Cluster Analysis

YING SHEN

SSE, TONGJI UNIVERSITY

DEC. 2016

Cluster analysis

Cluster analysis groups data objects based only on the attributes in the data.

The main objective is that

- The objects within a group be similar to one another and
- They are different from the objects in the other groups.

Cluster analysis

Cluster analysis is important in the following areas:

- Biology
- Information retrieval
- Medicine
- Business

Cluster analysis

Cluster analysis provides an abstraction from individual data objects to the clusters in which those data objects reside.

Some clustering techniques characterize each cluster in terms of a cluster prototype.

The prototype is a data object that is representative of the other objects in the cluster.

Different types of clusterings

We consider the following types of clusterings

- Partitional versus hierarchical
- Exclusive versus fuzzy
- Complete versus partial

Partitional versus hierarchical

A partitional clustering is a division of the set of data objects into subsets (clusters).

A hierarchical clustering is a set of nested clusters that are organized as a tree.

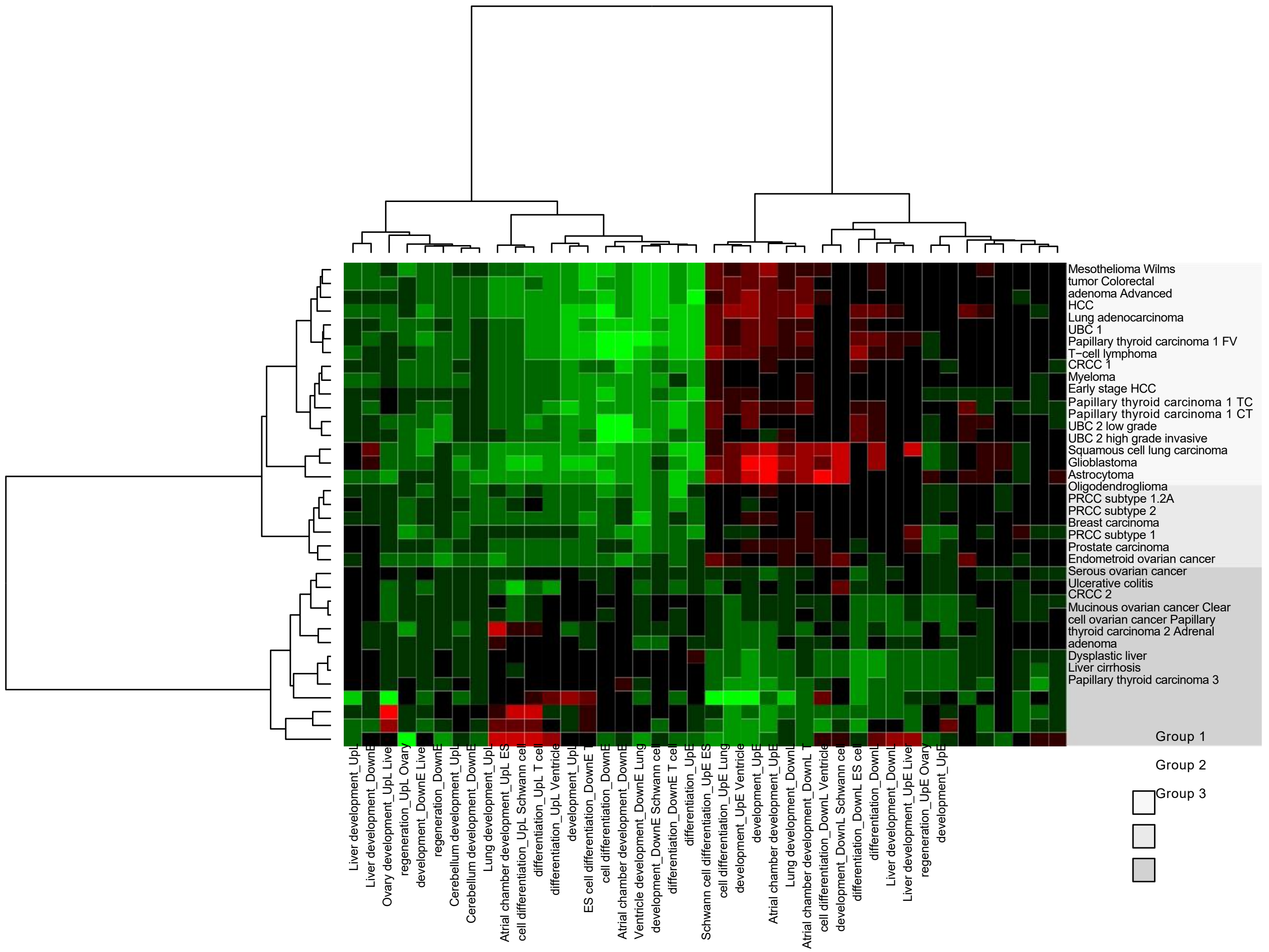
Each node (cluster) in the tree (except for the leaf nodes) is the union of its children (sub-clusters).

The root of the tree is the cluster containing all the objects.

Often, but not always, the leaves of the tree are singleton clusters of individual data objects.

Partitional versus hierarchical

An example of hierarchical clustering

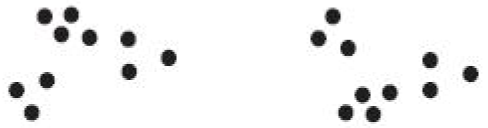


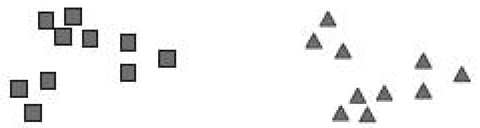
Partitional versus hierarchical

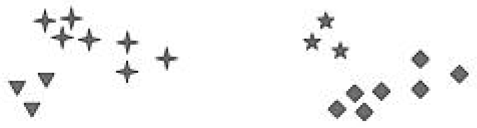
The following figures form a hierarchical (nested) clustering with 1, 2, 4 and 6 clusters on each level.

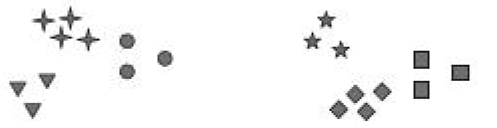
A hierarchical clustering can be viewed as a sequence of partitional clusterings.

A partitional clustering can be obtained by taking any member of that sequence, i.e. by cutting the hierarchical tree at a certain level.


(a) Original points.


(b) Two clusters.

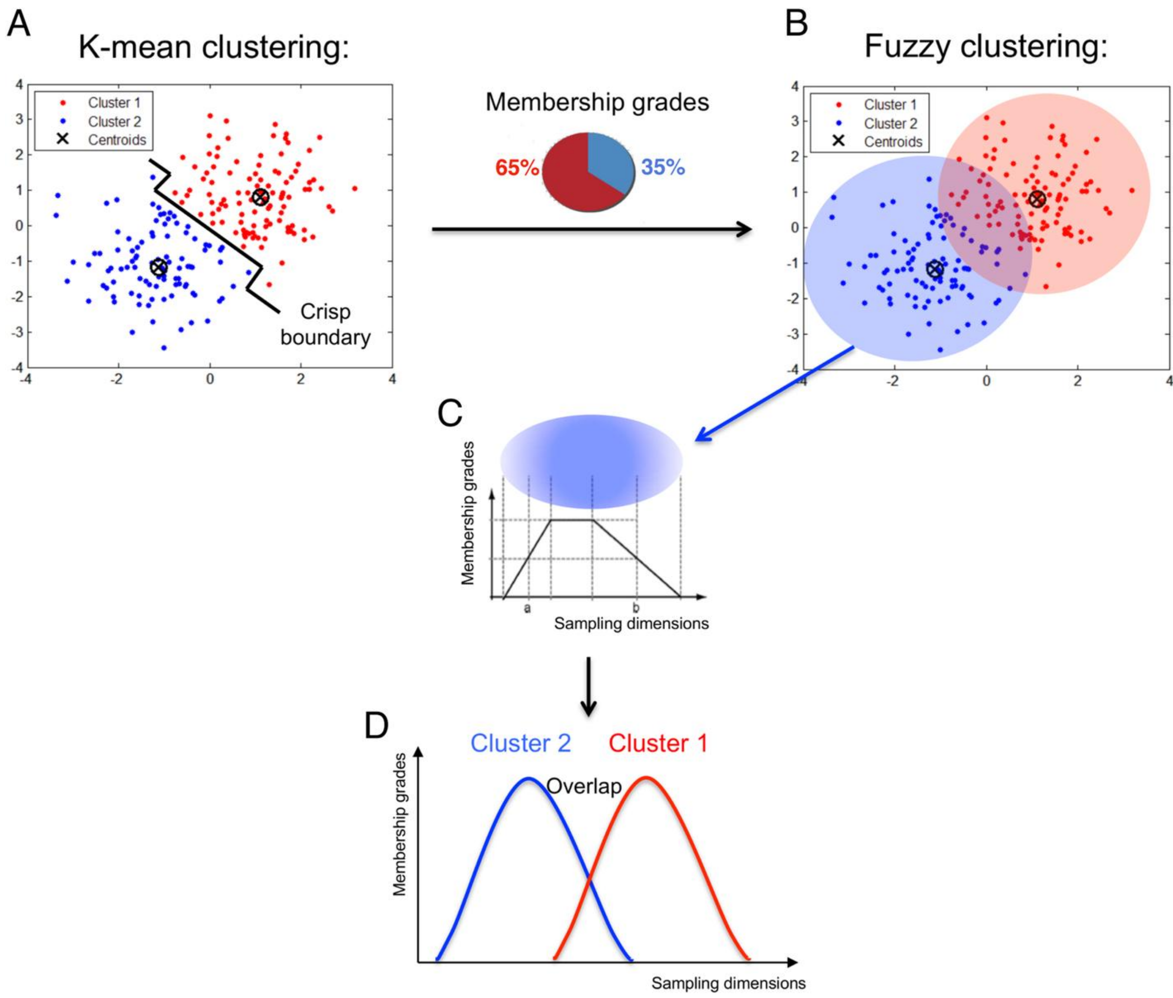

(c) Four clusters.


(d) Six clusters.

Exclusive versus fuzzy

In an exclusive clustering, each object is assigned to a single cluster.

However, there are many situations in which a point could reasonably be placed in more than one cluster.



Exclusive versus fuzzy

In a fuzzy clustering, every object belongs to every cluster with a membership weight that is between

- 0 (absolutely does not belong) and
- 1 (absolutely belongs).

This approach is useful for avoiding the arbitrariness of assigning an object to only one cluster when it is close to several.

A fuzzy clustering can be converted to an exclusive clustering by assigning each object to the cluster in which its membership value is the highest.

Complete versus partial

A complete clustering assigns every object to a cluster.

A partial clustering does not assign every object to a cluster.

The motivation of partial clustering is that some objects in a data set may not belong to well-defined groups.

Instead, they may represent noise or outliers.

K-means

K-means is a prototype-based clustering technique which creates a one-level partitioning of the data objects.

Specifically, K-means defines a prototype in terms of the centroid of a group of points.

K-means is typically applied to objects in a continuous n-dimensional space.

K-means

The basic K-means algorithm is summarized below

1. Select K points as initial centroids
2. Repeat
    a. Form K clusters by assigning each point to its closest centroid.
    b. Recompute the centroid of each cluster.
3. Until centroids do not change.

K-means

We first choose K initial centroids, where K is a user-defined parameter, namely, the number of clusters desired.

Each point is then assigned to the closest centroid.

Each collection of points assigned to the same centroid is a cluster.

The centroid of each cluster is then updated based on the points assigned to the cluster.

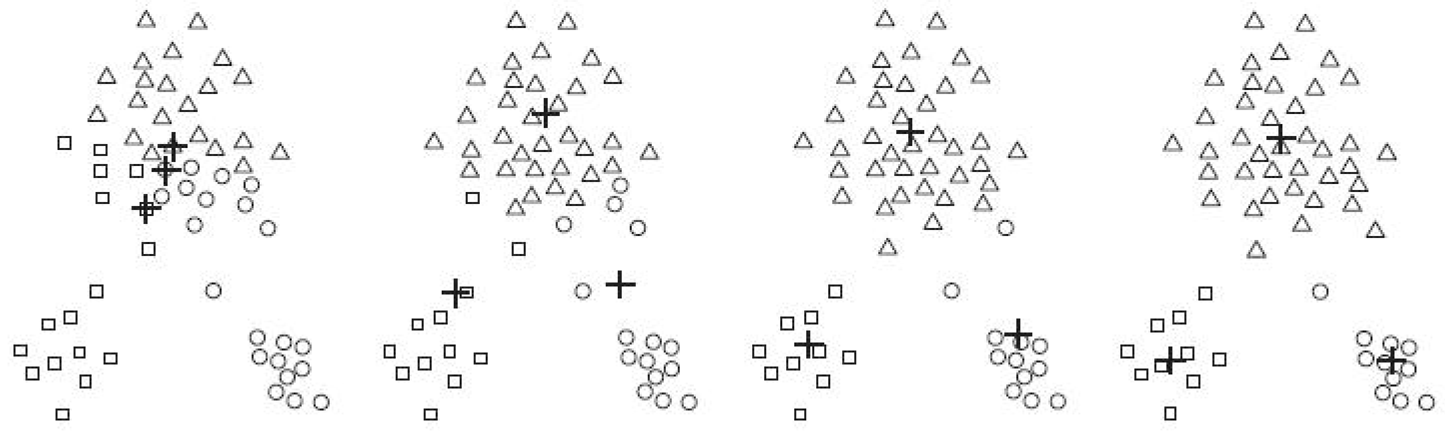
We repeat the assignment and update steps until the centroids remain the same.

K-means

These steps are illustrated in the following figures.

Starting from three centroids, the final clusters are found in four assignment-update steps.

K-means



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Each sub-figure shows

- The centroids at the start of the iteration and
- The assignment of the points to those centroids.

The centroids are indicated by the “+” symbol.

All points belonging to the same cluster have the same marker shape.

K-means

In the first step, points are assigned to the initial centroids, which are all in the largest group of points.

After points are assigned to a centroid, the centroid is then updated.

In the second step

- Points are assigned to the updated centroids and
- The centroids are updated again.

We can observe that two of the centroids move to the two small groups of points at the bottom of the figures.

When the K-means algorithm terminates, the centroids have identified the natural groupings of points.

Proximity measure

To assign a point to the closest centroid, we need a proximity measure that quantifies the notion of "closest".

Euclidean (L_2) distance is often used for data point in Euclidean space.

Proximity measure

The goal of the clustering is typically expressed by an objective function.

Consider data whose proximity measure is Euclidean distance.

