Computing

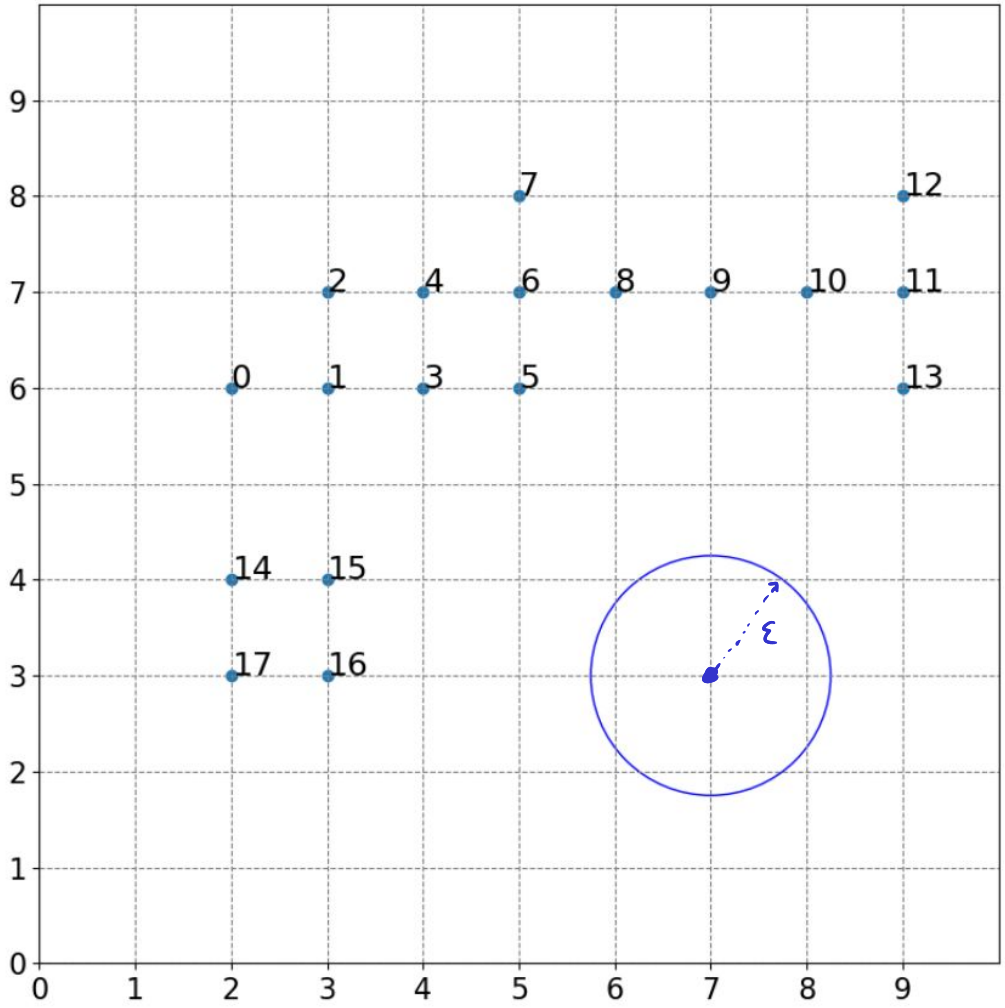
CS5228: Knowledge Discovery and Data Mining

