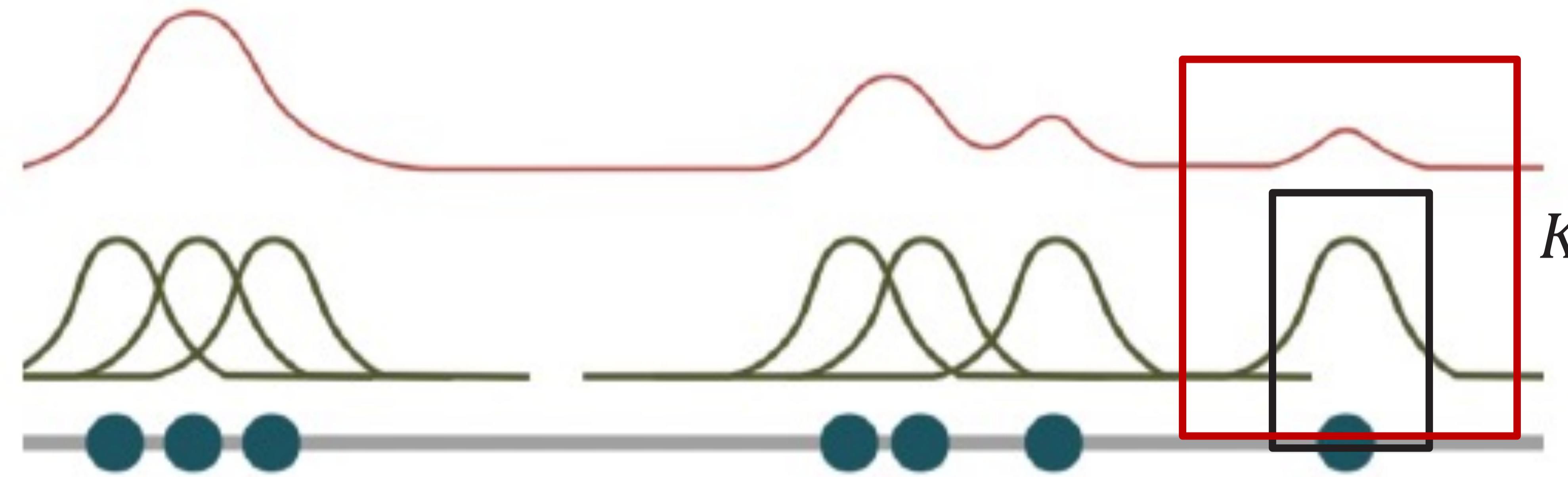


Changing bandwidth affecting accuracy of probability density estimation

Discovering Groups – Mean Shift Clustering

Usually use a Gaussian kernel with $\sigma = 1$, where $K(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|x\|^2}{2\sigma^2}}$

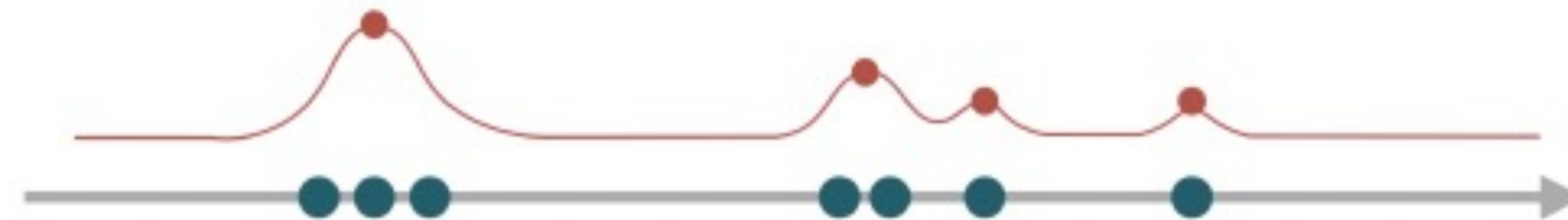
Convolve with the kernel function K to generate a smooth function $f(x)$
 Equivalent to superposition of multiple kernels centered at each data point



