Clustering

Unsupervised learning: describing data

1 Dimensionality Reduction

Developing new data representations

- Feature subset selection
- Feature projections
- Supervised approaches

2 Density Estimation

Quantifying data distributions

- Histograms
- Nonparametric density estimation
- Parametric models

3 Clustering

Grouping similar data

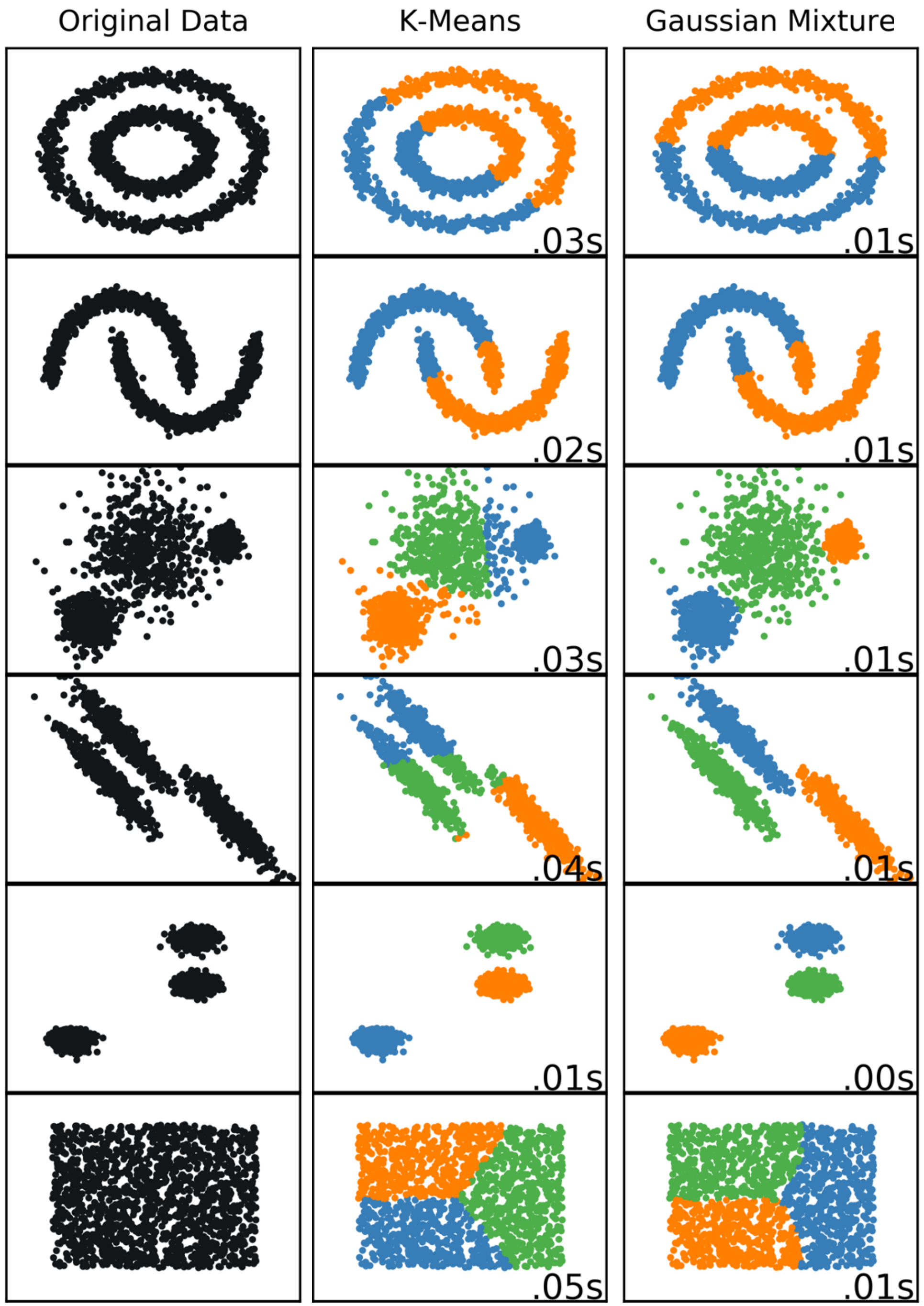
- Hierarchical
- Centroid-based
- Distribution-based
- Density-based

4 Other Unsupervised Learning Tools

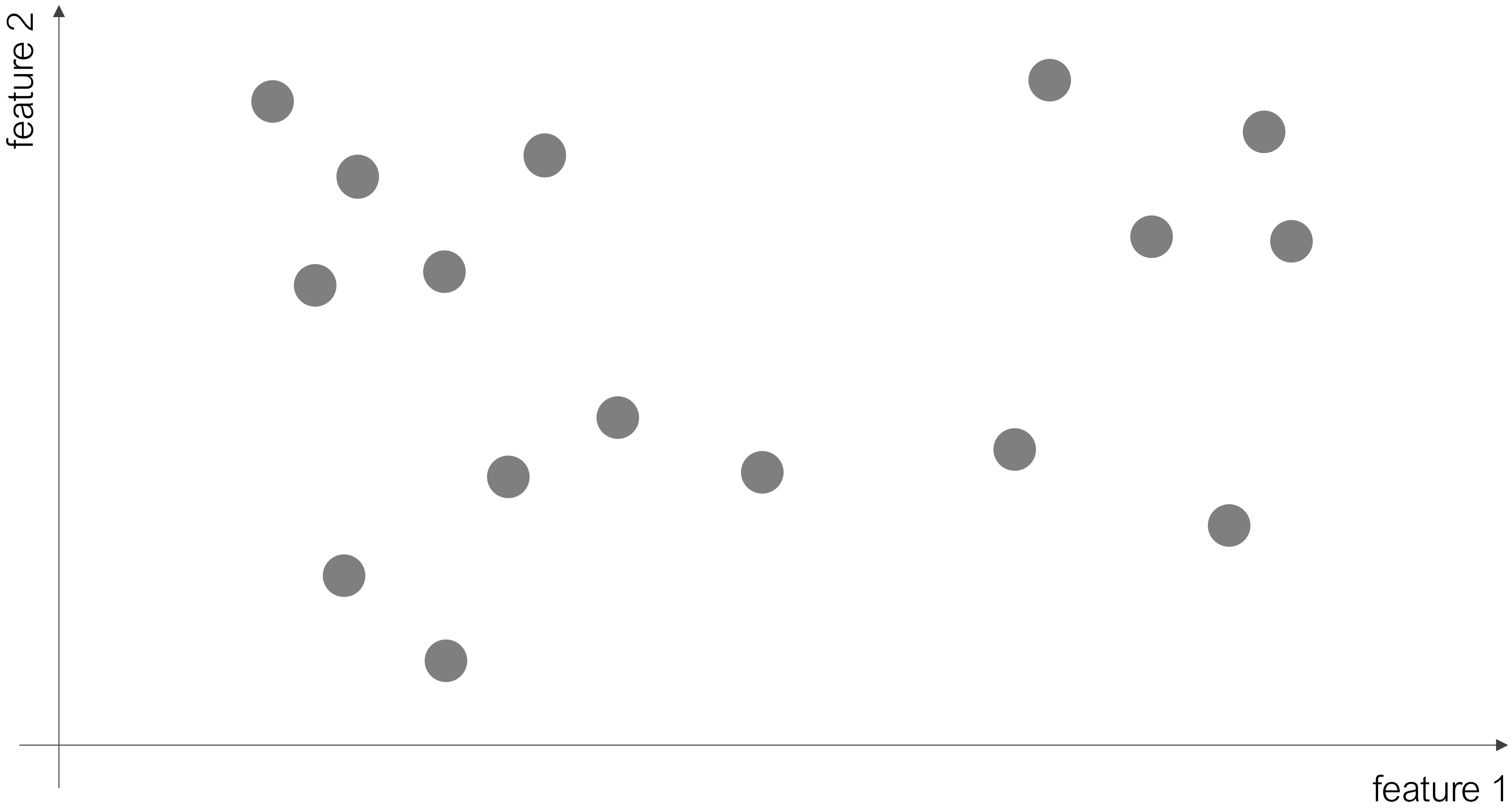
- Anomaly detection
- Representation learning
- Generative models

K-Means
+
Gaussian Mixture Models (GMMS)

Clustering and Density Estimation (GMMS)

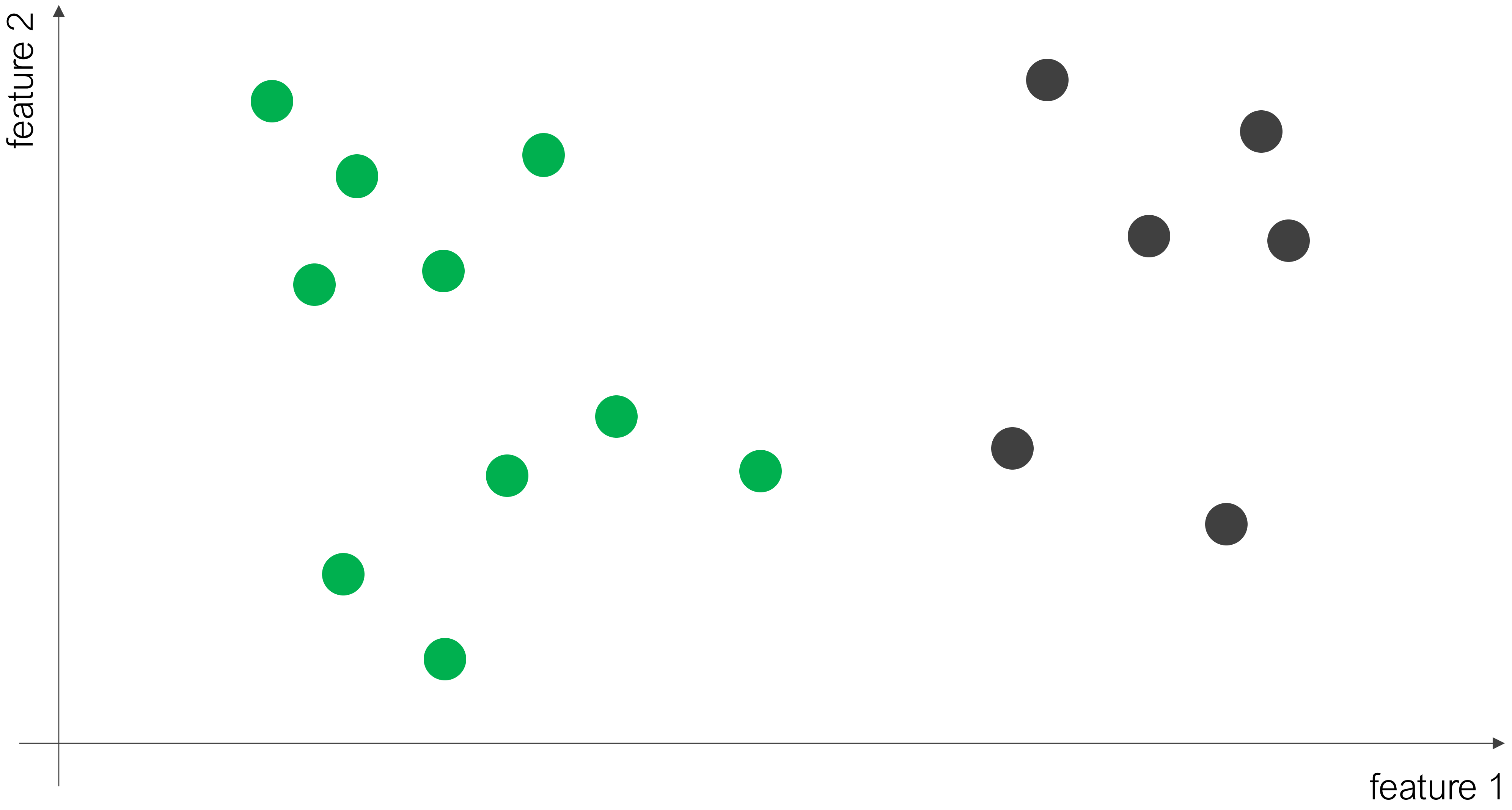


Clustering



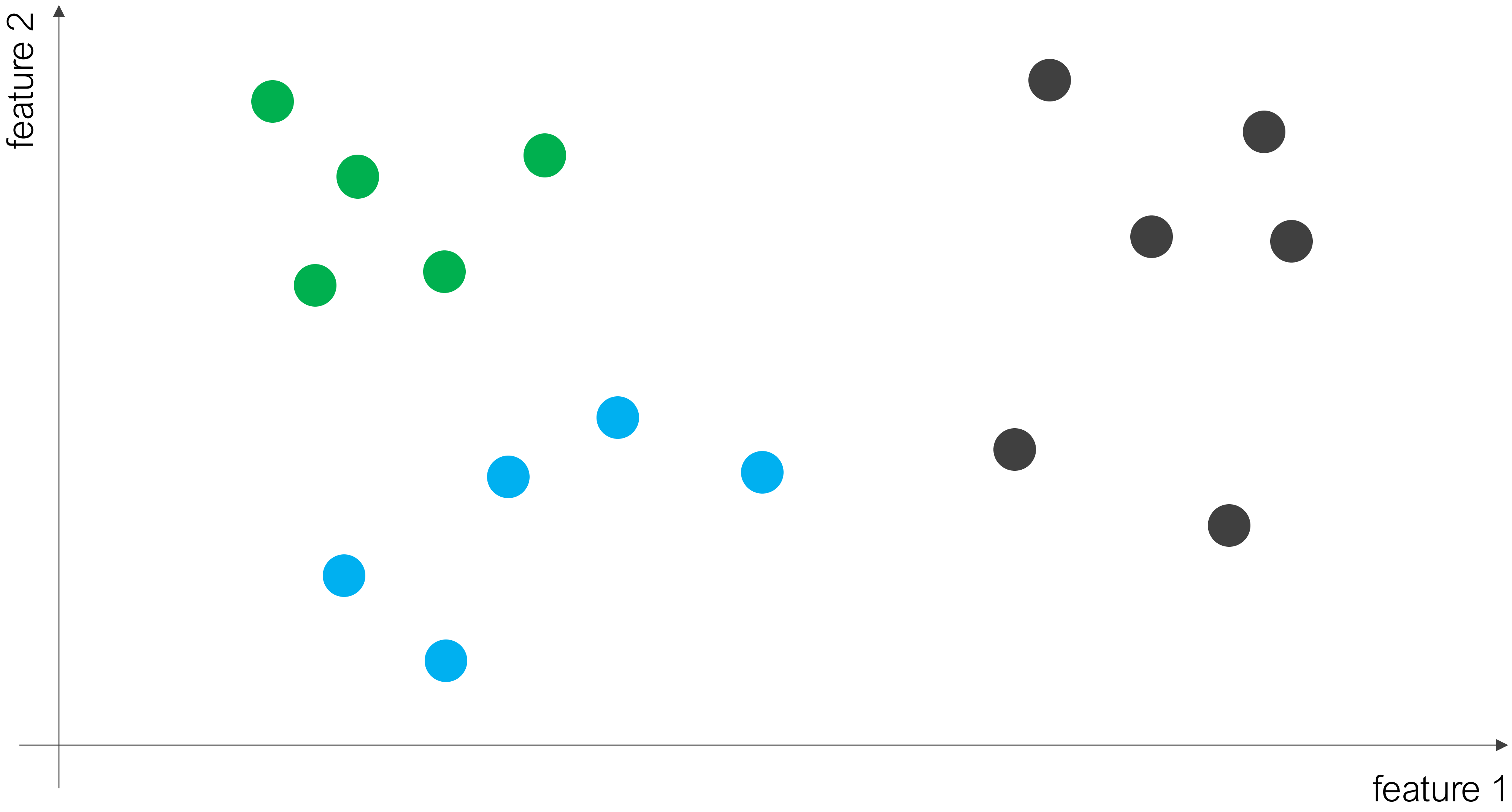
Clustering

Looks like 2 clusters…



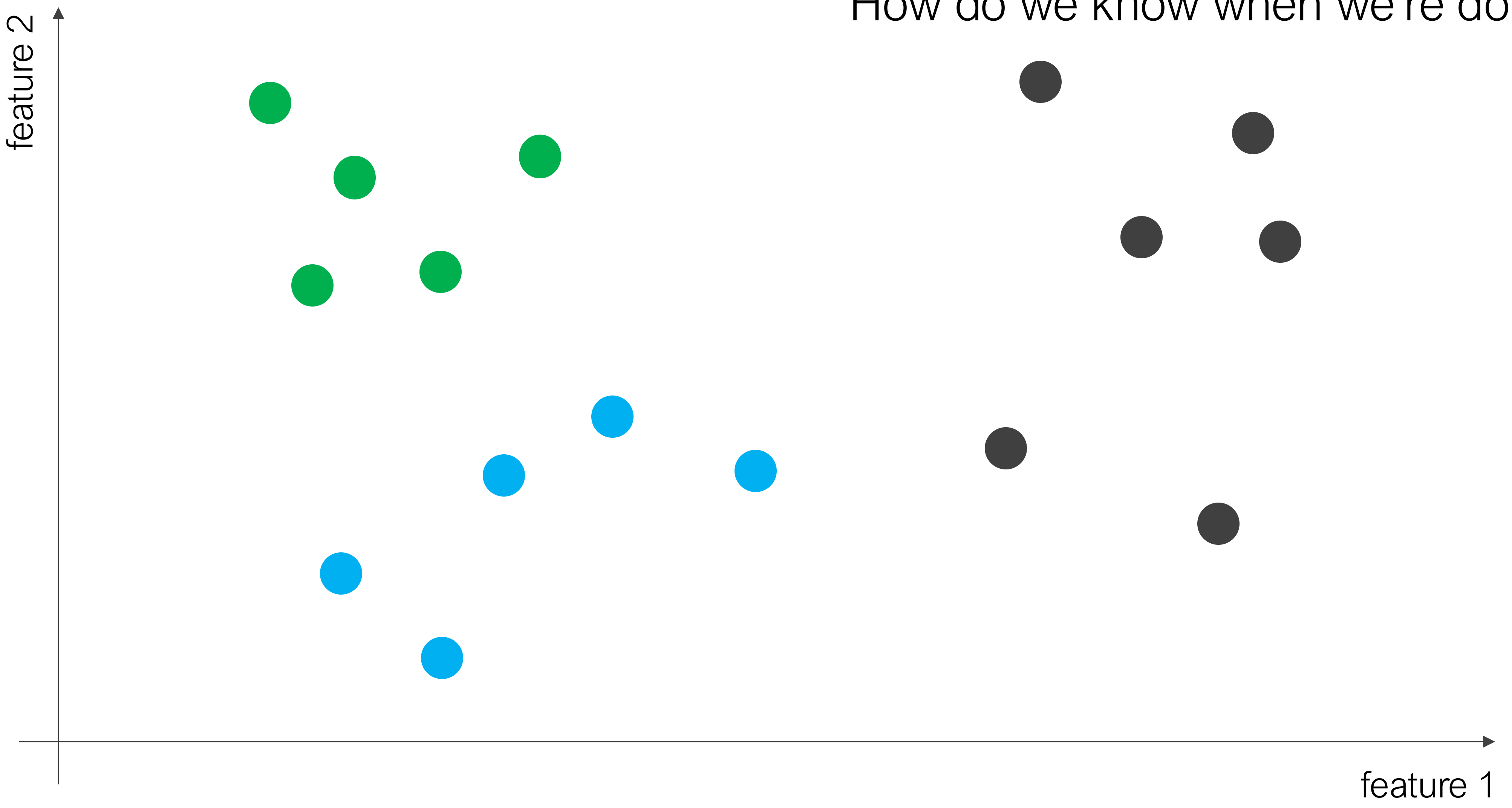
Clustering

… or maybe 3?



Clustering

How do we define “similarity”?
How do we choose the number of clusters?
How do we know when we’re doing well?



Example Applications

Customer segmentation for market research

Social network analysis and identifying communities

Crime tracking to identify hot spots for certain types of crimes

Types of clustering algorithms

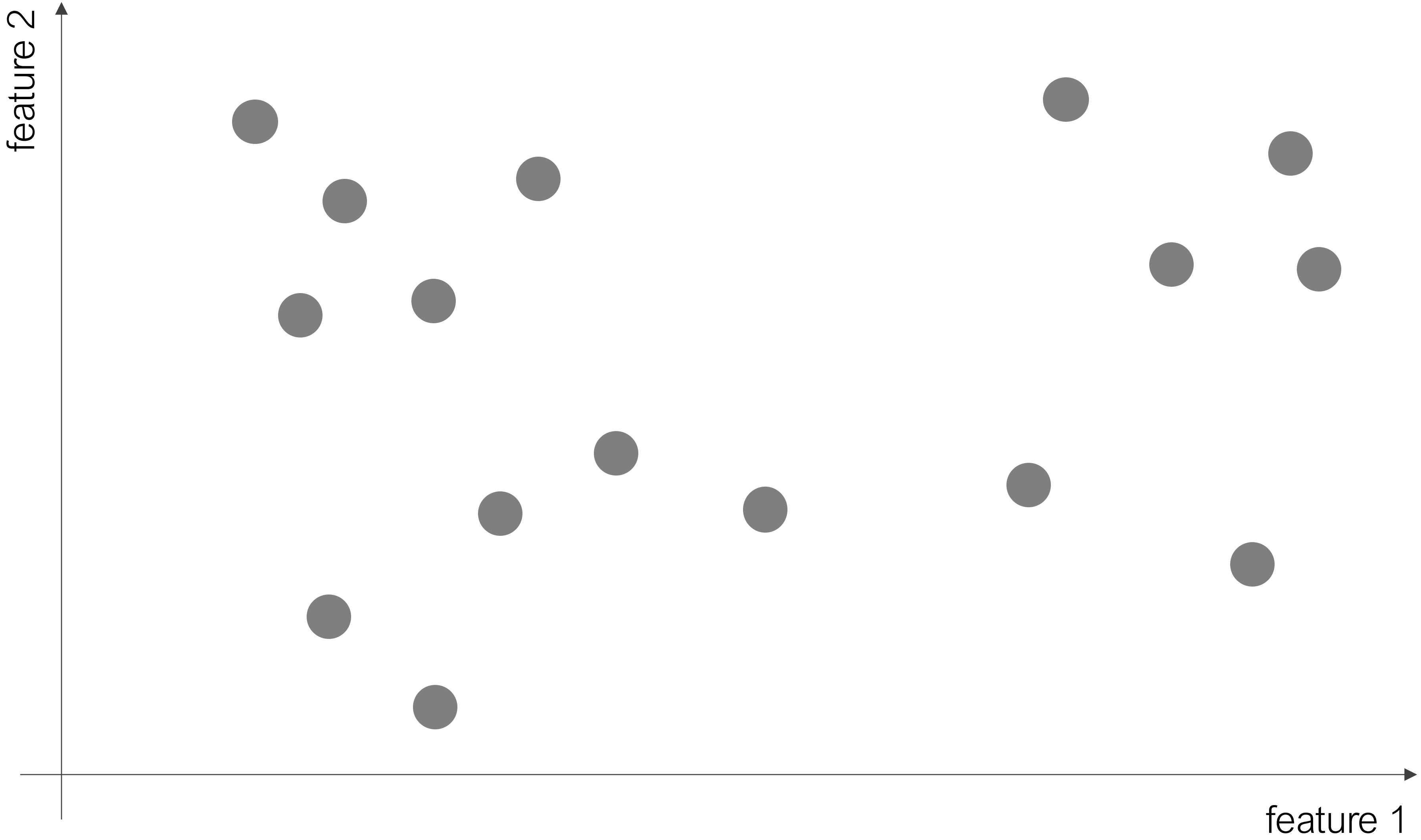
Methods

Distribution-based clustering (e.g. **Gaussian mixture model**)
Centroid-based clustering (e.g. K-Means)
Density-based clustering (e.g. DBSCAN)
Hierarchical clustering (e.g. agglomerative clustering)
a.k.a. connectivity-based clustering