Tutorial 2 — K-Means & DBSCAN

Question 1

DBSCAN

- ε = 1.25
- MinPts = 4



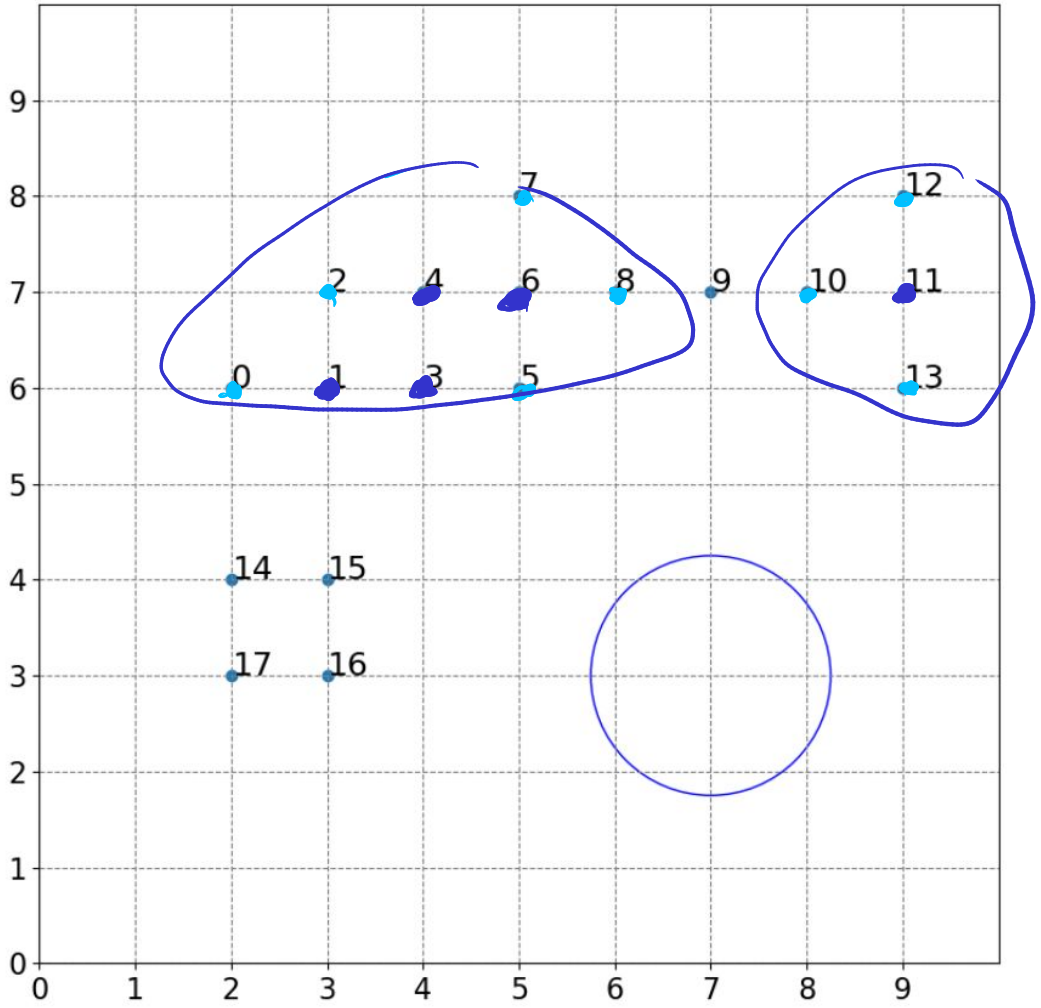
Question 1

DBSCAN

- ε = 1.25
- MinPts = 4

(a) Describe the result by listing all

- core points
- border points
- noise points



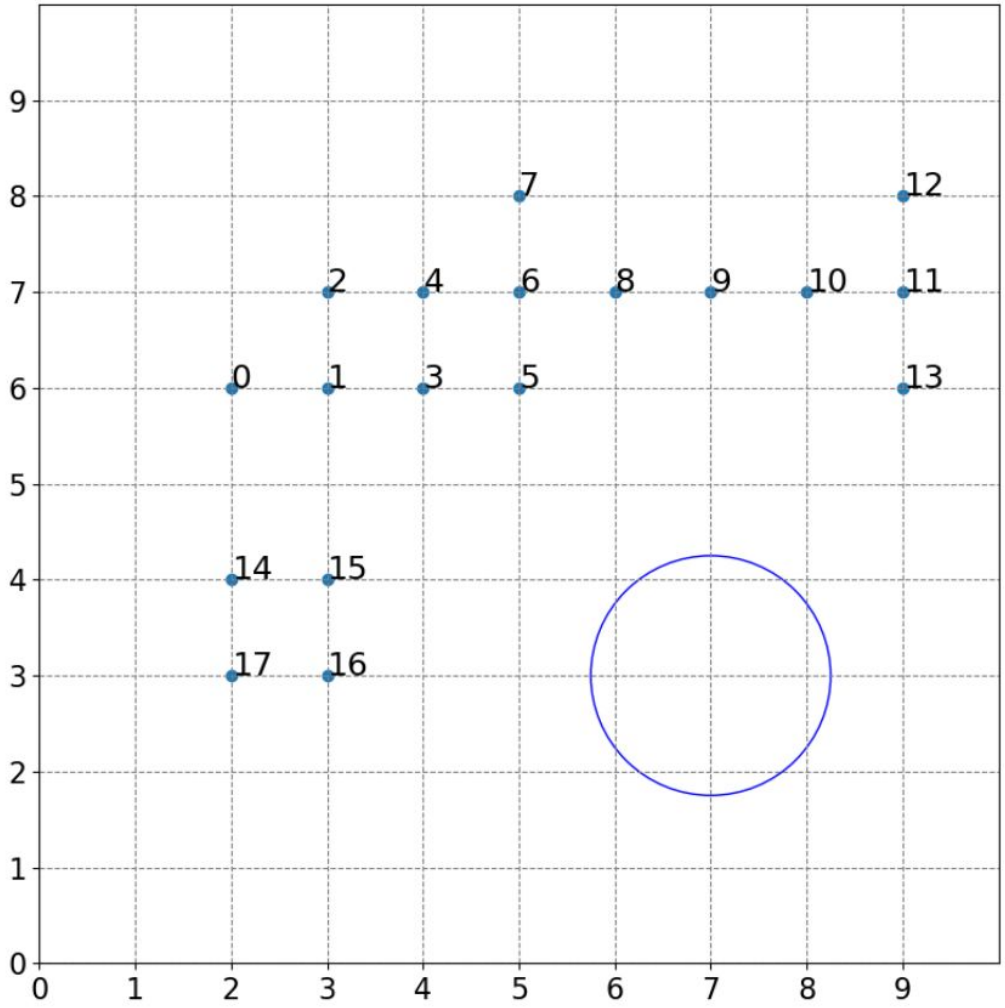
