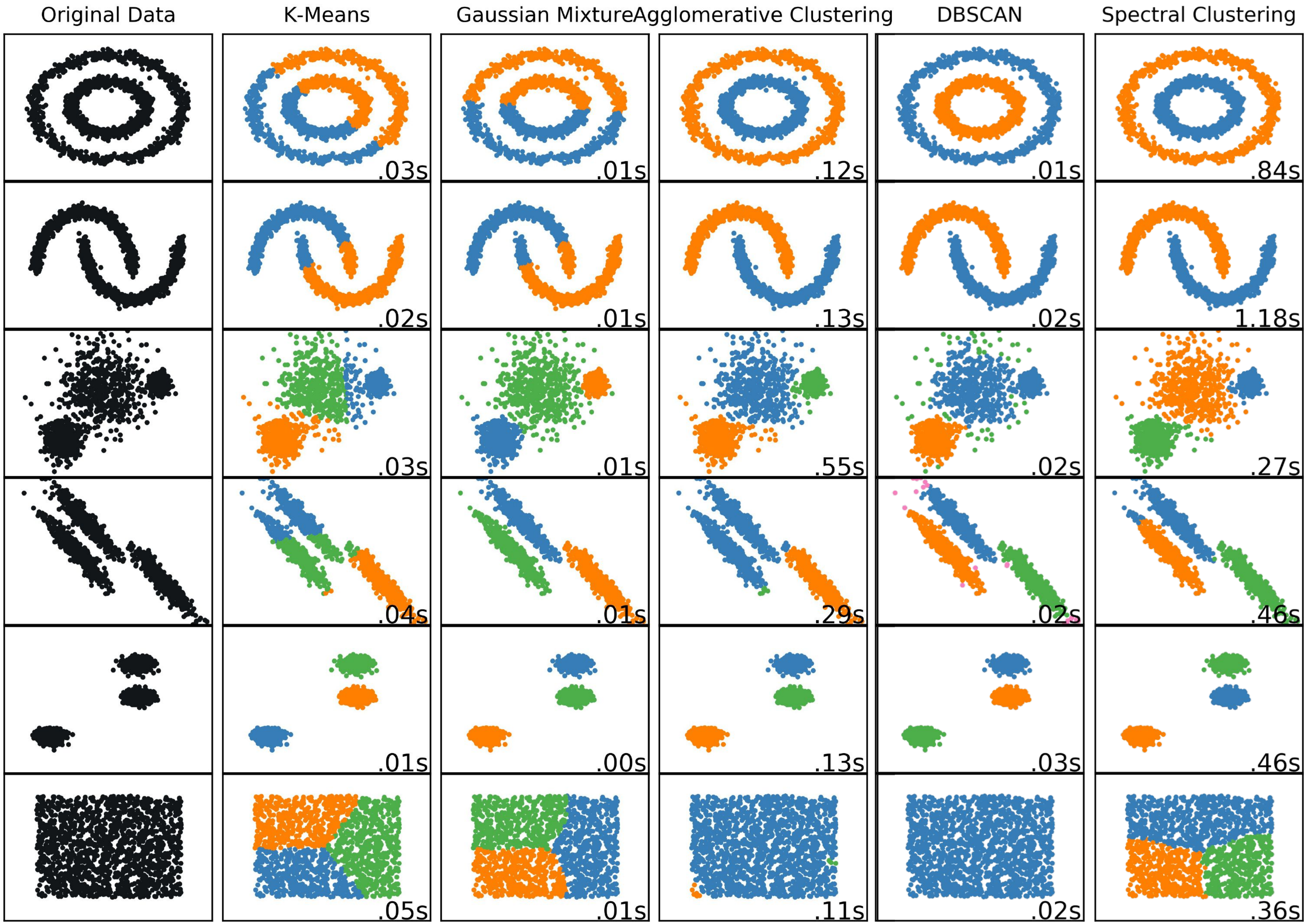
Clustering II

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

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

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

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

Linkage criterion

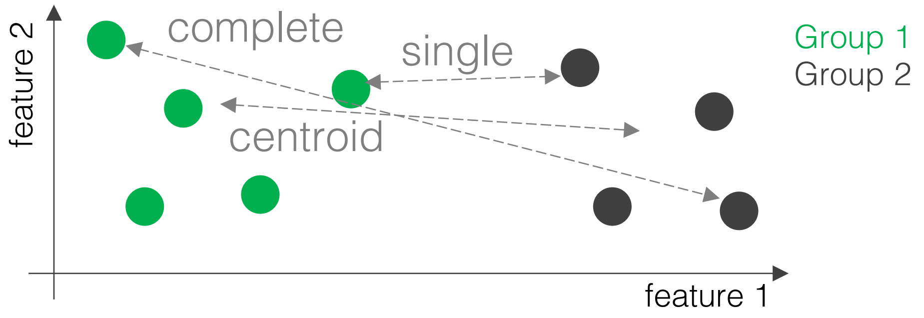
How to measure distance/dissimilarity between groups or sets

Complete = maximum intercluster dissimilarity

Single = minimum intercluster dissimilarity

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

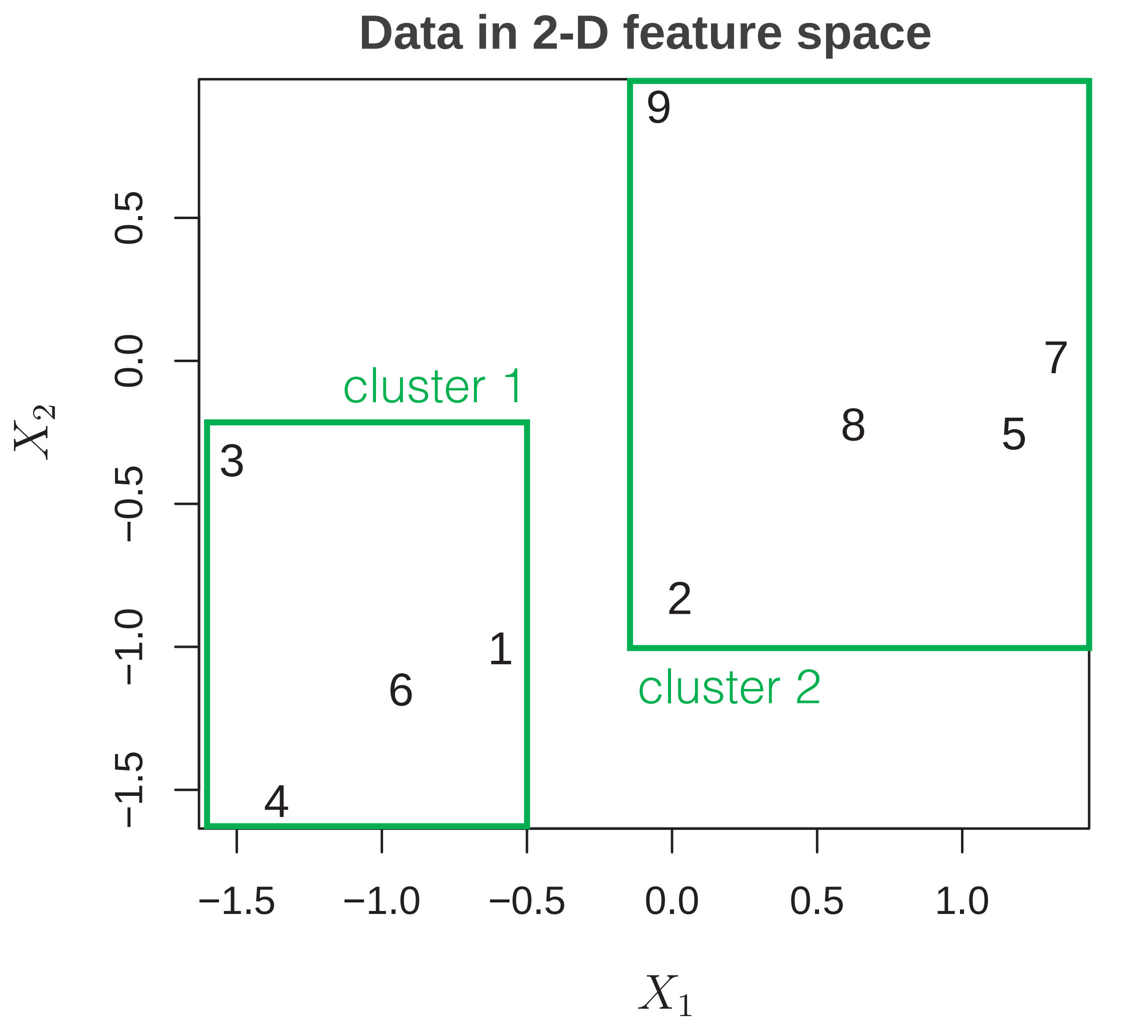
Centroid = dissimilarity between cluster centroids



