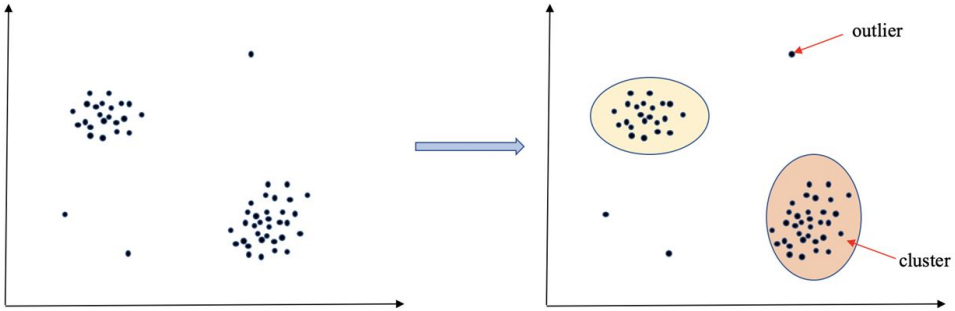
Large-scale Clustering

Group 4: Charles Radke, Jiecheng Song, Jiajun Gao

Content

Introduction

Clustering



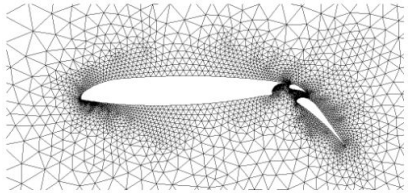
Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more **similar** to other data points in the **same group** and **dissimilar** to the data points in **other groups**. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

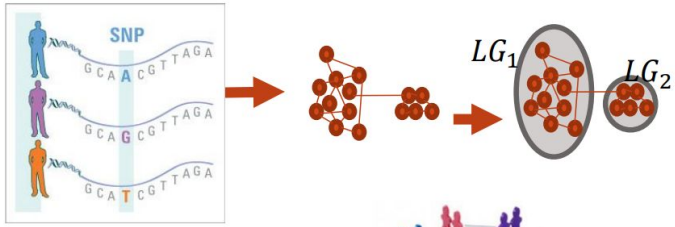
Source:https://www.geeksforgeeks.org/clustering-in-machine-learning/

Applications

For example:

- Human genetic clustering
- Grouping of shopping items
- E-commerce
- Social network analysis
- Recommender systems
- Crime analysis
- image segmentation





Challenges in the Large-Scale Setting

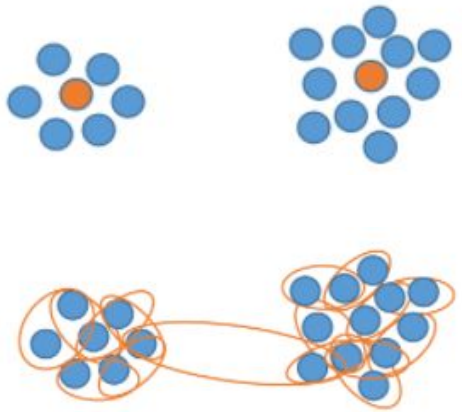
Big Data: Many popular clustering algorithms are inherently difficult to parallelize, and inefficient at large scales

The computational issue becomes critical when the application involves large-scale data.

There is no clustering algorithm that can be universally used to solve all problems.

Popular Clustering Techniques

- *K-means*
  - k-means/ k-medians/ k-medoids
  - BFR
- *Hierarchical*
  - Single Linkage
  - BIRCH
  - CURE
- *Others*
  - Spectral
  - DBSCAN
  - ...



Source: https://sites.cs.ucsb.edu/~veronika/MAE_LargeScaleClustering_strnadova.pdf

Distance Measurements

Euclidean Space

Let's consider d-dimensional Euclidean space. Given two random points $[x_1, x_2, \cdots, x_d]$ and $[y_1, y_2, \cdots, y_d]$

$$Euclidean\ Distance = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \cdots + |x_d - y_d|^2}$$

$$Manhattan\ Distance = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_d - y_d|$$

$$Minkowski\ Distance = \sqrt[q]{|x_1 - y_1|^q + |x_2 - y_2|^q + \cdots + |x_d - y_d|^q}$$

Other types of distance

cosine distance: vectors

Jaccard distance: sets

edit distance: strings

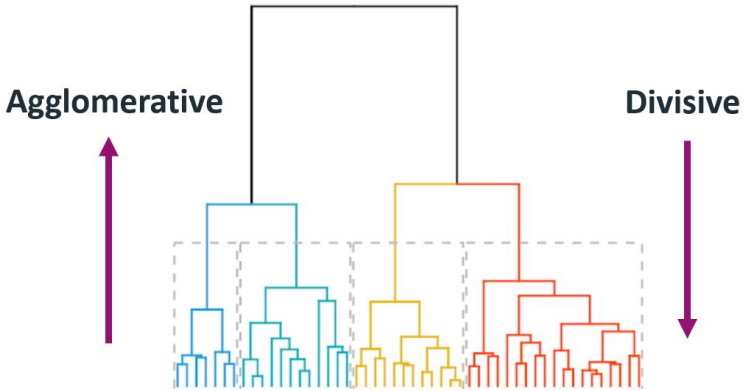
Hierarchical Clustering

Hierarchical Clustering

A clustering algorithm that repeatedly merges (**agglomerative**) two close clusters or splits (**divisive**) a cluster into two smaller ones.

Key questions:

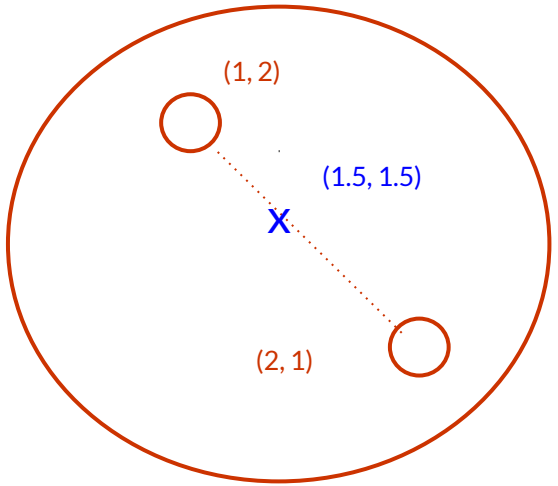
1. How can a cluster be represented?
2. How is cluster closeness measured?
3. When will clusters stop being merged?



Hierarchical Clustering in Euclidean Space

(1) How is a cluster with more than one point represented?

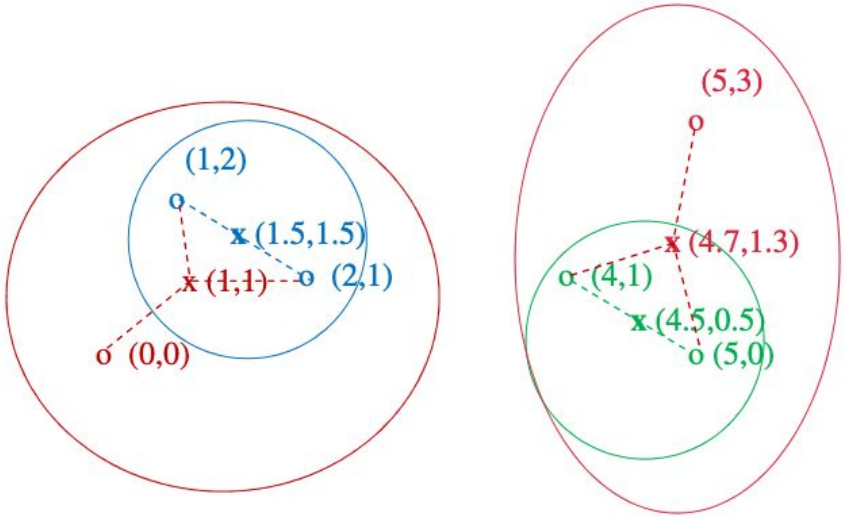
Assign **centroids** - average of points in each cluster



Hierarchical Clustering in Euclidean Space

(2) How is closeness measured?

Calculate the **distances** between centroids

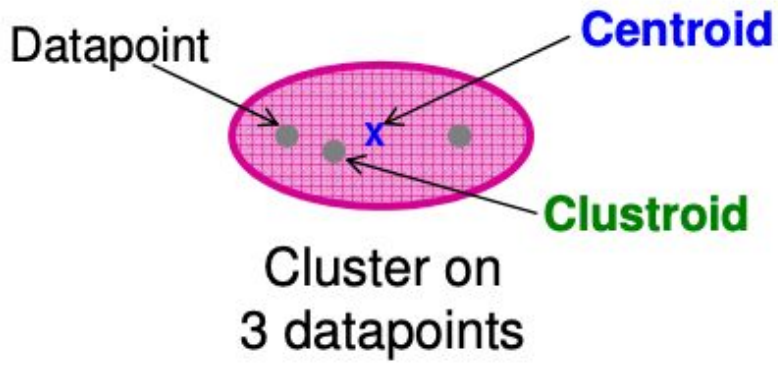


Hierarchical Clustering in Non-Euclidean Space

(1) How is a cluster with more than one point represented?

Assign **clustroids** - points “closest” to the other point

- ➔ Distances minimax
- ➔ Minimum average distance
- ➔ Smallest sum of squares of distances



Hierarchical Clustering in Non-Euclidean Space

(2) How is cluster closeness measured?