Question 1

DBSCAN

- ε = 1.25
- MinPts = 4

(a) Describe the result by listing all

- core points
- border points
- noise points



Solution

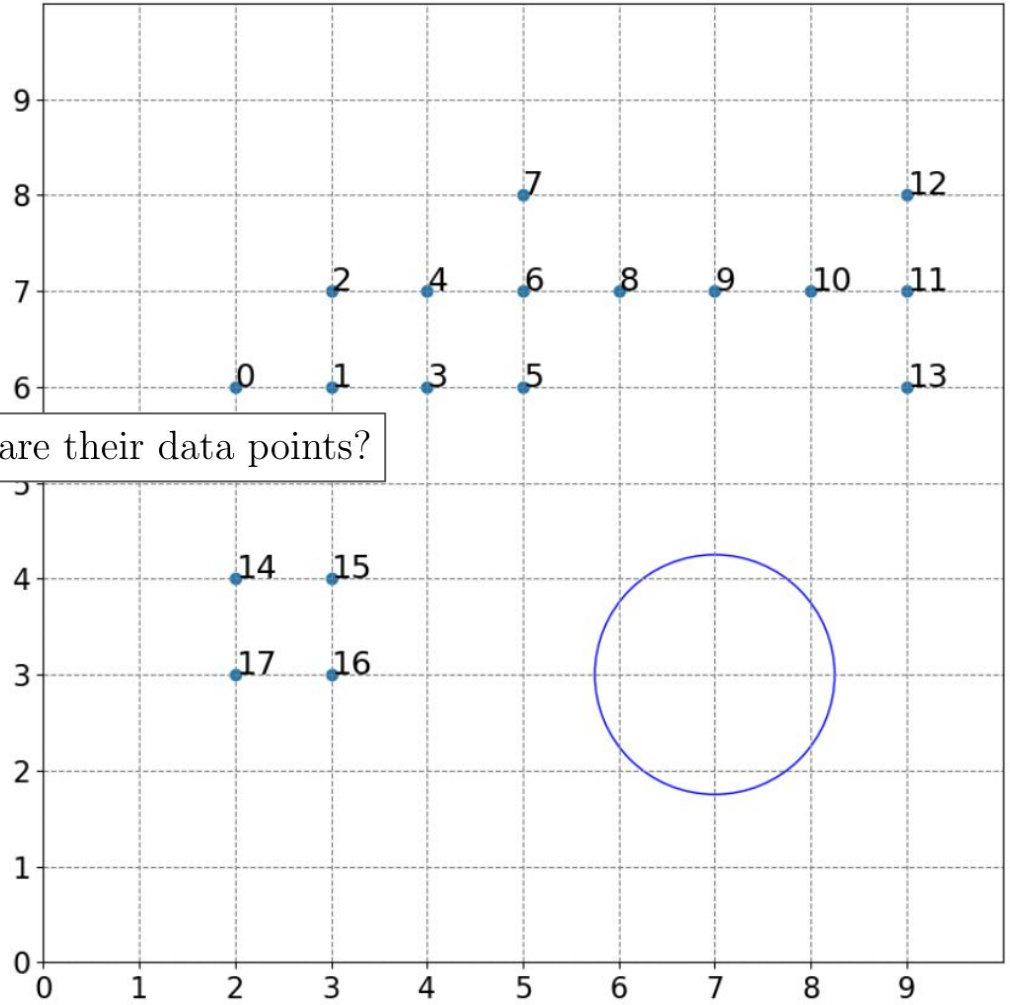
- core points = [1, 3, 4, 6, 11]
- border points = [0, 2, 5, 7, 8, 10, 12, 13]
- noise points = [9, 14, 15, 16, 17]