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



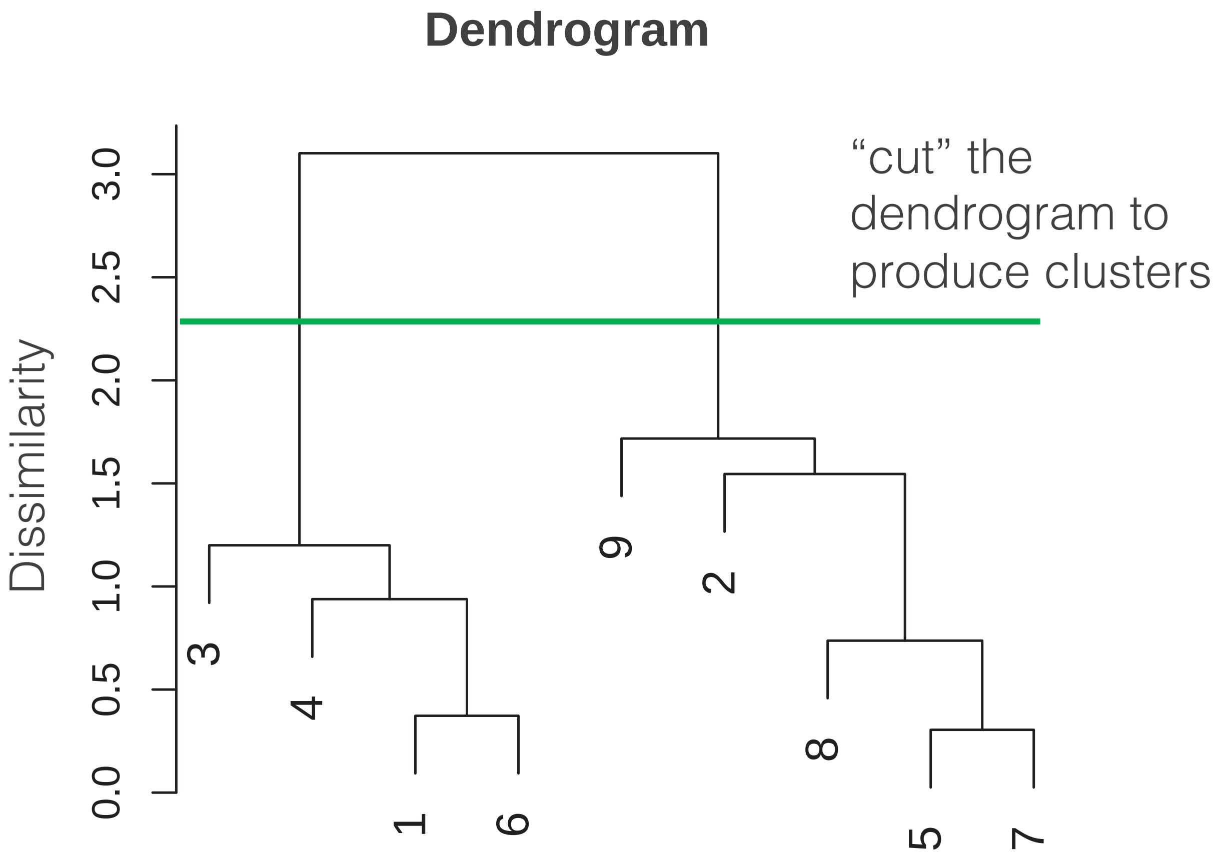
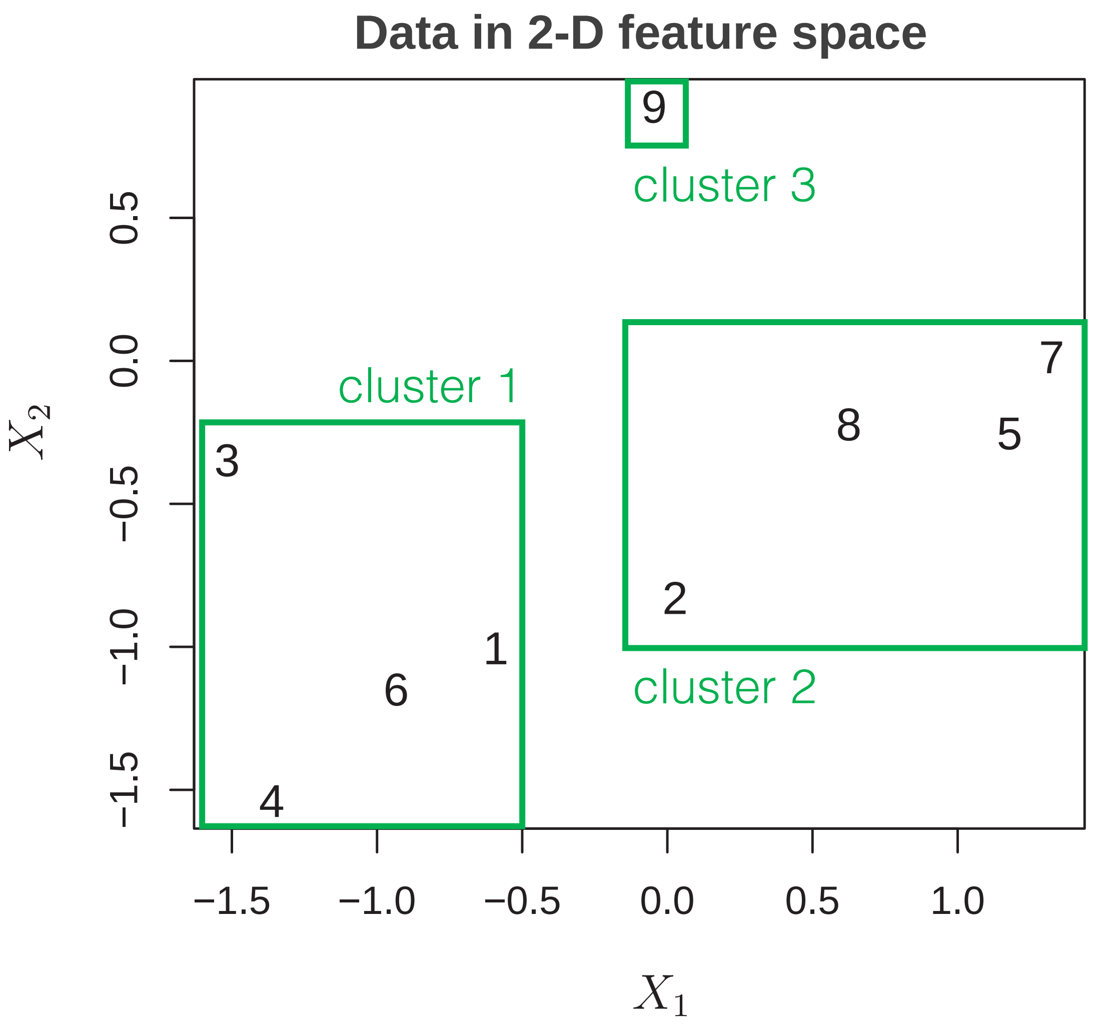


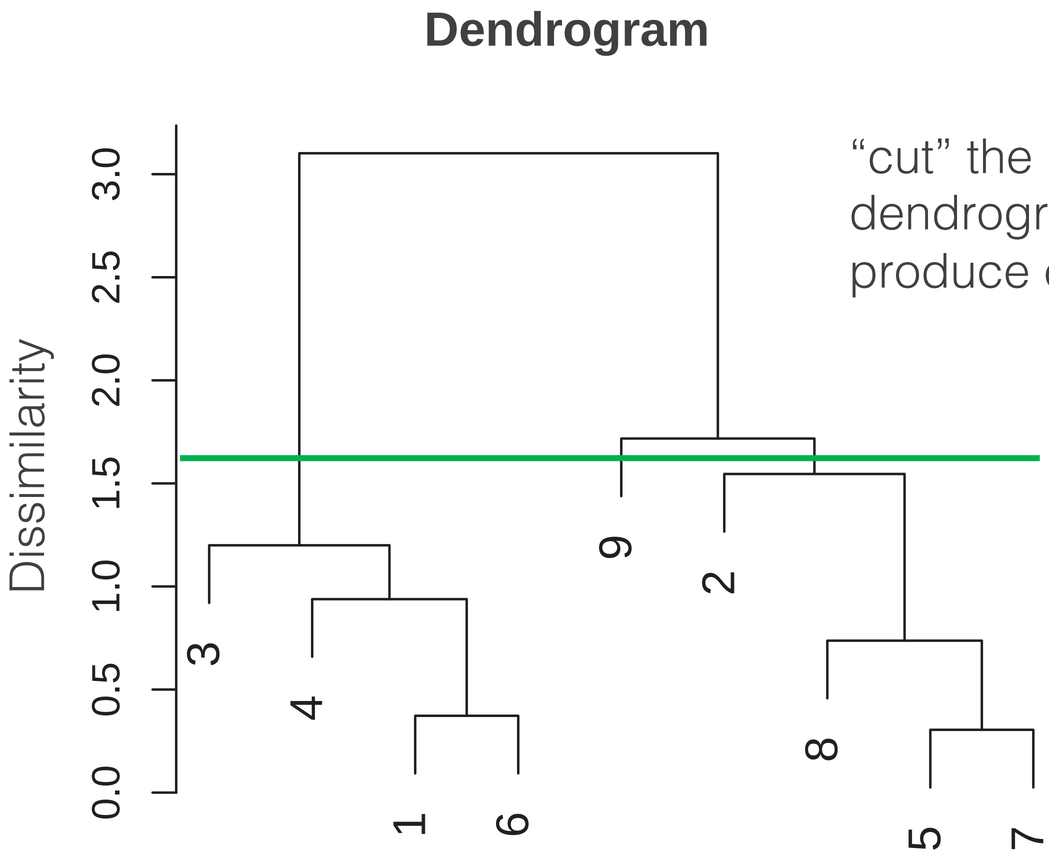
Image from James et al., Introduction to Statistical Learning, 2013

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





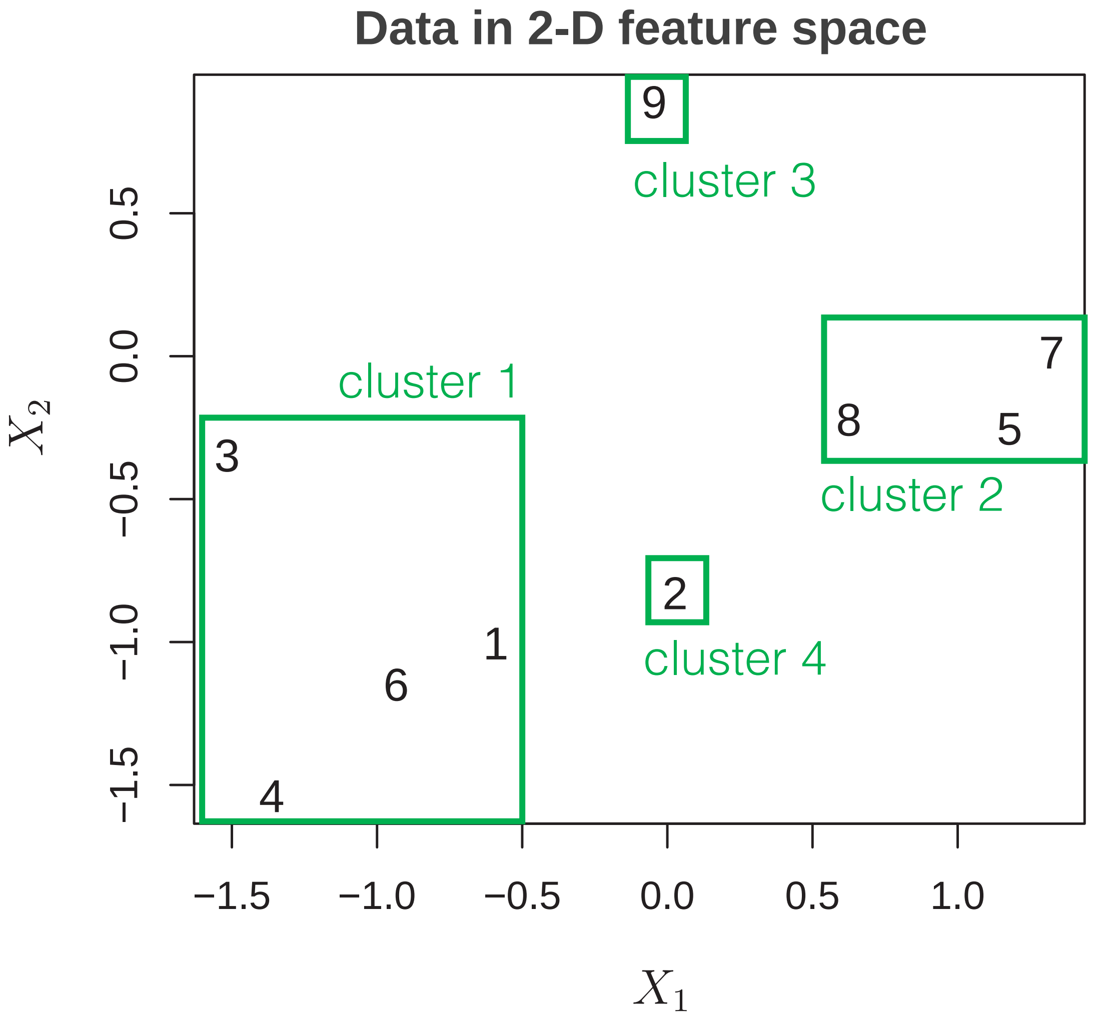
"cut" the dendrogram to produce clusters

