Practical Data Science: Clustering

Dr. Yongli Ren

(yongli.ren@rmit.edu.au)

Computer Science & IT

School of Science

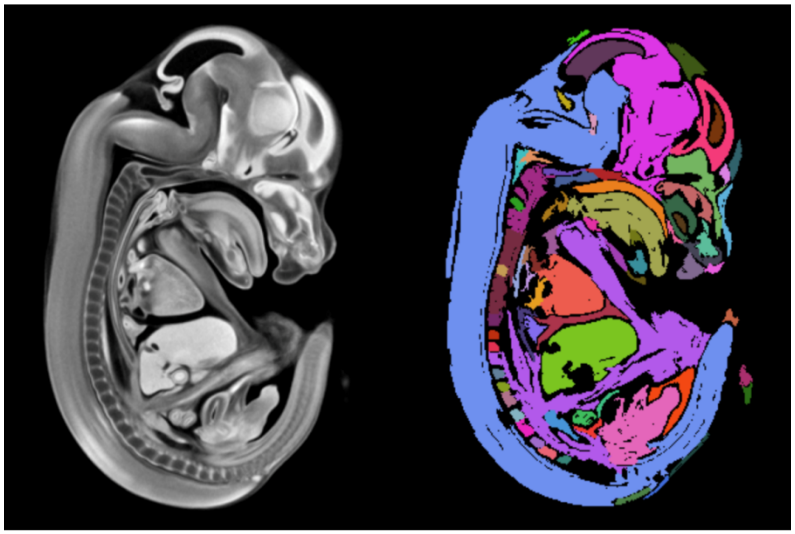
Outline

- Part 1: What is DBSCAN
- Part 2: DBSCAN in *sklearn*

Practical Data Science – COSC2670

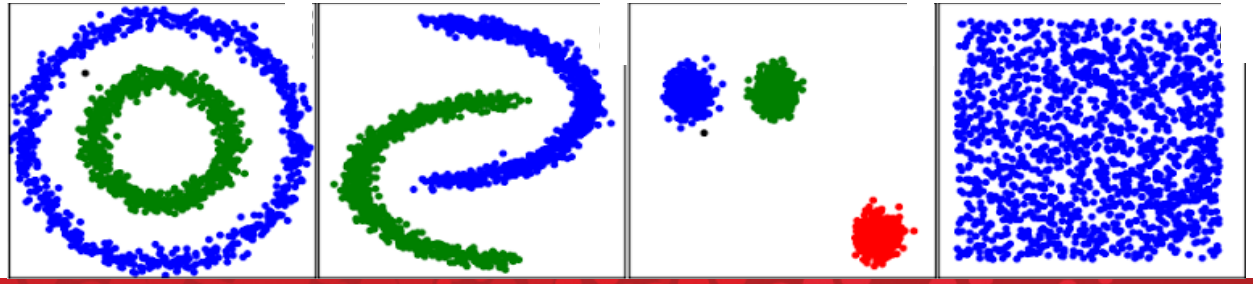
PART 1:
WHAT IS DBSCAN

What is DBSCAN?



http://yaikhom.com/2015/12/29/image-segmentaion-using-dbscan-clustering.html

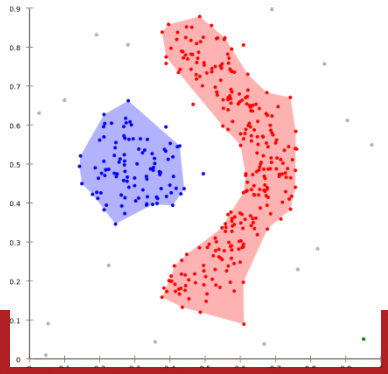
- A clustering technique
  - Is it supervised or unsupervised?
- It is mysterious:
  - It is young.
    - k means: the idea goes back to 1957.
    - DBSCAN: was proposed in 1996.
  - It has attracted lots of attention in both industries and academics.
    - Granted the test of time award in 2014 by an academic organisation.
  - It discovers clusters with arbitrary shape.
    - k means: spherical clusters.
  - It does not require the number of clusters.



http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py

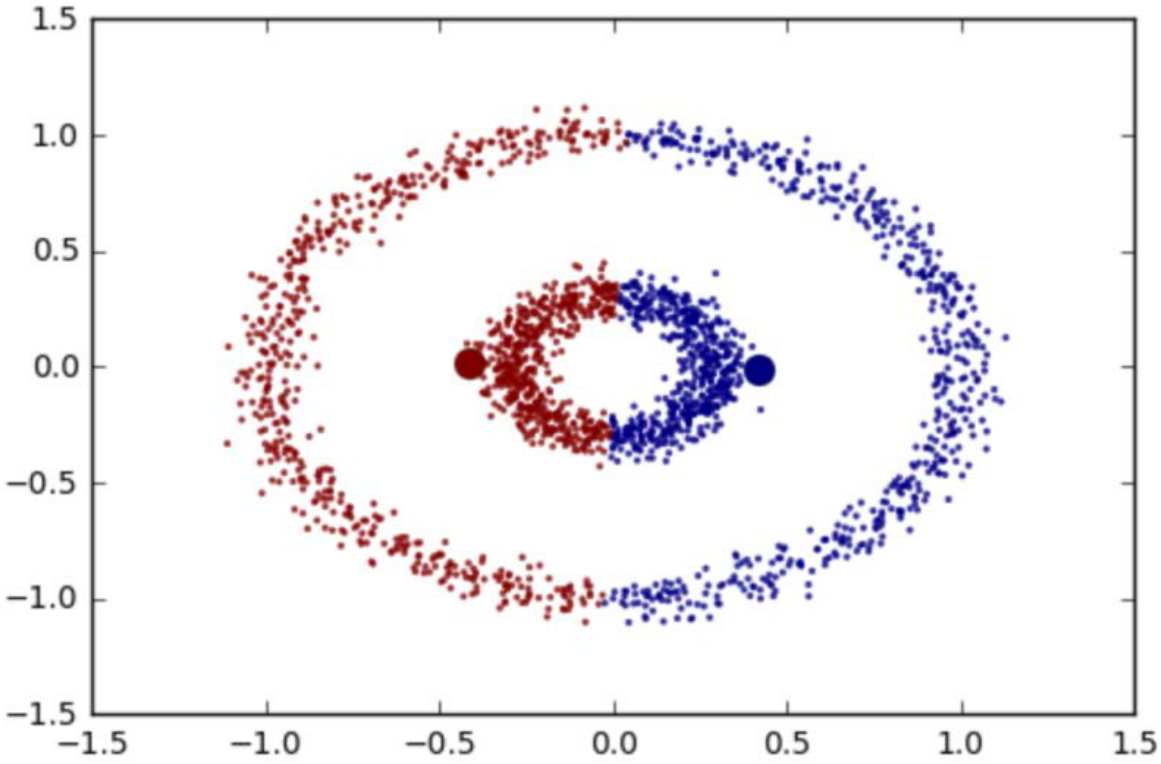
DBSCAN

- **Density-based spatial clustering of applications with noise** (**DBSCAN**)
  - A data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996.
- It is a density-based clustering algorithm:
  - given a set of points in some space,
  - it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).
- A density-based notion of *cluster*:
  - a cluster is defined as a maximal set of density-connected points.
- DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.



Why DBSCAN?

- k means isn't always a good option.
  - It can only identify Spherical clusters.
  - k must be specified.
    - This is not easy.