Credit: Christopher M. Bishop

Discovering Groups – Mean Shift Clustering

Find the modes of the probability density function (PDF), i.e. where the gradient is zero. $\Delta f(x) = 0$



Discovering Groups – Mean Shift Clustering

Mean shift algorithm:

Algorithm 5: Mean Shift Procedure

Data: N data points with feature vectors X_i $i = 1 \dots N$

Start with initial estimate - x ;

while x_t not $= x_{t+1}$ **do**

$m_h(x_t)$ = computeMeanShiftVect();
 $x_{t+1} = x_t + m_h(x_t);$

end

The mean shift algorithm seeks a mode or local maximum of density of a given distribution!

1. Compute the mean shift vector
2. Translate the kernel window
3. Repeat Step 1 until convergence

Discovering Groups – Mean Shift Clustering

Mean shift algorithm:

Algorithm 5: Mean Shift Procedure

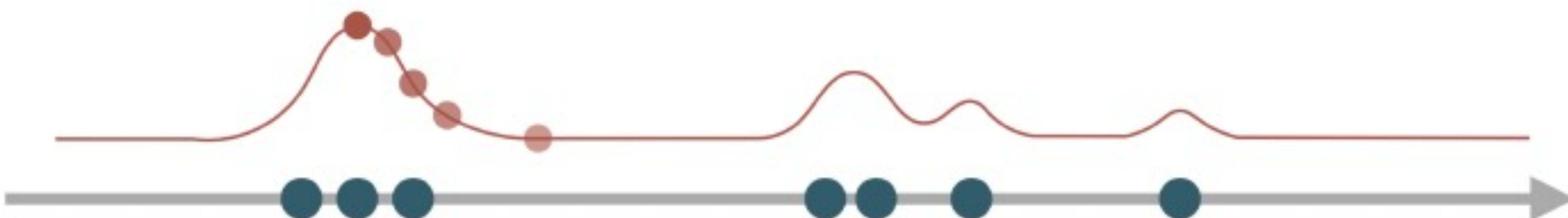
Data: N data points with feature vectors X_i $i = 1 \dots N$

Start with initial estimate - x ;

while $x_t \text{ not } = x_{t+1}$ **do**

$m_h(x_t)$ = computeMeanShiftVect();
 $x_{t+1} = x_t + m_h(x_t);$

end

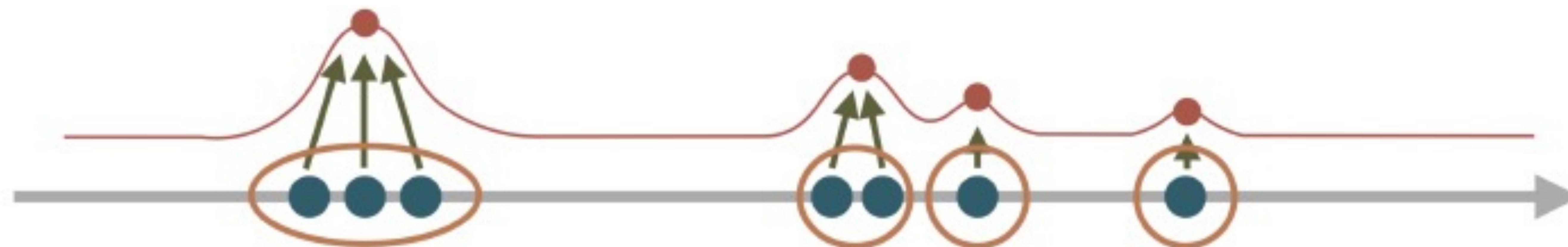


Discovering Groups – Mean Shift Clustering

For each feature vector:

