Mikhail Hushchyn

Clustering #2

K-Means Limitations. Hierarchical Clustering.
DBSCAN.

2021

MLHEP 2021

NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS

LAMBDA • HSE

Yandex

SCHOOL OF DATA ANALYSIS

EPFL

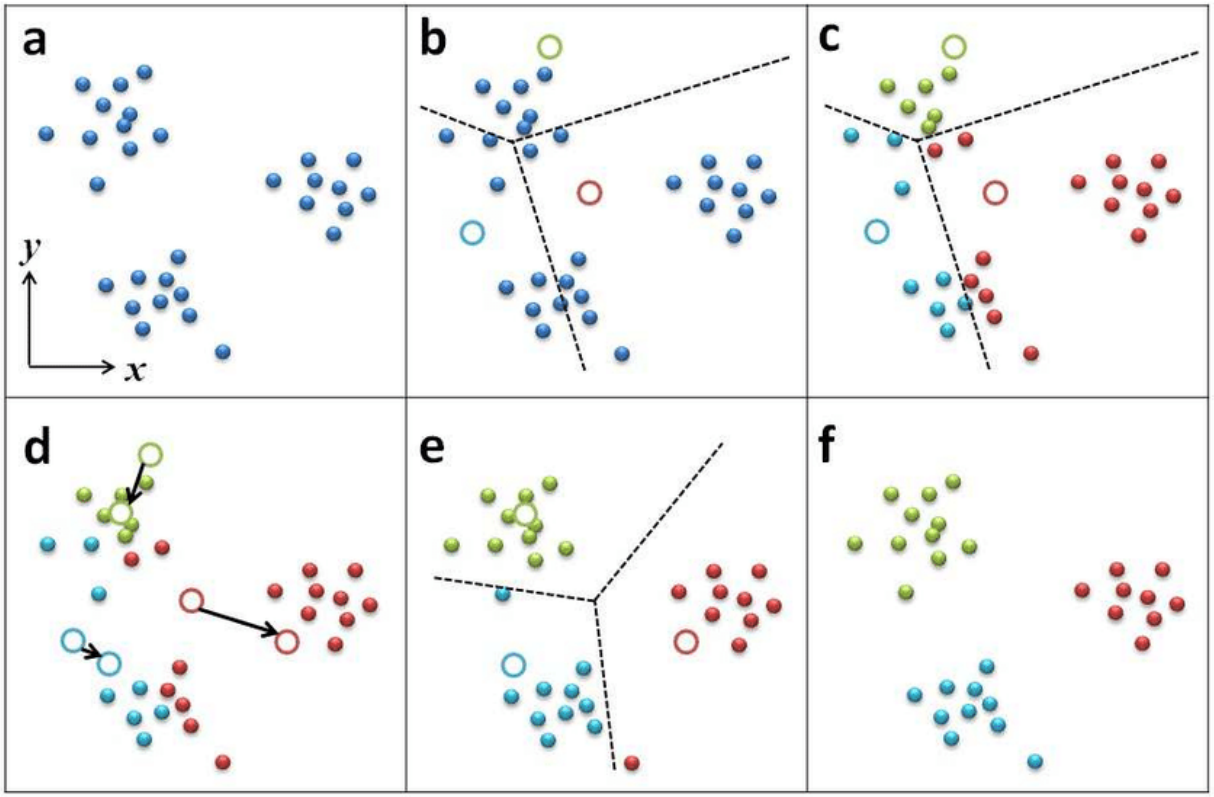
SIT Schaffhausen Institute of Technology