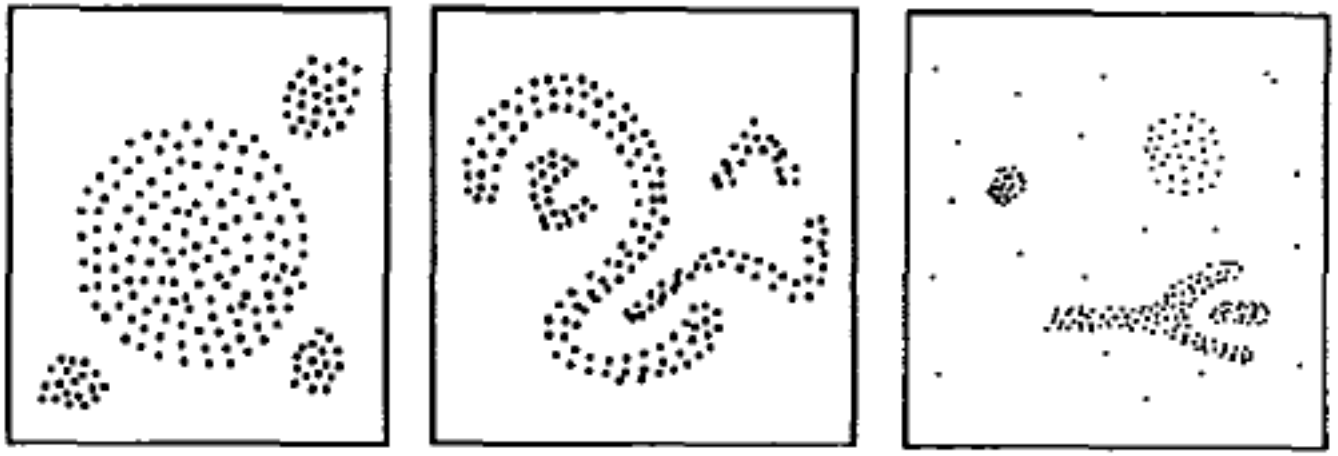


Why DBSCAN?

- Applications to large and new datasets need:
  - Minimal requirements of domain knowledge to determine the input parameters;
  - Discovery of clusters with arbitrary shape;
  - Good efficiency.
- DBSCAN relies on a density-based notion of clusters,
  - It is designed to discover clusters of arbitrary shape.
  - It requires only one input parameter.
  - It supports the user in determining an appropriate value for it.

Density-based Notion of Clusters

- Try to identify clusters and noise from the following data sets:

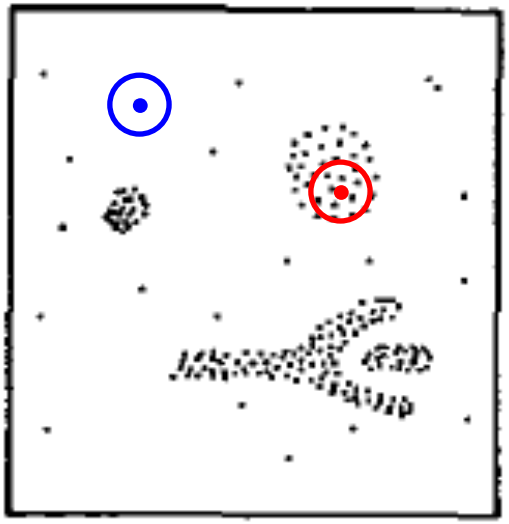


- The main reason why we recognize the clusters is that
  - Within each cluster, we have a typical density of points that is considerably higher than outside of the cluster.
- Furthermore, the density within the areas of noise is lower than the density in any of the clusters.

Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996)

DBSCAN

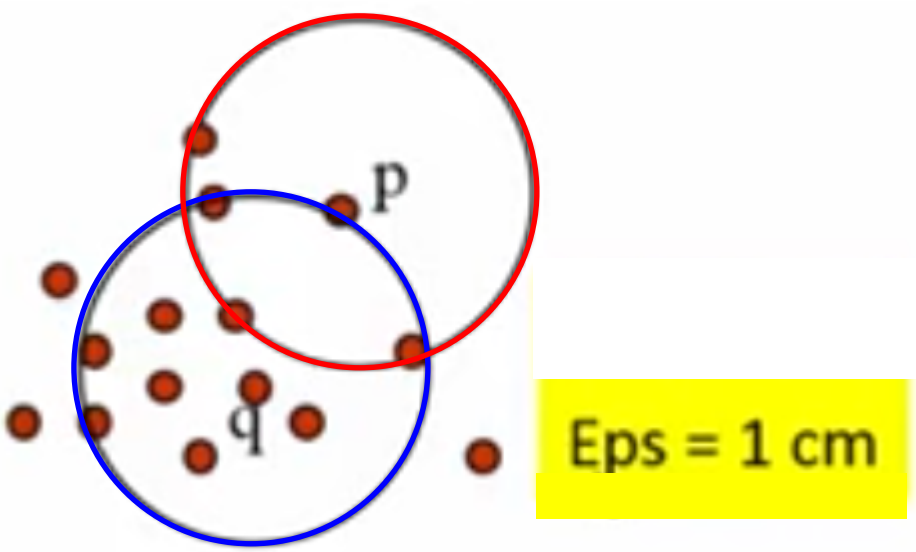
- How to identify these "clusters" and "noise" in set of points of some k-dimensional space?
  - Note that this applies to 2D or 3D Euclidean space, as well as to some high dimensional feature space.
- The key idea is that
  - For each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points,
    - i.e. the density in the neighborhood has to exceed some threshold.



Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996)

DBSCAN
- *Eps*-Neighborhood

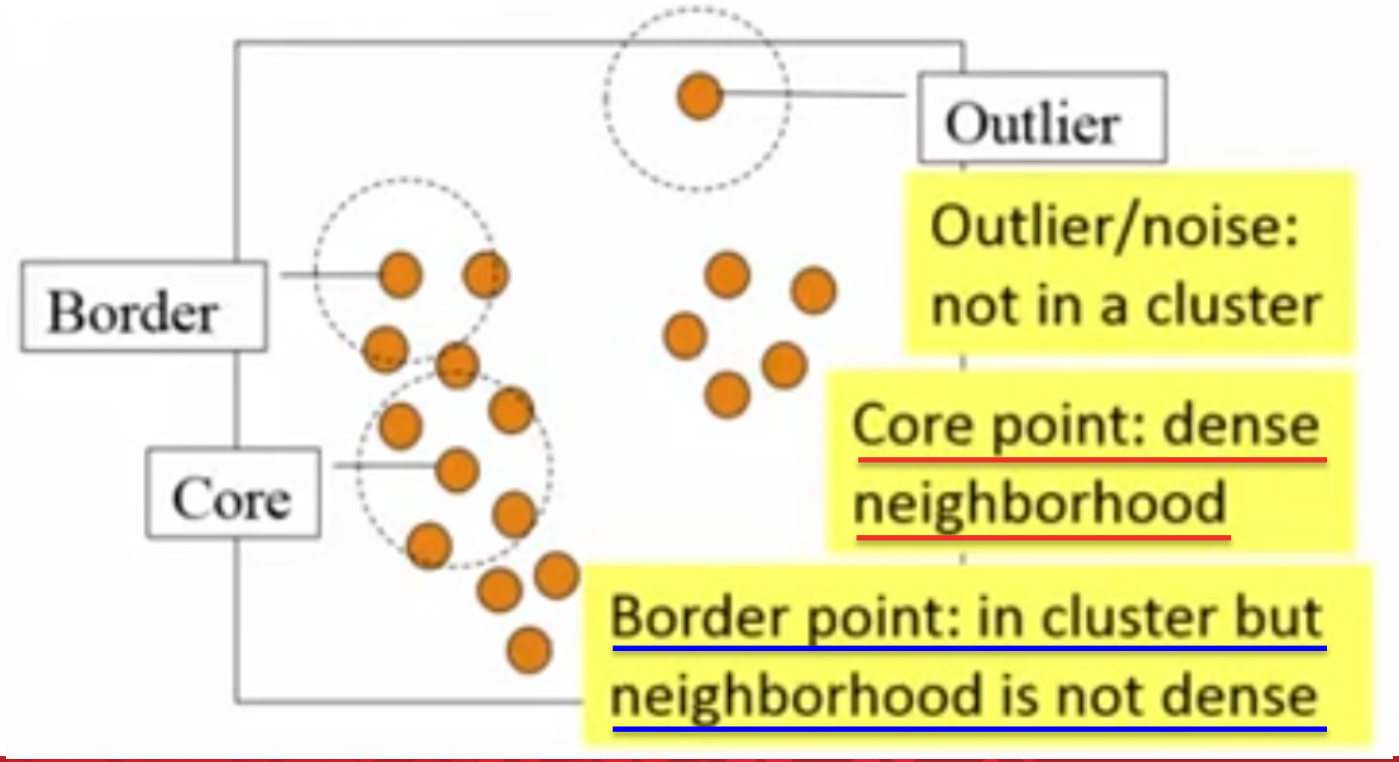
- The *Eps*-neighborhood of a point q:
  - Includes all data point whose distance to q is not larger than *Eps*.