For our objective function, which measures the quality of a clustering, we can use the sum of the squared error (SSE).

Proximity measure

We calculate the Euclidean distance of each data point to its closest centroid.

We then compute the total sum of the squared distances, which is also known as the sum of the squared error (SSE).

A small value of SSE means that the prototypes (centroids) of this clustering are a better representation of the points in their cluster.

Proximity measure

The SSE is defined as follows:

$$SSE = \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in C_i} d(\boldsymbol{x}, \boldsymbol{c}_i)^2$$

In this equation

- $\boldsymbol{x}$ is a data object.
- C_i is the i-th cluster.
- $\boldsymbol{c}_i$ is the centroid of cluster C_i.
- d is the Euclidean (L_2) distance between two objects in Euclidean space.

Proximity measure

It can be shown that the mean of the data points in the cluster minimizes the SSE of the cluster.

The centroid (mean) of the i-th cluster is defined as

$$\boldsymbol{c}_i = \frac{1}{m_i} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$$

In this equation, m_i is the number of objects in the i-th cluster

Proximity measure

Steps 2a and 2b of the K-means algorithm attempt to minimize the SSE.

Step 2a forms clusters by assigning points to their nearest centroid, which minimizes the SSE for the given set of centroids.

Step 2b recomputes the centroids so as to further minimize the SSE.

Choosing initial centroids

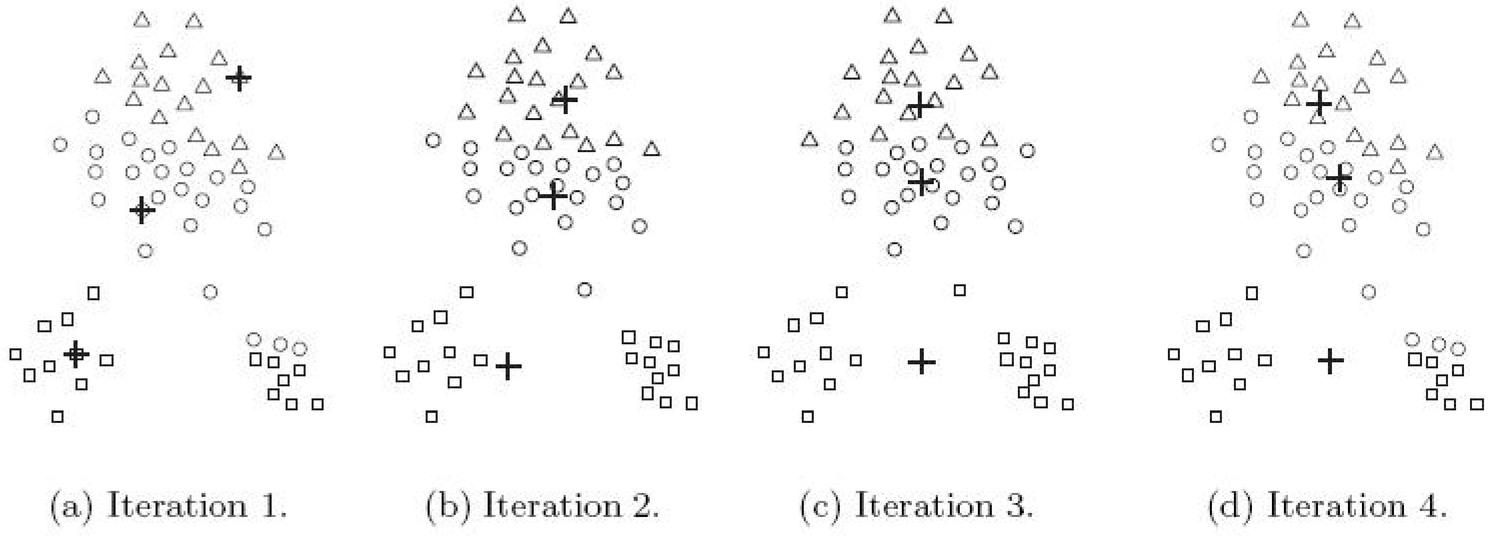
Choosing the proper initial centroids is the key step of the basic K-means procedure.

A common approach is to choose the initial centroids randomly.

Randomly selected initial centroids may be poor choices.

This is illustrated in the following figures.

Choosing initial centroids



Choosing initial centroids

One technique that is commonly used to address the problem of choosing initial centroids is to perform multiple runs.

Each run uses a different set of randomly chosen initial centroids.

We then choose the set of clusters with the minimum SSE.

Outliers

When the Euclidean distance is used, outliers can influence the clusters that are found.

When outliers are present, the resulting cluster centroids may not be as representative as they otherwise would be.

The SSE will be higher as well.

Because of this, it is often useful to discover outliers and eliminate them beforehand.

Outliers

To identify the outliers, we can keep track of the contribution of each point to the SSE.

We then eliminate those points with unusually high contributions to the SSE.

We may also want to eliminate small clusters, since they frequently represent groups of outliers.

Post-processing

Two post-processing strategies that decrease the SSE by increasing the number of clusters are

- Split a cluster
  - The cluster with the largest SSE is usually chosen.
- Introduce a new cluster centroid
  - Often the point that is farthest from its associated cluster center is chosen.
- We can determine this if we keep track of the contribution of each point to the SSE.

Post-processing

Two post-processing strategies that decrease the number of clusters, while trying to minimize the increase in total SSE, are

- Disperse a cluster
  - This is accomplished by removing the centroid that corresponds to the cluster.
  - The points in that cluster are then re-assigned to other clusters.
  - The cluster that is dispersed should be the one that increases the total SSE the least.
- Merge two clusters
  - We can merge the two clusters that result in the smallest increase in total SSE.

Bisecting K-means

Bisecting K-means algorithm is an extension of the basic K-means algorithm.

The main steps of the algorithm are described as follows

- To obtain K clusters, split the set of all points into two clusters.
- Select one of these clusters to split.
- Continue the process until K clusters have been produced.

Bisecting K-means

There are a number of different ways to choose which cluster to split.

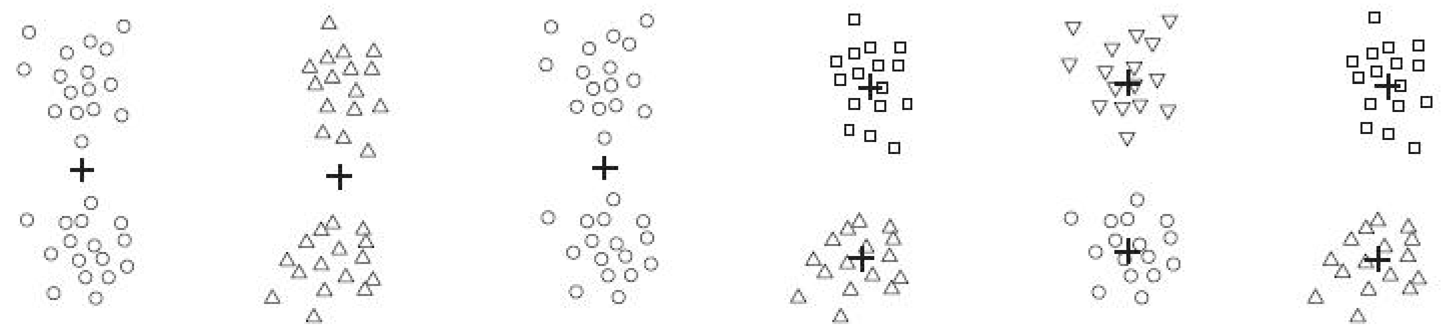
- We can choose the largest cluster at each step.
- We can also choose the one with the largest SSE.
- We can also use a criterion based on both size and SSE.

Different choices result in different clusters.

We often refine the resulting clusters by using their centroids as the initial centroids for the basic K-means algorithm.

The bisecting K-means algorithm is illustrated in the following figure.

Bisecting K-means


(a) Iteration 1.

(b) Iteration 2.

(c) Iteration 3.

Limitations of K-means

K-means and its variations have a number of limitations with respect to finding different types of clusters.

In particular, K-means has difficulty detecting clusters with non-spherical shapes or widely different sizes or densities.

This is because K-means is designed to look for globular clusters of similar sizes and densities, or clusters that are well separated.

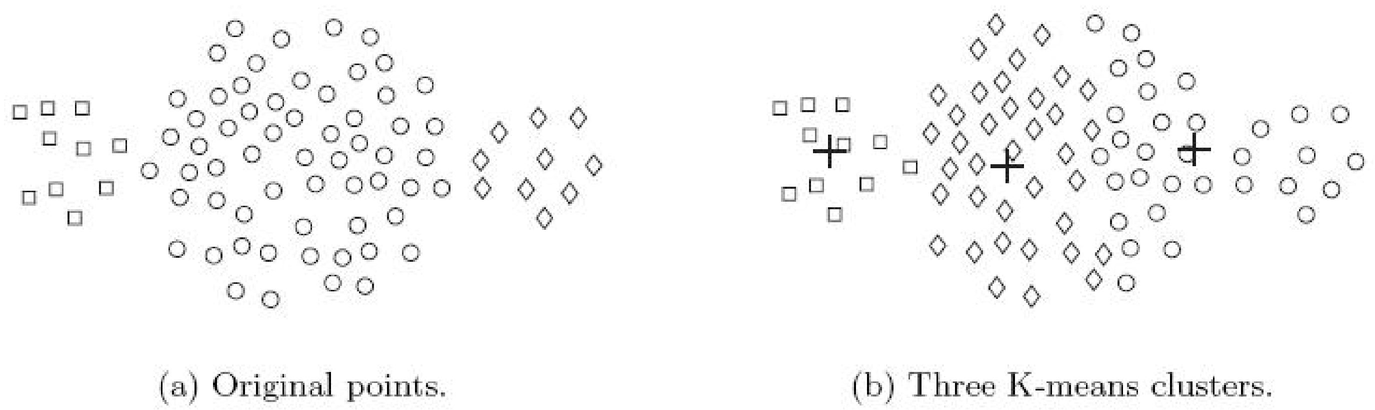
This is illustrated in the following examples.

Limitations of K-means

In this example, K-means cannot find the three natural clusters because one of the clusters is much larger than the other two.

As a result, the largest cluster is divided into sub-clusters.

At the same time, one of the smaller clusters is combined with a portion of the largest cluster.

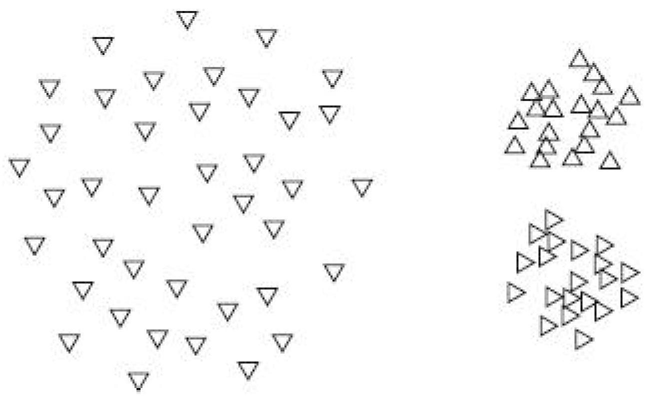


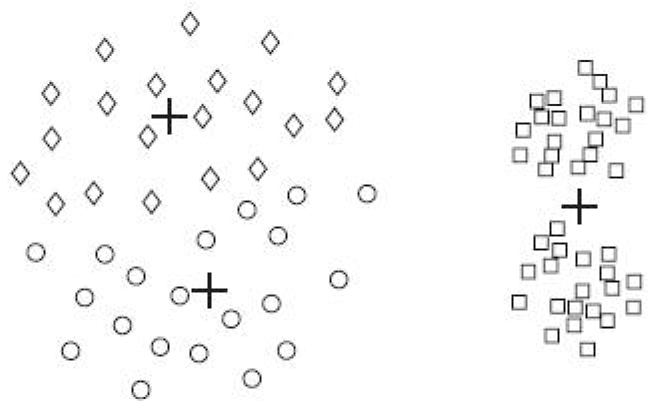
(a) Original points. (b) Three K-means clusters.

Limitations of K-means

In this example, K-means fails to find the three natural clusters.

This is because the two smaller clusters are much denser than the largest cluster.

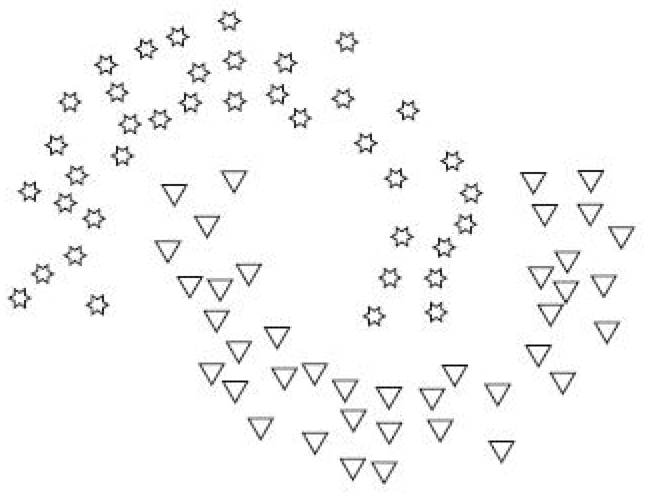

(a) Original points.


(b) Three K-means clusters.

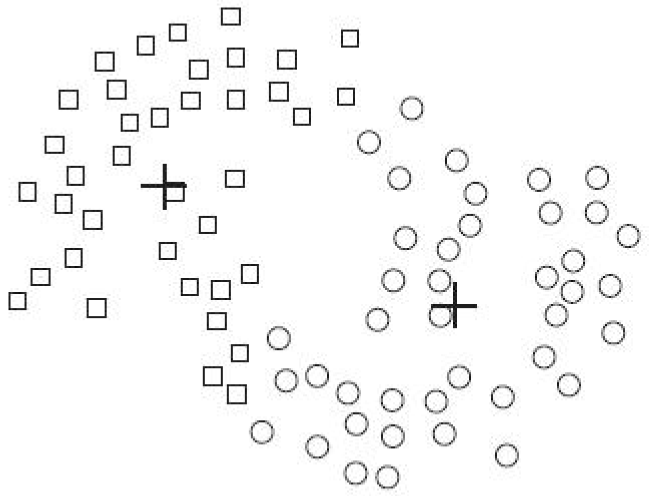
Limitations of K-means

In this example, K-means finds two clusters that mix portions of the two natural clusters.

This is because the shape of the natural clusters is not globular.



(a) Original points.



(b) Two K-means clusters.

DBSCAN

Main idea – Locate regions of high density that are separated from one another by regions of low density

So, how do we measure density of a region ?