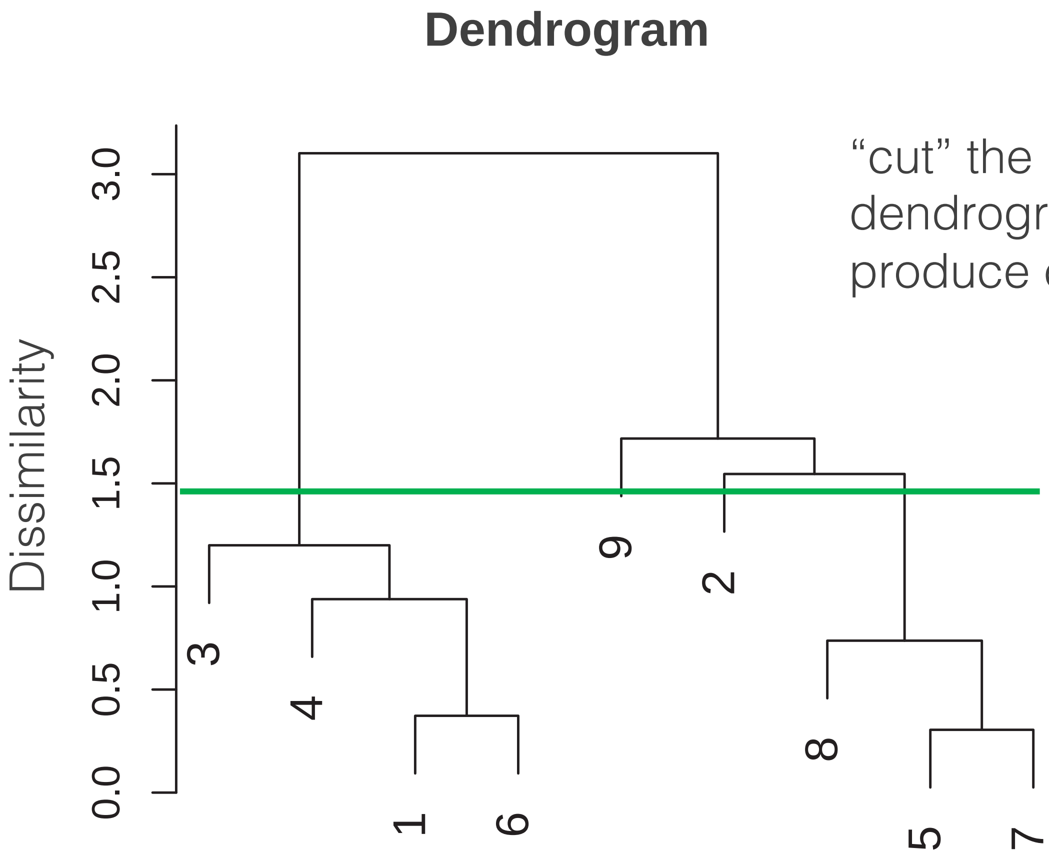
Image from James et al., Introduction to Statistical Learning, 2013

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





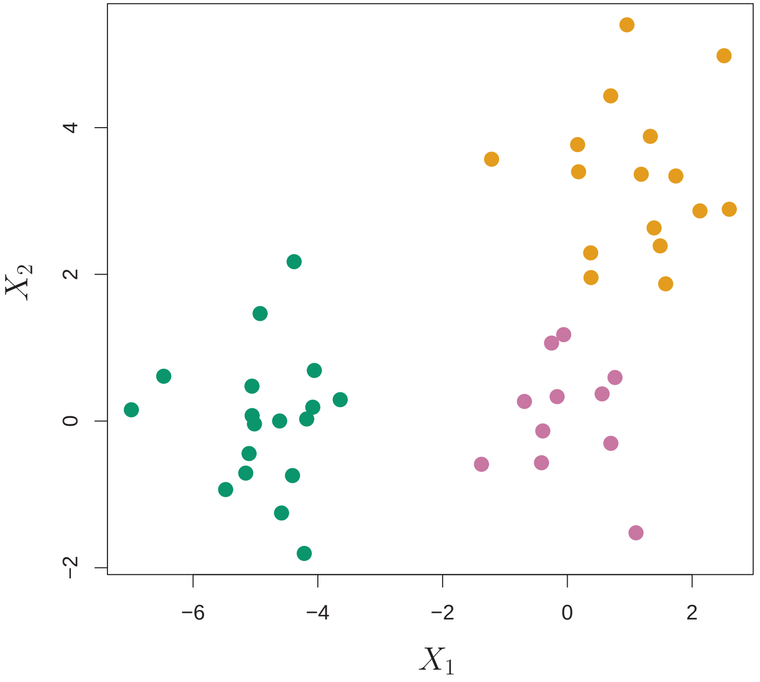
"cut" the dendrogram to produce clusters

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

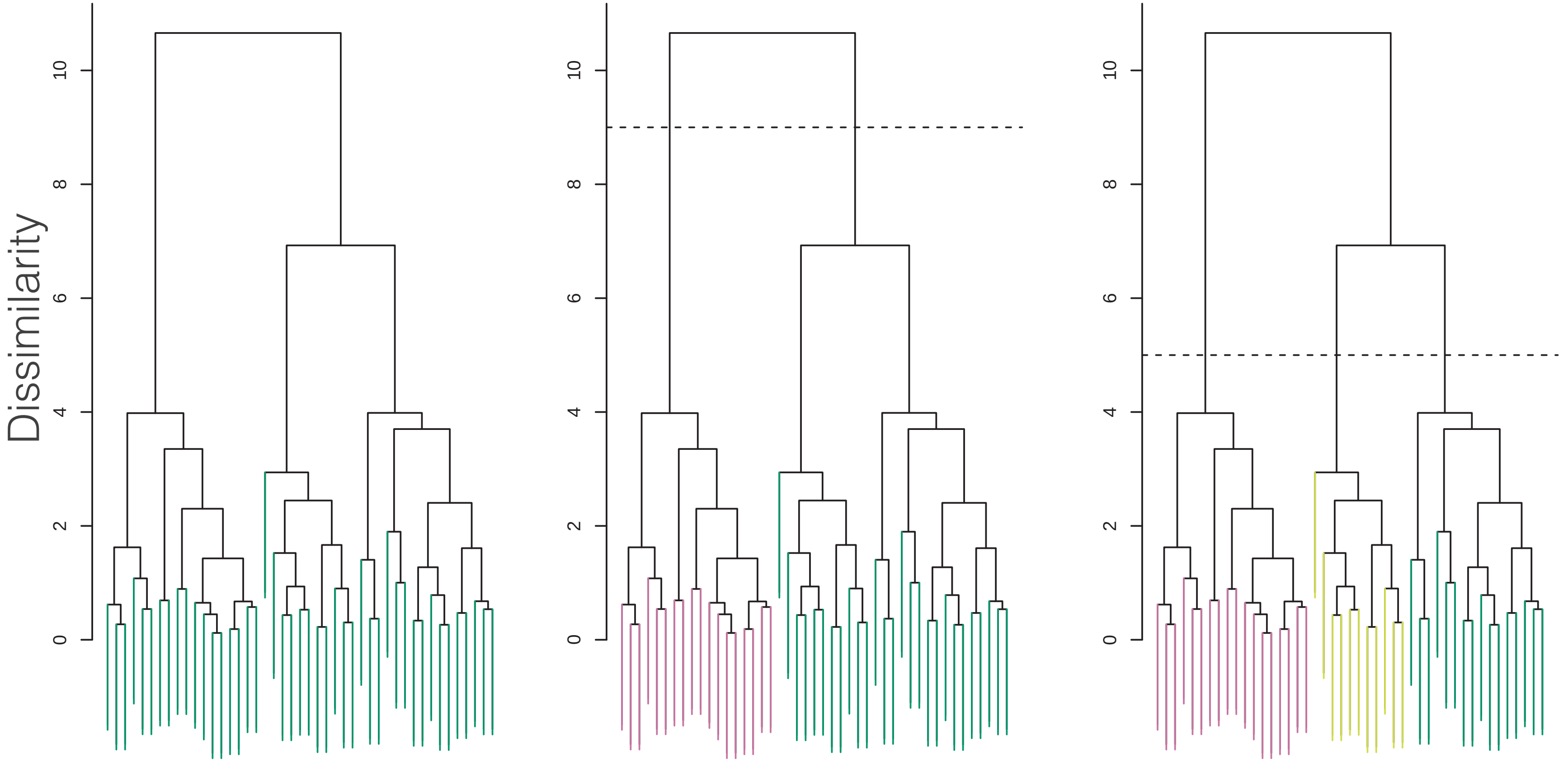
Example of agglomerative clustering

With complete linkage and Euclidean distance

Data in 2-D feature space



Original Dendrogram | **Dendrogram cut for 2 clusters** | **Dendrogram cut for 3 clusters**



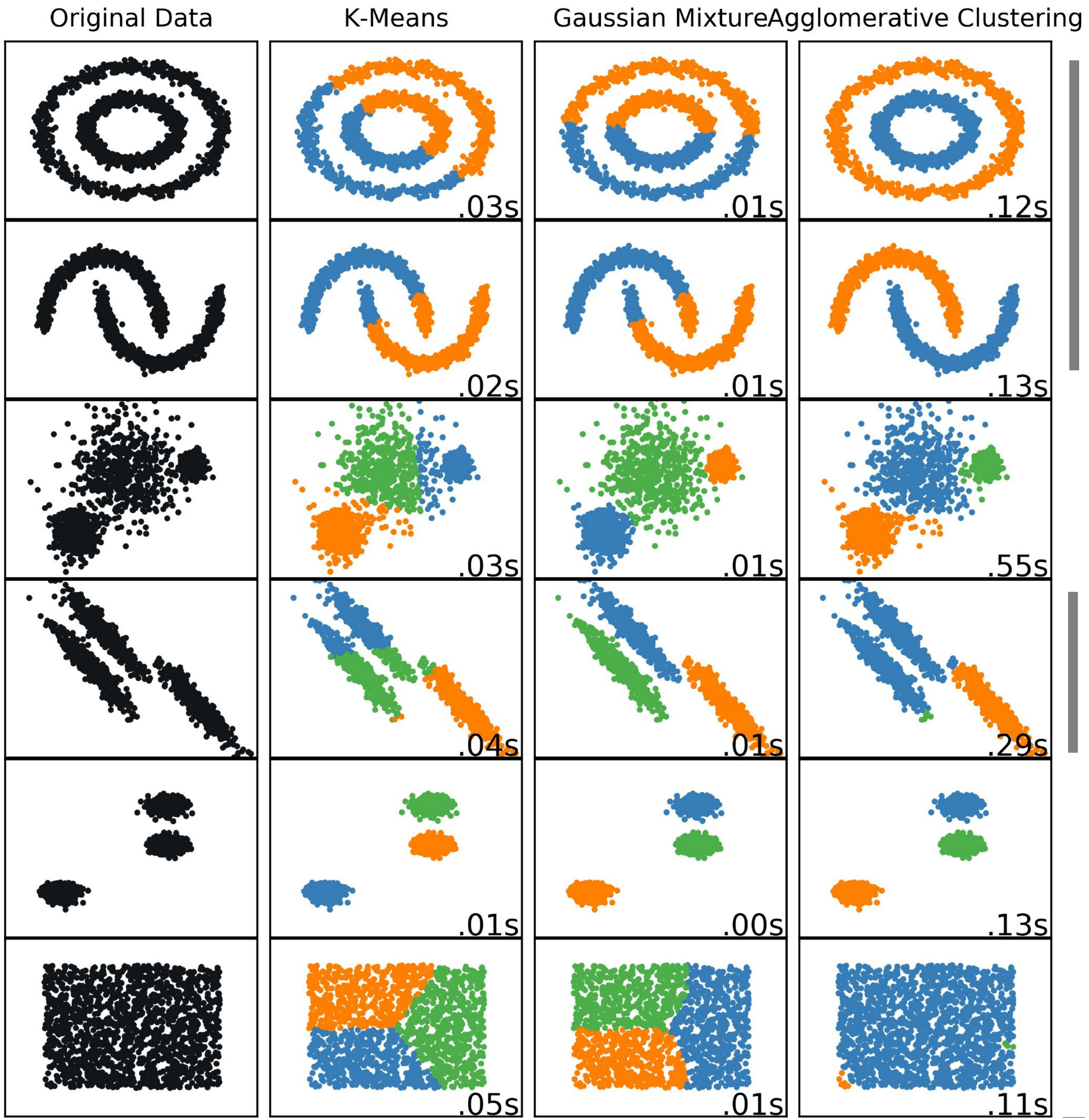
Note: colors do not directly map to plot on the left