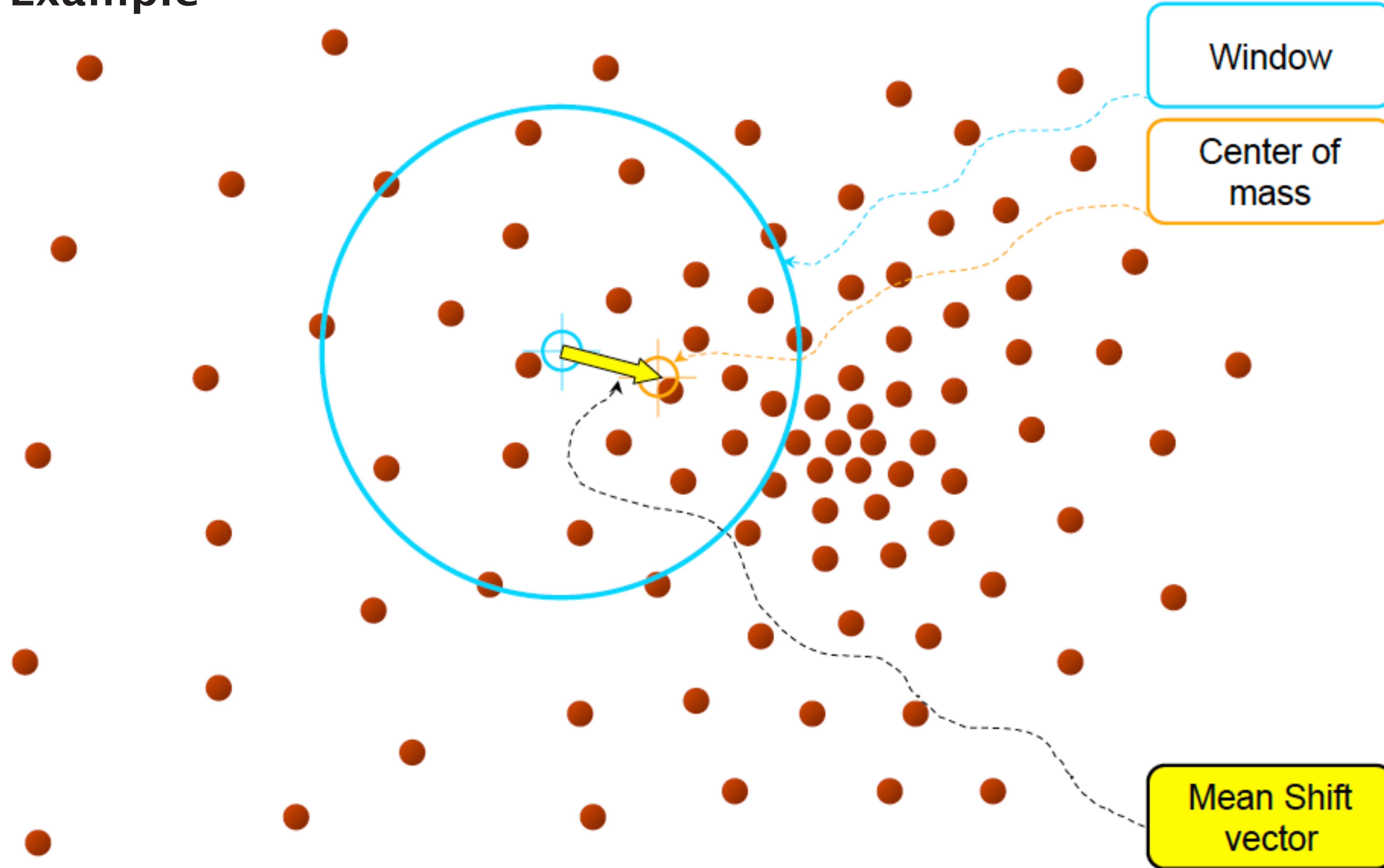
- ▶ apply mean shift procedure until convergence
- ▶ store resultant mode

Set of feature vectors that converge to the same mode define the basin of attraction of that mode



