Clustering II

Lecture 15

Types of clustering algorithms

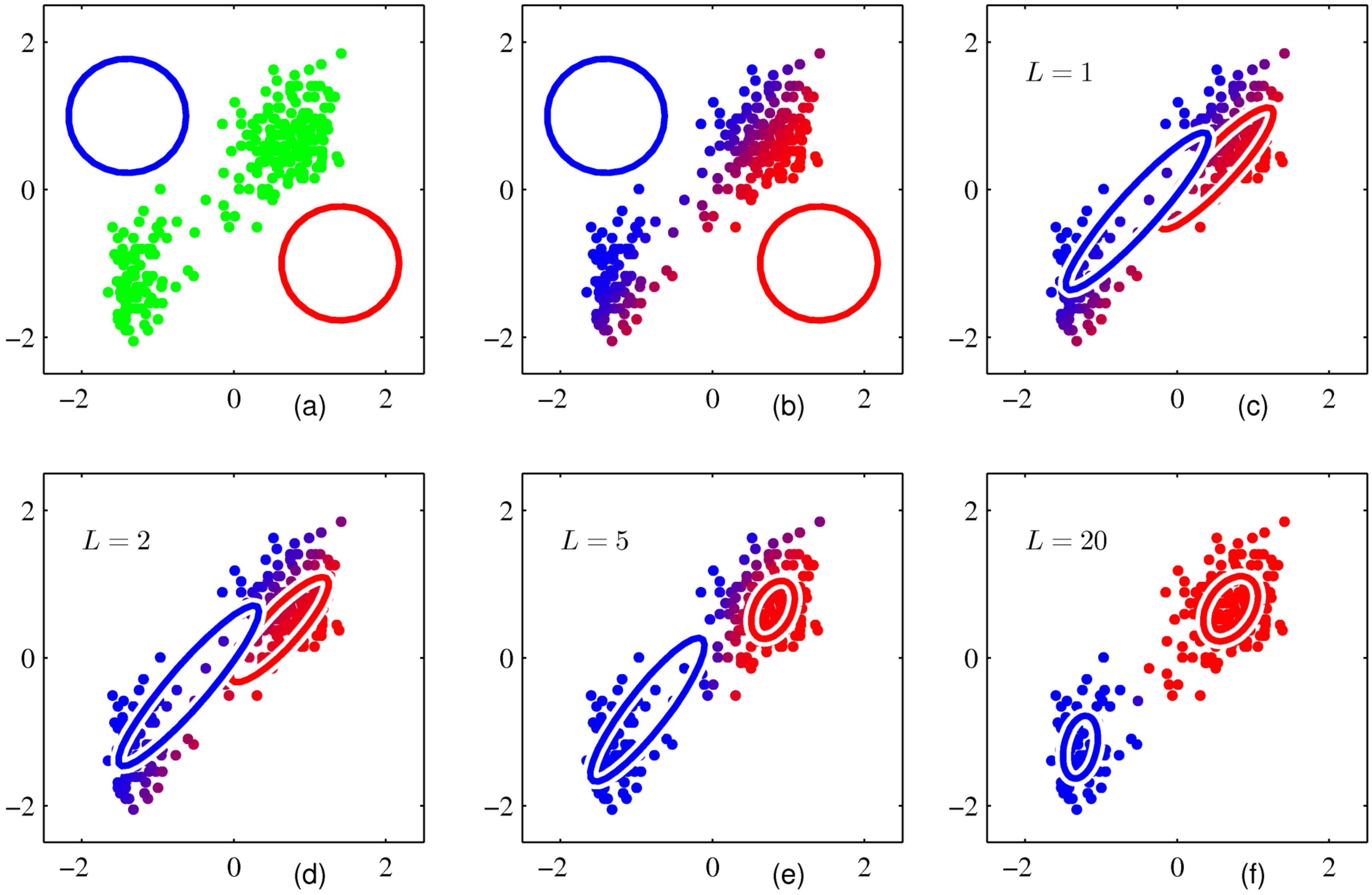
Methods

Centroid-based clustering (e.g. **K-Means**)
Distribution-based clustering (e.g. **Gaussian mixture model**)
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)

Expectation Maximization for GMM Example



L = number of EM cycles

Image from Bishop, Pattern Recognition, 2006

Expectation Maximization for a GMM

Goal: maximize the log likelihood of the data given the model parameters:

$$\ln P(\boldsymbol{X}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{i=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k N(\boldsymbol{x}_i|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \right]$$

0. Initialization

Initialize all the parameters
(often K-means is used for this purpose)

1. Expectation-step

Calculate the "responsibilities" based on the model parameters

$$\gamma(z_{ik}) \triangleq P(z_k = 1|x_i)$$

$$= \frac{\pi_k N(\boldsymbol{x}_i|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k N(\boldsymbol{x}_i|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}$$

2. Maximization-step

Use the "responsibilities" to update the model parameters to maximize the log likelihood

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) \boldsymbol{x}_i$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{new})^T$$

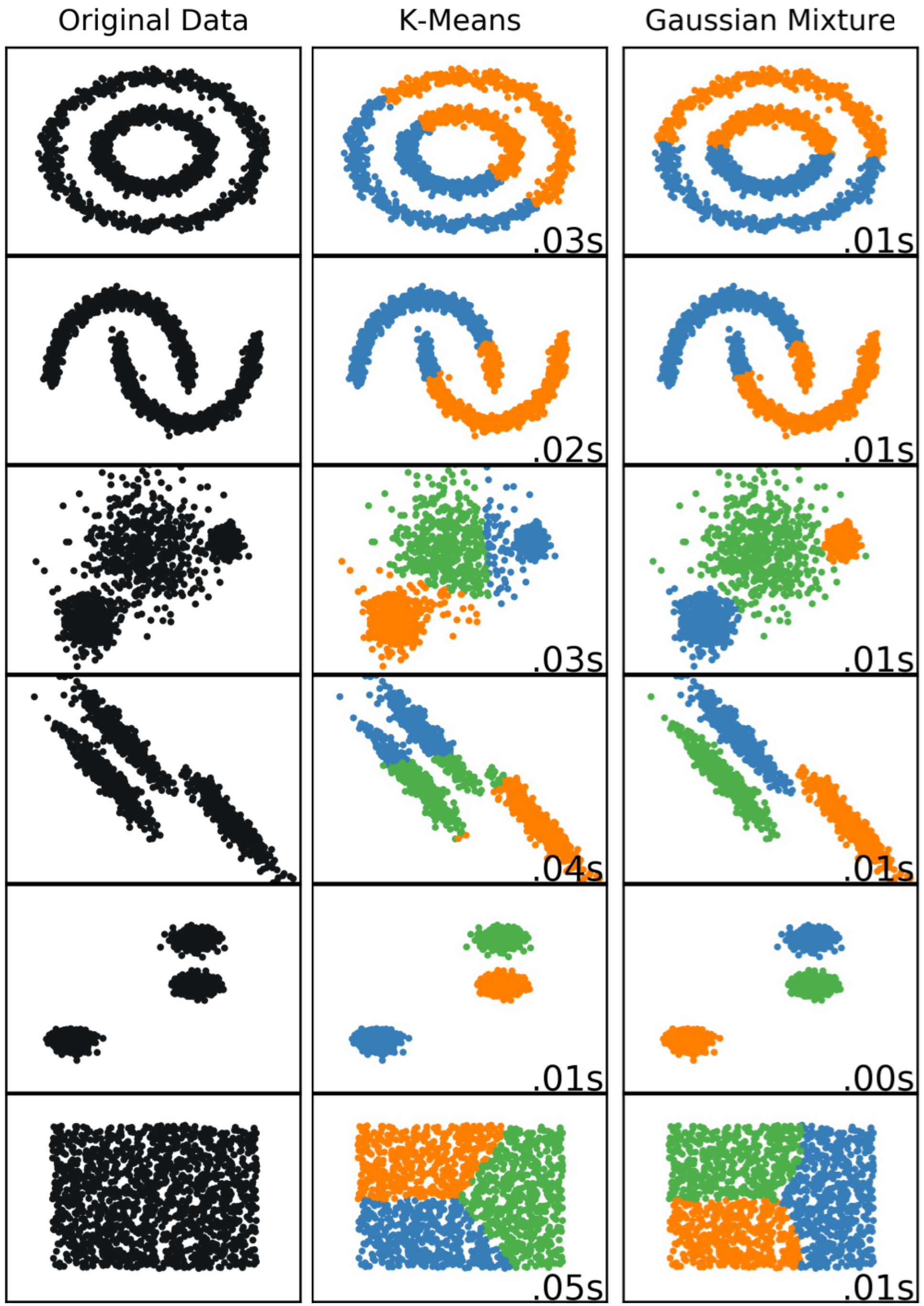
$$\pi_k^{new} = \frac{N_k}{N}$$

Where $N_k = \sum_{i=1}^{N} \gamma(z_{ik})$

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

Gaussian Mixture Models

Generative models: model $P(\boldsymbol{X}|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the model parameters

Very useful for density estimation

Produce hard or soft (fuzzy) clustering

When you restrict the covariance matrix to be diagonal and equal for all clusters, the GMM and K-means algorithm become the same

Expectation Maximization

Iterative method to find maximum likelihood parameter estimates when the model depends on unobserved latent variables, when this can’t be solved directly

The E-step updates the latent variable distribution estimates, so that we can calculate the likelihood function given the current parameter values

The M-step identifies the parameters that maximize the likelihood

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

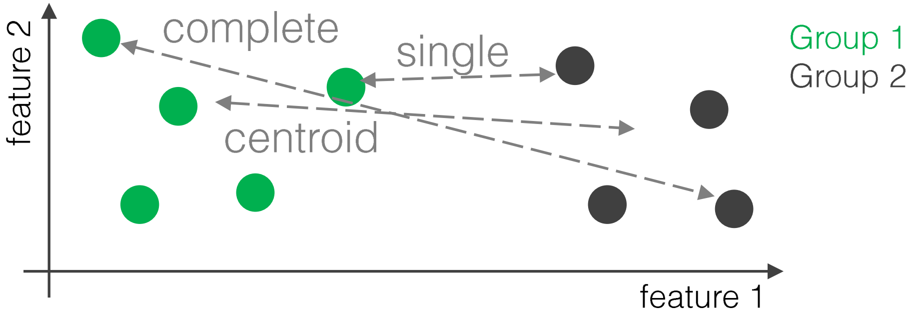
How to measure distance/dissimilarity between groups or sets