- Density at a point P: Number of points within a circle of Radius *Eps* (ϵ) from point P.
- Dense Region: For each point in the cluster, the circle with radius ϵ contains at least minimum number of points (*MinPts*).

DBSCAN

The Epsilon neighborhood of a point p in the database D is defined as

$$N\ (p)\ =\ \{q\ \in\ D\ |\ dist(p,\ q)\ \leq\ \epsilon\}$$

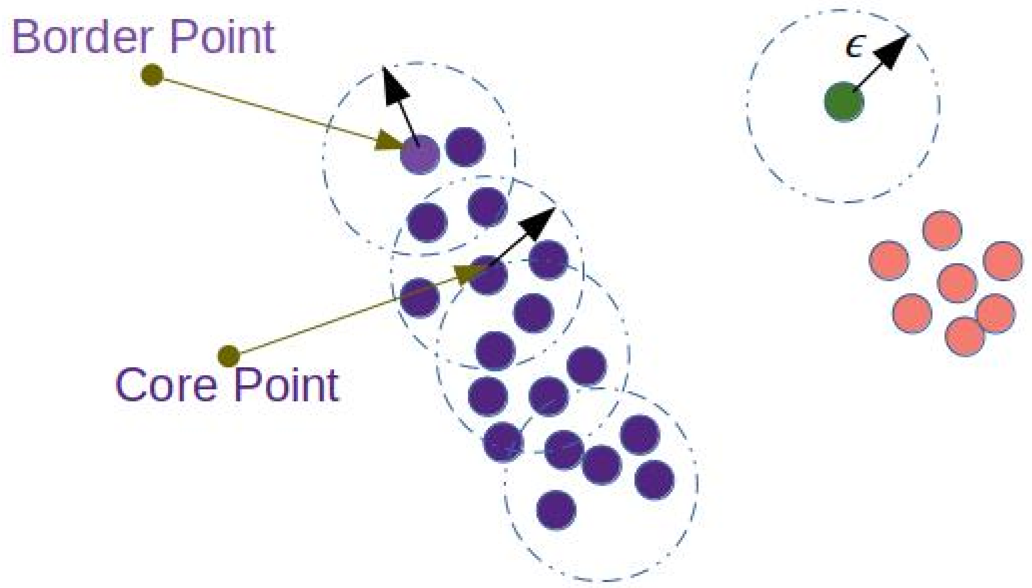
Following the definition of dense region , a point can be classified as a ***Core Point*** *if* $|N(p)| \geq MinPts.$

- Core point lie usually within the interior of a cluster.

A ***Border Point*** has fewer than *MinPts* within its *ϵ-neighborhood* (N) , but it lies in the neighborhood of another core point.

Noise is any data point that is neither core nor border point.

DBSCAN



$$N_{Eps}(p) = \{q \in D \text{ such that } dist(p,q) \leq \epsilon\}$$

ϵ = 1 unit, MinPts = 7

DBSCAN

Directly Density Reachable: Data-point a is directly density reachable from a point b if

- $|N(b)| \geq MinPts$; *i.e. b is a core point.*
- $a \in N(b)$ *i.e. a is in the epsilon neighborhood of b.*

Density Reachable: Point a is density reachable from a point b with respect to ϵ and *MinPts*, if

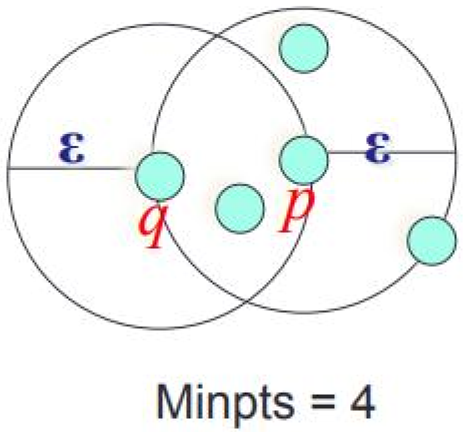
- For a chain of points $b_1, b_2, \ldots, b_n$, where $b_1 = b$, $b_n = a$, such that b_{i+1} is directly density reachable from b_i
- Density reachable is transitive in nature but, just like direct density reachable, it is not symmetric.

Density Connected: A point a is density connected to a point b with respect to ϵ and *MinPts*, if there is a point c such that, both a and b are density reachable from c w.r.t. to ϵ and *MinPts*.

DBSCAN

Directly Density Reachable: Data-point a is directly density reachable from a point b if

- $|N(b)| \geq MinPts;$ *i.e.* b *is a core point.*
- $a \in N(b)$ *i.e.* a *is in the epsilon neighborhood of* b.



Q. p is directly density-reachable from q?

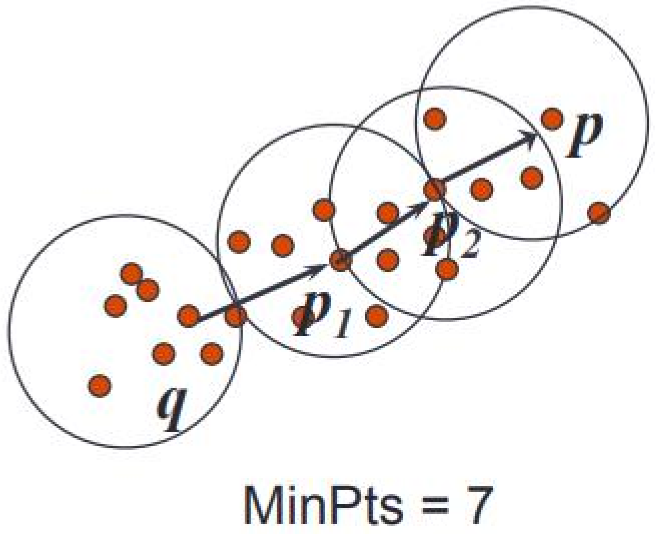
No, why?

Q. Density-reachability is asymmetric

DBSCAN

Density Reachable: Point a is density reachable from a point b with respect to ϵ and $MinPts$, if

- For a chain of points $b_1, b_2, \ldots, b_n$, where $b_1 = b$, $b_n = a$, such that b_{i+1} is directly density reachable from b_i
- Density reachable is transitive in nature but, just like direct density reachable, it is not symmetric.



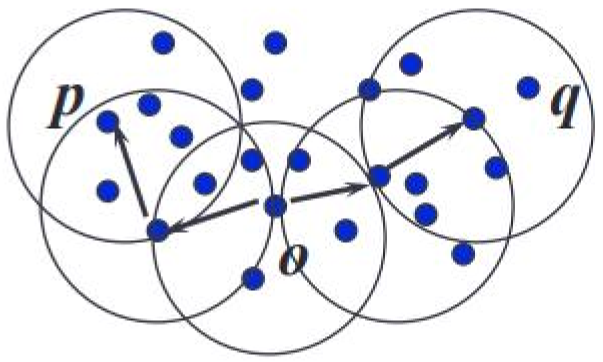
- p1 is directly density-reachable from q
- p2 is directly density-reachable from p1
- p is directly density-reachable from p2
- There is a chain from q to p (q→p1→p2→q)

Q. q is density-reachable from p?

No, why?

DBSCAN

Density Connected: A point a is density connected to a point b with respect to ϵ and $MinPts$, if there is a point c such that, both a and b are density reachable from c w.r.t. to ϵ and $MinPts$.



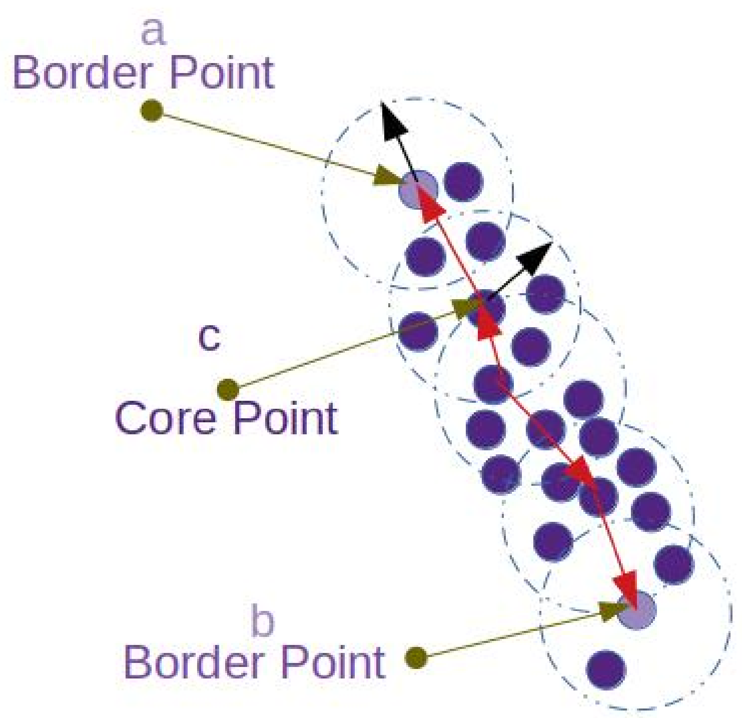
MinPts = 7

Q. o is density-reachable from p?

Yes, why?

Q. Density-connectivity is symmetric

DBSCAN

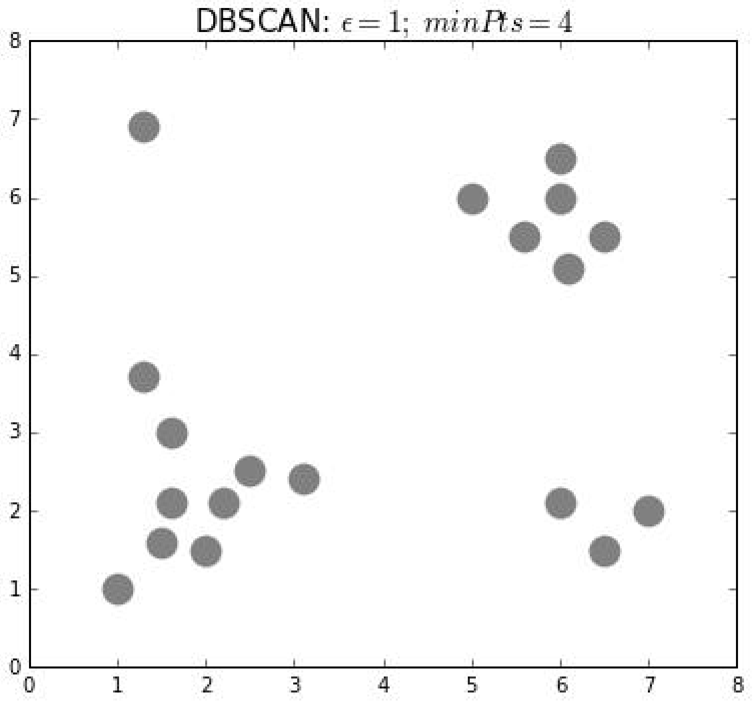


a, b are Density Reachable from a core point c.

a, b are called Density Connected points.

DBSCAN

DBSCAN Algorithm



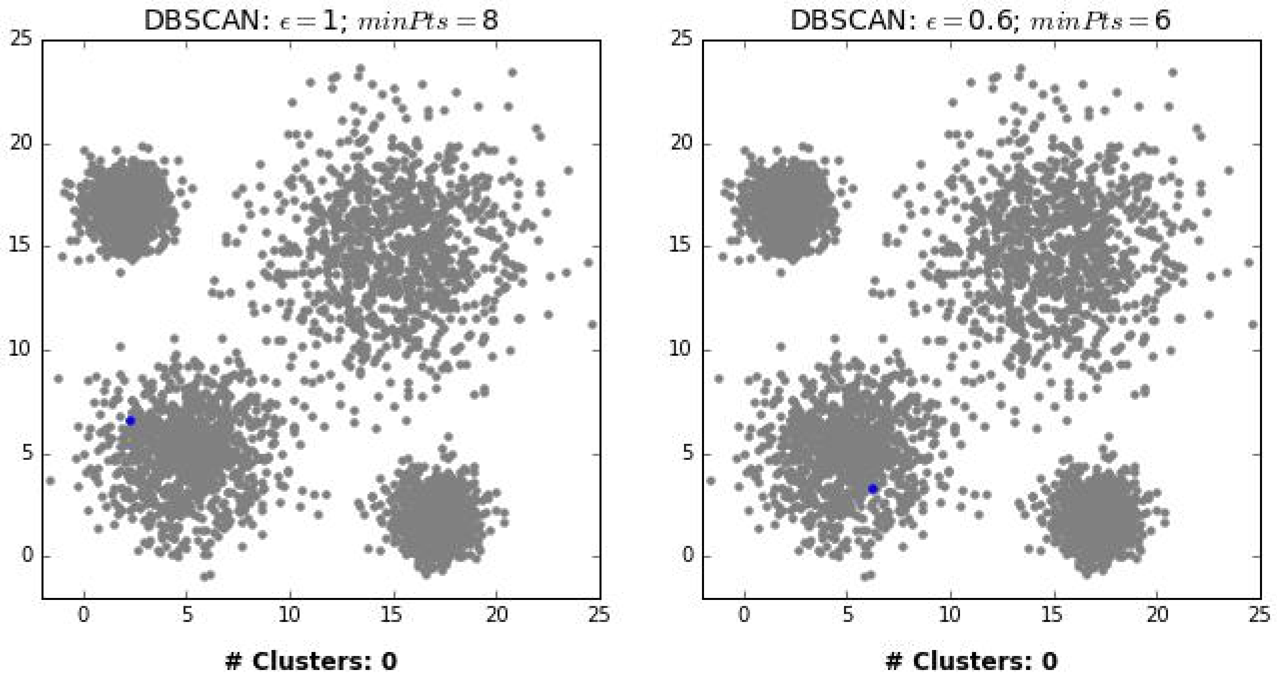
DBSCAN

DBSCAN Algorithm

1. Select an arbitrary point which has not been visited and its neighborhood information is retrieved from the ϵ parameter.
2. If this point contains *MinPts* within ϵ neighborhood, cluster formation starts. Otherwise the point is labeled as noise.
3. If a point is found to be a core point then the points within the ϵ neighborhood is also part of the cluster.
4. The above process continues until the density-connected cluster is completely found.
5. The process restarts with a new point which can be a part of a new cluster or labeled as noise.