https://www.youtube.com/watch?v=2yxM4zVVwiw

DBSCAN
- Core and Border Points

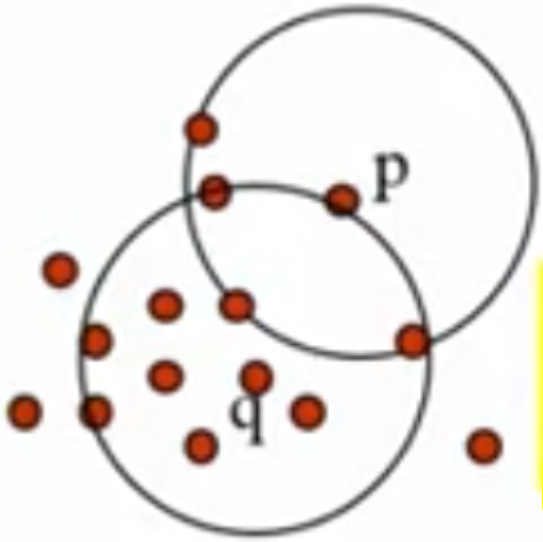
- There are two kinds of points in a cluster,
  - Core points: are points inside of the cluster;
  - Border points: are points on the border of the cluster.
- Outlier: data points that are not in any cluster.



https://www.youtube.com/watch?v=2yxM4zVVwiw

DBSCAN
- Core and Border Points

- In general, an *Eps*-neighborhood of a border point contains significantly fewer points than an *Eps*-neighborhoood a core point.
- Therefore, we would have to set the minimum number of points to a relatively low value in order to include all points belonging to the same cluster.
- This value, however, will not be characteristic for the respective cluster, particularly in the presence of noise.

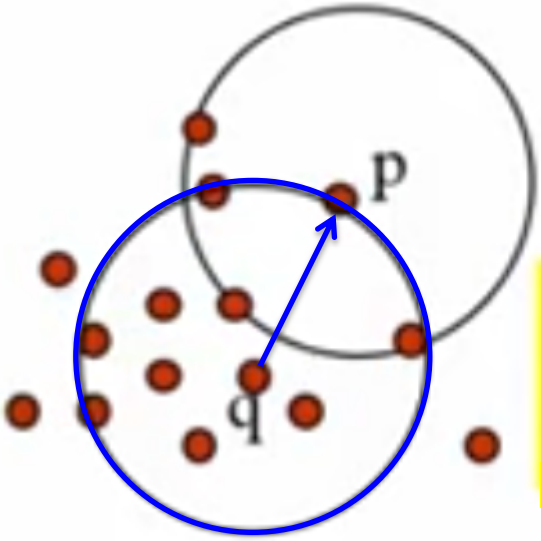


MinPts = 5

Eps = 1 cm

DBSCAN
- Directly Density-reachable

- Directly density-reachable:
  - For every point p in a cluster, there is a point q in the cluster so that p is inside of the Eps-neighborhood of q that contains at least $MinPts$ points.
- In other words
  - a point p is directly density-reachable from a point q w.r.t. Eps and $MinPts$ if
    - p belongs to q's neighborhood and
    - q is a core point.

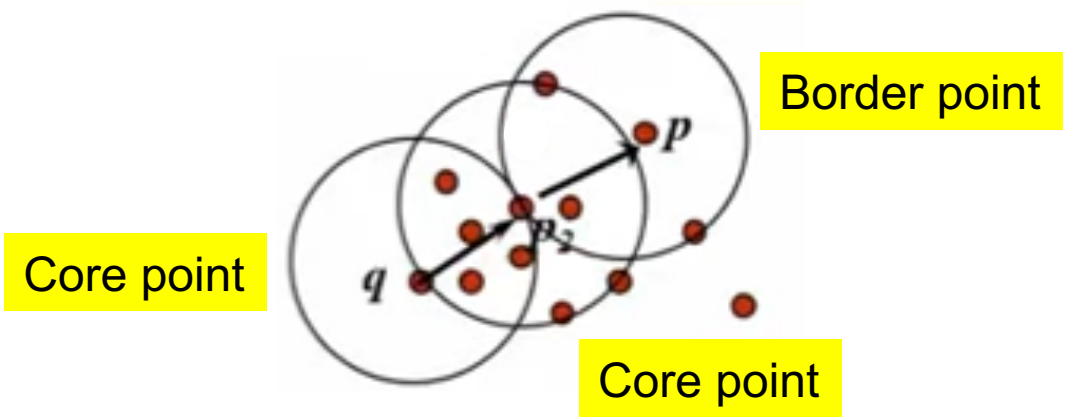


MinPts = 5

Eps = 1 cm

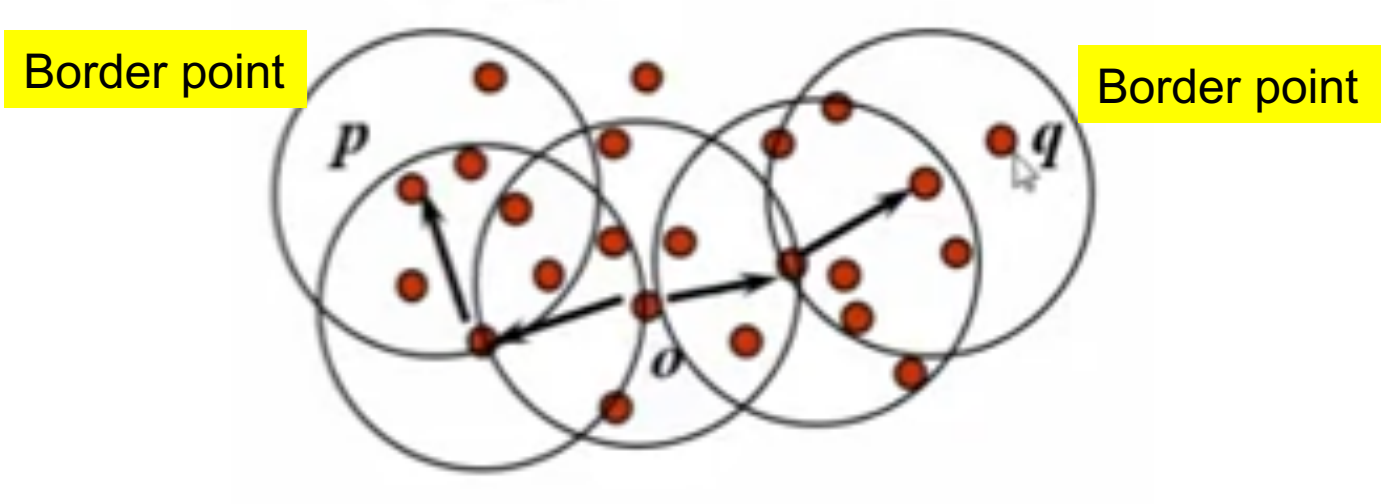
DBSCAN
- Density-reachable

- This density-reachable relation is transitive.
- Density-reachable:
  - a point p is density-reachable from a point q w.r.t. Eps and $MinPts$ if
    - there is a chain of points p_1, ..., p_n, $p_1 = q$, $p_n = p$, such that p_{i+1} is directly density-reachable from p_i
- Density-reachability is a canonical extension of direct density-reachability.



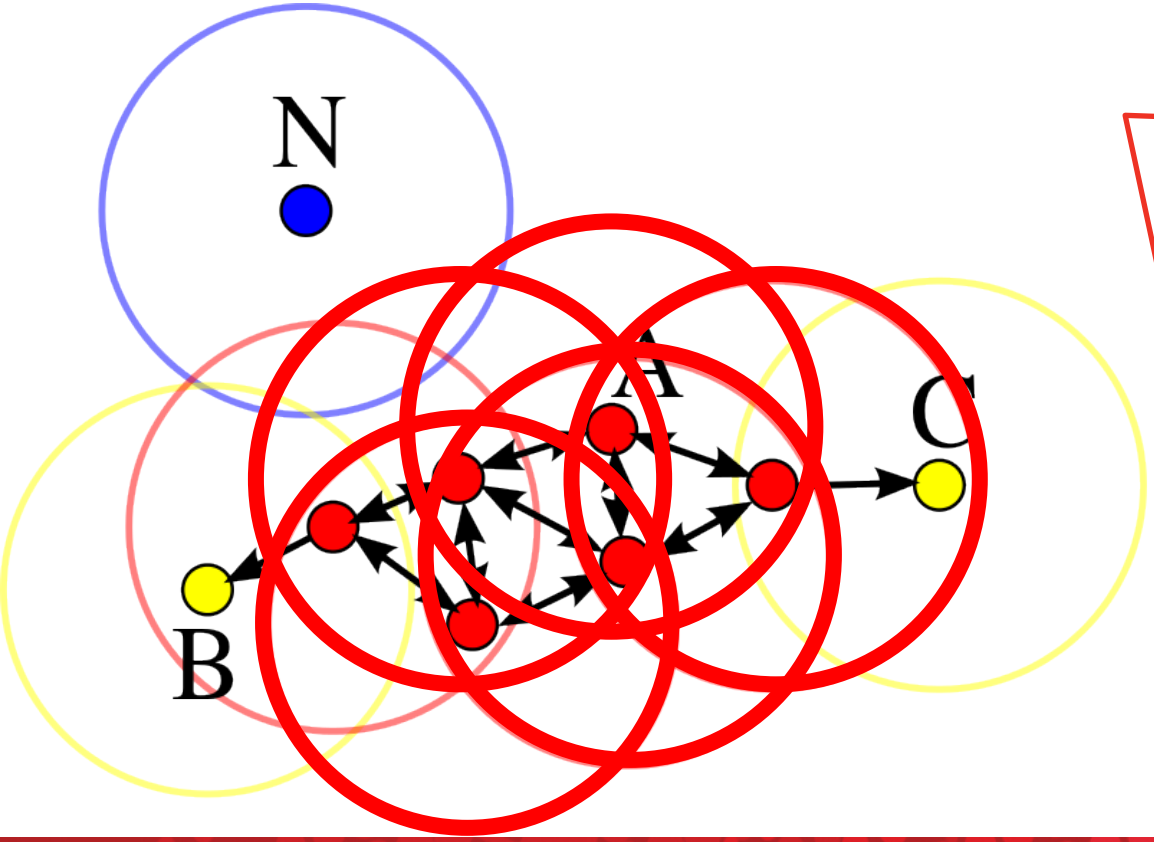
DBSCAN
-Density-Connectivity

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



DBSCAN
- Cluster and Noise

- A cluster:
  - Includes all density-connected data points.
- Points not reachable from any other point are noise.



1. Point A and other red points are core points: because the area surrounding these points in an *Eps* radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster.
2. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well.
3. Point N is a noise point that is neither a core point nor density-reachable.

https://en.wikipedia.org/wiki/DBSCAN

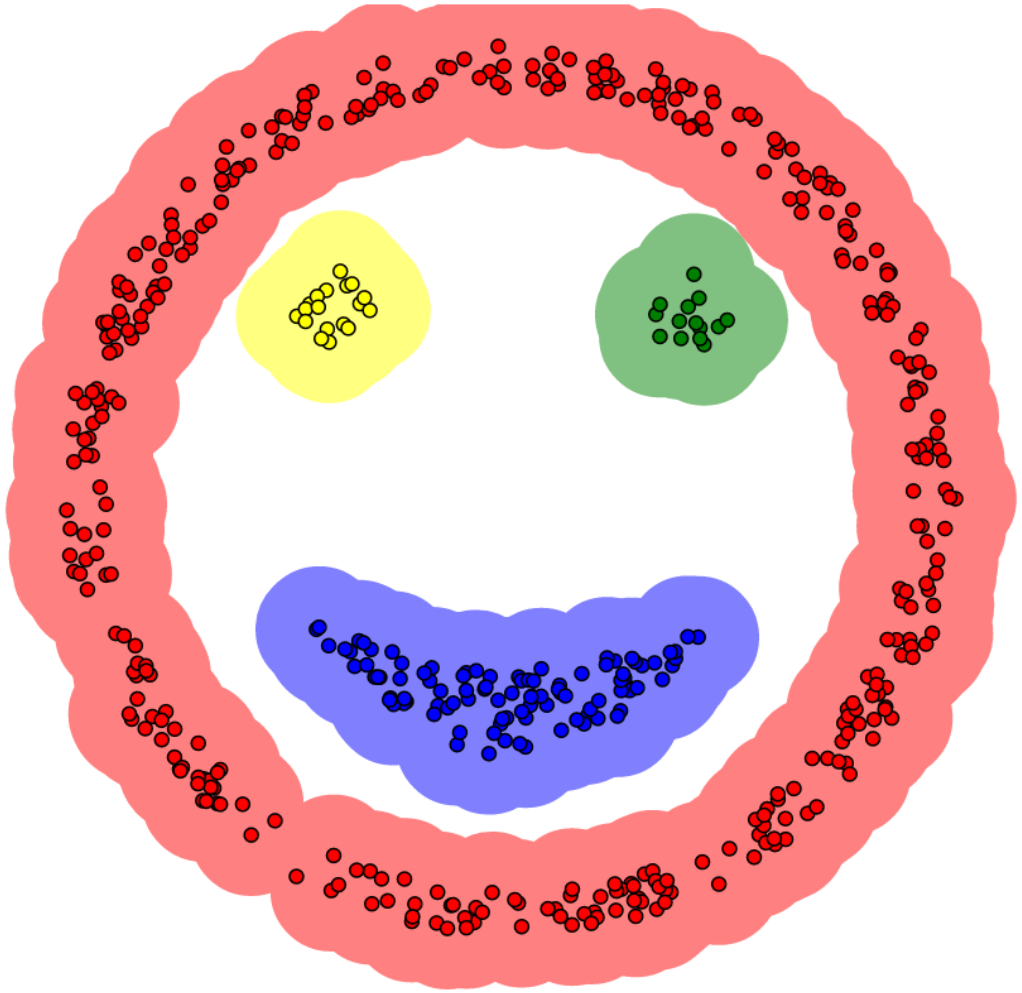
DBSCAN Algorithm

- Arbitrarily select a point p;
- Retrieve all points density-reachable from p w.r.t. Eps 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 in the dataset;
- Continue the process until all of the points have been processed.

https://en.wikipedia.org/wiki/DBSCAN

DBSCAN Visualisation

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



epsilon = 1.32
minPoints = 5

How to Determine the Parameters? - *Eps* and *MinPts*

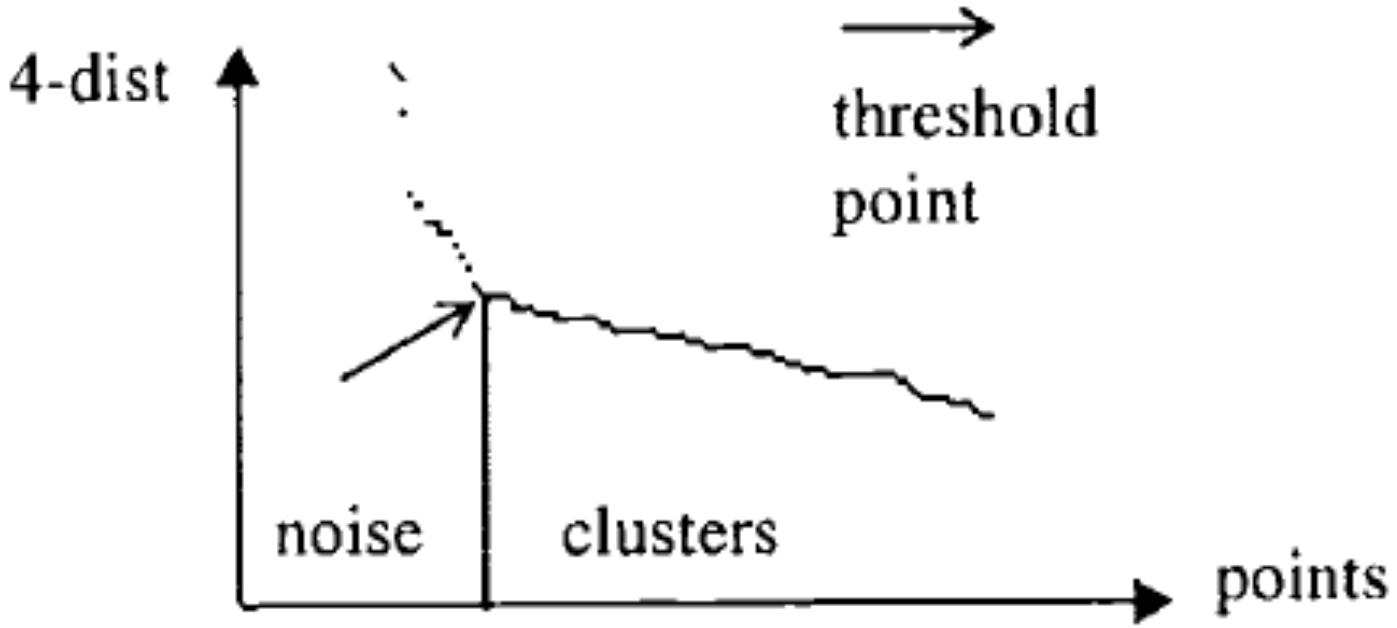
- For DBSCAN, the parameters *Eps* and *minPts* are needed.
  - The parameters must be specified by the user.
  - Ideally,
    - the value of *Eps* is given by the problem to solve (e.g. a physical distance), and
    - *minPts* is then the desired minimum cluster size.
- *MinPts*:
  - As a rule of thumb, a minimum *minPts* can be derived from the number of dimensions D in the data set, as $minPts \geq D + 1$.
  - The low value of $minPts = 1$ does not make sense, as then every point on its own will already be a cluster.
  - Therefore, *minPts* must be chosen to be at least 3.
  - However, larger values are usually better for data sets with noise and will yield more significant clusters.
  - The larger the data set, the larger the value of *minPts* that should be chosen.

How to Determine the Parameters? - *Eps* and *MinPts*

- *Eps*:
  - The value for *Eps* can then be chosen by using a k-distance graph,
    - plotting the distance to the $k = minPts$ nearest neighbor.
  - Good values of *Eps* are where this plot shows a strong bend:
    - If *Eps* is chosen much too small, a large part of the data will not be clustered;
    - whereas for a too high value of *Eps*, clusters will merge and the majority of objects will be in the same cluster.
    - In general, small values of *Eps* are preferable
      - as a rule of thumb only a small fraction of points should be within this distance of each other.

How to Determine the Parameters?

- *k distance graph*



Example of sorted 4-dist graph

DBSCAN
- The Shape of Neighborhood

- The shape of a neighborhood is determined by the choice of a distance function.
  - For instance, when using the Manhattan distance in 2D space, the shape of the neighborhood is rectangular.
- Note, DBSCAN works with any distance function so that an appropriate function can be chosen for some given application.
- Distance function:
  - The choice of distance function is tightly coupled to the choice of Eps, and has a major impact on the results.
  - In general, it will be necessary to first identify a reasonable measure of similarity for the data set, before the parameter Eps can be chosen.

Practical Data Science – COSC2670

PART 2:
DBSCAN IN *SKLEARN*

DBSCAN in sklearn

sklearn.cluster.DBSCAN(eps, min_samples, metric)

- **eps** : float, optional
  - The maximum distance between two samples for them to be considered as being in the same neighborhood.
- **min_samples** : int, optional
  - The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.
- **metric** : string, or callable
  - The metric to use when calculating distance between instances in a feature array. If metric is a string or callable, it must be one of the options allowed by metrics.pairwise.calculate_distance for its metric parameter. If metric is “precomputed”, X is assumed to be a distance matrix and must be square. X may be a sparse matrix, in which case only “nonzero” elements may be considered neighbors for DBSCAN.

DBSCAN in sklearn

Load packages.

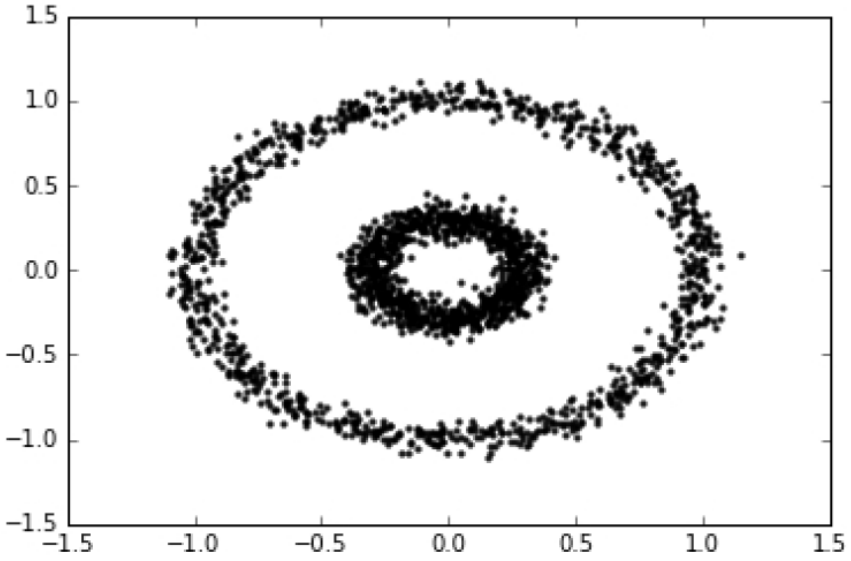
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

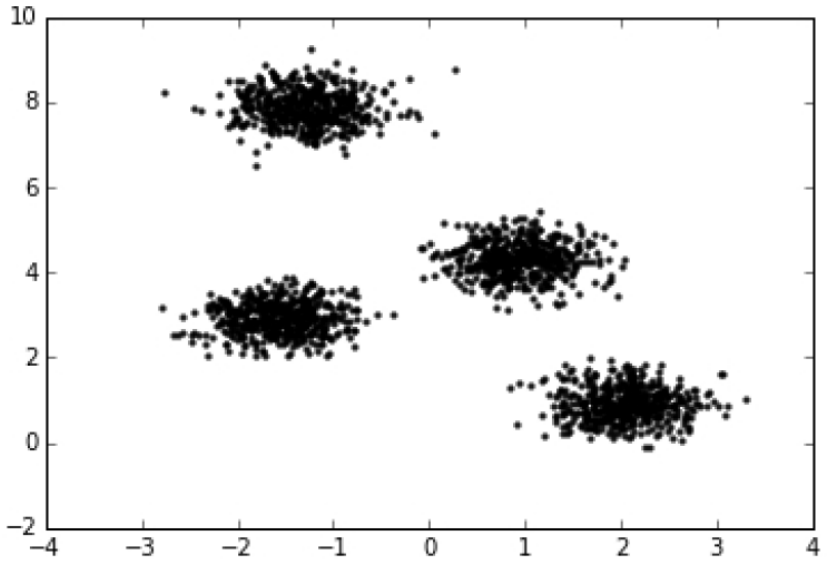
Generate the two dummy datasets by using *sklearn.datasets.make_circles* and *sklearn.datasets.make_blobs*

```
N_samples = 2000
dataset_1 = np.array(datasets.make_circles(n_samples=N_samples, noise=0.05, factor=0.3)[0])
dataset_2 = np.array(datasets.make_blobs(n_samples=N_samples, centers=4, cluster_std=0.4, random_state=0)[0])
```

```
plt.scatter(dataset_1[:,0], dataset_1[:,1], c='k', alpha=0.8, s=5.0)
plt.show()
plt.scatter(dataset_2[:,0], dataset_2[:,1], c='k', alpha=0.8, s=5.0)
plt.show()
```

Plot each dataset *matplotlib.pyplot.scatter*





DBSCAN in sklearn

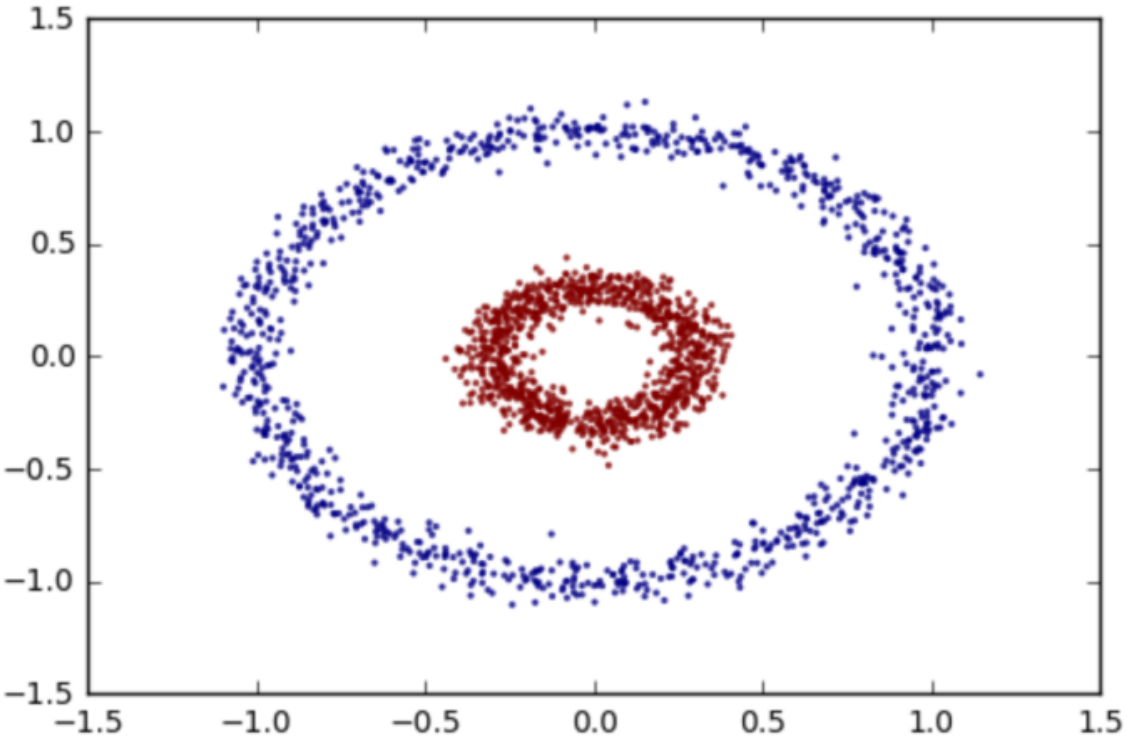
```
from sklearn.cluster import DBSCAN
dbs_1 = DBSCAN(eps=0.3)
labels_1 = dbs_1.fit(dataset_1).labels_

plt.scatter(dataset_1[:,0], dataset_1[:,1], c=labels_1, alpha=0.8, s=5.0, lw = 0)
plt.show()

labels_1
```

Load packages.

Build DBSCAN model

Fit the model and get the labels of training data in dataset_1

Plot the clustering results by using labels_1 as color index



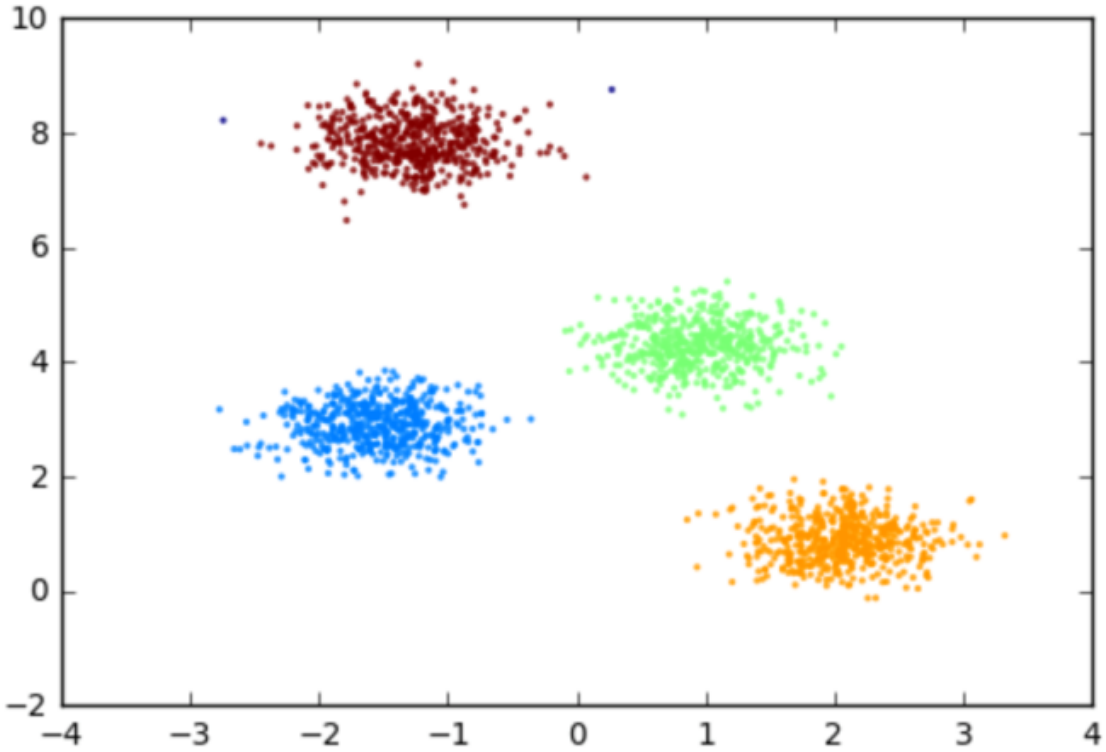
DBSCAN in sklearn

```
dbs_2 = DBSCAN(eps=0.5)
labels_2 = dbs_2.fit(dataset_2).labels_

plt.scatter(dataset_2[:,0], dataset_2[:,1], c=labels_2, alpha=0.8, s=5.0, lw = 0)
plt.show()
```

Build DBSCAN model

Fit the model and get the labels of training data in dataset_1

Plot the clustering results by using labels_1 as color index



DBSCAN on Iris Dataset

```
import pandas as pd
from sklearn import datasets

data = datasets.load_iris()
X = pd.DataFrame(data.data, columns = list(data.feature_names))
X.head(5)

from sklearn import cluster
model = cluster.DBSCAN(eps=0.4)
results = model.fit(X)
X['cluster'] = results.labels_
X['target'] = data.target
X['c'] = 'lookatmeIamimportant'
X.head(5)

classification_result = X[['cluster','target','c']].groupby(['cluster', 'target']).agg('count')
print(classification_result)

import matplotlib.pyplot as plt

plt.scatter(X.ix[:,0], X.ix[:,1], c=results.labels_, alpha = 0.8)
plt.show()
```

Load packages, datasets

Load cluster package
Build the DBSCAN model
Fit the model

Get the labels of the training points and put in in an additional column in X

An dummy column for counting purposes later

Three parts in this line: 1) select the cluster, target, and c columns; 2) group data by the cluster and target column; 3) aggregate the rows of the group with a simple count aggregation

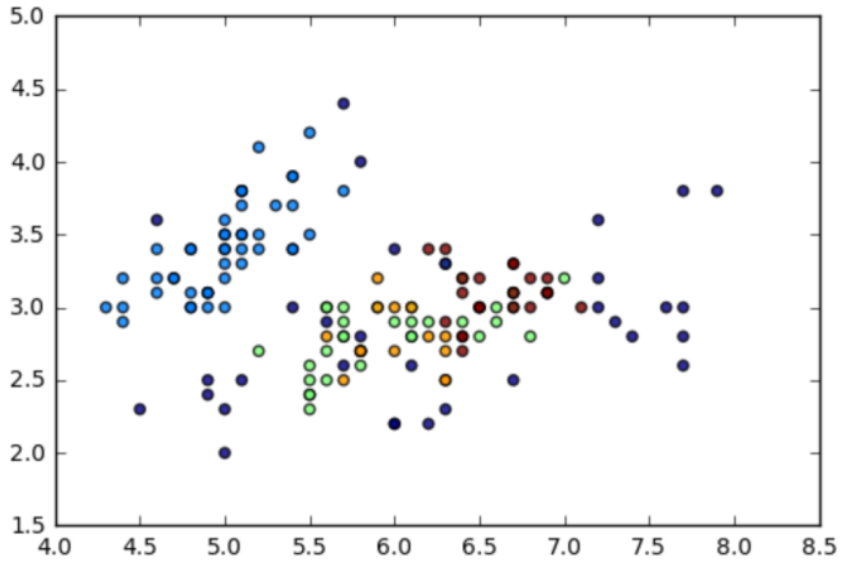
Plot the clustering results

DBSCAN on Iris Dataset

Plot the clustering results

```
import matplotlib.pyplot as plt

plt.scatter(X.ix[:,0], X.ix[:,1], c=results.labels_, alpha = 0.8)
plt.show()
```

cluster	target	c
-1	0	4
	1	11
	2	17
0	0	46
1	1	36
2	1	3
	2	11
3	2	22



Determine *Eps*
- *k* Distance Graph in sklearn

```
from sklearn.neighbors import NearestNeighbors

nbrs = NearestNeighbors().fit(X)
distances, indices = nbrs.kneighbors(dataset_1, 20)
kDis = distances[:, 10]
kDis.sort()
kDis = kDis[list(range(len(kDis) - 1, 0, -1))]
plt.plot(list(range(0, len(kDis), kDis)))
plt.show()

from sklearn.cluster import DBSCAN

dbs_1 = DBSCAN(eps=0.08, min_samples=11)
labels_1 = dbs_1.fit(dataset_1).labels_
plt.scatter(dataset_1[:, 0], dataset_1[:, 1], c=labels_1, alpha=0.8, s=5.0, lw=0)
plt.show()
```

Load NearestNeighbors package: implementation of neighbor searches.

Fit the model

Select the first 20 nearest neighbors

Select the distances of the 11-th neighbors of all data points

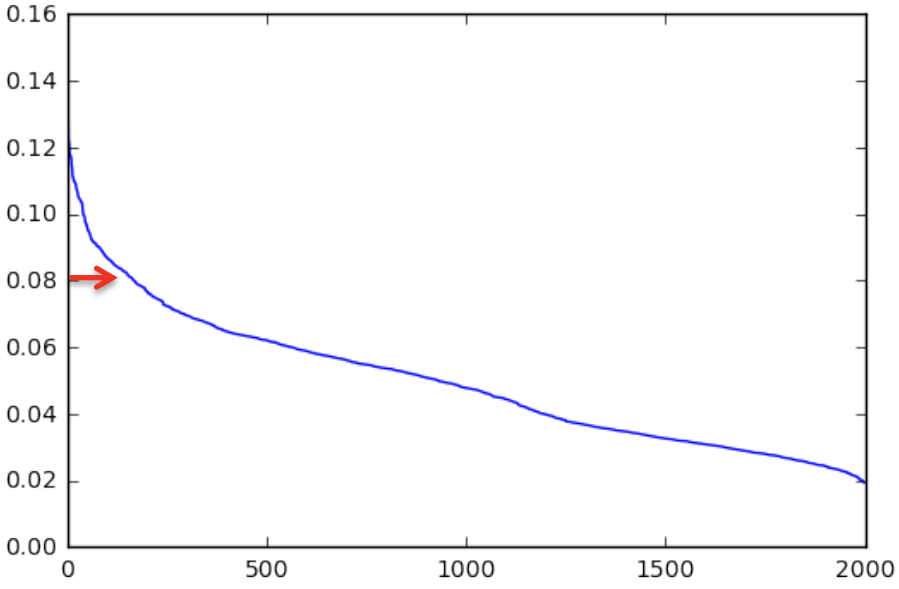
Sort in ascending order, then order them in descending order by using reverse index