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

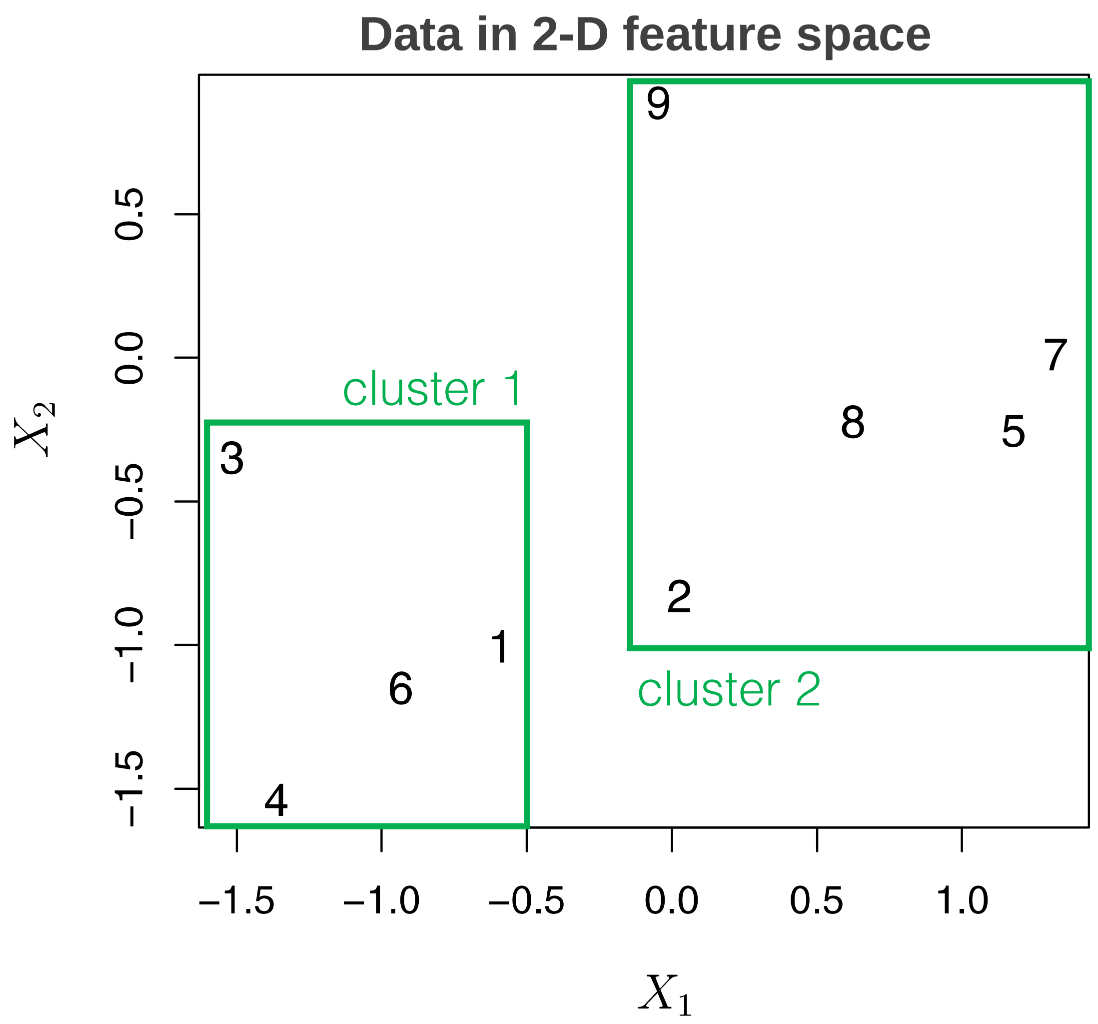


Agglomerative clustering

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



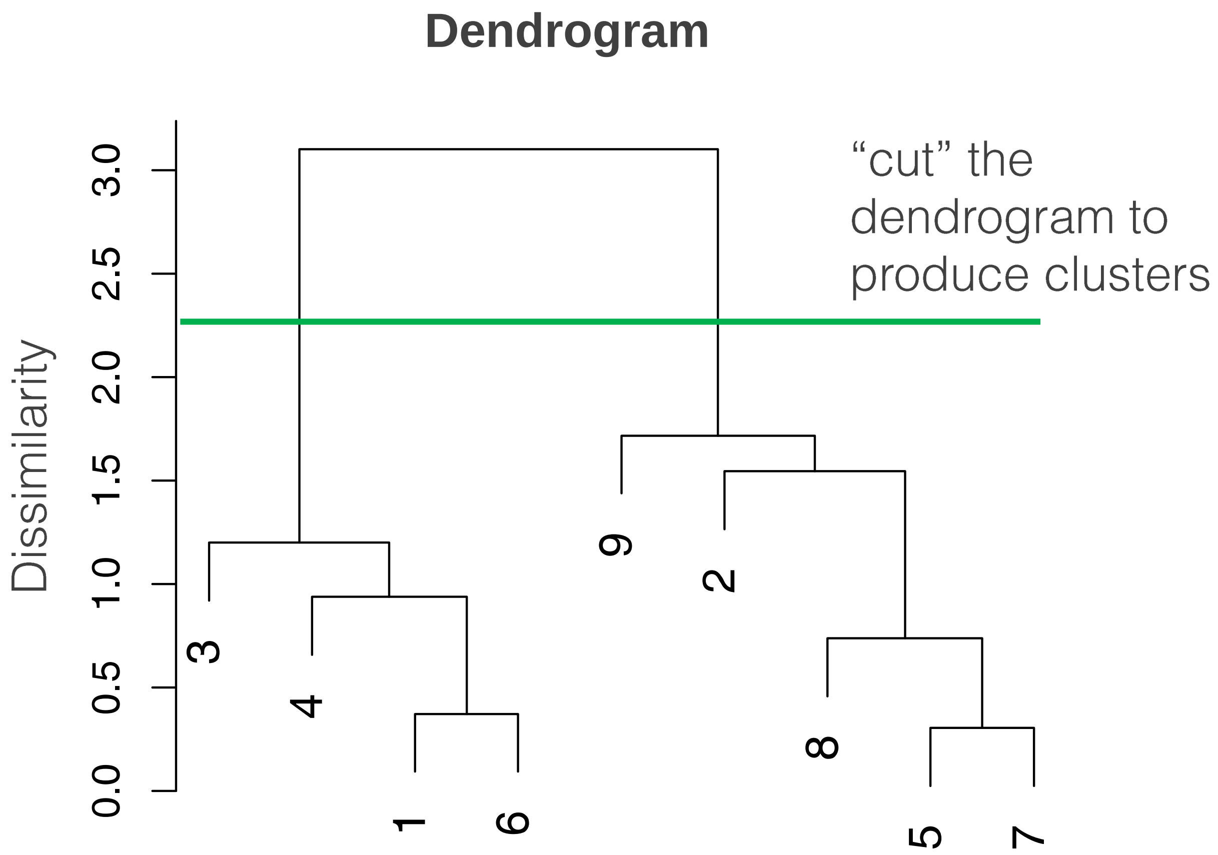
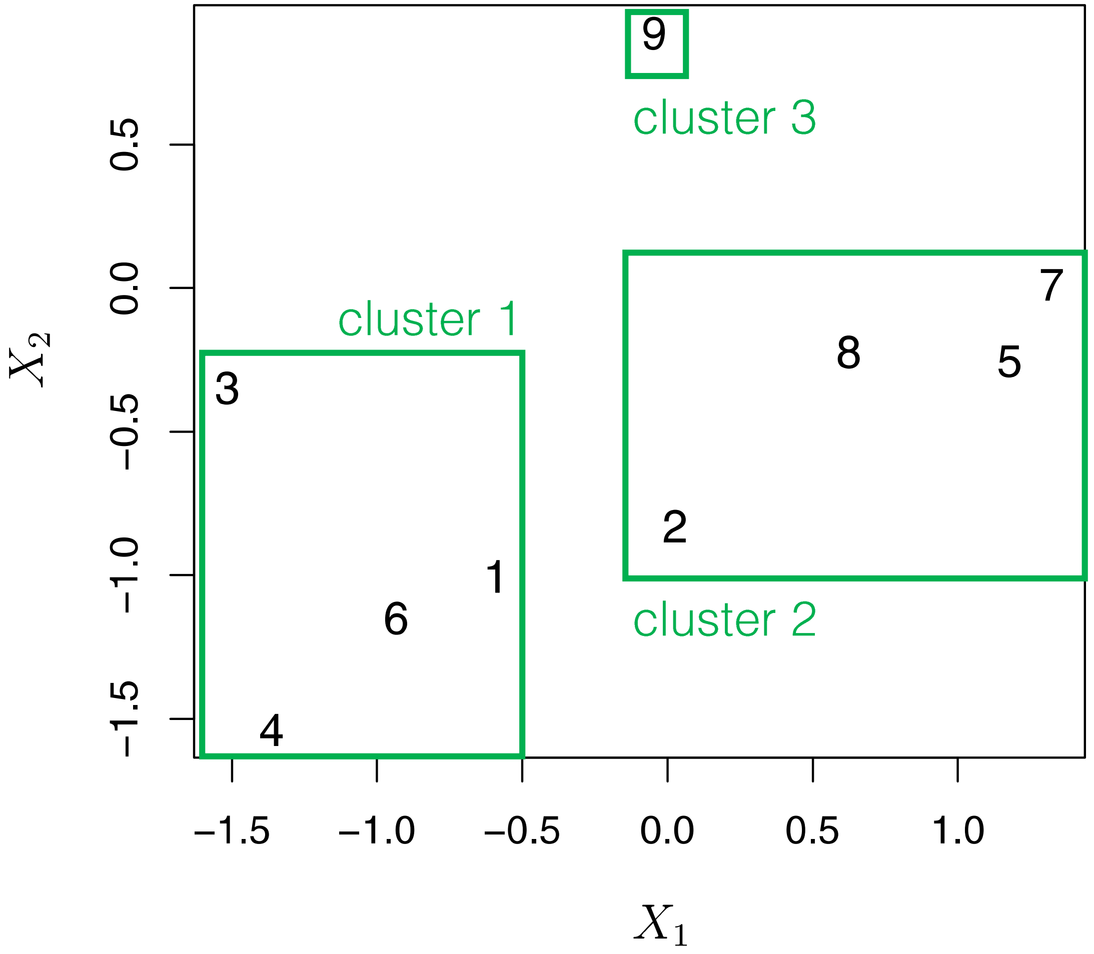


Image from James et al., Introduction to Statistical Learning, 2013

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



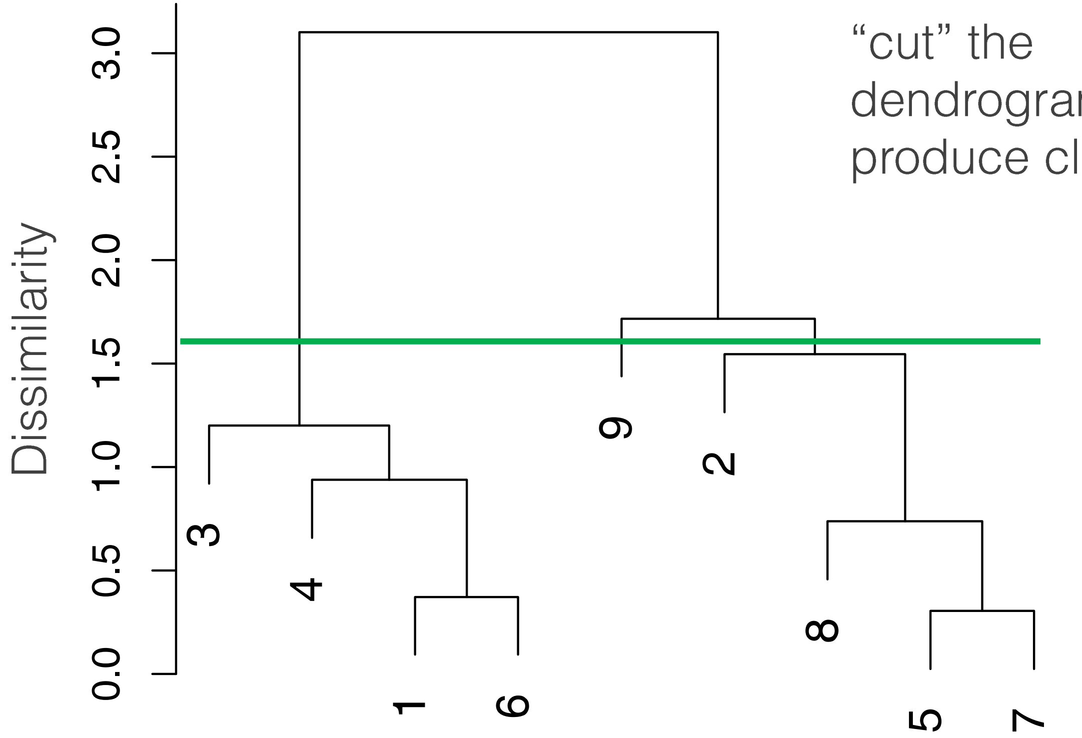
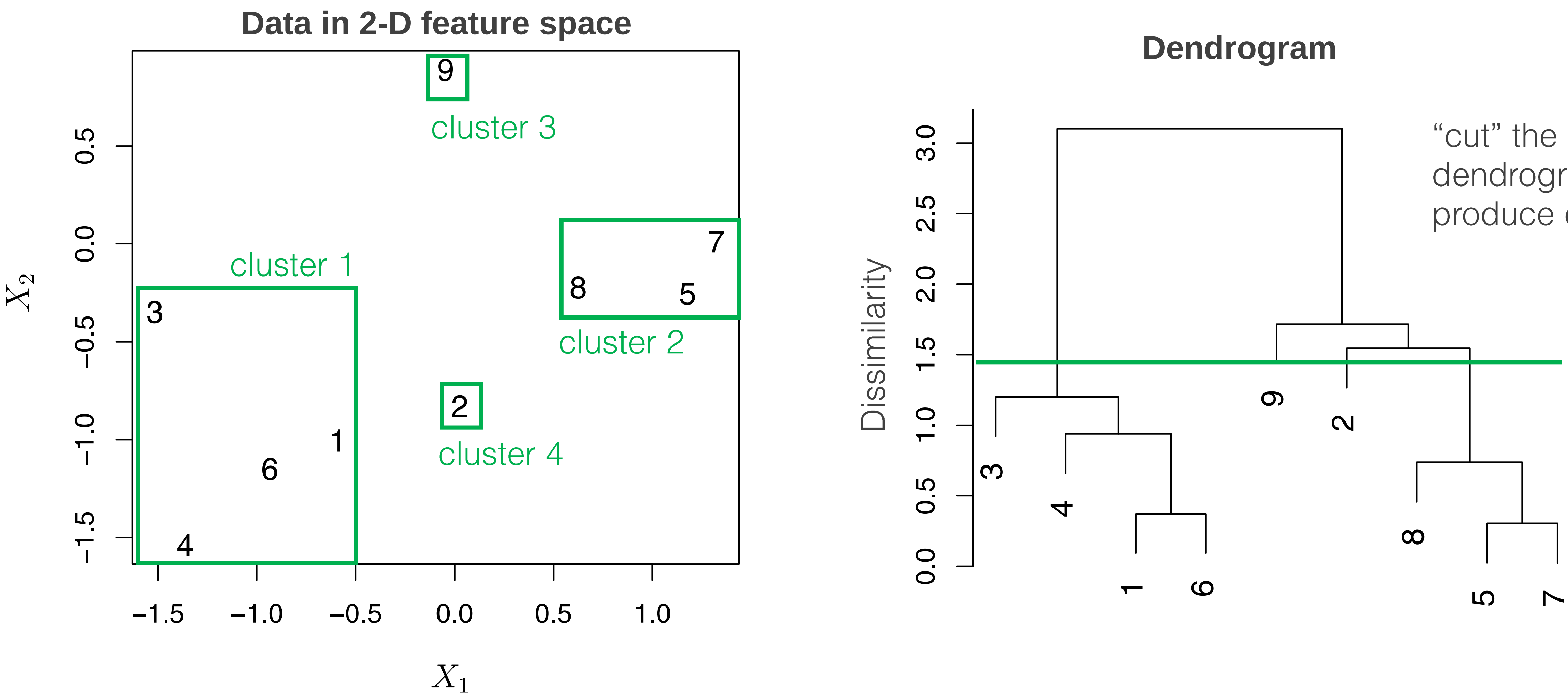