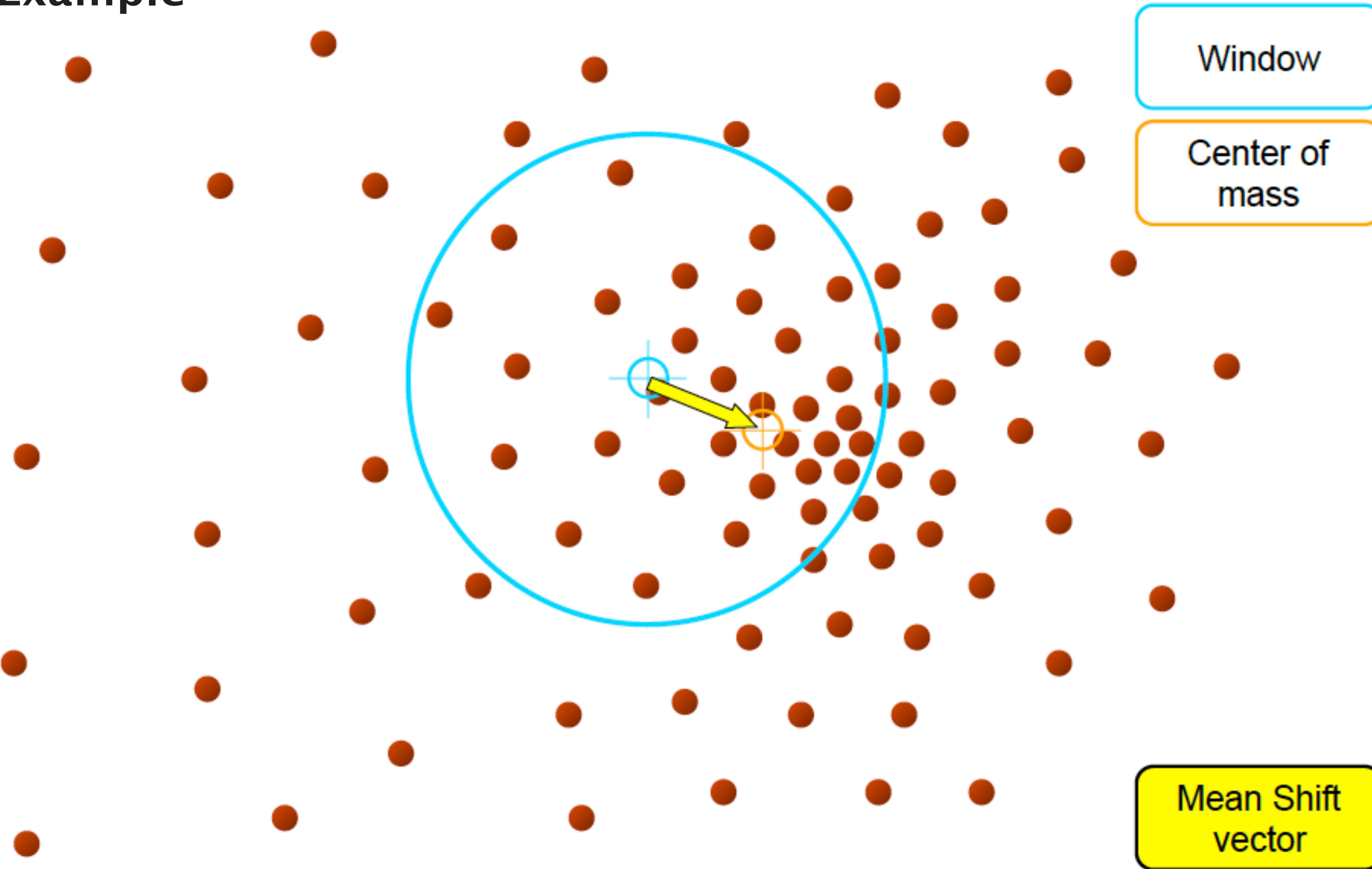
Discovering Groups – Mean Shift Clustering

