Artificial Intelligence

Clustering 2

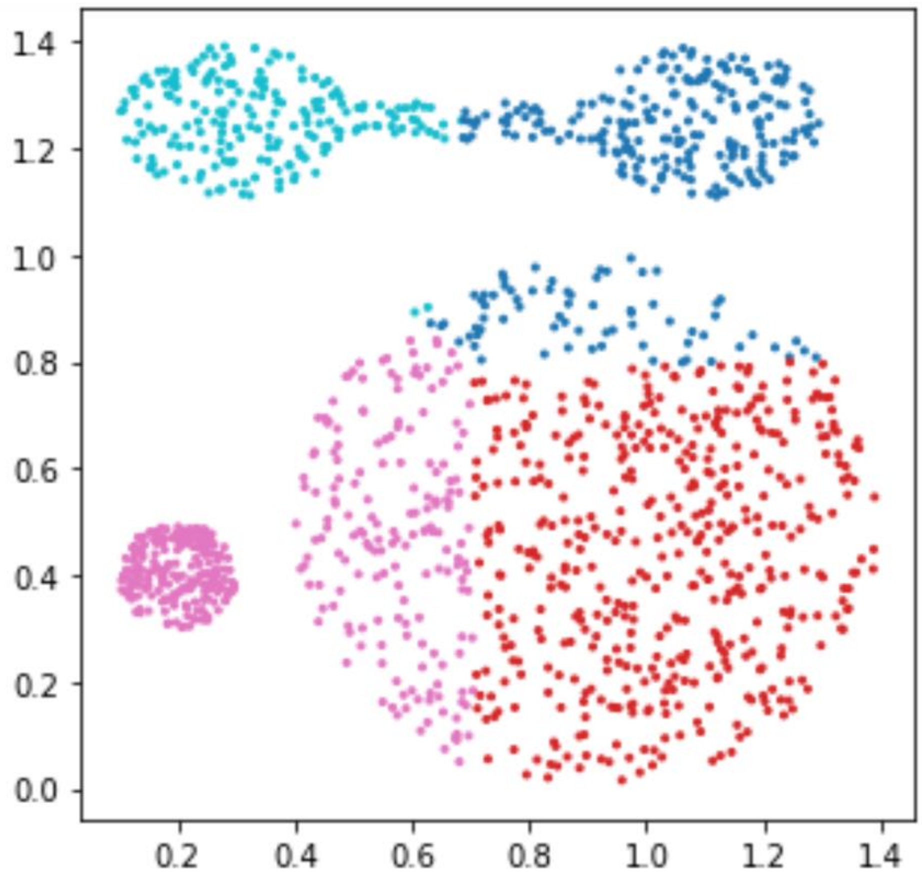
Extended from Kyuseok Shim's slides

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING
인공지능학과
Department of Artificial Intelligence

정 우 환 (whjung@hanyang.ac.kr)

Fall 2022

K-means clustering

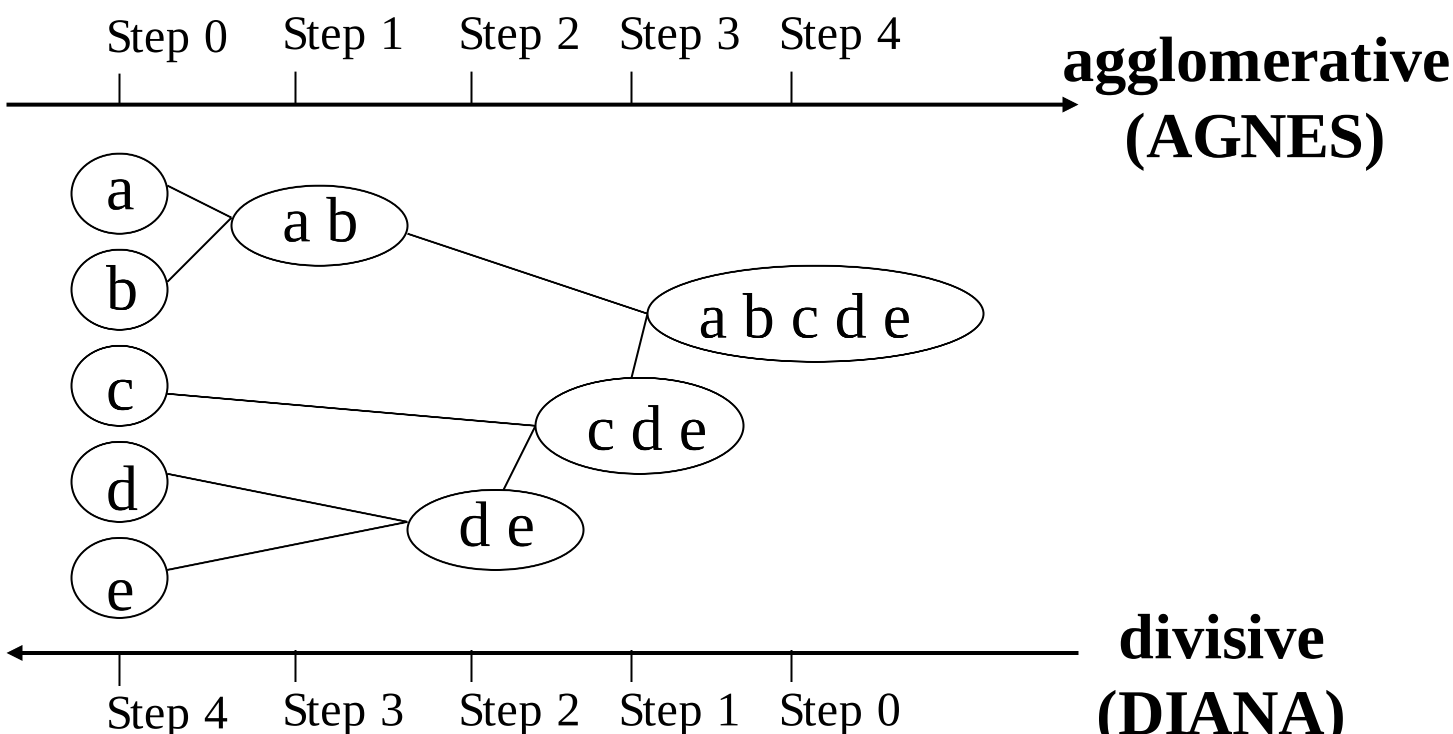


Hierarchical Clustering Approach

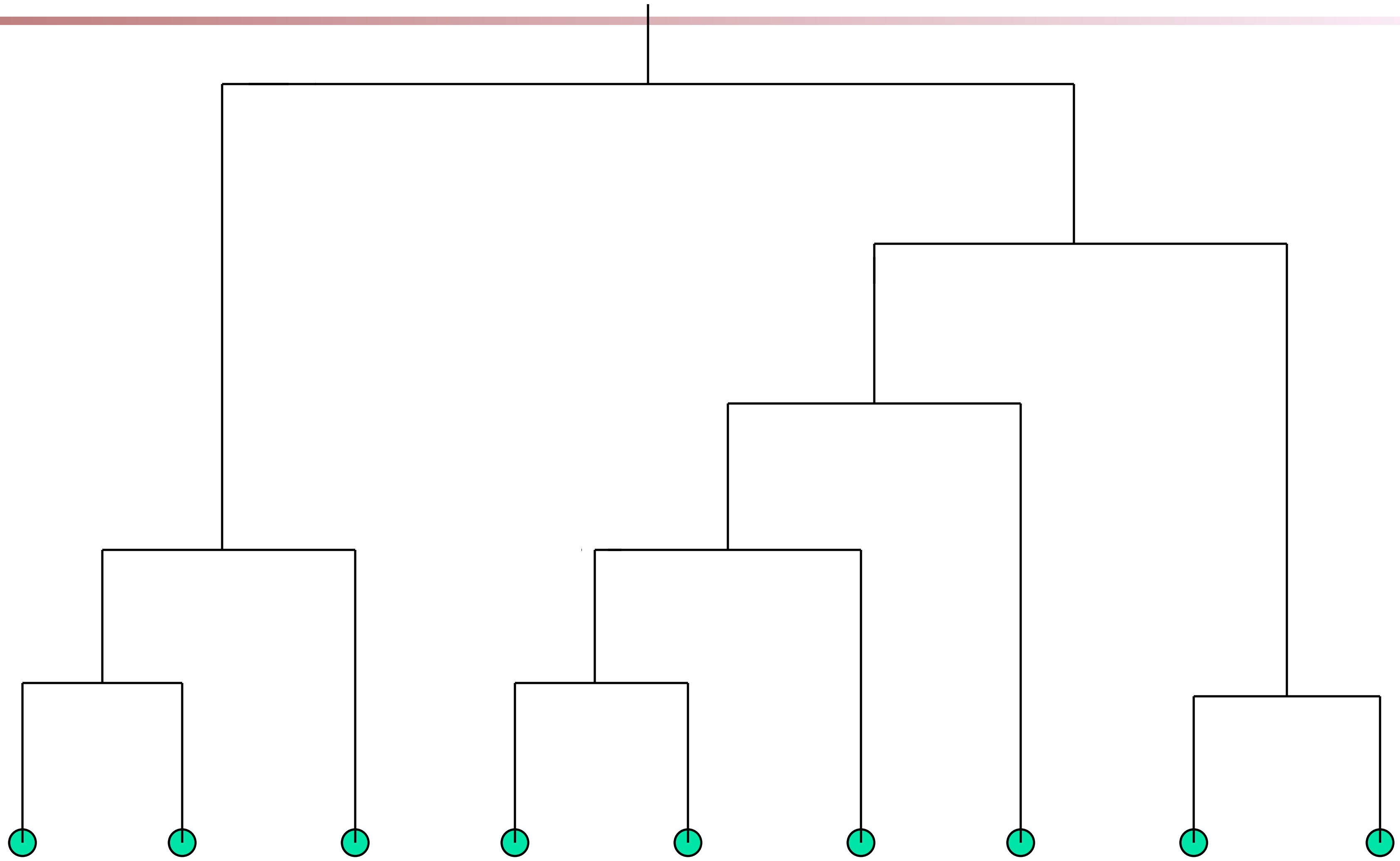
- Given *k*, the *hierarchical* algorithm is implemented in four steps:
  - Say "Every point is it's own cluster"
  - Find “most similar” pair of clusters
  - Merge it into a parent cluster
  - Repeat…until you've merged the whole dataset into k clusters