Question 1

DBSCAN

- ε = 1.25
- MinPts = 4

(b) How many clusters are there, and what are their data points?

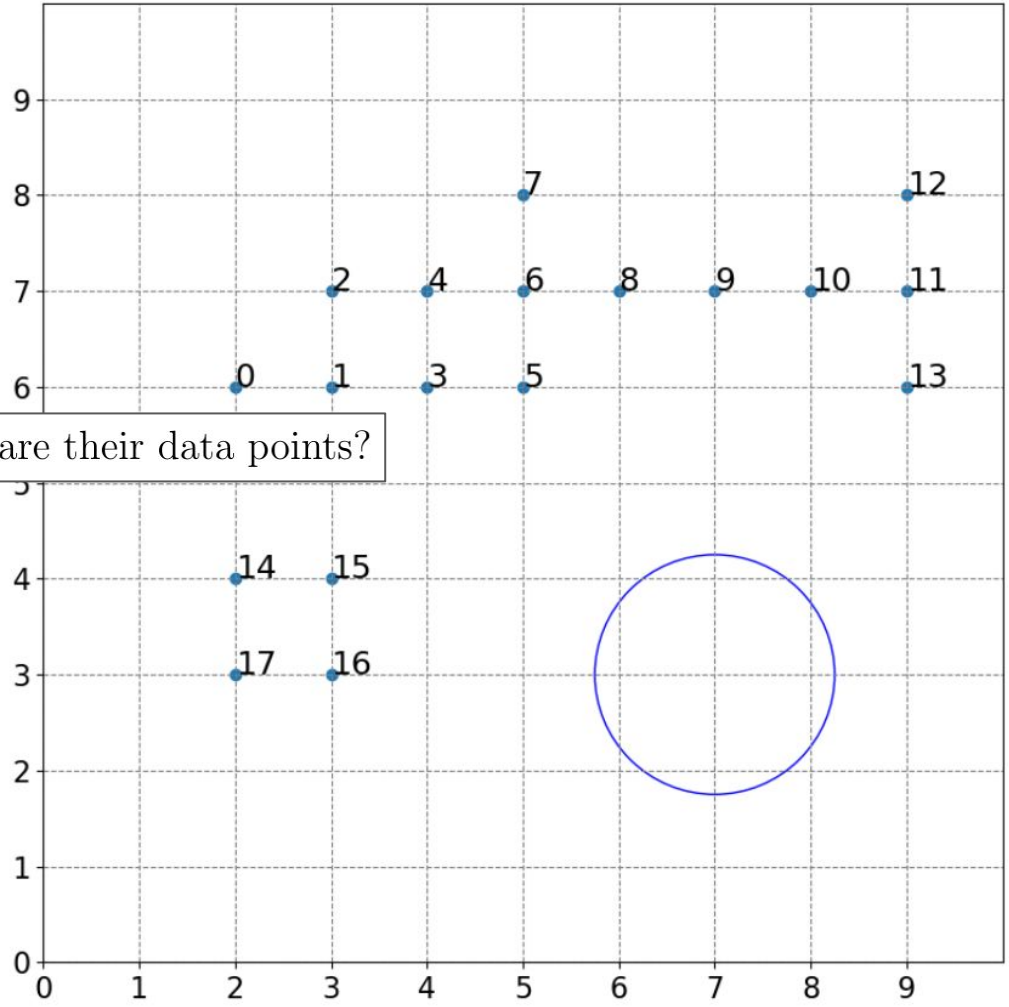


Question 1

DBSCAN

- ε = 1.25
- MinPts = 4