2D Example



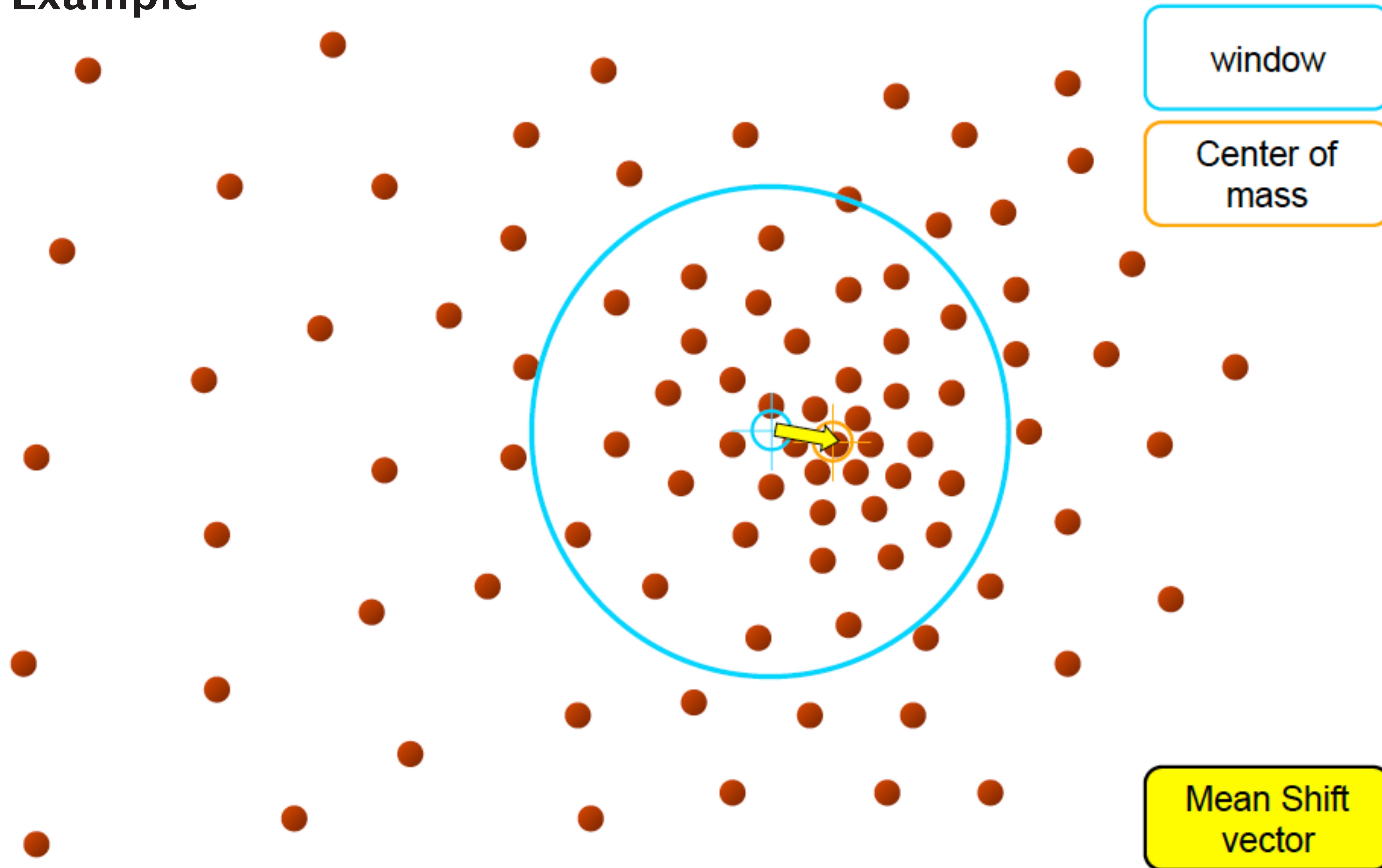
Discovering Groups – Mean Shift Clustering