Outline

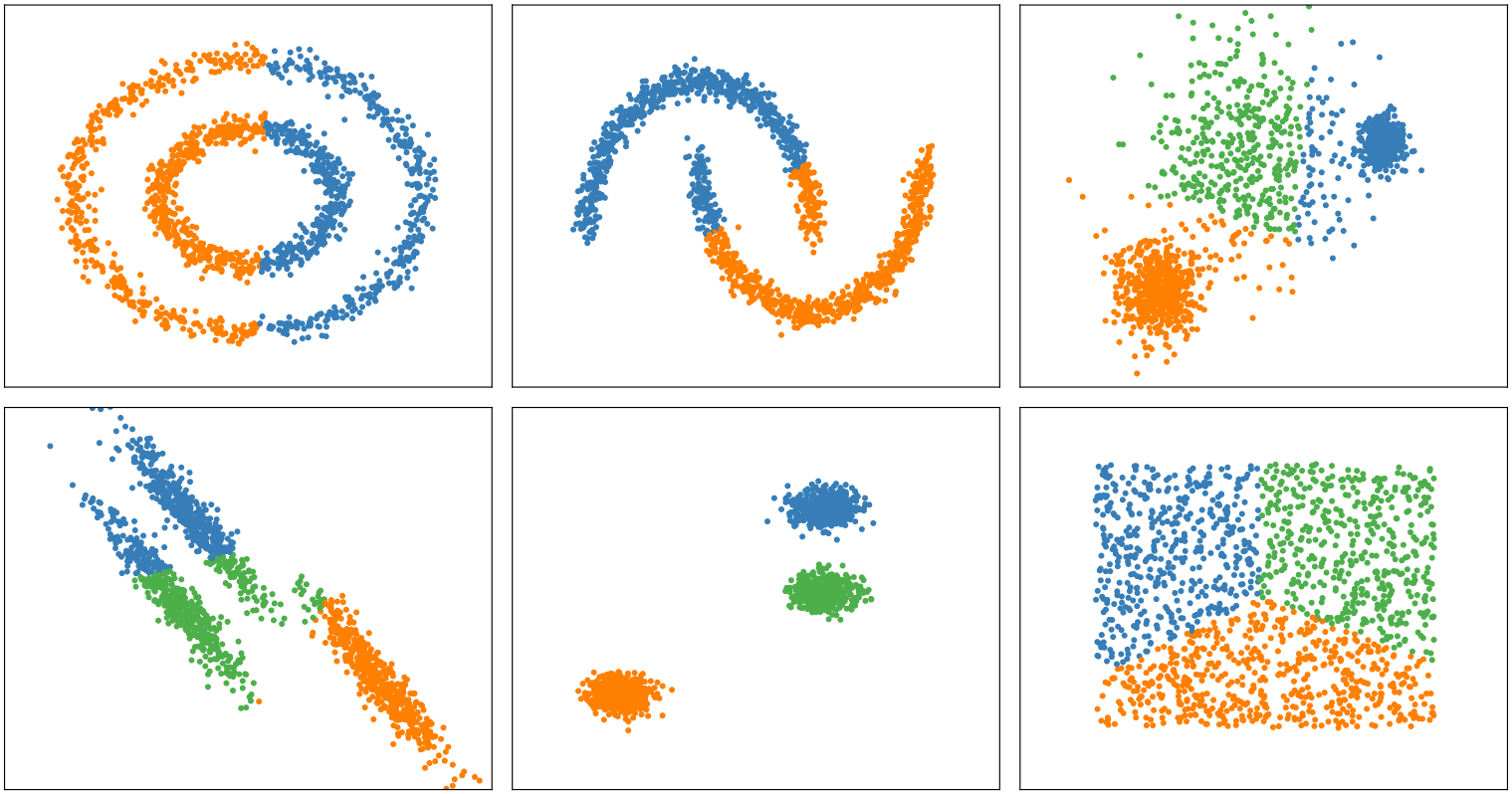
- K-Means limitations
- Hierarchical clustering
- DBSCAN

K-Means Limitations

K-Means recap



K-Means examples



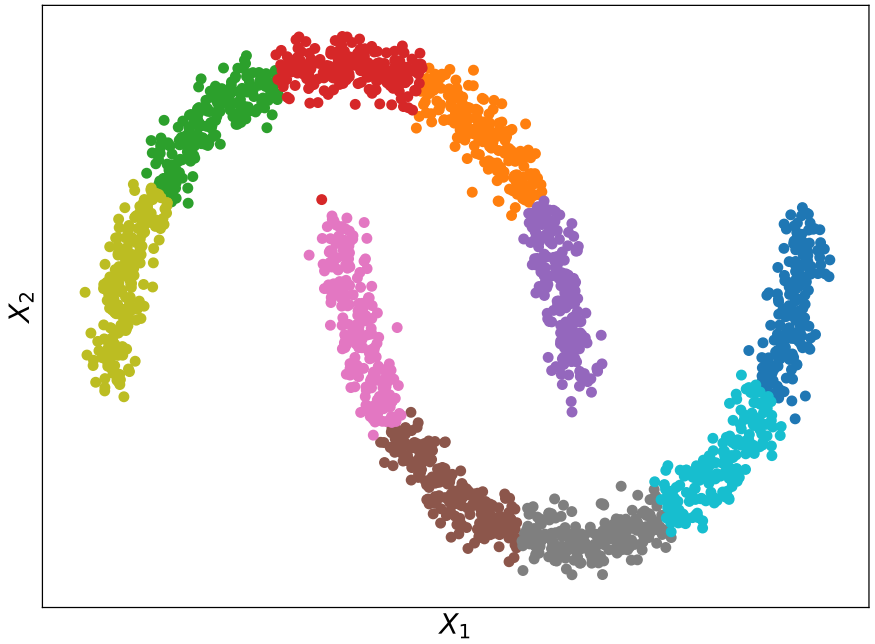
Limitations

- K-Means looks for cluster centers. All objects are separated between these centers based on the closest distances
- In result, objects of a cluster concentrated around its center
- Real clusters may have more difficult forms, like circles or moons
- They are divided by K-Means between several centers