(b) How many clusters are there, and what are their data points?



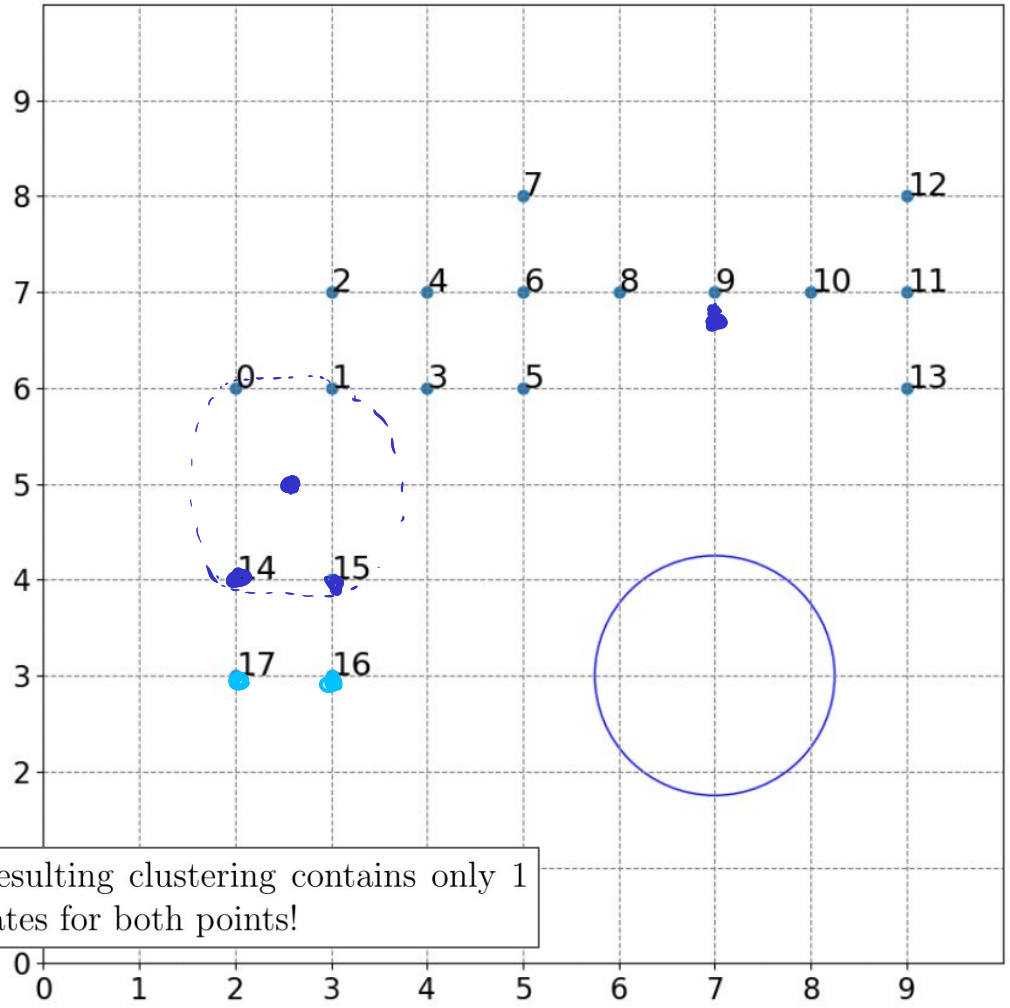
Solution

- Cluster 1 = [0, 1, 2, 3, 4, 5, 6, 7, 8]
- Cluster 2 = [10, 11, 12, 13]