2D Example



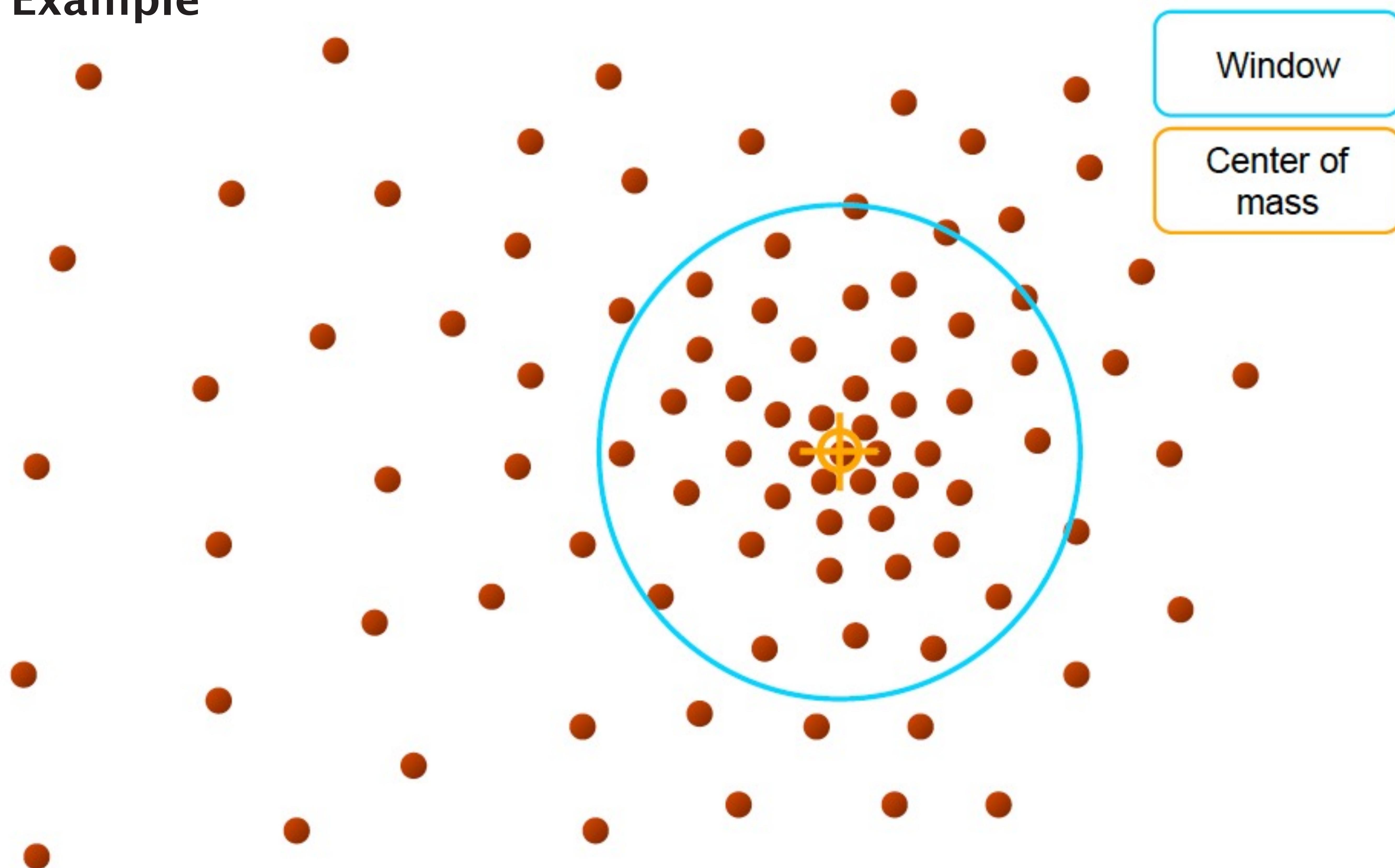
Discovering Groups – Mean Shift Clustering