Image from James et al., Introduction to Statistical Learning, 2013

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

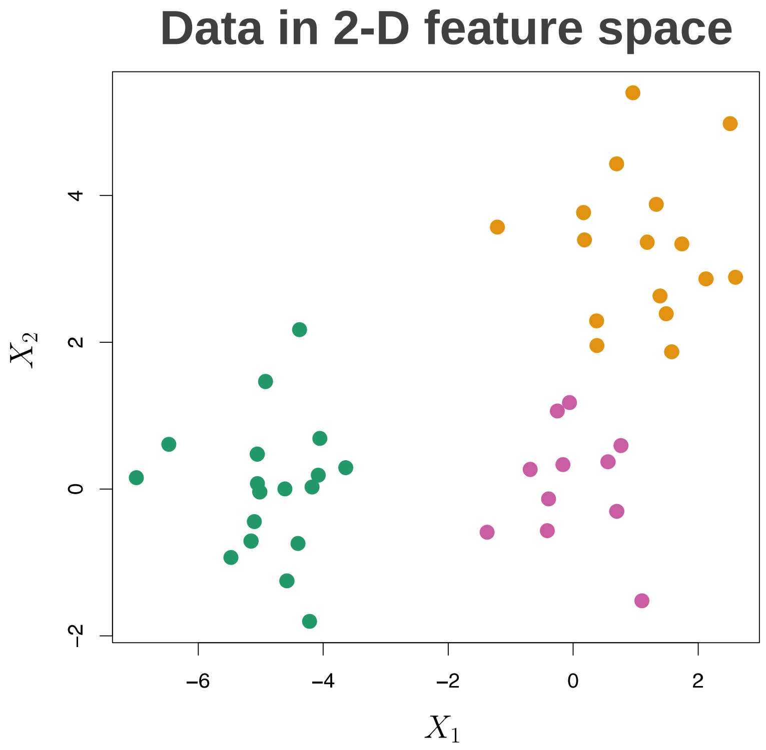


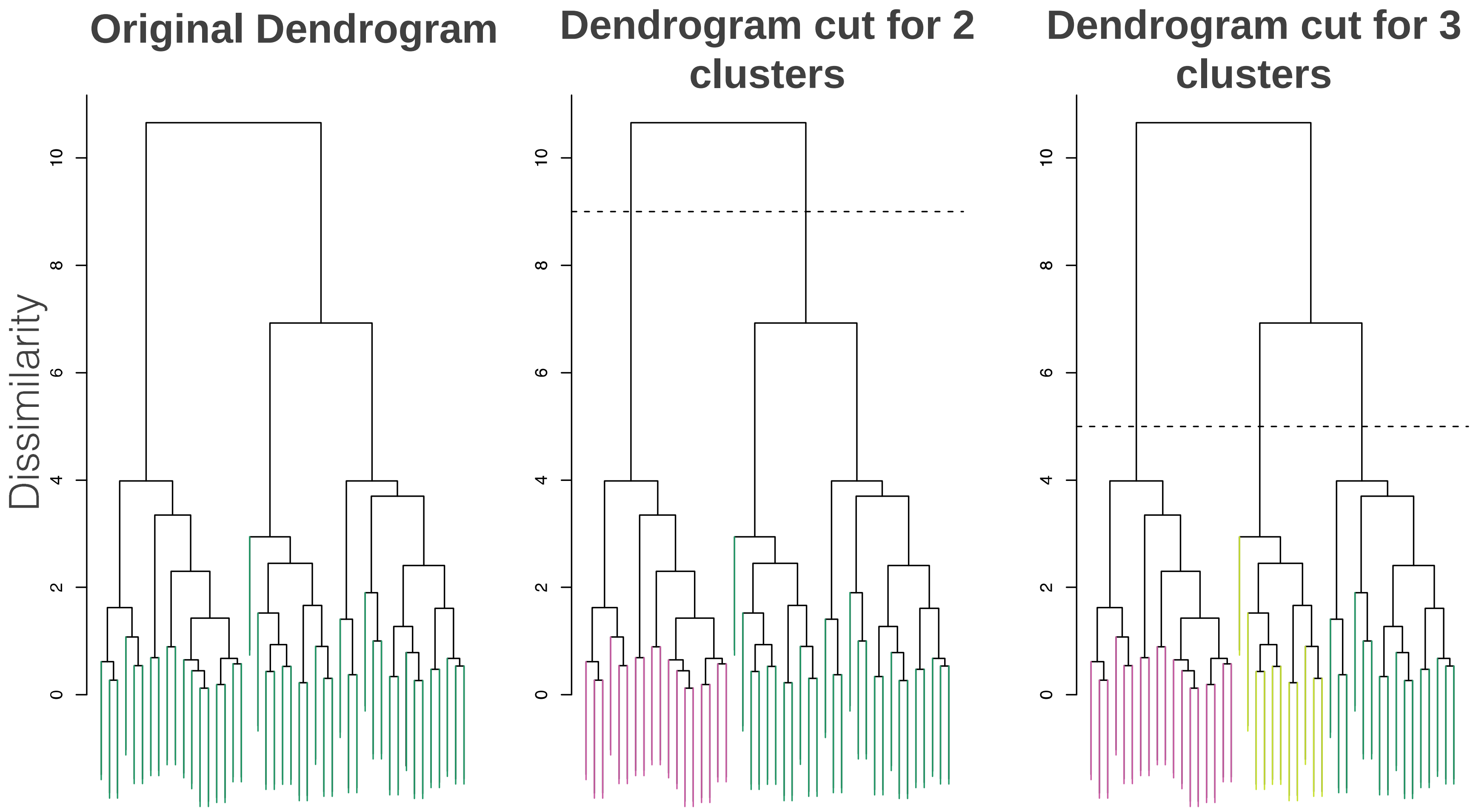
“cut” the dendrogram to produce clusters

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

Example of agglomerative clustering

With complete linkage and Euclidean distance





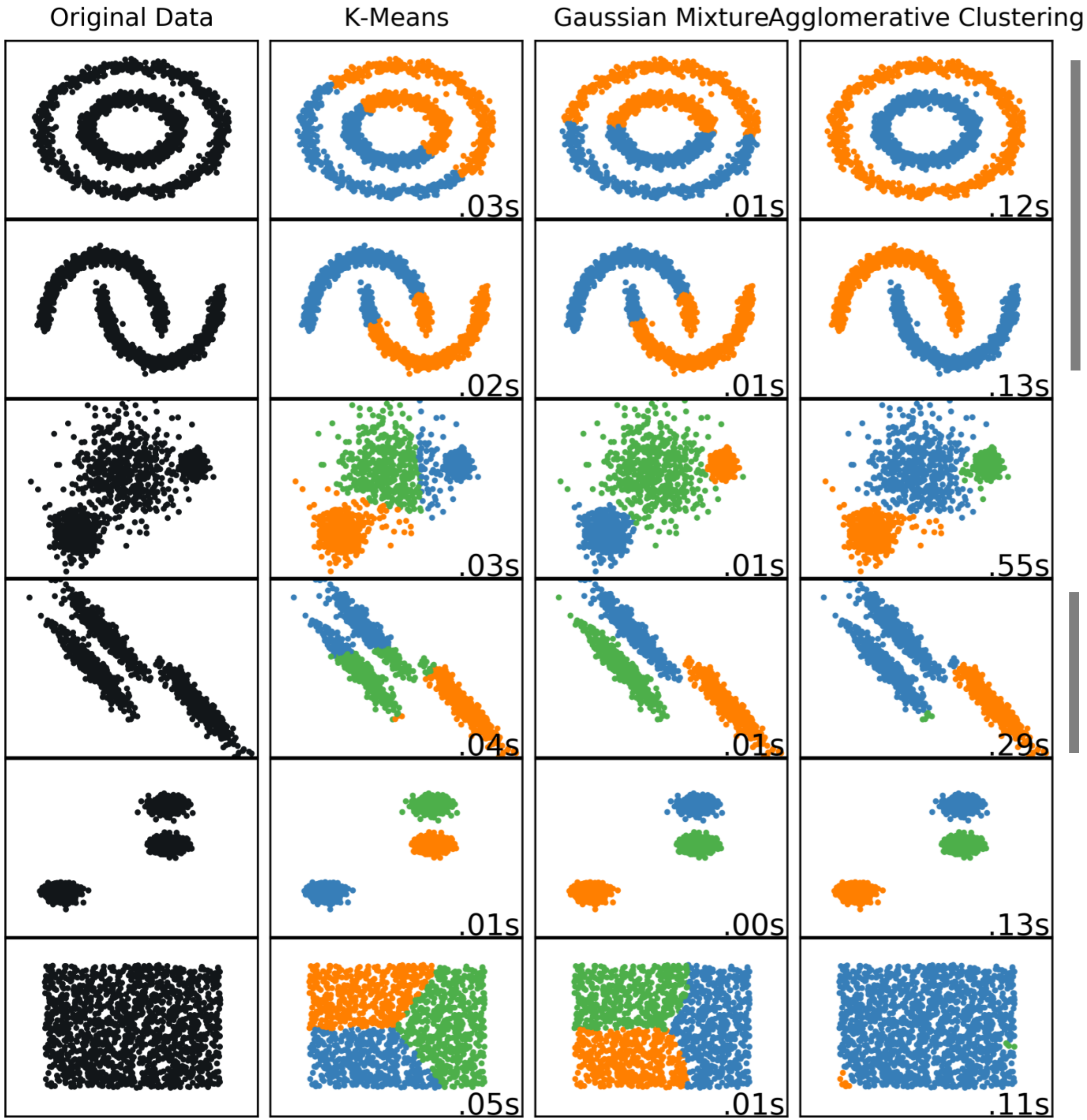
Note: colors do not directly map to plot on the left

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

Examples: Agglomerative clustering

Need to choose where to cut the dendrogram

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



Performs well when clusters are well-separated

Struggles when intercluster distance is not sufficient to distinguish between clusters

DBSCAN Clustering

Density-based spatial clustering of applications with noise

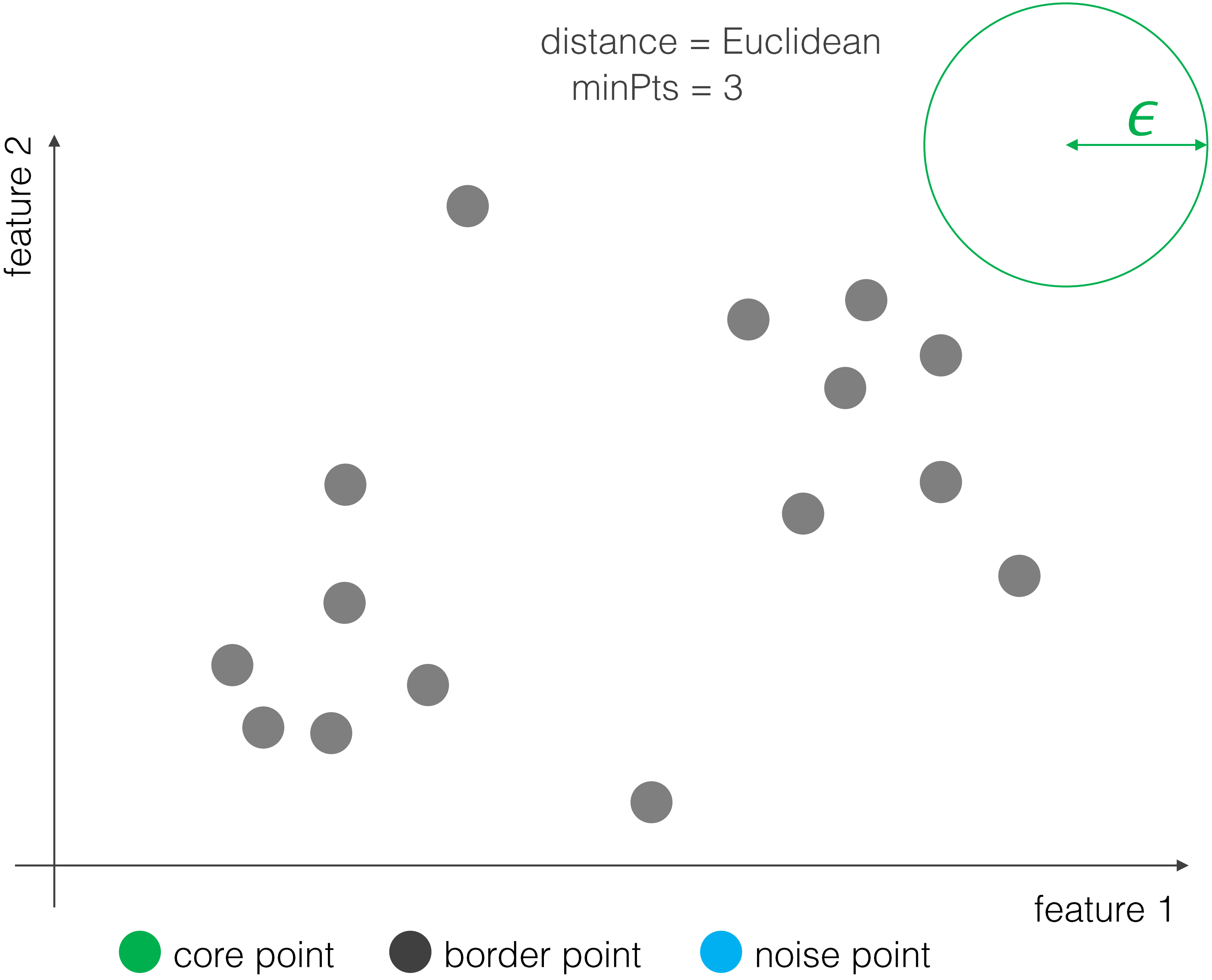
By Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, 1996