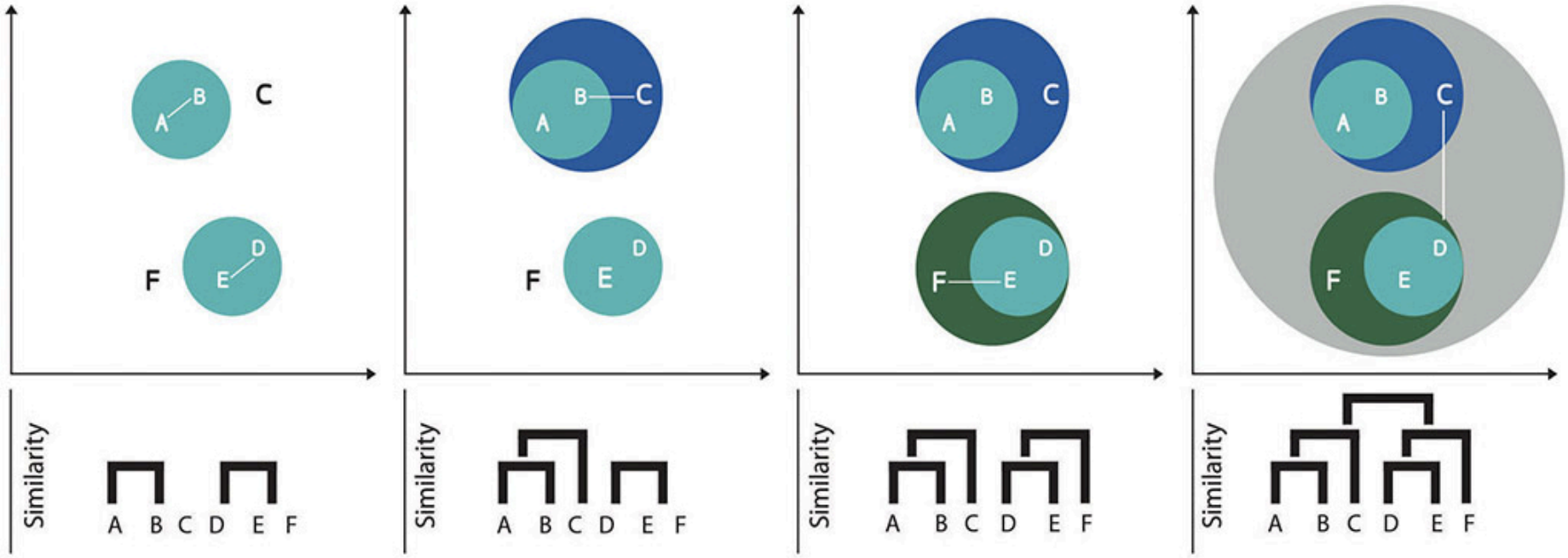
Hierarchical Clustering

Intuition

- Let’s ask K-Means to find many clusters
- Each found cluster will be inside a real cluster
- Now, let’s unite neighbor found clusters into one
- In result, we will get clusters with more complex shapes