Cluster assignment

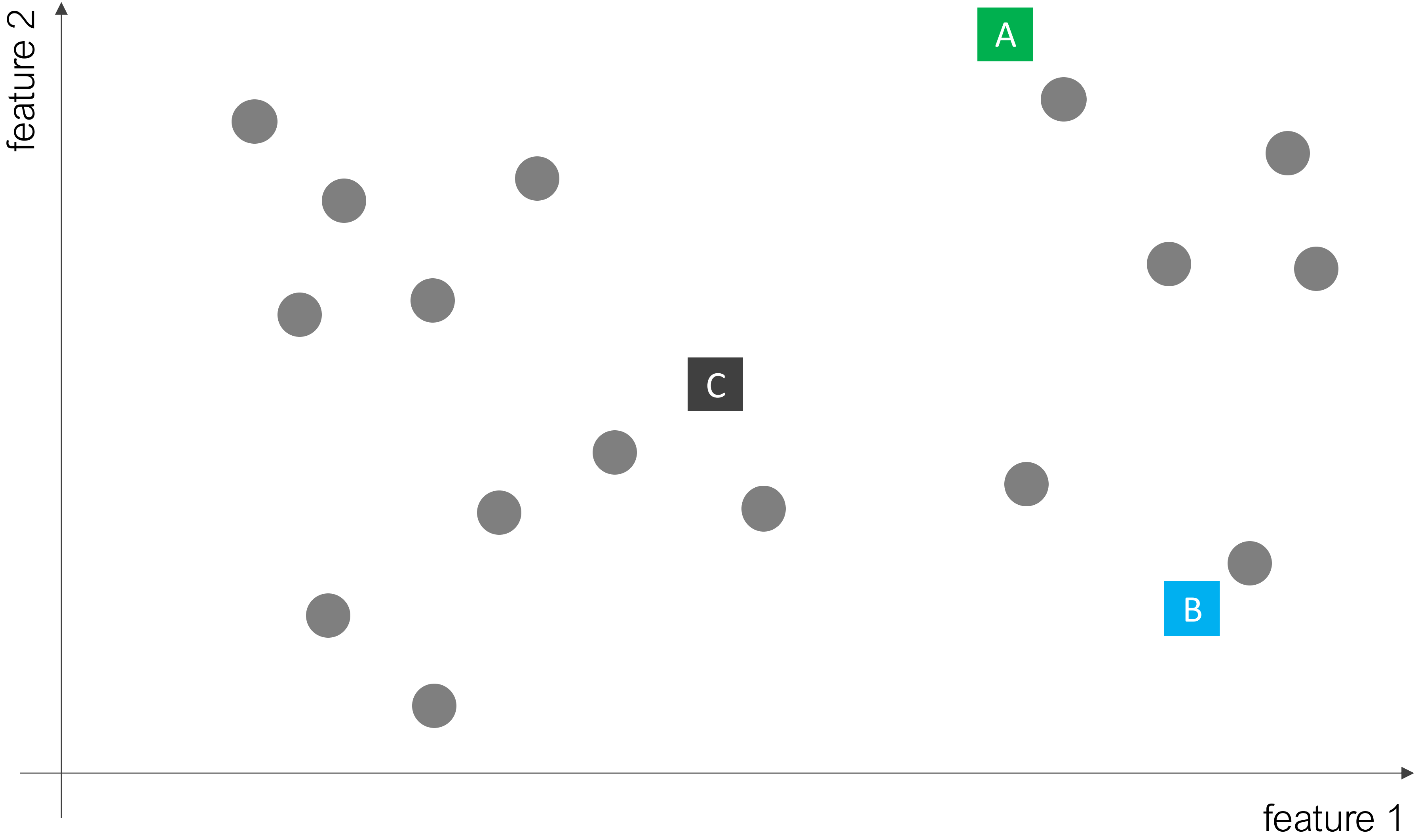
Hard clustering
Soft clustering (a.k.a. fuzzy clustering)

K-means clustering



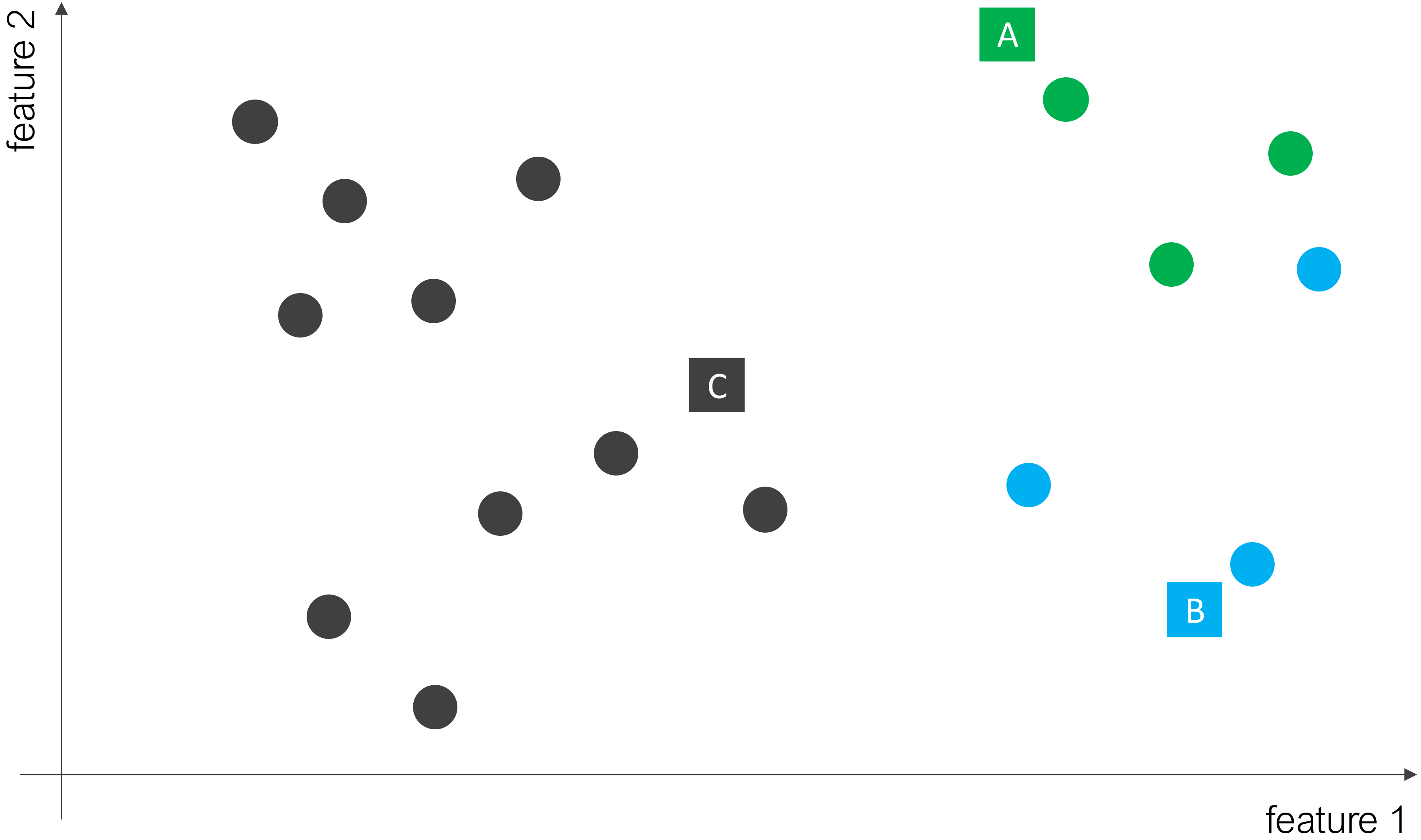
K-means clustering

1 Select k and randomly initialize k mean values

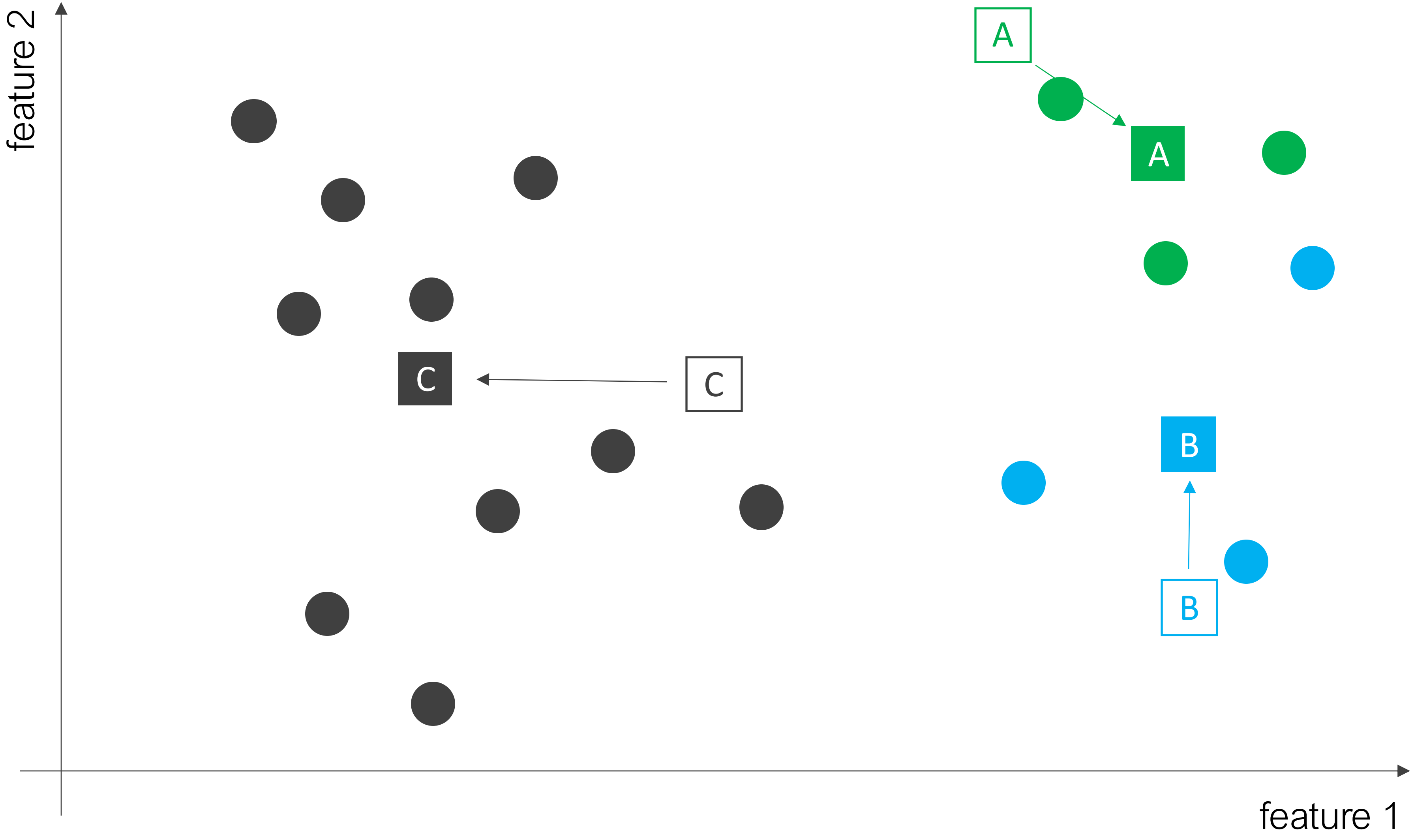


K-means clustering

1. Select k and randomly initialize k mean values
2. Assign observations to the nearest mean

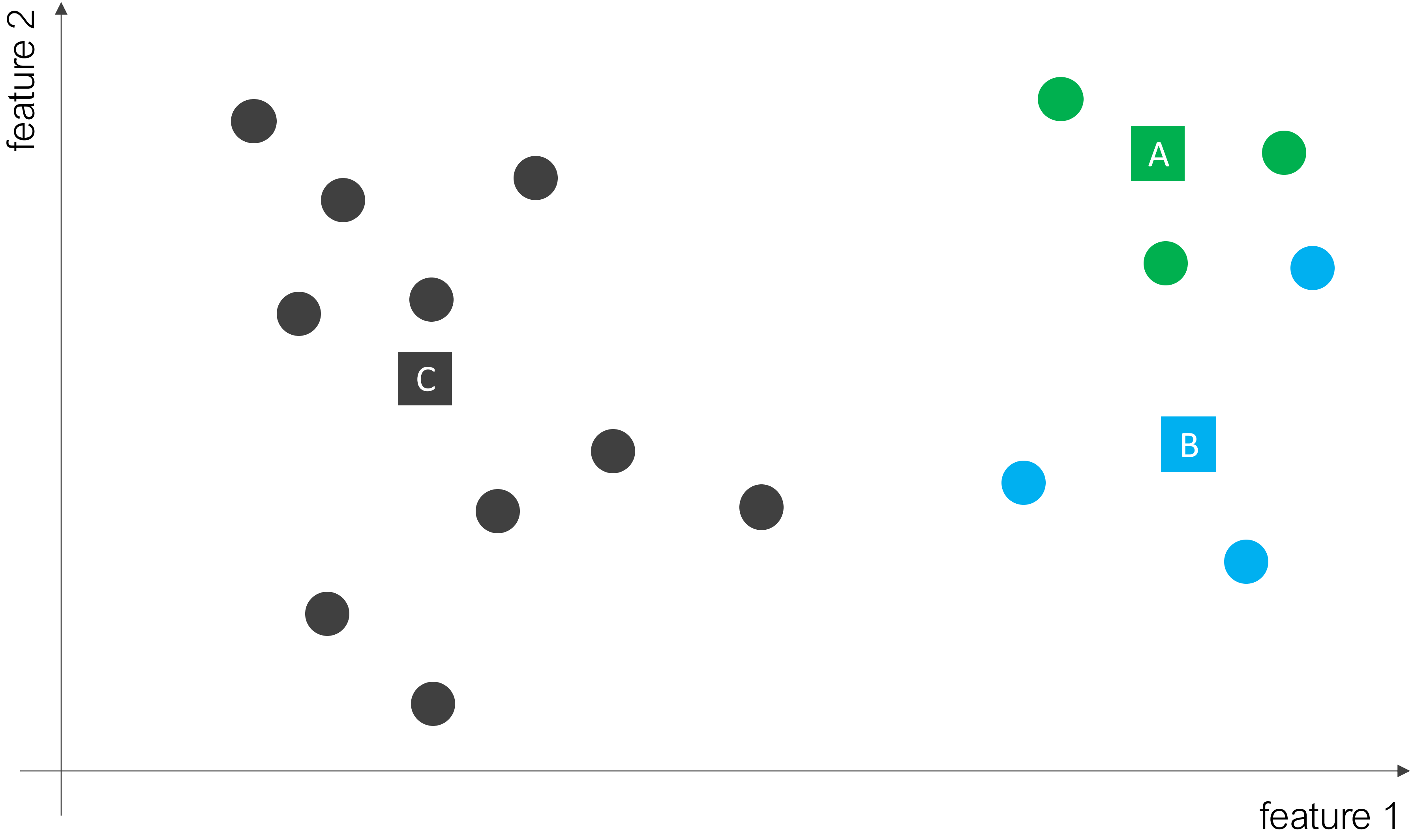


K-means clustering



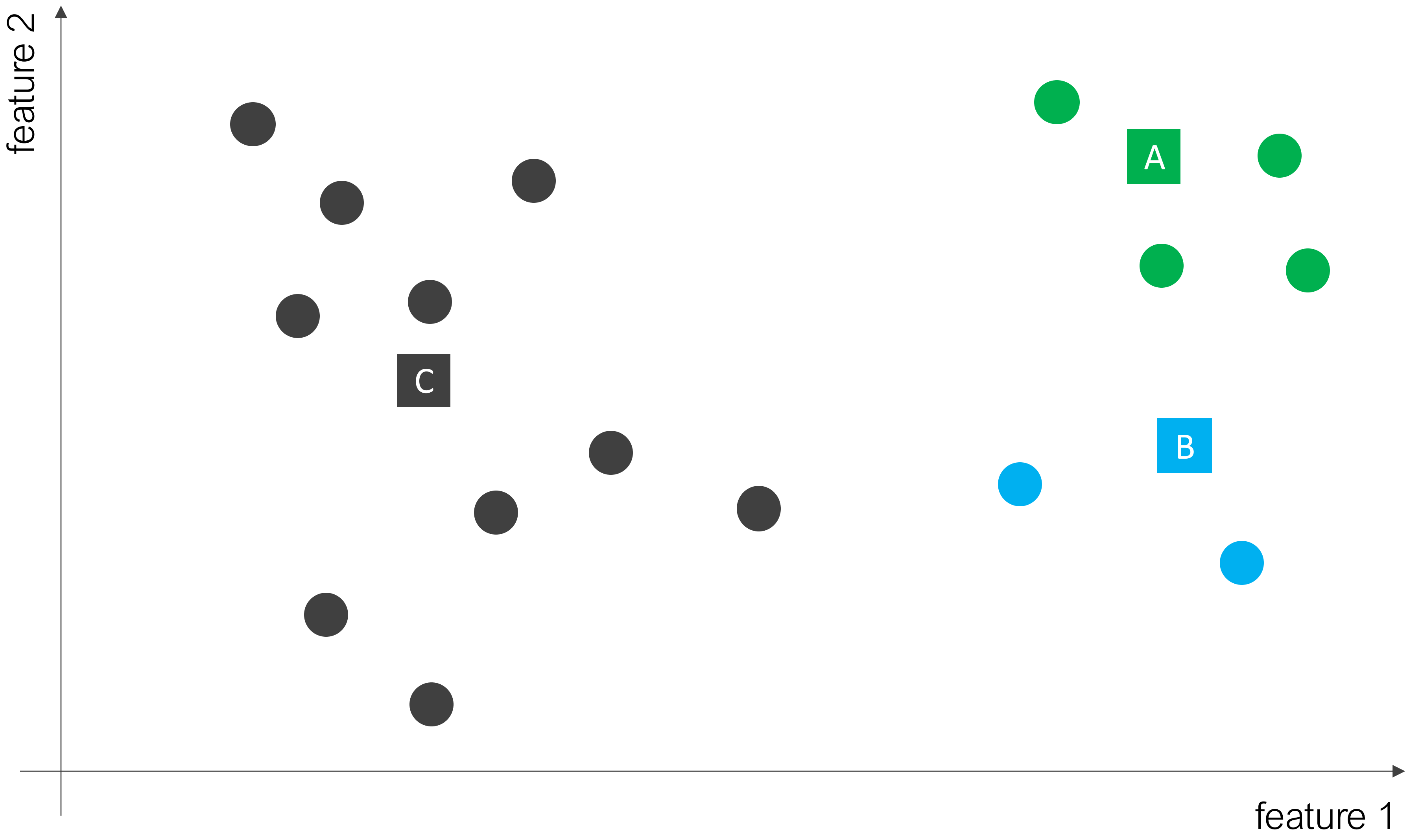
1. Select k and randomly initialize k mean values
2. Assign observations to the nearest mean
3. Update the mean to be the centroid of the labeled data

K-means clustering



1. Select k and randomly initialize k mean values
2. Assign observations to the nearest mean
3. Update the mean to be the centroid of the labeled data
4. Repeat steps 2 and 3 until convergence

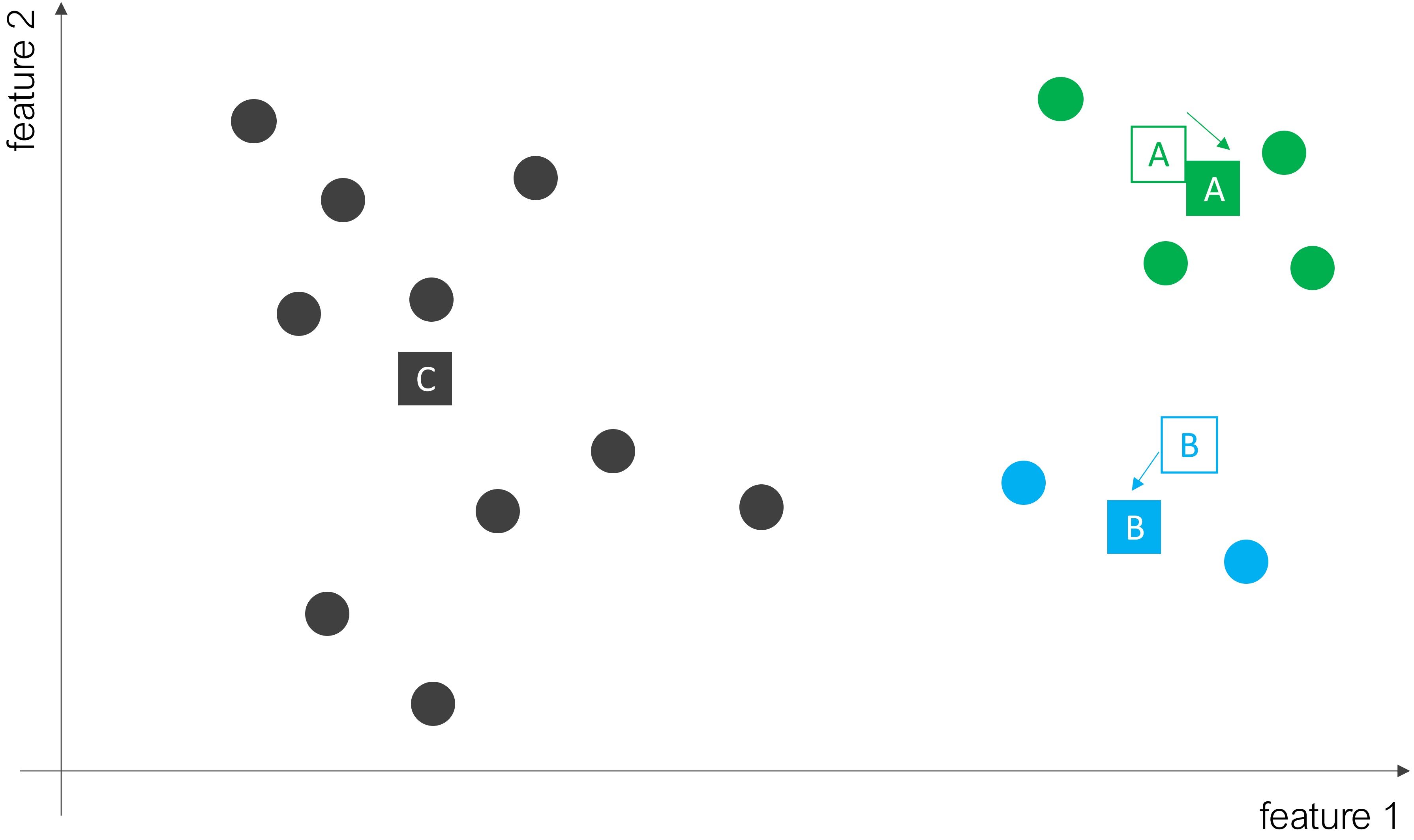
K-means clustering



1. Select k and randomly initial k mean values
2. Assign observations to the nearest mean
3. Update the mean to be the centroid of the labeled data
4. Repeat steps 2 and 3 until convergence

…Iteration 2

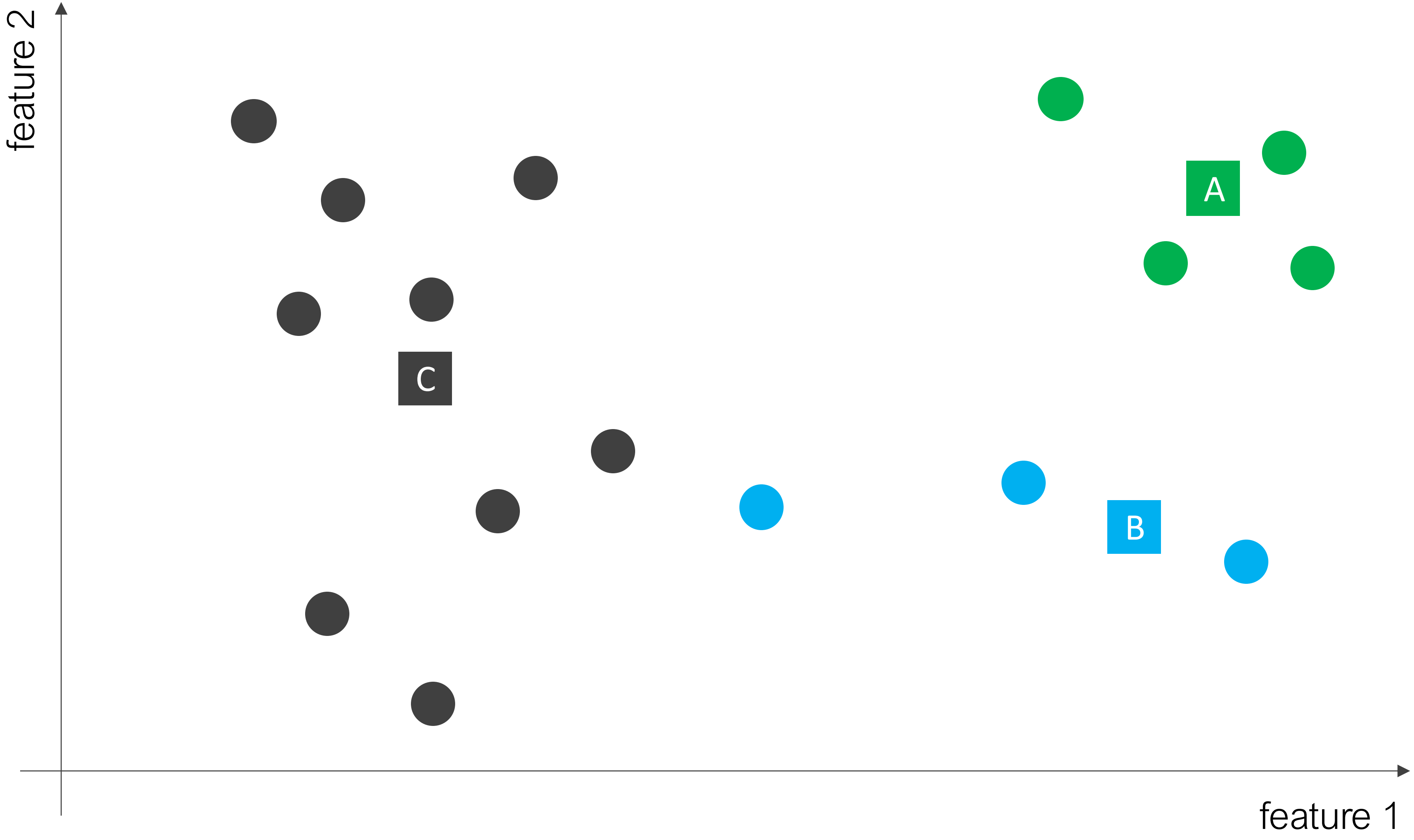
K-means clustering



1. Select k and randomly initialize k mean values
2. Assign observations to the nearest mean
3. Update the mean to be the centroid of the labeled data
4. Repeat steps 2 and 3 until convergence