Hierarchical Clustering

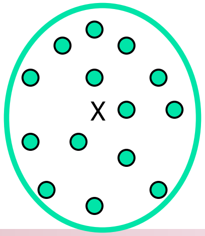
- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition

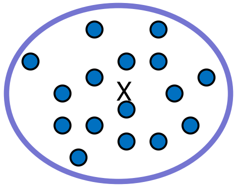


Dendrogram: Shows How Clusters are Merged



Distance between Clusters





- Single link: smallest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = min(t_{ip}, t_{jq})$
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = max(t_{ip}, t_{jq})$
- Average: avg distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = avg(t_{ip}, t_{jq})$
- Centroid: distance between the centroids of two clusters, i.e., $dist(K_i, K_j) = dist(C_i, C_j)$
- Medoid: distance between the medoids of two clusters, i.e., $dist(K_i, K_j) = dist(M_i, M_j)$
  - Medoid: a chosen, centrally located object in the cluster

Centroid, Radius and Diameter of a Cluster (for numerical data sets)

- Centroid: the "middle" of a cluster

$$C_m = \frac{\sum_{i=1}^{N} (t_{ip})}{N}$$

- Radius: square root of average distance from any point of the cluster to its centroid

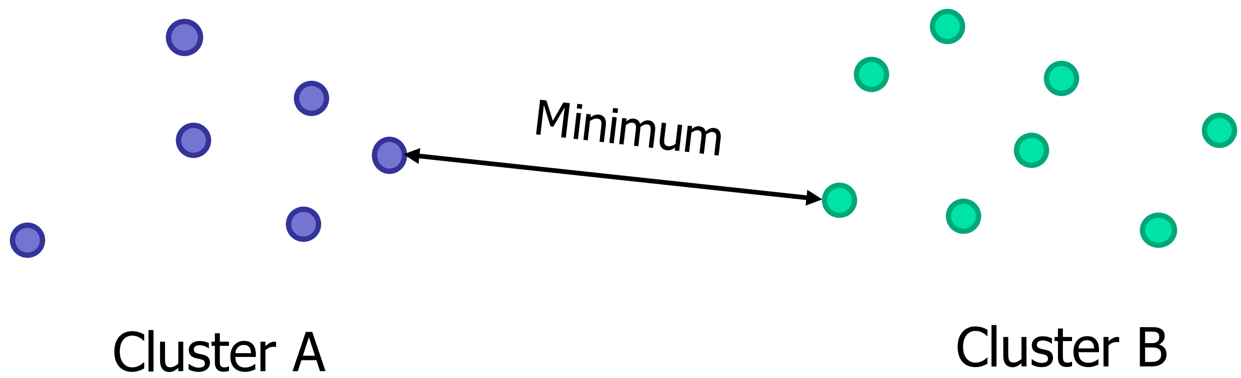
$$R_m = \sqrt{\frac{\sum_{i=1}^{N} (t_{ip} - c_m)^2}{N}}$$

- Diameter: square root of average mean squared distance between all pairs of points in the cluster

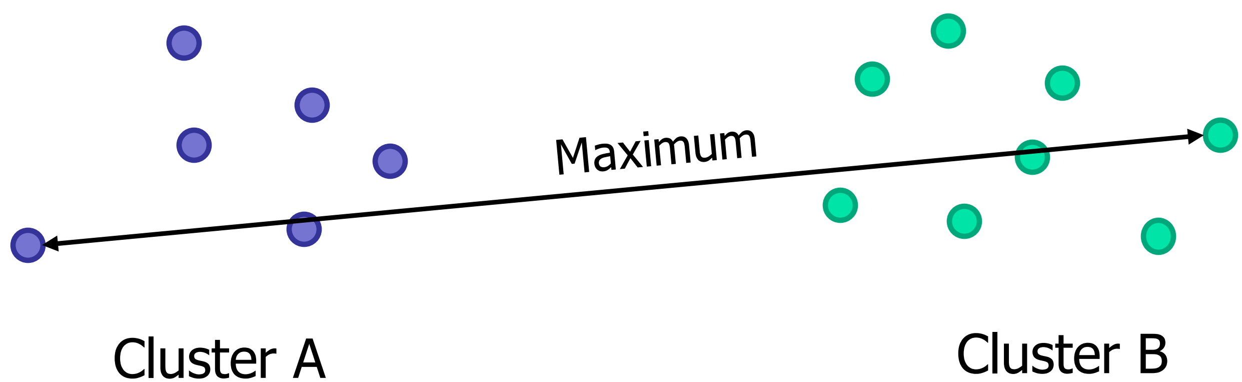
$$D_m = \sqrt{\frac{\sum_{i=1}^{N} \sum_{i=1}^{N} (t_{ip} - t_{iq})^2}{N(N-1)}}$$

Hierarchical Algorithms

- Single-link

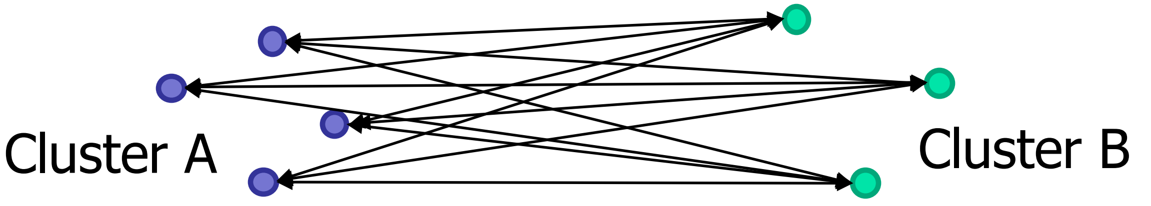


- Complete-link

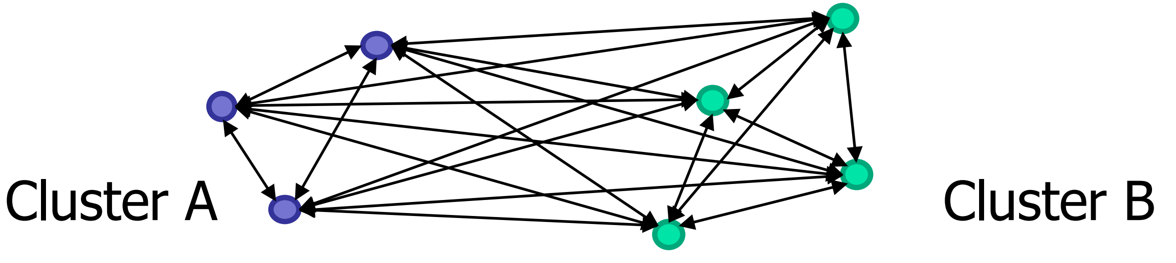


Hierarchical Algorithms

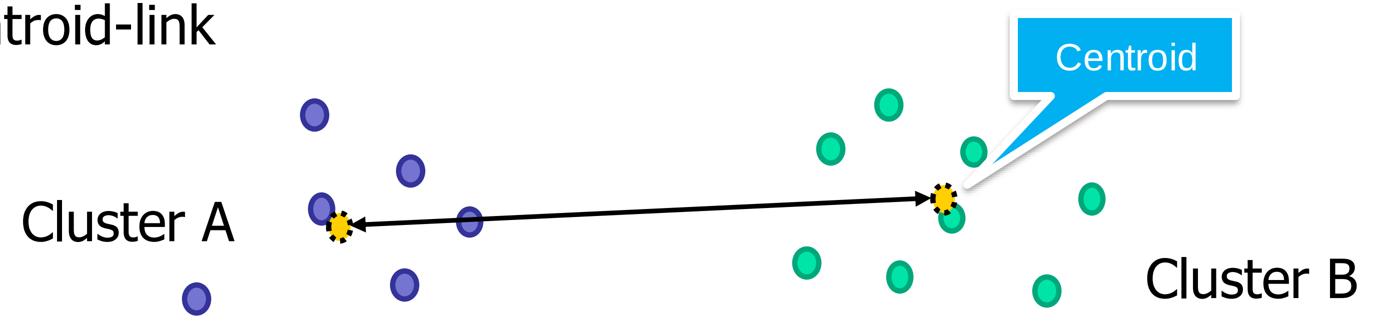
- Average-link



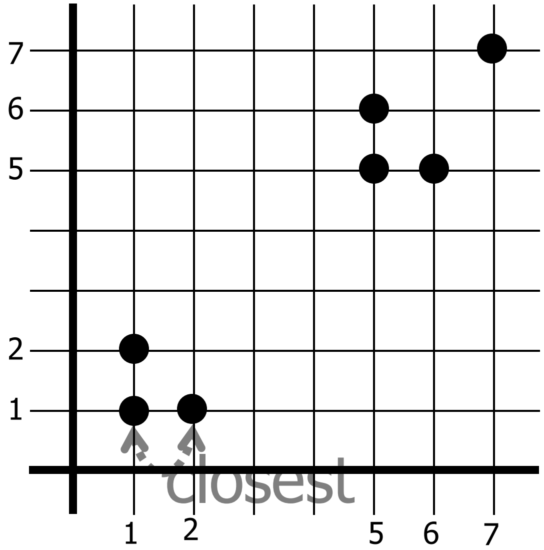
- Mean-link



- Centroid-link

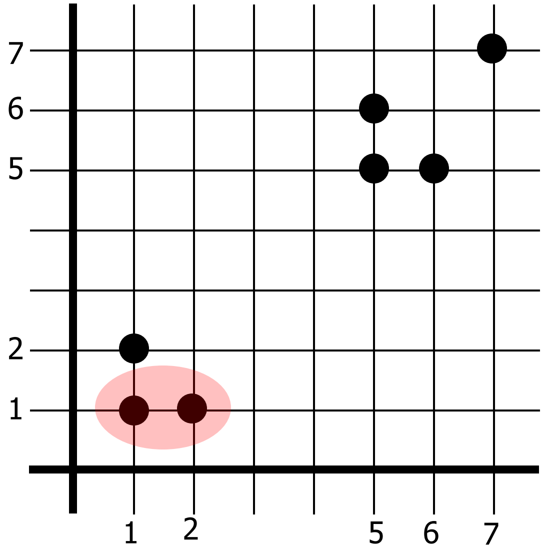


An Example of Single-Link Clustering Algorithm



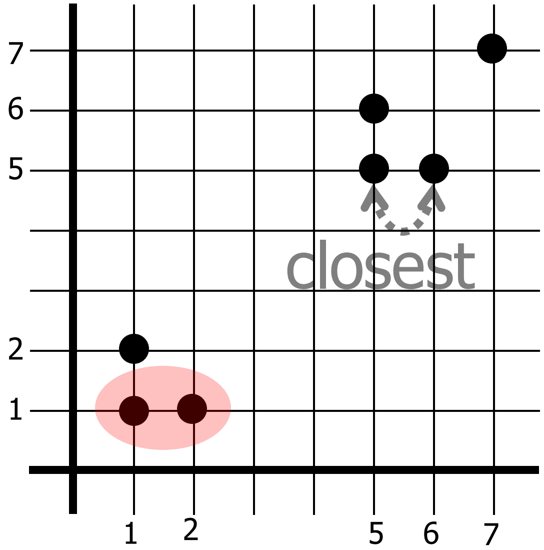
(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