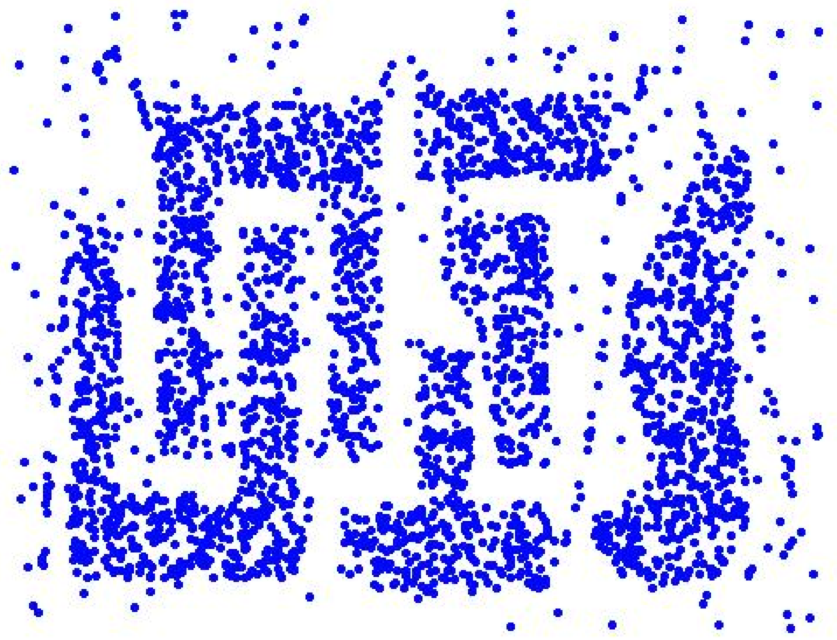
DBSCAN

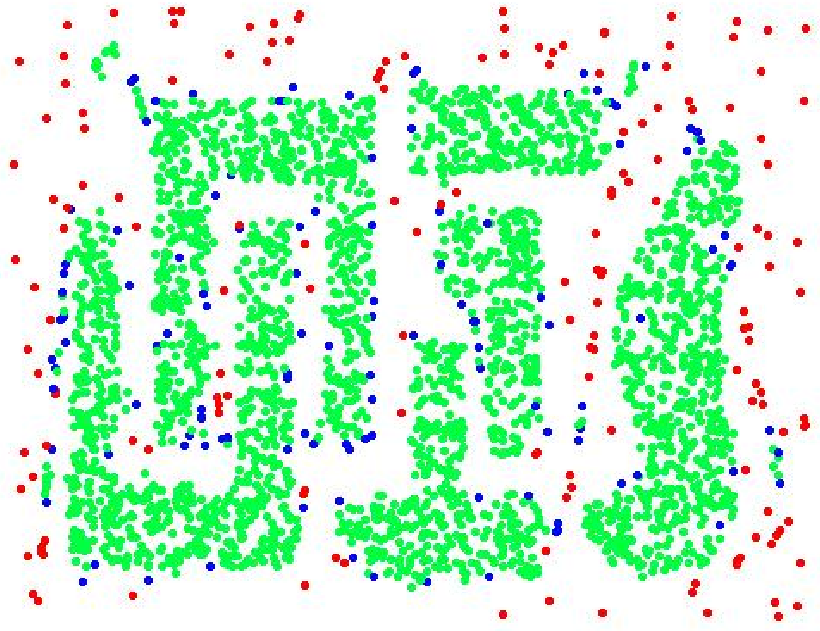
DBSCAN Algorithm



DBSCAN

Example

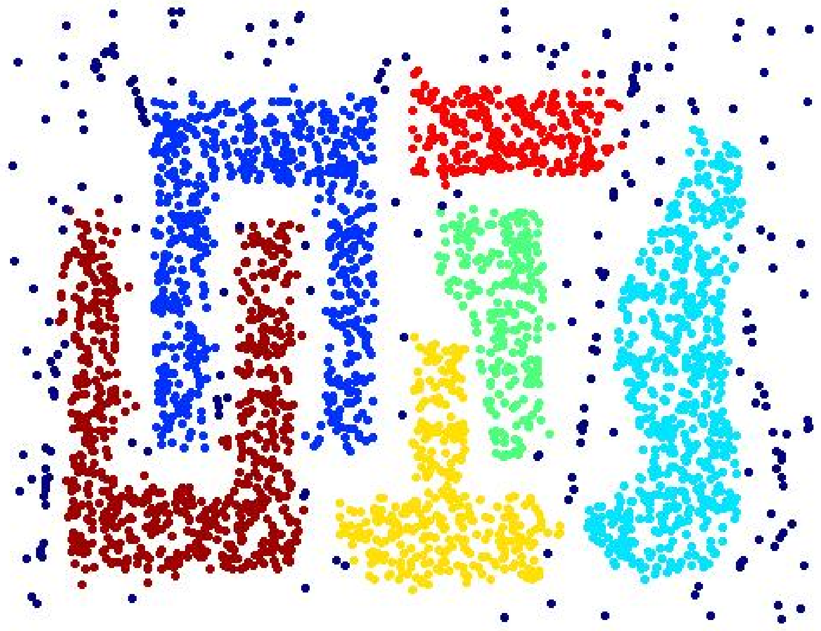

Original Points


Point types: core, border and outliers

DBSCAN

Resistant to Noise

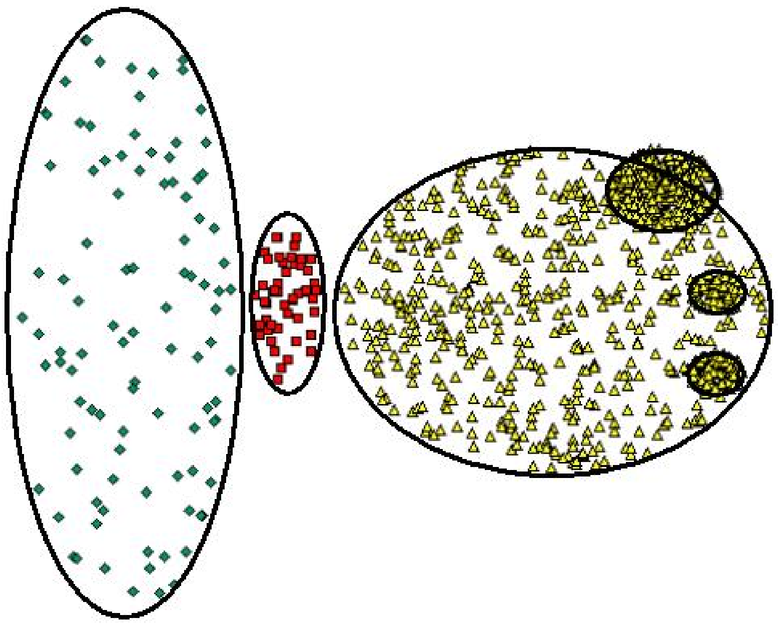
Can handle clusters of different shapes and sizes

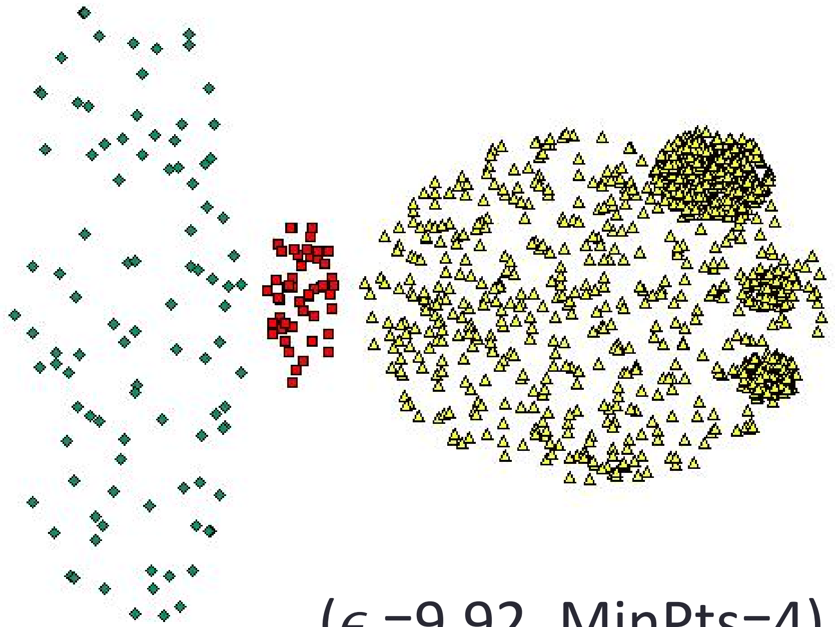


DBSCAN

Drawbacks

- If the database has data points that form clusters of varying density, then DBSCAN fails to cluster the data points well
- If the data and features are not so well understood by a domain expert then, setting up ϵ and *MinPts* could be tricky and, may need comparisons for several iterations with different values of ϵ and *MinPts*.





(ϵ =9.92, MinPts=4)

Hierarchical clustering

A hierarchical clustering is a set of nested clusters that are organized as a tree.

There are two basic approaches for generating a hierarchical clustering

- Agglomerative
- Divisive

Hierarchical clustering

In agglomerative hierarchical clustering, we start with the points as individual clusters.

At each step, we merge the closest pair of clusters.

This requires defining a notion of cluster proximity.

Hierarchical clustering

In divisive hierarchical clustering, we start with one, all-inclusive cluster.

At each step, we split a cluster.

This process continues until only singleton clusters of individual points remain.

In this case, we need to decide

- Which cluster to split at each step and
- How to do the splitting.

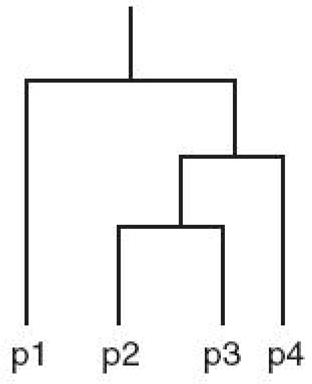
Hierarchical clustering

A hierarchical clustering is often displayed graphically using a tree-like diagram called the dendrogram.

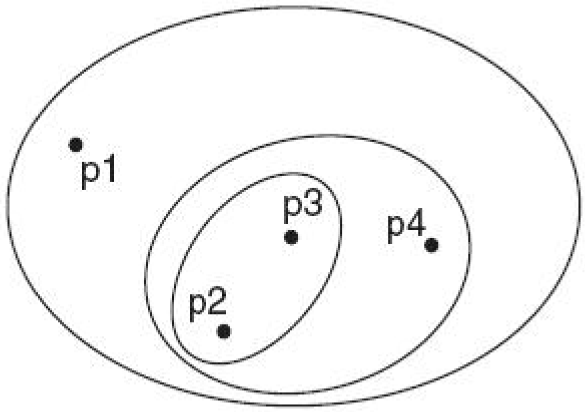
The dendrogram displays both

- the cluster-subcluster relationships and
- the order in which the clusters are merged (agglomerative) or split (divisive).

For sets of 2-D points, a hierarchical clustering can also be graphically represented using a nested cluster diagram.



(a) Dendrogram.

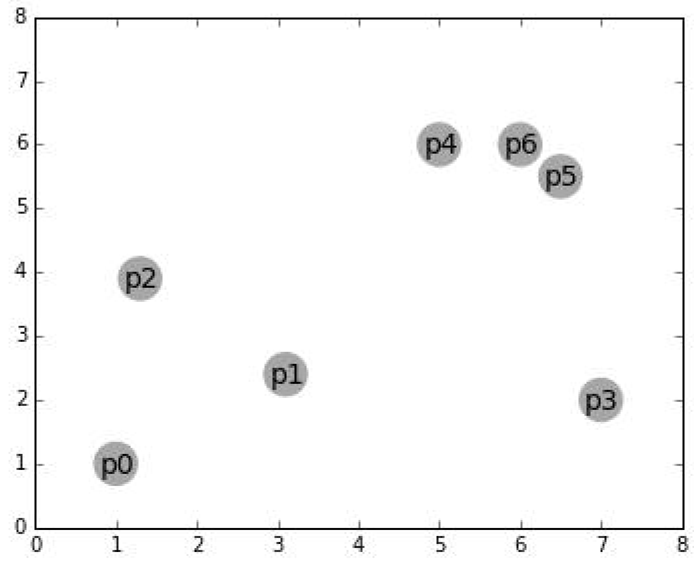


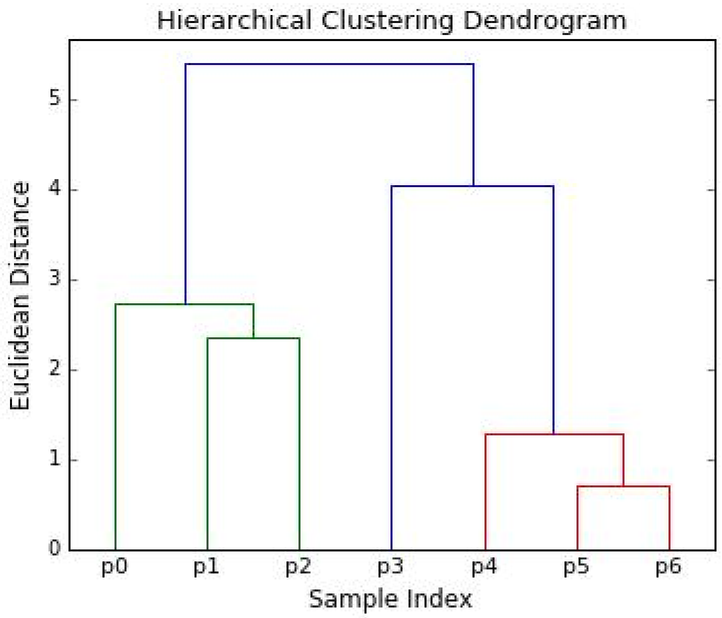
(b) Nested cluster diagram.

Hierarchical clustering

The basic agglomerative hierarchical clustering algorithm is summarized as follows

- Compute the proximity matrix.
- Repeat
  - Merge the closest two clusters
  - Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
- Until only one cluster remains





Hierarchical clustering

Different definitions of cluster proximity leads to different versions of hierarchical clustering.