…Iteration 2

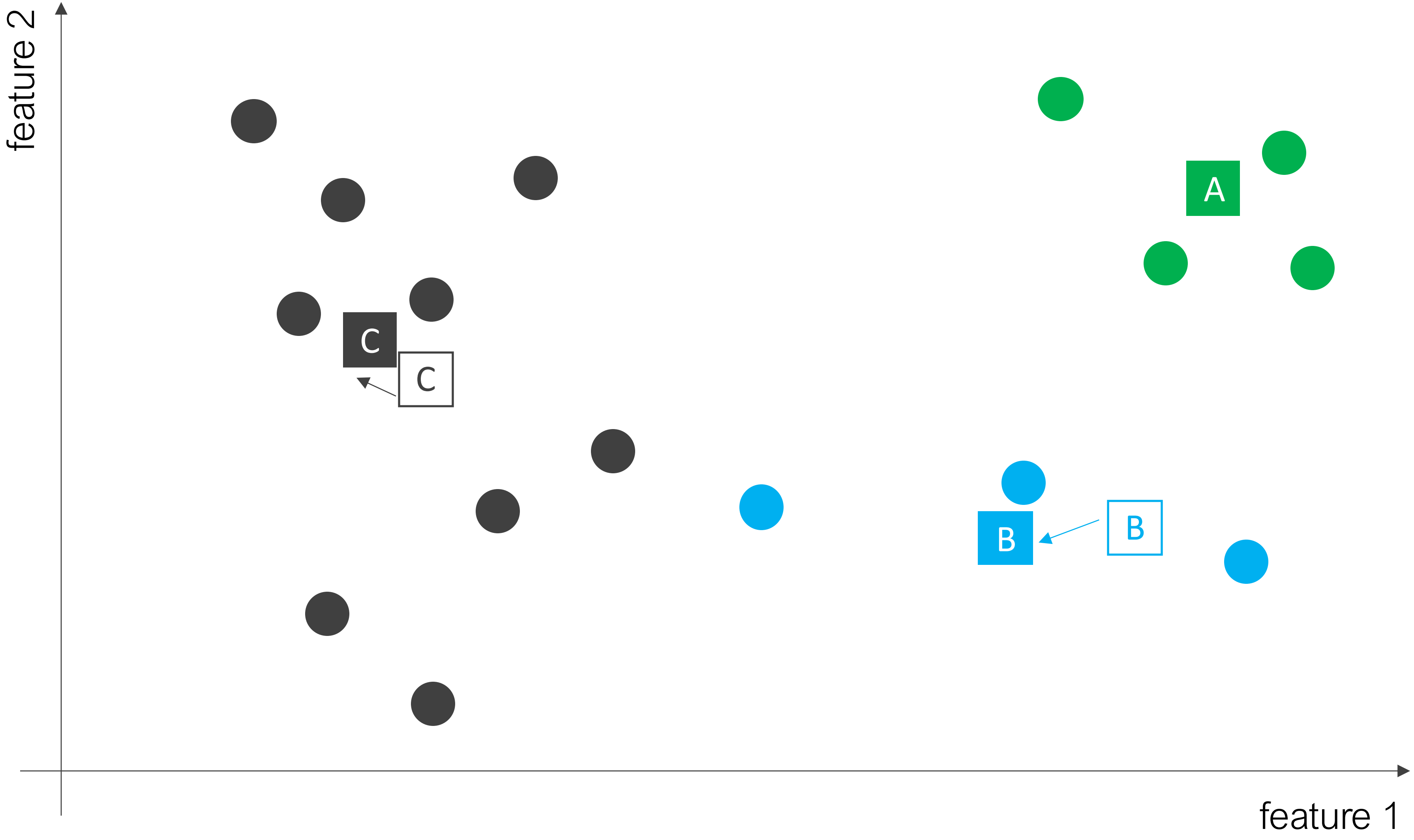
K-means clustering



1. Select k and randomly initialize k mean values
2. Assign observations to the nearest mean
3. Update the mean to be the centroid of the labeled data
4. Repeat steps 2 and 3 until convergence

…Iteration 3

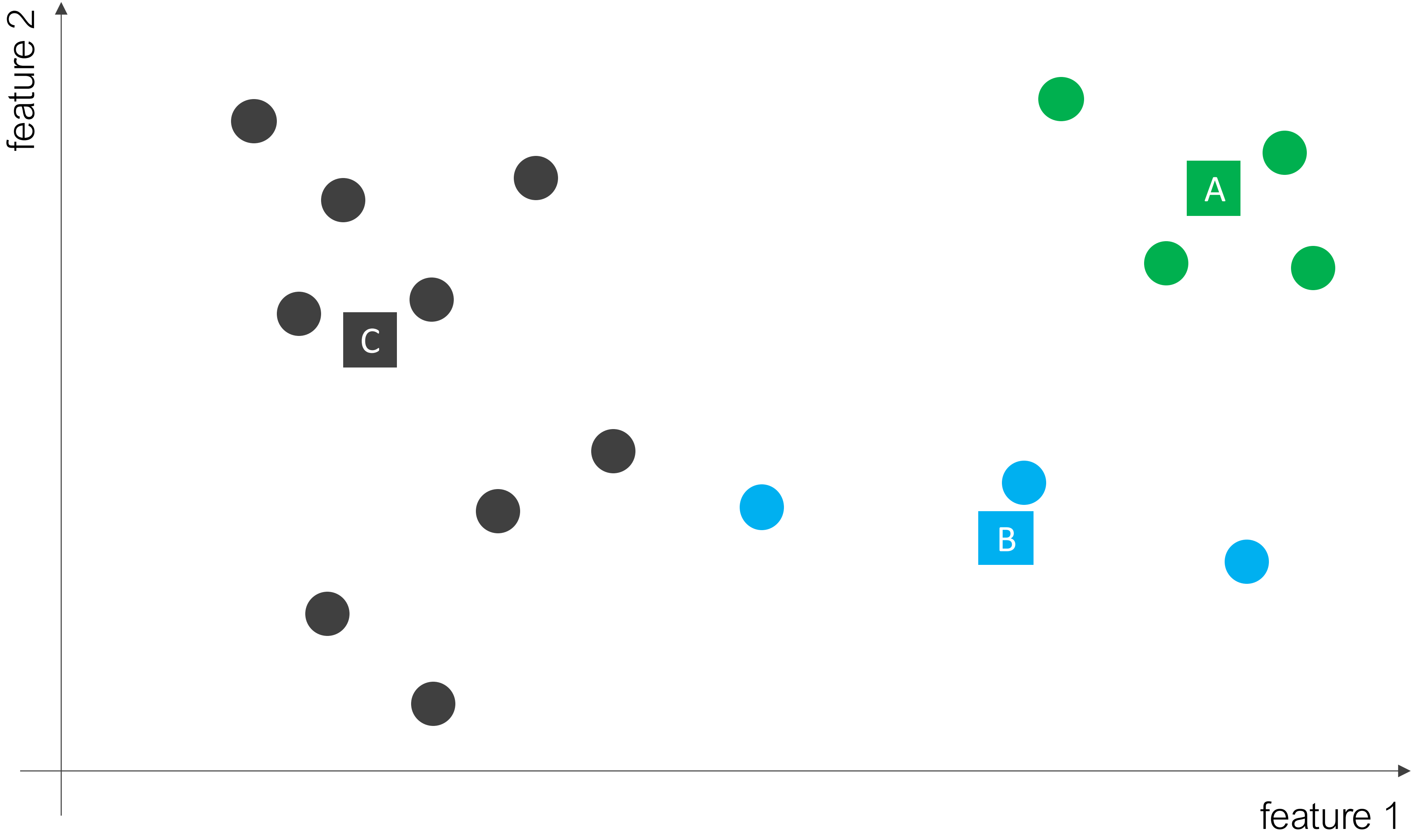
K-means clustering



1. Select k and randomly initialize k mean values
2. Assign observations to the nearest mean
3. Update the mean to be the centroid of the labeled data
4. Repeat steps 2 and 3 until convergence

…Iteration 3

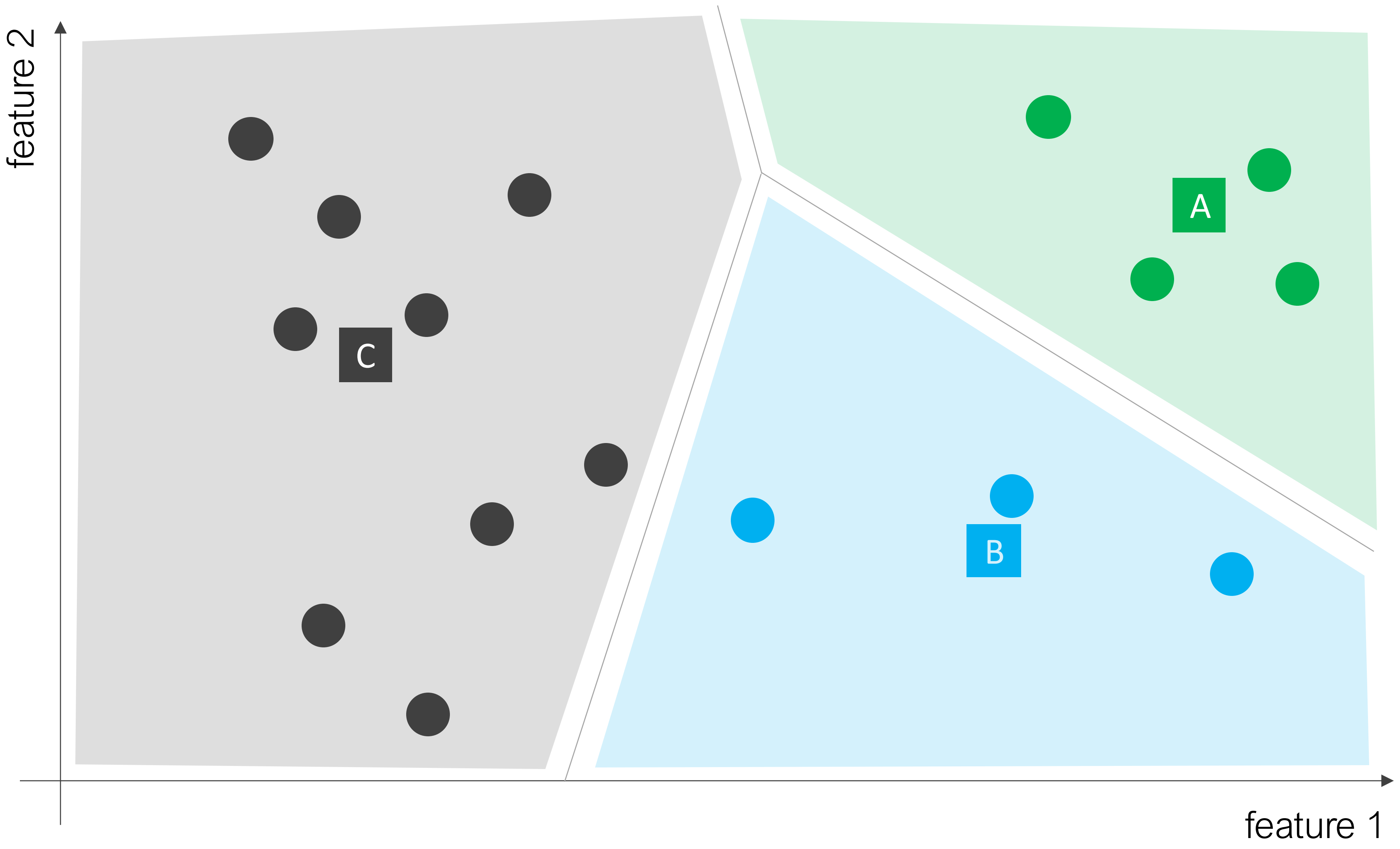
K-means clustering



1. Select k and randomly initialize k mean values
2. Assign observations to the nearest mean
3. Update the mean to be the centroid of the labeled data
4. Repeat steps 2 and 3 until convergence

…converged

K-means partitions the space into Voronoi cells



Under the hood, we minimize a cost function

Objective: For our N samples, identify K means, $\boldsymbol{\mu}_k$, such that the set of closest points in feature space are the minimum distance away.

$$r_{ik} = \begin{cases} 1 \text{ if } \boldsymbol{x}_i \text{ is closest to the kth mean } \boldsymbol{\mu}_k \\ 0 \text{ else} \end{cases}$$

responsibility

L2 norm

$$C(\boldsymbol{x}_i, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K) = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_2^2$$

1. E-step

Re-evaluate r_{ik}

$$r_{ik} = \begin{cases} 1 \text{ if } \boldsymbol{x}_i \text{ is closest to the kth mean } \boldsymbol{\mu}_i \\ 0 \text{ else} \end{cases}$$

Assign new "expected" cluster assignments

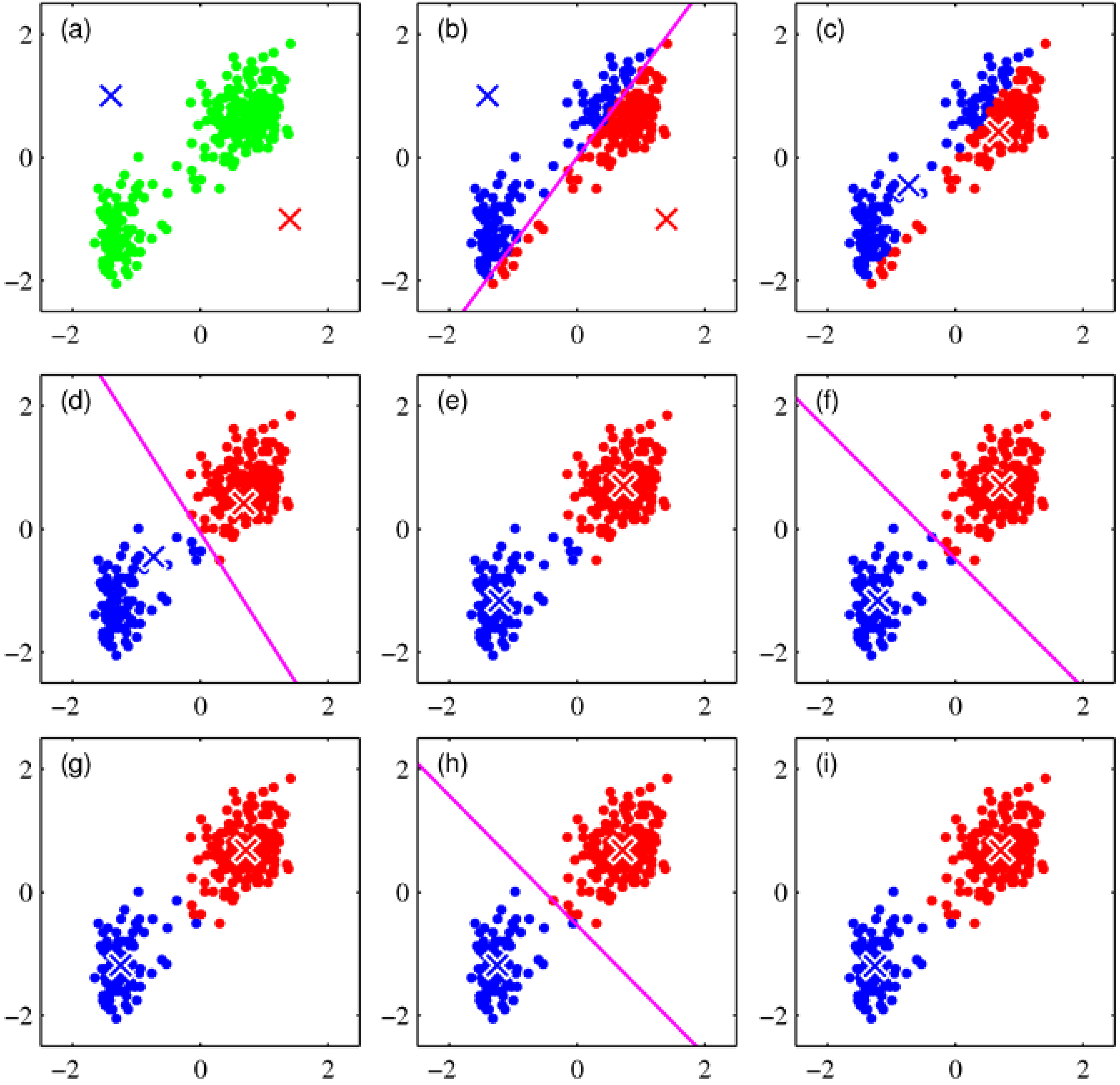
2. M-step

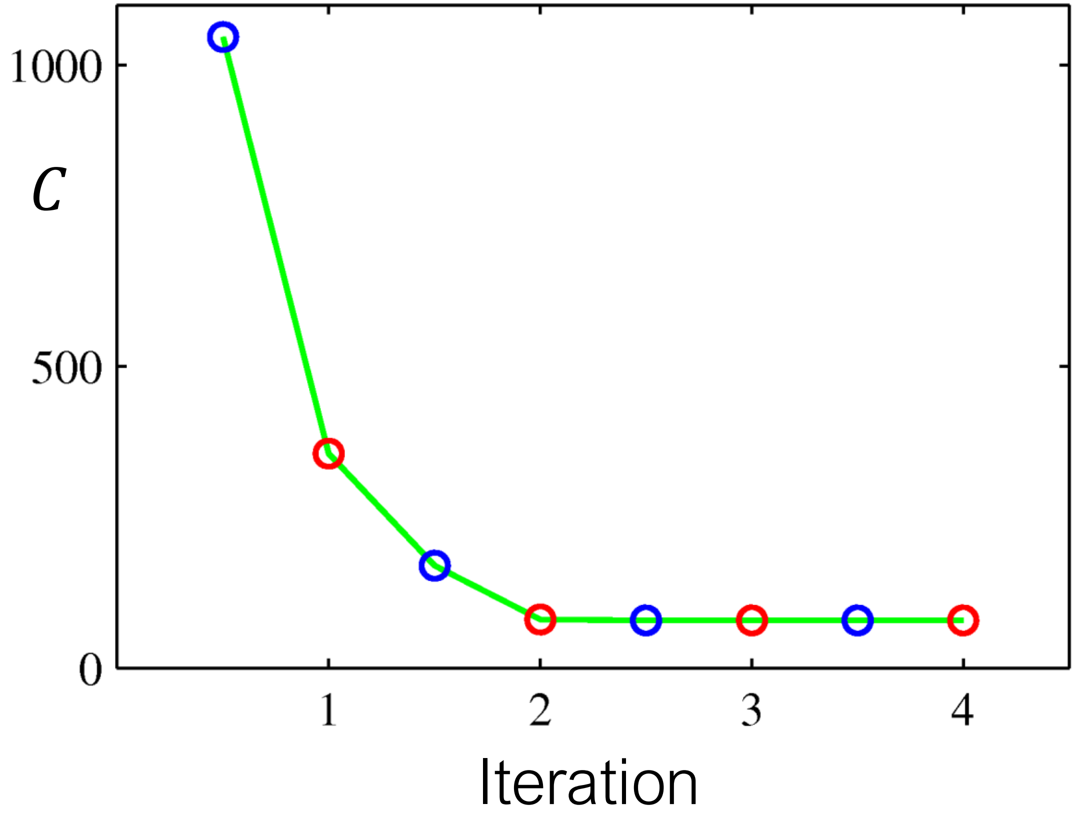
Minimize C wrt $\boldsymbol{\mu}_i$

$$\boldsymbol{\mu}_k = \frac{\sum_i r_{ik}\, \boldsymbol{x}_i}{\sum_i r_{ik}}$$

Update the cluster means to maximize the likelihood

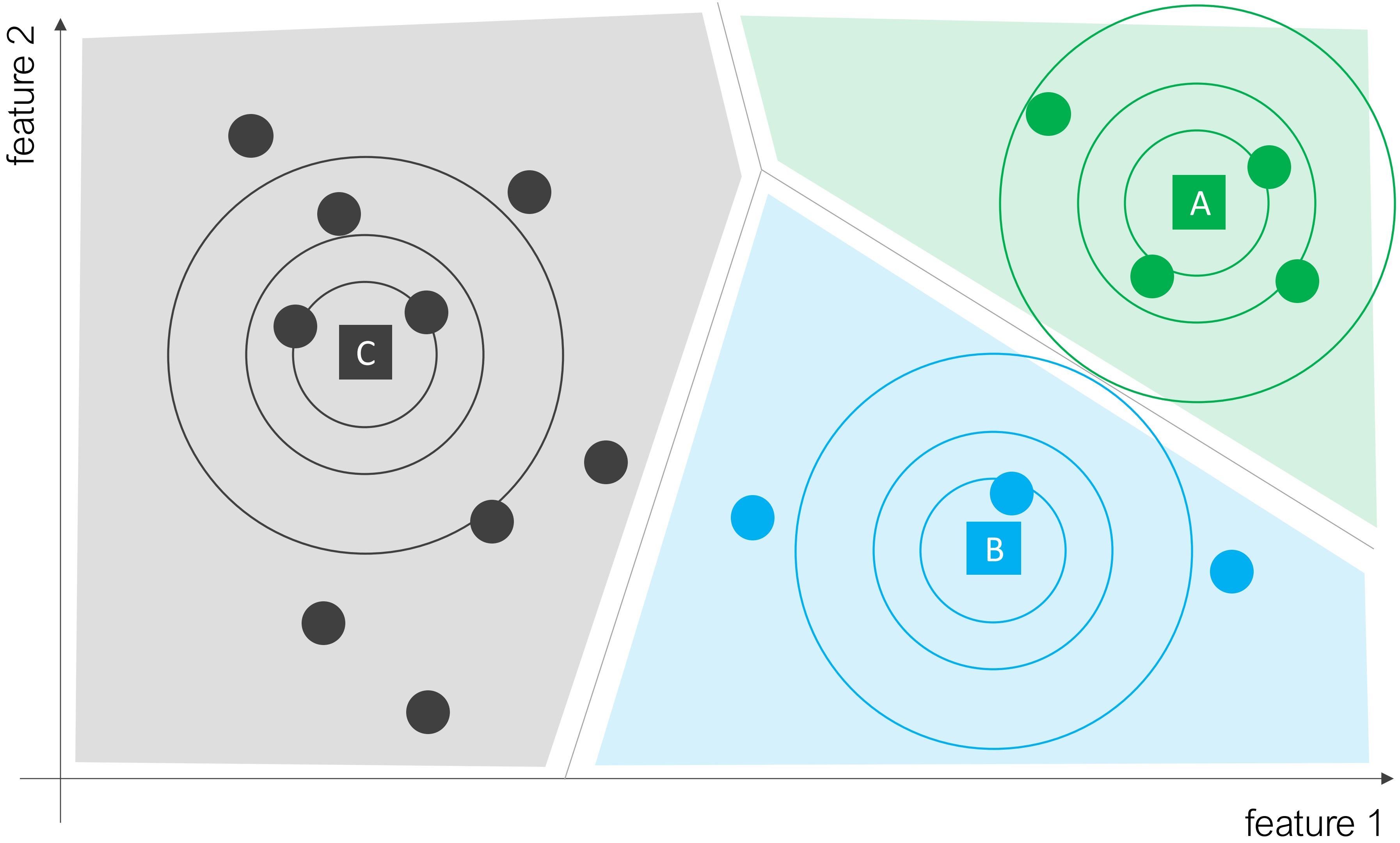
Convergence





Bishop, Pattern Recognition, 2006

Relationship to Gaussian distributions



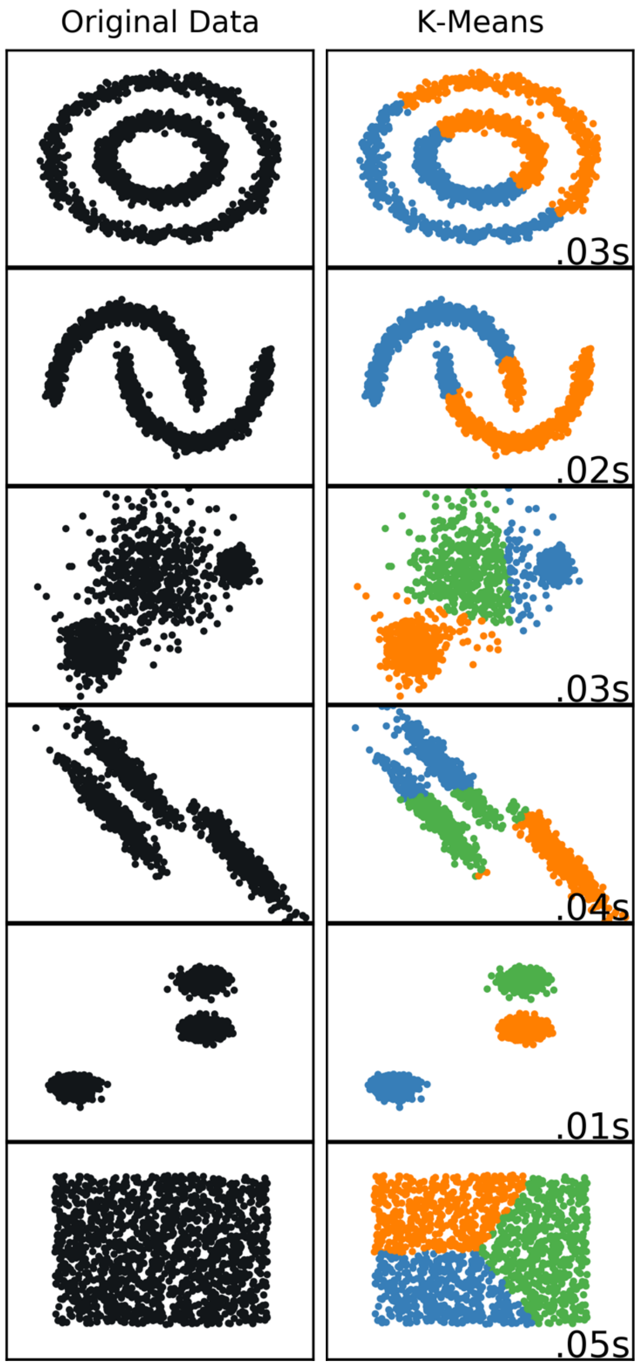
Assumes the clusters are **Gaussians** centered at the mean, each with **identical covariance matrices**, where all the features are independent:

$$\mathbf{\Sigma}_{\mathrm{k}} = \sigma^2 \boldsymbol{I} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Examples: K-Means