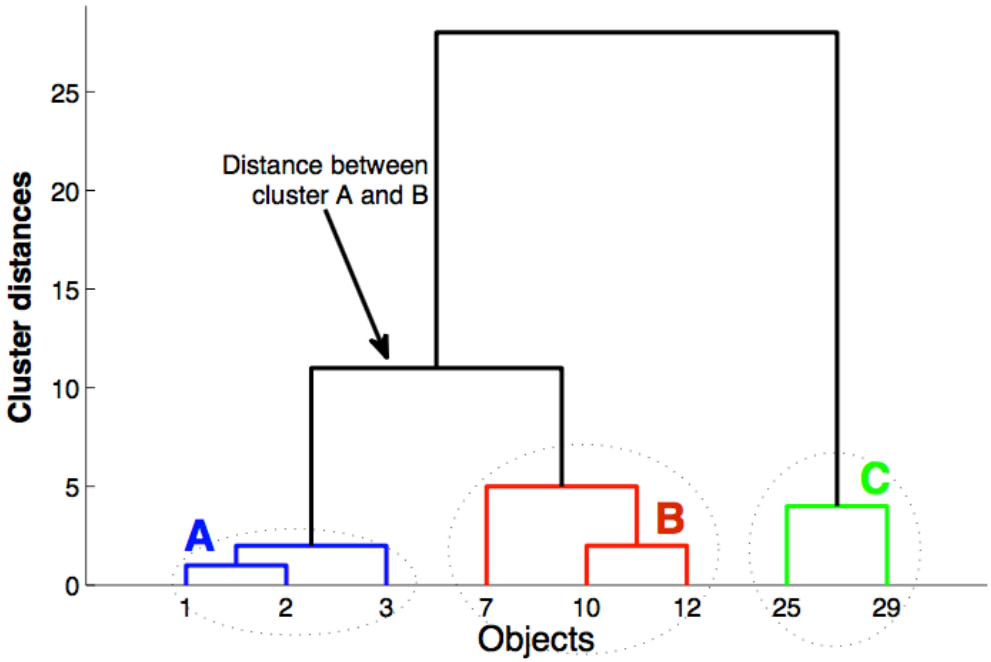
Agglomerative clustering



Link: https://www.brandidea.com/hierarchicalclustering.html

Dendrogram

- Agglomerative clustering algorithms build a dendrogram
- Dendrogram shows hierarchy of clusters in a data sample
- It contains information about objects inside each cluster and distances between these clusters



Algorithm

initialize distance matrix $M \in \mathbb{R}^{N \times N}$ between singleton clusters $\{x_1\}, \ldots \{x_N\}$

REPEAT:

1) pick closest pair of clusters i and j
2) merge clusters i and j
3) delete rows/columns i, j from M and add new row/column for merged cluster
4) recalculate distances between clusters

UNTIL 1 cluster is left

RETURN hiearchical clustering of objects

Distance between clusters #1

- Nearest neighbor (single link):

$$\rho(A, B) = \min_{a \in \mathrm{A},\, b \in B} \rho(a, b)$$

- Furthest neighbor (complete link):

$$\rho(A, B) = \max_{a \in \mathrm{A},\, b \in B} \rho(a, b)$$

where $A = \{x_{i_1}, x_{i_2}, \dots\}$ and $\mathrm{B} = \{x_{j_1}, x_{j_2}, \dots\}$ are two clusters

Distance between clusters #2

- Average (group average link):

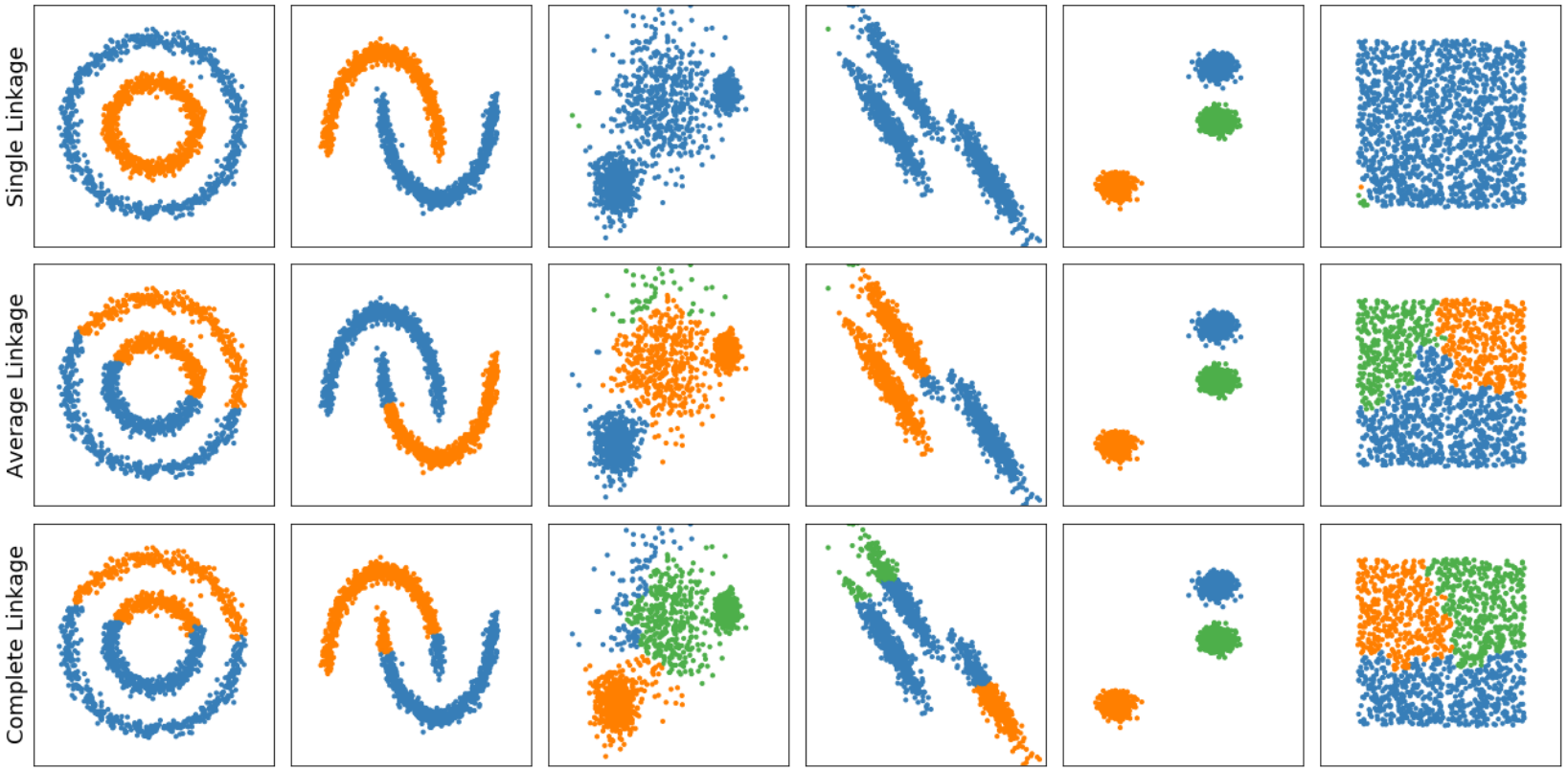
$$\rho(A, B) = \frac{1}{N_A N_B} \sum_{a \in A,\, b \in B} \rho(a, b)$$