2D Example



Discovering Groups – Mean Shift Clustering

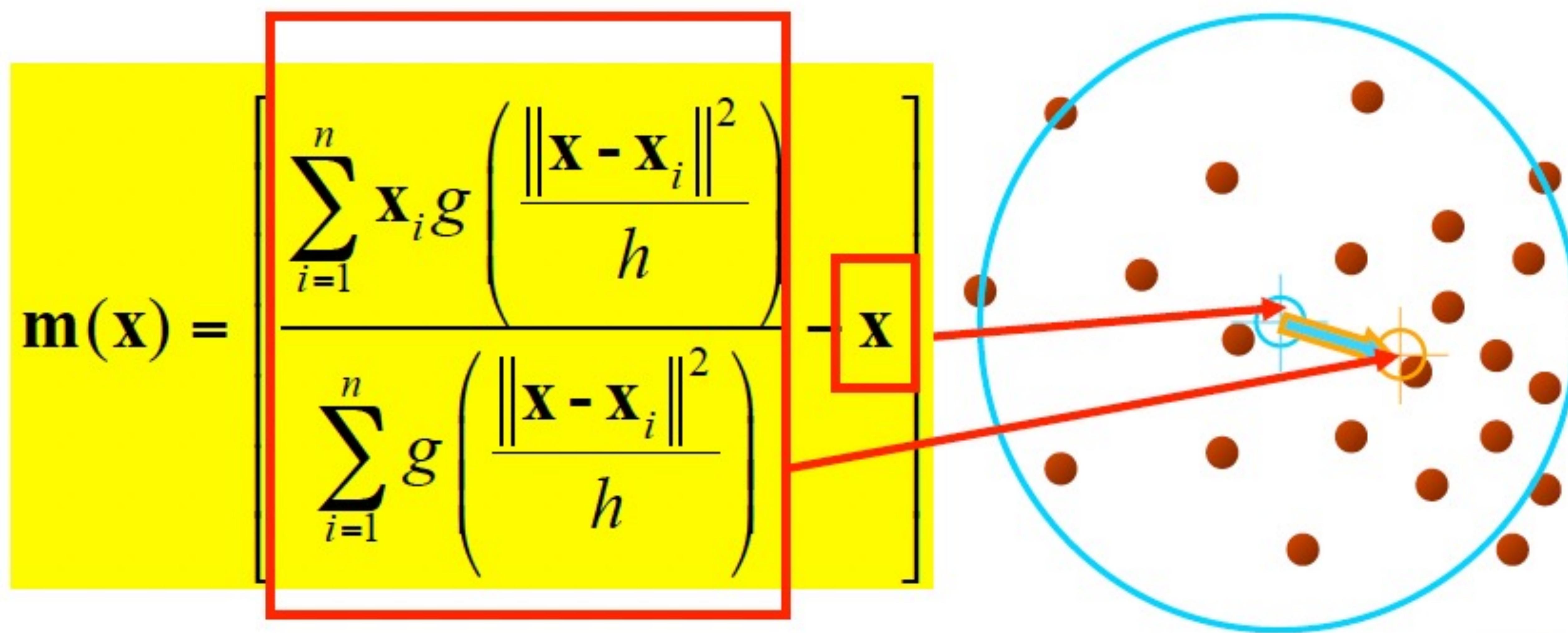
2D Example



Discovering Groups – Mean Shift Clustering

Simple Mean Shift procedure:

- Compute mean shift vector
- Translate the Kernel window by $\mathbf{m}(\mathbf{x})$



Source: Savarese slides.

Discovering Groups – Mean Shift Clustering

- Pros
 - Does not assume spherical clusters
 - Just a single parameter (window size)
 - Finds variable number of modes
 - Robust to outliers
- Cons
 - Output depends on window size
 - Computationally expensive
 - Does not scale well with dimension of feature space

Source: Savarese slides.