Converges very quickly

Sensitive to initialization of means



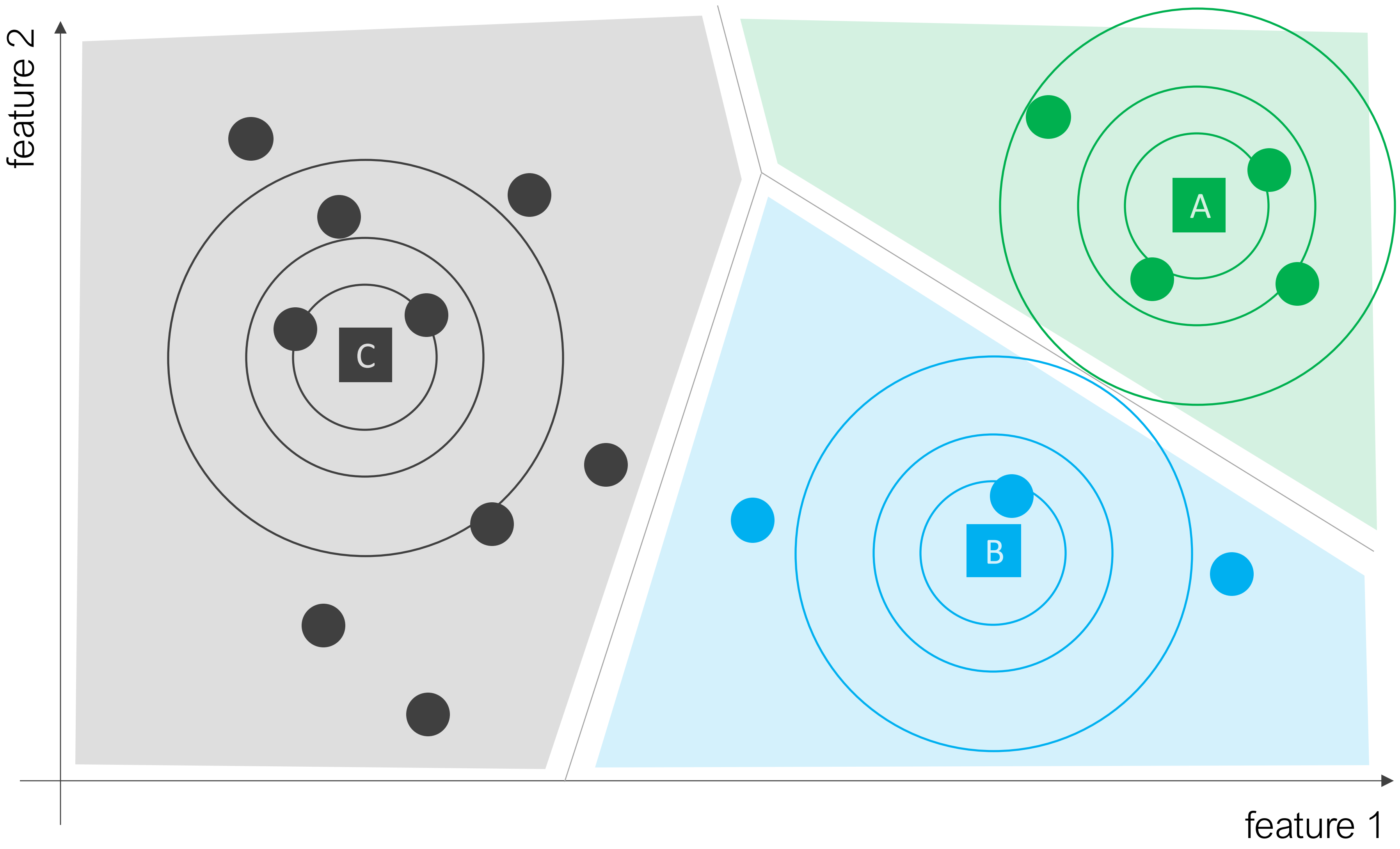
Struggles when there are **nonlinear** boundaries between clusters

Struggles in situations with **variation in cluster variance** and **correlation between features**

Excels with clusters of **equal variance**

Will divide into k clusters even when there are not k

Relaxing our assumptions on covariance…

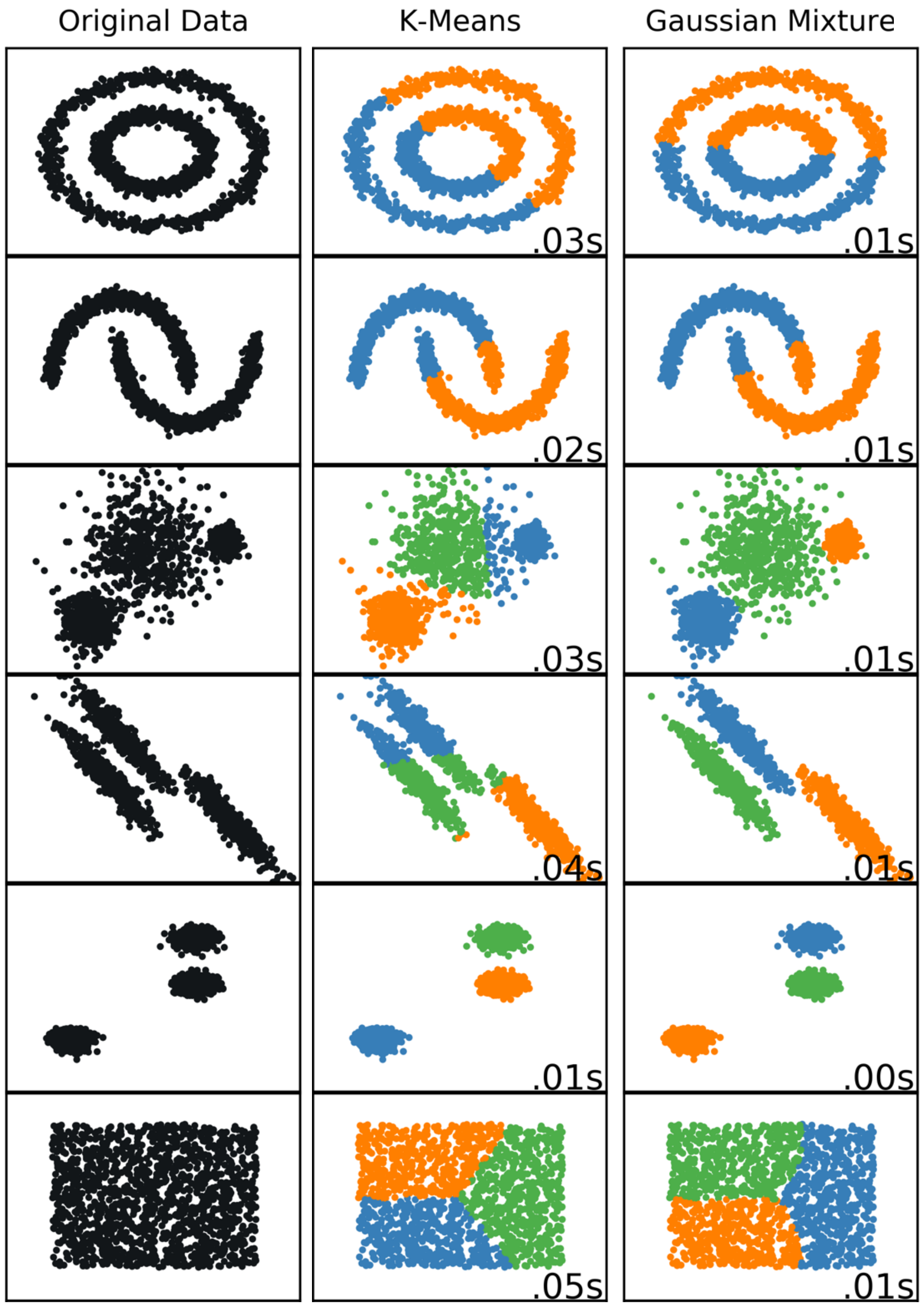


What if we **don't** assume the Gaussian clusters have **identical, diagonal covariance matrices**?

Examples: GMM

Can produce soft clustering

Estimates the density / distribution of the data



Struggles when the clusters are not approximately Gaussian

Excels in situations with **variation in cluster variance** and **correlation between features**

Excels with clusters of **equal variance**

Will divide into k clusters even when there are not k

How to choose k: Elbow method

Run k-means for various k

Choose the value of k at the “elbow” of the curve

Increasing k will improve the fit, but at the cost of potentially overfitting the data

Other approaches: silhouette (graphical approach to evaluating cluster fit), supervised techniques

Cluster evaluation considerations:
- Within-cluster cohesion (compactness)
- Between-cluster separation (isolation)

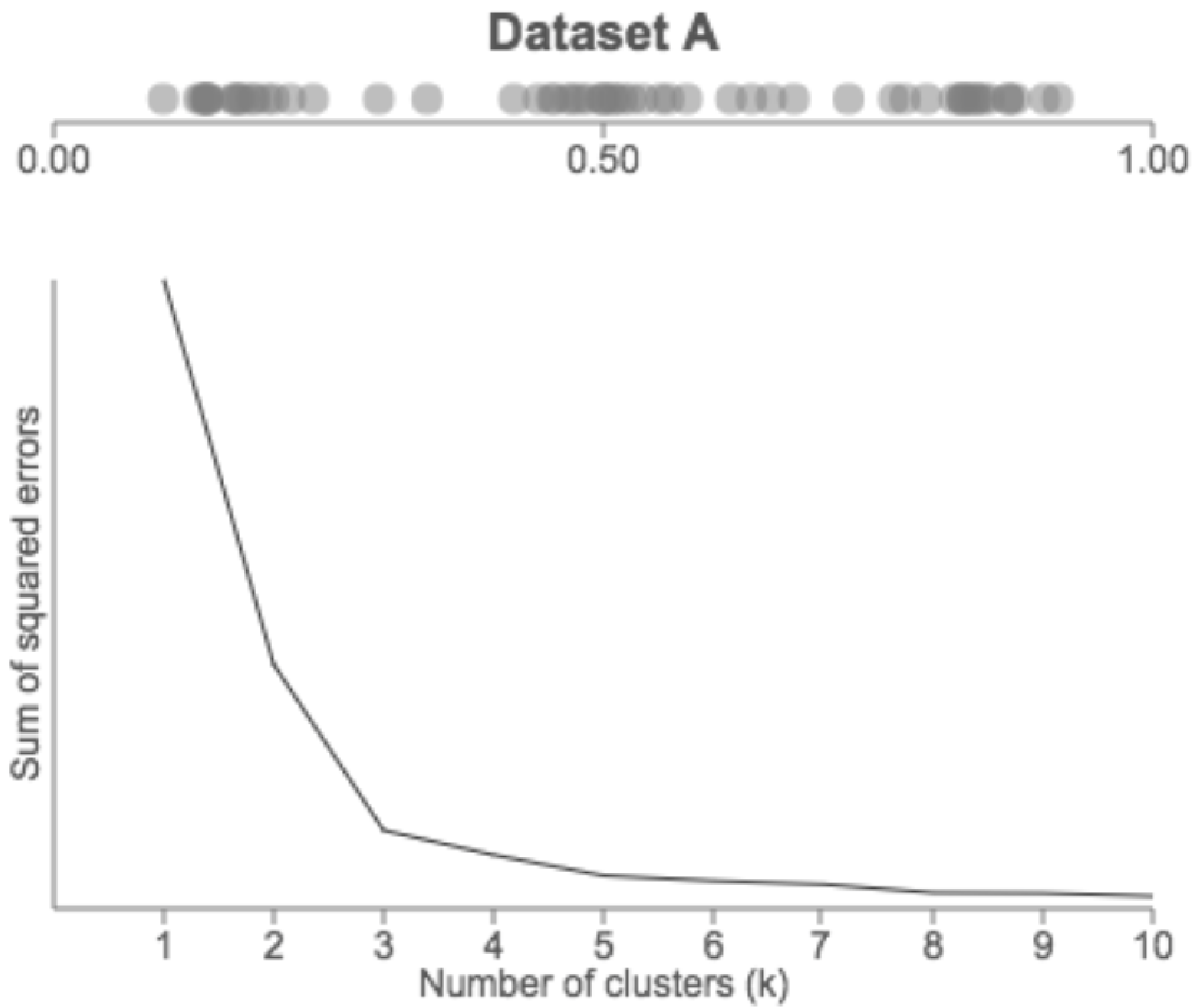


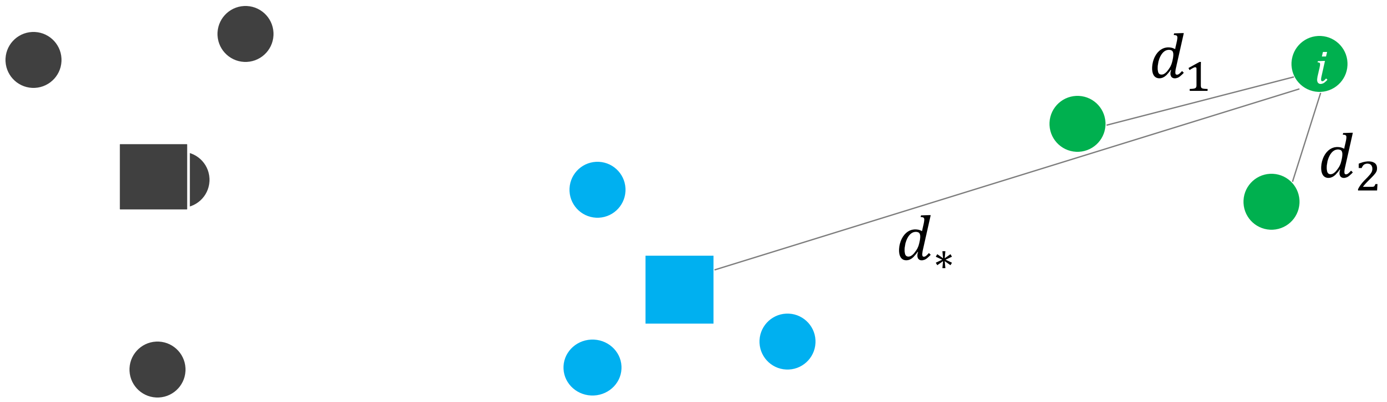
Image by Robert Gove: https://bl.ocks.org/rpgove/0060ff3b656618e9136b

Silhouette Score

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

$a(i)$ mean distance between i and all other data points in the **same cluster**

$b(i)$ **smallest** mean distance of i to **all points** in **any other cluster**

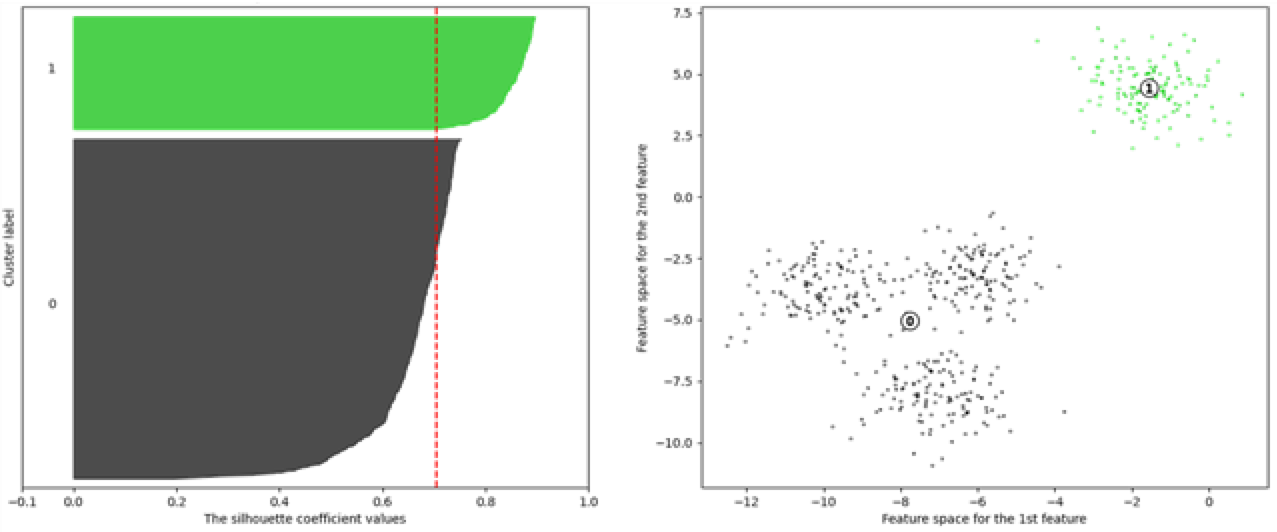


Next best fit cluster for point i

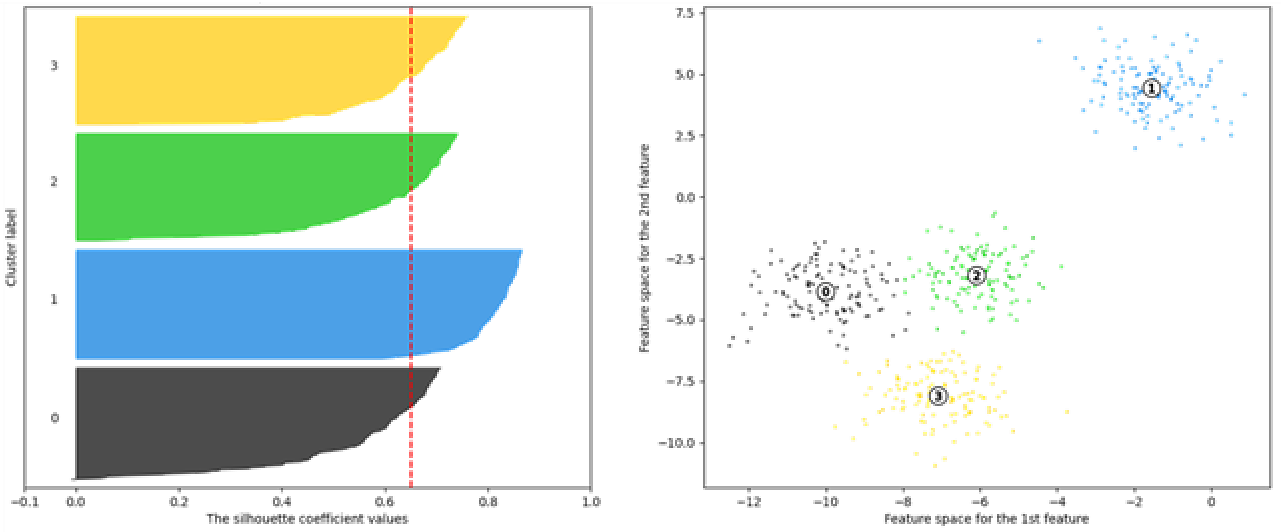
$$a(i) = \tfrac{1}{2}(d_1 + d_2)$$

$$b(i) = d_*$$

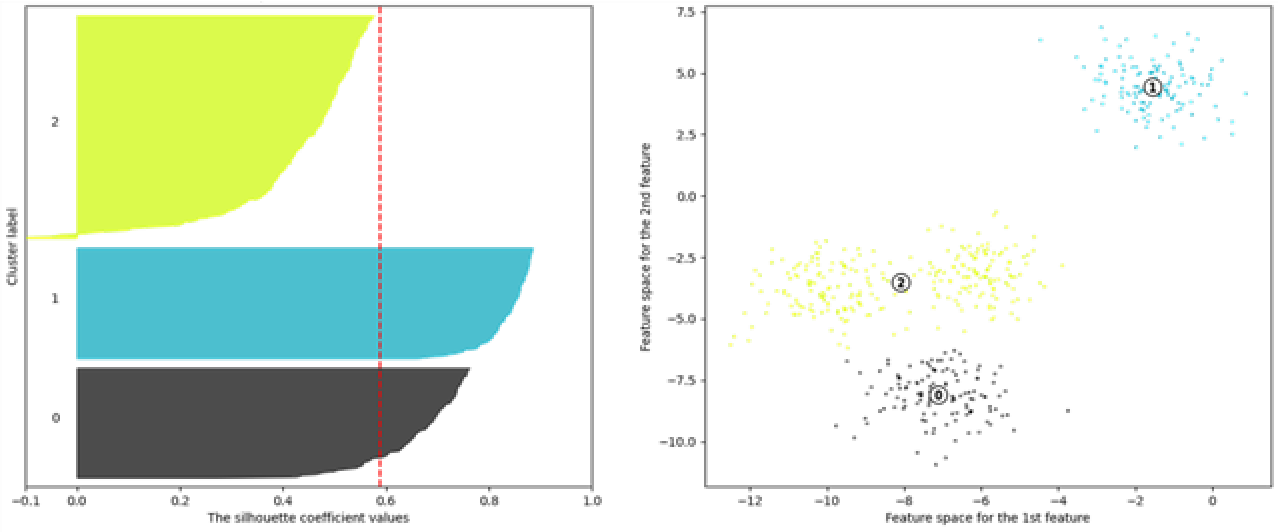
Clusters = 2, Score = 0.705



Clusters = 4, Score = 0.651



Clusters = 3, Score = 0.588



Clusters = 5, Score = 0.561

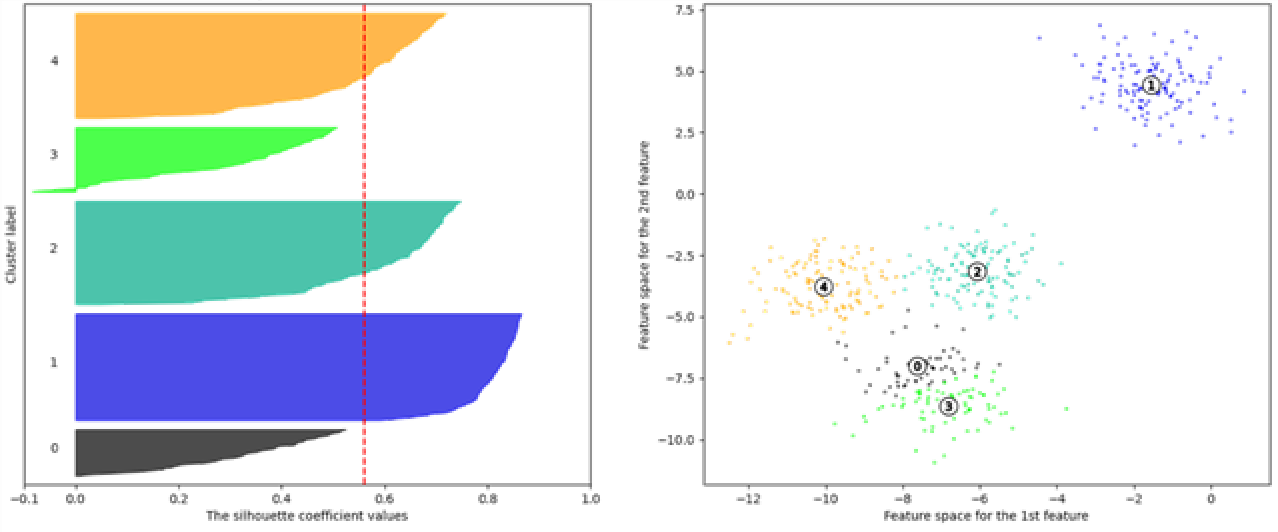


Image source: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Process for selecting a clustering approach