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

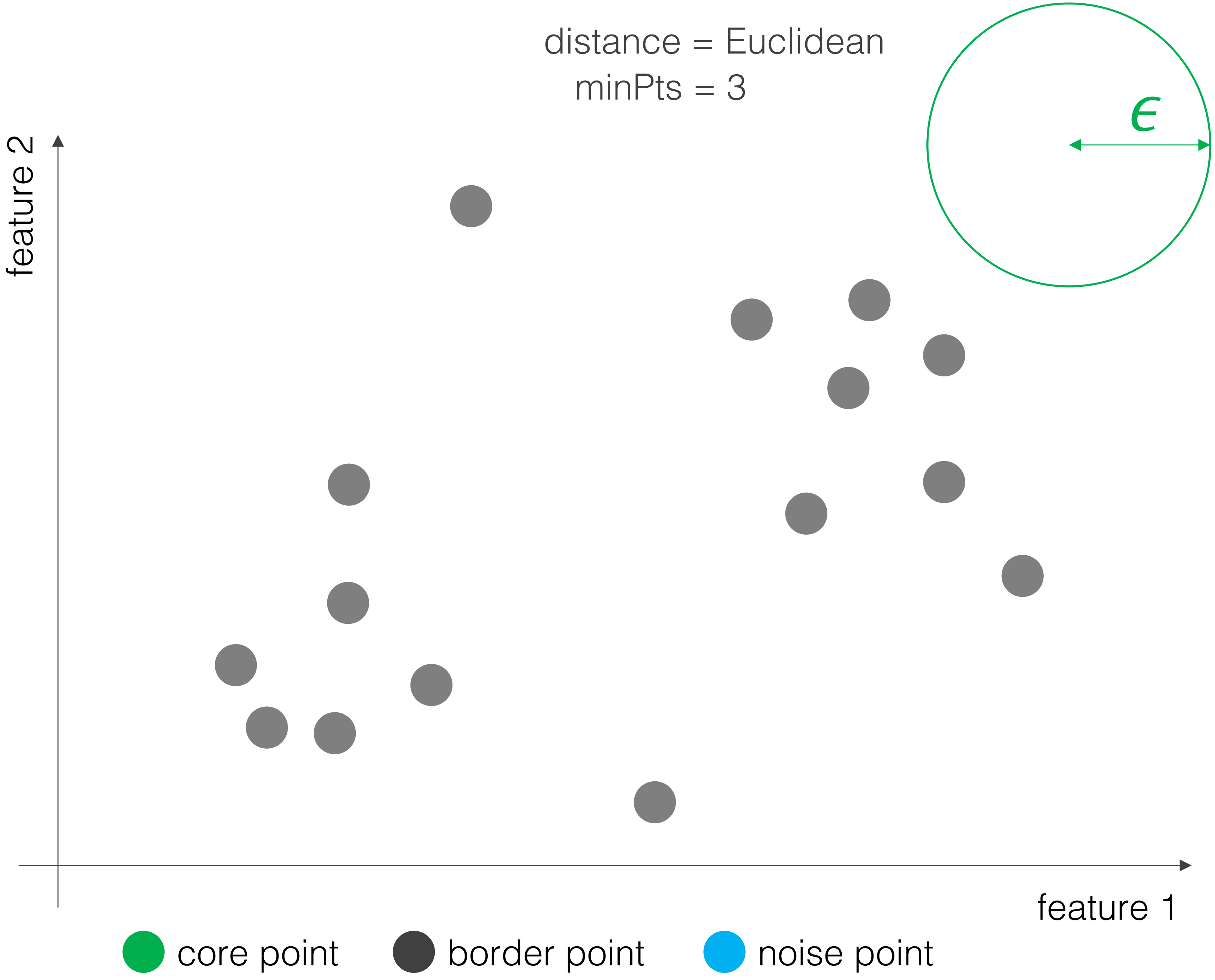
By Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, 1996