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

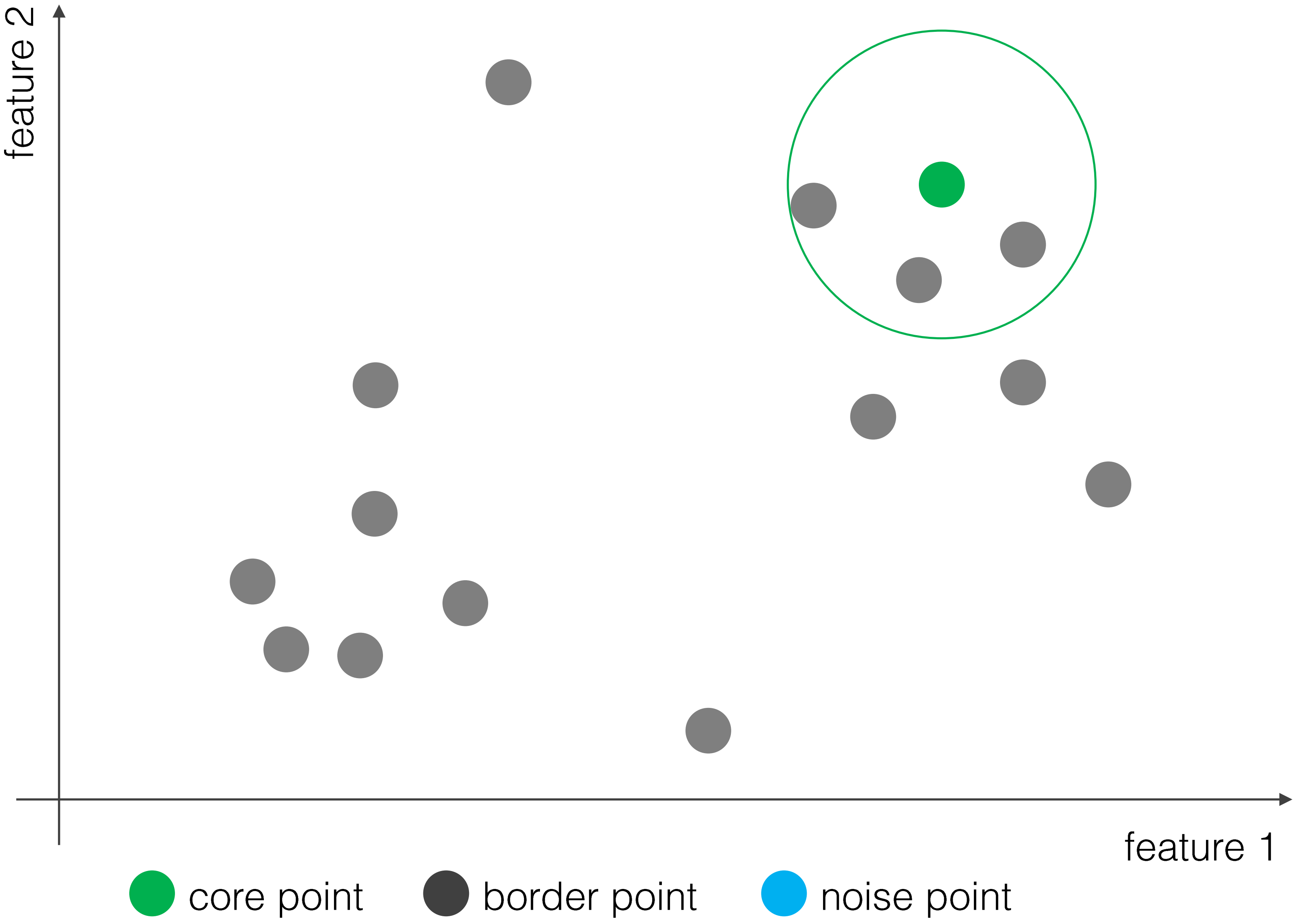


DBSCAN

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

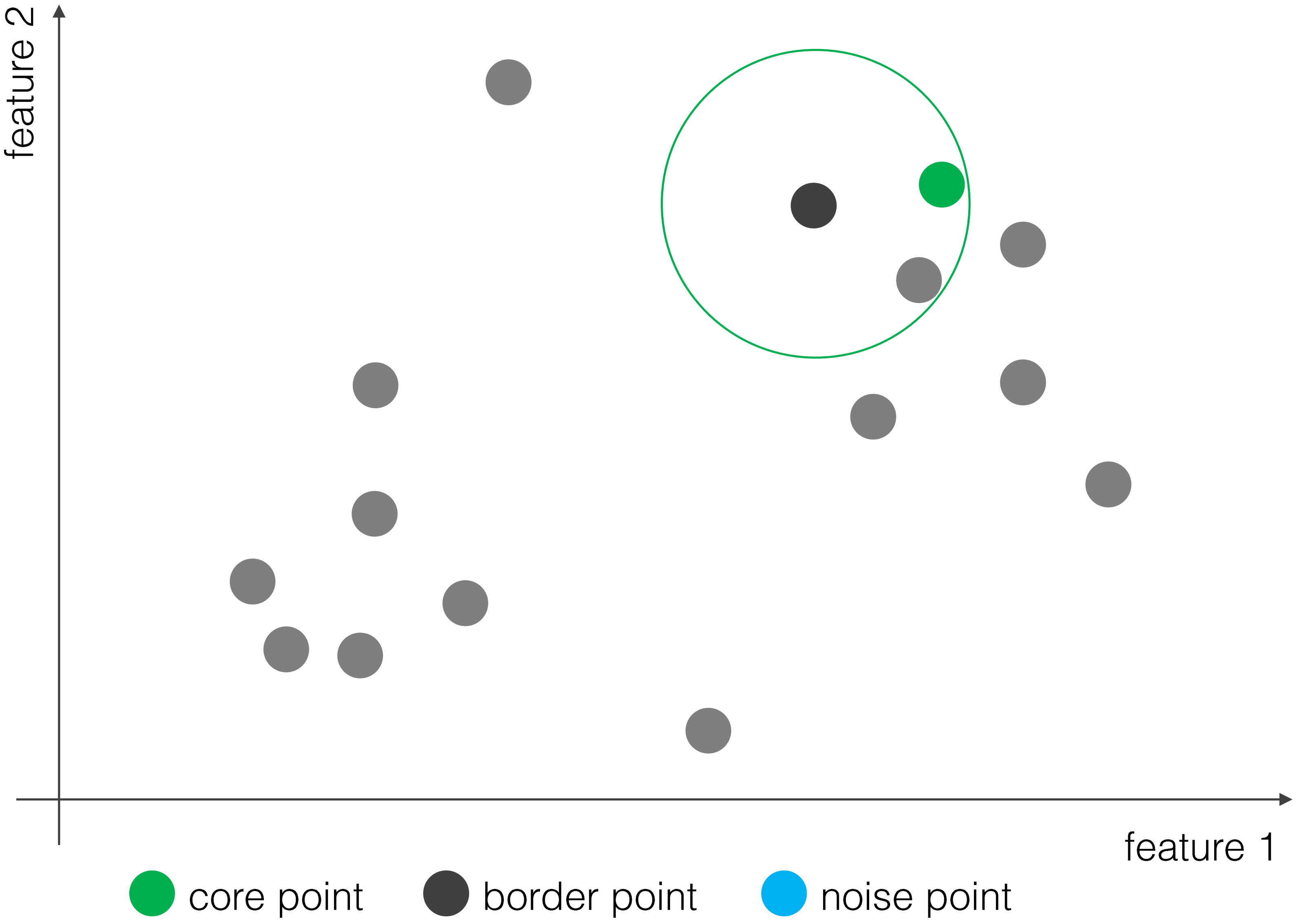


DBSCAN

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

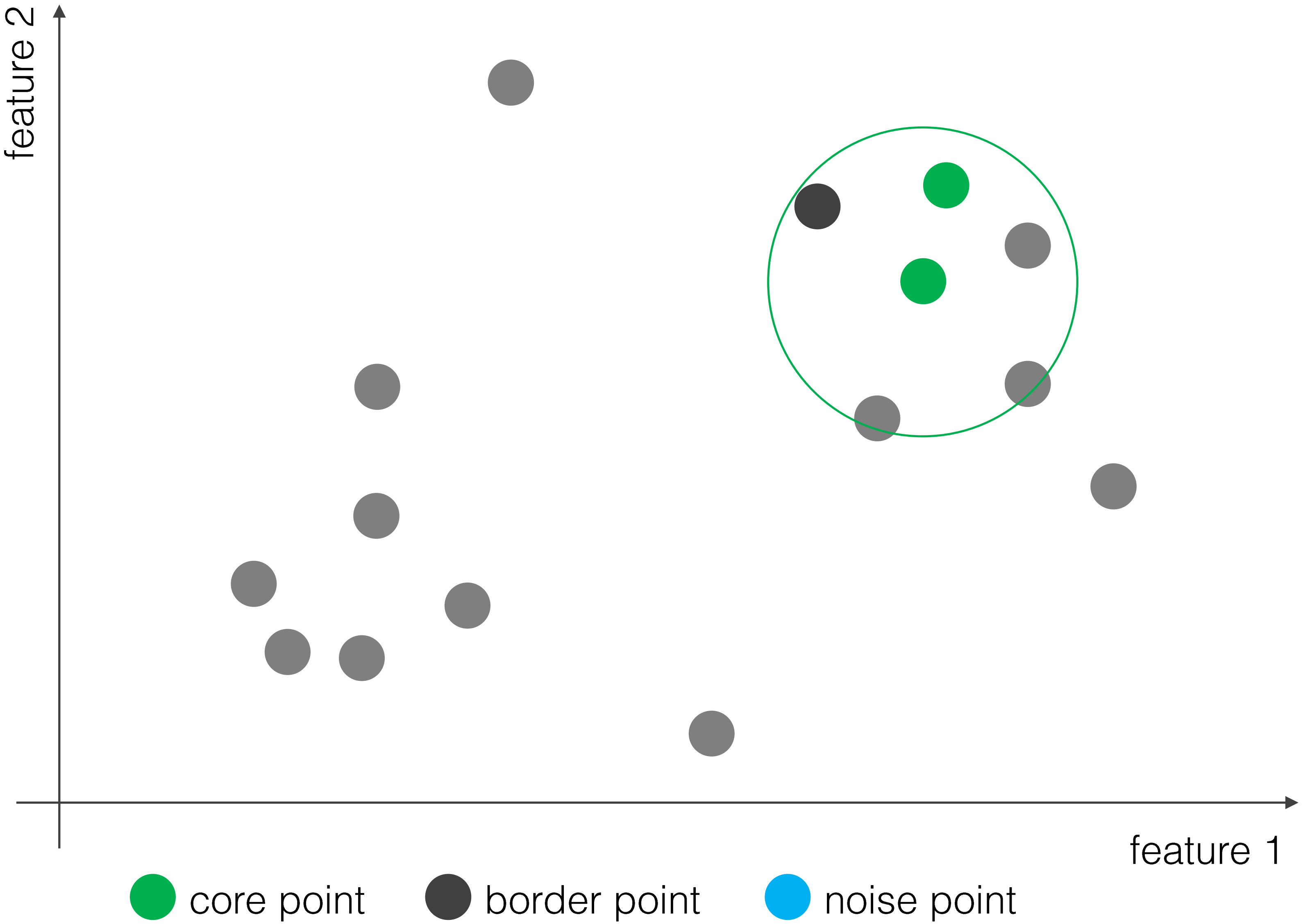


DBSCAN

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

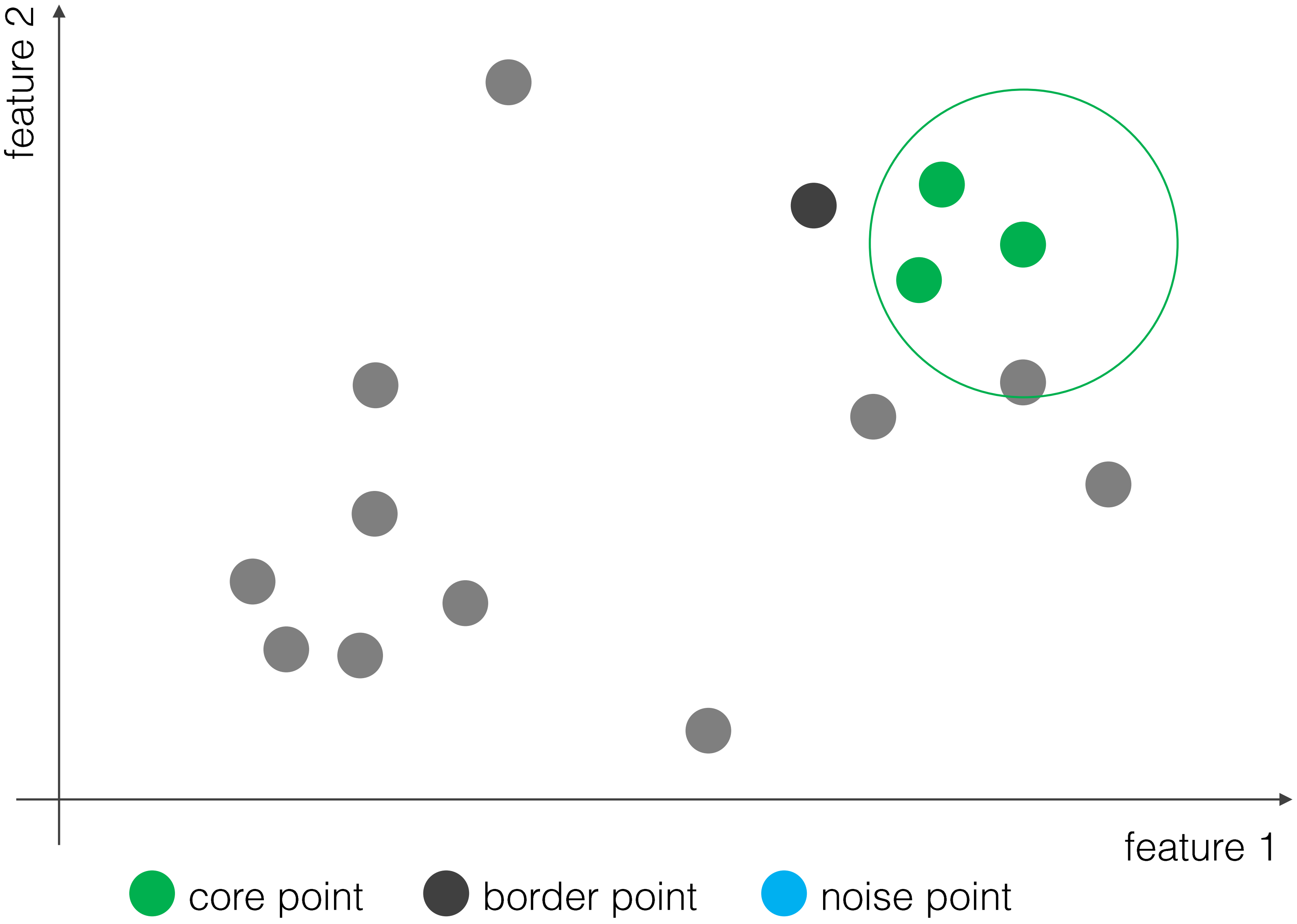