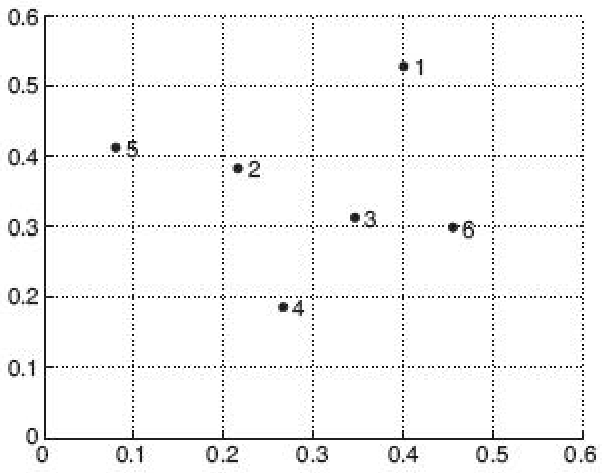
These versions include

- Single link or MIN
- Complete link or MAX
- Group average
- Ward’s method

Hierarchical clustering

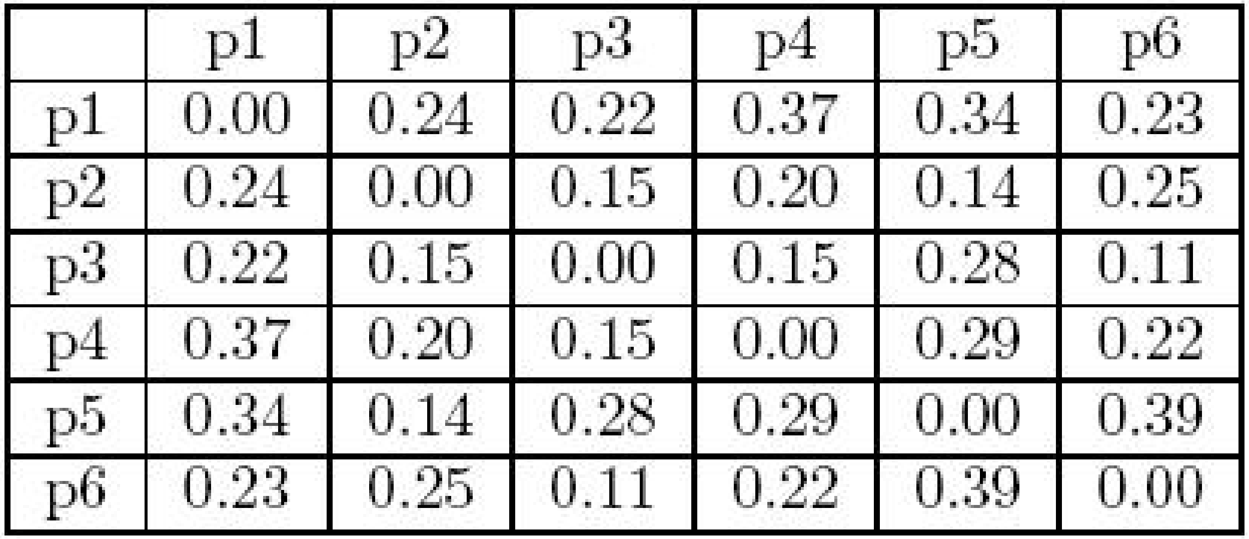
We consider the following set of data points.

The Euclidean distance matrix for these data points is shown in the following slide.



Point	x Coordinate	y Coordinate
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



	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Single link

We now consider the single link or MIN version of hierarchical clustering.

In this case, the proximity of two clusters is defined as the minimum of the distance between any two points in the two different clusters.

This technique is good at handling non-elliptical shapes.

However, it is sensitive to noise and outliers.

Single link

The following figure shows the result of applying single link technique to our example data.

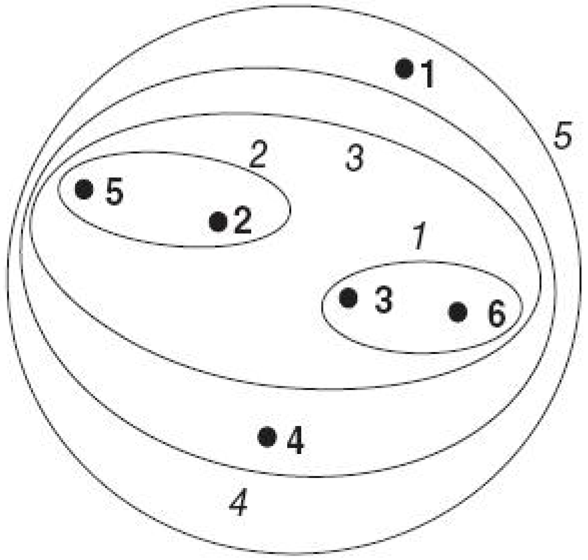
The left figure shows the nested clusters as a sequence of nested ellipses.

The numbers associated with the ellipses indicate the order of the clustering.

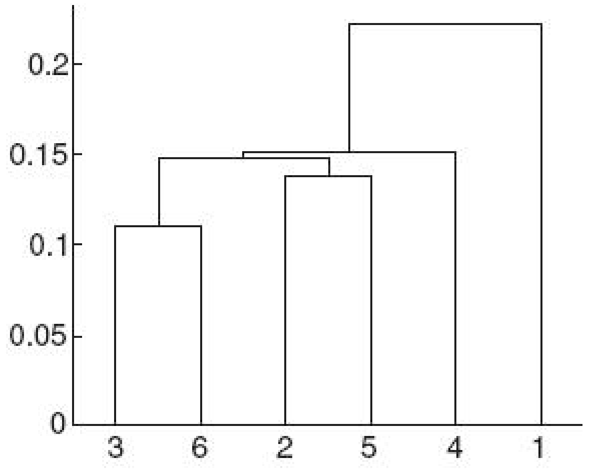
The right figure shows the same information in the form of a dendrogram.

The height at which two clusters are merged in the dendrogram reflects the distance of the two clusters.

Single link



(a) Single link clustering.



(b) Single link dendrogram.

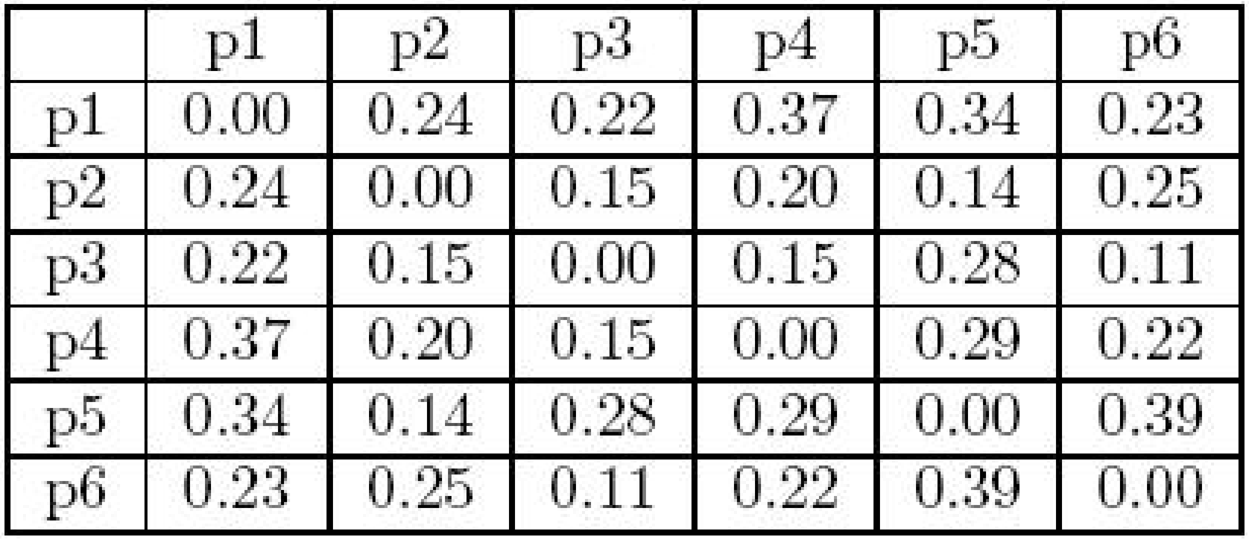
Single link

For example, we see that the distance between points 3 and 6 is 0.11.

That is the height at which they are joined into one cluster in the dendrogram.

As another example, the distance between clusters {3,6} and {2,5} is

$$\begin{aligned} d(\{3,6\},\{2,5\}) &= \min(d(3,2), d(6,2), d(3,5), d(6,5)) \\ &= \min(0.15, 0.25, 0.28, 0.39) \\ &= 0.15 \end{aligned}$$



	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Complete link

We now consider the complete link or MAX version of hierarchical clustering.

In this case, the proximity of two clusters is defined as the maximum of the distance between any two points in the two different clusters.

Complete link is less susceptible to noise and outliers.

However, it tends to produce clusters with globular shapes.

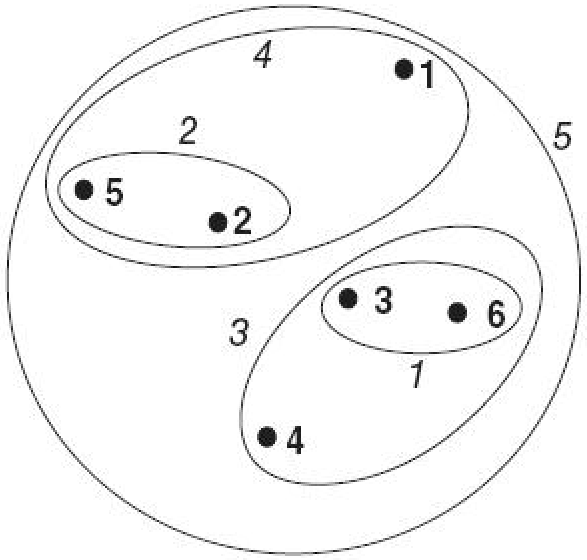
Complete link

The following figure shows the results of applying the complete link approach to our sample data points.

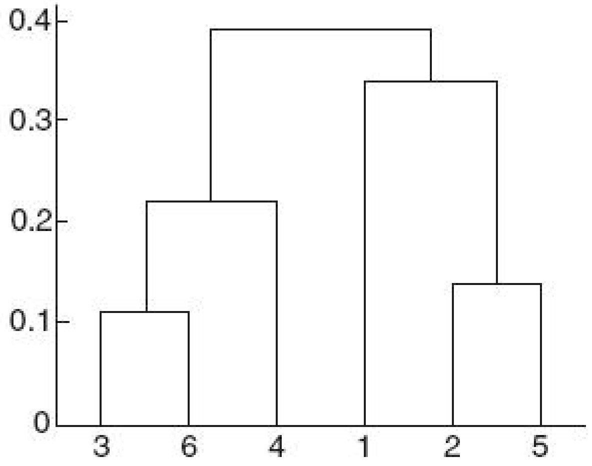
As with single link, points 3 and 6 are merged first.

However, {3,6} is merged with {4}, instead of {2,5} or {1}.

Complete link



(a) Complete link clustering.



(b) Complete link dendrogram.

Complete link

This can be explained by the following calculations

$$
\begin{aligned}
d(\{3,6\},\{4\}) &= \max\left(d(3,4), d(6,4)\right) \\
&= \max(0.15, 0.22) \\
&= 0.22
\end{aligned}
$$

$$
\begin{aligned}
d(\{3,6\},\{2,5\}) &= \max\left(d(3,2), d(6,2), d(3,5), d(6,5)\right) \\
&= \max(0.15, 0.25, 0.28, 0.39) \\
&= 0.39
\end{aligned}
$$

$$
\begin{aligned}
d(\{3,6\},\{1\}) &= \max\left(d(3,1), d(6,1)\right) \\
&= \max(0.22, 0.23) \\
&= 0.23
\end{aligned}
$$

Group average

We now consider the group average version of hierarchical clustering.

In this case, the proximity of two clusters is defined as the average pairwise proximity among all pairs of points in the different clusters.

This is an intermediate approach between the single and complete link approaches.

Group average

We consider two clusters C_i and C_j, which are of sizes m_i and m_j respectively.

The distance between the two clusters can be expressed by the following equation

$$d(C_i, C_j) = \frac{\sum_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y})}{m_i m_j}$$

Group average

The following figure shows the results of applying the group average to our sample data.

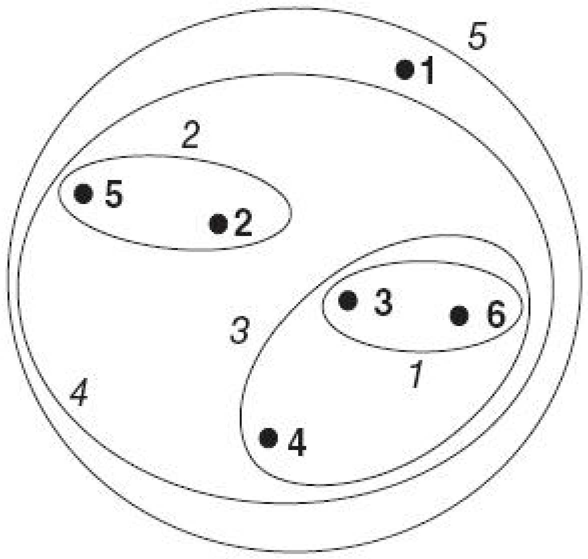
The distance between some of the clusters are calculated as follows:

$$d(\{3,6,4\},\{1\}) = \frac{0.22 + 0.37 + 0.23}{3 \times 1} = 0.27$$

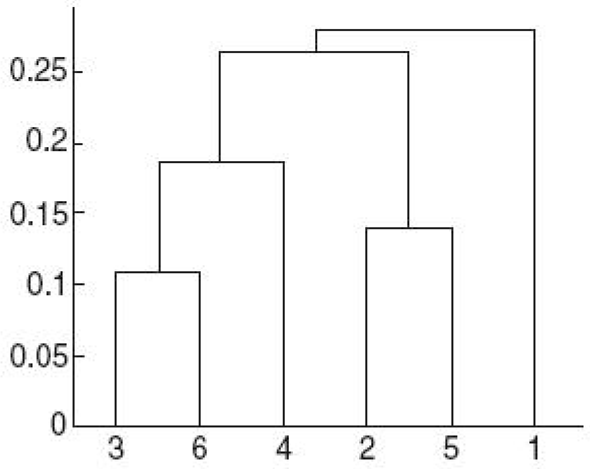
$$d(\{2,5\},\{1\}) = \frac{0.24 + 0.34}{2 \times 1} = 0.29$$

$$d(\{3,6,4\},\{2,5\}) = \frac{0.15 + 0.28 + 0.25 + 0.39 + 0.20 + 0.29}{3 \times 2} = 0.26$$

Group average



(a) Group average clustering.



(b) Group average dendrogram.

Group average

We observe that d({3,6,4},{2,5}) is smaller than d({3,6,4},{1}) and d({2,5},{1}).

As a result, {3,6,4} and {2,5} are merged at the fourth stage.

Ward’s method

We now consider Ward’s method for hierarchical clustering.

In this case, the proximity between two clusters is defined as the increase in the sum of the squared error that results when they are merged.