1. Apply clustering metrics to narrow down model parameter choices
2. Visualize the clusters (using dimensionality reduction techniques)
3. Use domain knowledge to determine whether the clusters make sense or add information to the business or scientific task
4. (optional) Check cluster stability (are the clusters consistent across runs)
5. (optional) Consider efficiency (will the model scale with larger datasets)

Types of clustering algorithms

Methods

Distribution-based clustering (e.g. **Gaussian mixture model**)
Centroid-based clustering (e.g. **K-Means**)
Density-based clustering (e.g. DBSCAN)
Hierarchical clustering (e.g. agglomerative clustering)
Graph-based clustering (e.g. spectral clustering)

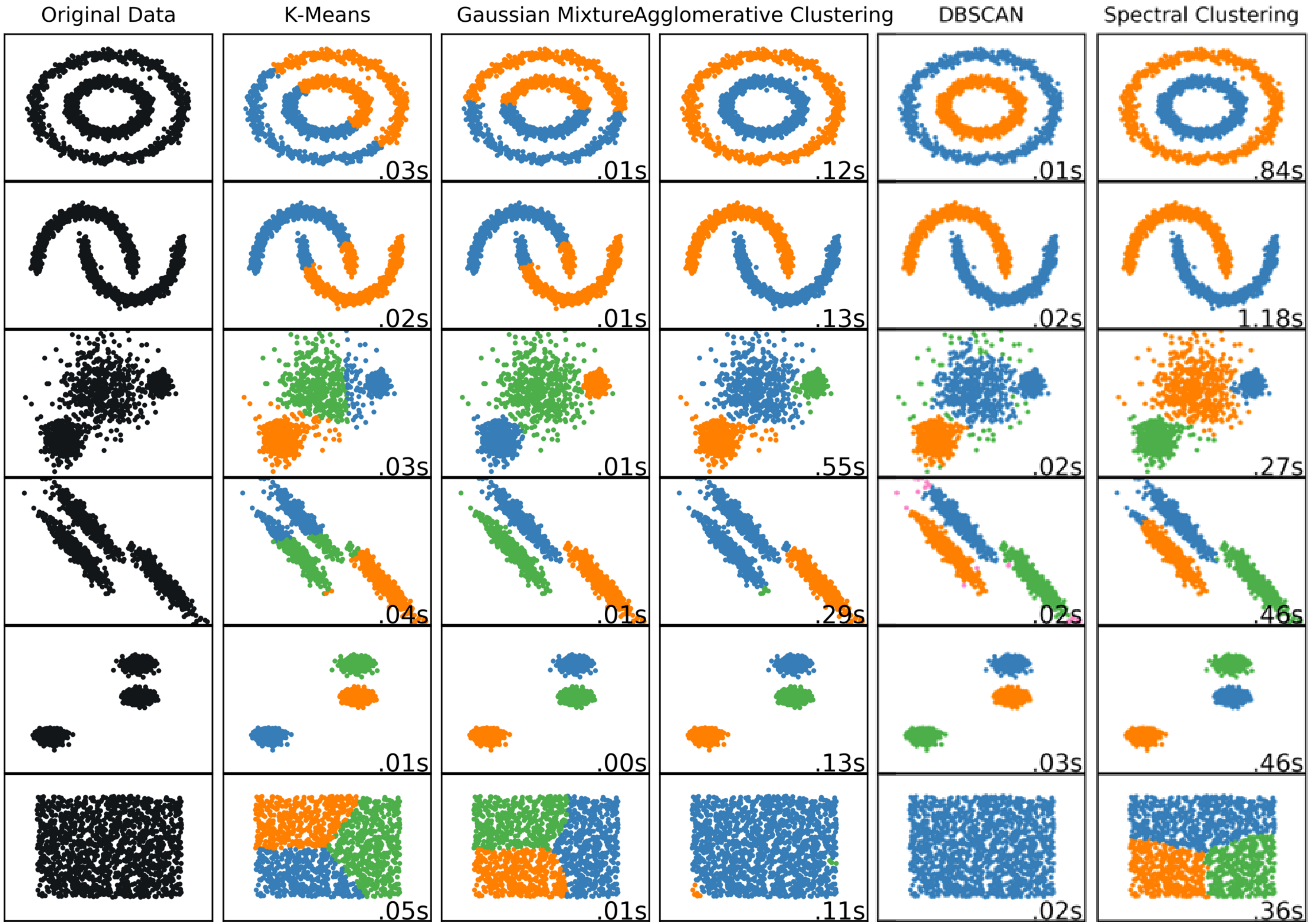
Cluster assignment

Hard clustering
Soft clustering (a.k.a. fuzzy clustering)

Agglomerative Clustering

DBSCAN

Spectral Clustering



Hierarchical Clustering

agglomerative (bottom-up) clustering

divisive (top-down) clustering

Agglomerative clustering components

Distance metric

How we measure distance/dissimilarity

Euclidean distance (L_2 norm) $\quad D(\boldsymbol{a},\boldsymbol{b}) = \|\boldsymbol{a}-\boldsymbol{b}\|_2$

Squared Euclidean distance $\quad D(\boldsymbol{a},\boldsymbol{b}) = \|\boldsymbol{a}-\boldsymbol{b}\|_2^2$

Manhattan distance (L_1 norm) $\quad D(\boldsymbol{a},\boldsymbol{b}) = \|\boldsymbol{a}-\boldsymbol{b}\|_1$

Maximum distance

$$D(\boldsymbol{a},\boldsymbol{b}) = \|\boldsymbol{a}-\boldsymbol{b}\|_\infty = \max_i |a_i - b_i|$$

Linkage criterion

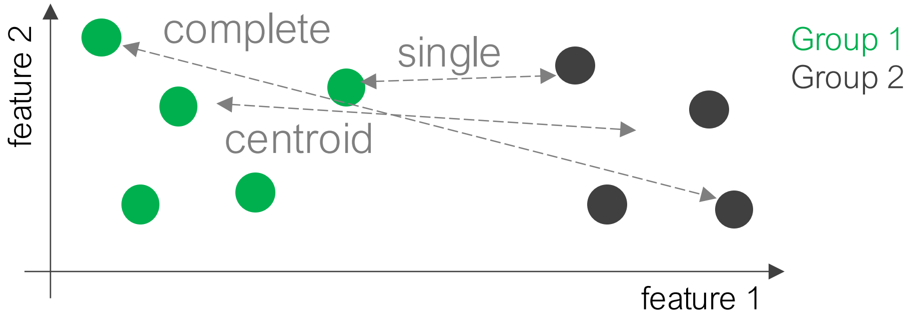
How to measure distance/dissimilarity between groups or sets

Complete = maximum intercluster dissimilarity

Single = minimum intercluster dissimilarity

Average = average intercluster dissimilarity (calculate the dissimilarity between all pairs of points, take the average)

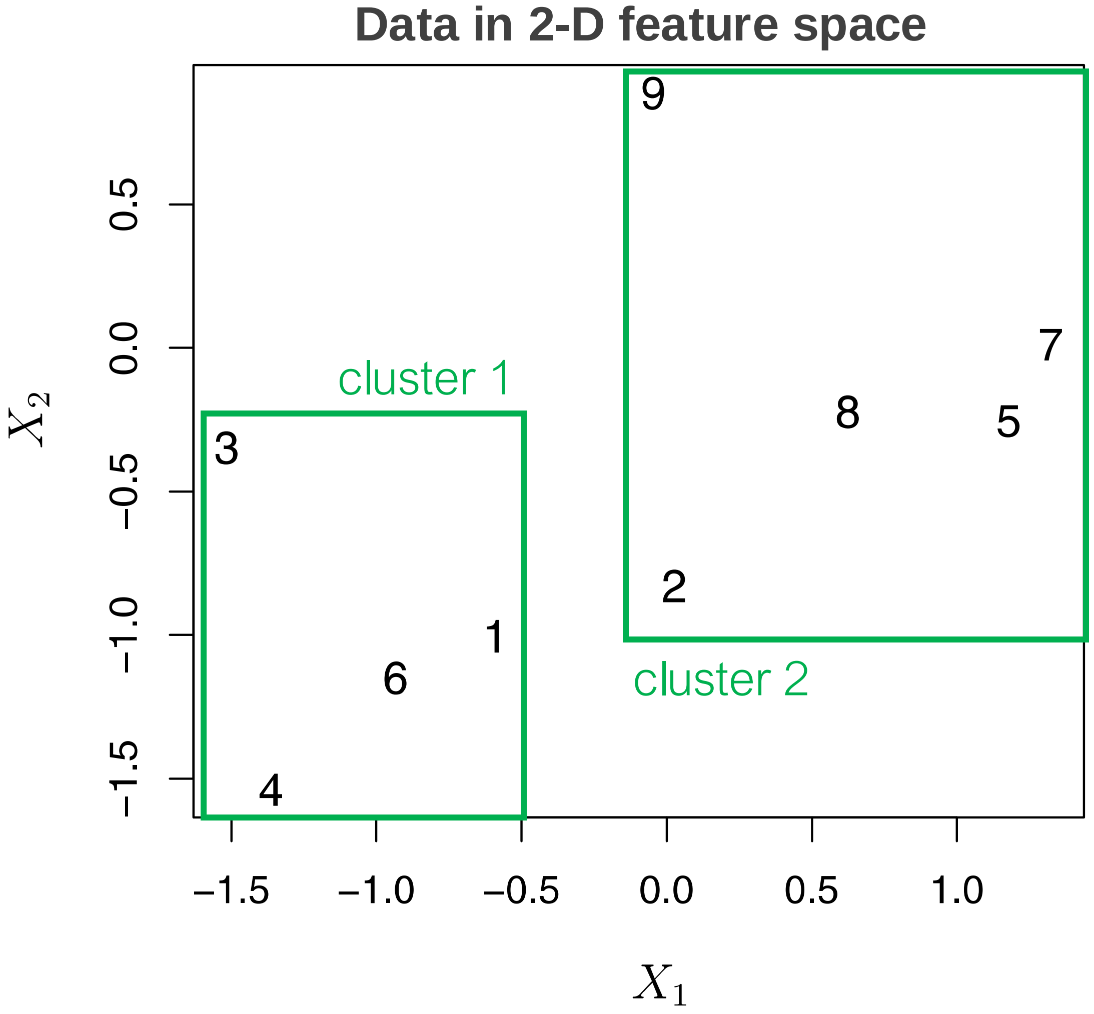
Centroid = dissimilarity between cluster centroids

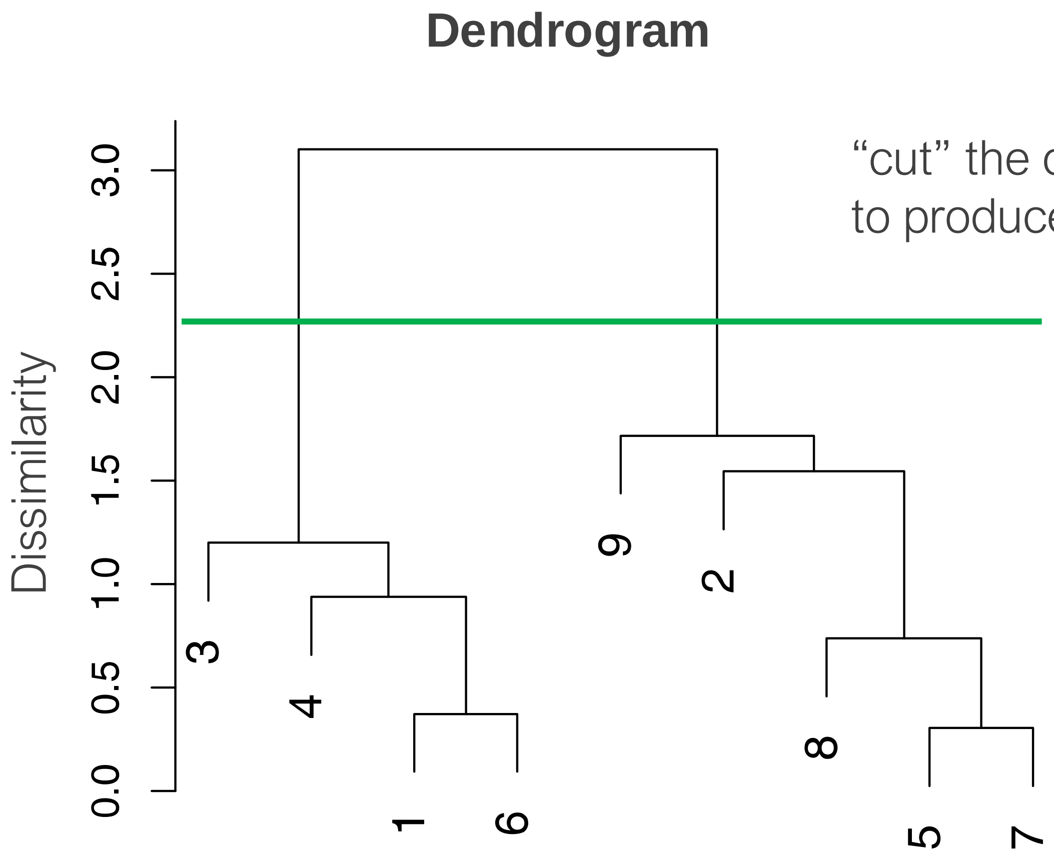


Agglomerative clustering

With complete linkage and Euclidean distance

Algorithm:
1. Select a measure of dissimilarity and linkage
2. Set each observation as a unique cluster
3. Group the two closest clusters together
4. Repeat until there is only one cluster





“cut” the dendrogra
to produce clusters

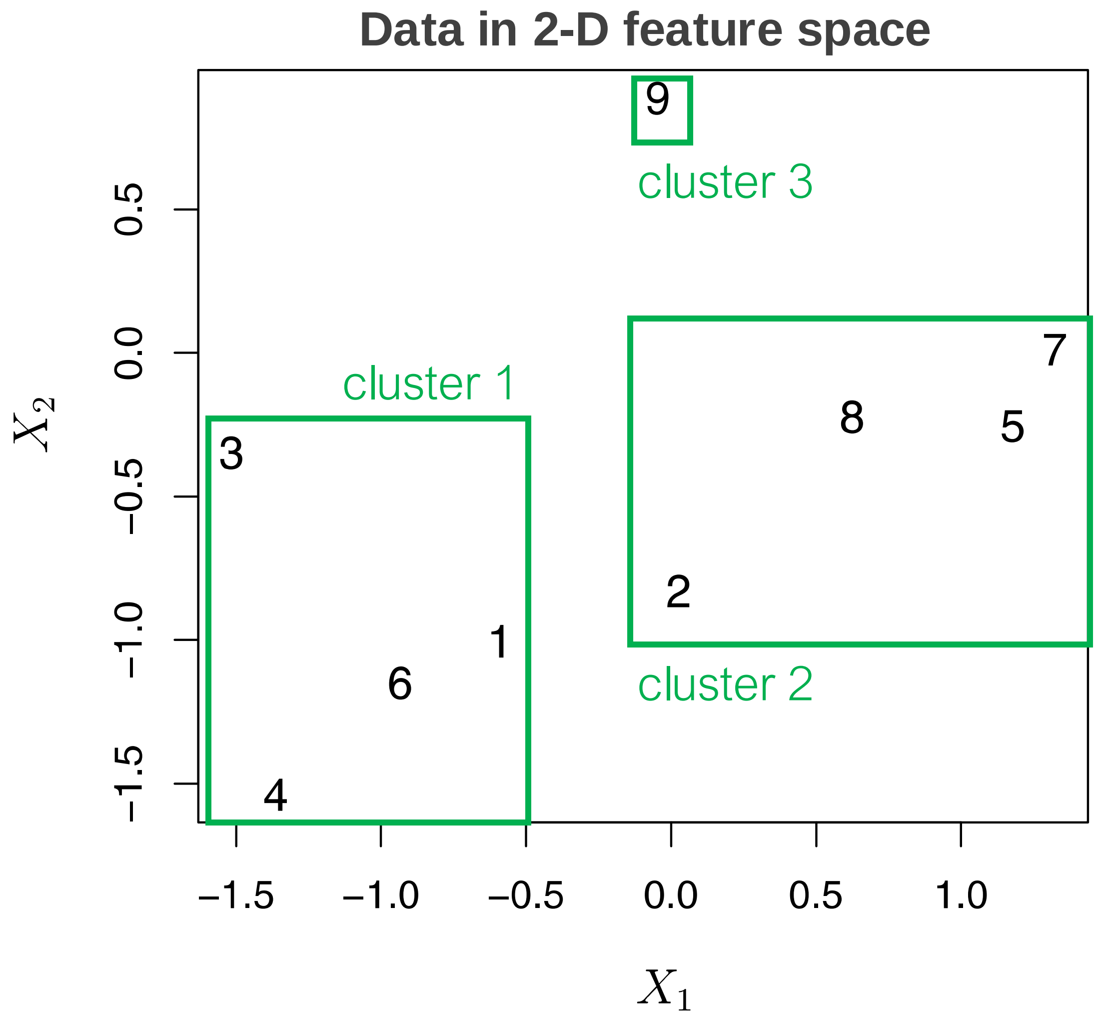
Image from James et al., Introduction to Statistical Learning, 2013

Agglomerative clustering

With complete linkage and Euclidean distance

Algorithm:

1. Select a measure of dissimilarity and linkage
2. Set each observation as a unique cluster
3. Group the two closest clusters together
4. Repeat until there is only one cluster



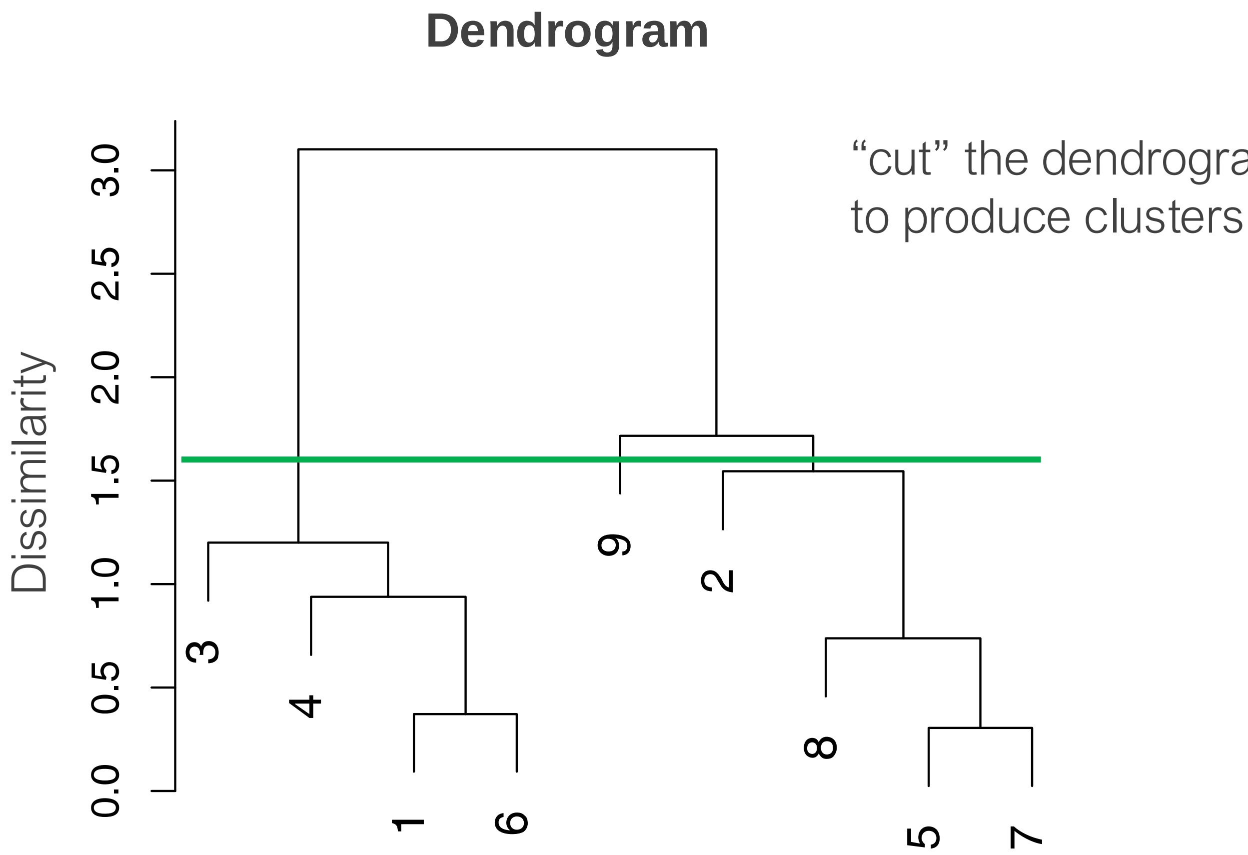


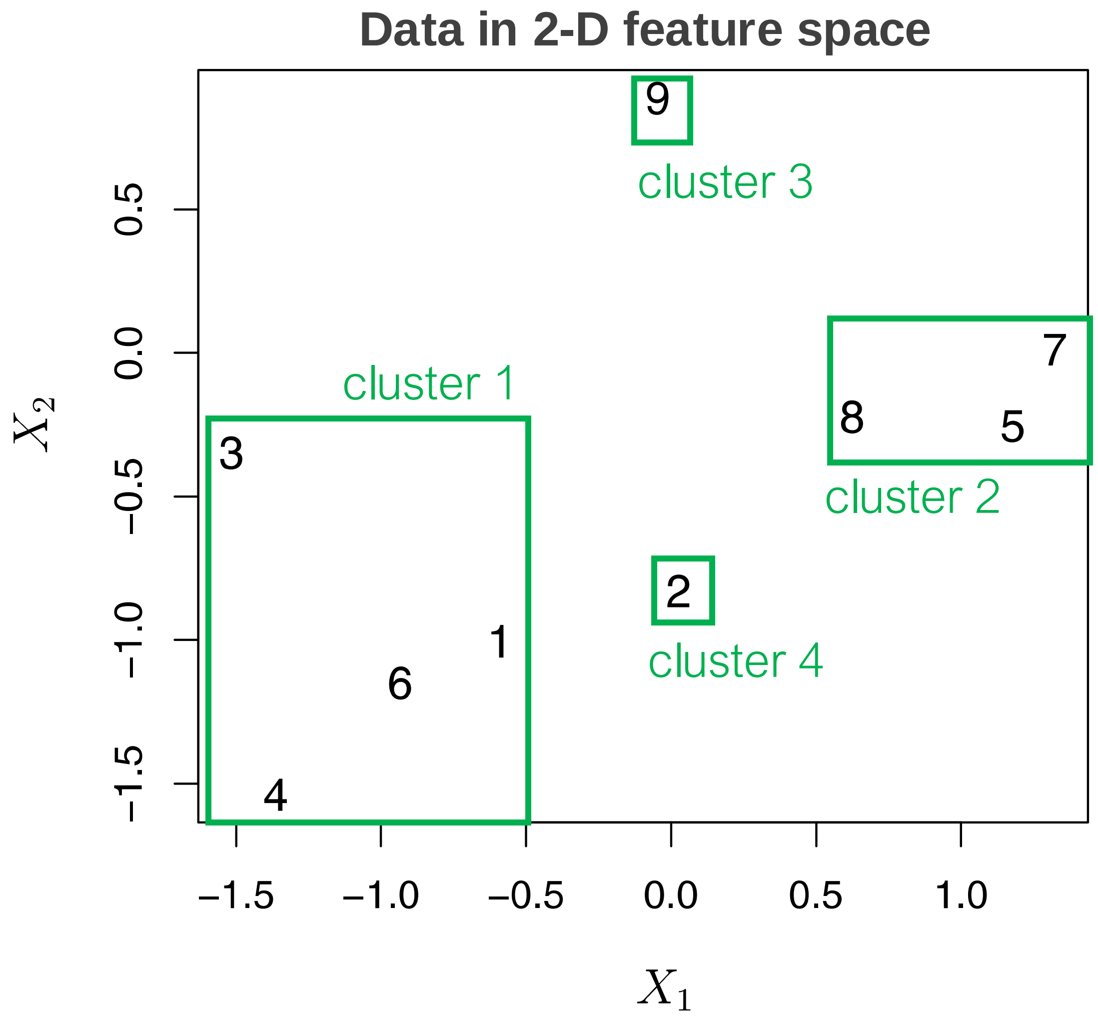
Image from James et al., Introduction to Statistical Learning, 2013