DBSCAN

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

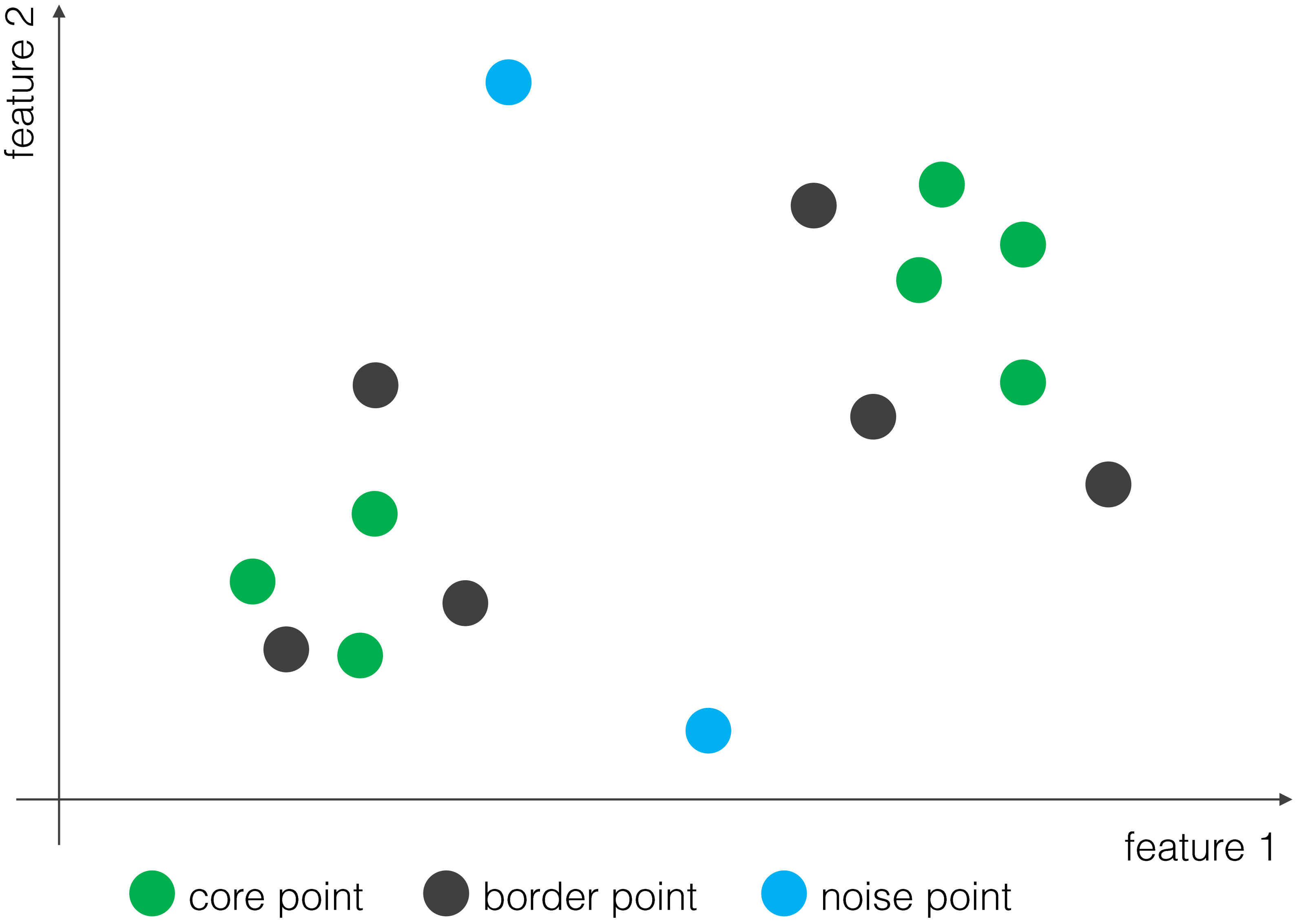


DBSCAN

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

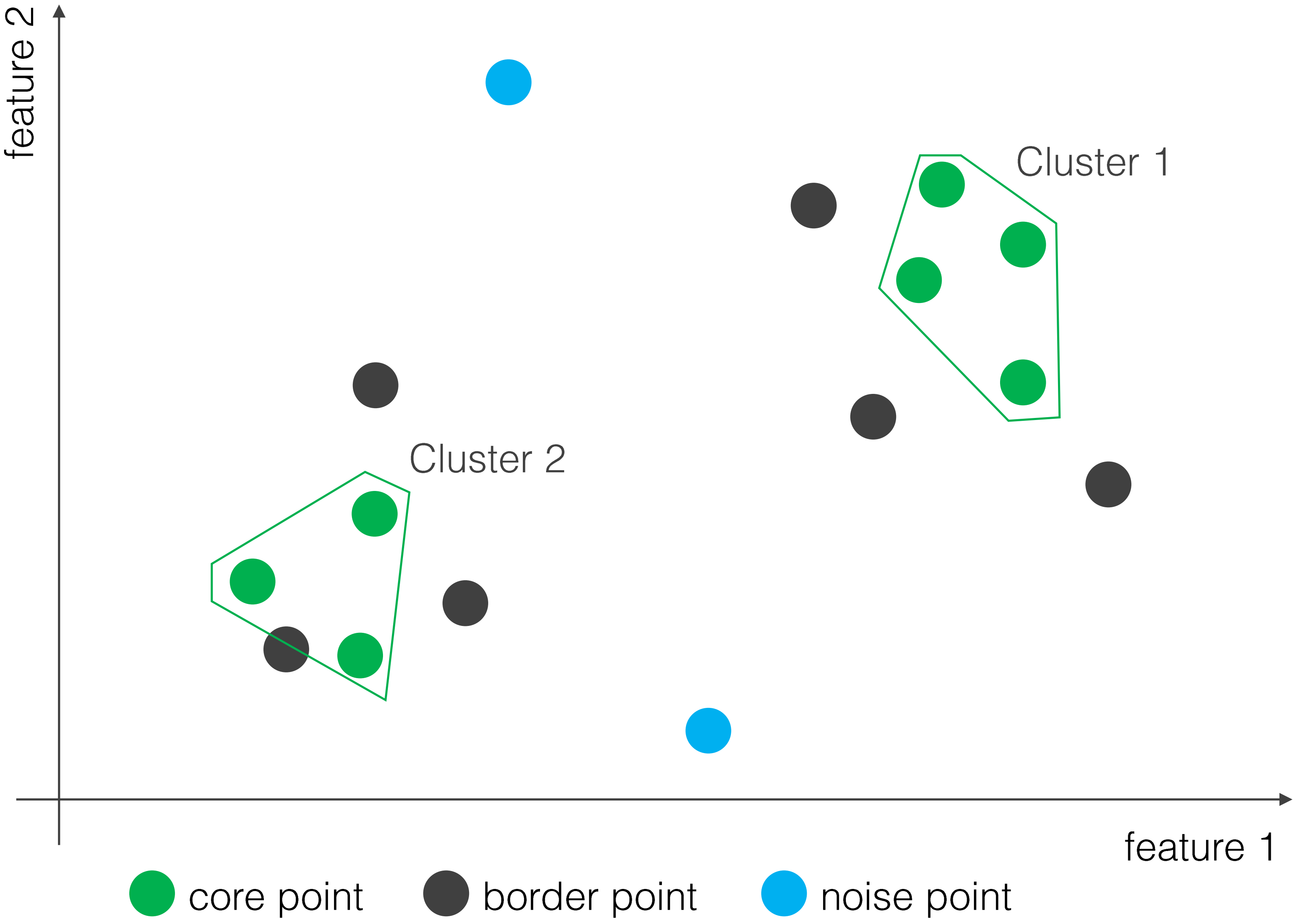


DBSCAN

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

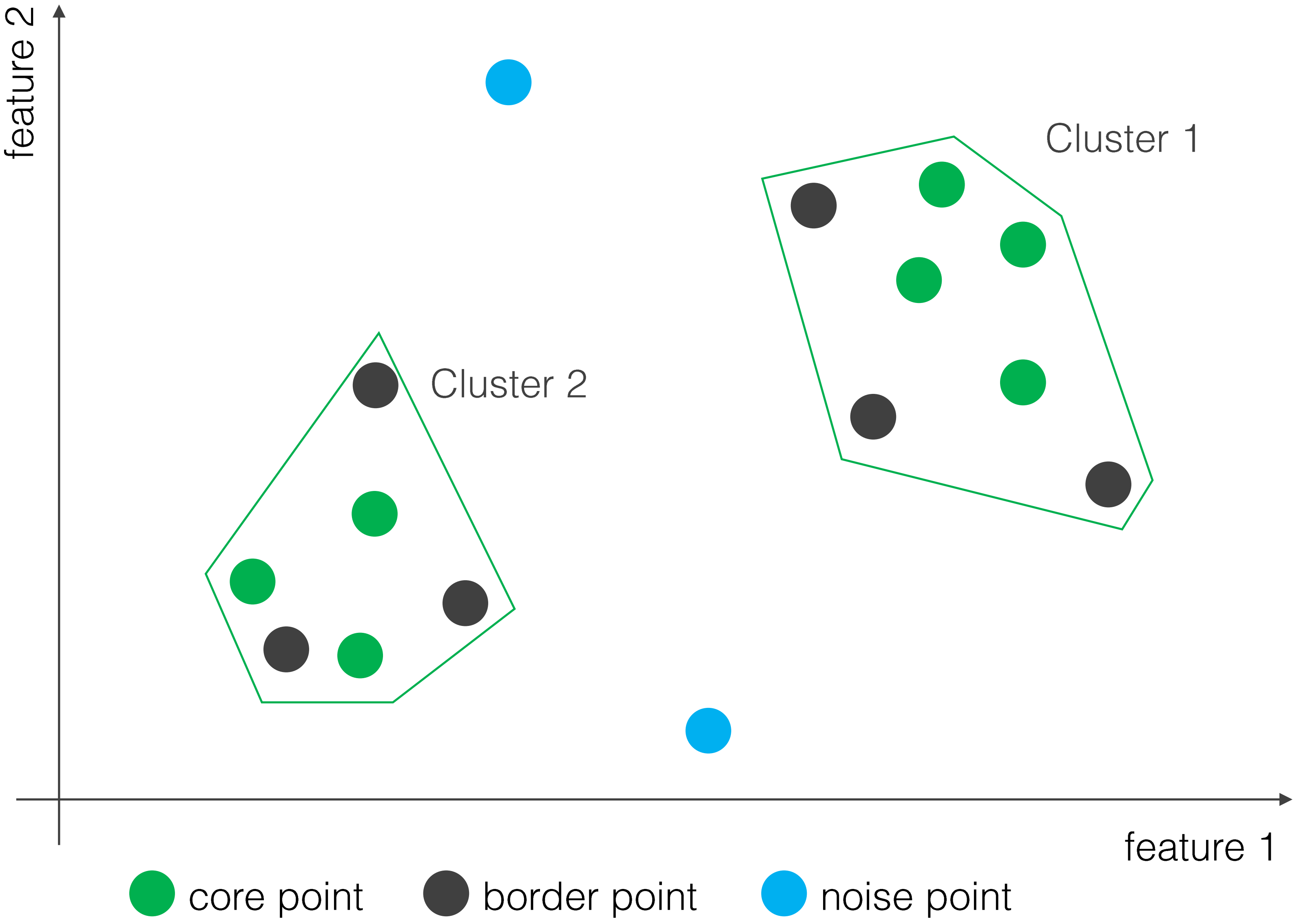


DBSCAN

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