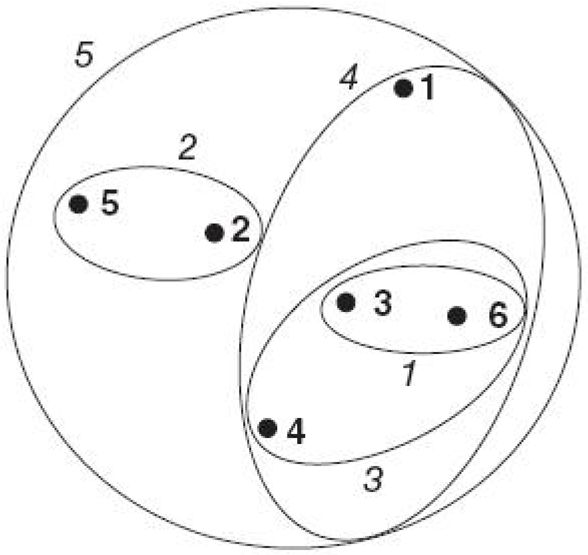
Thus, this method uses the same objective function as k-means clustering.

Ward’s method

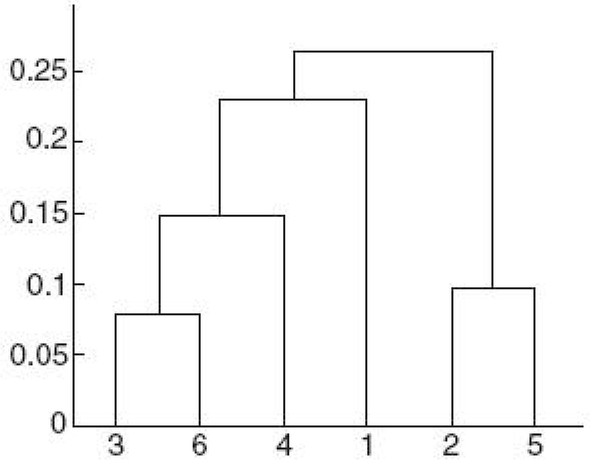
The following figure shows the results of applying Ward’s method to our sample data.

The clustering that is produced is different from those produced by single link, complete link and group average.

Ward's method



(a) Ward's clustering.



(b) Ward's dendrogram.

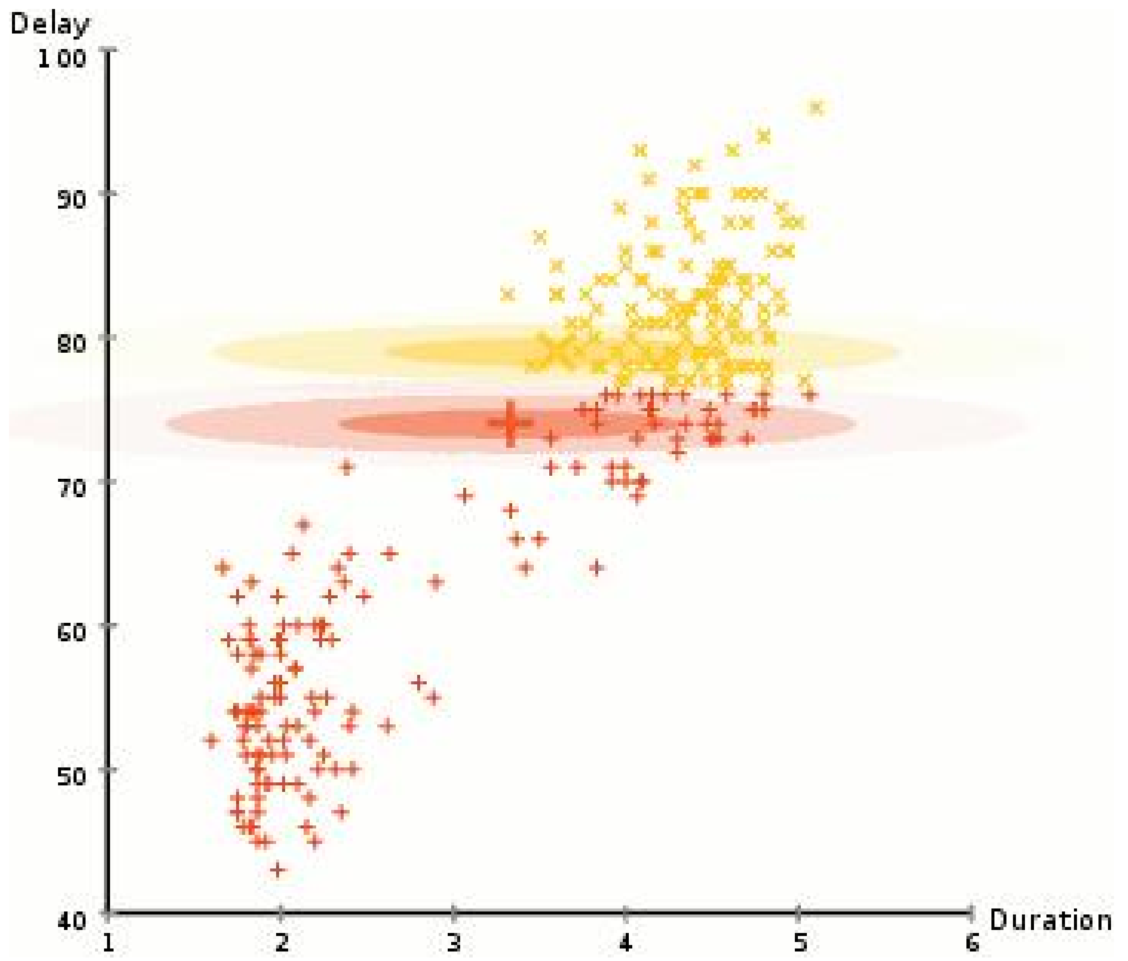
Key issues

Hierarchical clustering is effective when the underlying application requires the creation of a multi-level structure.

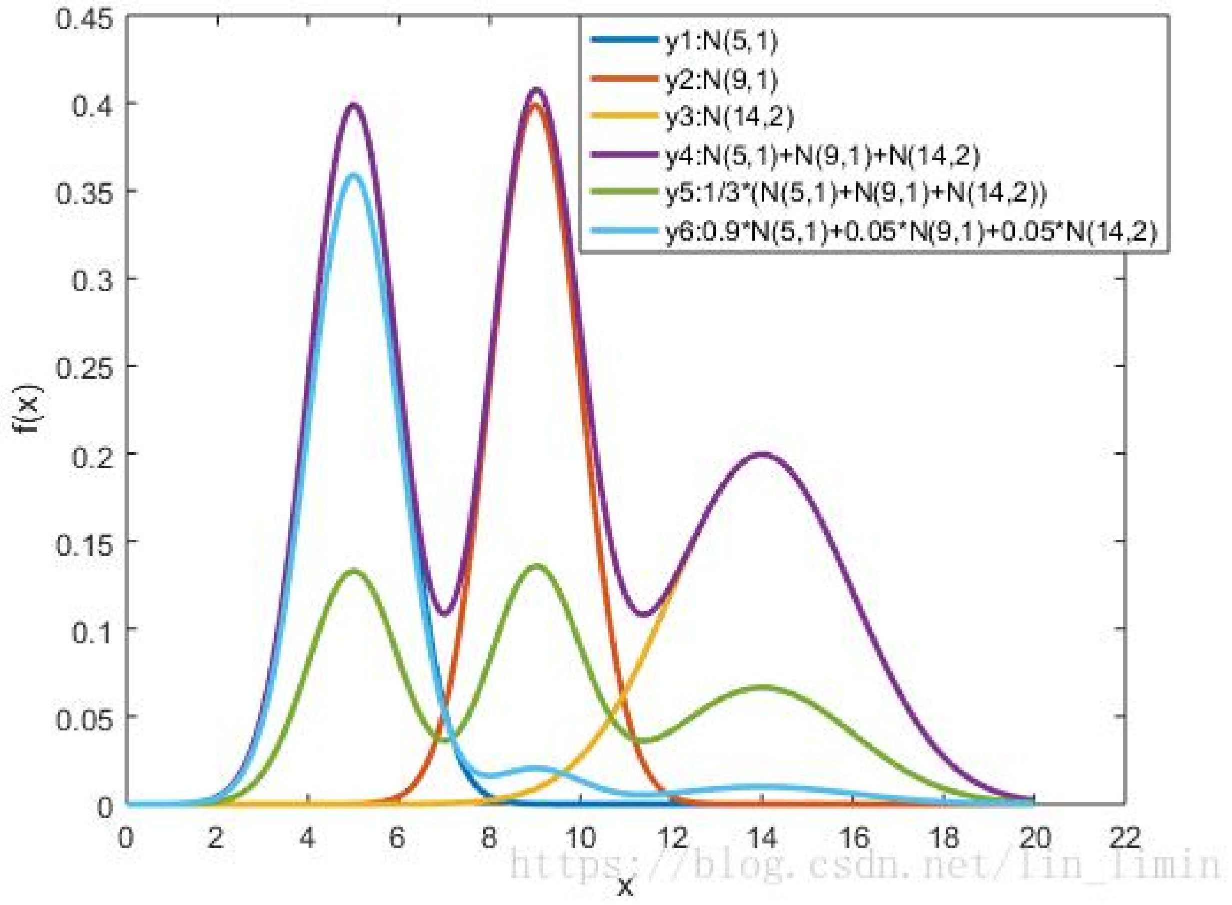
However, they are expensive in terms of their computational and storage requirements.

In addition, once a decision is made to merge two clusters, it cannot be undone at a later time.