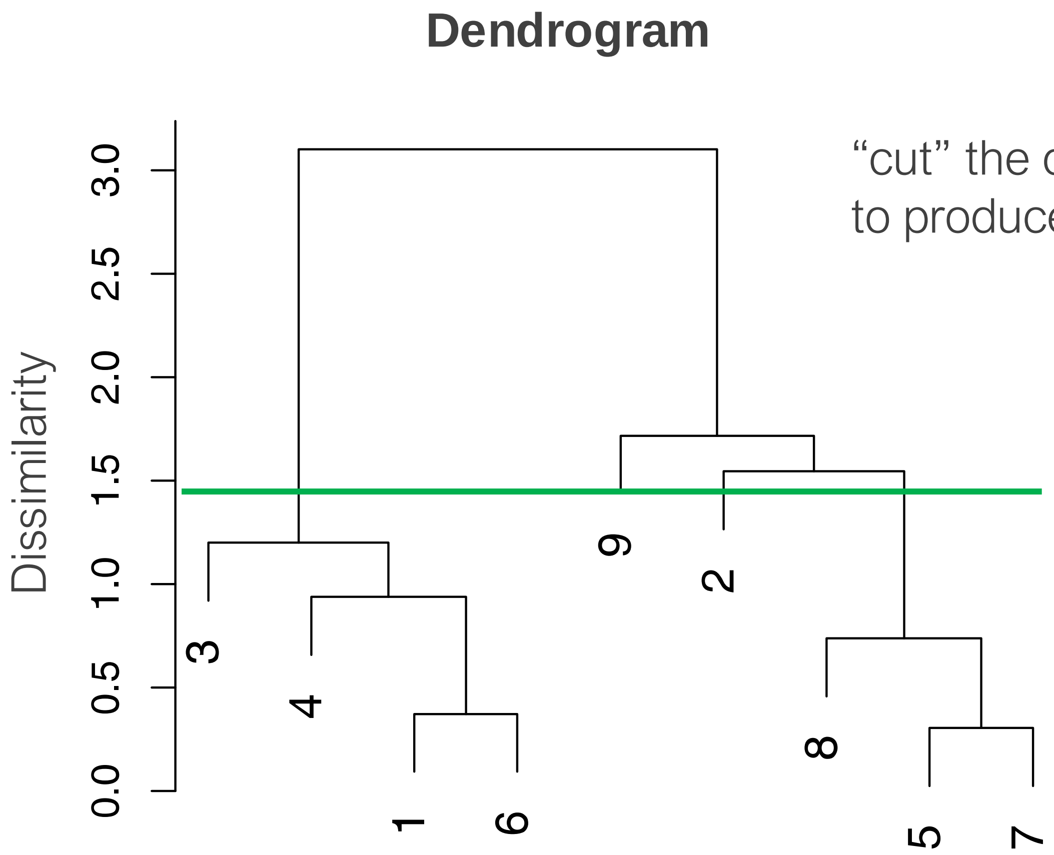
Agglomerative clustering

With complete linkage and Euclidean distance

Algorithm:

1. Select a measure of dissimilarity and linkage
2. Set each observation as a unique cluster
3. Group the two closest clusters together
4. Repeat until there is only one cluster



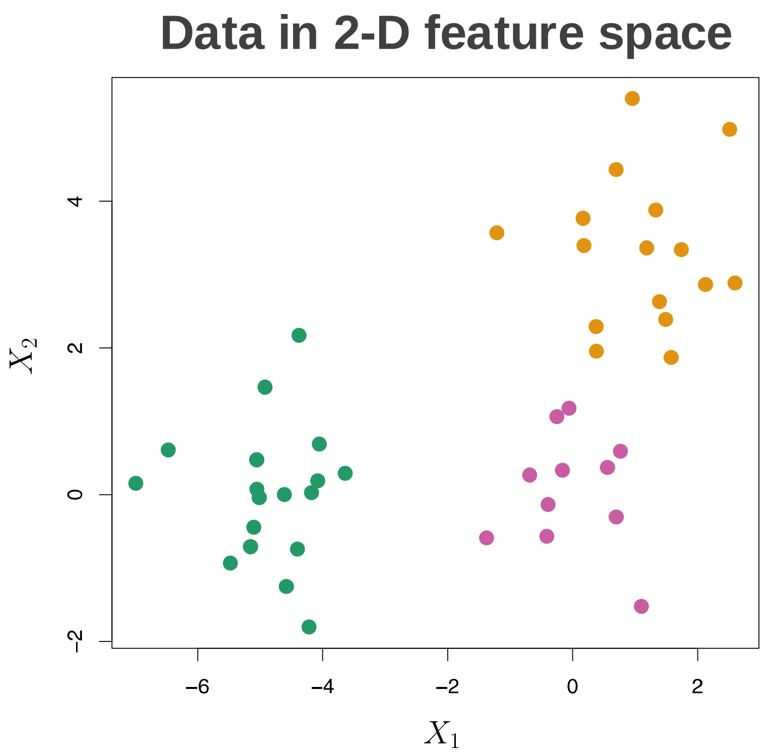


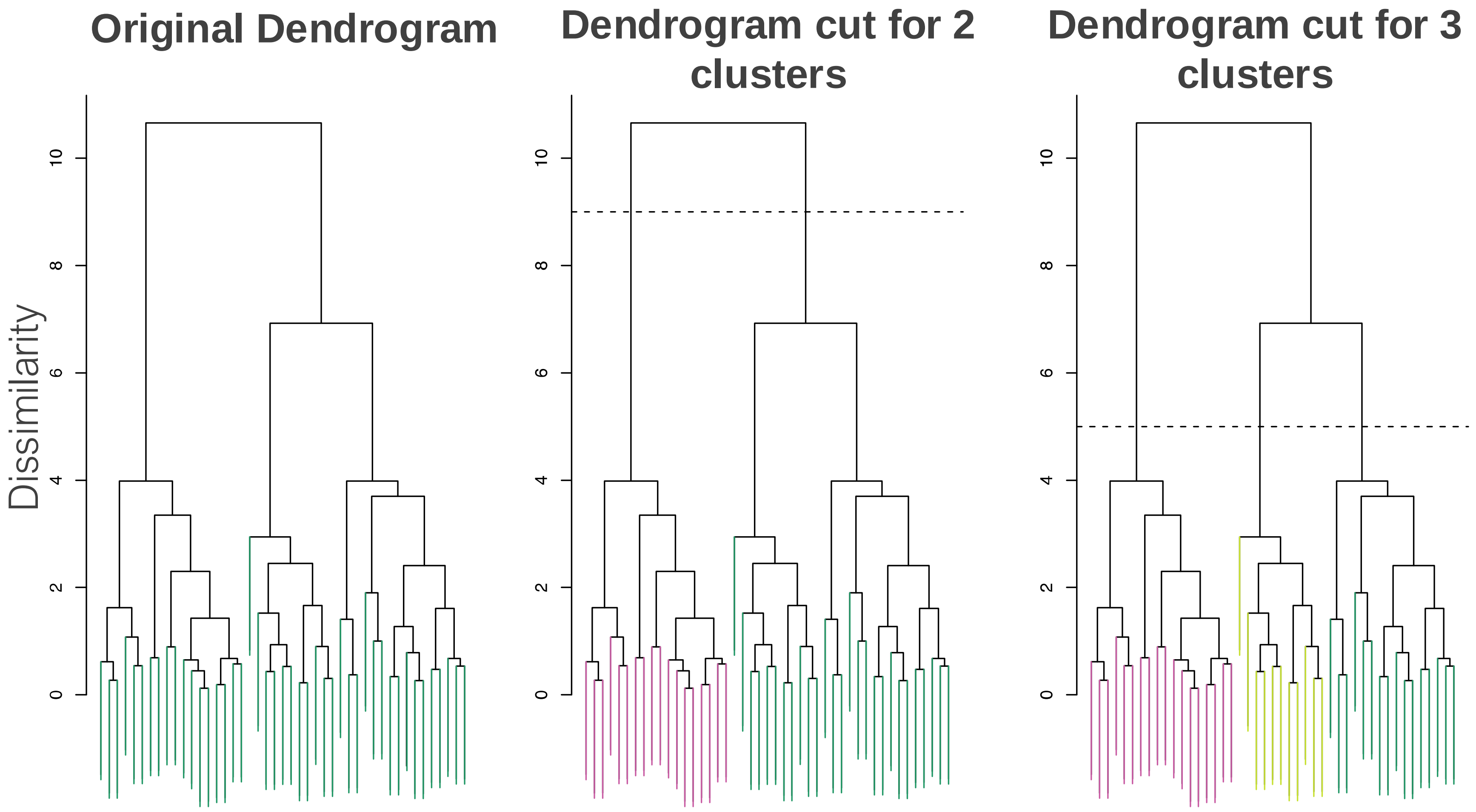
"cut" the dendrogra
to produce clusters

Image from James et al., Introduction to Statistical Learning, 2013

Example of agglomerative clustering

With complete linkage and Euclidean distance

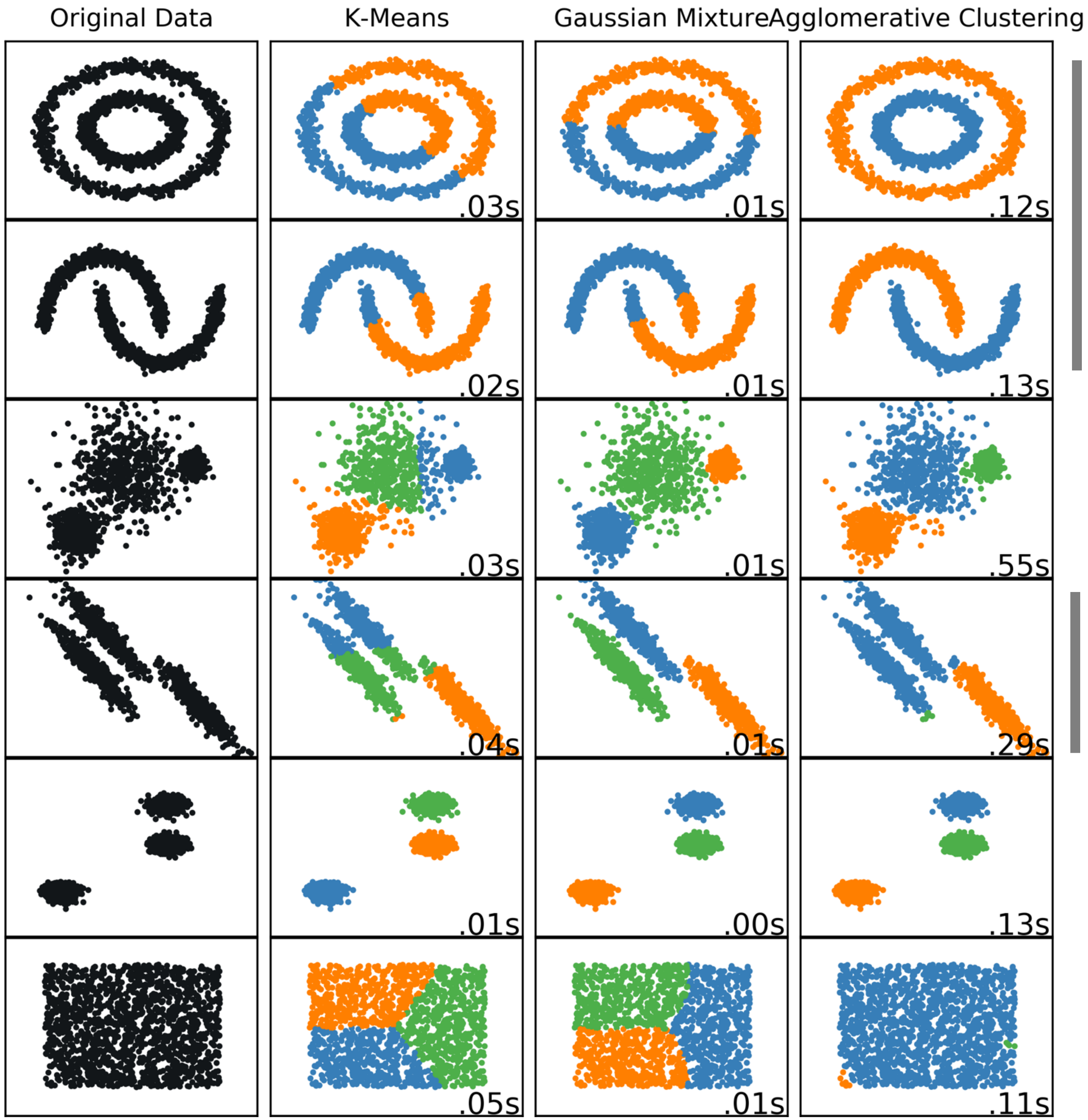




Note: colors do not directly map to plot on the left

Image from James et al., Introduction to Statistical Learning, 2013

Examples: Agglomerative clustering



Performs well when clusters are well-separated

Struggles when intercluster distance is not sufficient to distinguish between clusters

Need to choose where to cut the dendrogram

Can be slow since all pairwise distances between clusters need to be evaluated

DBSCAN Clustering

Density-based spatial clustering of applications with noise

By Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, 1996

DBSCAN

Parameters:

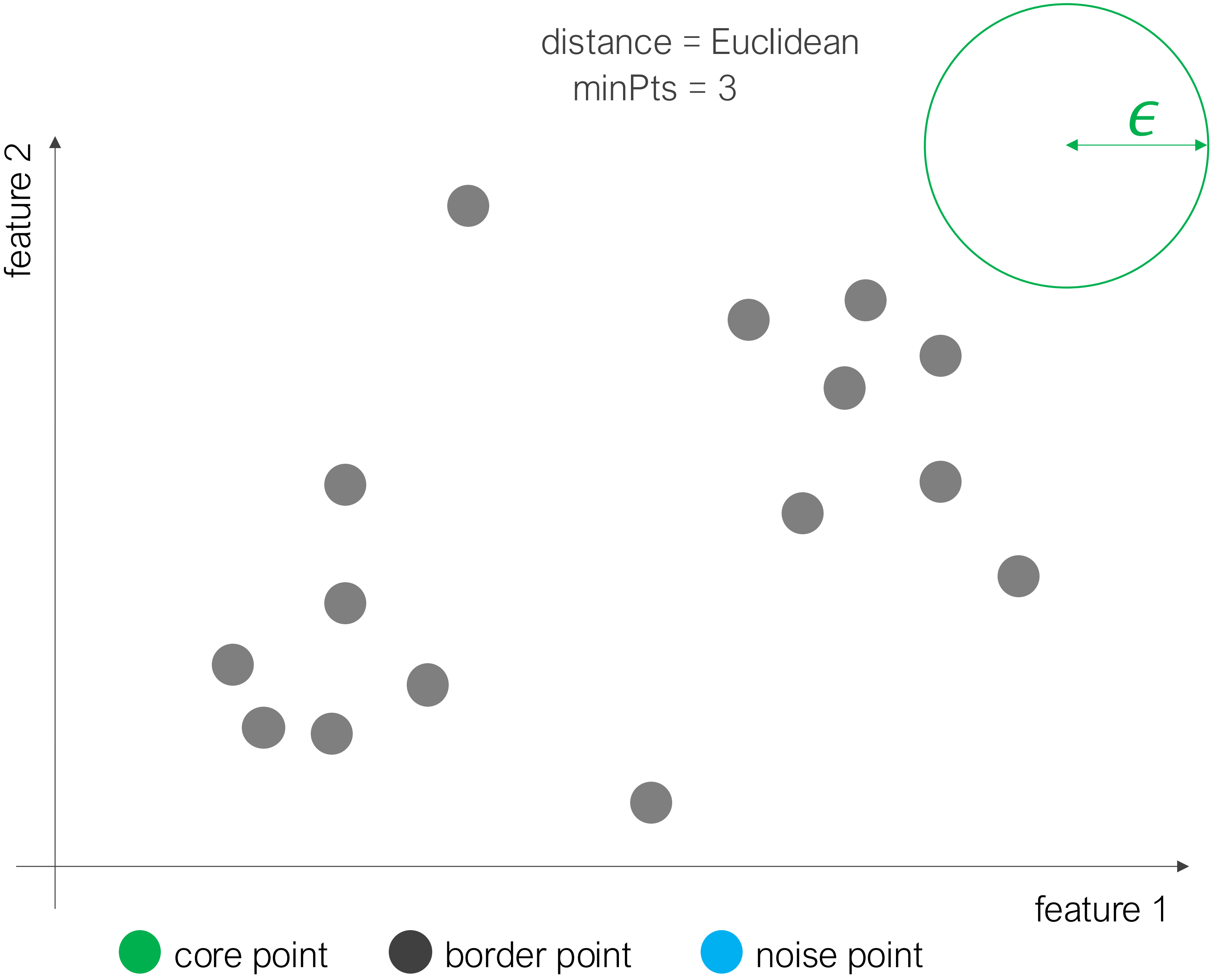
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

Types of points:

- **Core**: a point with at least minPts neighbors
- **Border**: a non-core point that neighbors a core point
- **Noise**: Other points

Algorithm:

1. Label core and border points
2. Group neighboring core points
3. Add border points that are neighbors of core points



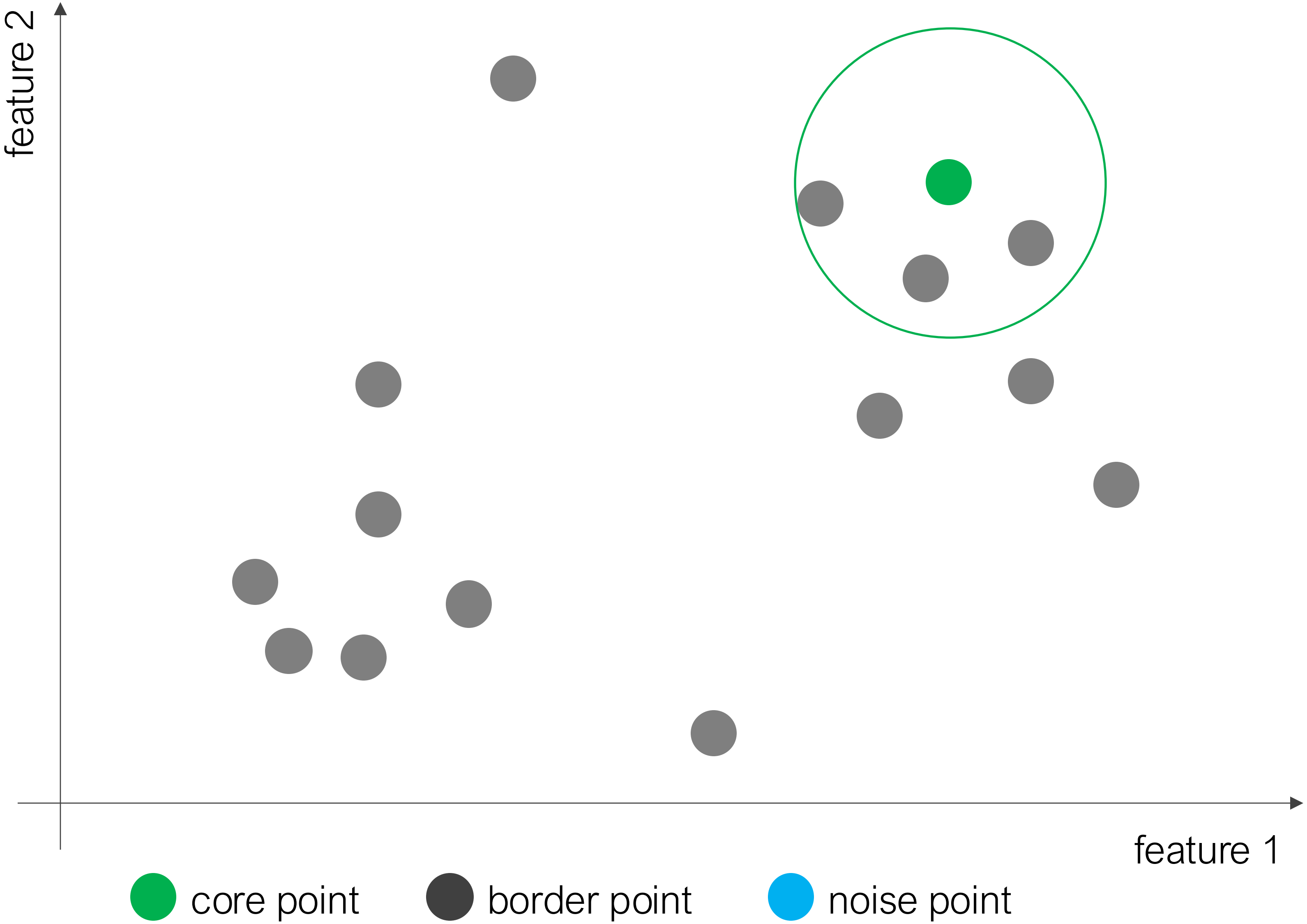
DBSCAN

Parameters:
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

Types of points:
- **Core**: a point with at least minPts neighbors
- **Border**: a non-core point that neighbors a core point
- **Noise**: Other points

Algorithm:
1. Label core and border points
2. Group neighboring core points
3. Add border points that are neighbors of core points

distance = Euclidean
minPts = 3

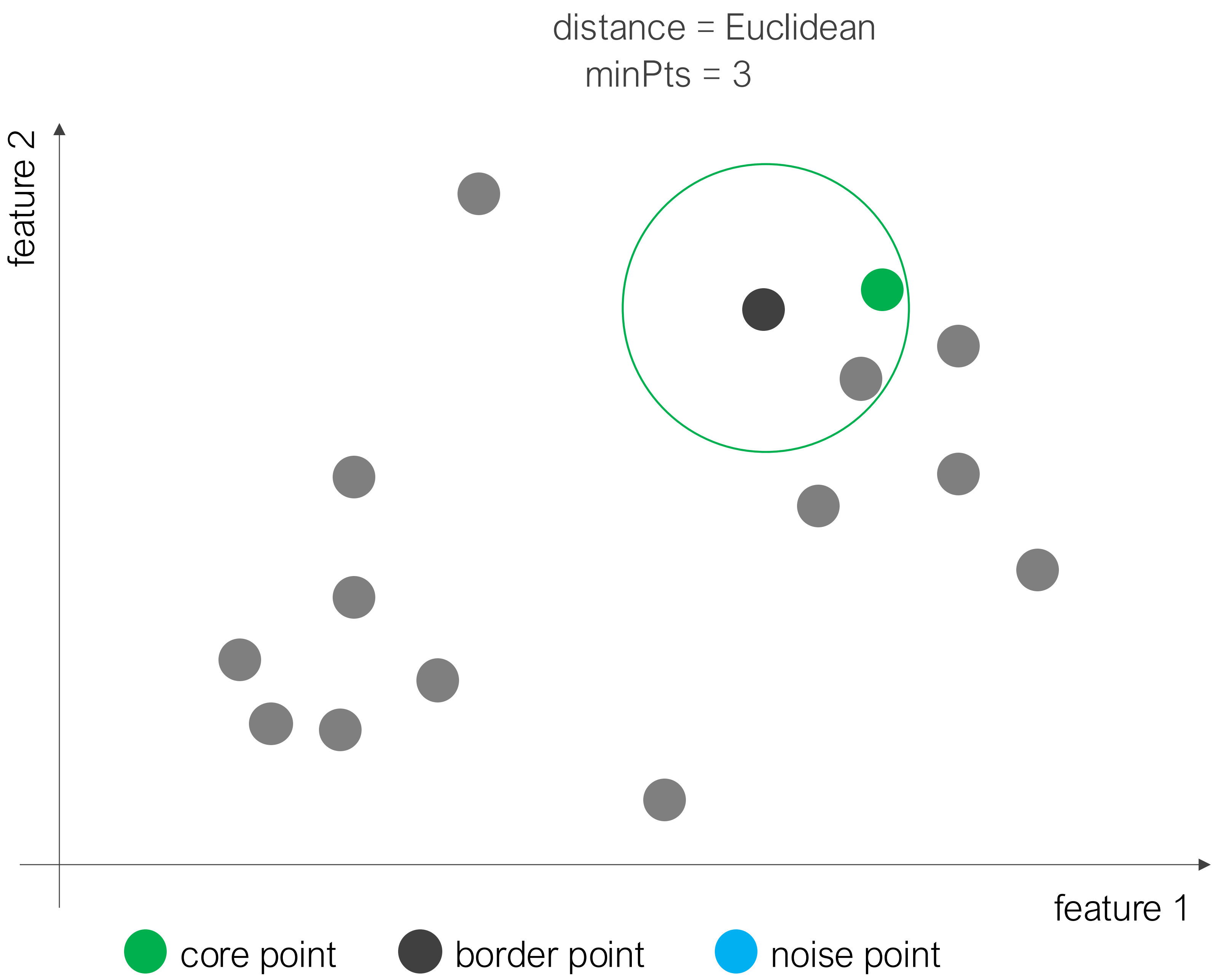


DBSCAN

Parameters:
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

Types of points:
- **Core**: a point with at least minPts neighbors
- **Border**: a non-core point that neighbors a core point
- **Noise**: Other points