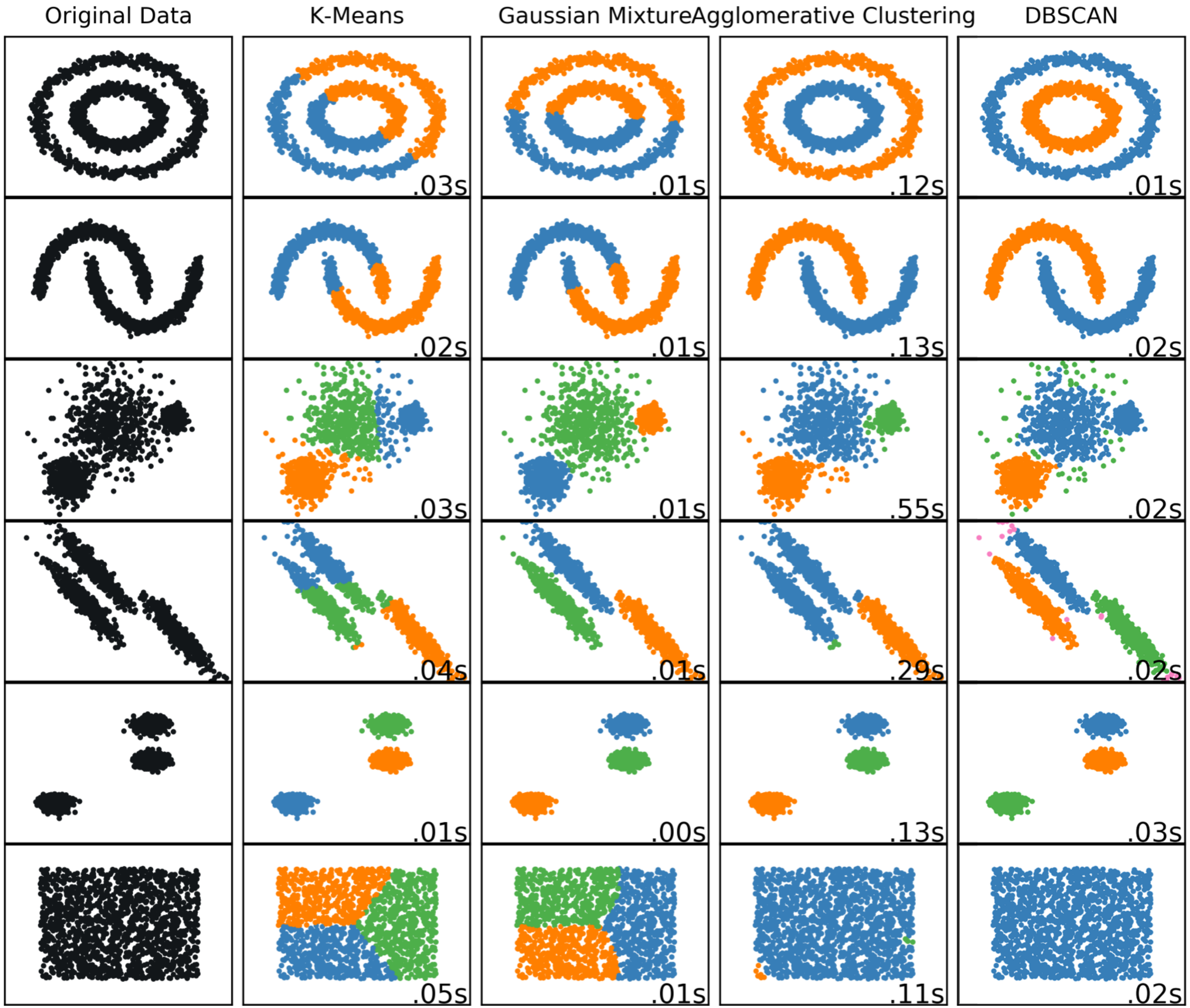


DBSCAN

- 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

Clustering in a low dimensional space based on data similarity

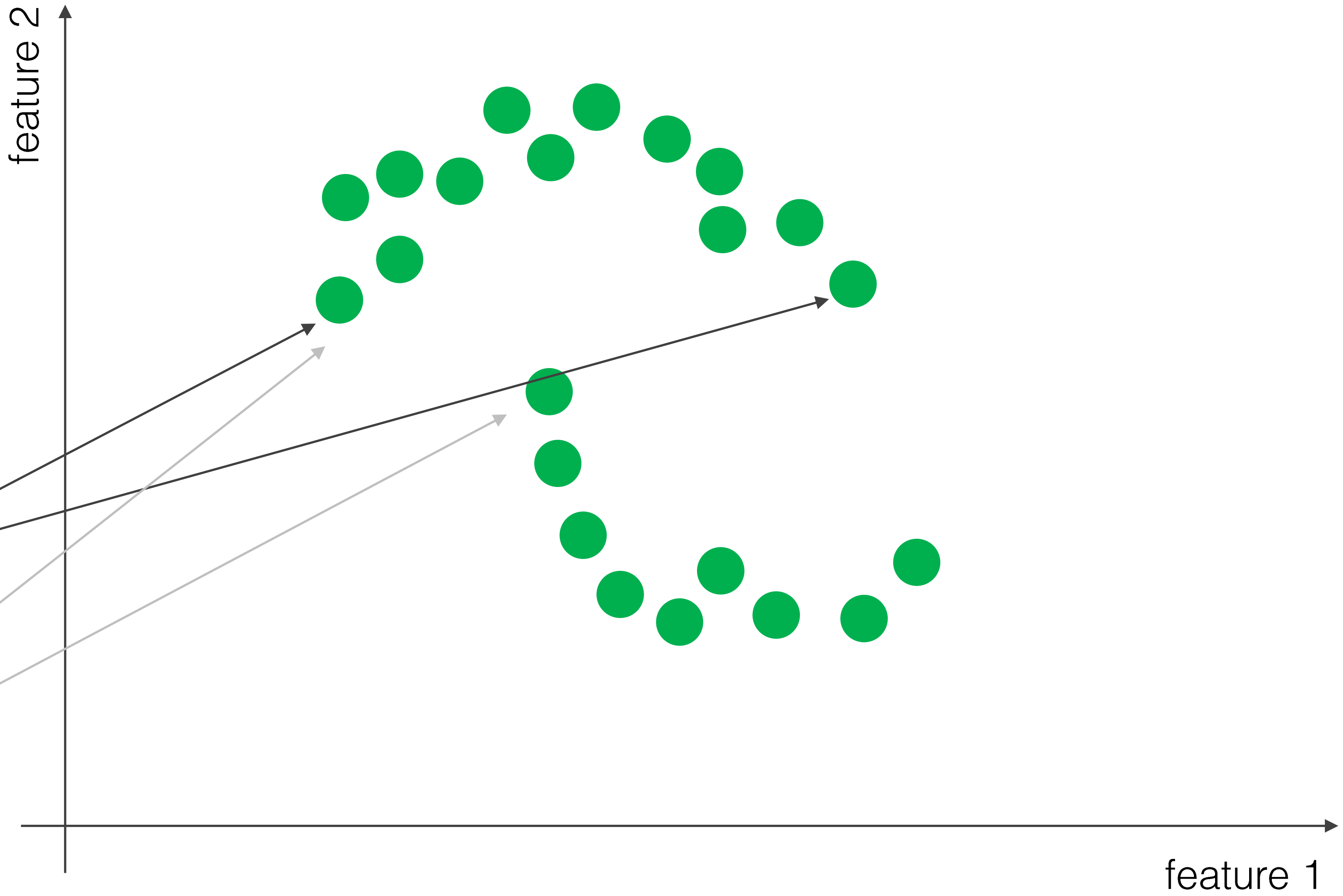
Spectral Clustering

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

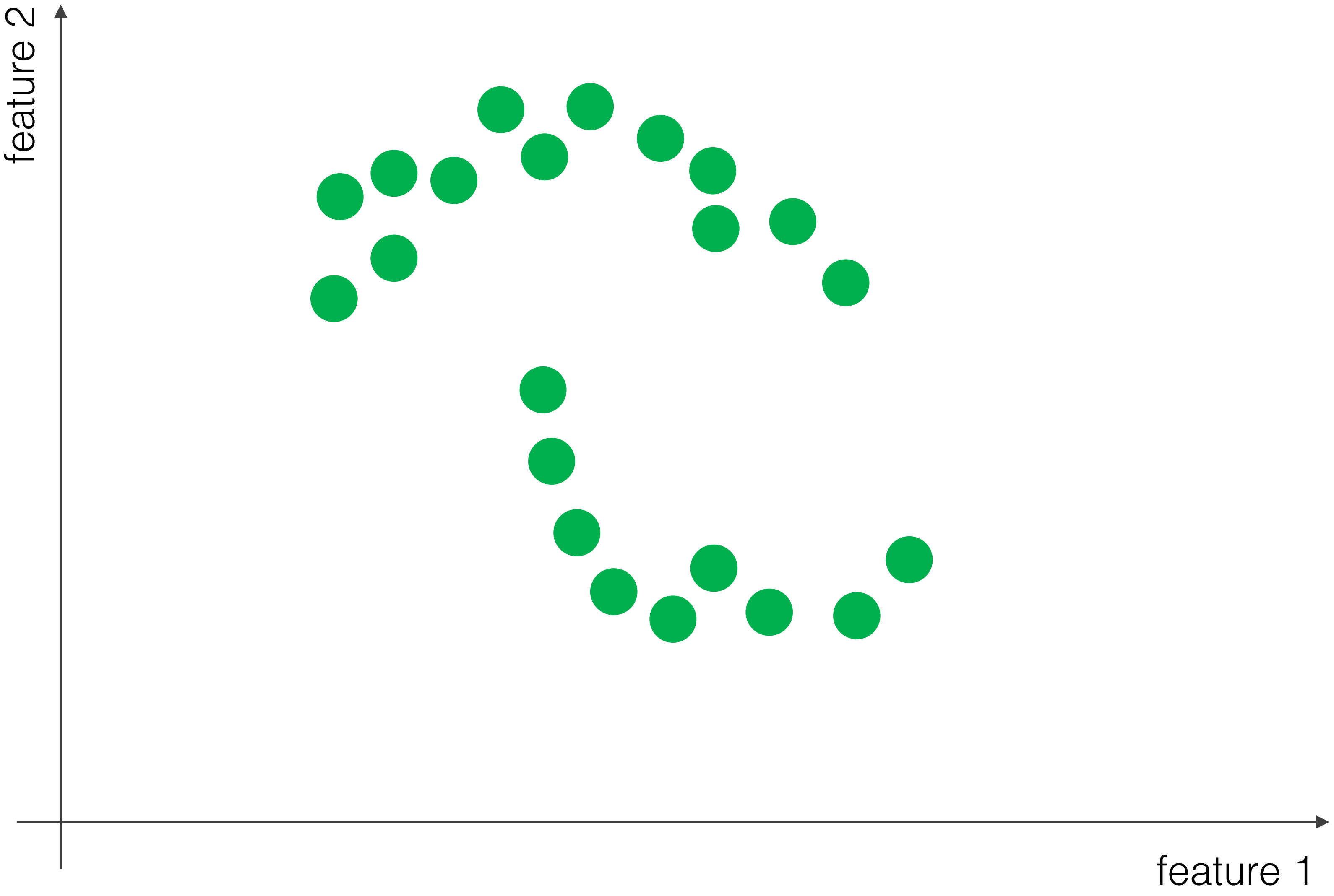
Spectral Clustering

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:

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

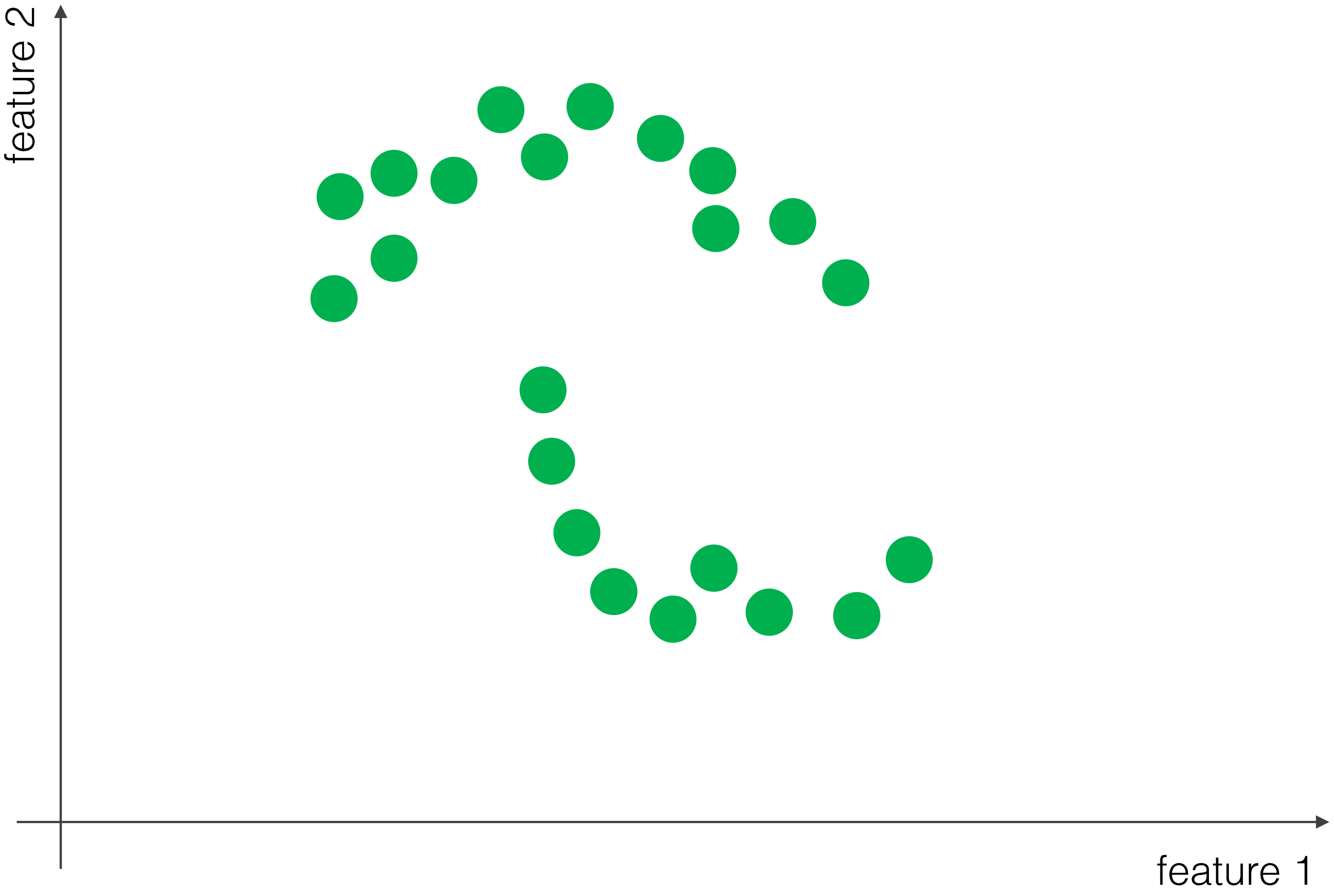


Concept from Sebastian Thrun and Peter Norvig

Spectral Clustering

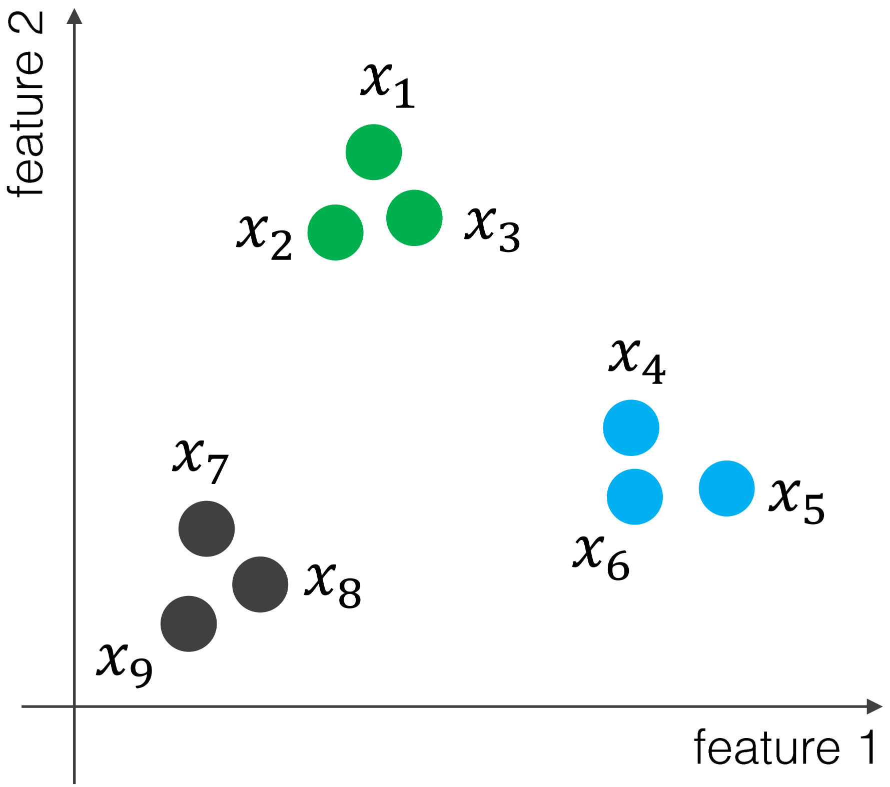
Algorithm

1. Construct an affinity matrix based on the data (works best when this matrix is sparse)
2. Get the principal components of the affinity matrix, reduce dimensions of the data
3. Perform clustering in this space