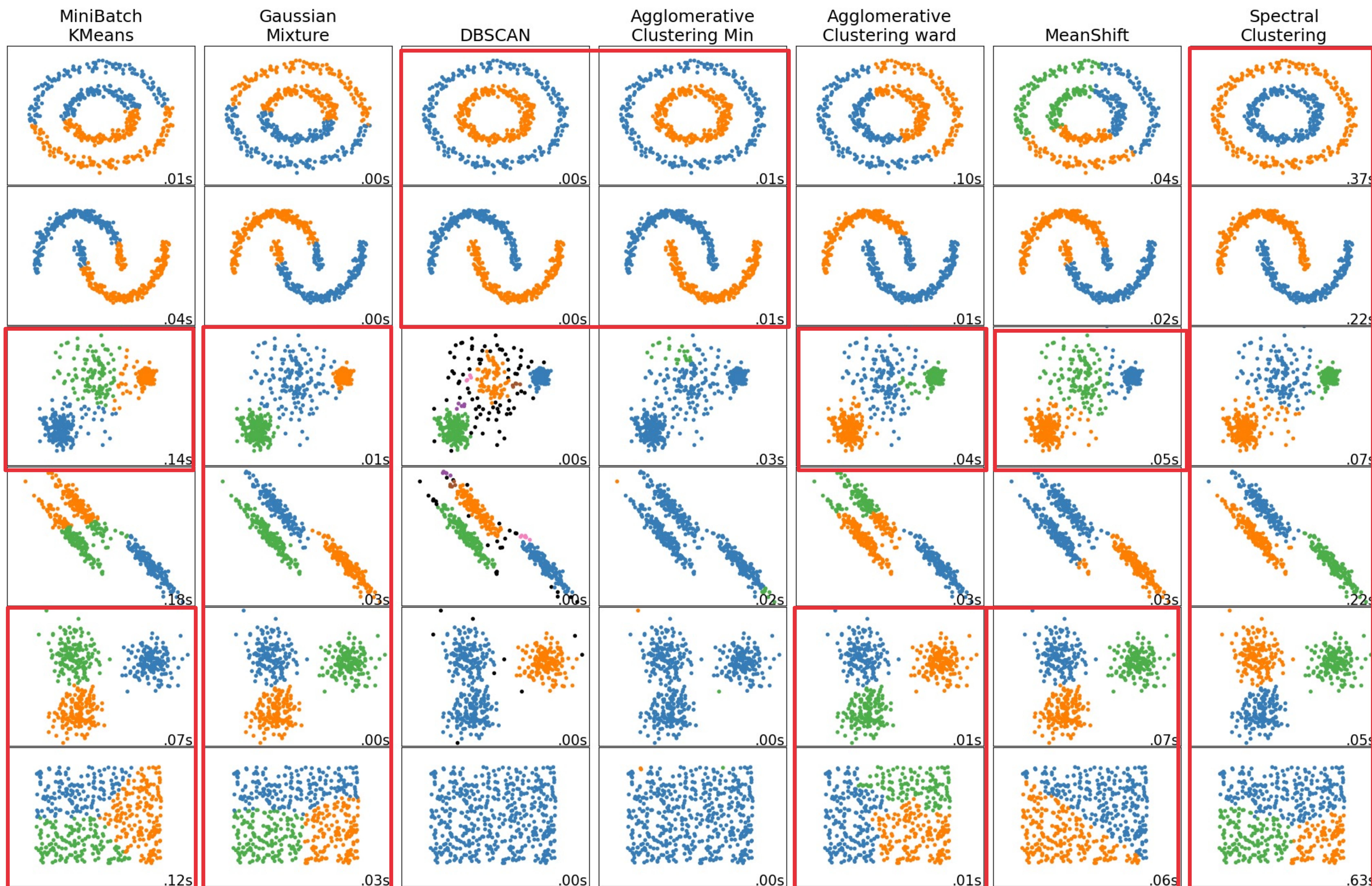
Discovering Groups – Summary

Clustering is a key way to understand your data.

There are many different approaches

- ▶ K Means - Need to chose K
- ▶ DBSCAN - need to choose min points and radius
- ▶ Hierarchical Agglomerative Clustering - needs a threshold or number of clusters
- ▶ Mean Shift Clustering - needs bandwidth

They are a very good way to start exploring a dataset



Source: https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py