Then, plot the k distance graph

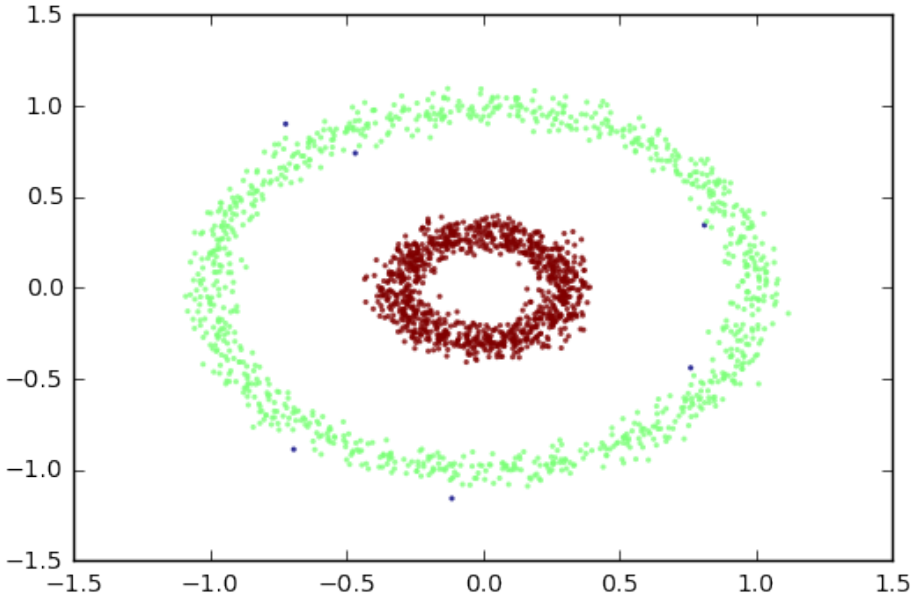
Find the right value for Eps, then run DBSCAN

Plot the clustering results

Determine *Eps*
- *k* Distance Graph in sklearn

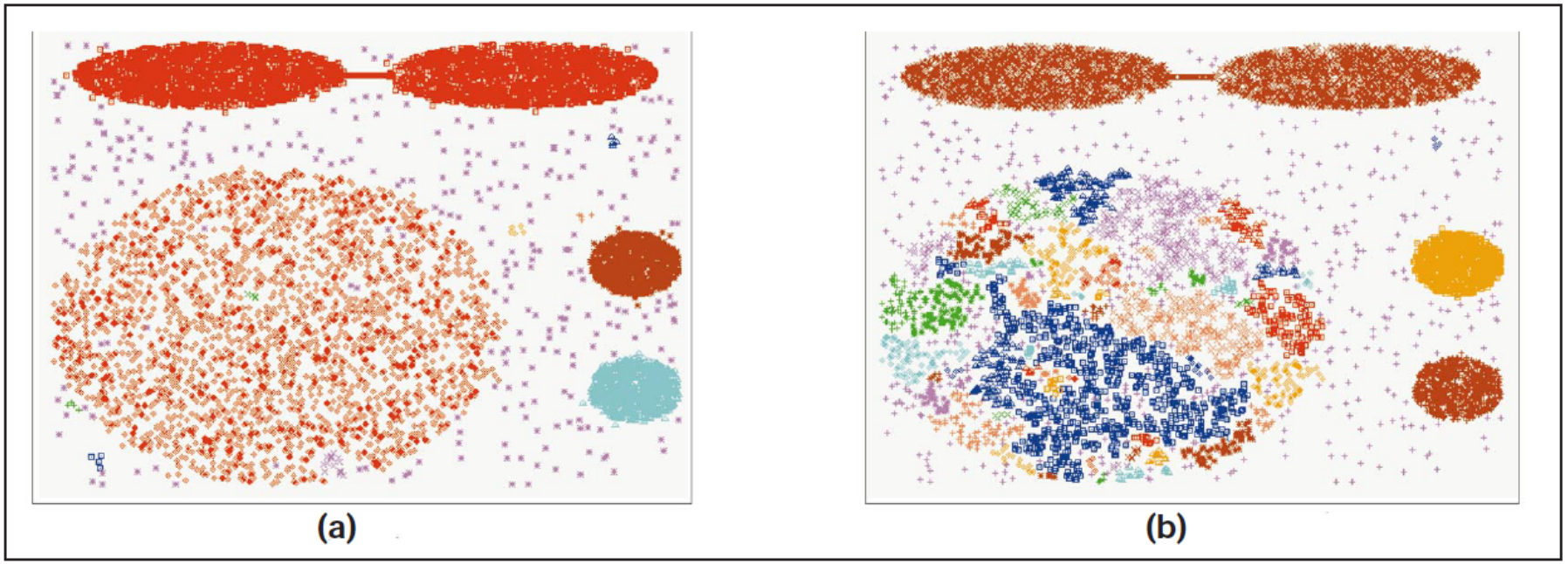


k-distance graph



Clustering results

Sensitive To Parameters

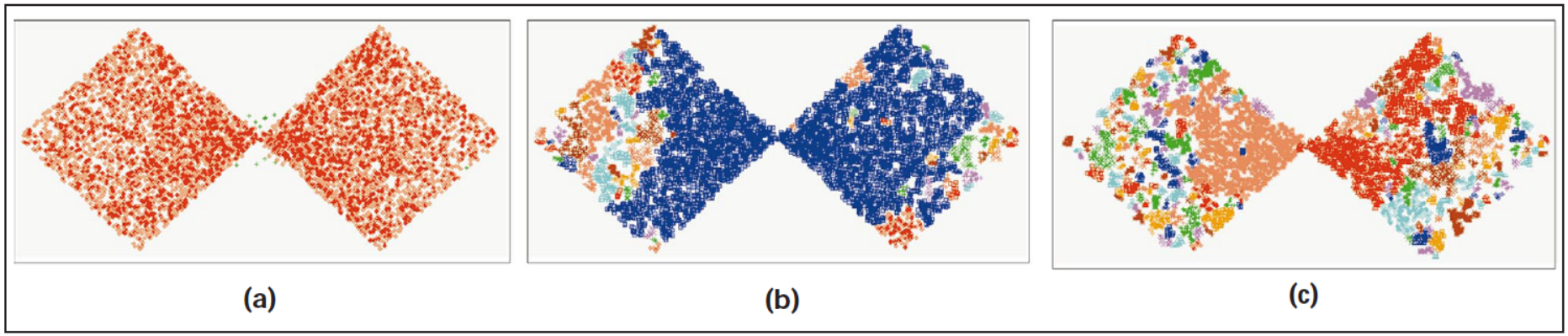


Eps = 0.5

Eps = 0.4

George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. 1999

Sensitive To Parameters



Eps = 0.5 *Eps* = 0.35 *Eps* = 0.3

Advantage

- DBSCAN does not require users to specify the number of clusters in the data a priori, as opposed to k-means.
- DBSCAN can find arbitrarily shaped clusters.
  - It can even find a cluster completely surrounded by (but not connected to) a different cluster.
- DBSCAN has a notion of noise, thus is robust to outliers.
- DBSCAN requires just two parameters.
- DBSCAN is mostly insensitive to the ordering of the points in the database.
- DBSCAN is designed for use with databases that can accelerate region queries.
- The parameters $minPts$ and Eps can be set by a domain expert, if the data is well understood.

Disadvantage

- DBSCAN is not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed.
  - Fortunately, this situation does not arise often, and has little impact on the clustering result.
- The quality of DBSCAN depends on the distance measure
  - The most common distance metric used is Euclidean distance.
  - Especially for high-dimensional data, this metric can be rendered almost useless due to the so-called "Curse of dimensionality", making it difficult to find an appropriate value for Eps.
    - When using many coordinates, there is *little difference* in the distances between different pairs of samples.
  - This effect, however, is also present in any other algorithm based on Euclidean distance.

Disadvantage

- DBSCAN cannot cluster data sets well with large differences in densities,
  - because the *minPts*-*Eps* combination cannot then be chosen appropriately for all clusters.
- If the data and scale are not well understood, choosing a meaningful distance threshold *Eps* can be difficult.

References and Further Reading

- A. Boschetti and L. Massaron, *Python Data Science Essentials*, Chapters 4 and 6
- Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M., eds. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. CiteSeerX 10.1.1.121.9220Freely accessible. ISBN 1-57735-004-9.
- George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. 1999. Chameleon: Hierarchical Clustering Using Dynamic Modeling. Computer 32, 8 (August 1999), 68-75.

Data
Science

Thanks!