Algorithm:
1. Label core and border points
2. Group neighboring core points
3. Add border points that are neighbors of core points



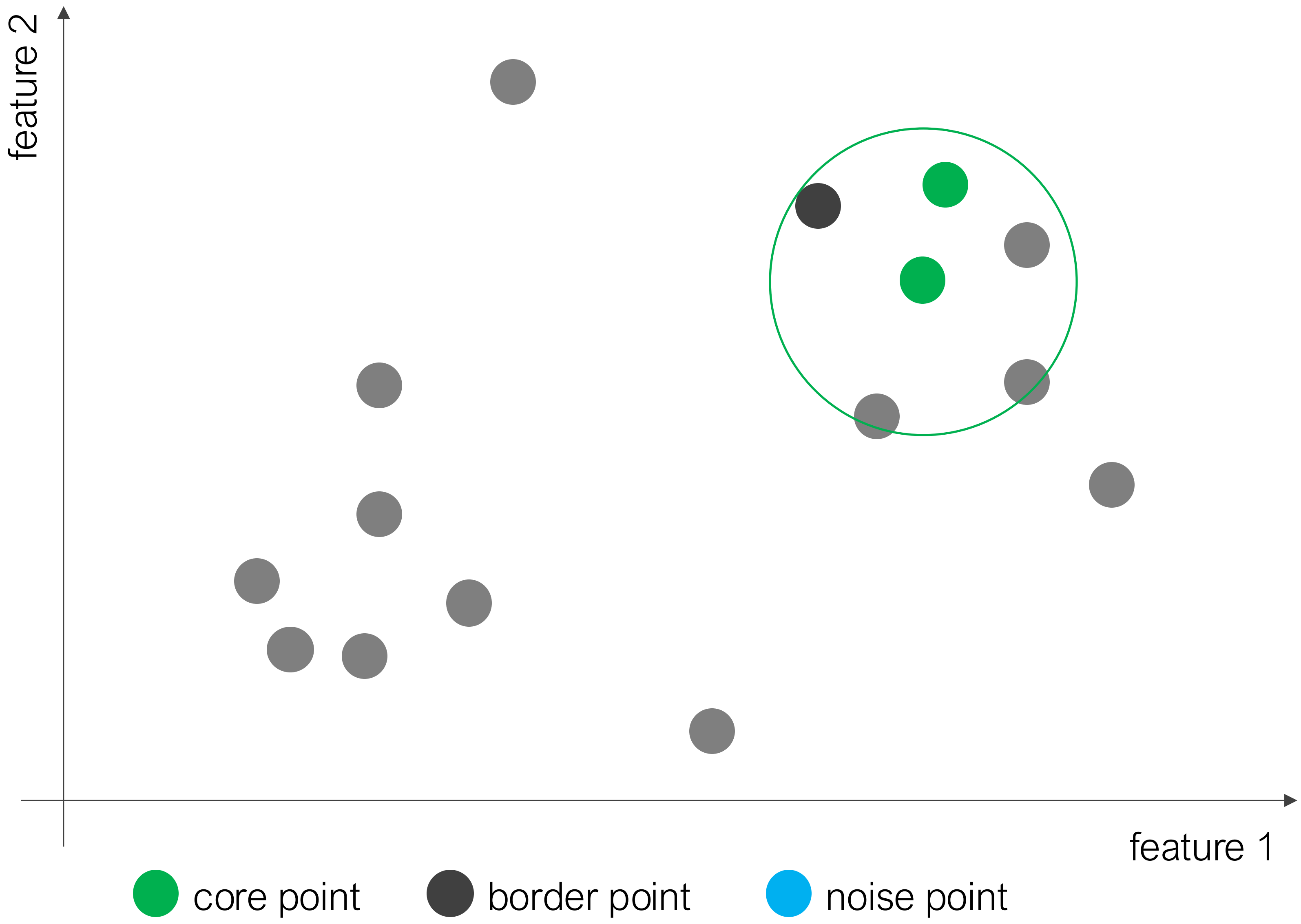
DBSCAN

Parameters:
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

Types of points:
- **Core**: a point with at least minPts neighbors
- **Border**: a non-core point that neighbors a core point
- **Noise**: Other points

Algorithm:
1. Label core and border points
2. Group neighboring core points
3. Add border points that are neighbors of core points

distance = Euclidean
minPts = 3

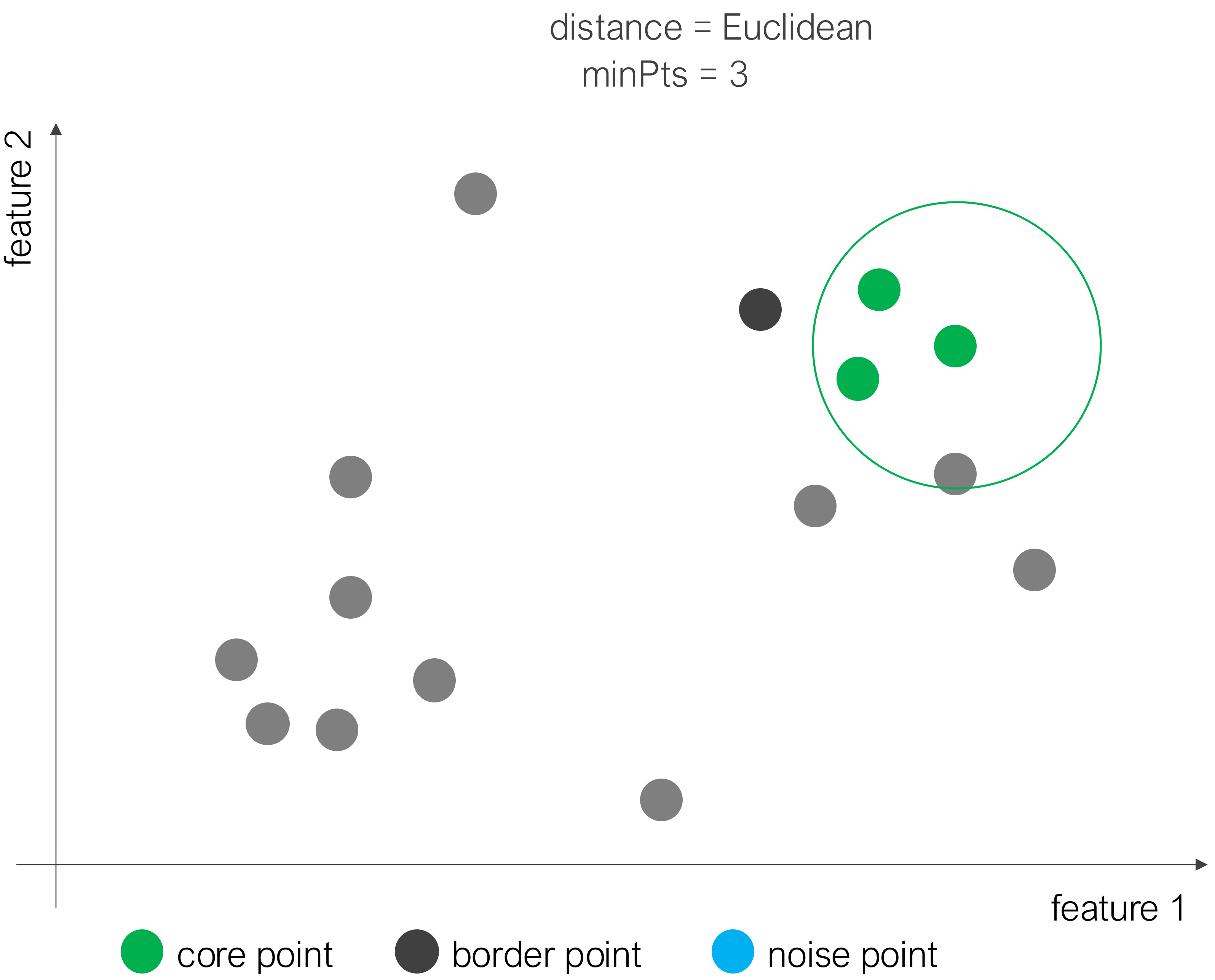


DBSCAN

Parameters:
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

Types of points:
- **Core**: a point with at least minPts neighbors
- **Border**: a non-core point that neighbors a core point
- **Noise**: Other points

Algorithm:
1. Label core and border points
2. Group neighboring core points
3. Add border points that are neighbors of core points

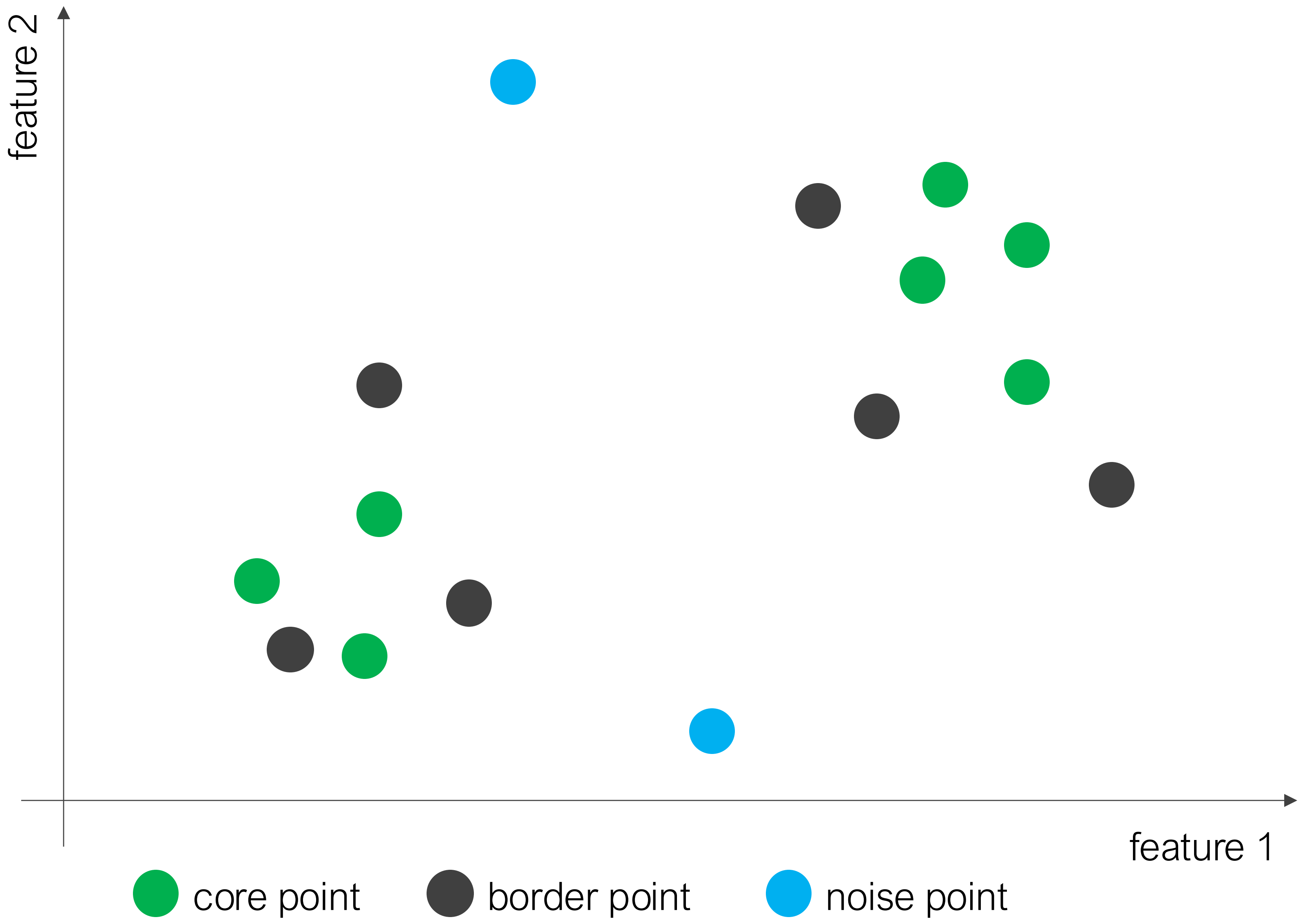


DBSCAN

Parameters:
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

Types of points:
- **Core**: a point with at least minPts neighbors
- **Border**: a non-core point that neighbors a core point
- **Noise**: Other points

Algorithm:
1. Label core and border points
2. Group neighboring core points
3. Add border points that are neighbors of core points

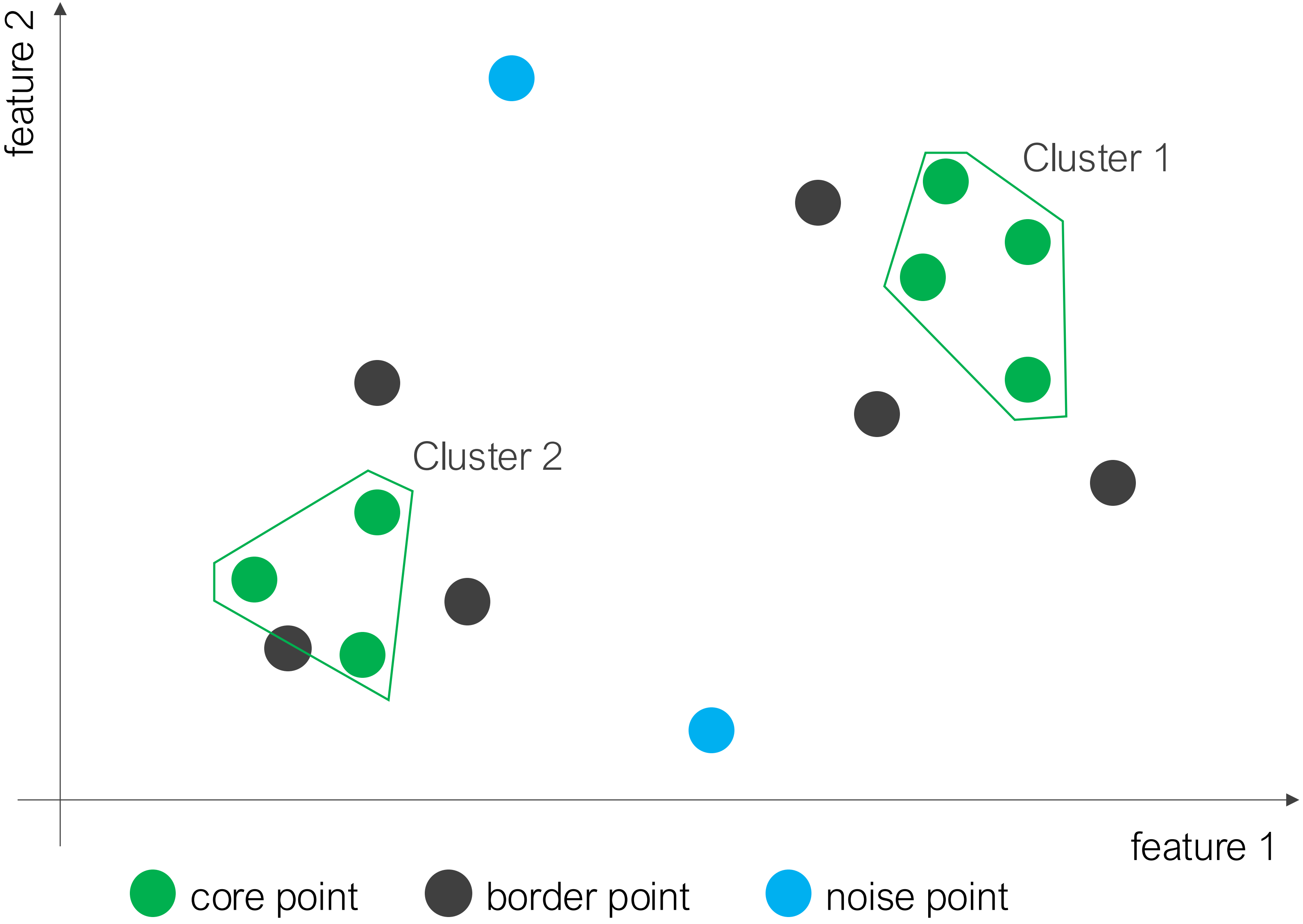


DBSCAN

Parameters:
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

Types of points:
- **Core**: a point with at least minPts neighbors
- **Border**: a non-core point that neighbors a core point
- **Noise**: Other points

Algorithm:
1. Label core and border points
2. Group neighboring core points
3. Add border points that are neighbors of core points

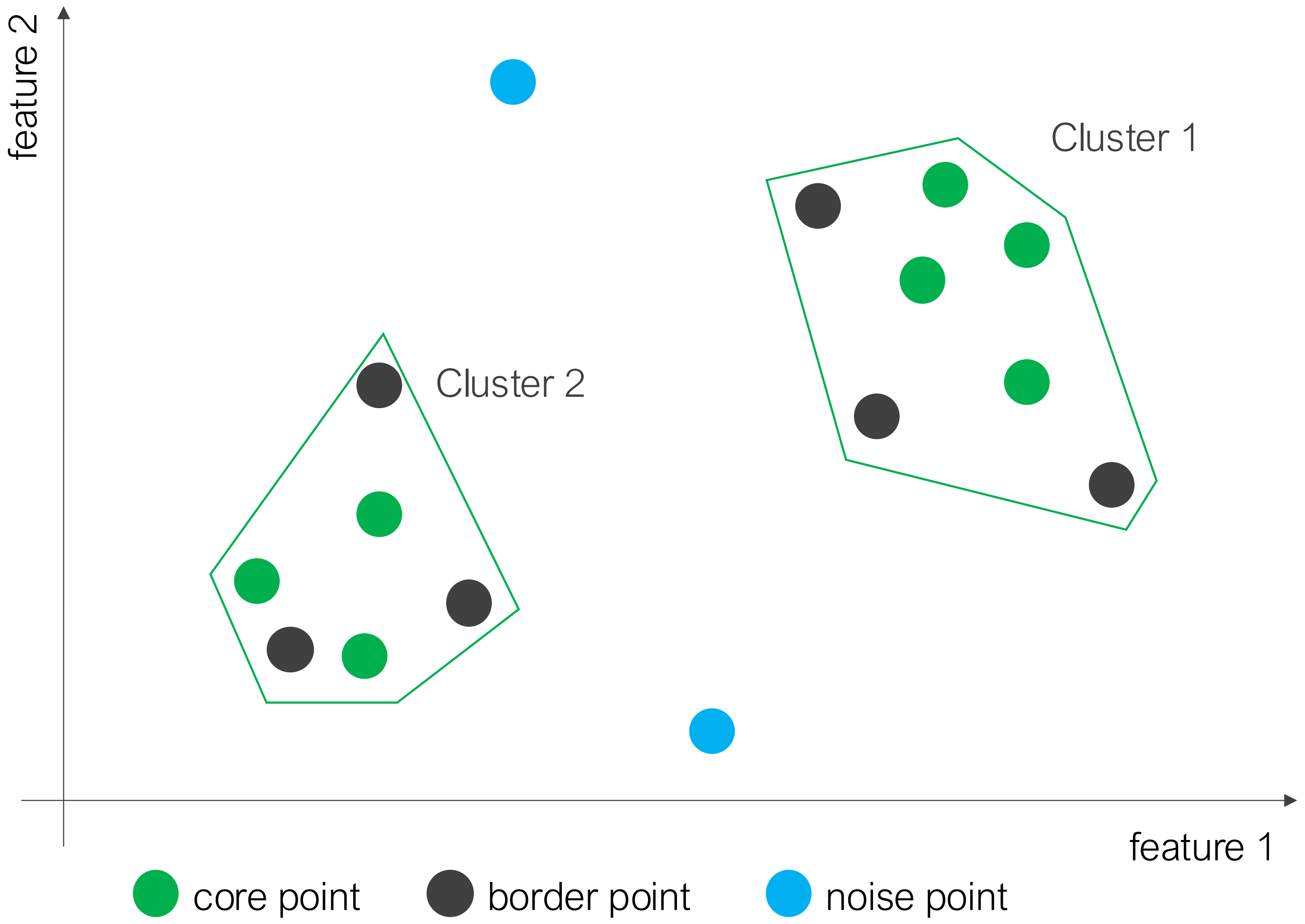


DBSCAN

Parameters:
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

Types of points:
- **Core**: a point with at least minPts neighbors
- **Border**: a non-core point that neighbors a core point
- **Noise**: Other points

Algorithm:
1. Label core and border points
2. Group neighboring core points
3. Add border points that are neighbors of core points

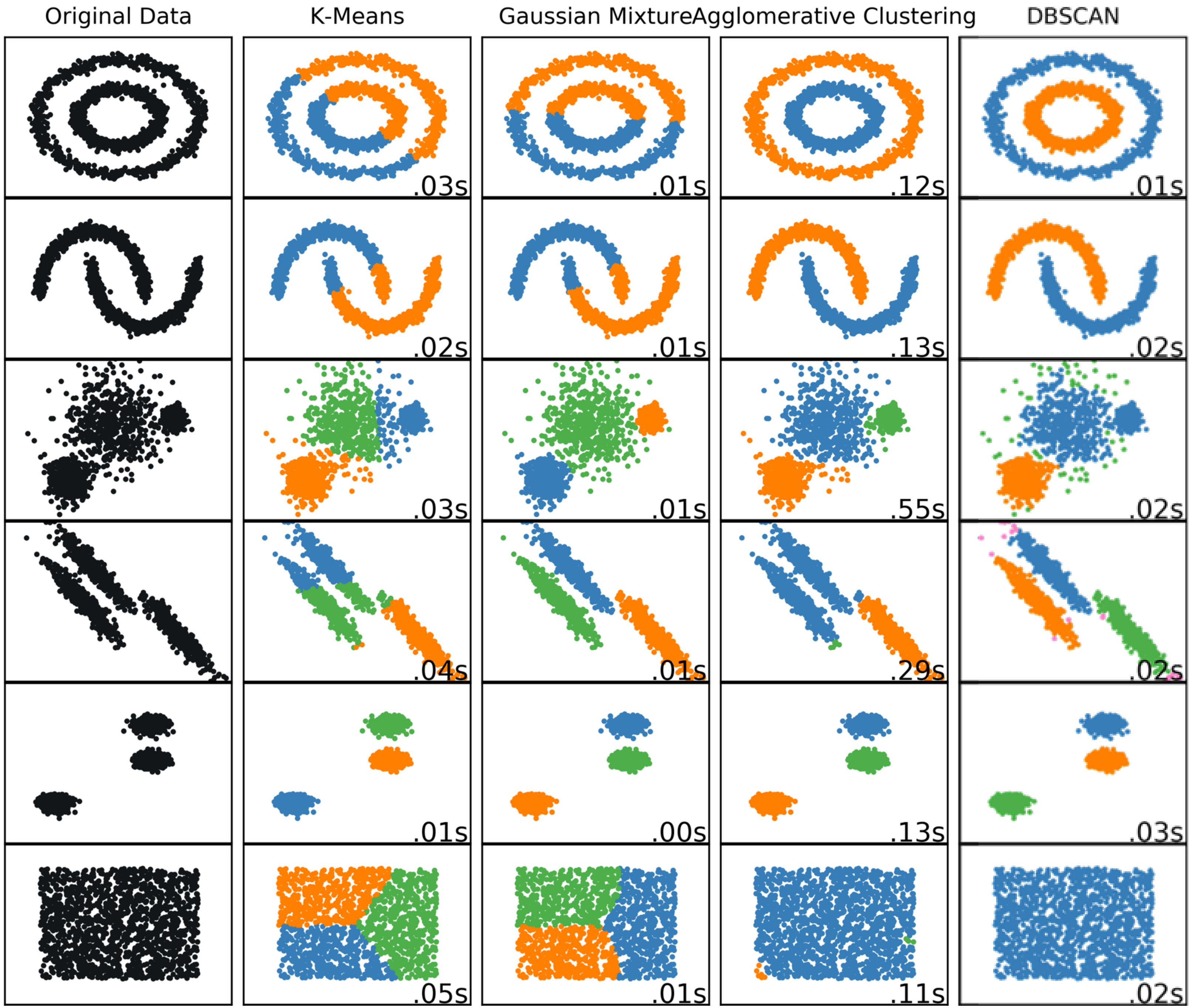


DBSCAN

- The number of clusters is chosen as part of the algorithm
- Can find arbitrarily shaped clusters
- Robust to outliers