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



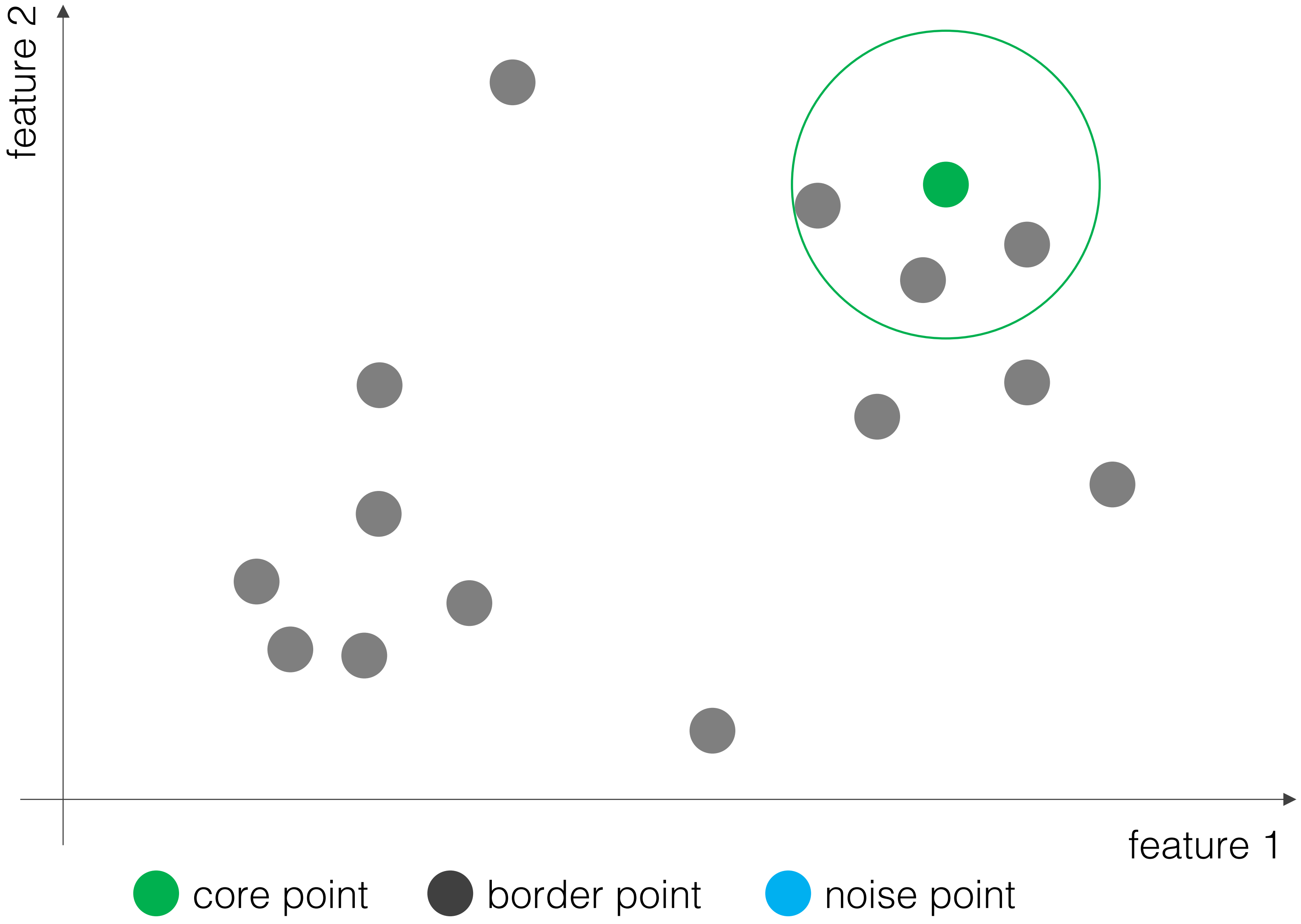
DBSCAN

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

distance = Euclidean
minPts = 3

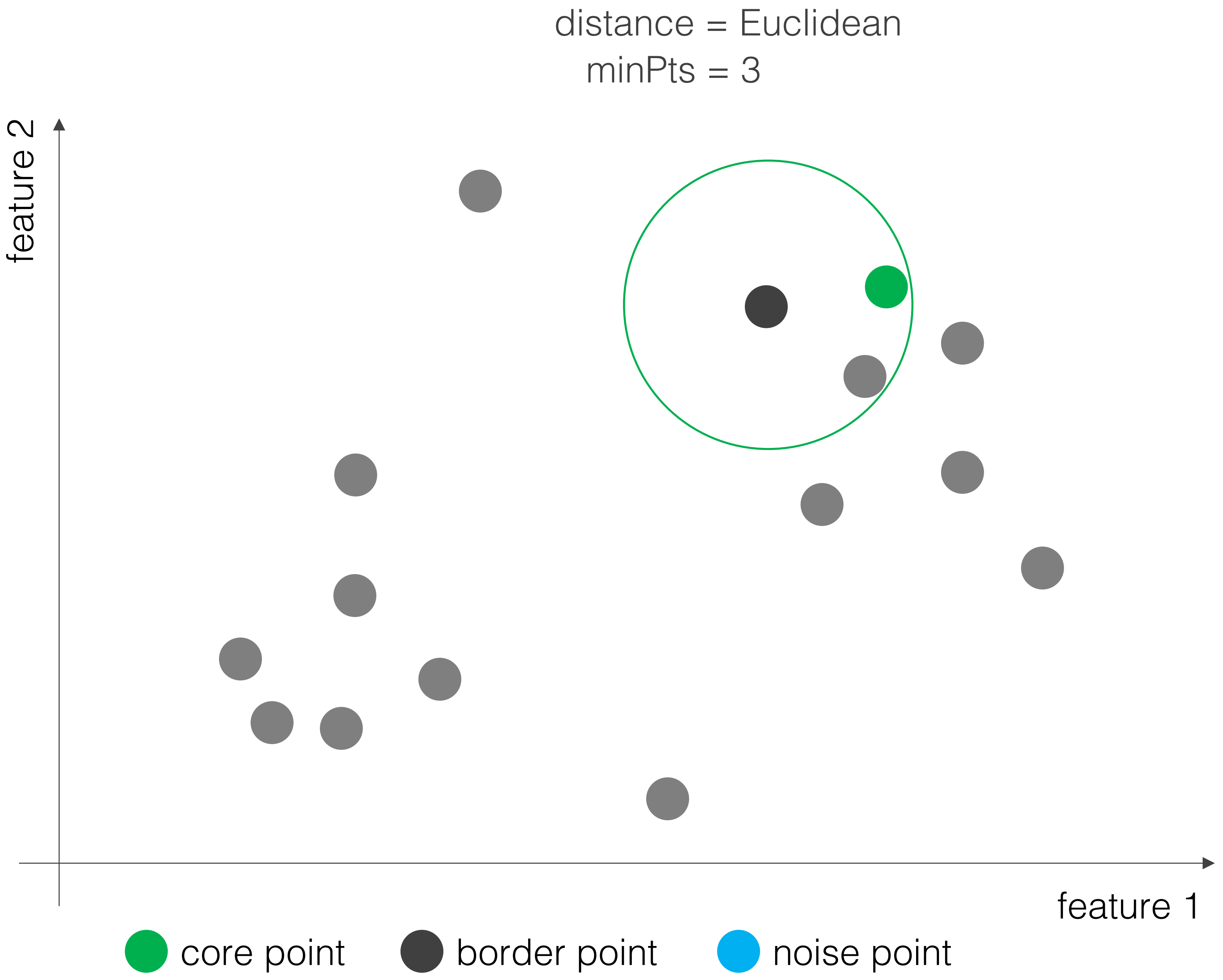


DBSCAN

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

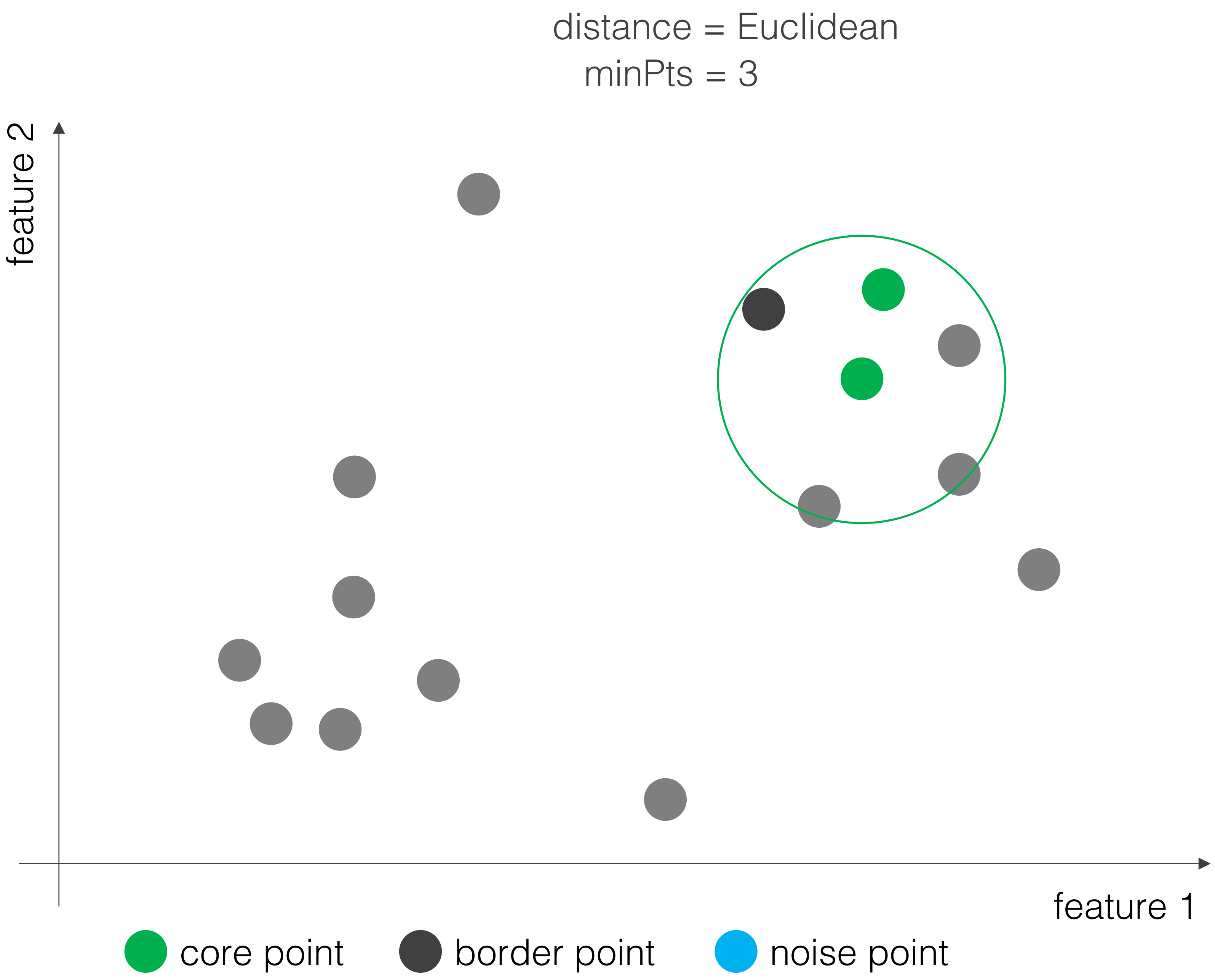


DBSCAN

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