Three Approaches:

1. Compute **clustroid distances** between clusters

2. Calculate the **intercluster distance** - minimum distance between two points, each residing in a unique cluster

3. Combine clusters that is most "**cohesive**", can be measured by…
   - ➔ the maximum distances between two cluster points ("diameter")
   - ➔ the average distance between cluster points
   - ➔ the diameter or average distance divided by number of points ("density")

Hierarchical Clustering in Either Space

(3) When will clusters stop being merged?

Cluster splitting/merging can be halted after meeting some predetermined condition

Stop if…

- ➔ a merge/split results in a density/diameter/avg. distance exceeding some threshold
- ➔ we have reached “k” clusters
- ➔ there is evidence of a merge/split creating a poor cluster
  ie) average diameter of a cluster merge decreases instead of increasing

Hierarchical Clustering Complexity

For a dataset consisting of **N points**

Less efficient: Naive implementation of hierarchical clustering:

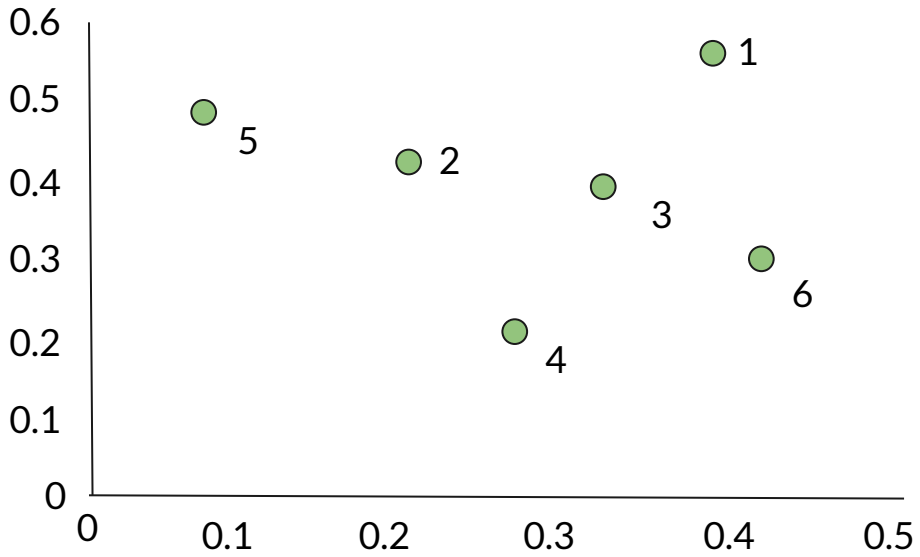
- ➔ Each step, compute all distances between all cluster pairs to merge accordingly
- ➔ O(N^3) time

More efficient: Priority queue implementation

- ➔ Each cluster has its own priority queue which identifies cluster similarities
- ➔ O(N^2 log N) time

Regardless of method, still **restrained to smaller datasets** / lower dimensionality

Hierarchical Clustering: Example



	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

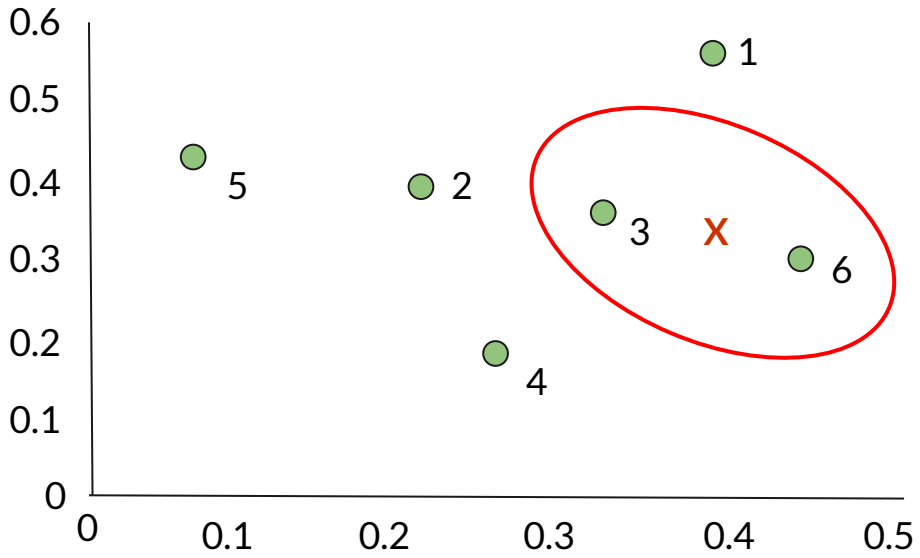
Hierarchical Clustering: Example

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.24	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0

Hierarchical Clustering: Example

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.24	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	**0.11**	0.22	0.39	0

Hierarchical Clustering: Example



	X	Y
P1	0.40	0.53
P2	0.22	0.38
(P3,P6)	0.40	0.31
P4	0.26	0.19
P5	0.08	0.41

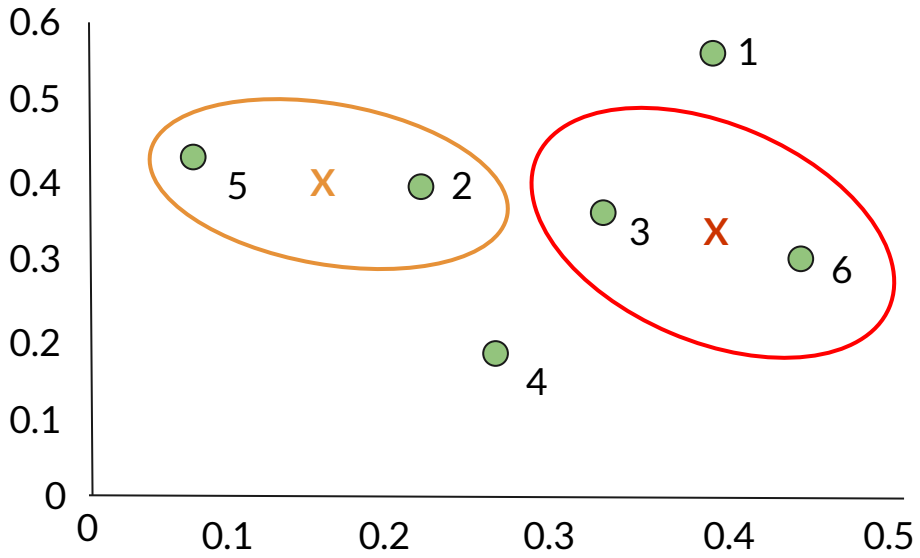
Hierarchical Clustering: Example

	P1	P2	(P3, P6)	P4	P5
P1	0				
P2	0.23	0			
(P3,P6)	0.22	0.19	0		
P4	0.37	0.20	0.18	0	
P5	0.34	0.14	0.28	0.34	0

Hierarchical Clustering: Example

	P1	P2	(P3, P6)	P4	P5
P1	0				
P2	0.23	0			
(P3,P6)	0.22	0.19	0		
P4	0.37	0.20	0.18	0	
P5	0.34	**0.14**	0.28	0.34	0

Hierarchical Clustering: Example



	X	Y
P1	0.40	0.53
(P2, P5)	0.15	0.40
(P3,P6)	0.40	0.31
P4	0.26	0.19

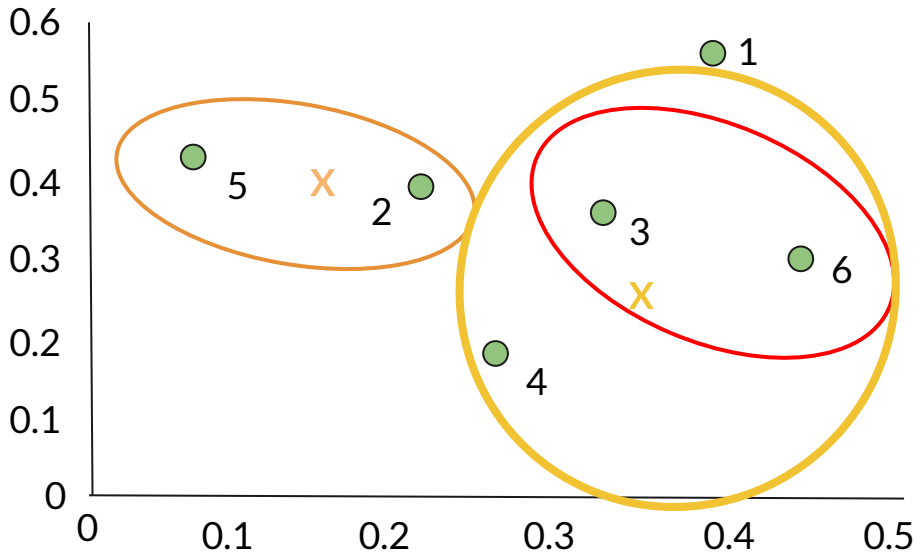
Hierarchical Clustering: Example

	P1	(P2, P5)	(P3, P6)	P4
P1	0			
(P2, P5)	0.28	0		
(P3,P6)	0.22	0.27	0	
P4	0.37	0.24	0.18	0

Hierarchical Clustering: Example

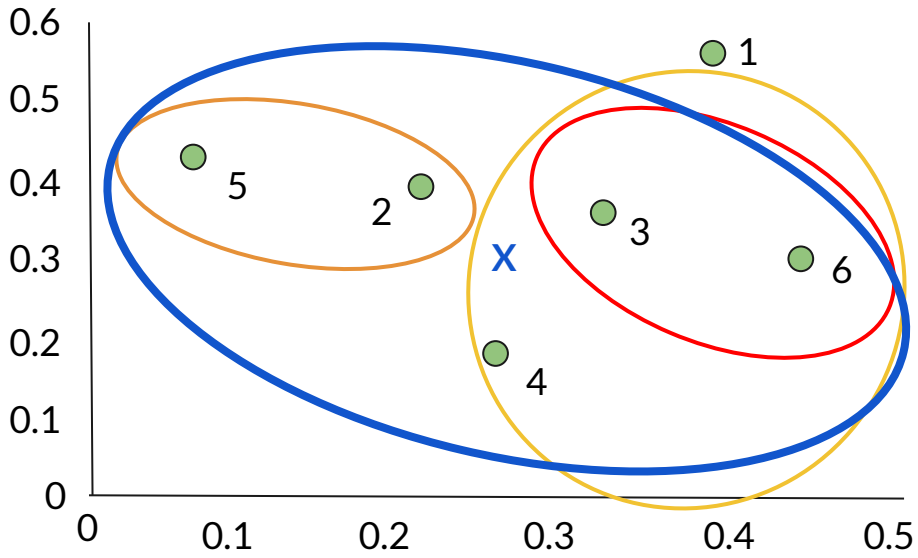
	P1	(P2, P5)	(P3, P6)	P4
P1	0			
(P2, P5)	0.28	0		
(P3,P6)	0.22	0.27	0	
P4	0.37	0.24	**0.18**	0

Hierarchical Clustering: Example



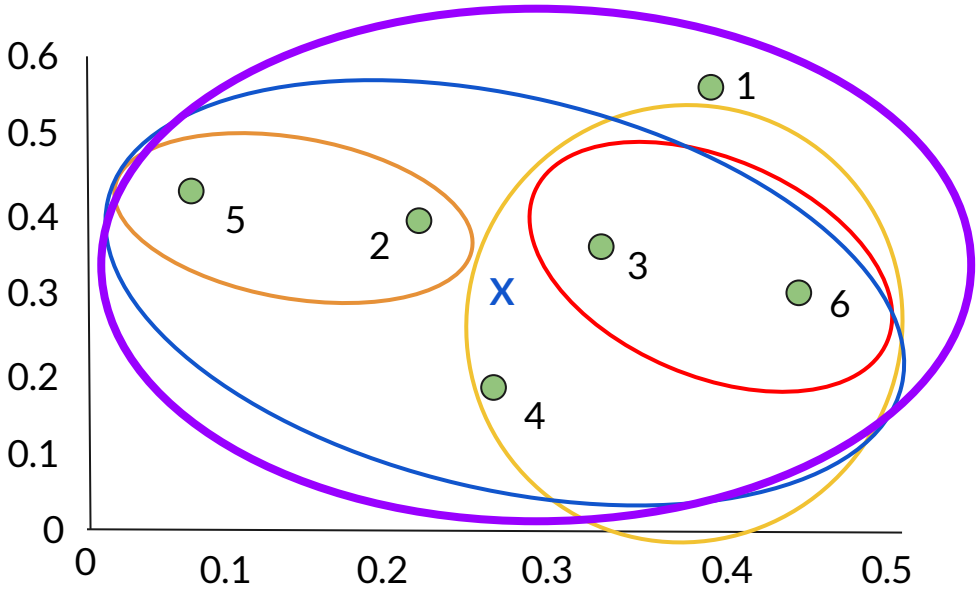
	X	Y
P1	0.40	0.53
(P2, P5)	0.15	0.40
(P3, P4, P6)	0.35	0.27

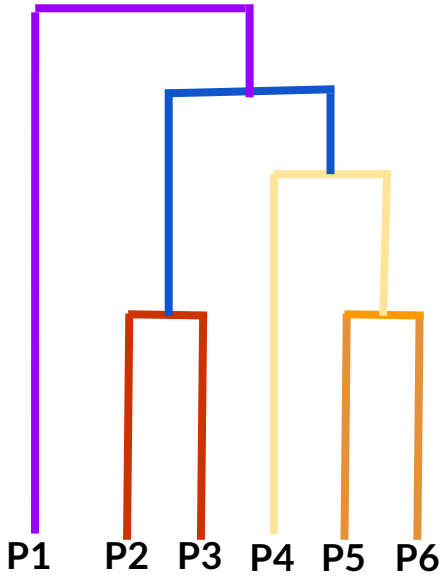
Hierarchical Clustering: Example



	X	Y
P1	0.40	0.53
(P2,P3, P4, P5 ,P6)	0.27	0.31

Hierarchical Clustering: Example



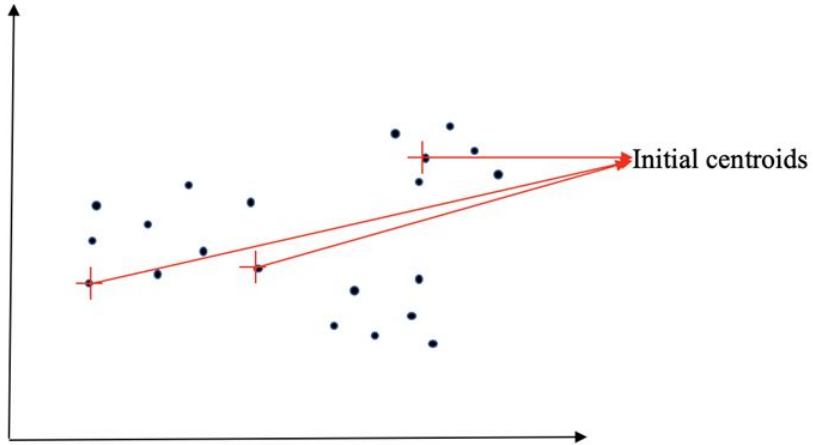


K-Means Clustering

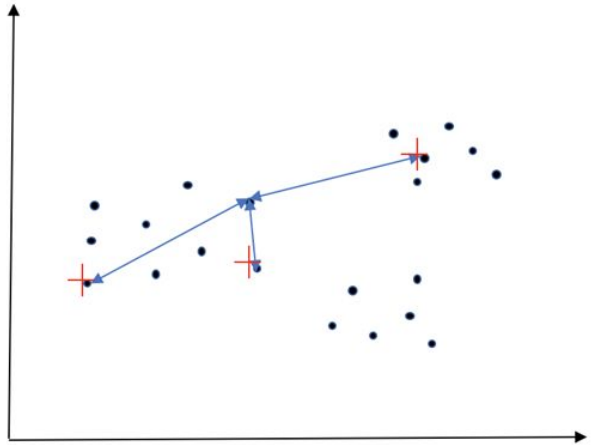
K-Means Algorithm

1. Picking k **initial centroids** of cluster
2. For each point, place it in the cluster whose current centroid is the nearest
3. After all points are assigned, update the locations of centroid for each cluster
4. Reassign all points to their closest centroid form new clusters
5. Repeat 3 and 4 **until convergence**

Example K=3

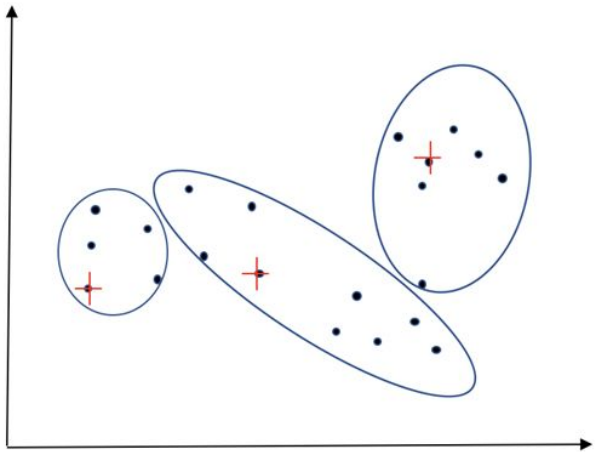


Step 1

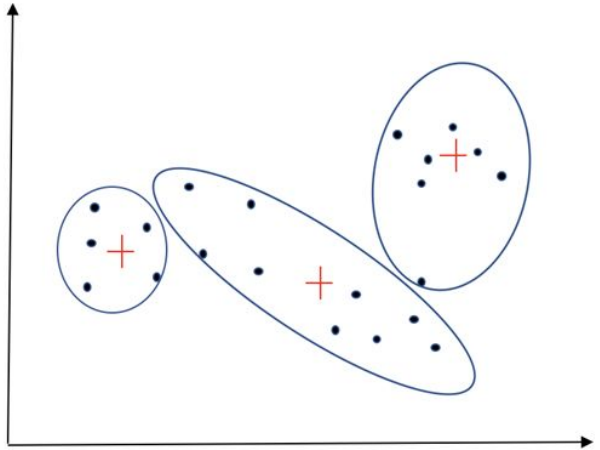


Step 2

Example K=3

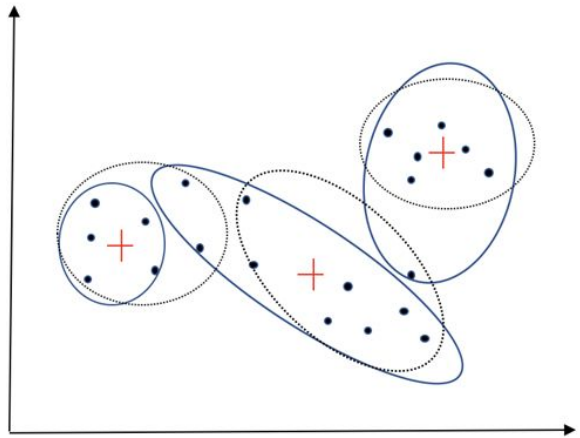


Result of Step 2

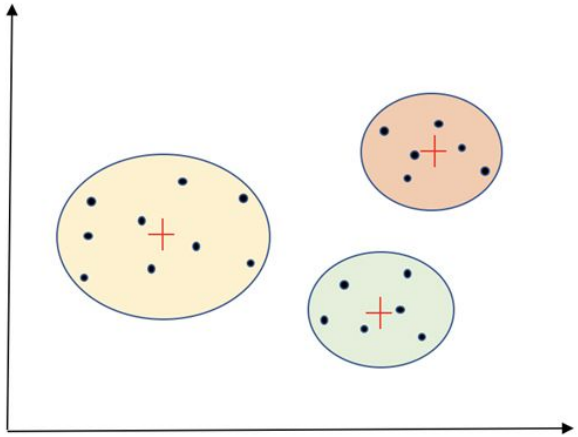


Step 3

Example K=3



Step 4



Final result after repeating 3&4

Pros and Cons of K-Means

Advantages:

- Relatively simple to implement.
- Scales to large data sets. (Time complexity = O(N))
- Guarantees convergence.
- Can warm-start the positions of centroids.
- Easily adapts to new examples.
- Generalizes to clusters of different shapes and sizes, such as elliptical clusters.

Disadvantages:

- Choosing k manually
- Being dependent on initial values
- Clustering data of varying sizes and density
- Clustering outliers

Source: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages

Picking Initial Centroids

Approach 1 : Cluster a small sample of data (e.g. hierarchical clustering) to obtain k clusters. Pick a point from each cluster (e.g. the point closest to the centroid of the cluster)

Approach 2 : Pick the “dispersed” set of points.

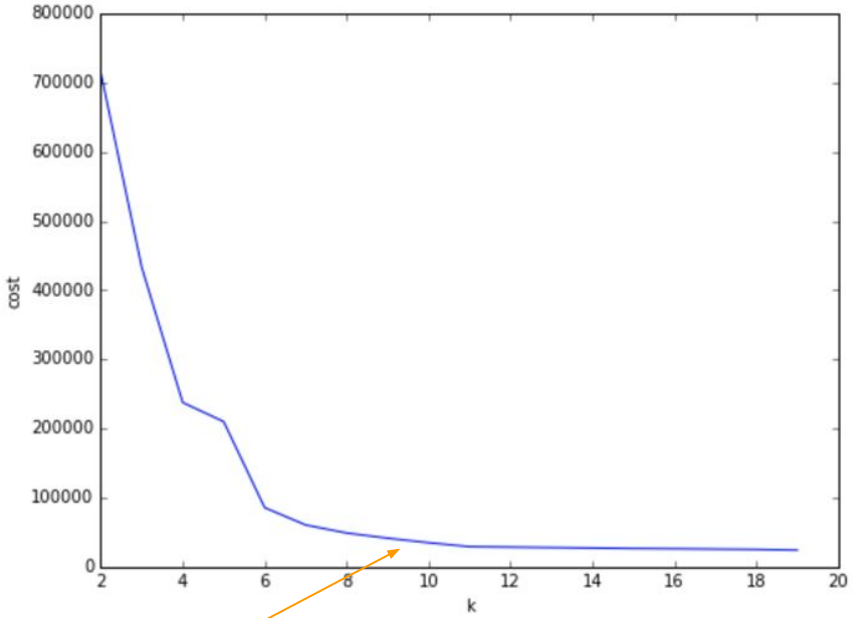
1. Pick the first point at random.
2. Pick the next point whose minimum distance from the selected points is as large as possible.
3. Repeat 2 until we have k points.

K-Means ++ Algorithm

1. Choose one center uniformly at random among the data points.
2. For each data point x, compute D(x), the distance between x and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to D(x)^2.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard k-means clustering.

How to Select K?

1. Try different **k**, looking at the change in the average distance to centroid as **k** increases.
2. Average distance falls rapidly until convergence to get the right k.

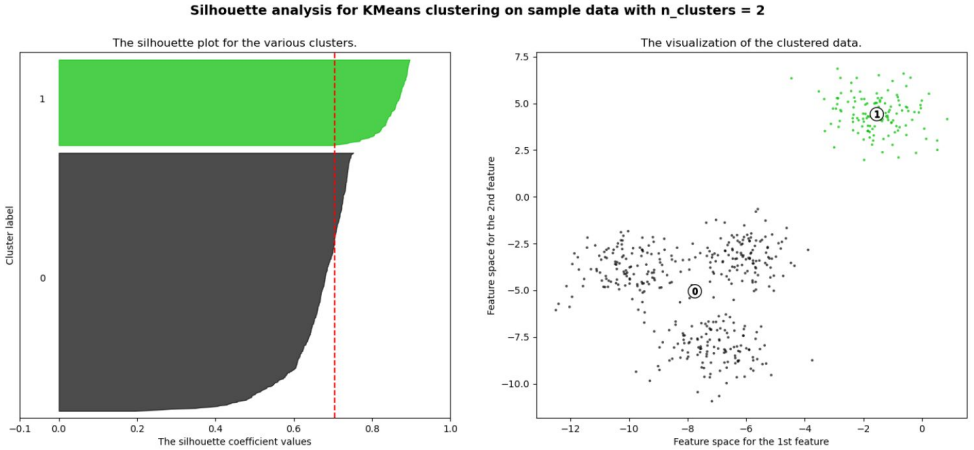


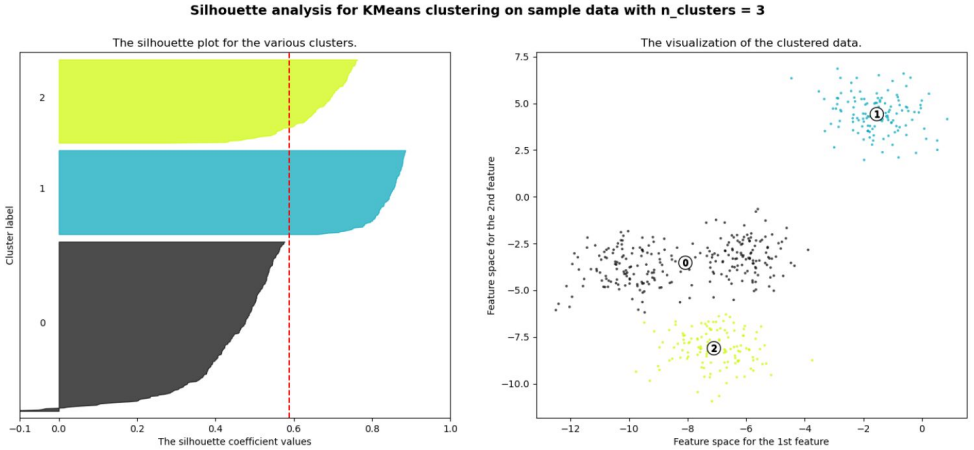
Another Way to Select K

Silhouette

1. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.
2. The silhouette can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance.

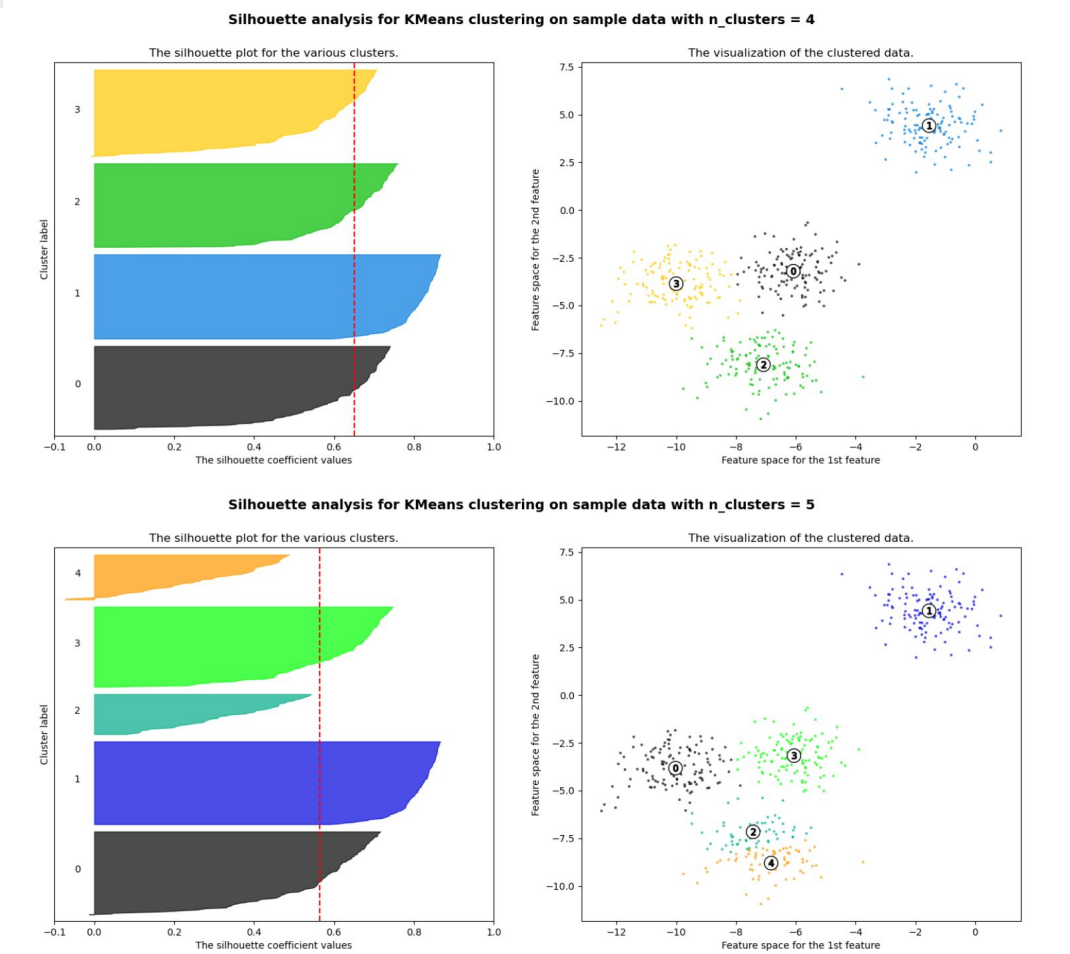
Example
- Silhouette





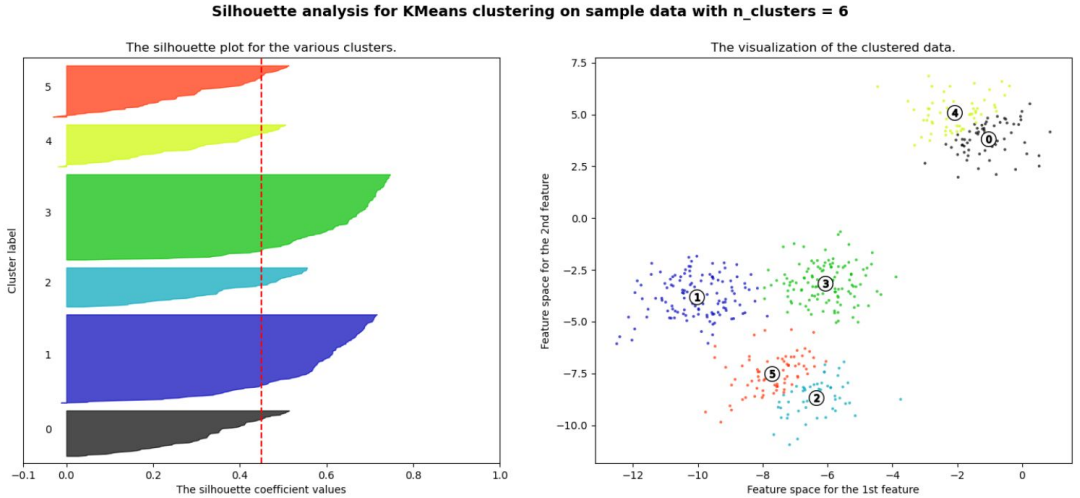
Source:
https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Example
- Silhouette



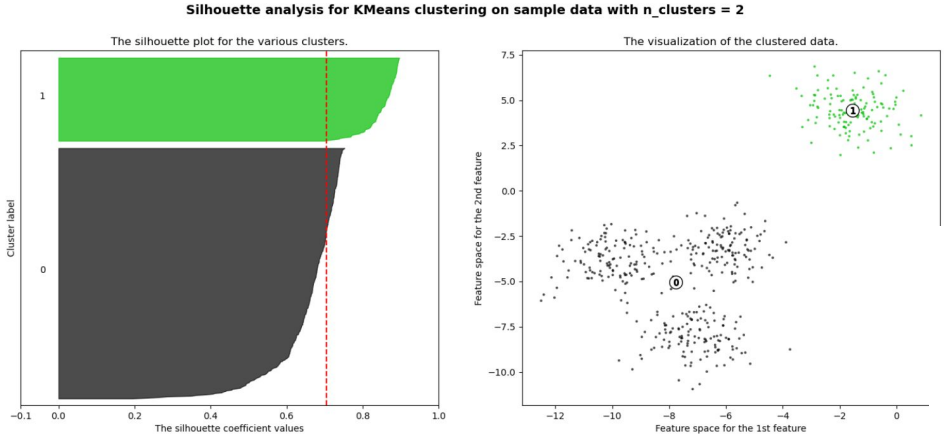
Source:
https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

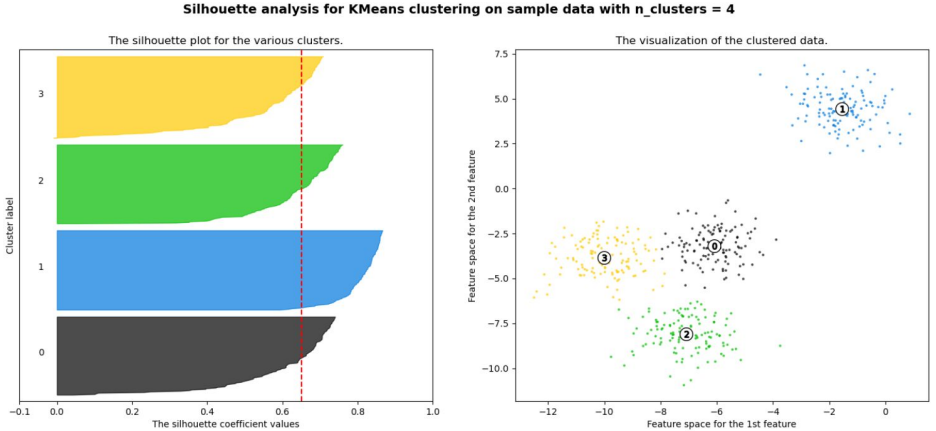
Example
- Silhouette




```
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721
For n_clusters = 4 The average silhouette_score is : 0.6505186632729437
For n_clusters = 5 The average silhouette_score is : 0.56376469026194
For n_clusters = 6 The average silhouette_score is : 0.4504666294372765
```

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






```
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721
For n_clusters = 4 The average silhouette_score is : 0.6505186632729437
For n_clusters = 5 The average silhouette_score is : 0.56376469026194
For n_clusters = 6 The average silhouette_score is : 0.4504666294372765
```

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

Map Reduce for K-Means

Given $x_1, x_2, \cdots, x_n \in R^d$ and k

Initialize $\mu_1, \mu_2, \cdots, \mu_k$ and keep as a global variable.

In each iteration:

1. Map: find optimal assignments of x_i to μ_j by $\min_j ||x_i - \mu_j||_2^2$

   the key value pair is (j, xi)

2. Reduce: $Re-evaluate\ \mu_j\ given\ \mu_j = \frac{1}{n_j} \sum_{key=j} x_i$

Large-Scale K-Means Example - Using PySpark

Dataset

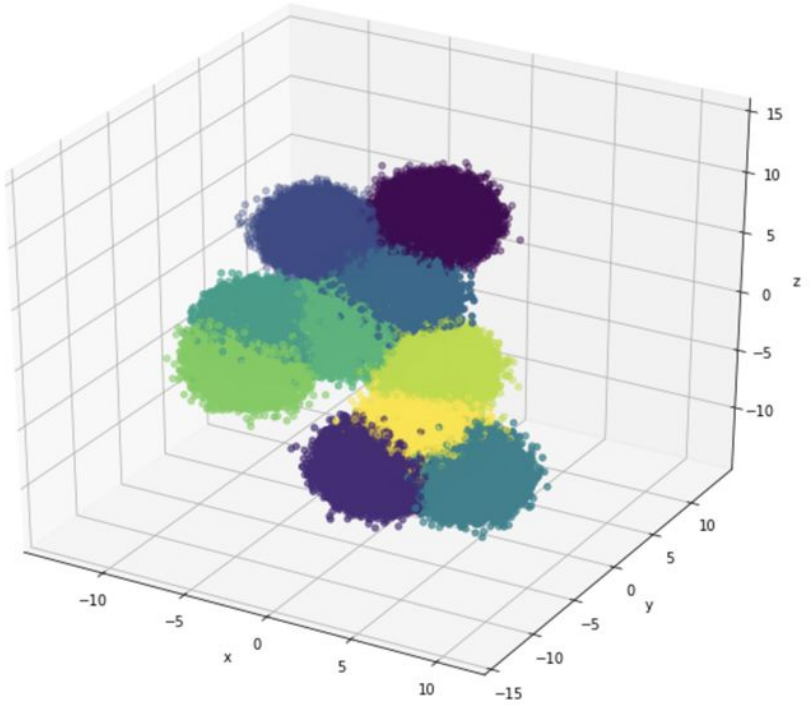
"make_blobs" function was used to generate 10 isotropic Gaussian blobs for clustering (1M rows in total).

	x	y	z	id
0	-0.925077	5.587636	-5.790470	row0
1	-0.209863	2.396010	-8.351261	row1
2	-8.170319	6.774959	2.321300	row2
3	1.583764	-6.613721	-7.266501	row3
4	6.063059	-5.543061	-6.526857	row4
...	...	...	...	...
999995	2.096836	-6.492231	-6.061796	row999995
999996	-2.595222	8.970140	4.528399	row999996
999997	6.086246	-4.899465	-6.866043	row999997
999998	4.003696	-9.901627	9.472756	row999998
999999	-7.014003	-3.534386	-2.252550	row999999

1000000 rows × 4 columns

Visualization

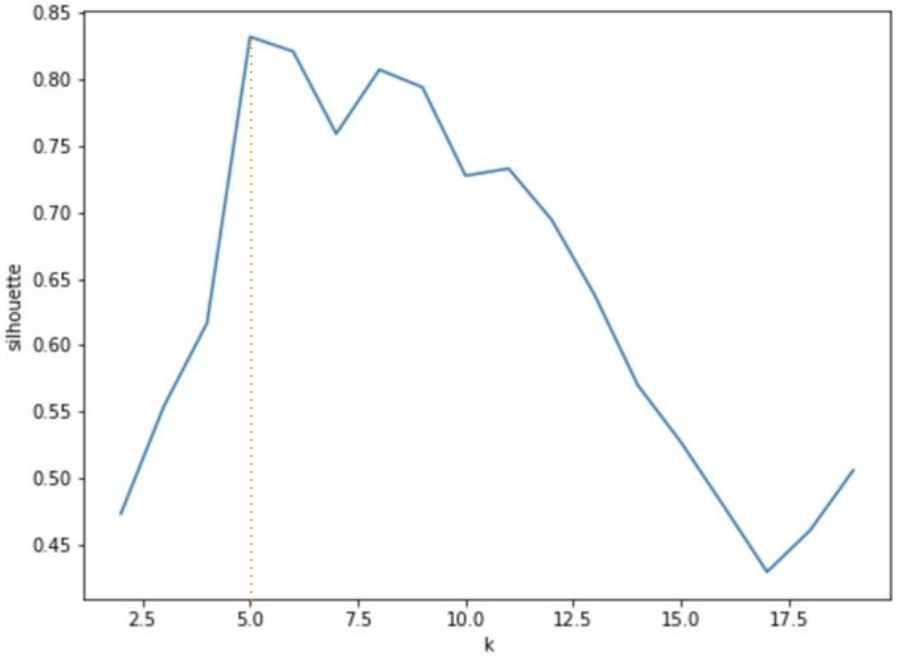
There are 10 different colors representing the true clusters generated by “make_blobs”



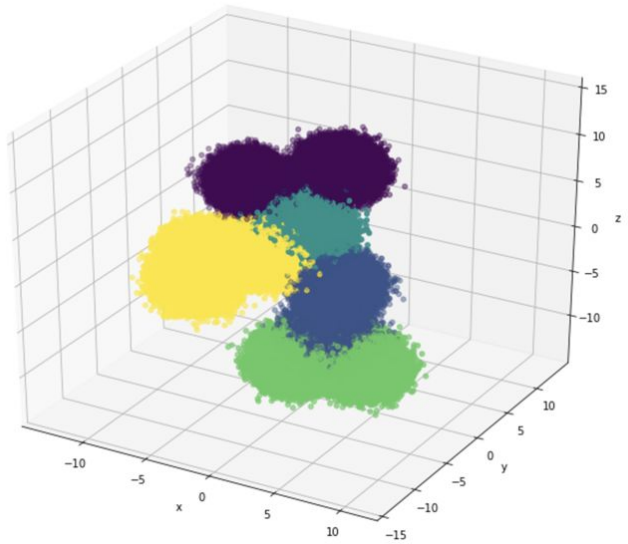
Choose K

“for” loop was used to get the silhouette for k from 2 to 20.

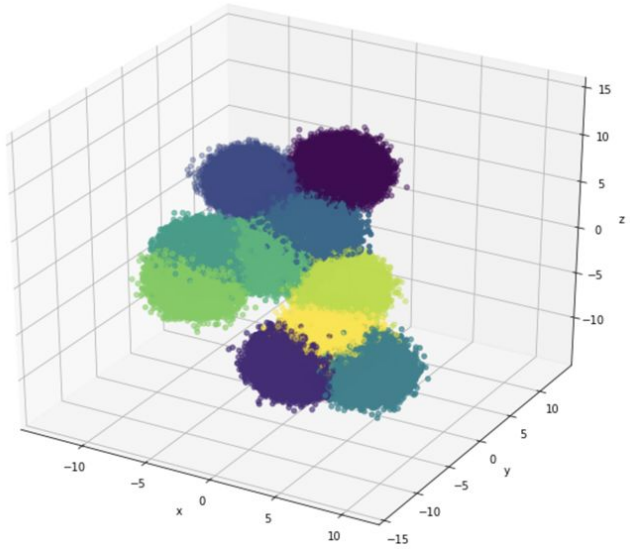
It’s clear that k=5 has the maximum silhouette value.



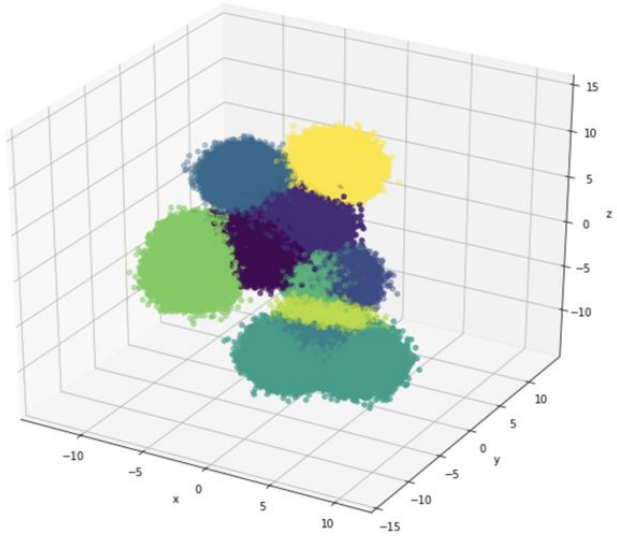
K=5 Clustering Result



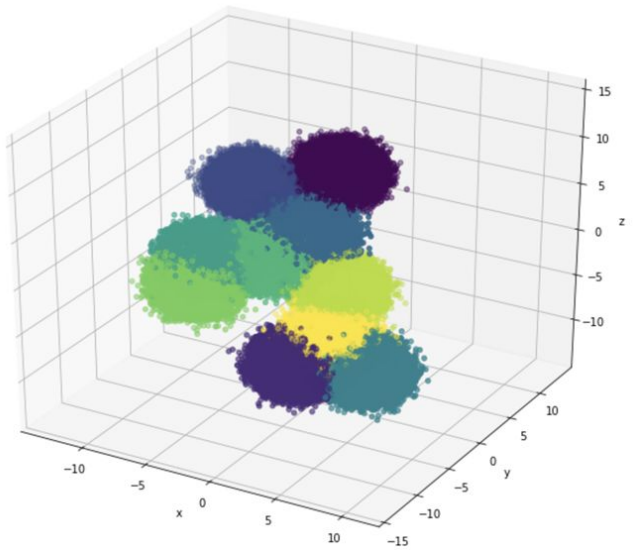
Ground truth



What about k=10?



Ground truth



Why?

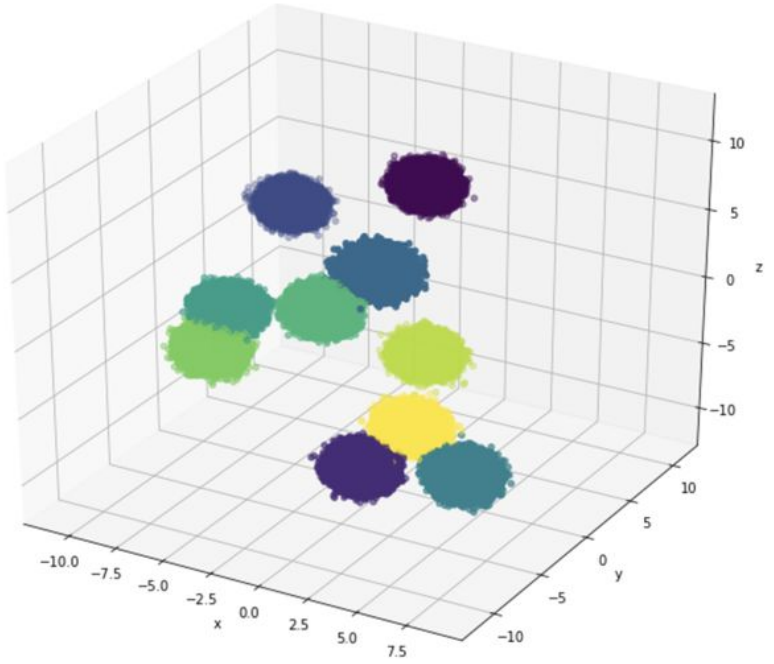
- 1 million data points were generated in our dataset and each blob has 100k points. Too many data points may lead to blurry boundary for the clusters.
- In the “make_blobs” function, the standard deviation was set to be 1. It’s relatively large, so some clusters are mixed together and some of the neighboring clusters may be similar.

```
n_samples=1000000
n_features=3
X, y = make_blobs(n_samples=n_samples,cluster_std=1, centers=10, n_features=n_features, random_state=42)
```

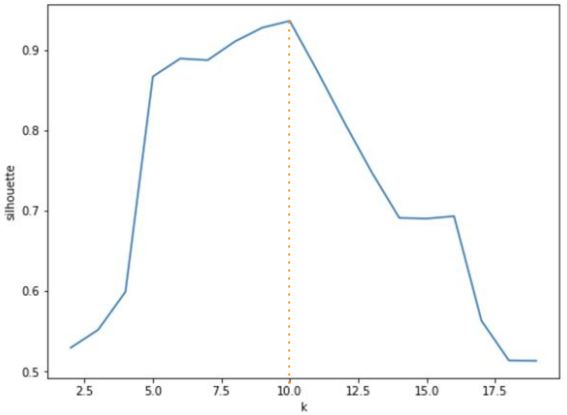
Another Dataset

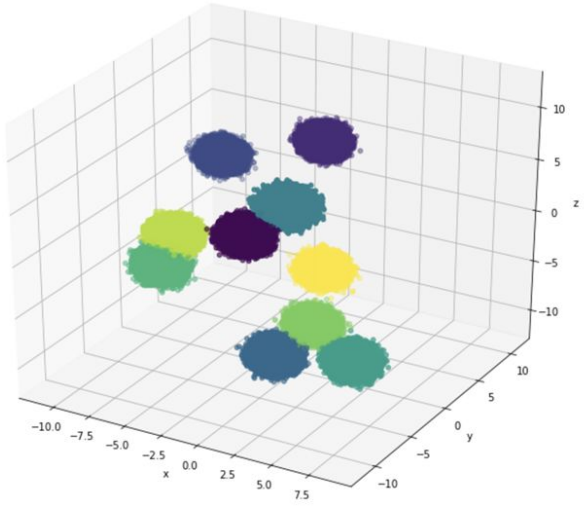
Set the standard deviation to be 0.5 to generate another dataset.

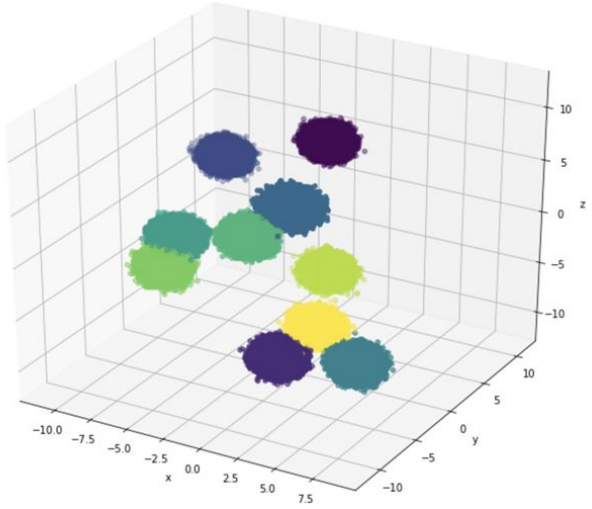
Now the 10 clusters are more separable.



Silhouette and Result







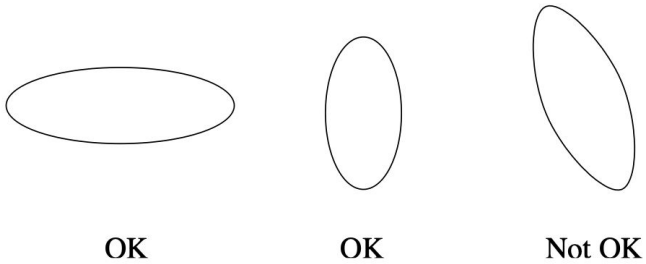
Ground truth

BFR Algorithm & CURE Algorithm

BFR Algorithm (Bradley, Fayyad and Reina)

A variant of k-means that is designed to cluster big dataset in a **high-dimensional** Euclidean space.

Strong assumption: the clusters must normally distributed about a centroid, and each dimension should be independent with each other.



Reference: Mining of Massive Data, Jure Leskovec, Anand Rajaraman, Jeff Ullman

BFR Initialization

To begin, from the initial load we select the initial k centroids by some sensible approach.

Probably include:

1. Take a small random sample and cluster optimally.

2. Take a sample; pick a random point, and then k – 1 more points, each as far from the previously selected points as possible.

Reference:
http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf

Three sets of Points

Each time, we read one part of dataset into main-memory and classify the points into one of following datasets:

1. The discard set : points close enough to a centroid to be summarized.

2. The compression set : groups of points that are close together but not close to any centroid. They are summarized, but not assigned to a cluster.

3. The retained set : isolated points.

Reference:
http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf

Summaries of clusters

For each cluster, the discard set can be summarized by some statistics:

1. The number of points, **N**
2. The vector **SUM**, the ith component of the SUM is the summation of coordinates of points in ith dimension
3. The vector **SUMSQ**, the ith component of SUMSQ is the summation of square of coordinates of points in ith dimension

2d+1 values to represent any number of points in one cluster (d is dimension number)

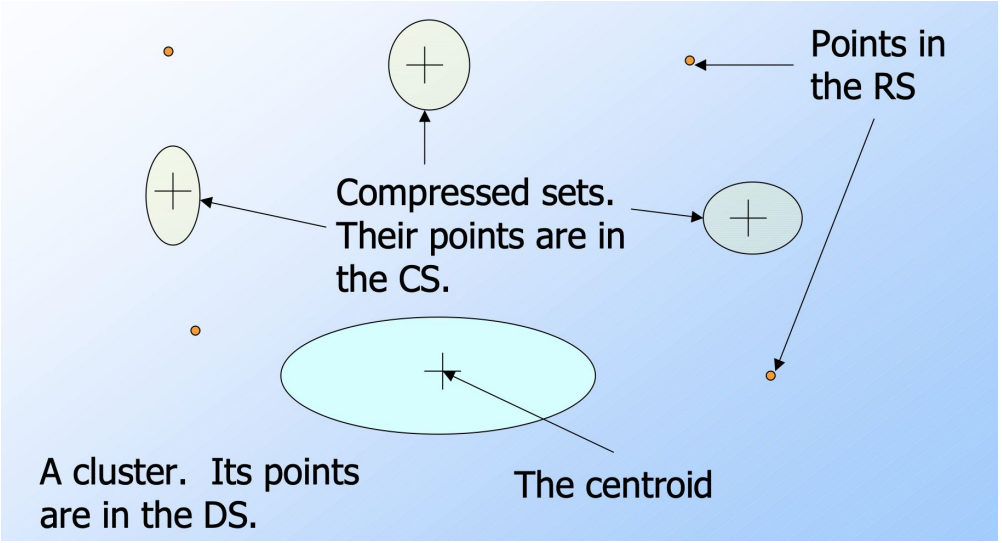
Average in each dimension: SUM/N

Variance in each dimension: SUMSQ/N-(SUM/N)^2

The same statistics can represent any compression set

Reference:
http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf

Image of three classes



Reference:
http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf

BFR Algorithm

0. Initialization (find k centroids)

1. Find those points that are “sufficiently close” to a cluster centroid; add those points to that cluster and the DS.

2. Use any main-memory clustering algorithm to cluster the remaining points and the old RS. (Clusters go to the CS; outlying points to the RS.)

Reference:
http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf

BFR Algorithm steps

3. Adjust statistics of the clusters to account for the new points. (Add N's, SUM's, SUMSQ's.)

4. Consider merging compressed sets in the CS

5. If this is the last round, merge all compressed sets in the CS and all RS points into their nearest cluster.

Reference:
http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf

BFR Details

1. What is close enough? (decide if put a new point into a cluster (discard set)):

a. The Mahalanobis distance is less than a threshold, e.g 2*sqrt(d).

Mahalanobis distance: $$d(x,c) = \sqrt{\sum_{i=1}^{d} \left(\frac{x_i - c_i}{\sigma_i}\right)^2}$$

b. High Likelihood of the point belonging to currently nearest centroid

Reference:
http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf

BFR Details

2. Should two compressed sets should be combined?

Compute the variance of the combined subcluster. (N, SUM, and SUMSQ allow us to make that calculation quickly)

Combine if the variance is below some threshold. (Many alternatives: treat dimensions differently, consider density.)

Reference:
http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf

BFR Example

To compare with original K-means algorithm, we used the same datasets with K-means algorithm. Generated 1000000 samples with “make_blob” function with 0.5 as the standard deviation.

And the BFR code, we used the code from Github https://github.com/Yuhan-Wg/massive-data-mining/blob/master/python/datming/clustering/bfr.py as reference.

We splitted 1M data into 10 pieces, and 100000 samples in each iteration, and choose 4*sqrt(d) as the threshold, merge Compression sets if their distance is less than 4*sqrt(d).

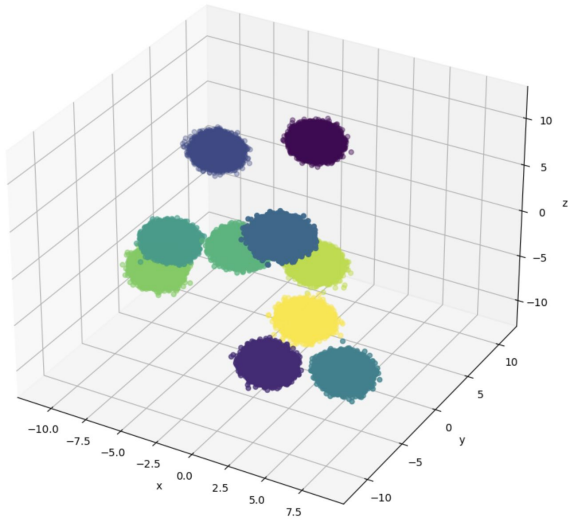
BFR Example

```
Tasks start. Start to print intermediate results ...
Round 1:
     Number of points in DS = 100000
     Number of clusters in CS = 0
     Number of points in CS = 0,
     Number of points in RS = 0
Round 2:
     Number of points in DS = 199289
     Number of clusters in CS = 85
     Number of points in CS = 711,
     Number of points in RS = 0
Round 3:
     Number of points in DS = 298538
     Number of clusters in CS = 43
     Number of points in CS = 1412,
     Number of points in RS = 50
Round 4:
     Number of points in DS = 397734
     Number of clusters in CS = 30
     Number of points in CS = 2213,
     Number of points in RS = 53
Round 5:
     Number of points in DS = 496899
     Number of clusters in CS = 29
     Number of points in CS = 3048,
     Number of points in RS = 53
```

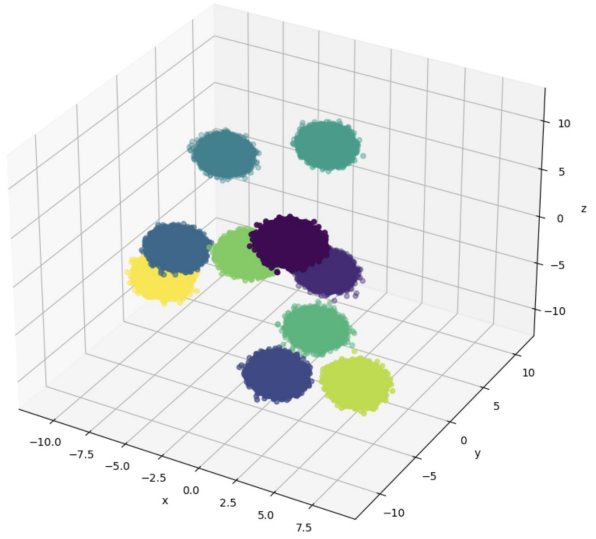
```
Round 6:
     Number of points in DS = 596092
     Number of clusters in CS = 28
     Number of points in CS = 3855,
     Number of points in RS = 53
Round 7:
     Number of points in DS = 695279
     Number of clusters in CS = 27
     Number of points in CS = 4668,
     Number of points in RS = 53
Round 8:
     Number of points in DS = 794485
     Number of clusters in CS = 27
     Number of points in CS = 5462,
     Number of points in RS = 53
Round 9:
     Number of points in DS = 893628
     Number of clusters in CS = 27
     Number of points in CS = 6318,
     Number of points in RS = 54
Round 10:
     Number of points in DS = 992742
     Number of clusters in CS = 27
     Number of points in CS = 7204,
     Number of points in RS = 54
Round final:
     Number of points in DS = 999946
     Number of clusters in CS = 1
     Number of points in CS = 0,
     Number of points in RS = 54
```

BFR Example

Time compared with K-means:
BFR : about 20 seconds
K-means: about 3 minutes



Ground Truth (How did we generate data)



BFR clustering result

CURE (Clustering Using REpresentatives)

1. Assume in Euclidean Space
2. Not assume the shape of cluster, can be not normally distributed, even strange bend or S-shape
3. Use a collection of representative points to represent the clusters but not their centroids

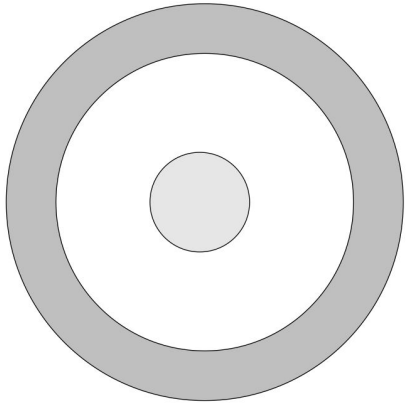


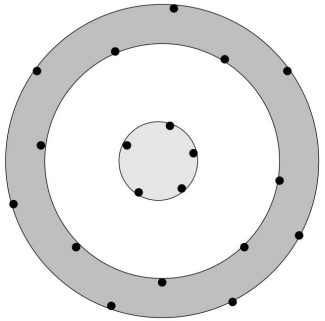
Figure 7.12: Two clusters, one surrounding the other

Reference: Mining of Massive Data, Jure Leskovec, Anand Rajaraman, Jeff Ullman

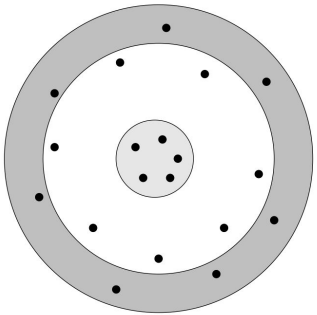
Initialization of CURE

We begin the CURE algorithm by:

1. **Take a small sample of the data and cluster it in main memory.** (Any clustering methods can use, but to solve oddly shaped clusters, usually use hierarchical clustering).
2. **Select a small set of points from each cluster to be representative points.** (These points should be chosen to be as far from one another as possible).
3. **Move each of the representative points a fixed fraction of the distance between its location and the centroid of its cluster.** (Perhaps 20% is a good fraction to choose, and this step needs an Euclidean space).



Select representative points from each cluster, as far from one another as possible



Moving the representative points 20% of the distance to the cluster's centroid

Reference: Mining of Massive Data, Jure Leskovec, Anand Rajaraman, Jeff Ullman

Next Steps

Merge two clusters if they have a pair of representative points, one from each cluster, that are sufficiently close. This merging step can repeat, until there are no more sufficiently close clusters.

Two user choices:

1.The fraction of the distance to the centroid that we move the representative points

2. The choice of how far apart representative points of two clusters need to be to avoid merge

Reference: Mining of Massive Data, Jure Leskovec, Anand Rajaraman, Jeff Ullman

Last Step

Point assignment: Each point p is brought from secondary storage and compared with the representative points. We assign p to the cluster of the representative point that is closest to p.

Reference: Mining of Massive Data, Jure Leskovec, Anand Rajaraman, Jeff Ullman

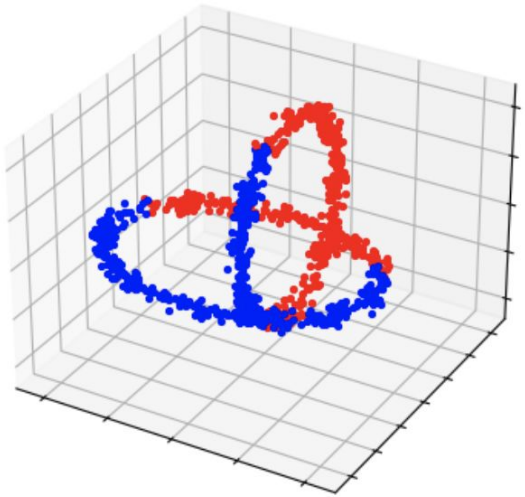
CURE Example

To show the CURE algorithm properties, we used some special shape cluster to test CURE algorithm. We used the data in ‘pyclustering’ module and the ‘cure’ function in ‘pyclustering’ module to conduct this test.

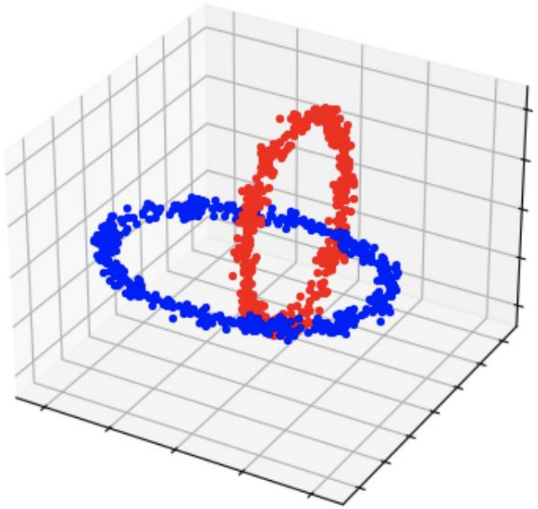
The reference code and data can be found in following website:

https://github.com/annoviko/pyclustering

CURE Example

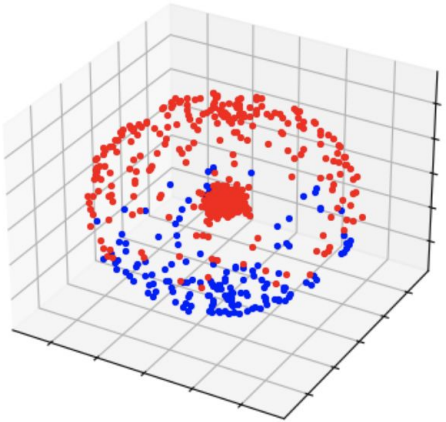


K-means result

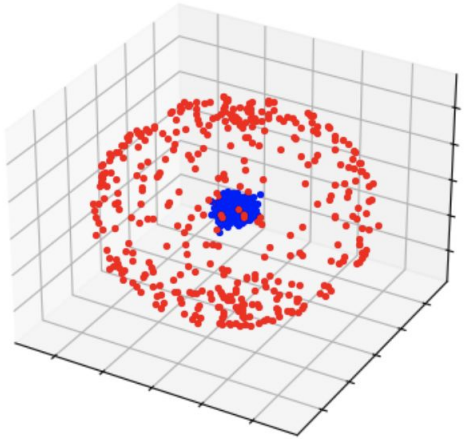


CURE result

CURE Example



K-means result



CURE result

Reference

- https://rsandstroem.github.io/sparkkmeans.html
- https://www.geeksforgeeks.org/clustering-in-machine-learning/
- https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages
- https://en.wikipedia.org/wiki/Silhouette_(clustering)
- https://sites.cs.ucsb.edu/~veronika/MAE_LargeScaleClustering_strnadova.pdf
- https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html
- http://infolab.stanford.edu/~ullman/mmds/ch7.pdf
- http://infolab.stanford.edu/~ullman/mining/2009/clustering.pdf
- https://www.youtube.com/watch?v=RdT7bhm1M3E&t=675s

Thanks for listening!

Questions? Comments?