An Example of Single-Link Clustering Algorithm



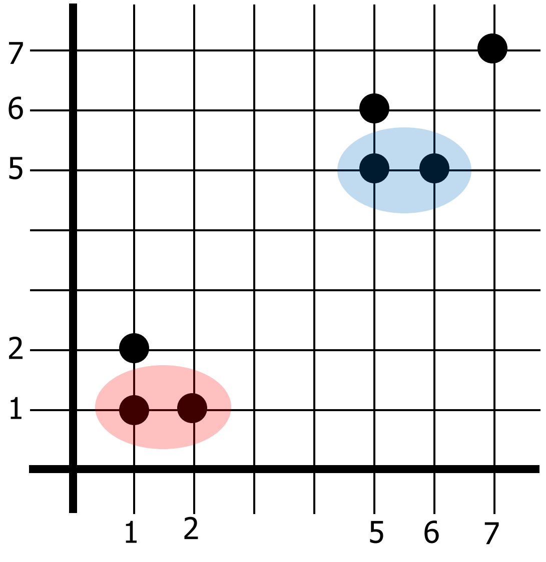
(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

An Example of Single-Link Clustering Algorithm



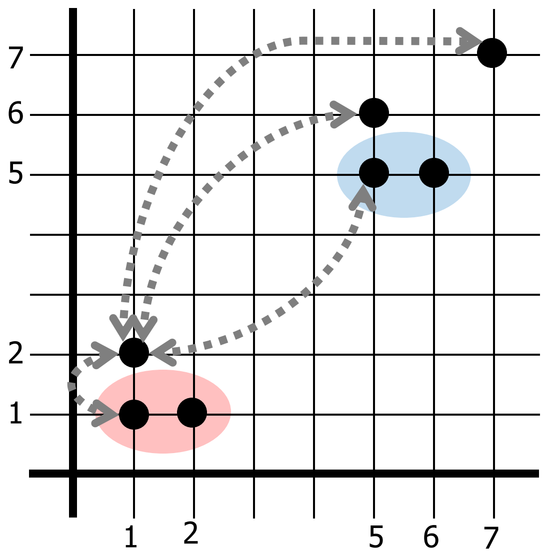
(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

An Example of Single-Link Clustering Algorithm



(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

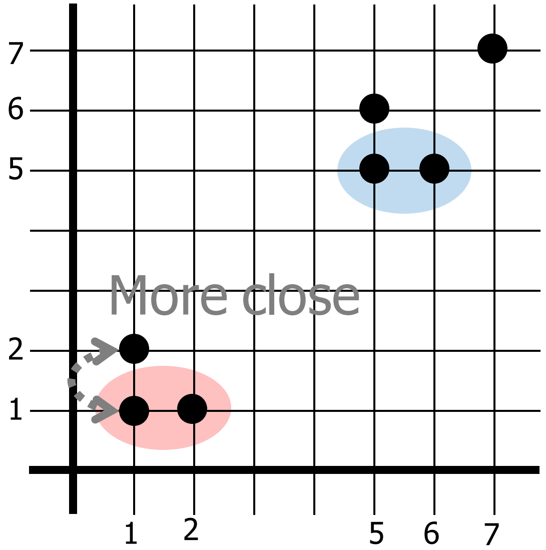
An Example of Single-Link Clustering Algorithm



Compare each distance from closest item in clusters

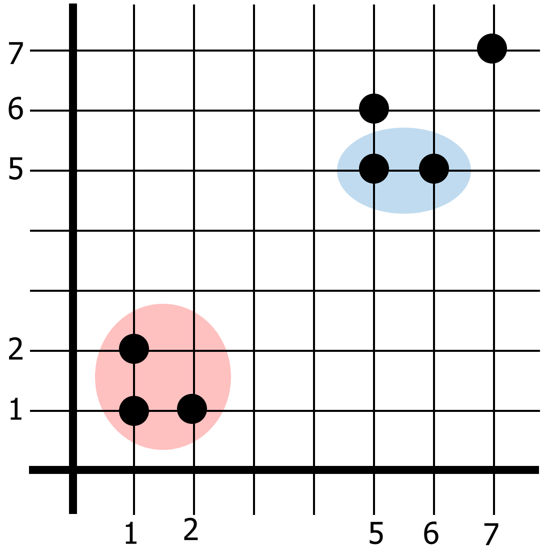
(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

An Example of Single-Link Clustering Algorithm



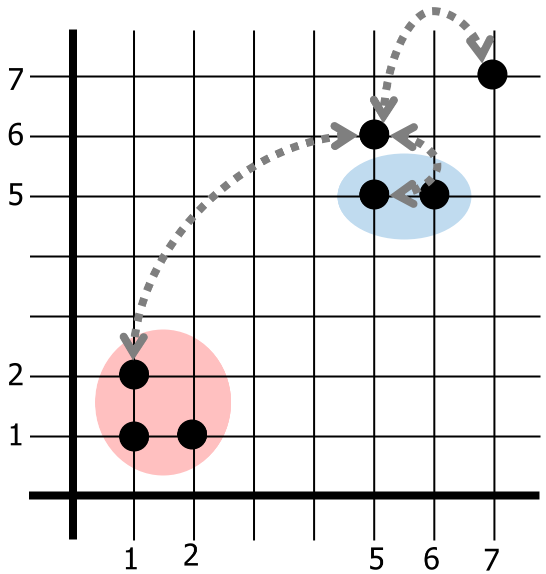
(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

An Example of Single-Link Clustering Algorithm



(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

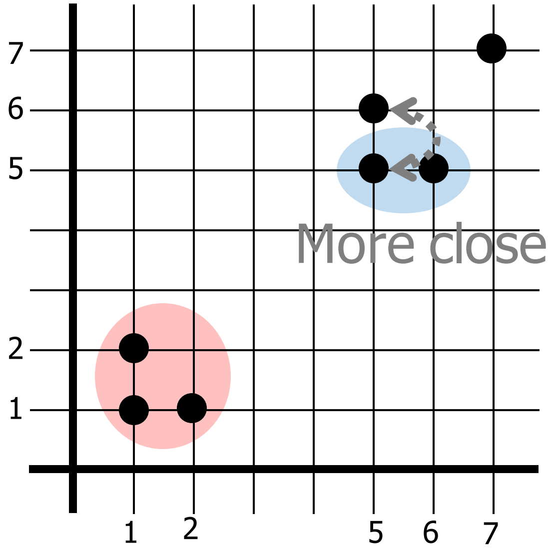
An Example of Single-Link Clustering Algorithm



(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

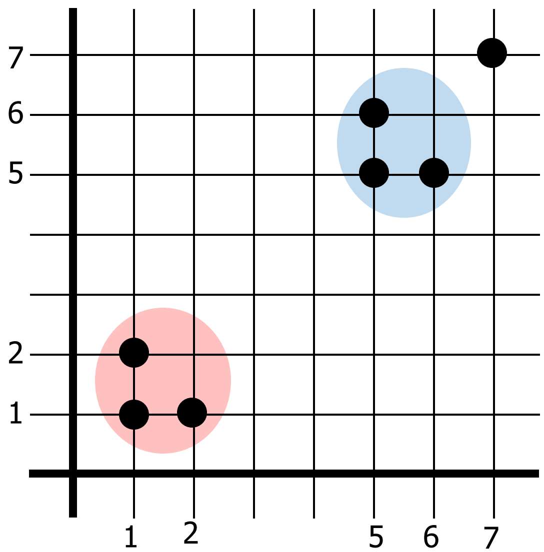
Compare each distance from closest item in clusters

An Example of Single-Link Clustering Algorithm



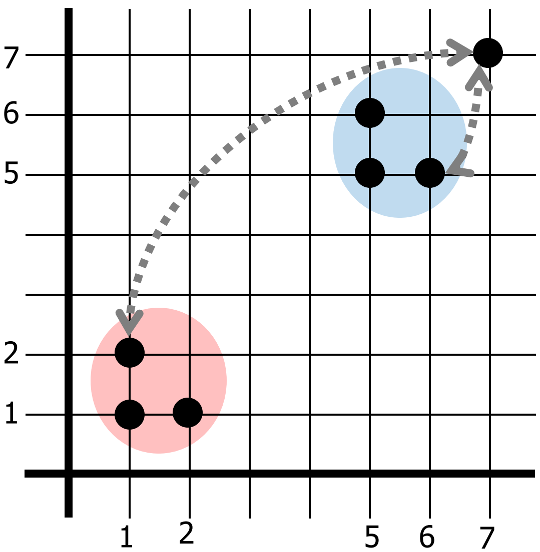
(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

An Example of Single-Link Clustering Algorithm



(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

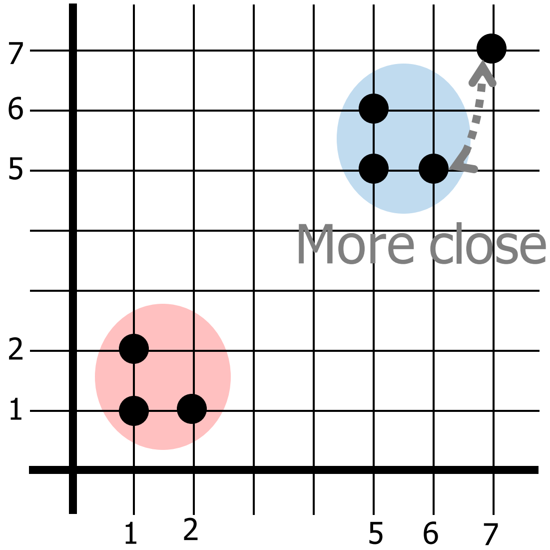
An Example of Single-Link Clustering Algorithm



(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

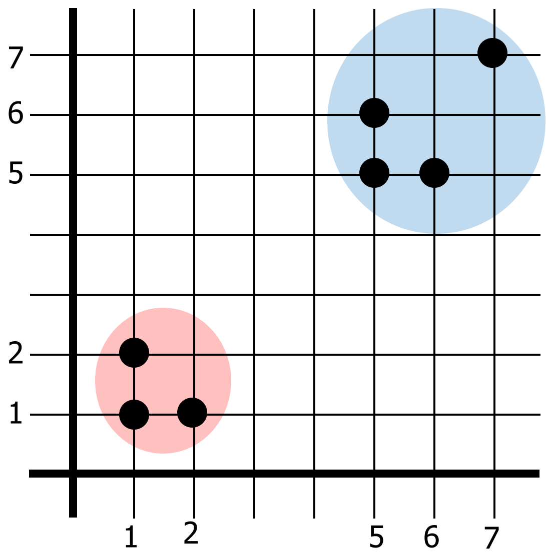
Compare each distance from closest item in clusters

An Example of Single-Link Clustering Algorithm



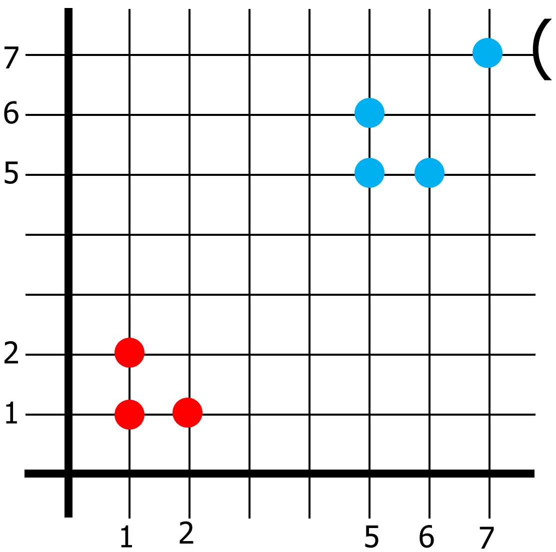
(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

An Example of Single-Link Clustering Algorithm



(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

An Example of Single-Link Clustering Algorithm



(1, 1) (2, 1) (1, 2) (5, 5) (6, 5) (5, 6) (7, 7)

Class A	Class B
(1, 1) (2, 1) (1, 2)	(5, 5) (6, 5) (5, 6) (7, 7)

Python – Hierarchical Clustering

Enumerate function

The iteration index is stored in it

```
for i, x in enumerate(('A', 'B', 'C')):
    print("i: {}, x: {}".format(i, x))
```

```
i: 0, x: A
i: 1, x: B
i: 2, x: C
```

Hierarchical Clustering

```python
from sklearn.cluster import AgglomerativeClustering
for i, linkage in enumerate(('single', 'complete')):
    clustering = AgglomerativeClustering(
        linkage=linkage, n_clusters=4)
    y_pred = clustering.fit_predict(X)
    plt.figure(i + 1, figsize=(5, 5))
    plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=4, cmap=cmap)
    plt.title(linkage)
plt.show()
```

Hierarchical Clustering

Link Types

```
from sklearn.cluster import AgglomerativeClustering
for i, linkage in enumerate(('single', 'complete')):
    clustering = AgglomerativeClustering(
        linkage=linkage, n_clusters=4)
    y_pred = clustering.fit_predict(X)
    plt.figure(i + 1, figsize=(5, 5))
    plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=4, cmap=cmap)
    plt.title(linkage)
plt.show()
```

Hierarchical Clustering

Iteration Index

```
from sklearn.cluster import AgglomerativeClustering
for i, linkage in enumerate(('single', 'complete')):
    clustering = AgglomerativeClustering(
        linkage=linkage, n_clusters=4)
    y_pred = clustering.fit_predict(X)
    plt.figure(i + 1, figsize=(5, 5))
    plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=4, cmap=cmap)
    plt.title(linkage)
plt.show()
```

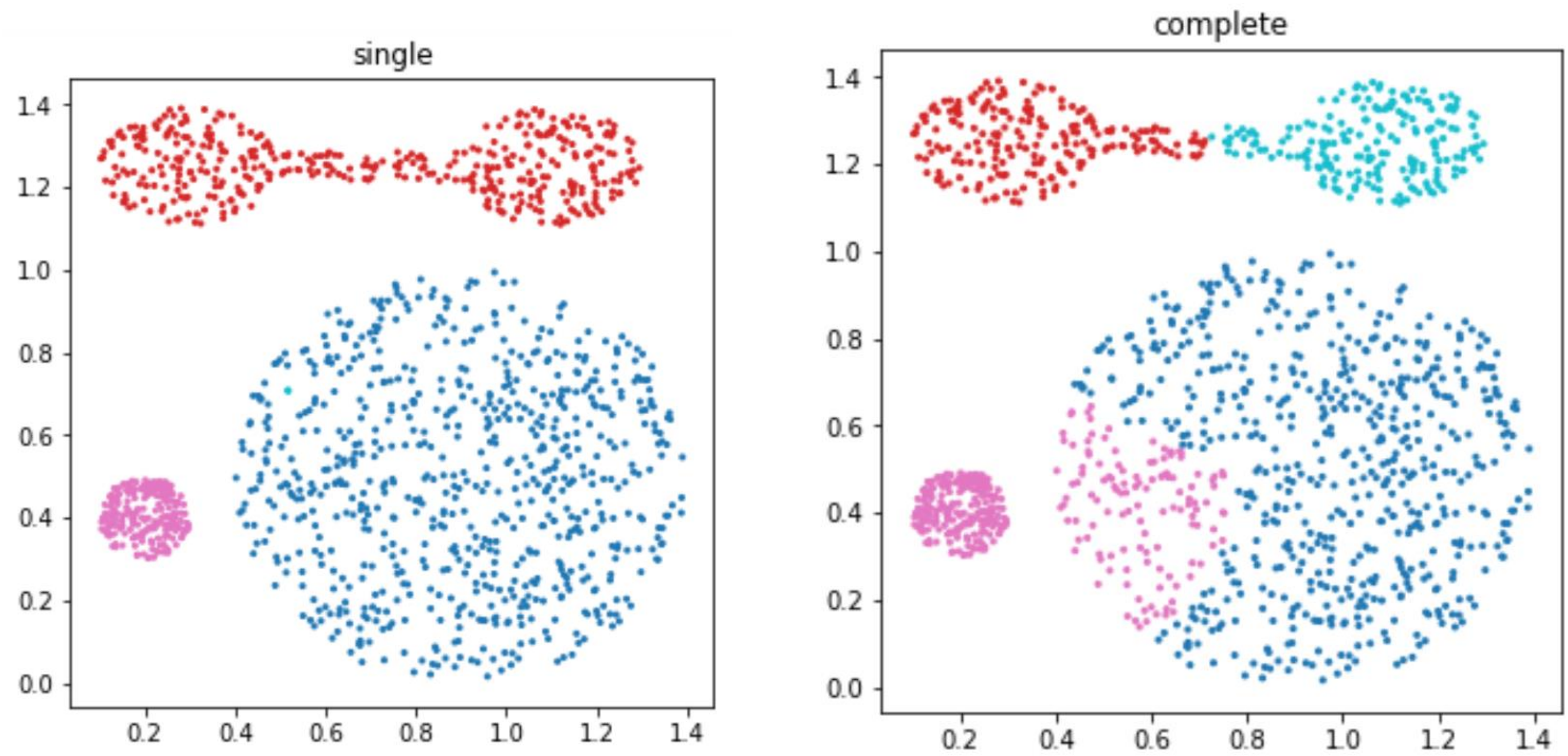
Hierarchical Clustering

```
from sklearn.cluster import AgglomerativeClustering
for i, linkage in enumerate(('single', 'complete')):
    clustering = AgglomerativeClustering(
        linkage=linkage, n_clusters=4)
    y_pred = clustering.fit_predict(X)
    plt.figure(i + 1, figsize=(5, 5))
    plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=4, cmap=cmap)
    plt.title(linkage)
plt.show()
```

The index of each figure

The title of figure is named by the link type

Hierarchical Clustering



Parameters

```
clustering = AgglomerativeClustering(
    linkage=linkage, n_clusters=4)
```

- `n_clusters`: the number of clusters to find
- `linkage`: the link type
  - Single
  - Complete
  - Average
  - Ward : minimize the variance of distances
- `affinity`: the distance metric
  - Default: `"euclidean"`

Clustering

Summary of Drawbacks of Traditional Methods

- Partitional algorithms split large clusters
- Centroid-based method splits large and non-hyperspherical clusters
  - Centers of subclusters can be far apart
- Single-link clustering algorithm is sensitive to outliers and slight change in position
  - Exhibits chaining effect on string of outliers
- Cannot scale up for large databases

DENSITY-BASED CLUSTERING

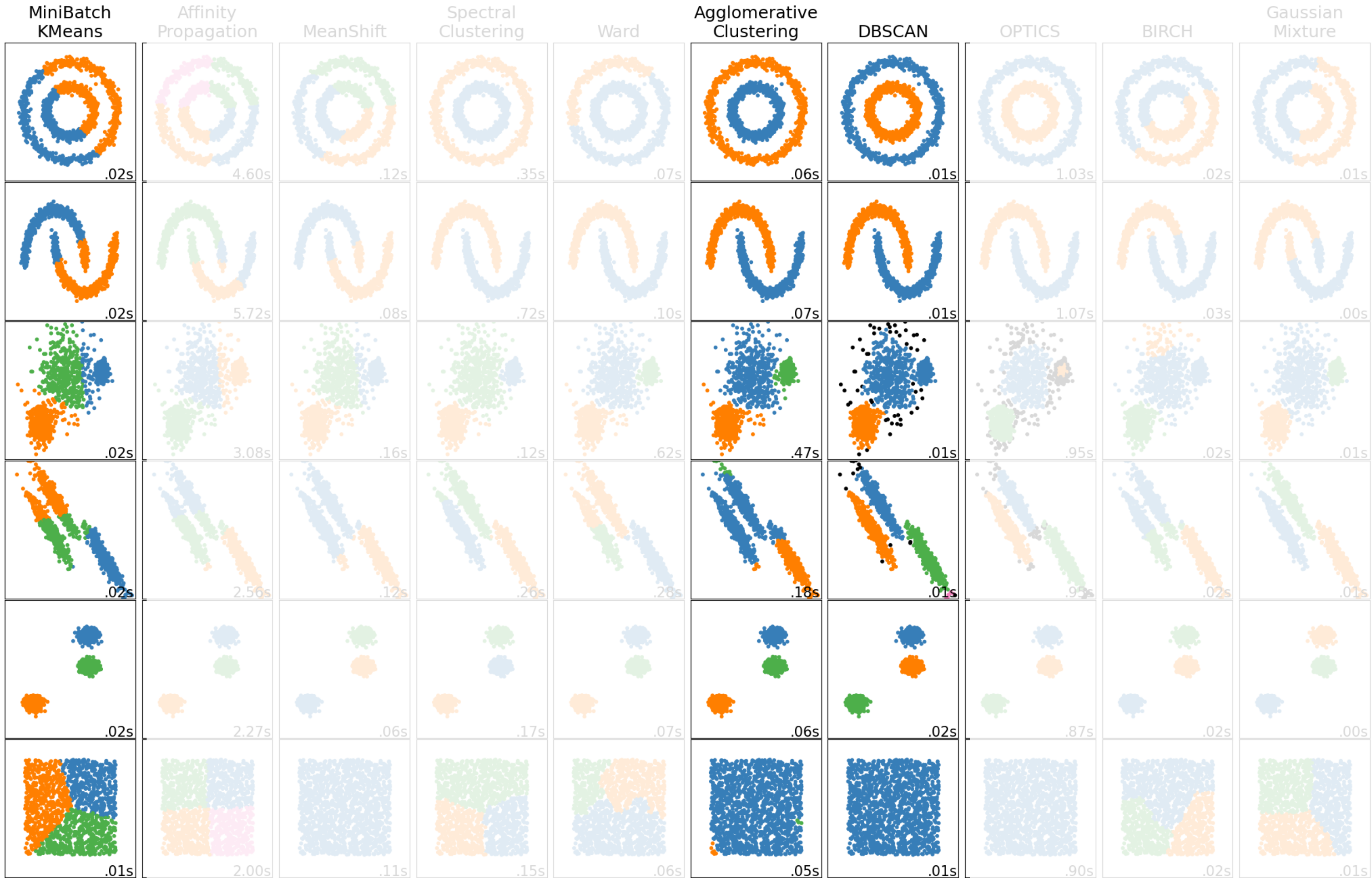
Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

DBSCAN

- http://primo.ai/index.php?title=Density-Based_Spatial_Clustering_of_Applications_with_Noise_(DBSCAN)

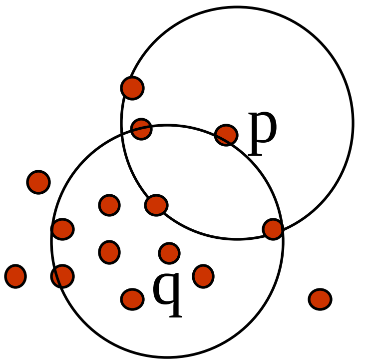
Comparing different clustering algorithms



https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

Density-Based Clustering: Basic Concepts

- Two parameters:
  - ε: Maximum radius of the neighbourhood
  - *MinPts*: Minimum number of points in an Eps-neighbourhood of that point
- $N_\varepsilon(q)$: {q belongs to D | dist(p,q) ≤ ε}
- Directly density-reachable: A point *p* is directly density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if
  - *p* belongs to $N_\varepsilon(q)$
  - If $|N_\varepsilon(q)| \geq$ *MinPts*
    - q is a core point
  - Otherwise, q is a border point

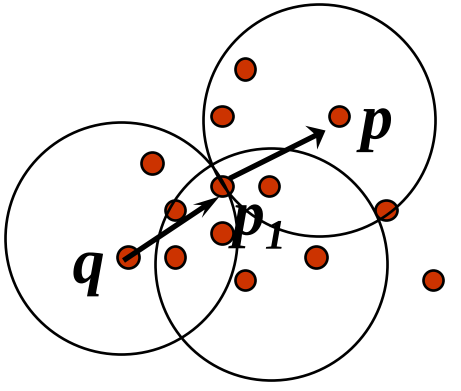


MinPts = 5

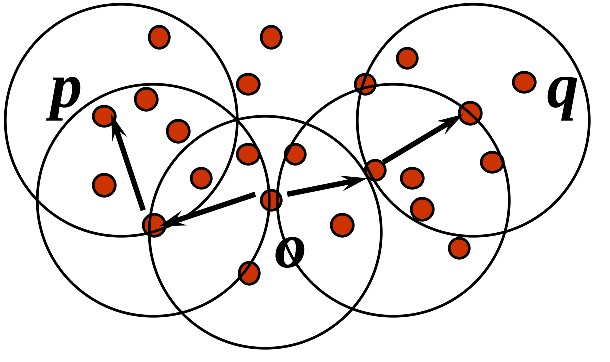
ε = 1 cm

Density-Reachable and Density-Connected

- Density-reachable:
  - A point p is density-reachable from a point q w.r.t. ε, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i



- Density-connected
  - A point p is density-connected to a point q w.r.t. ε, *MinPts* if there is a point o such that both, p and q are density-reachable from o w.r.t. *Eps* and *MinPts*

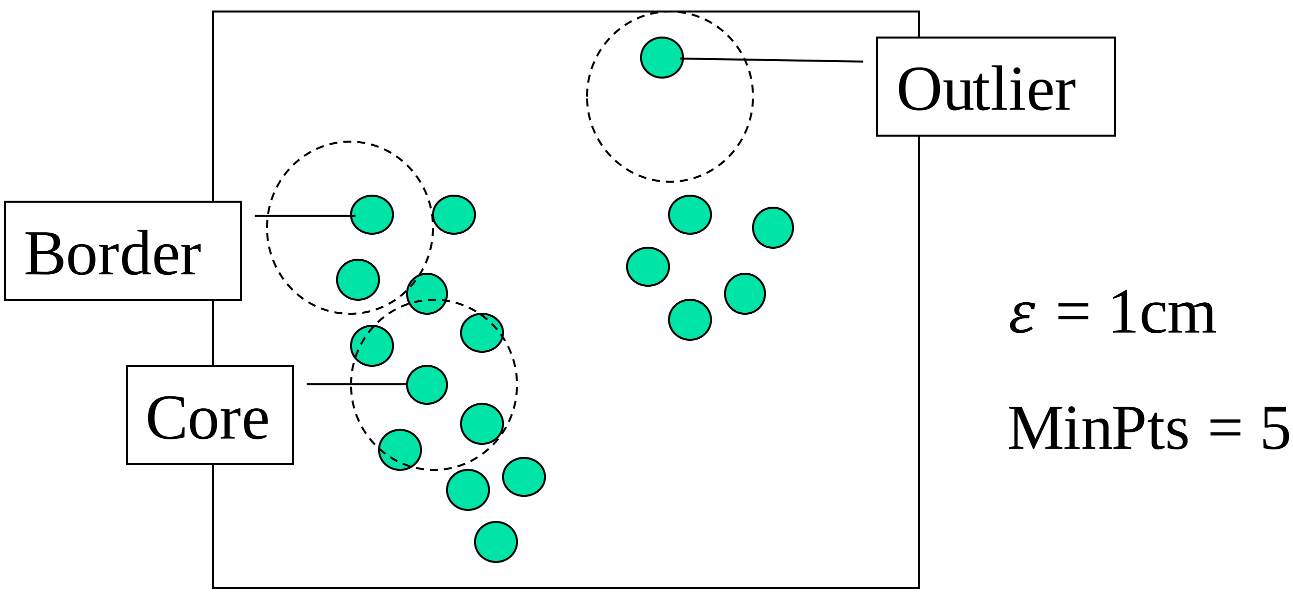


Definition: Density-based Clusters

- Given a set of points D, a cluster C is a density-based cluster w.r.t. ε and MinPts if
  - (Maximality) For every p_i, p_j in D, if $p_i \in C$ and p_j is density-reachable from p_i, p_j belongs to C
  - (Connectivity) For every p_i, p_j in C, p_i is density-connected from p_j
- Outlier
  - If a point in p is a border point and does not belongs the any other clusters, p is a outlier

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

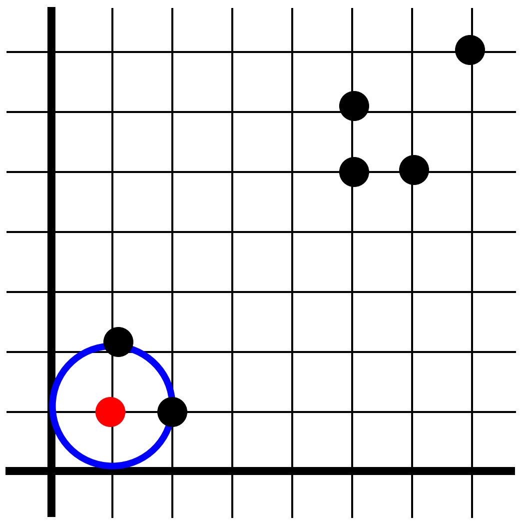
- Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* w.r.t. ε and *MinPts*
- If *p* is a core point, a cluster is formed
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed

An Example of DBSCAN



$\varepsilon = 1$
MinNumPoints = 3

Start from (1,1) point

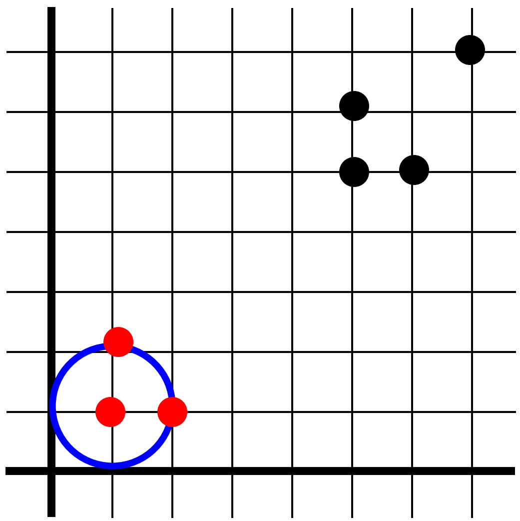
Class A

(1,1)

Unclassified

(5,5)(5,6)
(6,5)(7,7)

An Example of DBSCAN



$\varepsilon = 1$
MinNumPoints = 3

Start from (1,1) point

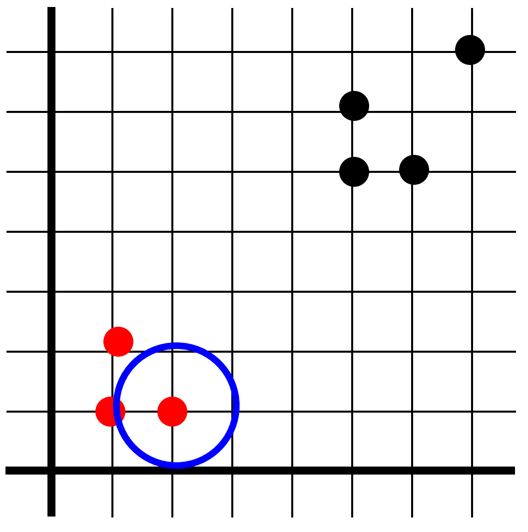
Class A

(1,1)(1,2)
(2,1)

Unclassified

(5,5)(5,6)
(6,5)(7,7)

An Example of DBSCAN



$\varepsilon = 1$
MinNumPoints = 3

Start from (2,1) point

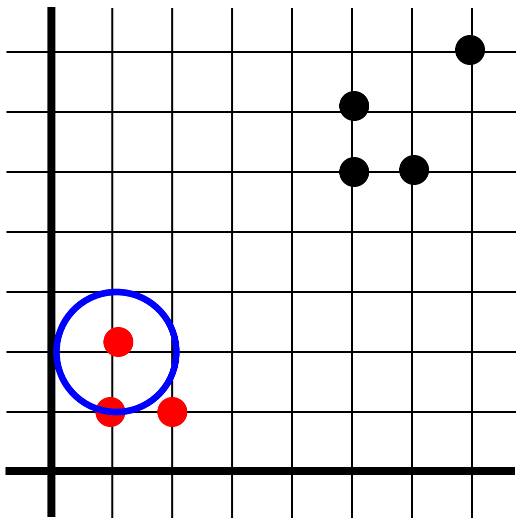
Class A

(1,1)(1,2)
(2,1)

Unclassified

(5,5)(5,6)
(6,5)(7,7)

An Example of DBSCAN



ε = 1
MinNumPoints = 3

Start from (1,2) point

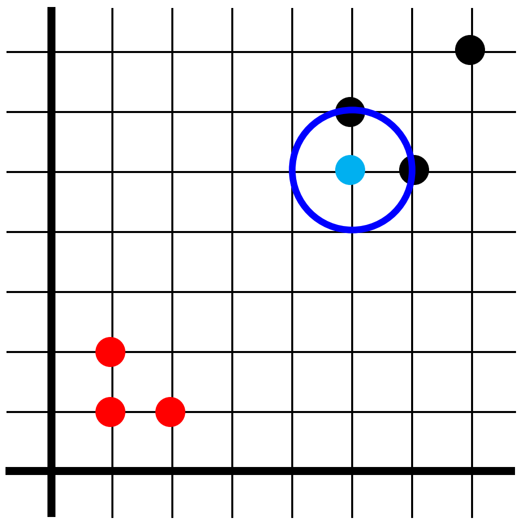
Class A

(1,1)(1,2)
(2,1)

Unclassified

(5,5)(5,6)
(6,5)(7,7)

An Example of DBSCAN



$\varepsilon = 1$
MinNumPoints = 3

Start from (5,5) point

Class A

(1,1)(1,2)
(2,1)

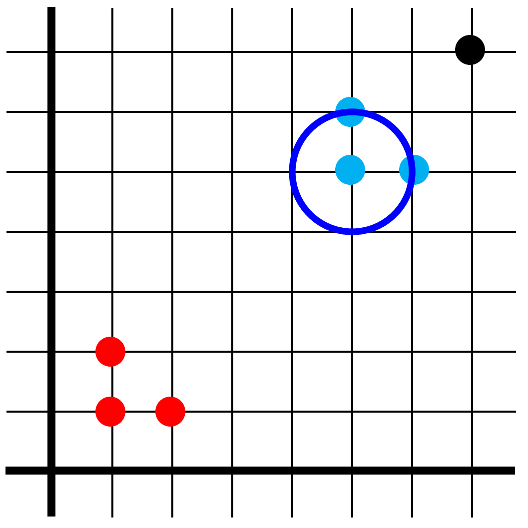
Class B

(5,5)

Unclassified

(5,6)
(6,5)(7,7)

An Example of DBSCAN



$\varepsilon = 1$
MinNumPoints = 3

Start from (5,5) point

Class A

(1,1)(1,2)
(2,1)

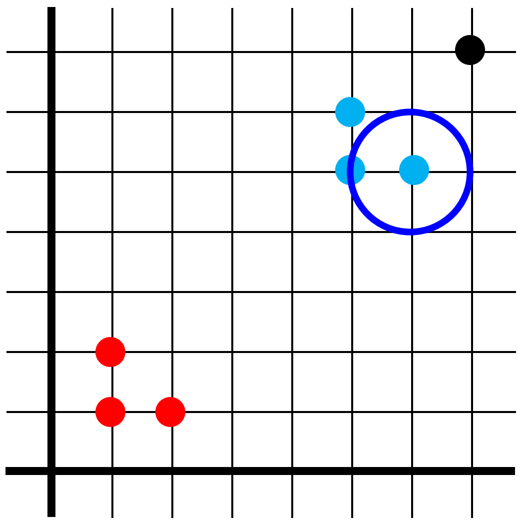
Class B

(5,5) (5,6)
(6,5)

Unclassified

(7,7)

An Example of DBSCAN



$\varepsilon = 1$
MinNumPoints = 3

Start from (6,5) point

Class A

(1,1)(1,2)
(2,1)

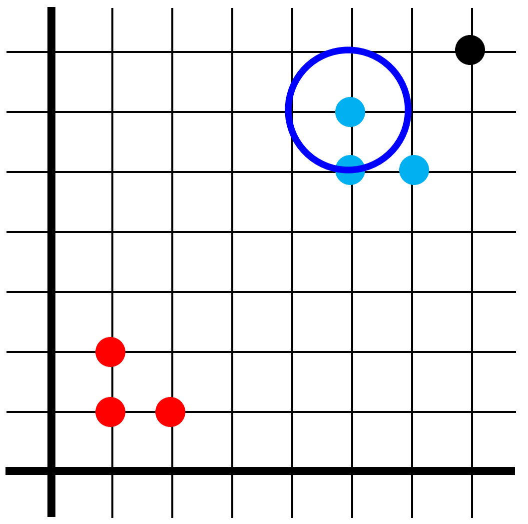
Class B

(5,5) (5,6)
(6,5)

Unclassified

(7,7)

An Example of DBSCAN



$\varepsilon = 1$
MinNumPoints = 3

Start from (5,6) point

Class A

(1,1)(1,2)
(2,1)

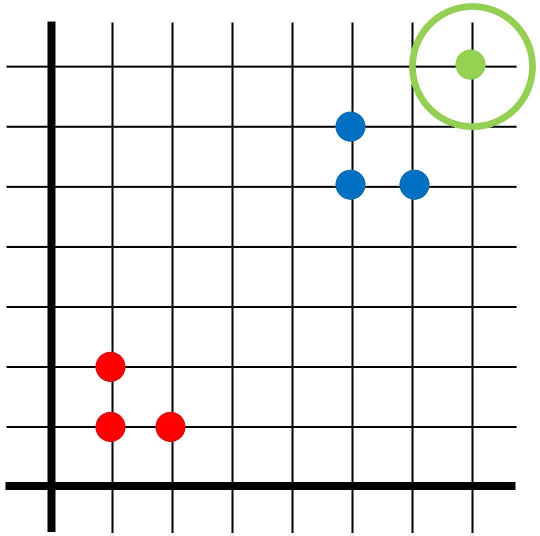
Class B

(5,5) (5,6)
(6,5)

Unclassified

(7,7)

An Example of DBSCAN



(5,6) point : Border Point

Class A

(1,1)(1,2)
(2,1)

Class B

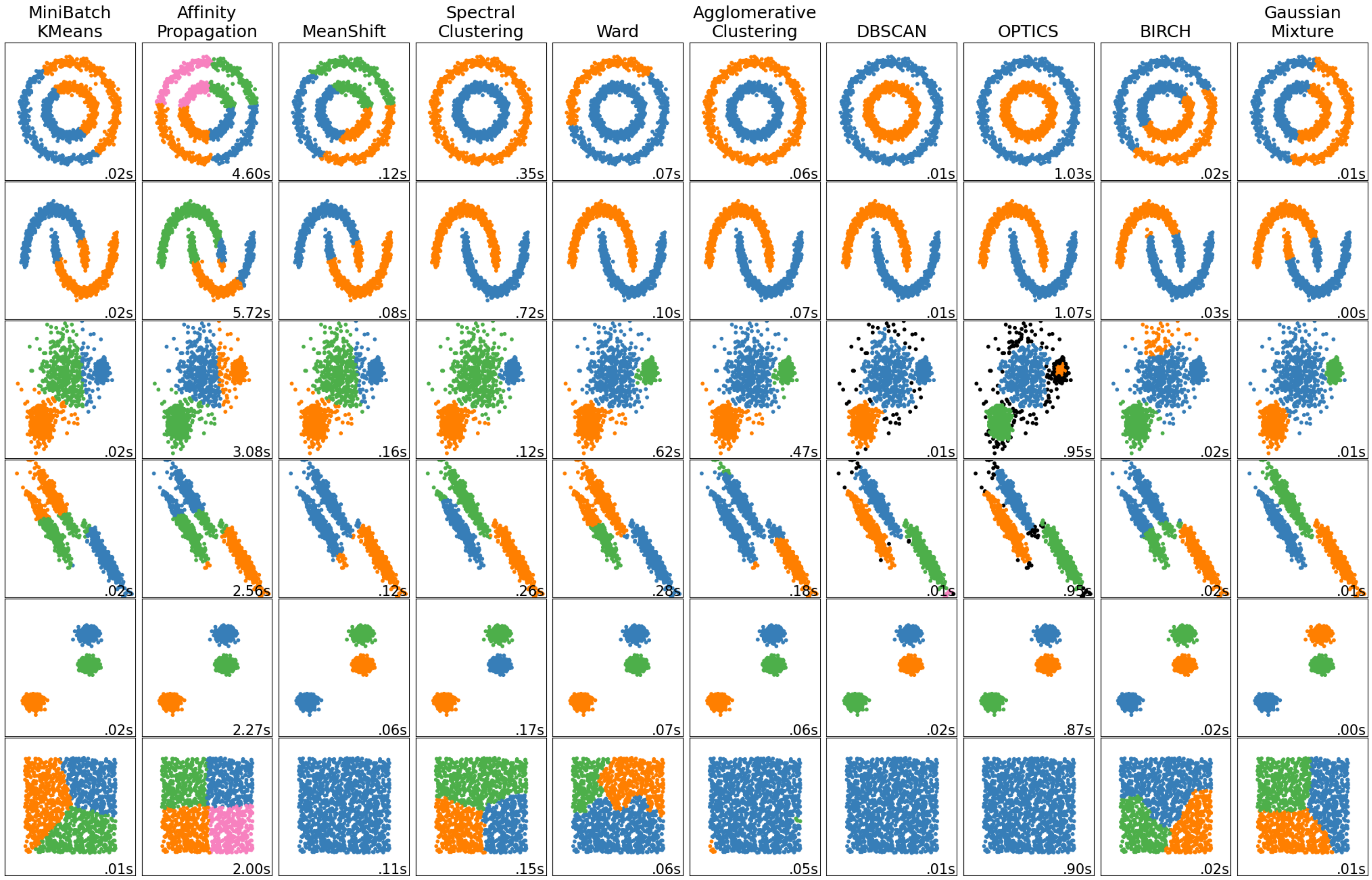
(5,5)(6,5)
(5,6)

Class C

(7,7)

→ Outlier

Comparing different clustering algorithms



https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

Python – DBSCAN

DBSCAN

```python
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.05, min_samples=20)
y_pred = dbscan.fit_predict(X)
print(y_pred[:10])
```

```
[ 0  4 -1 -1 -1 -1 -1 -1 -1 -1]
```

Maximum radius of the neighborhood

The number of points in a neighborhood to be a core point

DBSCAN

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.05, min_samples=20)
y_pred = dbscan.fit_predict(X)
print(y_pred[:10])
```

```
[ 0  4 -1 -1 -1 -1 -1 -1 -1 -1]
```

Perform clustering and output the cluster index for each data point

Outliers are indexed as -1

DBSCAN

Pairs of (eps, min_samples)

```
for i, (eps, min_samples) in enumerate(
        ((0.05, 20), (0.06, 20), (0.06, 15), (0.06, 6))):
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    y_pred = dbscan.fit_predict(X)
    plt.figure(i + 1, figsize=(5, 5))
    plt.scatter(X[:, 0], X[:, 1],
                cmap=cmap, s=4, c=y_pred)
    plt.title("eps: {}, min_samples: {}".format(
        eps, min_samples))
plt.show()
```

DBSCAN

Iterate over 4 pairs of (eps, min_samples)

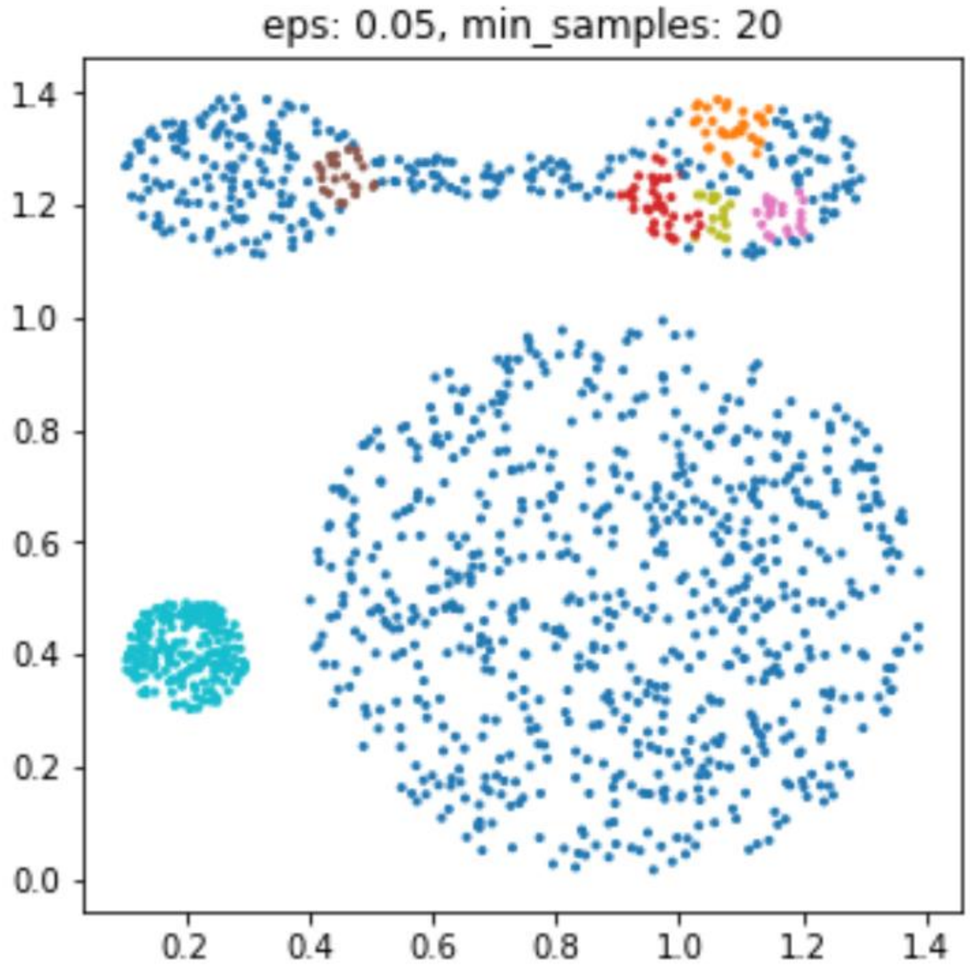
```
for i, (eps, min_samples) in enumerate(
        ((0.05, 20), (0.06, 20), (0.06, 15), (0.06, 6))):
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    y_pred = dbscan.fit_predict(X)
    plt.figure(i + 1, figsize=(5, 5))
    plt.scatter(X[:, 0], X[:, 1],
                cmap=cmap, s=4, c=y_pred)
    plt.title("eps: {}, min_samples: {}".format(
        eps, min_samples))
plt.show()
```

Parameters

```
dbscan = DBSCAN(
    eps=eps,
    min_samples=min_samples)
```

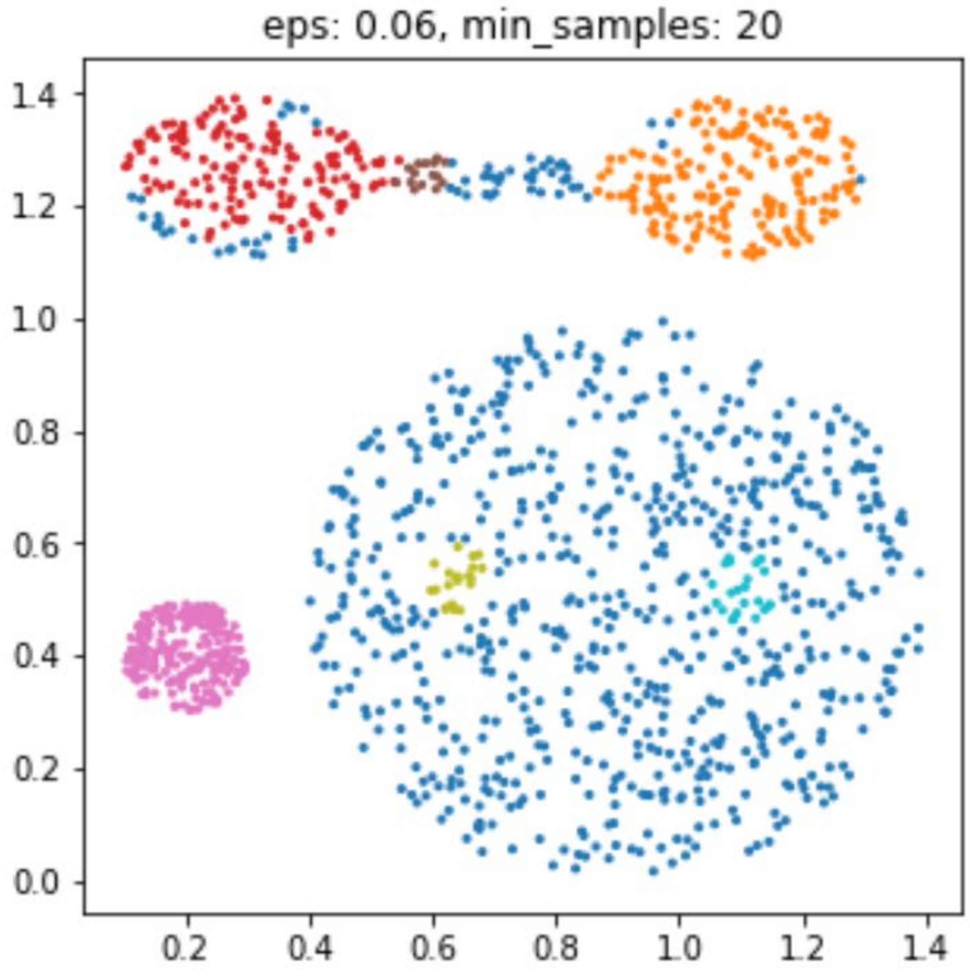
- `eps`: maximum radius of the neighborhood
- `min_samples`: the number of points in a neighborhood to be a core point
- `metric`: the distance metric (default: `"euclidean"`)

Clustering Result

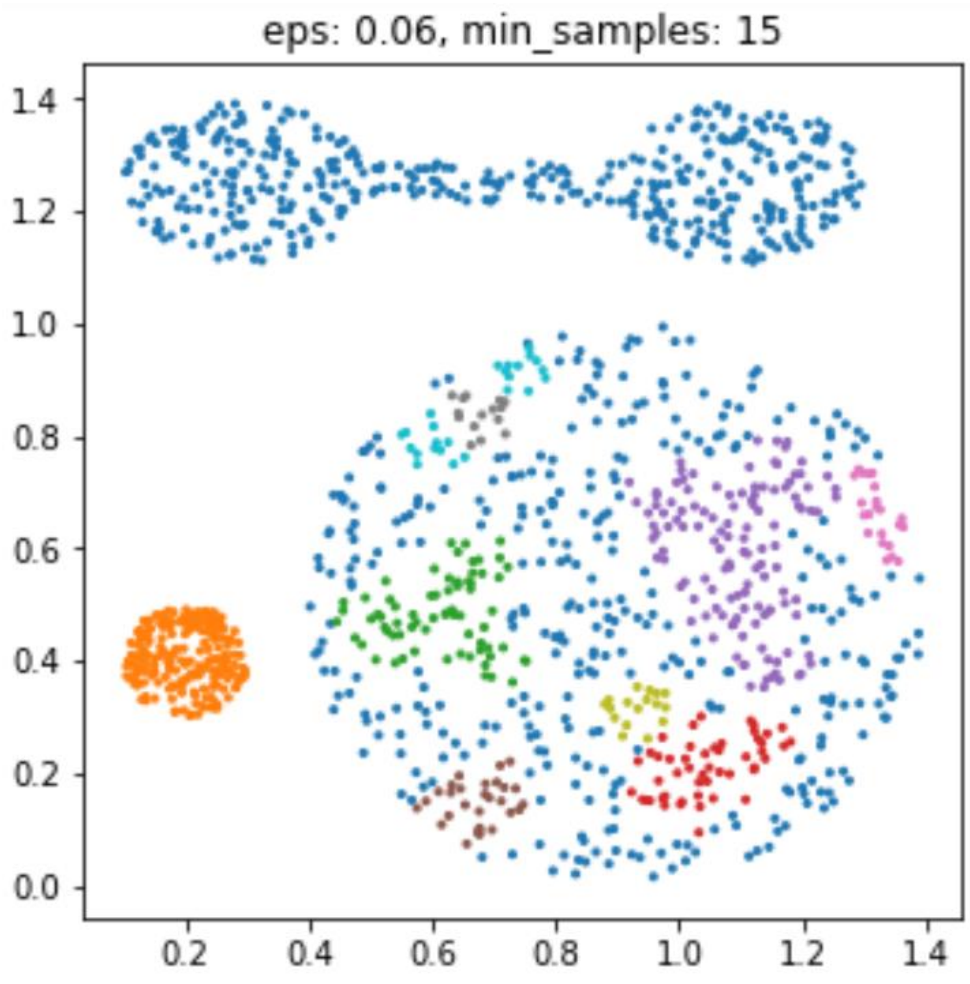


Outliers are colored by blue

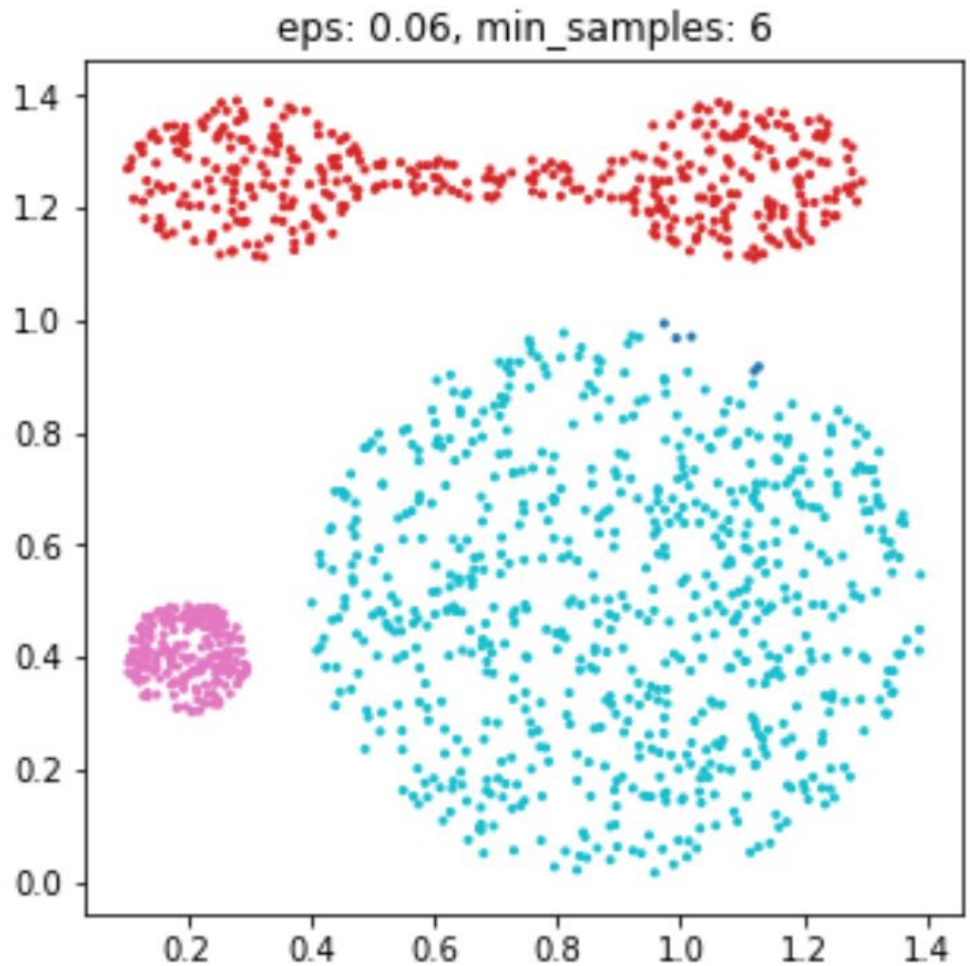
Clustering Result



Clustering Result



Clustering Result



References (1)

- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD'98
- M. R. Anderberg. Cluster Analysis for Applications. Academic Press, 1973.
- M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure, SIGMOD'99.
- Beil F., Ester M., Xu X.: "Frequent Term-Based Text Clustering", KDD'02
- M. M. Breunig, H.-P. Kriegel, R. Ng, J. Sander. LOF: Identifying Density-Based Local Outliers. SIGMOD 2000.
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. KDD'96.
- M. Ester, H.-P. Kriegel, and X. Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. SSD'95.
- D. Fisher. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2:139-172, 1987.
- D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamic systems. VLDB'98.
- V. Ganti, J. Gehrke, R. Ramakrishan. CACTUS Clustering Categorical Data Using Summaries. KDD'99.

References (2)

- D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamic systems. In Proc. VLDB'98.
- S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. SIGMOD'98.
- S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *ICDE'99*, pp. 512-521, Sydney, Australia, March 1999.
- A. Hinneburg, D.l A. Keim: An Efficient Approach to Clustering in Large Multimedia Databases with Noise. KDD'98.
- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Printice Hall, 1988.
- G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *COMPUTER*, 32(8): 68-75, 1999.
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, 1990.
- E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. VLDB'98.

References (3)

- G. J. McLachlan and K.E. Bkasford. Mixture Models: Inference and Applications to Clustering. John Wiley and Sons, 1988.
- R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. VLDB'94.
- L. Parsons, E. Haque and H. Liu, Subspace Clustering for High Dimensional Data: A Review, SIGKDD Explorations, 6(1), June 2004
- E. Schikuta. Grid clustering: An efficient hierarchical clustering method for very large data sets. Proc. 1996 Int. Conf. on Pattern Recognition
- G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. VLDB’98.
- A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-Based Clustering in Large Databases, ICDT'01.
- A. K. H. Tung, J. Hou, and J. Han. Spatial Clustering in the Presence of Obstacles, ICDE'01
- H. Wang, W. Wang, J. Yang, and P.S. Yu. Clustering by pattern similarity in large data sets, SIGMOD’02
- W. Wang, Yang, R. Muntz, STING: A Statistical Information grid Approach to Spatial Data Mining, VLDB’97
- T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH : An efficient data clustering method for very large databases. SIGMOD'96
- X. Yin, J. Han, and P. S. Yu, “LinkClus: Efficient Clustering via Heterogeneous Semantic Links”, VLDB'06