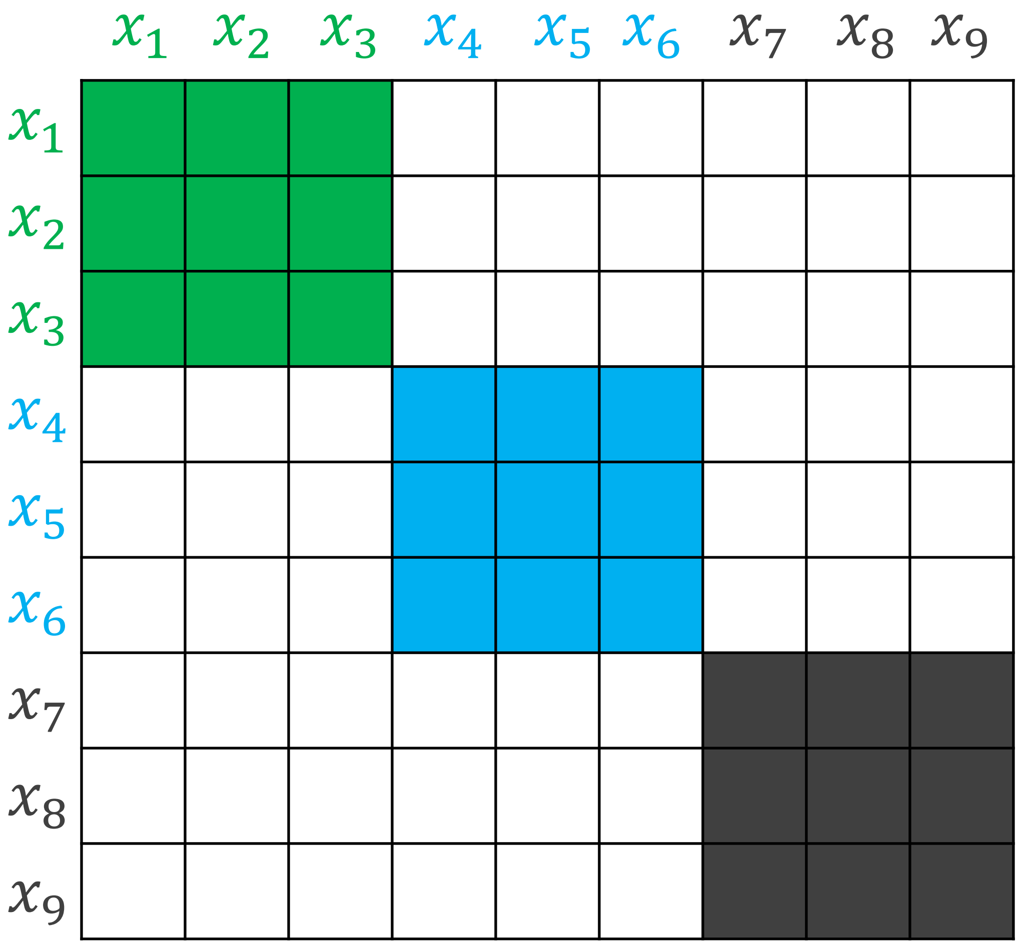


Concept from Sebastian Thrun and Peter Norvig

Spectral Clustering

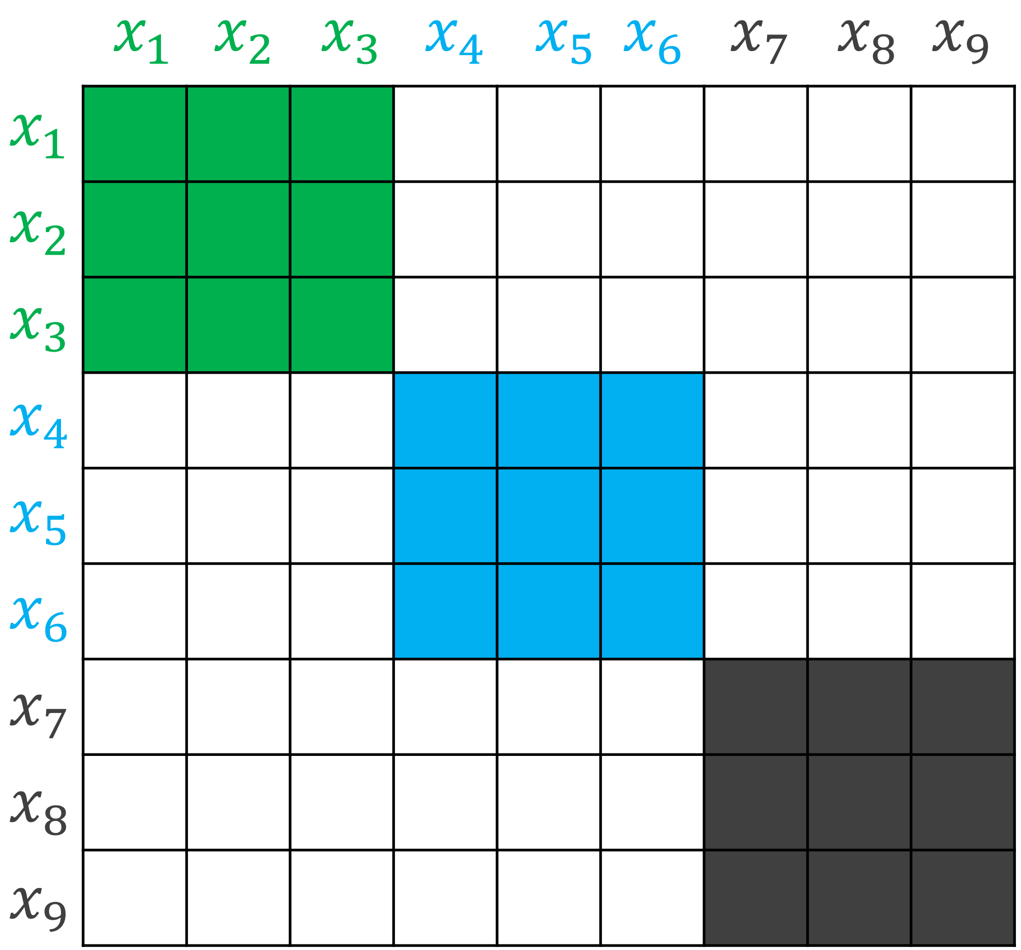


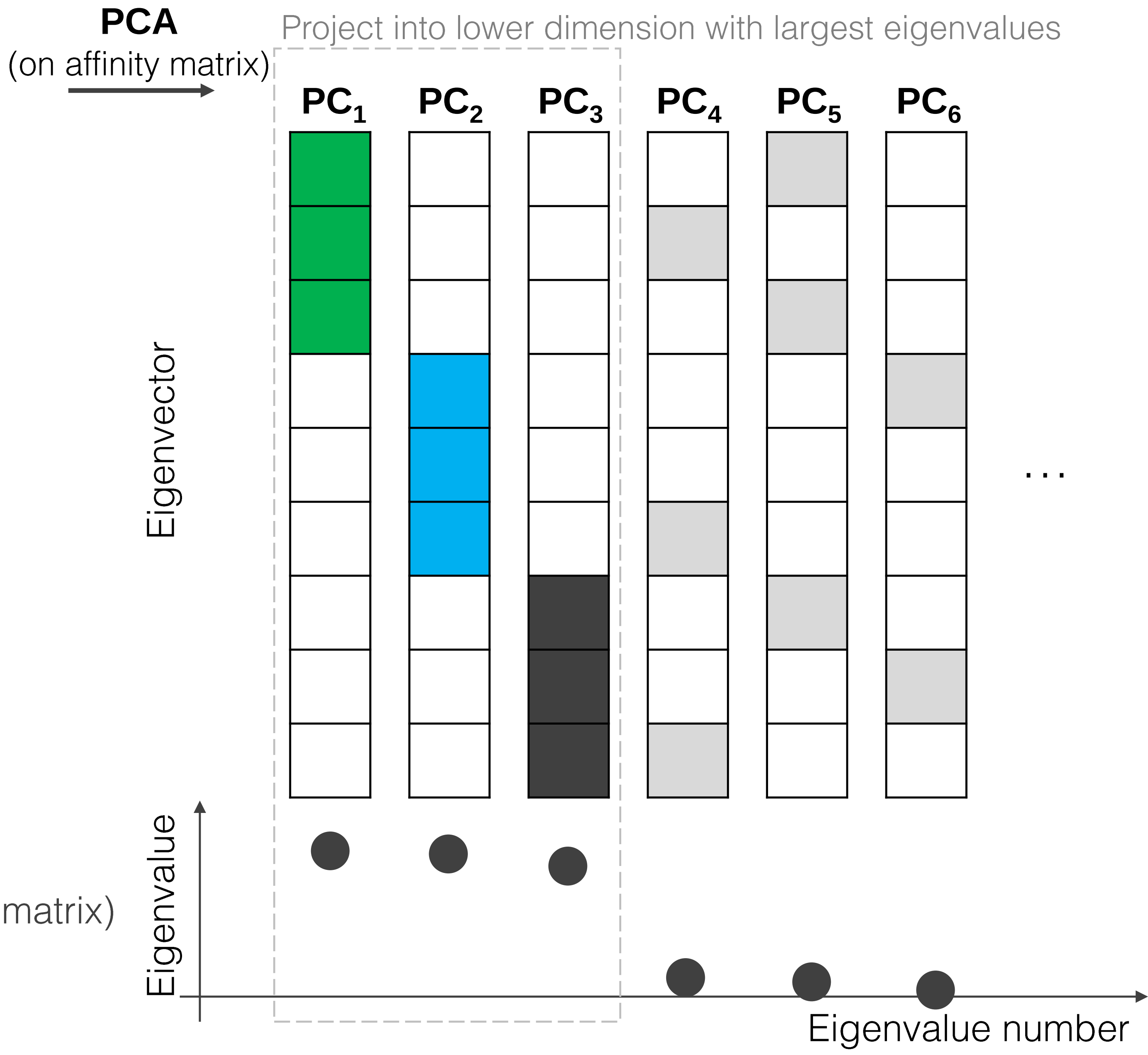
Affinity Matrix



Concept from Sebastian Thrun and Peter Norvig

Affinity matrix



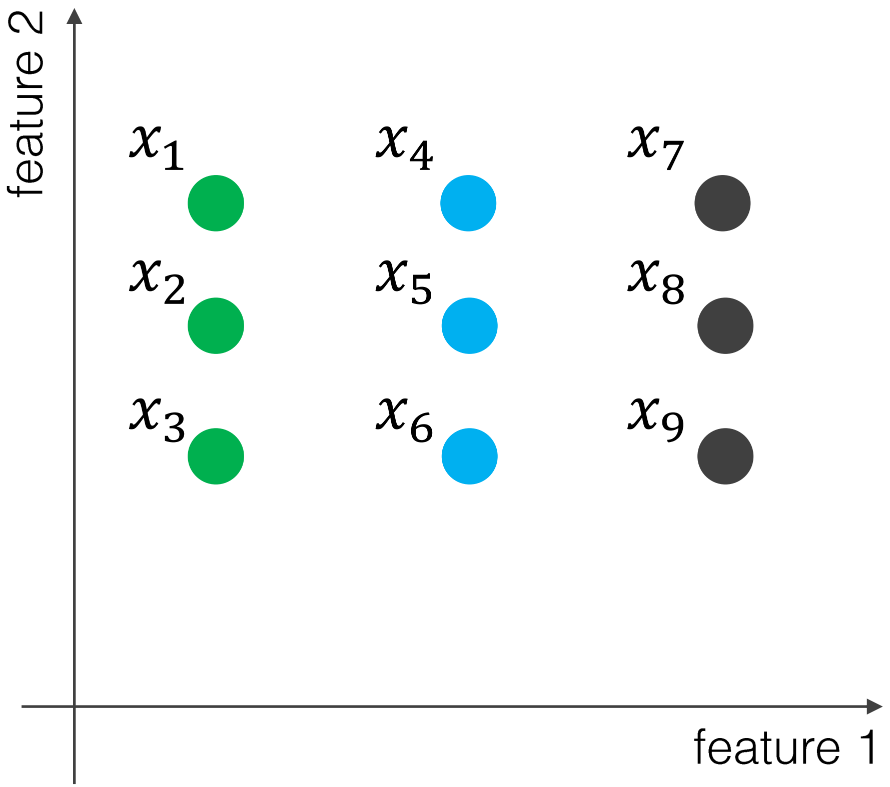


Spectral Clustering

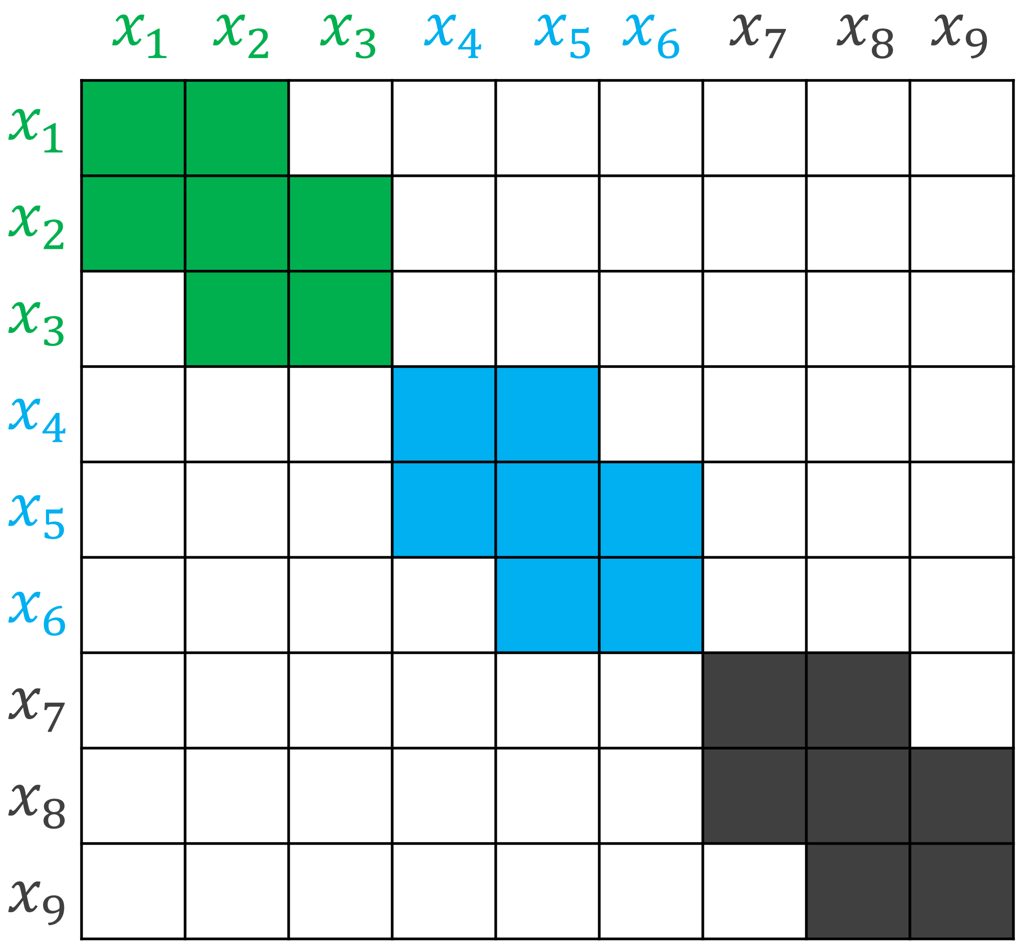
1. Project into lower dimension (PCA on affinity matrix)
2. Perform clustering in the lower dimension (often K-means)

Concept from Sebastian Thrun and Peter Norvig

Spectral Clustering: another Example

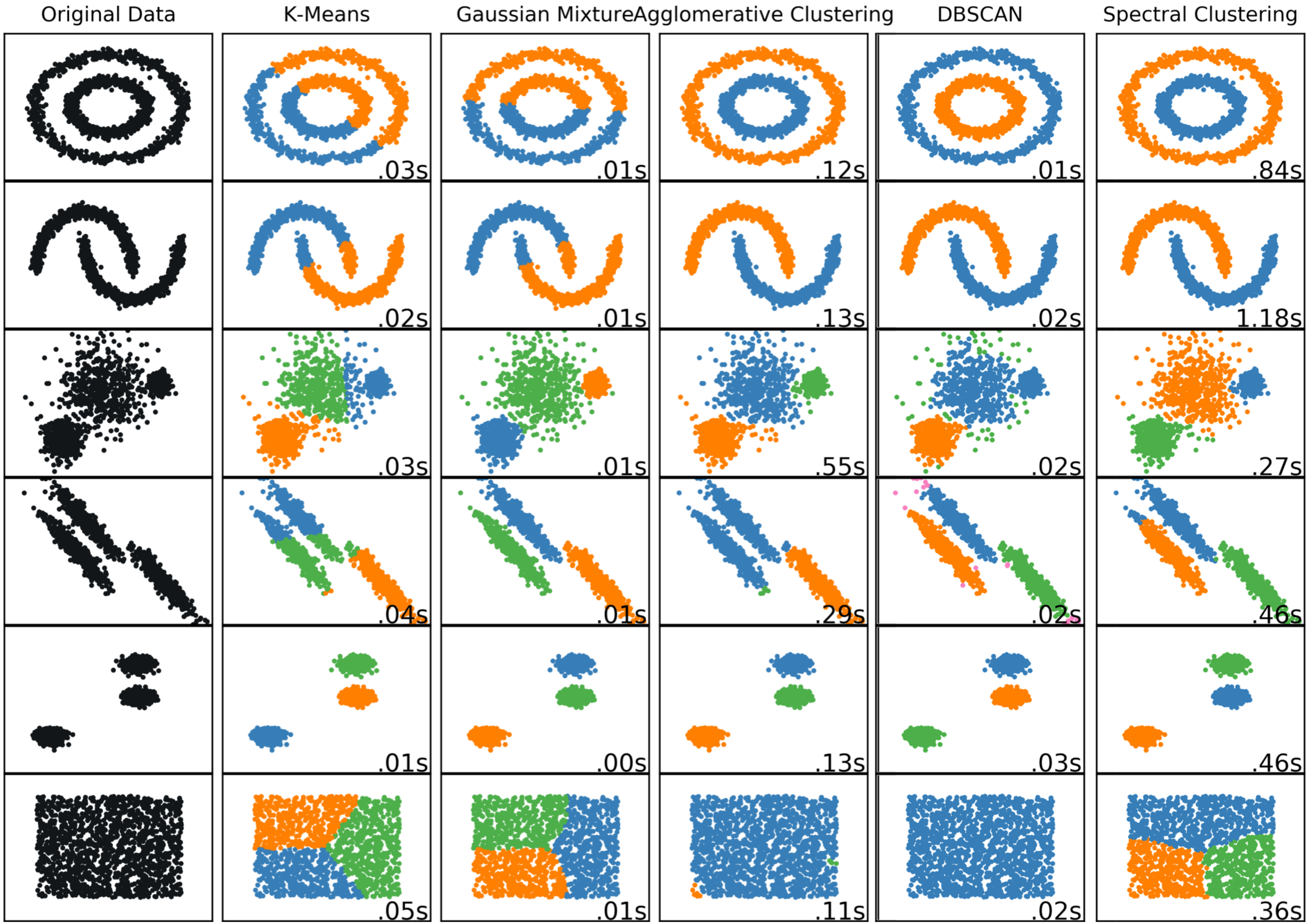


Affinity Matrix



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

Centroid-based clustering (e.g. **K-Means**)
Distribution-based clustering (e.g. **Gaussian mixture model**)
Density-based clustering (e.g. **DBSCAN**, mean-shift)
Hierarchical clustering (e.g. **agglomerative clustering**)
  a.k.a. connectivity-based clustering
Graph-based clustering (e.g. **spectral clustering**, affinity propagation)

Cluster assignment

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

Clustering choices:

1. How should the data be scaled?

2. For K-means and GMMs: how many clusters to estimate?

3. For hierarchical clustering: dissimilarity measure, linkage, where to cut dendrogram

Approach: try multiple options, and select the one with the most useful or interpretable solution

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