Question 1

DBSCAN

- ε = 1.25
- MinPts = 4



(c) Can you add 2 data points such that the resulting clustering contains only 1 cluster and no noise? If so, give the coordinates for both points!

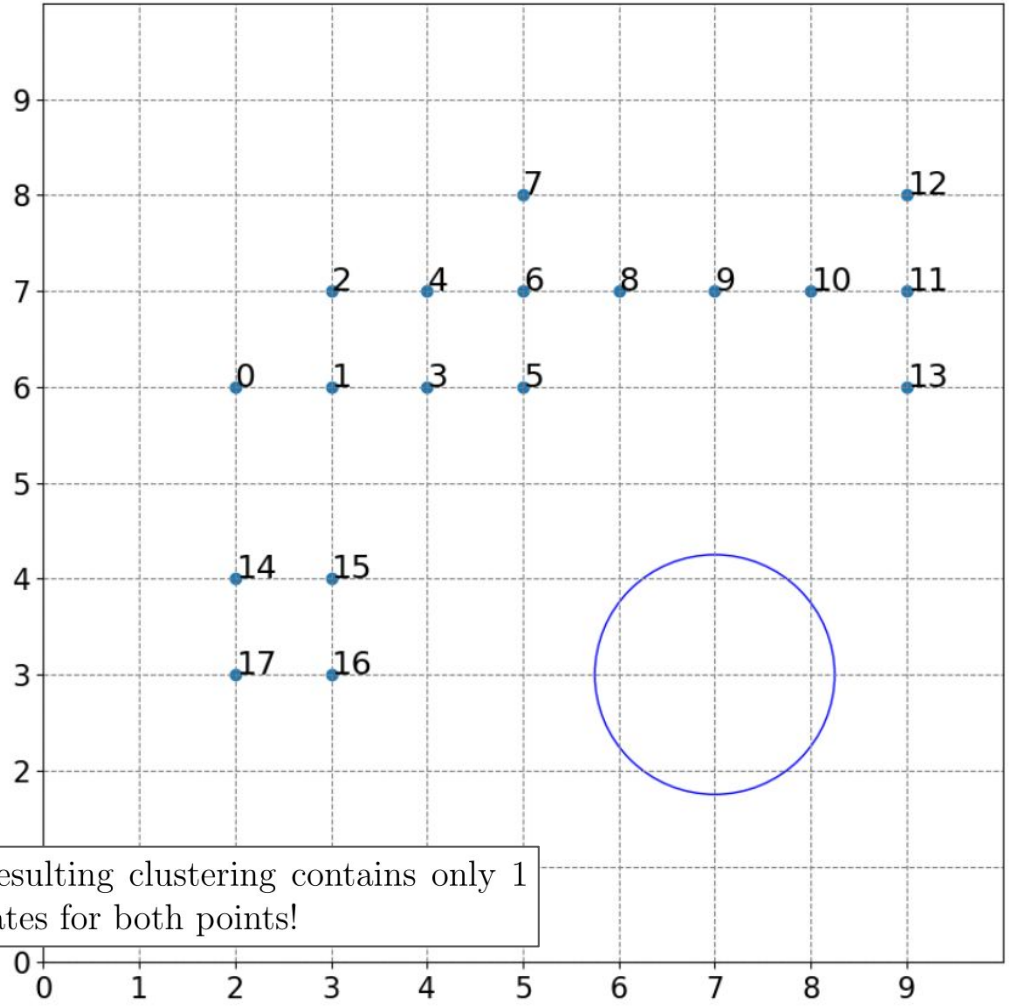
Question 1

DBSCAN

- ε = 1.25
- MinPts = 4

Solution

- Example solution: (2.5, 5), (7, 6.8)
- There's no unique solution here

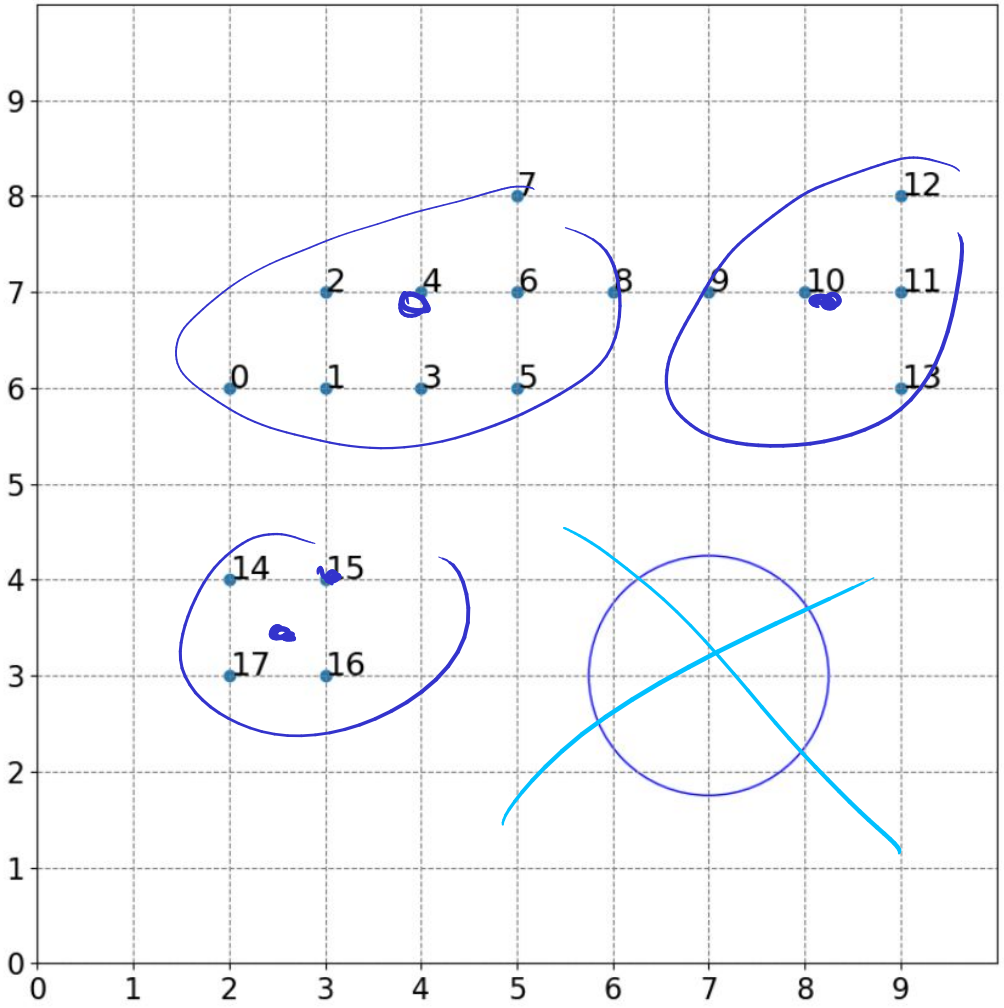


(c) Can you add 2 data points such that the resulting clustering contains only 1 cluster and no noise? If so, give the coordinates for both points!

Question 2

K-Means

- K = 3



Question 2

(a) For $K = 3$, can you find locations for the initial centroids so that the resulting clustering will contain 0, 1, 2, or 3 non-empty clusters? You can answer this question qualitatively; there is no need to list any exact coordinates for the initial centroids.

0: ~~not possible~~

~~1~~ empty

1: 2 centroids "within" data / 1 far away

2: 1 ~ / 2 ~

3: not possible

Question 2

(a) For $K = 3$, can you find locations for the initial centroids so that the resulting clustering will contain 0, 1, 2, or 3 non-empty clusters? You can answer this question qualitatively; there is no need to list any exact coordinates for the initial centroids.