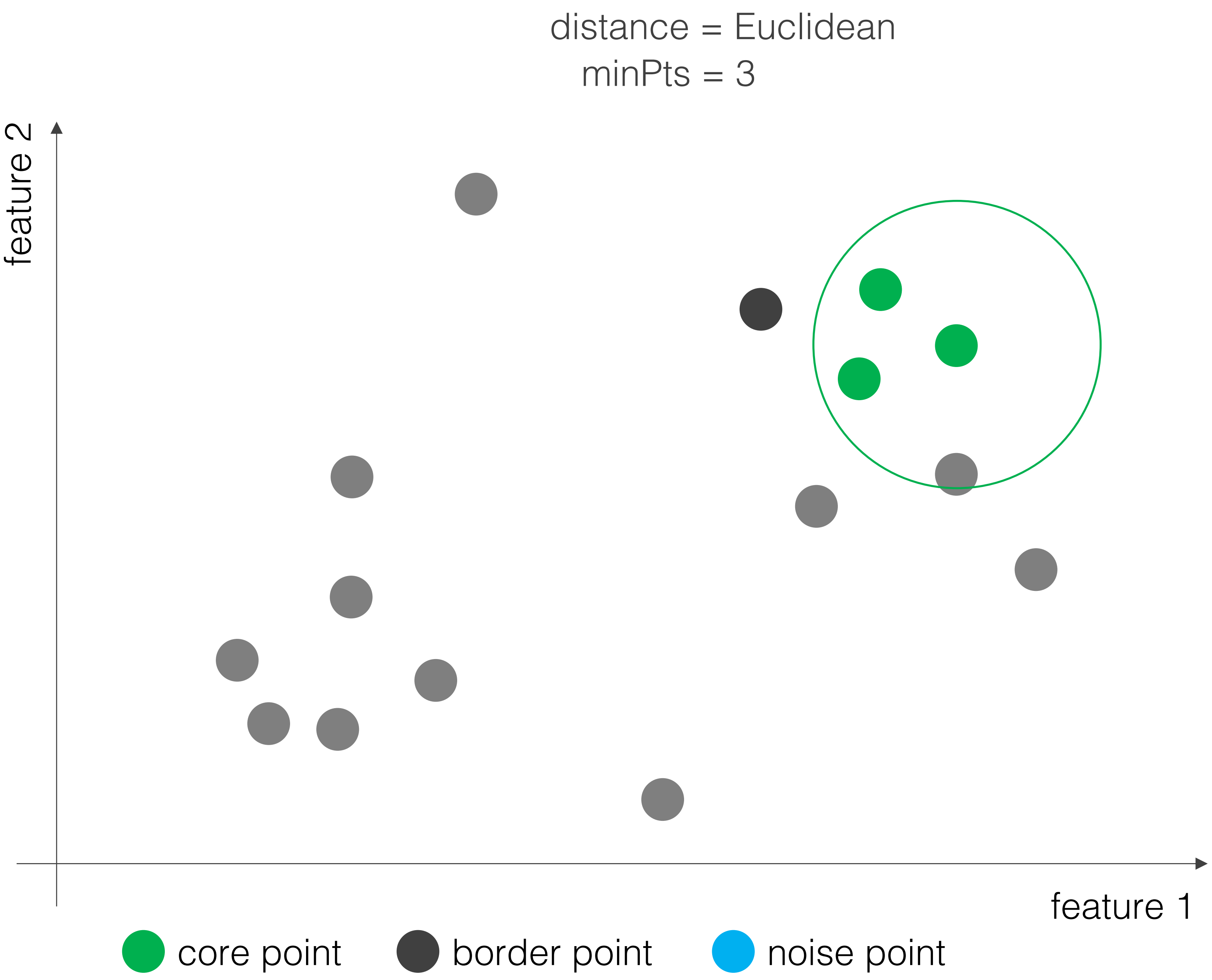


DBSCAN

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

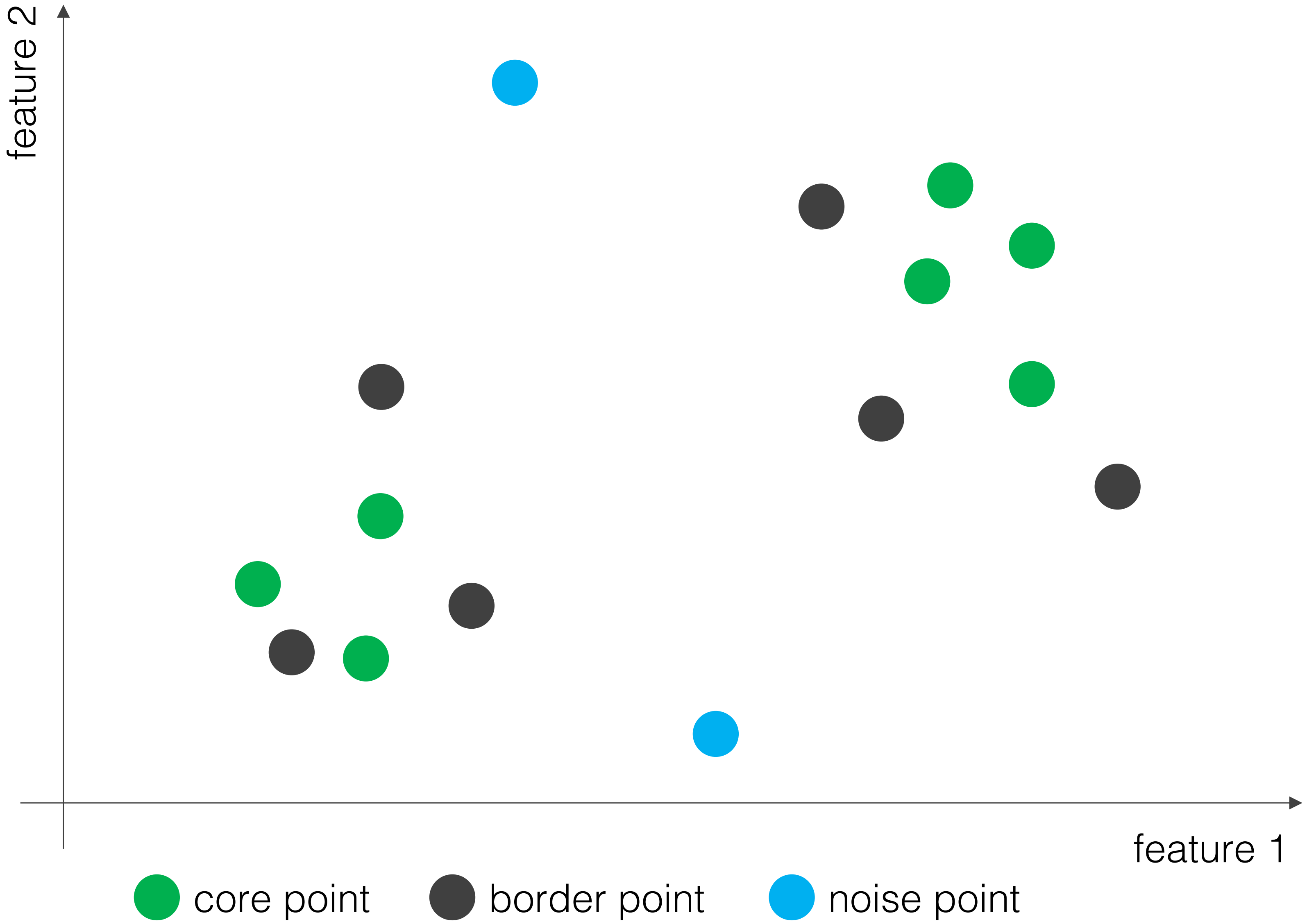


DBSCAN

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

Parameters:

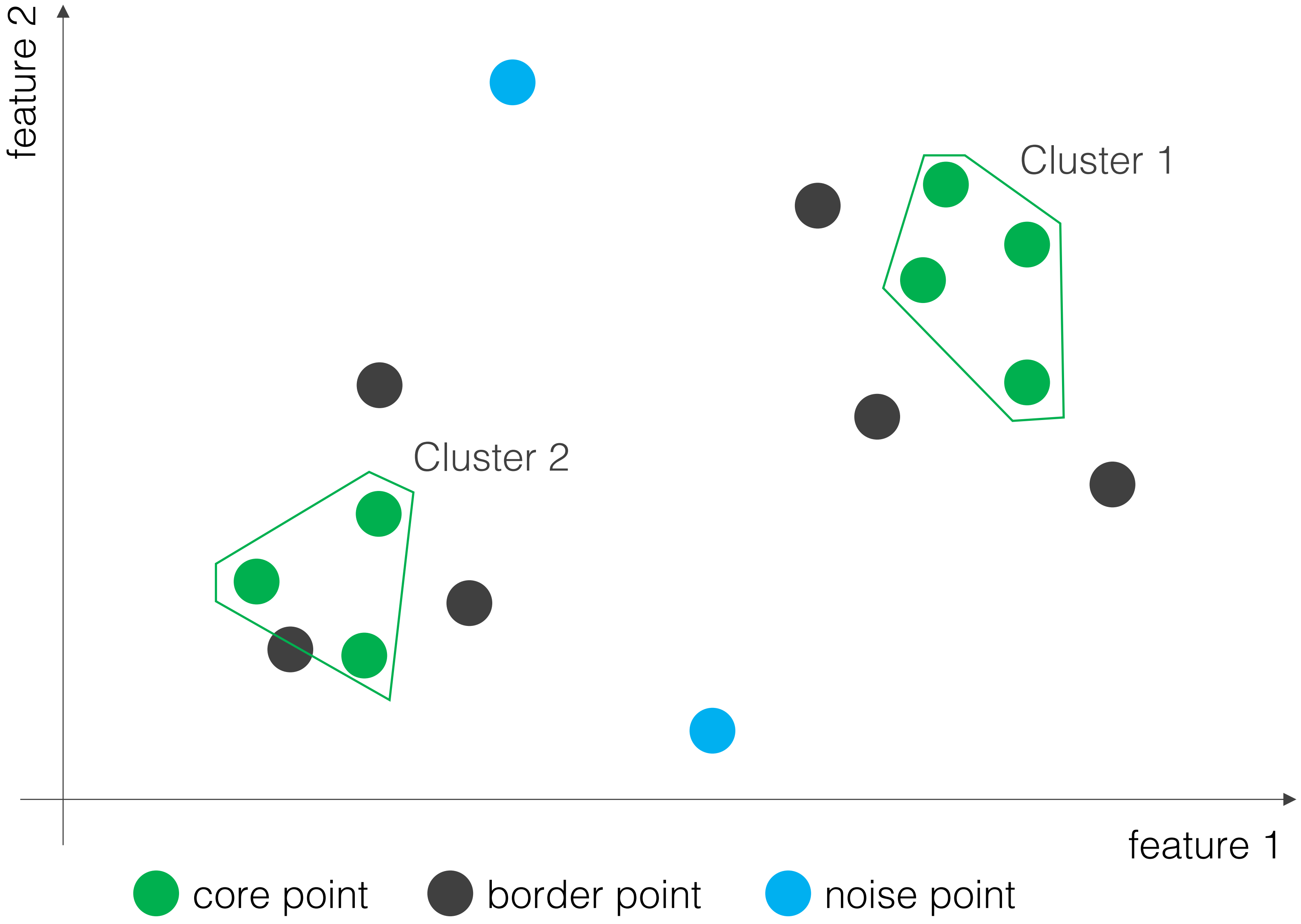
1. Distance measure
2. The radius of a neighbor, ϵ
3. 'minPts': The number of neighbors for a point to be considered a core point