- Cannot handle significant variation in cluster density
- Not entirely deterministic (border points reachable from more than one cluster may be assigned to either)

Examples: DBSCAN



Need to choose the density parameters

Does not require selecting the number of clusters beforehand

Spectral Clustering

Graph-based clustering based on data similarity

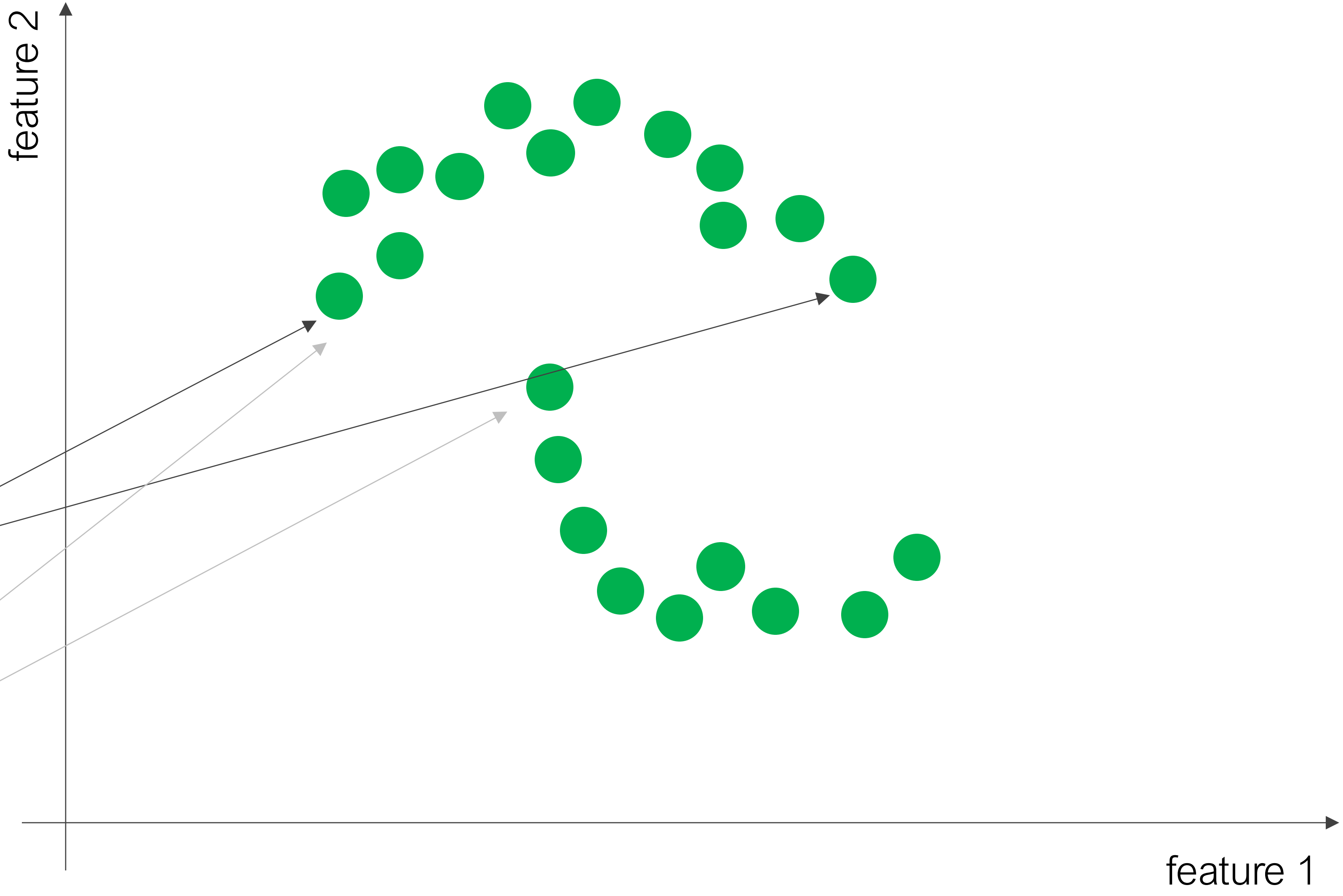
Spectral Clustering

Focuses on **connectedness** instead of compactness

The location alone does not determine **similarity** or "**affinity**"

These two points are likely connected by a cluster

These two points are NOT likely connected by a cluster



Concept from Sebastian Thrun and Peter Norvig

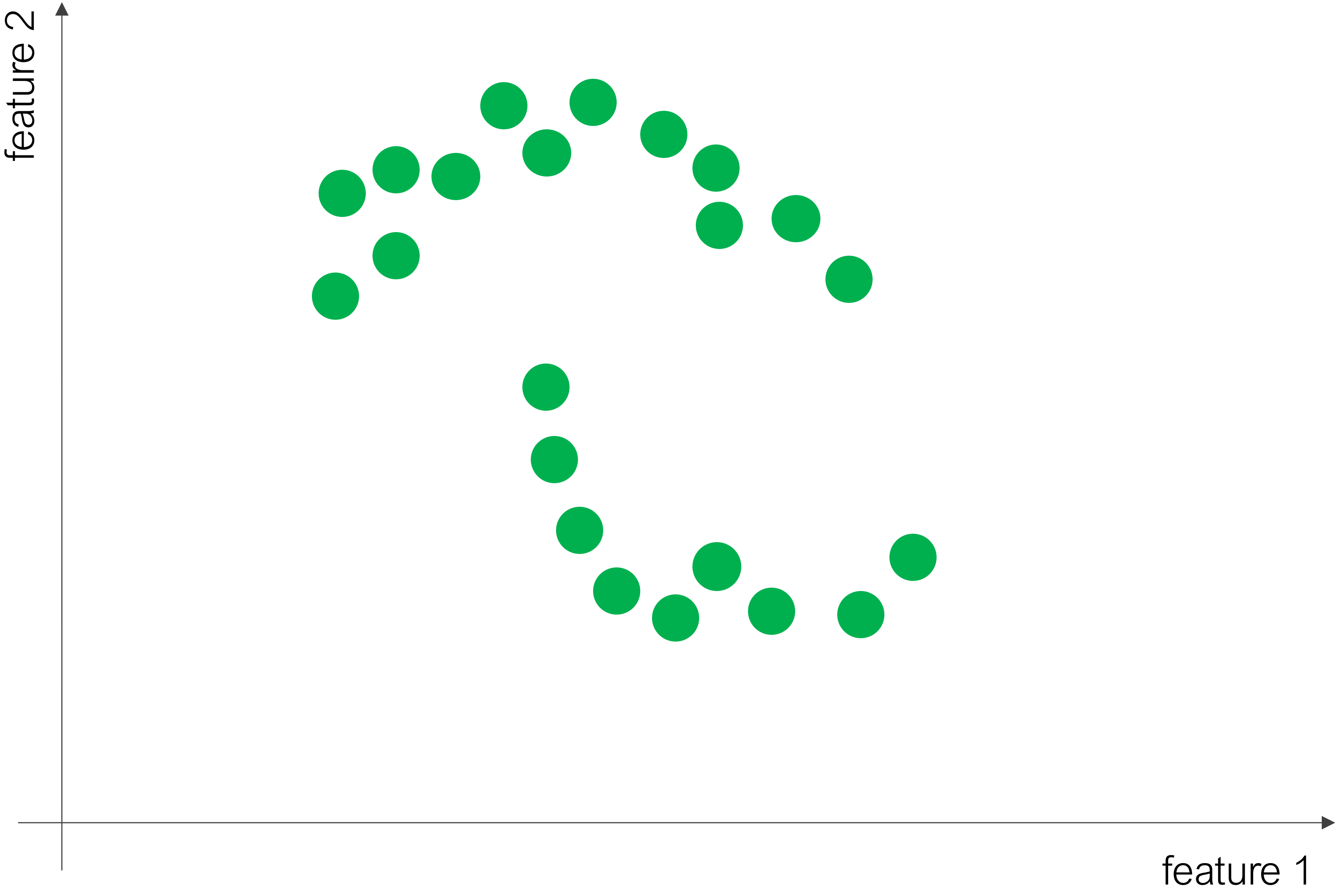
Spectral Clustering

Define **similarity** or **affinity** as the opposite of distance:

$$A(\boldsymbol{a}, \boldsymbol{b}) = -d(\boldsymbol{a}, \boldsymbol{b})$$

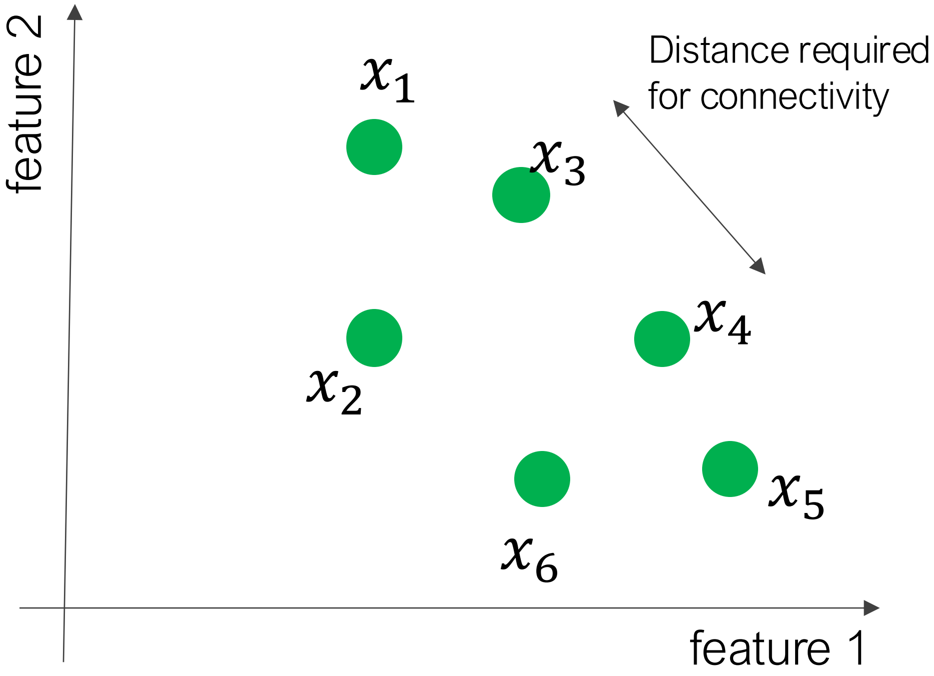
For example, using Euclidean distance, we could define affinity as:

$$A(\boldsymbol{a}, \boldsymbol{b}) = -\|\boldsymbol{a} - \boldsymbol{b}\|_2$$

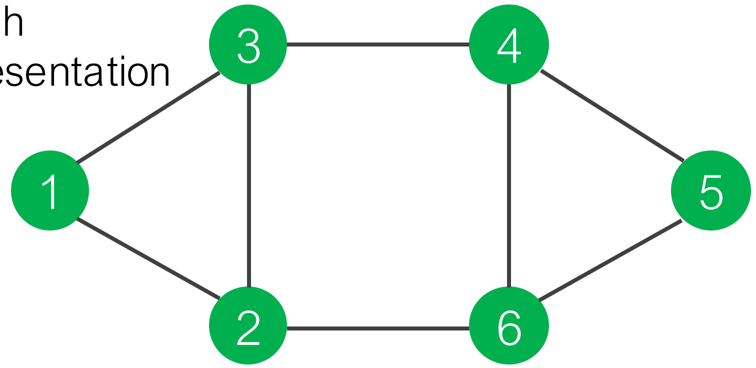


Concept from Sebastian Thrun and Peter Norvig

Spectral Clustering



Graph representation



Affinity Matrix (A)

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	1	1	0	0	0
x_2	1	0	1	0	0	1
x_3	1	1	0	1	0	0
x_4	0	0	1	0	1	1
x_5	0	0	0	1	0	1
x_6	0	1	0	1	1	0

If distance between points < threshold, consider there to be an "edge" connecting them in the graph

A vertex is not connected to itself

Degree Matrix (D)

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	2	0	0	0	0	0
x_2	0	3	0	0	0	0
x_3	0	0	3	0	0	0
x_4	0	0	0	3	0	0
x_5	0	0	0	0	2	0
x_6	0	0	0	0	0	3

The sum of edges connected to each vertex

Concept from Sebastian Thrun and Peter Norvig

Spectral Clustering

Degree Matrix (D)

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	2	0	0	0	0	0
x_2	0	3	0	0	0	0
x_3	0	0	3	0	0	0
x_4	0	0	0	3	0	0
x_5	0	0	0	0	3	0
x_6	0	0	0	0	0	2

Affinity Matrix (A)

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	1	1	0	0	0
x_2	1	0	1	0	0	1
x_3	1	1	0	1	0	0
x_4	0	0	1	0	1	1
x_5	0	0	0	1	0	1
x_6	0	1	0	1	1	0

Graph Laplacian Matrix (L)

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	2	-1	-1	0	0	0
x_2	-1	3	-1	0	0	-1
x_3	-1	-1	3	-1	0	0
x_4	0	0	-1	3	-1	-1
x_5	0	0	0	-1	2	-1
x_6	0	-1	0	-1	-1	3

$$\mathbf{D} - \mathbf{A} = \mathbf{L}$$

Spectral Clustering

Graph Laplacian Matrix (L)

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	2	-1	-1	0	0	0
x_2	-1	3	-1	0	0	-1
x_3	-1	-1	3	-1	0	0
x_4	0	0	-1	3	-1	-1
x_5	0	0	0	-1	2	-1
x_6	0	-1	0	-1	-1	3

L

Eigenvectors of L

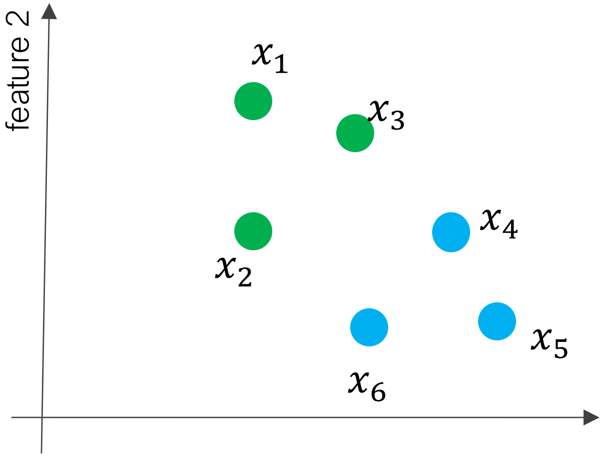
$\mathbf{u}_1$	$\mathbf{u}_2$	$\mathbf{u}_3$	$\mathbf{u}_4$	$\mathbf{u}_5$	$\mathbf{u}_6$
0.4	0.6	0.0	0.6	0.3	0.0
0.4	0.3	0.4	-0.4	-0.5	0.5
0.4	0.3	-0.4	-0.4	-0.1	-0.6
0.4	-0.3	-0.5	-0.1	0.3	0.6
0.3	-0.4	-0.2	0.5	-0.6	-0.1
0.5	-0.5	0.5	-0.1	0.4	-0.3

$\lambda_i =$

-0.1	1.0	2.7	3.3	4.1	4.8

Eigenvalues of L

	$\mathbf{u}_2$
x_1	**0.6**
x_2	**0.3**
x_3	**0.3**
x_4	**-0.3**
x_5	**-0.4**
x_6	**-0.5**

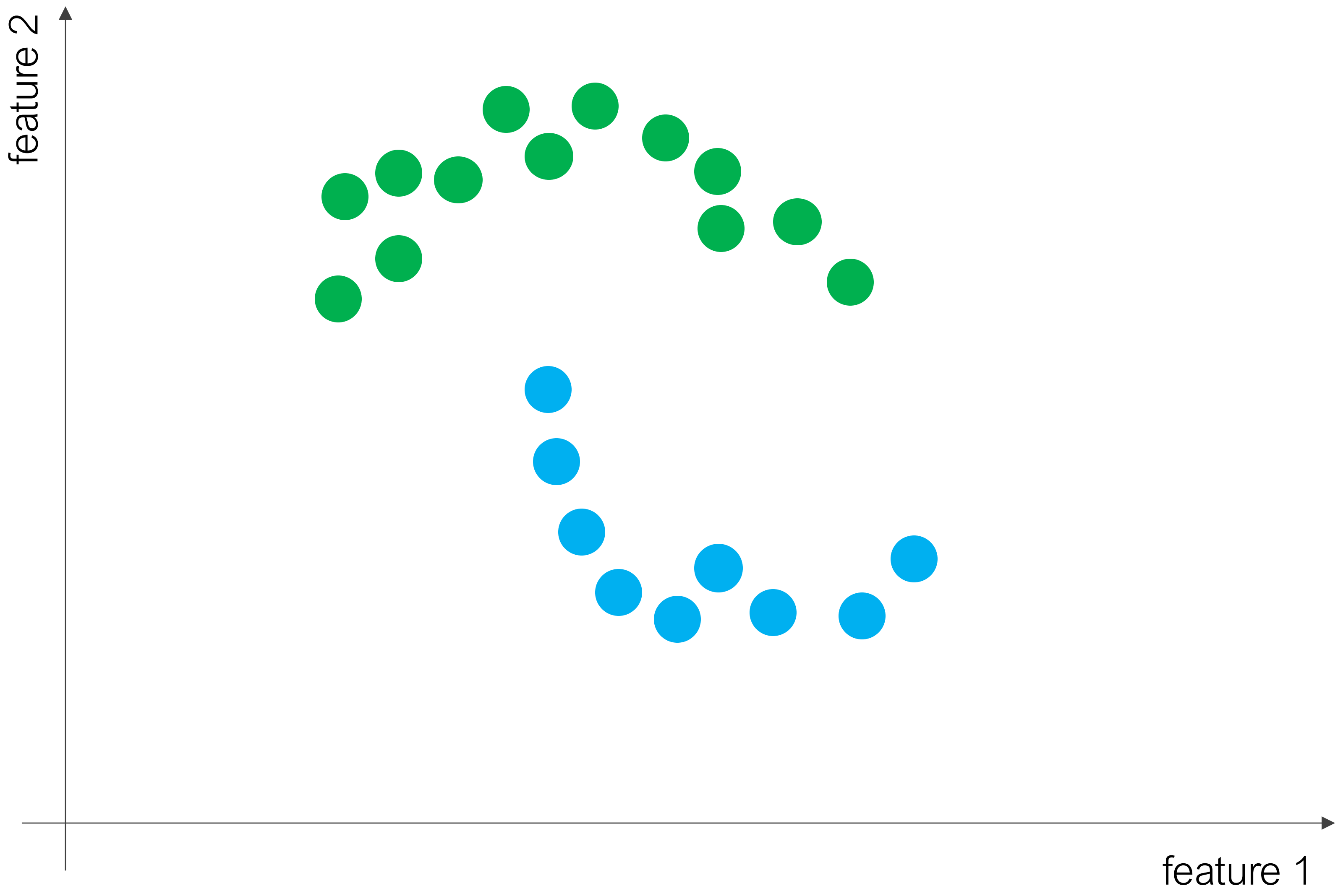


Get the eigenvectors of the Laplacian matrix, cluster points based on the eigenvectors (typically using k-means)

Spectral Clustering

Algorithm

1. Construct a graph representation of your data
2. Perform clustering based on the eigenvalues of the Laplacian matrix (often with K-means)



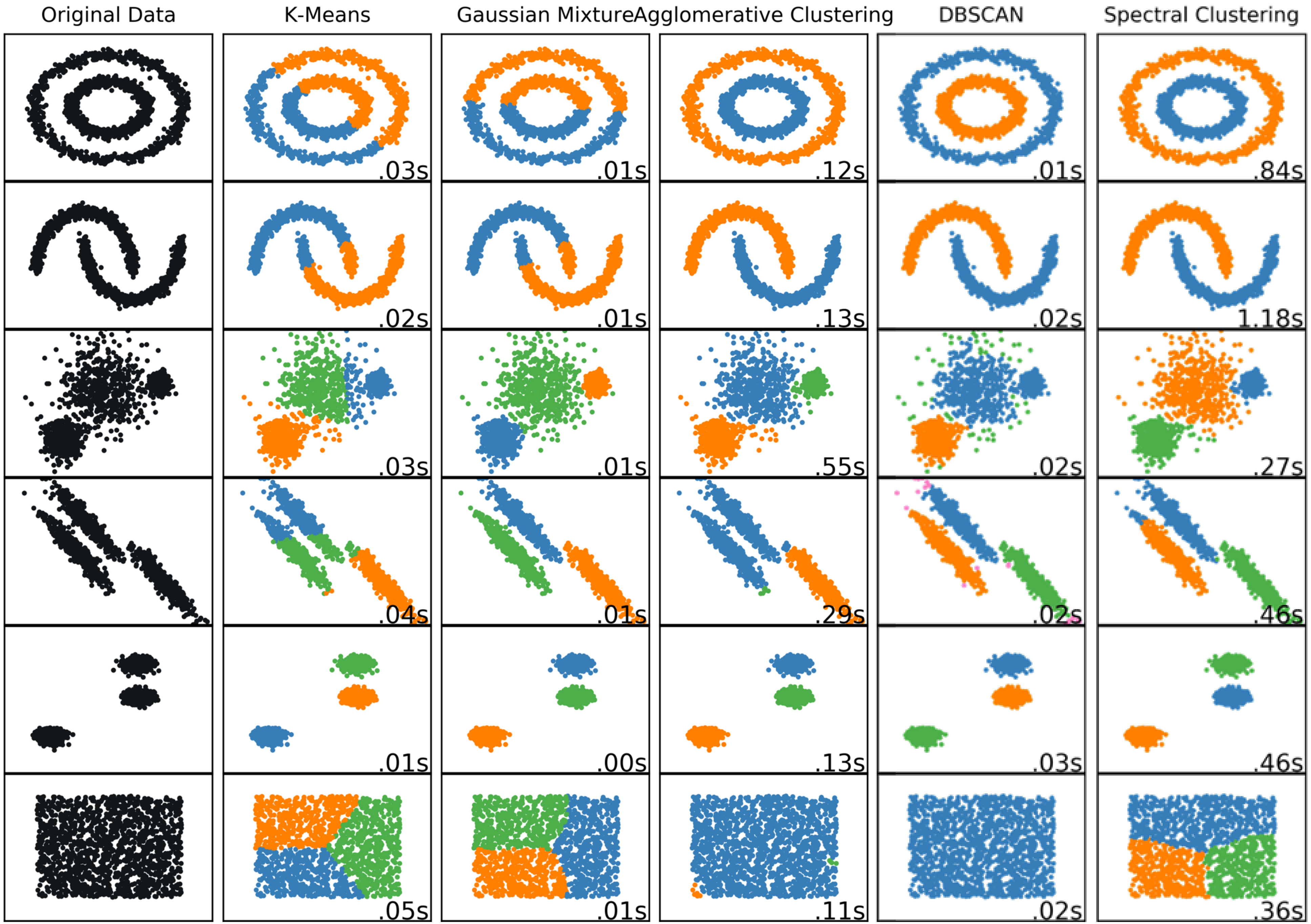
Concept from Sebastian Thrun and Peter Norvig

Examples: Spectral Clustering

Makes few assumptions about data, so often produces good clustering results

Slow for large datasets

Requires specifying number of clusters



Types of clustering algorithms

Methods

Distribution-based clustering (e.g. **Gaussian mixture model**)
Centroid-based clustering (e.g. **K-Means**)
Density-based clustering (e.g. **DBSCAN**)
Hierarchical clustering (e.g. **agglomerative clustering**)
Graph-based clustering (e.g. **spectral clustering**)

Cluster assignment

Hard clustering
Soft clustering (a.k.a. fuzzy clustering)

Clustering choices:

1. How should the data be scaled?

2. How many clusters to estimate?

3. How do we measure dissimilarity?

4. How do we evaluate "fit" of the clusters?

Image from James et al., Introduction to Statistical Learning, 2013