- Closest centroid (centroid link):

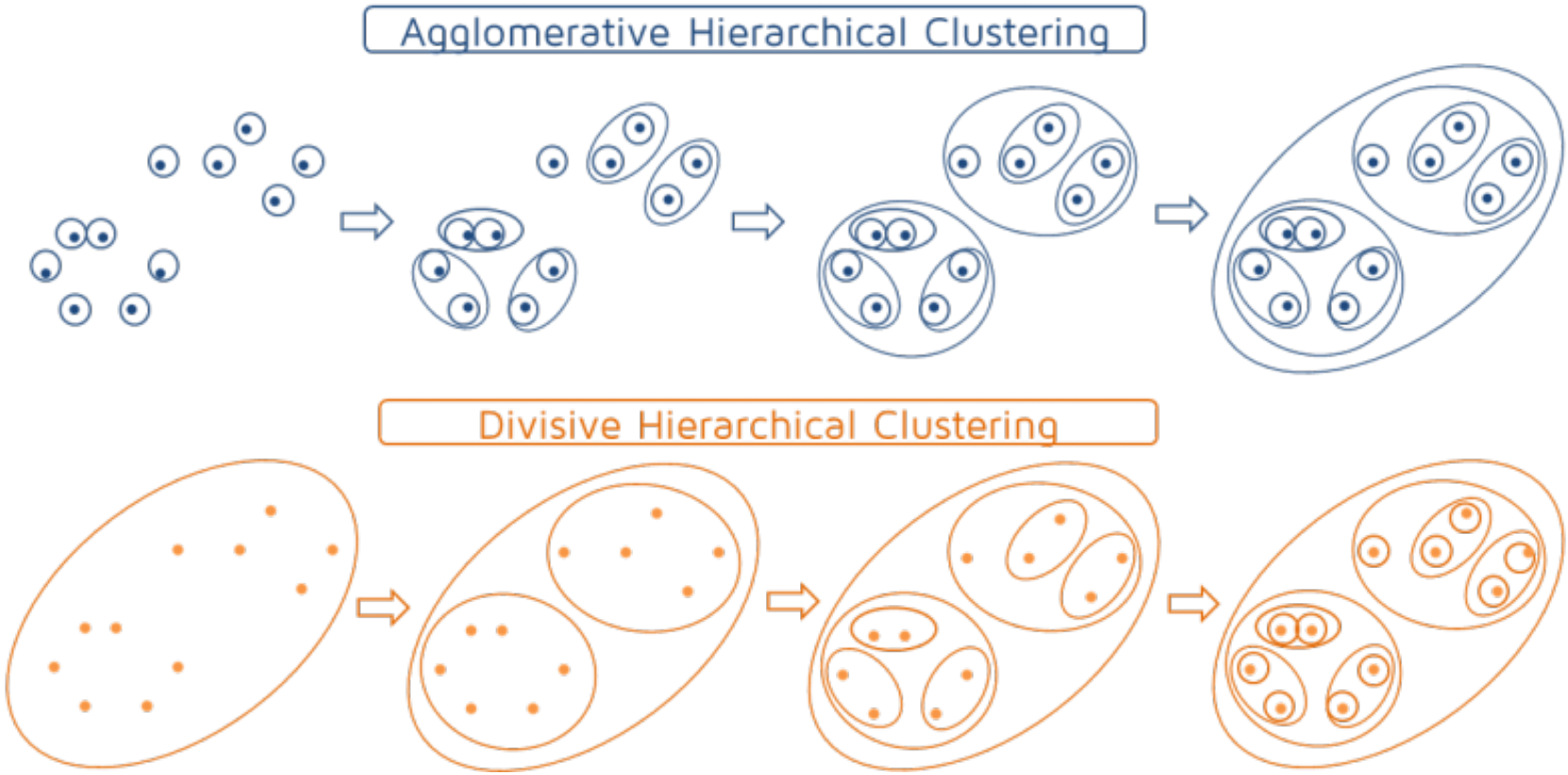
$$\rho(A, B) = \rho(\mu_A, \mu_B)$$

where μ_A and μ_B are cluster centers

Demonstration



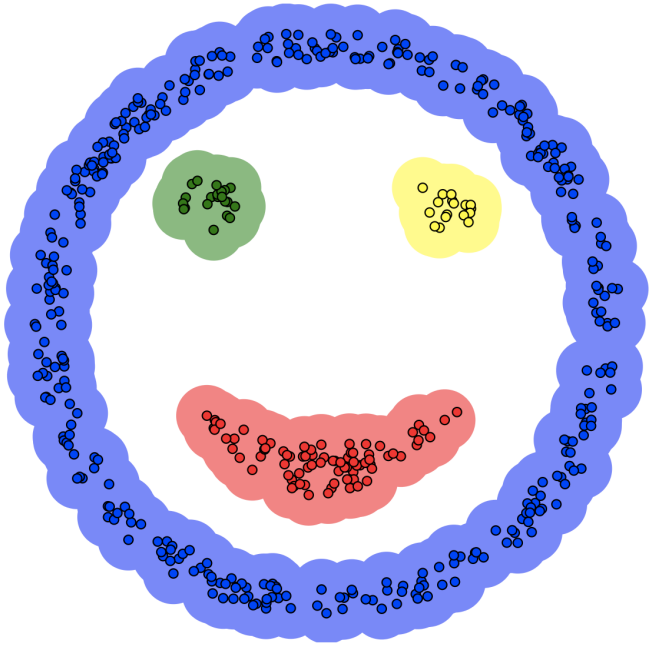
Alternative



DBSCAN

Intuition

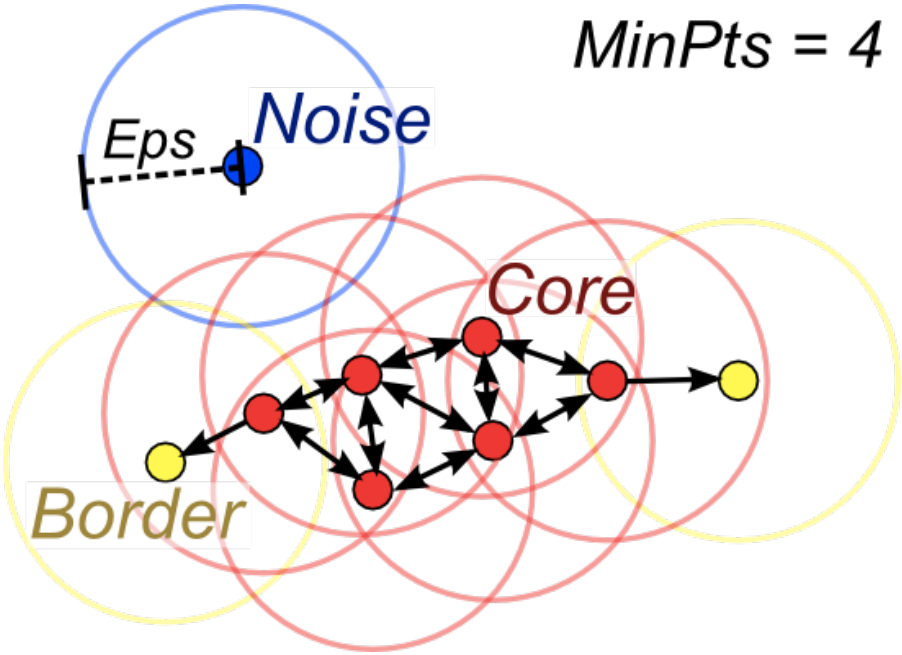
- It supposed that clusters form dense groups of objects
- Areas between the clusters are sparse, with very low densities
- Let's start from a random object and grow up a cluster by adding neighbor objects within some radius



DBSCAN idea #1

DBSCAN has two parameters:

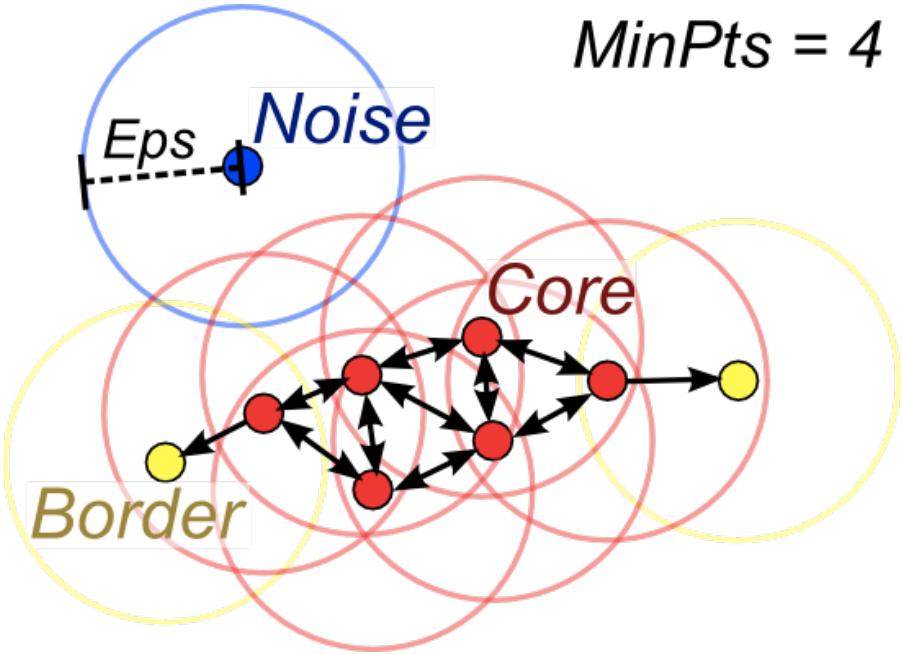
- ϵ – radius of neighborhood of each object
- **MinPts** – minimal number of objects inside the neighborhood



DBSCAN idea #2

Three types of objects:

- **Core:** has $\geq$ MinPts objects within its ϵ neighborhood
- **Border:** not core object, has at least 1 core object within its ϵ neighborhood
- **Noise:** neither a core nor a border point



Algorithm (short)

Algorithm 1: DBSCAN algorithm.

1: Label all objects as core, border, or noise objects.
2: Eliminate noise objects.
3: Put an edge between all core objects that are within ϵ of each other.
4: Make each group of connected core objects into a separate cluster.
5: Assign each border object to one of the clusters of its associated core objects.

Algorithm (detailed)

```
1.function dbscan(X, eps, min_pts):
2.  initialize NV = X # not visited objects
3.  for x in NV:
4.      remove(NV, x) # mark as visited
5.      nbr = neighbours(x, eps) # set of neighbours
6.      if nbr.size < min_pts:
7.          mark_as_noise(x)
8.      else:
9.          C = new_cluster()
10.         expand_cluster(x, nbr, C, eps, min_pts, NV)
11.         yield C
```

Link: https://shestakoff.github.io/hse_se_ml/2020/l14-cluster/lecture-clust.slides#/4/5

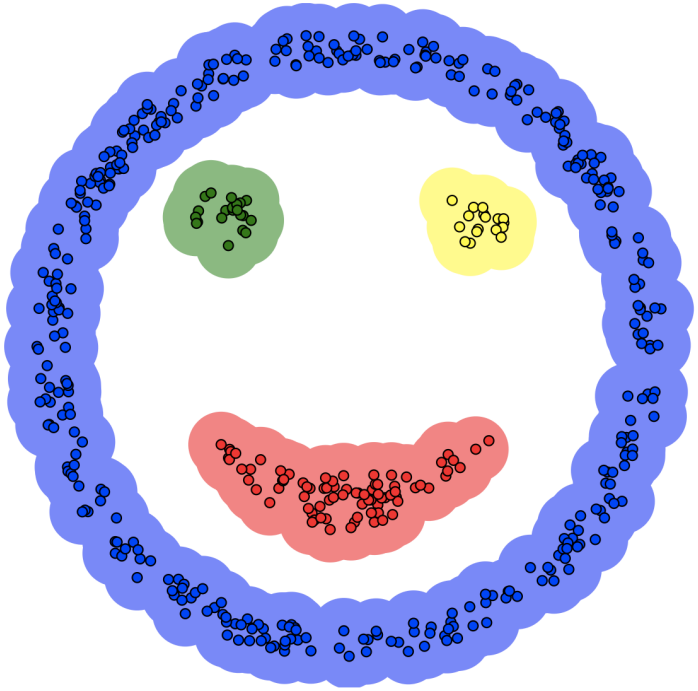
Algorithm (detailed)

```
1. function expand_cluster(x, nbr, C, eps, min_pts, NV):
2.  add(x, C)
3.  for x1 in nbr:
4.      if x1 in NV: # object not visited
5.          remove(NV, x1) # mark as visited
6.          nbr1 = neighbours(x1, eps)
7.          if nbr1.size >= min_pts:
8.              # join sets of neighbours
9.              merge(nbr, nbr_1)
10.     if x1 not in any cluster:
11.         add(x1, C)
```

Link: https://shestakoff.github.io/hse_se_ml/2020/l14-cluster/lecture-clust.slides#/4/5

Demonstration

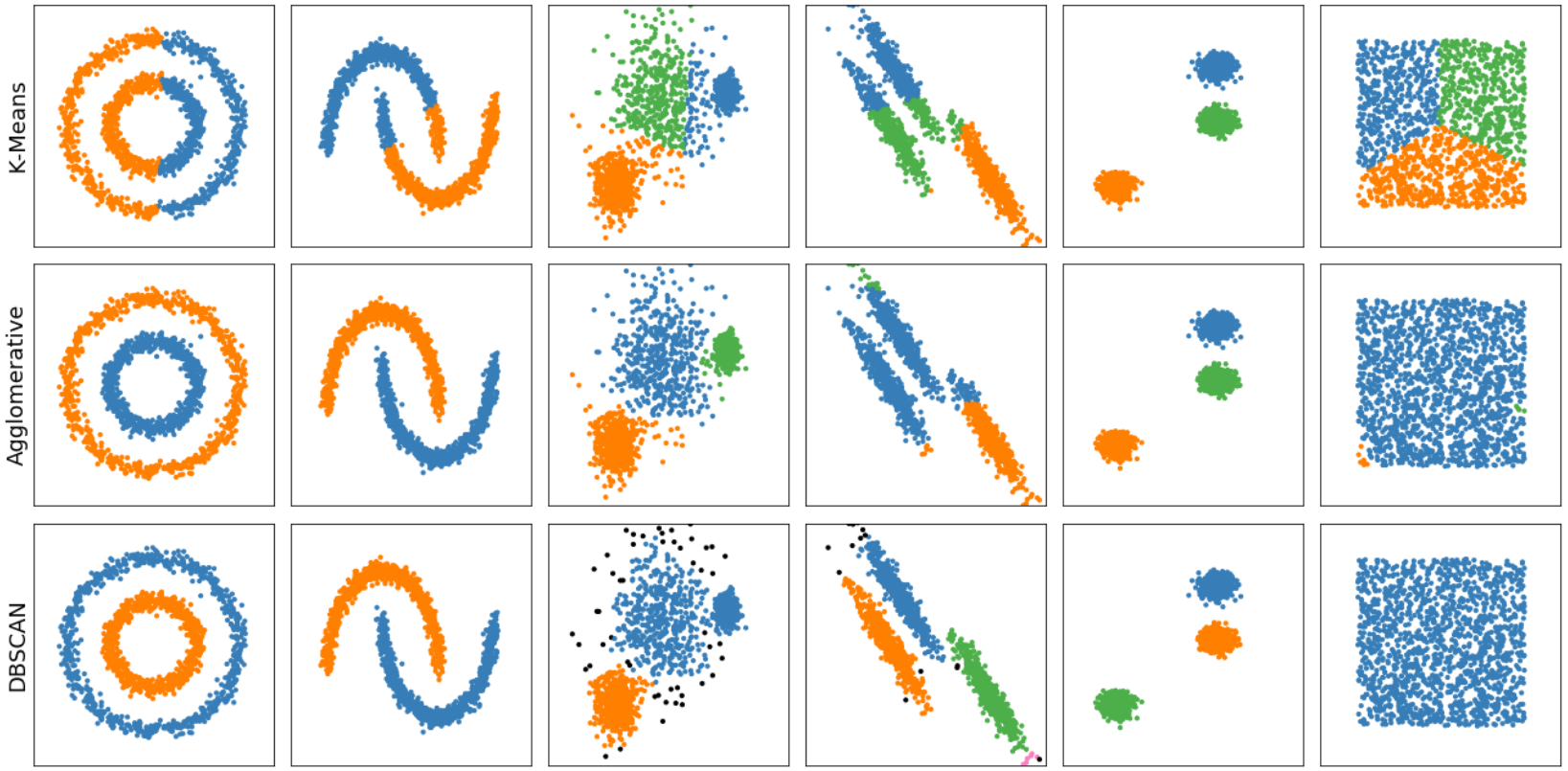
Demo: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/



Properties

- Number of clusters is estimated automatically
- Robust to outliers. They are recognized as a noise
- Can find clusters with complex shapes
- Sensitive to objects density variations

Overview



Summary

Summary

- K-Means limitations
  - Find clusters that concentrated only around one center
  - Can not find clusters with complex forms
- Hierarchical clustering
  - Agglomerative clustering
  - Dendrogram contains information about clusters structure
- DBSCAN
  - Density-Based Spatial Clustering of Applications with Noise
  - Able to find complex clusters and outliers