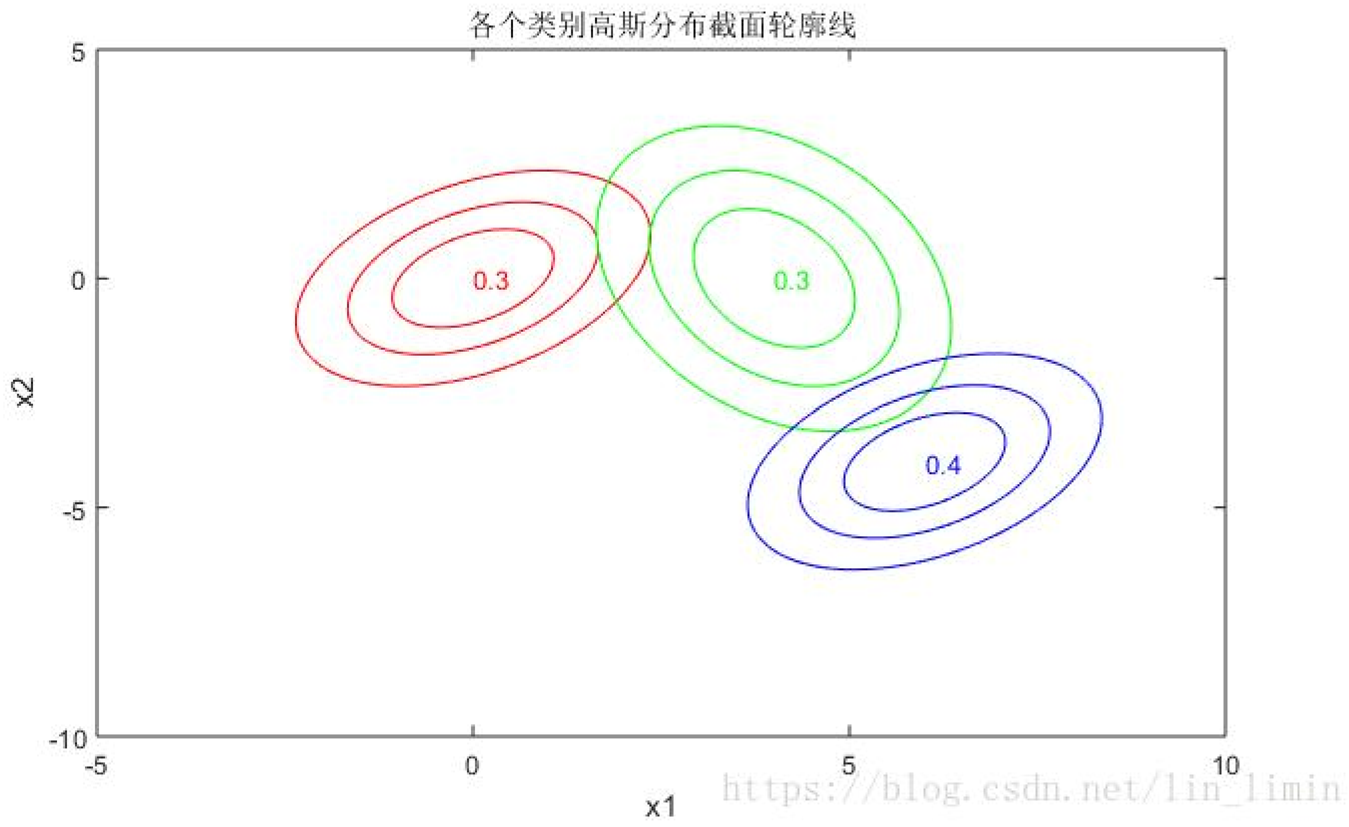
Cluster using Gaussian mixture model



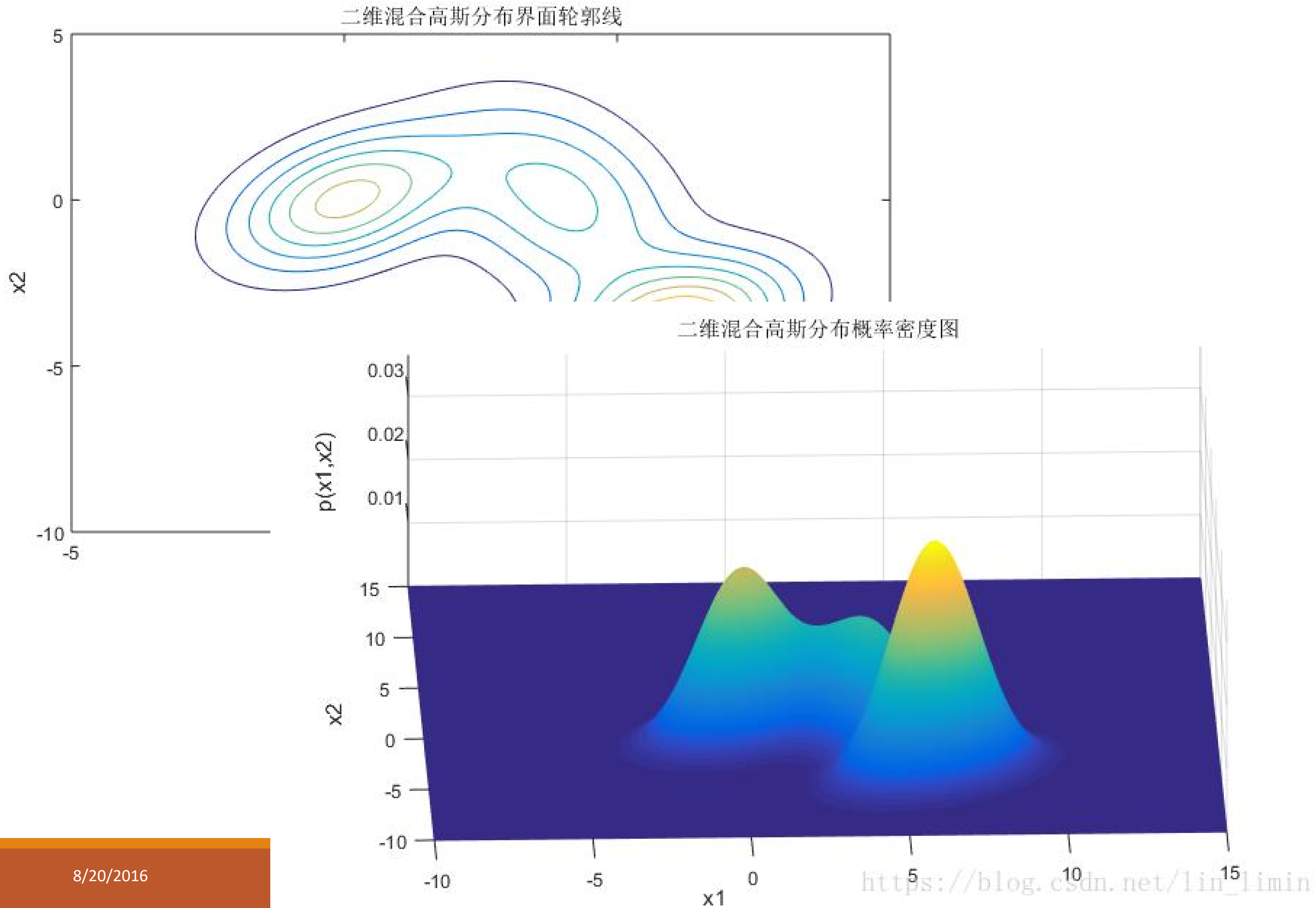
Cluster using Gaussian mixture model



Cluster using Gaussian mixture model



Cluster using Gaussian mixture model



Cluster using Gaussian mixture model

The PDF $p(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$ of multi-variate Gaussian distribution

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

Observed: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$

$\boldsymbol{Z} \sim$ Multinomial Distribution

$\boldsymbol{x} \sim N(\boldsymbol{\mu}_i, \Sigma_i)$

Gaussian mixture model

Mixture coefficient

$$p_{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^{k} \alpha_i \cdot p(\boldsymbol{x}|\boldsymbol{\mu}_i, \Sigma_i), \quad \sum_{i=1}^{k} \alpha_i = 1$$

Cluster using Gaussian mixture model

The process of generating a sample

- Select the i-th mixture component with probability α_i
- Sampling using pdf $p(\boldsymbol{z}|\boldsymbol{\mu}_i, \Sigma_i)$ of the i-th mixture component

Suppose samples in the training set $D = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_n\}$ are sampled in the forementioned process. Let $x_j \in \{1,2, \dots, k\}$ be the mixture component generating sample $\boldsymbol{z}_j$.

Based on Bayes theorem,

$$\underbrace{p_{\mathcal{M}}\left(x_j = i \middle| \boldsymbol{z}_j\right)}_{\gamma_{ji}} = \frac{P\left(x_j = i\right) \cdot p_{\mathcal{M}}(\boldsymbol{z}_j | x_j = i)}{p_{\mathcal{M}}(\boldsymbol{z}_j)}$$

$$= \frac{\alpha_i \cdot p(\boldsymbol{z}_j; \boldsymbol{\mu}_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{z}_j; \boldsymbol{\mu}_l, \Sigma_l)}$$

posterior probability of $\boldsymbol{z}_j$ generated by i-th component

Cluster using Gaussian mixture model

Training set D can be divided into k clusters $\{C_1, C_2, \dots, C_k\}$. The cluster λ_j of $\boldsymbol{z}_j$ can be determined by

$$\lambda_j = \arg \max_{i \in \{1,2,\dots,k\}} \gamma_{ji}, \ \lambda_j \in \{C_1, C_2, \dots, C_k\}$$

If $\alpha_i, \boldsymbol{\mu}_i, \Sigma_i$ are known, γ_{ji} and λ_j can be easily computed

However, $\alpha_i, \boldsymbol{\mu}_i, \Sigma_i$ are unknown

Complete data: $(\boldsymbol{x}_j, \gamma_{j1}, \gamma_{j2}, \dots, \gamma_{jk})$

How to determine parameters $\{(\alpha_i, \boldsymbol{\mu}_i, \Sigma_i) | 1 \le i \le k\}$ in GMM?

The expected value of a function

Let X be a random variable whose PDF is $q(x)$, and g a function of random variable X.

The expected value of $g(x)$ is

$$E[g(X)] = \int_x g(x)q(x)dx \triangleq E_{X\sim q}[g(X)]$$

Pre-requisite: Jensen's inequality

Finite Form: Suppose f is a real concave function, number $x_1,\ x_2,\ \ldots,\ x_n$ in its domain, and positive weights a_i, Jensen's inequality can be stated as:

$$f\left(\frac{\sum a_i x_i}{\sum a_i}\right) \geq \frac{\sum a_i f(x_i)}{\sum a_i}$$

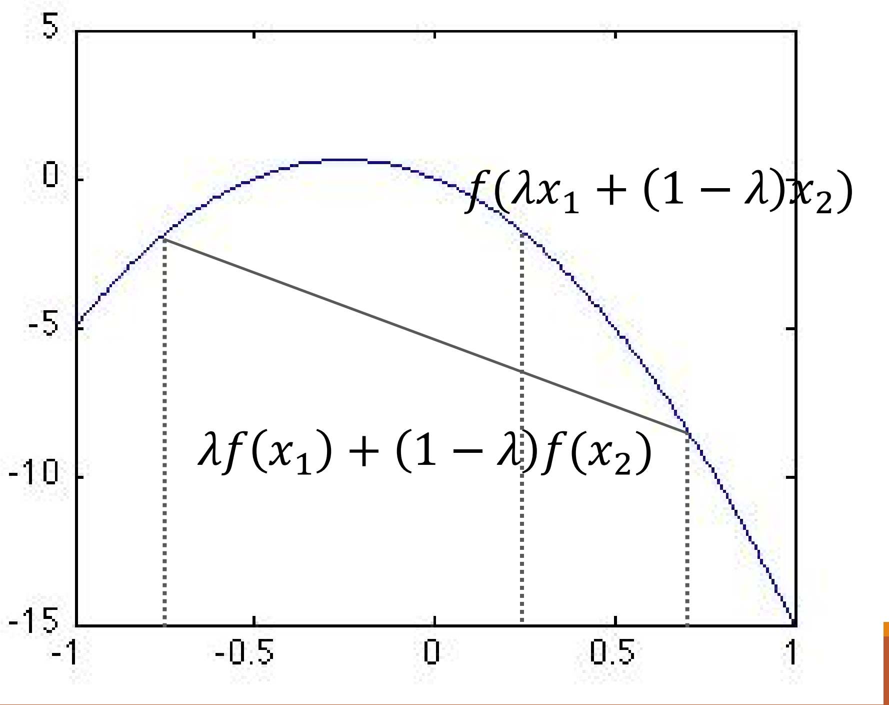
with equality holding only if f is an affine function

If $f\colon \mathbb{R}^m \rightarrow \mathbb{R}^n$ is affine, there is an $n \times m$ matrix M and a vector $\boldsymbol{b}$ in $\mathbb{R}^n$ such that:

$$f(\boldsymbol{x}) = M\boldsymbol{x} + \boldsymbol{b}$$

for all $\boldsymbol{x}$ in $\mathbb{R}^m$

$$a_1 = \lambda, a_2 = 1 - \lambda \Rightarrow$$



Pre-requisite: Jensen’s inequality

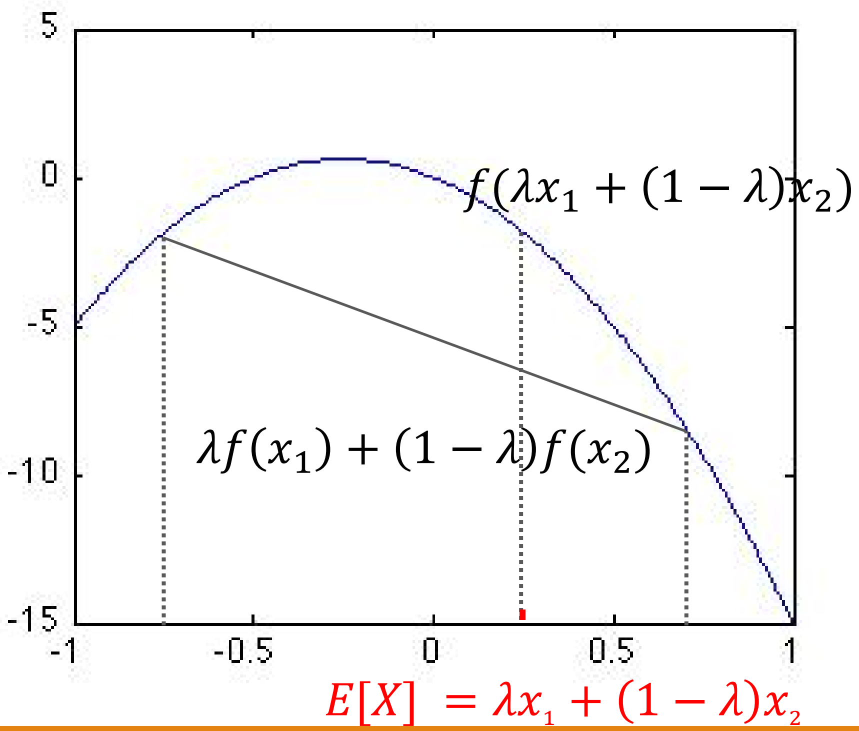
Suppose f is concave, then for all probability measures q we have that:

$$f\left(\int_x xq(x)dx\right) = f\left(E_{X\sim q}\right) \geq E_{X\sim q}[f(X)] = \int_x f(x)q(x)dx$$

with equality holding only if f is an affine function

Illustration:

$$P(X = x_1) = \lambda$$
$$P(X = x_2) = 1 - \lambda$$



Pre-requisite: EM algorithm

If there are latent variables, the set of which is denoted by Z, in samples, how to estimate parameters in a generative model?

The objective function

$$LL(\theta|Z,X) = \ln P(Z,X|\theta)$$

Example:

$$P(Z=1) = \frac{1}{2}, P(Z=2) = \frac{1}{2}$$

$$X|Z=1 \sim \mathcal{N}(\mu_1, 1)$$

$$X|Z=2 \sim \mathcal{N}(\mu_2, 1)$$

Given data $x_1, x_2, \ldots, x_m$ (but no z_i observed)

Find maximum likelihood estimates of μ_1, μ_2

Pre-requisite: EM algorithm

EM basic idea: if Z were known $\rightarrow$ two easy-to-solve separate ML problems

EM iterates over

- **E-step**: For $i = 1, \ldots, m$ fill in missing data z_i according to what is most likely given the current model μ
- **M-step**: run ML for completed data, which gives new model μ

Pre-requisite: EM algorithm

EM solves a Maximum Likelihood problem of the form

$$\max_{\theta} \log \int_{z} p(x, z; \theta) dz$$

θ: parameters of the probabilistic model we try to find

z: unobserved variables

x: observed variables

Pre-requisite: EM algorithm

$$\max_\theta \log \int_z p(x,z;\theta)dx = \max_\theta \log \int_z p(x,z;\theta)\frac{q(z)}{q(z)}dz$$

$$= \max_\theta \log \int_z \frac{p(x,z;\theta)}{q(z)}q(z)dz$$

$$= \max_\theta \log E_{Z\sim q}\left[\frac{p(Z,x;\theta)}{q(Z)}\right]$$

$$\geq \max_\theta E_{Z\sim q} \log\left[\frac{p(Z,x;\theta)}{q(Z)}\right]$$

Jensen's Inequality

$$= \max_\theta \int_z q(z)\log p(x,z;\theta)\, dz$$

$$- \int_z q(z)\log q(z)\, dz$$

Pre-requisite: EM algorithm

$$\max_\theta \log \int_z p(x,z;\theta)dz \geq \max_\theta \int_z q(z)\log p(x,z;\theta)\,dz - \int_z q(z)\log q(z)\,dz$$

Jensen's Inequality: equality holds when $f(x) = \log \frac{p(x,z;\theta)}{q(z)}$ is an affine function

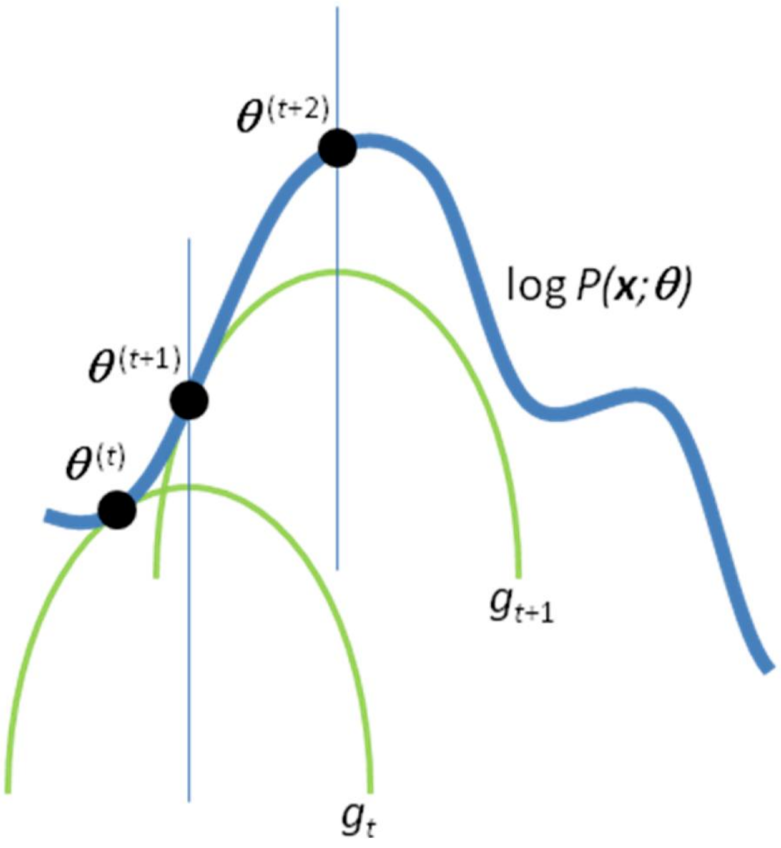
This is achieved for $q(z) = p(z|x;\theta) \propto p(x,z;\theta)$

EM Algorithm: Iterate

1. E-step: Compute $q(z) = p(z|x;\theta), \int_z q(z)\log p(x,z;\theta)\,dz$

2. M-step: Compute $\theta = \arg\max_\theta \int_z q(z)\log p(x,z;\theta)\,dz$

Pre-requisite: EM algorithm



Supplementary Figure 1 Convergence of the EM algorithm. Starting from initial parameters $\theta^{(t)}$, the E-step of the EM algorithm constructs a function g_t that lower-bounds the objective function $\log P(x;\theta)$. In the M-step, $\theta^{(t+1)}$ is computed as the maximum of g_t. In the next E-step, a new lower-bound g_{t+1} is constructed; maximization of g_{t+1} in the next M-step gives $\theta^{(t+2)}$, etc.