Solution

- K-Means never returns at least 1 non-empty cluster ➜ 0 non-empty clusters is out
- For 1 non-empty cluster: place 1 centroid "within" the data, and 2 centroids far outside
- For 2 non-empty cluster: place 2 centroids "within" the data, and 1 centroid far outside
- Example initialization where it works: place centroids as points 4, 11, and 16

Question 2

(b) Now we assume that K-Means++ initialization is used. For $K = 3$, what is the minimum and maximum number of clusters? (Comment: For this question you may need to consider arbitrary datasets and not just the toy dataset!)

trivial: $1 \leq C \leq K$

Question 3

(b) Now we assume that K-Means++ initialization is used. For $K = 3$, what is the minimum and maximum number of clusters? (Comment: For this question you may need to consider arbitrary datasets and not just the toy dataset!)

Solution

- Number of clusters C guaranteed to be in $1 \leq C \leq K$
- However, not guarantee for K non-empty clusters!
    - Centroids typically move away from location of data points after first update
    - New location might cause "starving" centroids ➜ empty cluster(s)

Question 3

(a) Apart from their implementation, what is the fundamental difference between K-Means and DBSCAN?

Question 3

(a) Apart from their implementation, what is the fundamental difference between K-Means and DBSCAN?

Solution (alternative answer possible)

- K-Means is defined as an optimization problem; hence there is a notion of local and global optimal solutions. The commonly used Lloyd's algorithm is a heuristic that does no guarantee to always find the global optimum
- K-Means considers relative similarities/distances
- K-Mean favors blob-like clusters
- DBSCAN is not defined as an optimization problem, so there's no notion of a local/global optimum. The DBSCAN algorithm is not a heuristic.
- DBSCAN considers absolute similarities/distances
- DBSCAN can handle non-blob-like clusters

Question 3

(b) What are meaningful criteria to decide whether K-Means or DBSCAN is the preferable clustering method for a certain task?

Questions 3

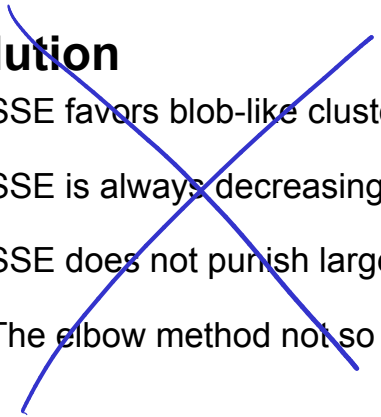
(b) What are meaningful criteria to decide whether K-Means or DBSCAN is the preferable clustering method for a certain task?

Solution

- DBSCAN has the notion of noise, which can be used for outlier detection
- DBSCAN better when clusters are decidedly not blobs
- DBSCAN can work well if the parameters for ε and MinPts can be intuitively set
- K-Means when the value for k is predefined by application context
- Since K-Means considers relative similarities/distances, it's arguably easier to use for an EDA to get a meso-view of the data

Question 3

(c) Come up with 5 example tasks and discuss why K-Means or DBSCAN would be your method of choice!



Solution

- SSE favors blob-like clusters
- SSE is always decreasing
- SSE does not punish large number of clusters
- The elbow method not so straightforward to apply

Question 3

(c) Come up with 5 example tasks and discuss why K-Means or DBSCAN would be your method of choice!

Solution (alternative answer possible)

- Traffic congestion along roads based on location of cars (DBSCAN)
- Identifying locations of low coverage, e.g., distribution of Starbucks outlets across a city (DBSCAN)
- Credit card fraud or intrusion detection to find outliers (DBSCAN) anomaly detection
- Organizing conference papers into a given set of research areas (K-Means) → fixed k
- Clustering people based on biological data (e.g., height, weight). Such data is typically always normally distributed so clusters are more likely to be blobs (K-Means).

Question 3

(d) Is there any example where fundamentally only K-Means is applicable but not DBSCAN, or vice versa?

Question 3

(d) Is there any example where fundamentally only K-Means is applicable but not DBSCAN, or vice versa?

Solution

- No, both methods only require a well-defined similarity/distance measure to be applicable.

Notes

Notes

Notes