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

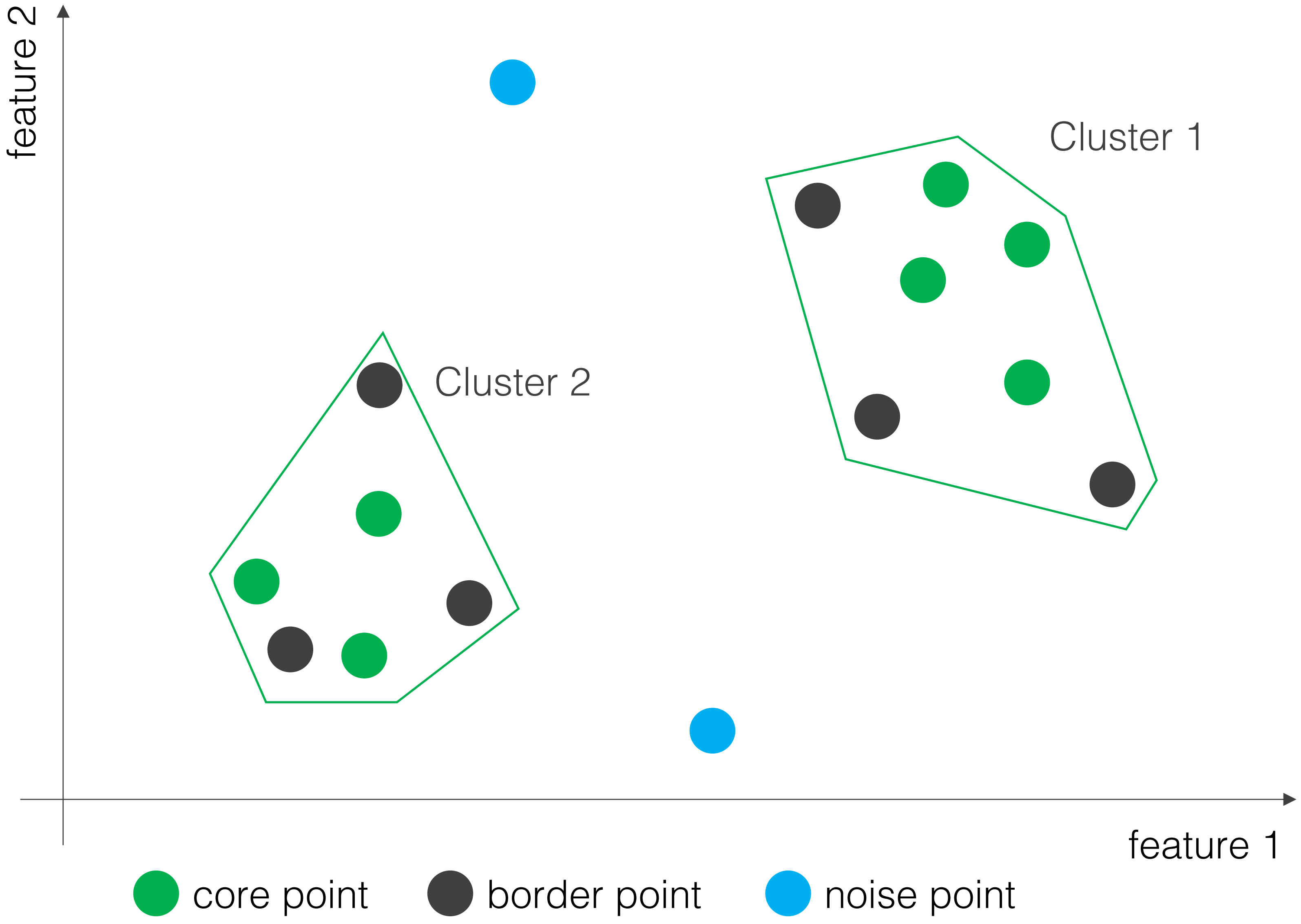


DBSCAN

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

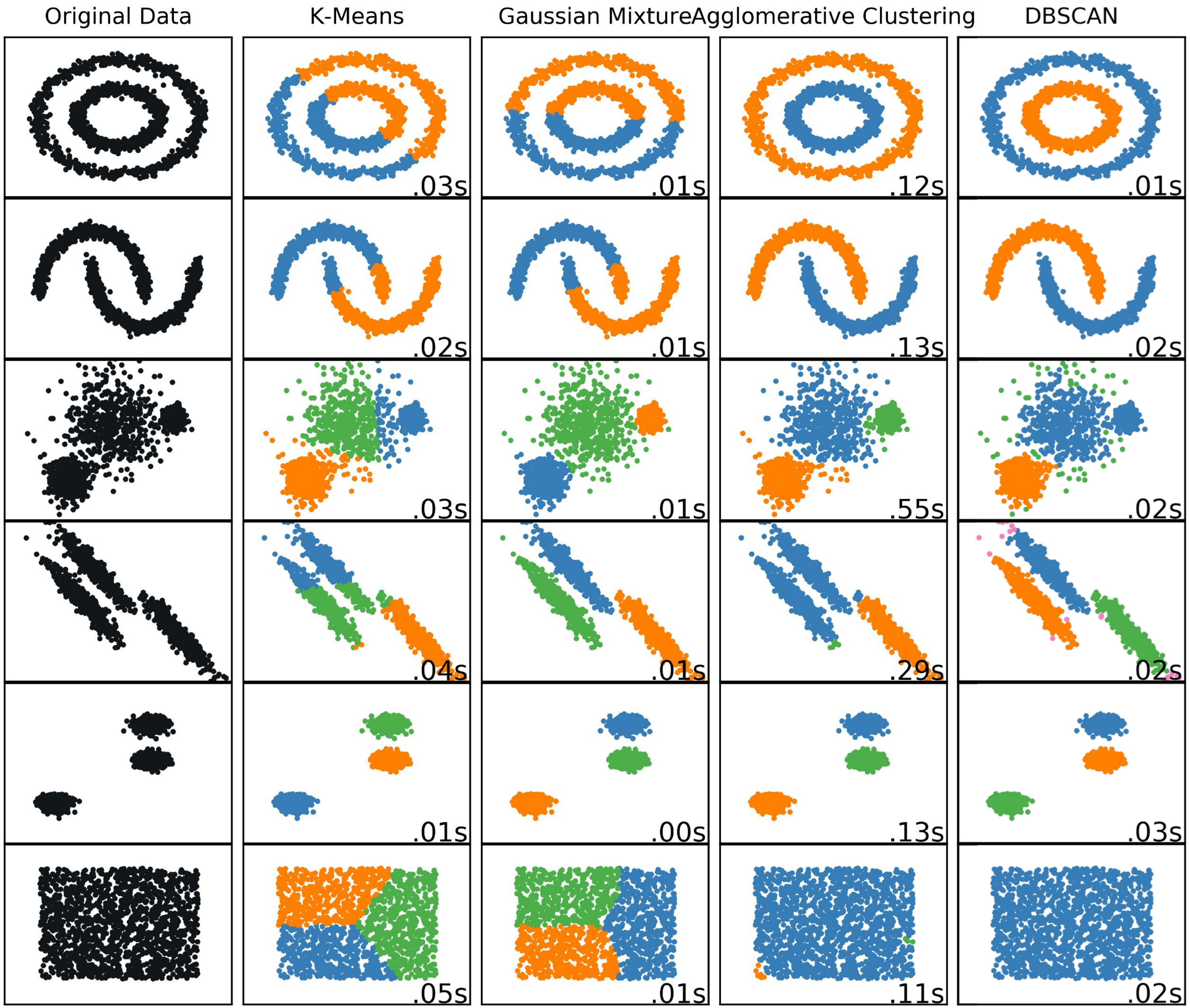


DBSCAN

- 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

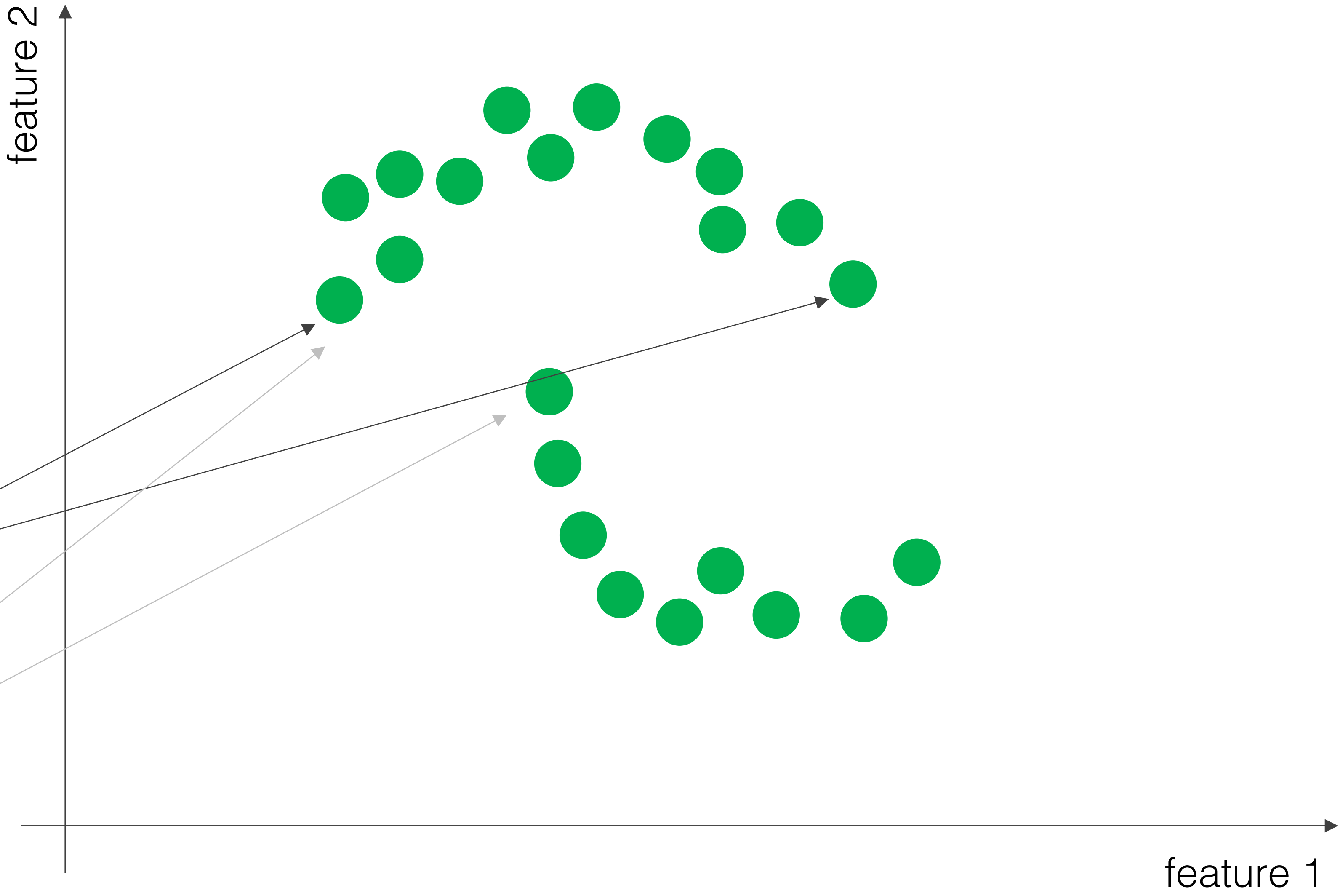
Spectral Clustering

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

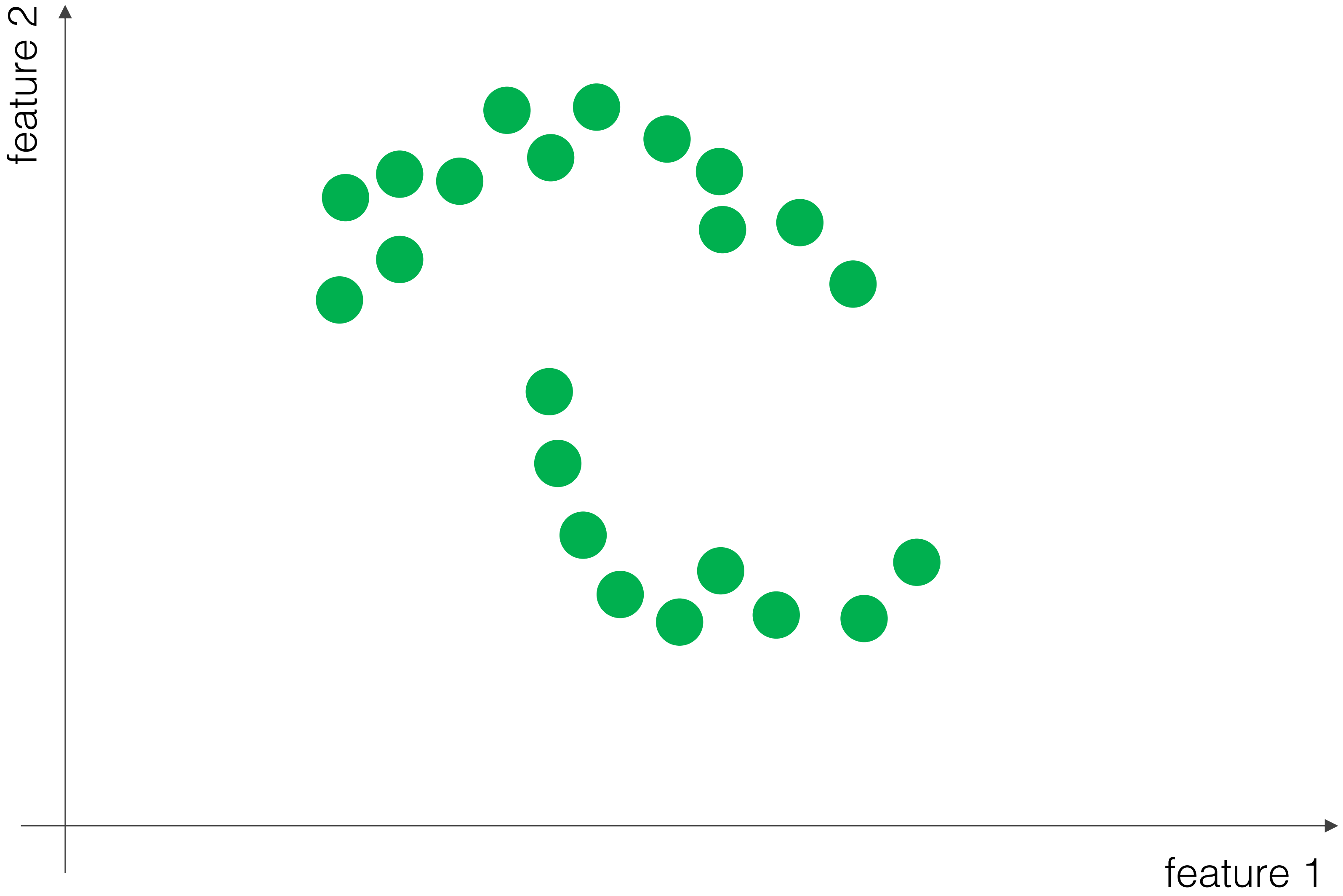
Spectral Clustering

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

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

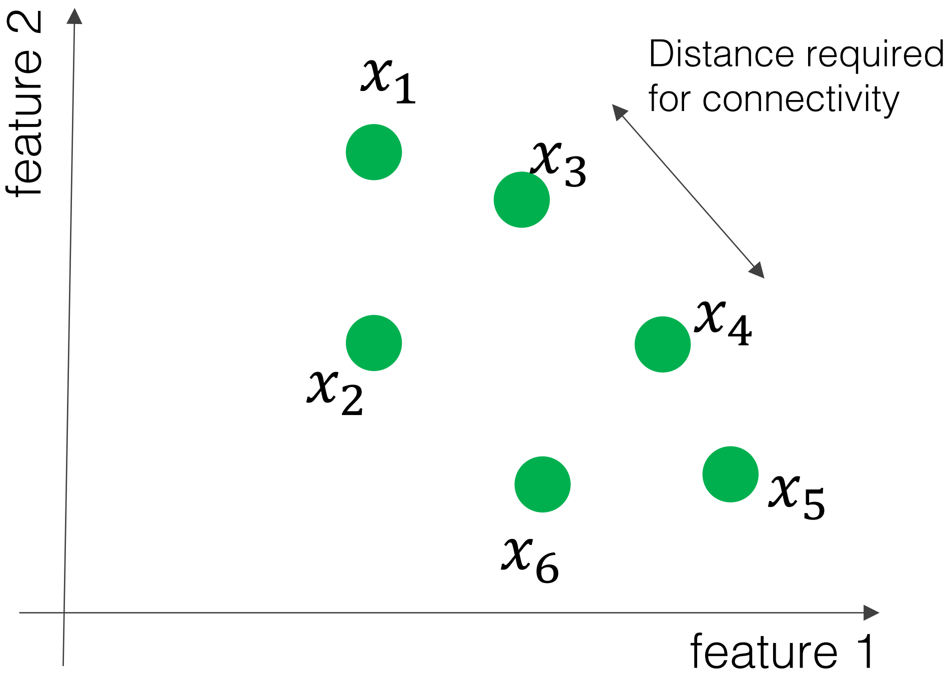
For example, using Euclidean distance, we could define affinity as:

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

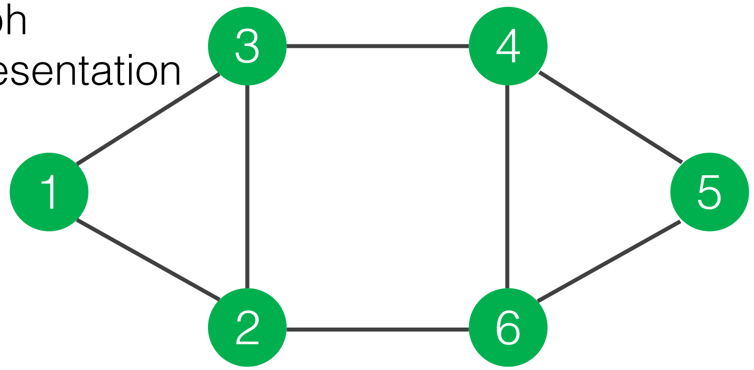


Concept from Sebastian Thrun and Peter Norvig

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	3	0
x_6	0	0	0	0	0	2

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	3	-1
x_6	0	-1	0	-1	-1	2

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

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	3	-1
x_6	0	-1	0	-1	-1	2

L

Eigenvectors of L

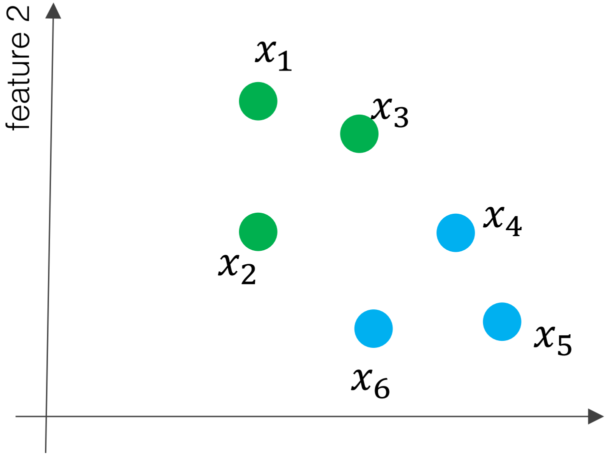
$\mathbf{u}_1$	$\mathbf{u}_2$	$\mathbf{u}_3$	$\mathbf{u}_4$	$\mathbf{u}_5$	$\mathbf{u}_6$
0.4	0.6	0.0	0.6	0.3	0.0
0.4	0.3	0.4	-0.4	-0.5	0.5
0.4	0.3	-0.4	-0.4	-0.1	-0.6
0.4	-0.3	-0.5	-0.1	0.3	0.6
0.3	-0.4	-0.2	0.5	-0.6	-0.1
0.5	-0.5	0.5	-0.1	0.4	-0.3

$\lambda_i =$

-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**

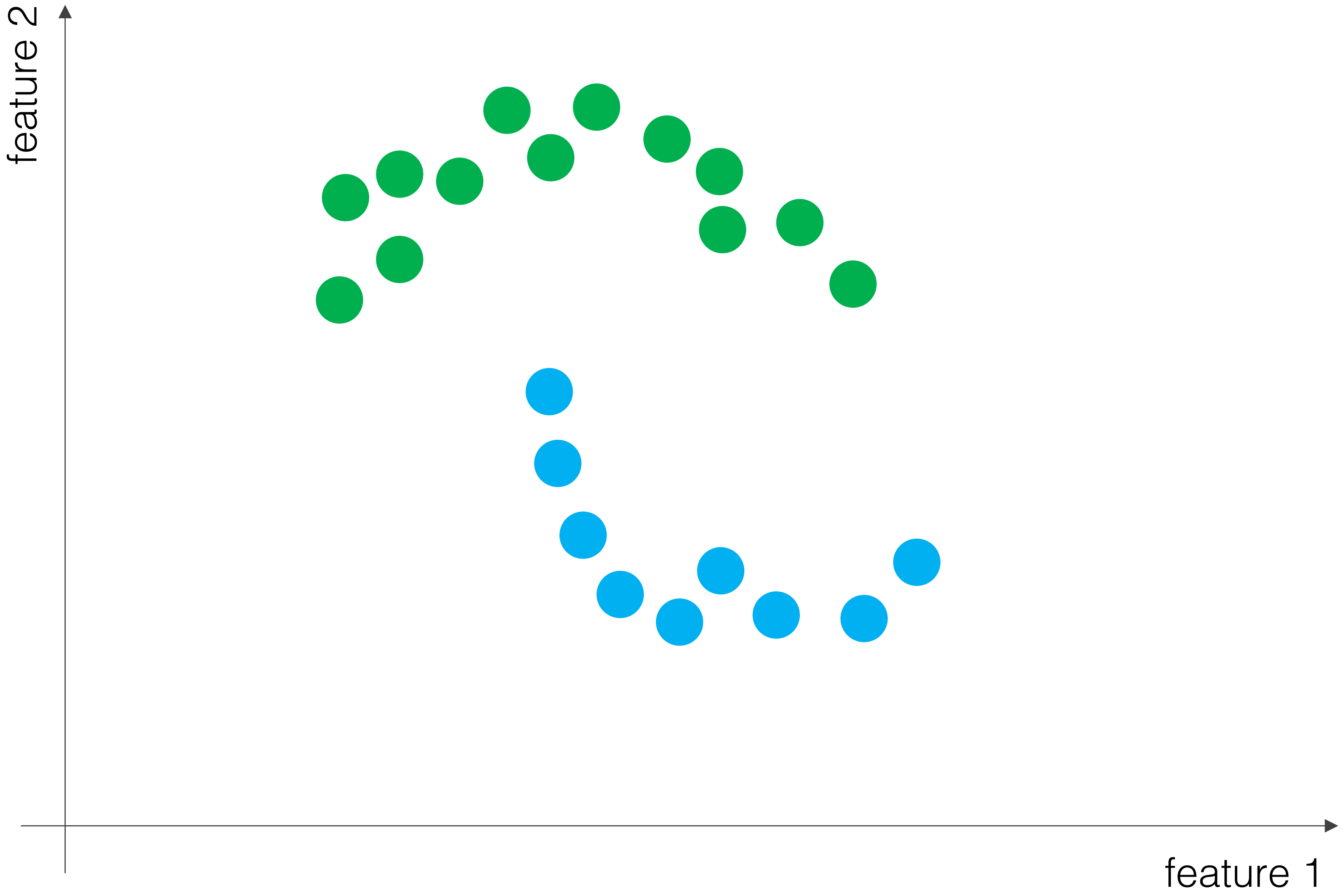


Get the eigenvectors of the Laplacian matrix, cluster points based on the eigenvectors (typically using k-means)

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)



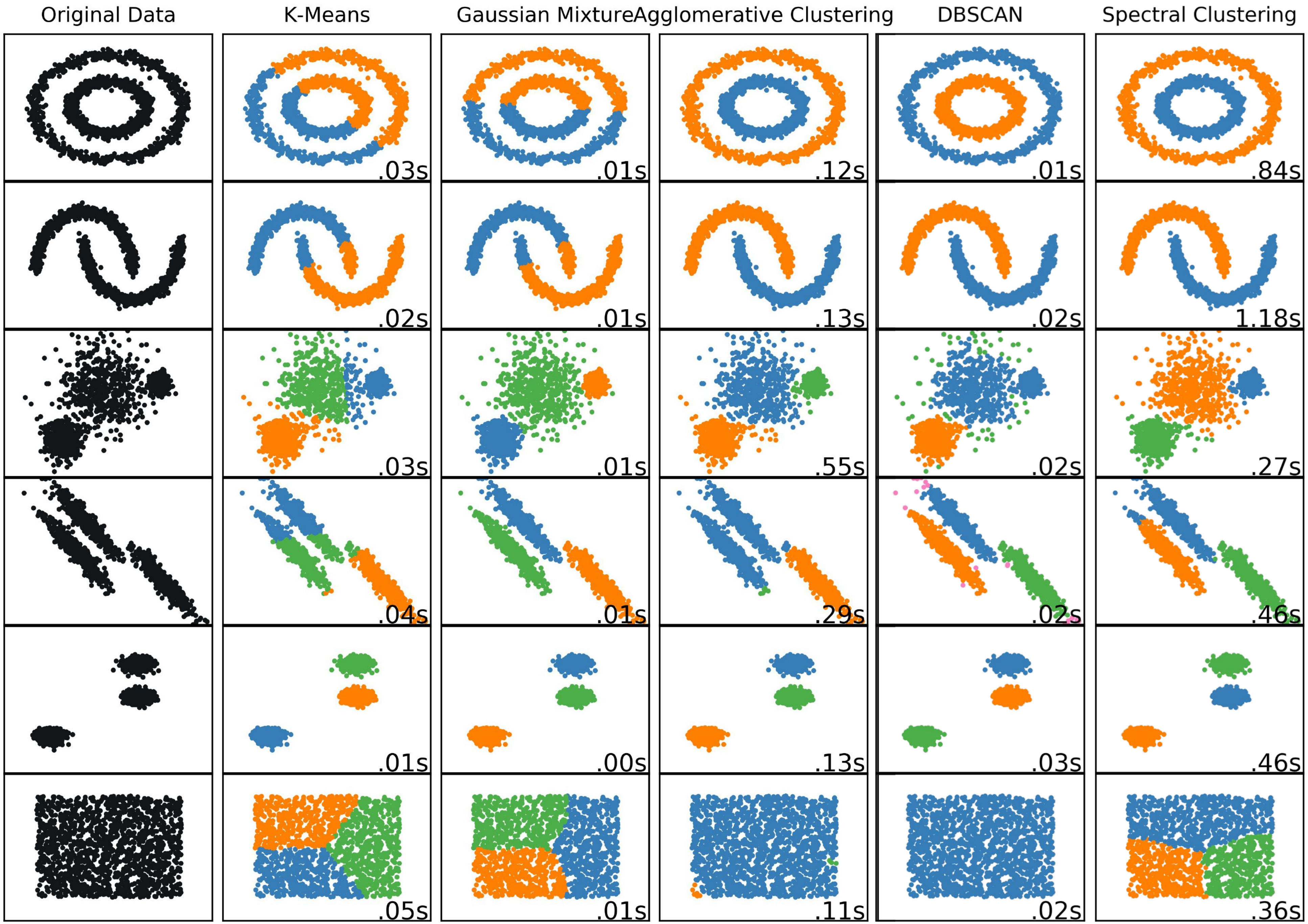
Concept from Sebastian Thrun and Peter Norvig

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

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**)
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