Pre-requisite: EM algorithm

M-step optimization can be done efficiently in most cases

E-step is usually the more expensive step

It does not fill in the missing data z with hard values, but finds a distribution $q(z)$

M-step objective is upper bounded by true objective

M-step objective is equal to true objective at current parameter estimate

Improvement in true objective is at least as large as improvement in M-step objective

Cluster using Gaussian mixture model

E-step:

$$\gamma_{ji} = p_{\mathcal{M}}\left(z_j = i \middle| \boldsymbol{x}_j; \alpha_i, \boldsymbol{\mu}_i, \Sigma_i\right) = \frac{p_{\mathcal{M}}(z_j = i, \boldsymbol{x}_j; \alpha_i, \boldsymbol{\mu}_i, \Sigma_i)}{p_{\mathcal{M}}(\boldsymbol{x}_j; \alpha_i, \boldsymbol{\mu}_i, \Sigma_i)}$$

$$= \frac{\alpha_i \cdot p(\boldsymbol{x}_j; \boldsymbol{\mu_i}, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j; \boldsymbol{\mu}_l, \Sigma_l)}$$

$$\propto p_{\mathcal{M}}\left(z_j = i, \boldsymbol{x}_j; \alpha_i, \boldsymbol{\mu}_i, \Sigma_i\right)$$

$$= \alpha_i \cdot p(\boldsymbol{x}_j; \boldsymbol{\mu}_i, \Sigma_i)$$

M-step:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\Sigma}} \sum_{j=1}^{n} \sum_{i=1}^{k} \gamma_{ji} \log\left(\alpha_i \cdot p(\boldsymbol{x}_j; \boldsymbol{\mu}_i, \Sigma_i)\right) \triangleq \max_{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\Sigma}} L(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Cluster using Gaussian mixture model

For α_i, there are two constraints: 1) $\alpha_i \geq 0$; 2) $\sum_{i=1}^{k} \alpha_i = 1$

Construct a Lagrange function $F(\boldsymbol{\alpha}, \lambda)$

$$F(\boldsymbol{\alpha}, \lambda) = L(\boldsymbol{\alpha}) + \lambda \left(\sum_{i=1}^{k} \alpha_i - 1 \right)$$

$$\left. \begin{aligned} \frac{\partial F}{\partial \alpha_i} = 0 \Rightarrow \frac{\sum_{j=1}^{n} \gamma_{ji}}{\alpha_i} + \lambda = 0 \\ \lambda = - n \end{aligned} \right\} \Rightarrow \alpha_i = \frac{1}{n} \sum_{j=1}^{n} \gamma_{ji}$$

Cluster using Gaussian mixture model

If $\{(\alpha_i, \boldsymbol{\mu}_i, \Sigma_i) | 1 \leq i \leq k\}$ maximize $LL(D) \Rightarrow$

$$\frac{\partial L}{\partial \boldsymbol{\mu}_i} = 0 \Rightarrow \frac{\partial\left(\sum_{j=1}^{n} \gamma_{ji} \log\left(a_i p(\boldsymbol{x}_j; \boldsymbol{\mu}_i, \Sigma_i)\right)\right)}{\partial \boldsymbol{\mu}_i} = 0$$

$$\Rightarrow \frac{\partial\left(\sum_{j=1}^{n} \gamma_{ji} \log\left(p(\boldsymbol{x}_j; \boldsymbol{\mu}_i, \Sigma_i)\right)\right)}{\partial \boldsymbol{\mu}_i} = 0$$

$$\Rightarrow \boldsymbol{\mu}_i = \frac{\sum_{j=1}^{n} \gamma_{ji} \boldsymbol{x}_j}{\sum_{j=1}^{n} \gamma_{ji}}$$

Similarly,

$$\frac{\partial L}{\partial \Sigma_i} = 0 \Rightarrow \Sigma_i = \frac{\sum_{j=1}^{n} \gamma_{ji} (\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n} \gamma_{ji}}$$

Cluster using Gaussian mixture model

EM algorithm in GMM

E step:

- Compute posterior probability γ_{ji} using current parameters

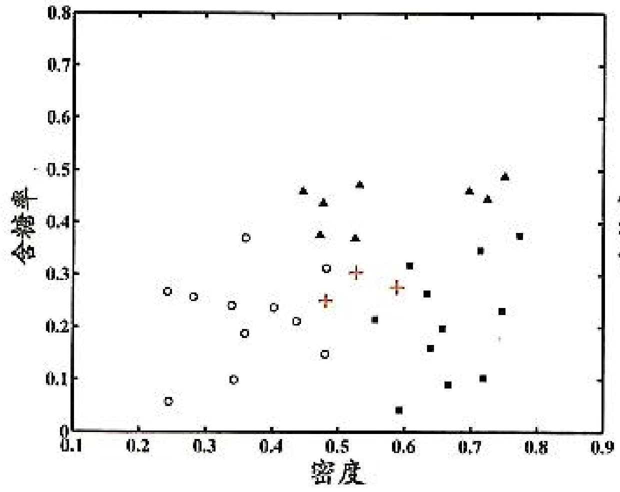
M step:

- Update parameters $\{(\alpha_i, \boldsymbol{\mu}_i, \Sigma_i)\}$

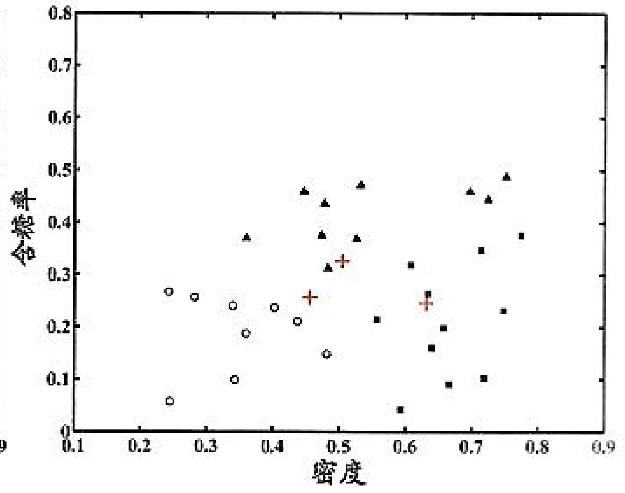
Cluster using Gaussian mixture model

```
Input: Dataset $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$; # of Gaussian mixture component
1:  Initialize parameters $\{(\alpha_i, \boldsymbol{\mu}_i, \Sigma_i)\}$
2:  repeat
3:      for $j = 1,2,..,n$ do
4:          compute $\gamma_{ji} = p_{\mathcal{M}}(z_j = i | \boldsymbol{x}_j)(1 \le i \le k)$
5:      for $i = 1,2,\ldots,k$ do
6:          compute $\boldsymbol{\mu}_i' = \frac{\sum_{j=1}^n \gamma_{ji}\boldsymbol{x}_j}{\sum_{j=1}^n \gamma_{ji}}, \Sigma_i' = \frac{\sum_{j=1}^n \gamma_{ji}(\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^n \gamma_{ji}}, \alpha_i' = \frac{1}{n}\sum_{j=1}^n \gamma_{ji}$
7:      update $\{(\alpha_i, \boldsymbol{\mu}_i, \Sigma_i)\}$ with $\{(\alpha_i', \boldsymbol{\mu}_i', \Sigma_i')\}$
8:  until stop criteria is satisfied
9:  $C_i = \emptyset (1 \le i \le k)$
10: for $j = 1,2,..,n$ do
11:     compute $\lambda_j$
12:     $C_{\lambda_j} = C_{\lambda_j} \cup \{\boldsymbol{x}_j\}$
Output: $\{C_1, C_2, \ldots, C_k\}$
```

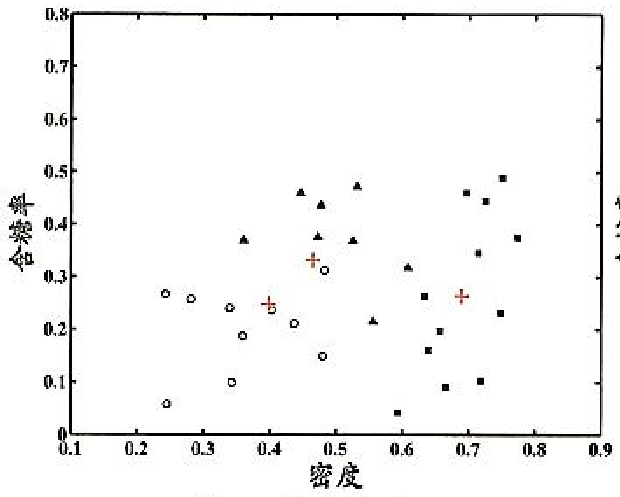
Cluster using Gaussian mixture model



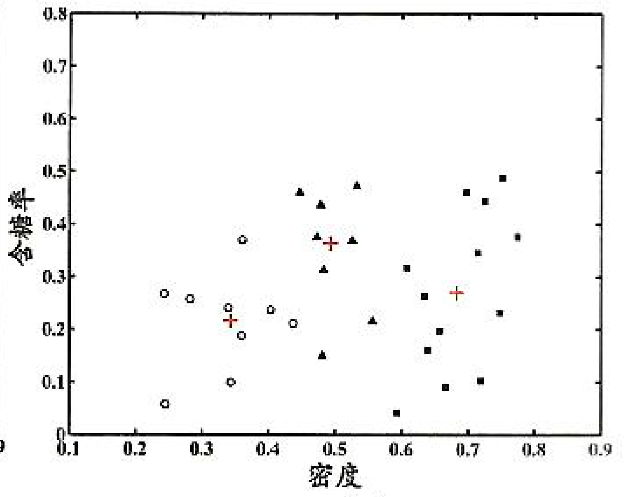
(a) 5 轮迭代后



(b) 10 轮迭代后

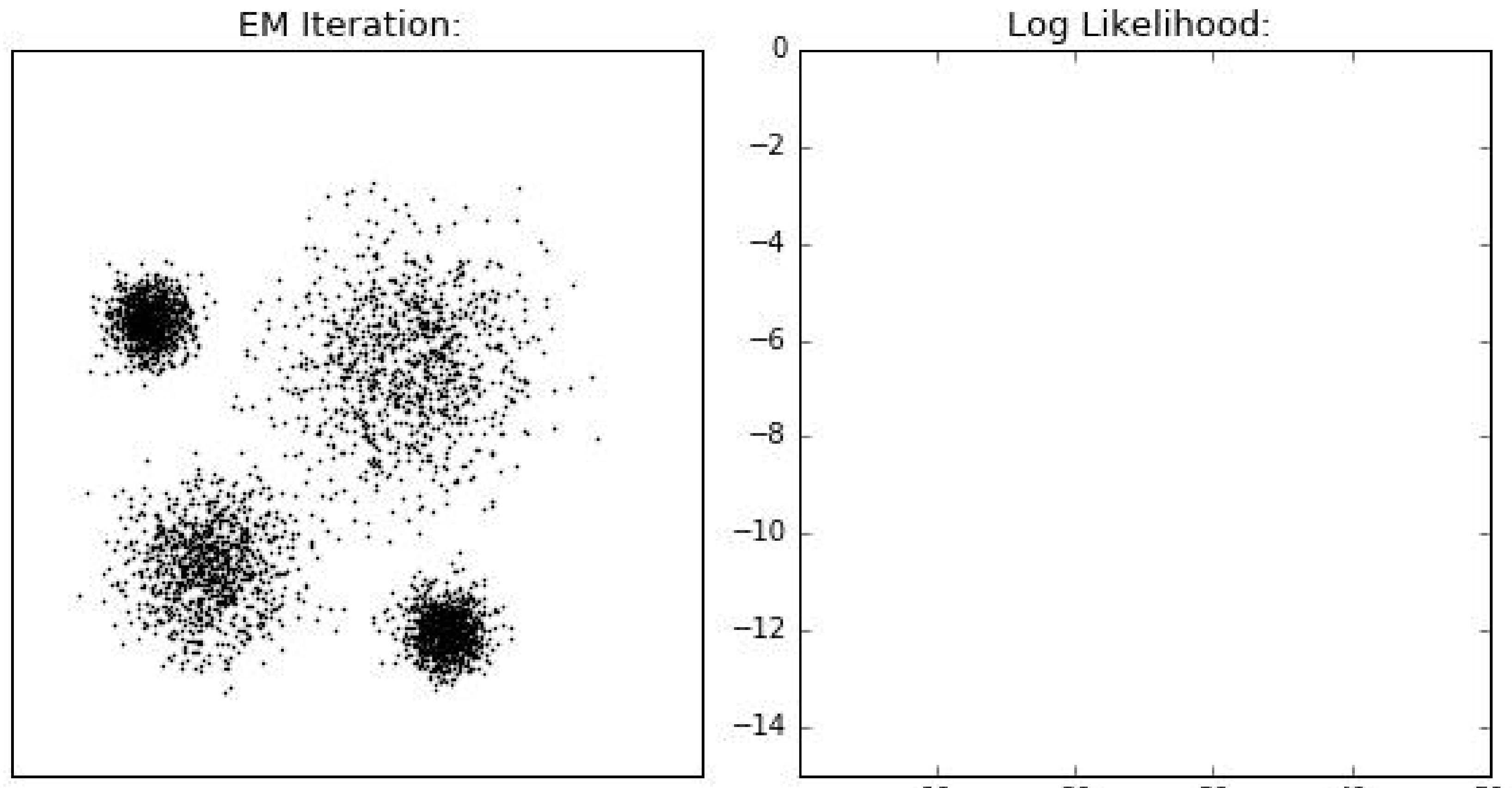


(c) 20 轮迭代后



(d) 50 轮迭代后

Cluster using Gaussian mixture model


EM Iteration: