Clustering

Fatemeh Mansoori

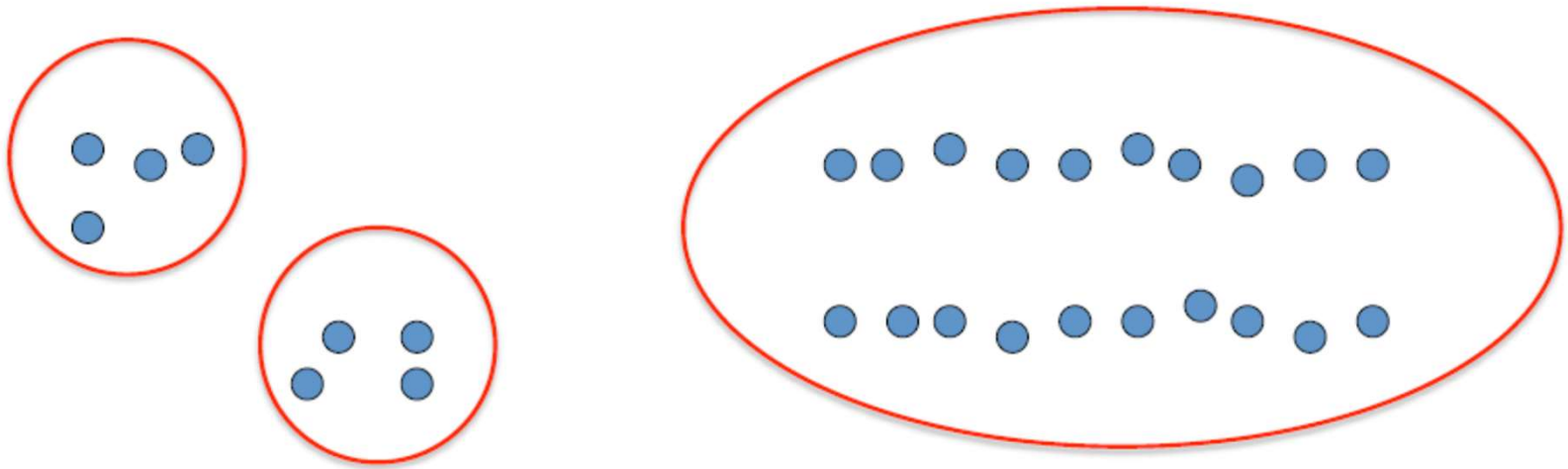
Cluster and Clustering

- What is a Cluster?
  - Comprising a group of data points whose inter-point distances are small compared with the distances to points outside of the cluster.

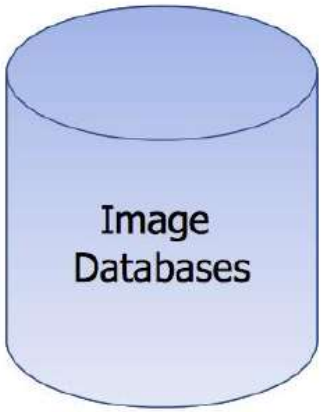
- What is Clustering ?
  - The grouping of objects such that objects in the same cluster are more similar to each other than they are to objects in another cluster

Clustering

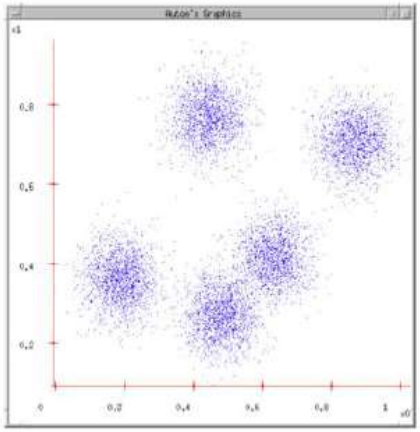
- Basic idea: group together similar instances
- Example: 2D point patterns



Clustering Images



Goal of clustering:
Divide object into groups, and objects within a group are more similar than those outside the group



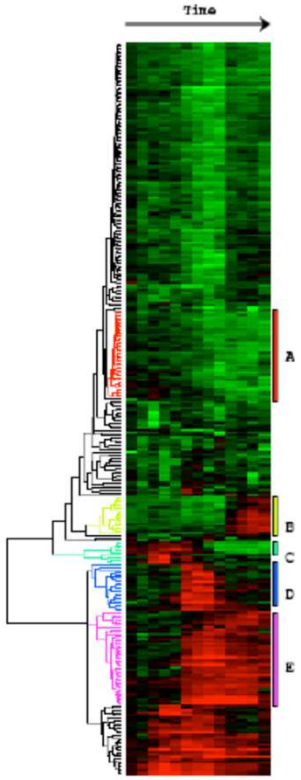
Clustering examples

Image segmentation

Goal: Break up the image into meaningful or perceptually similar regions

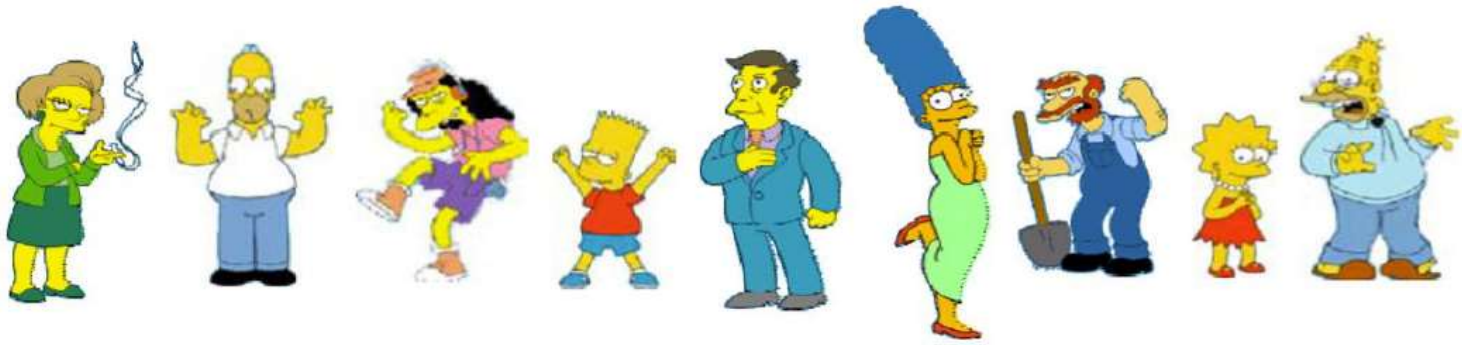
Clustering examples

Clustering gene expression data



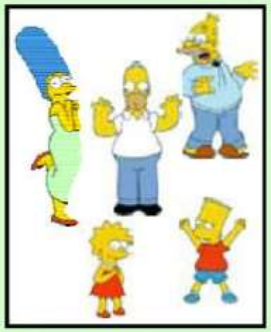
Eisen et al, PNAS 1998

Clustering is Subjective



What is consider similar/dissimilar?

Clustering is subjective



Simpson's Family

School Employees

Females

Males

Are they similar or not?

So What is Clustering in General?

You pick your similarity/dissimilarity function

The algorithm figures out the grouping of objects based on the chosen similarity/dissimilarity function

- Points within a cluster is similar
- Points across clusters are not so similar

Issues for clustering

- How to represent objects?
- What is a similarity/dissimilarity function for your data?
- What are the algorithm steps?

Distance Functions for Vectors

Suppose two data points, both in R^d

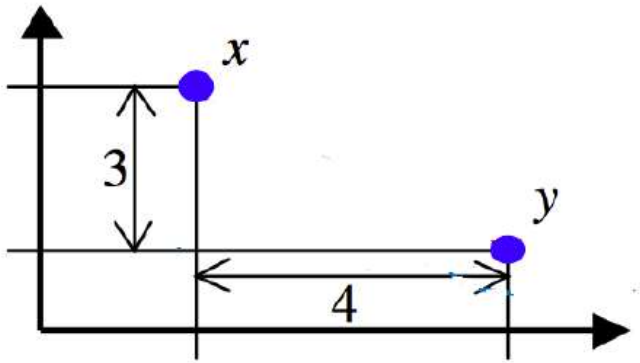
- $x = (x_1, x_2, \ldots, x_d)^T$
- $y = (y_1, y_2, \ldots, y_d)^T$

Euclidean distance: $d(x, y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$

Minkowski distance: $d(x, y) = \sqrt[p]{\sum_{i=1}^{d}(x_i - y_i)^p}$

- Euclidean distance: $p = 2$
- Manhattan distance: $p = 1, d(x, y) = \sum_{i=1}^{d}|x_i - y_i|$
- "inf"-distance: $p = \infty, d(x, y) = \max_{i=1}^{d}|x_i - y_i|$

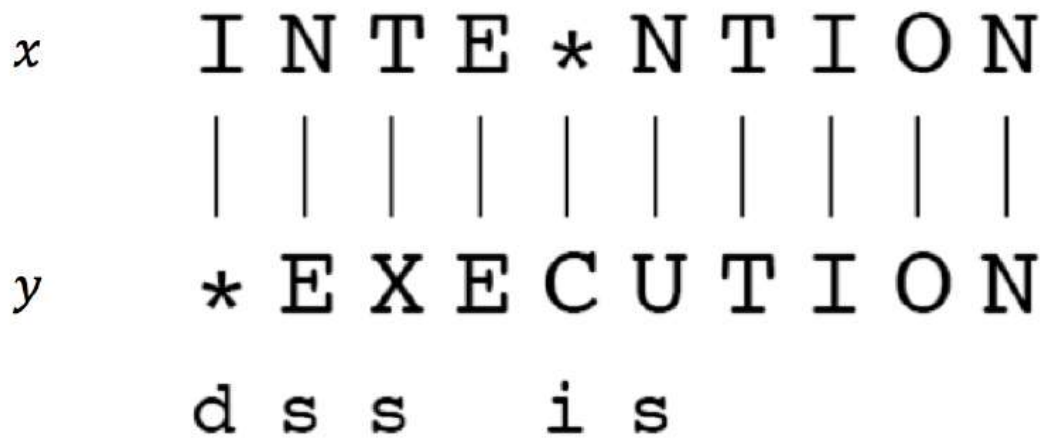
Example



- Euclidean distance: $\sqrt{4^2 + 3^2} = 5$
- Manhattan distance: $4 + 3 = 7$
- "inf"-distance: $max\{4,3\} = 4$

Edit Distance

- Transform one of the objects into the other, and measure how much effort it takes



d: deletion (cost 5)
s: substitution (cost 1)
i: insertion (cost 2)

$$d(x, y) = 5 \times 1 + 3 \times 1 + 1 \times 2 = 10$$

What properties should a distance measure have?

- Symmetric
  - $D(A,B)=D(B,A)$
  - Otherwise, we can say A looks like B but B does not look like A
- Positivity, and self-similarity
  - $D(A,B)\geq 0$, and $D(A,B)=0$ iff $A=B$
  - Otherwise there will different objects that we cannot tell apart
- Triangle inequality
  - $D(A,B)+D(B,C) \geq D(A,C)$
  - Otherwise one can say "A is like B, B is like C, but A is not like C at all"

Formal statement of the K-Means

Given n data points, $\{x^1, x^2, \ldots x^n\} \in R^d$

Find k cluster centers, $\{c^1, c^2, \ldots, c^k\} \in R^d$

And assign each data point i to one cluster, $\pi(i) \in \{1, \ldots, k\}$

Such that the averaged square distance from each data point to respective cluster center(distortion metric) is minimum:

$$\min_{c,\pi} \frac{1}{n} \sum_{i=1}^{n} \|x^i - c^{\pi(i)}\|^2$$

Clustering is NP-Hard

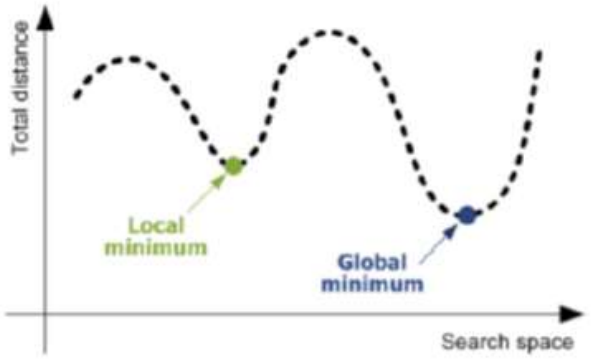
Find k cluster centers, $\{c^1, c^2, \ldots, c^k\} \in R^d$, and assign each data point i to one cluster, $\pi(i) \in \{1, \ldots, k\}$, to minimize

$$\min_{c,\pi} \frac{1}{n} \sum_{i=1}^{n} \left\| x^i - c^{\pi(i)} \right\|^2$$

NP-hard!

A search problem over the space of discrete assignments

- For all n data point together, there are k^n possibility
- The cluster assignment determines cluster centers, and vice versa



- For all n data point together, there are k^n possibility

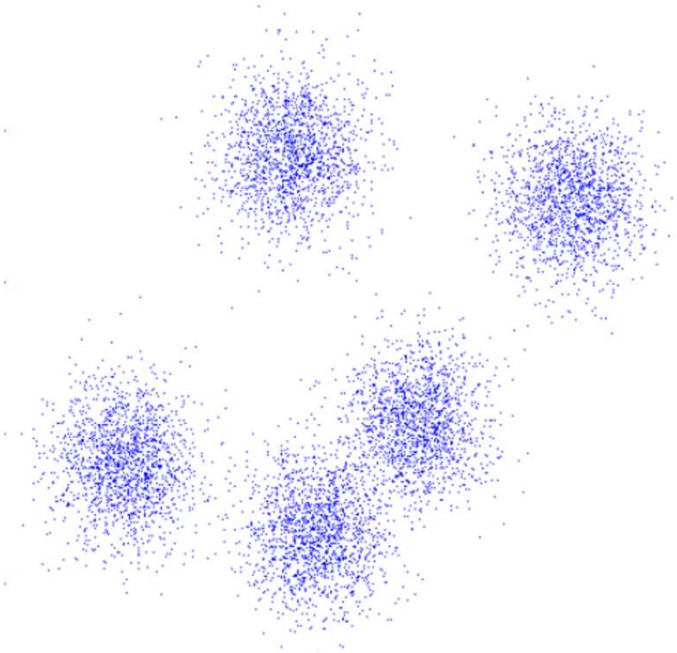
X = {A,B,C}
n=3 (data points)

k=2 clusters of two members

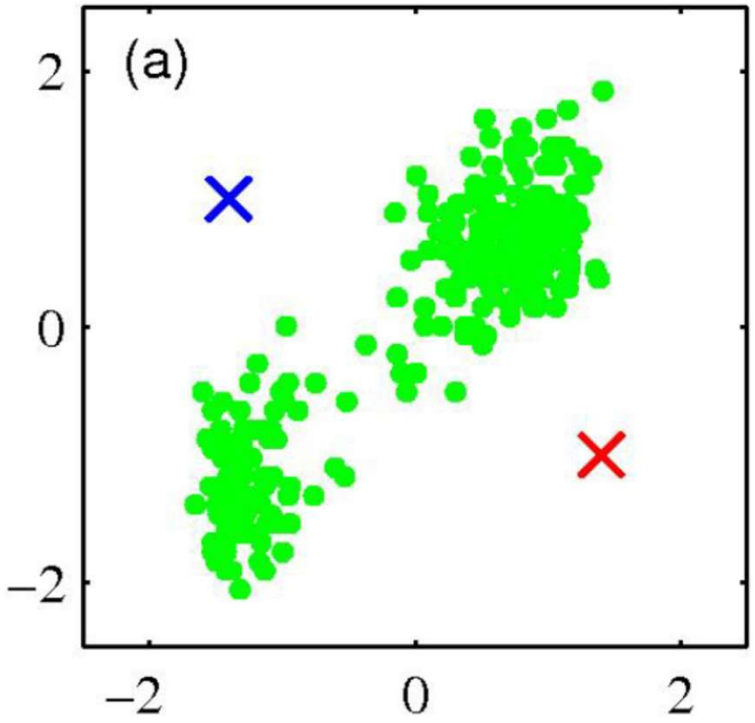
Cluster 1	Cluster 2
A	BC
AB	C
AC	B
ABC	{}
{}	ABC
BC	A
C	AB
B	AC

K-Means

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change



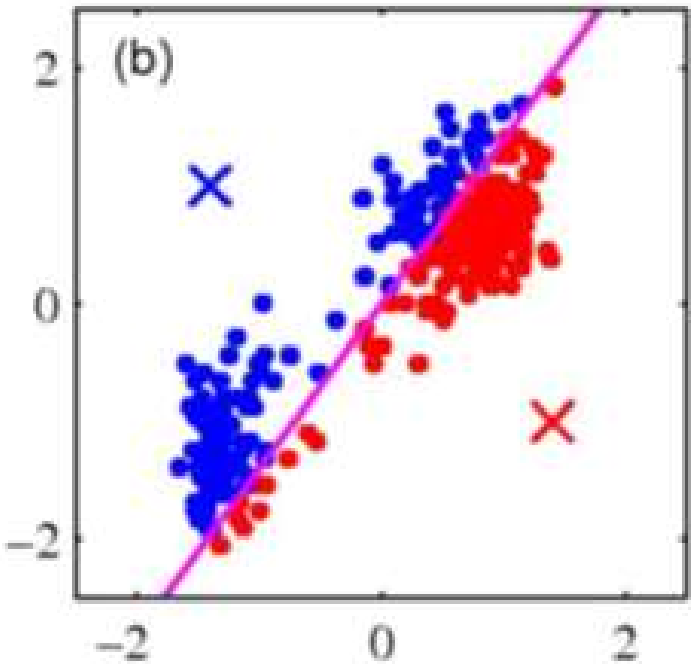
K-means clustering: Example



- Pick *K* random points as cluster centers (means)

Shown here for *K*=2

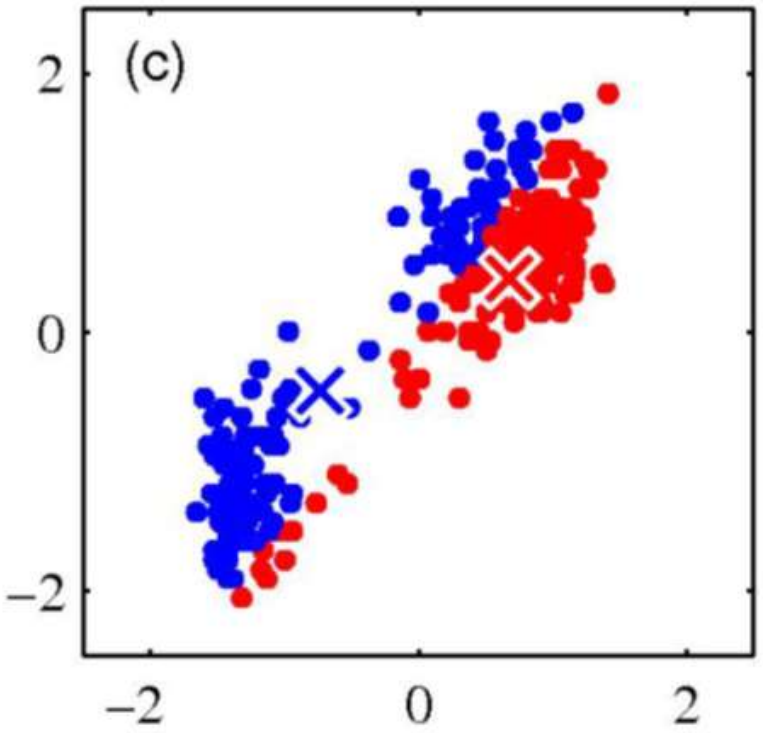
K-means clustering: Example



Iterative Step 1

- Assign data points to closest cluster center

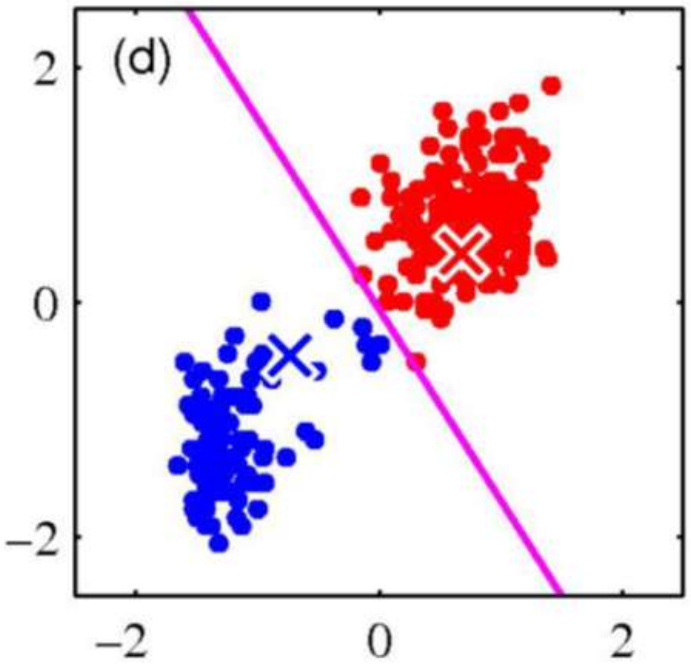
K-means clustering: Example



Iterative Step 2

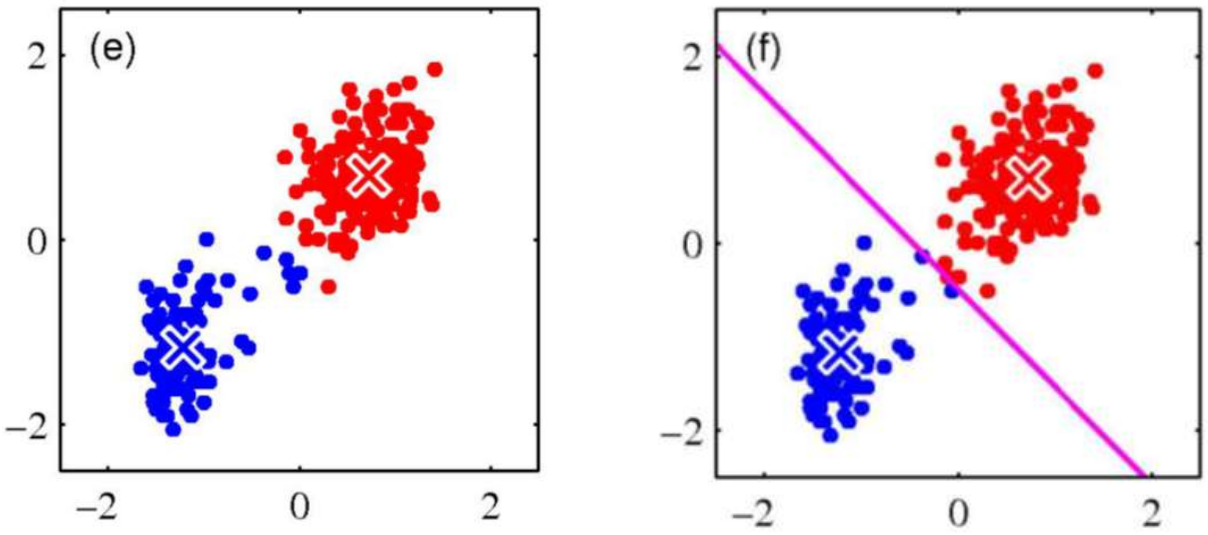
- Change the cluster center to the average of the assigned points

K-means clustering: Example



- Repeat until convergence

K-Mean clustering Example



Question

Will different initialization lead to different results?

- Yes
- No
- Sometimes

Will the algorithm always stop after some iteration?

- Yes
- No (we have to set a maximum number of iterations)
- Sometimes

Results of K-Means Clustering:

Image

Clusters on intensity

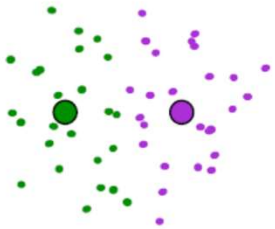
Clusters on color

K-means clustering using intensity alone and color alone

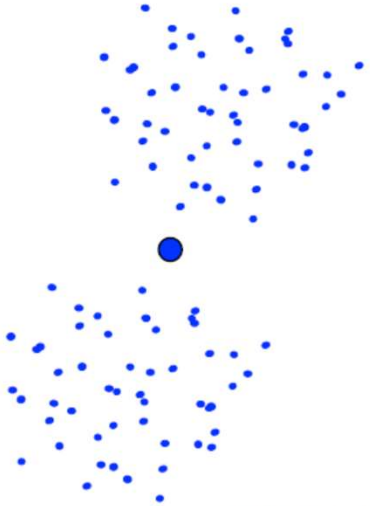
Example: K-Means for Segmentation

K-Means Getting Stuck

A local optimum:

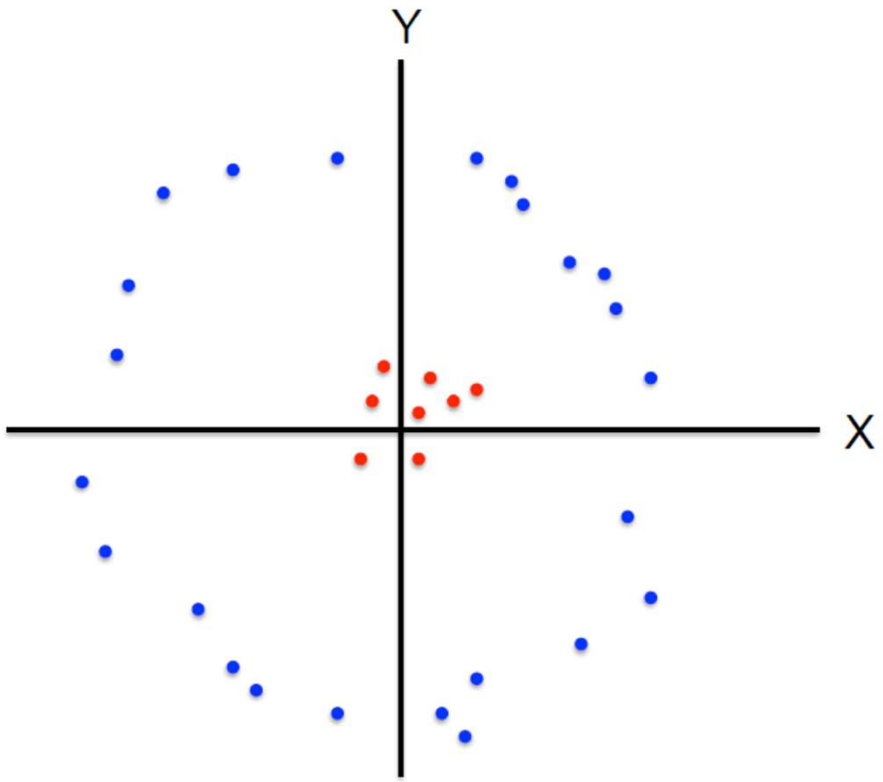


Would be better to have
one cluster here

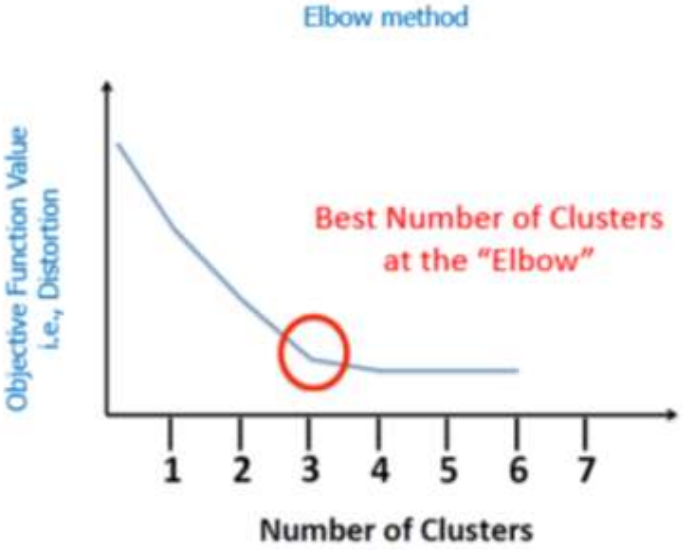


… and two clusters here

K-means not able to properly cluster



How to Choose K?



Distortion score: computing the sum of squared distances from each point to its assigned center

Davies-Bouldin(DB)index

To evaluate the quality of a clustering a plethora of validity indices have been proposed, one of which is Davies-Bouldin (DB) index.

Cluster Dispersion: Which can be interpreted as a generalized standard deviation.

$$\delta_k := \sqrt{\frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \|x_n - c_k\|^2}$$

Cluster Similarity: Is defined such that two clusters are considered similar if they have large dispersion relative to their distance.

$$S_{kl} := \frac{\delta_k + \delta_l}{\|c_k - c_l\|}$$

A good clustering should be characterized by clusters being as dissimilar as possible.

Considering aforementioned definitions, an overall validation of the clustering can be done by the DB index:

$$V_{DB} := \frac{1}{k} \sum_{k=1}^{K} \max_{l \neq k} S_{kl}$$

Davies-Bouldin(DB)index

- Lower value of DB Index means that the clusters are well-separated and compact
  - indicates that the clusters are distinct from each other
  - data points within each cluster are close to each other.
- DB index does not systematically depend on K and is suitable to find the best optimal number of clusters.

Properties of K-Means algorithm

- Guaranteed to converge in a finite number of iterations
- Running time per iterations:
  - Assign data points to closest data center : O(KN)
  - Change the cluster center to the average of its assigned data points : O(N)
- Sensitive to outliers
  - Consider six points in 1-D space having the values 1, 2, 3, 8, 9, 10, and 25
- How can we improve K-Means in this regard?

Kmeans ++

1. Choose one center uniformly at random among the data points.
2. For each data point x not chosen yet, 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 $D(x)$.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard

hierarchical clustering

- A **hierarchical clustering method** works by grouping data objects into a hierarchy or “tree” of clusters

- useful for data summarization and visualization

- hierarchical partitioning can be continued recursively until a desired granularity is reached.

- can be either *agglomerative* or *divisive*, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or top down (splitting) fashion

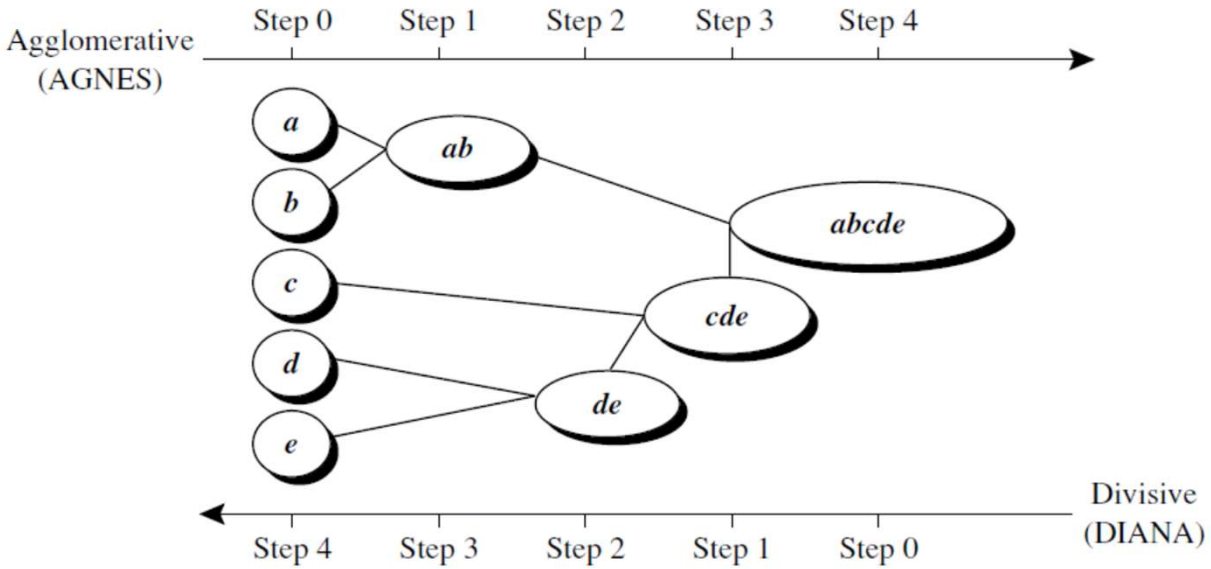
Agglomerative hierarchical clustering

- uses a bottom-up strategy.
- starts by letting each object form its own cluster and iteratively merges clusters
- into larger and larger clusters, until all the objects are in a single cluster or certain termination conditions are satisfied.
- For the merging step, it finds the two clusters that are closest to each other (according to some similarity measure), and combines the two to form one cluster
- Because two clusters are merged per iteration, where each cluster contains at least one object, an agglomerative method requires at most *n* iterations.

Divisive hierarchical clustering

- employs a top-down strategy
- starts by placing all objects in one cluster, which is the hierarchy's root. It then divides the root cluster into several smaller sub clusters, and recursively partitions those clusters into smaller ones
- The partitioning process continues until each cluster at the lowest level is coherent enough—either containing only one object, or the objects within a cluster are sufficiently similar to each other.

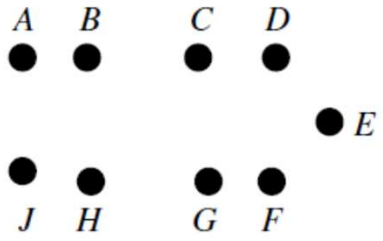
AGNES and DIANA



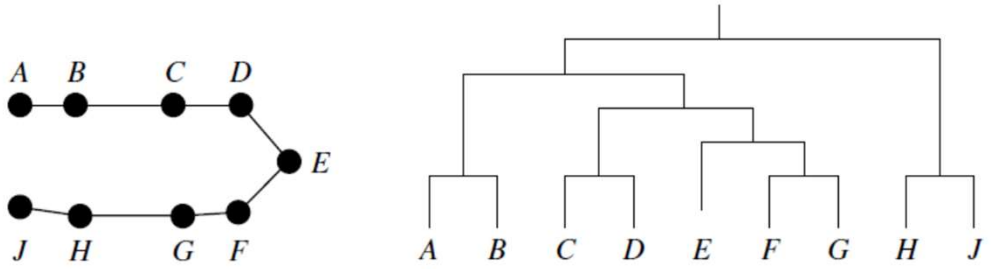
AGNES : the clusters are merges according some principle such as
single-linkage : the similarity between two clusters is measured by the similarity of the *closest* pair of data points

Complete-linkage: the distance between clusters equals the distance between those two elements (one in each cluster) that are farthest away from each other

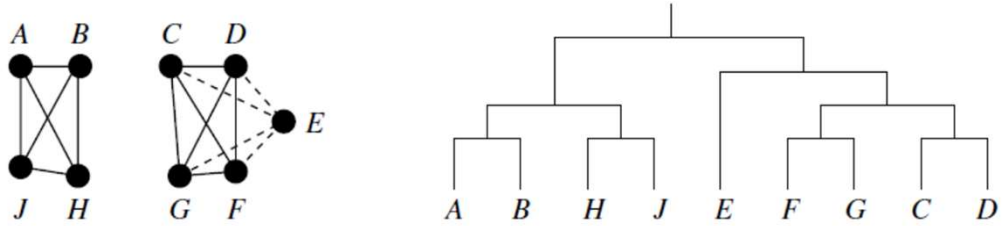
DIANA:
chooses the object with the maximum average dissimilarity and then moves all objects to this cluster that are more similar to the new cluster than to the remainder.



(a) Data set



(b) Clustering using single linkage



(c) Clustering using complete linkage

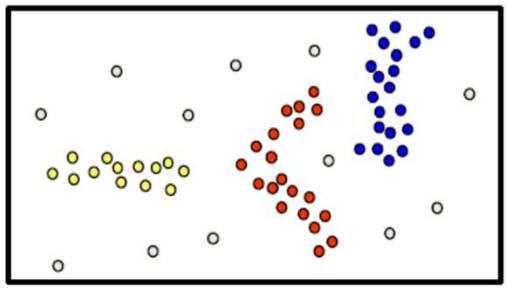
Density based methods

Basic idea

- Clusters are dense regions in the data space, separated by regions of lower object density
- A cluster is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape

Method

- DBSCAN



DBSCAN

- **Density-Based Clustering Based on Connected Regions with High Density**
- *"How can we find dense regions in density-based clustering?"*
  - *density* of an object ***o*** can be measured by the number of objects close to ***o***.

- DBSCAN : finds *core objects*, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters.

How quantify the neighborhood of an object

- A user-specified parameter $\varepsilon > 0$ is used to specify the radius of a neighborhood, consider for each object.

- The **ε-neighborhood** of an object ***o*** is the space within a radius centered at ***o***.

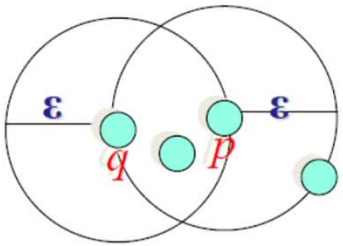
- Due to the fixed neighborhood size parameterized by ε , the **density of a neighborhood** can be measured simply by the number of objects in the neighborhood.

Density Definition

ε-Neighborhood – Objects within a radius of ε from an object.

$$N_{\varepsilon}(p) : \{q \mid d(p,q) \leq \varepsilon\}$$

"High density" - ε-Neighborhood of an object contains at least *MinPts* of objects.

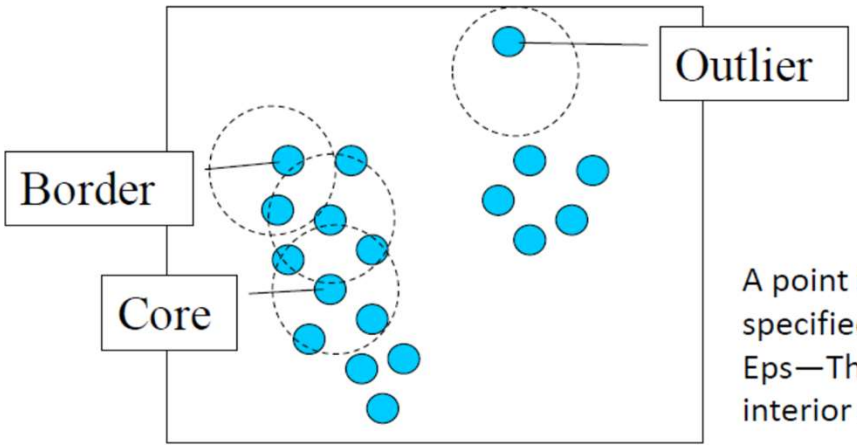


ε-Neighborhood of p
ε-Neighborhood of q

Density of p is "high" (MinPts = 4)

Density of q is "low" (MinPts = 4)

Core, border and outlier



$\varepsilon = 1$unit, MinPts = 5

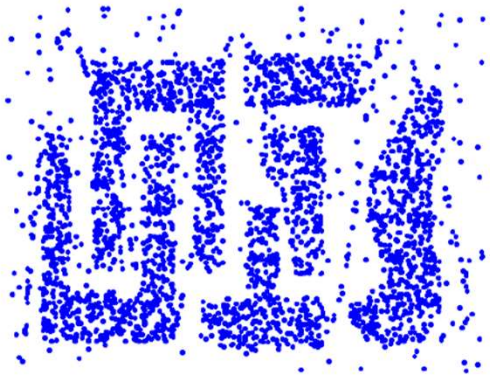
Given ε and *MinPts*, categorize the objects into three exclusive groups.

A point is a core point if it has more than a specified number of points (MinPts) within Eps—These are points that are at the interior of a cluster.

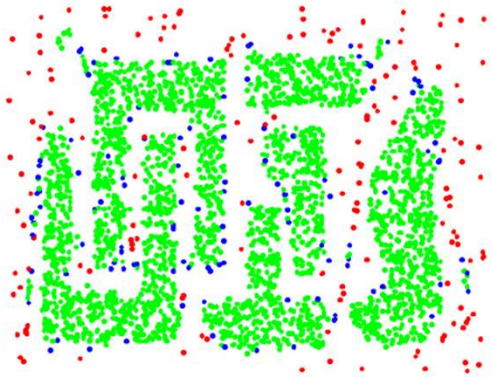
A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

Example



Original Points



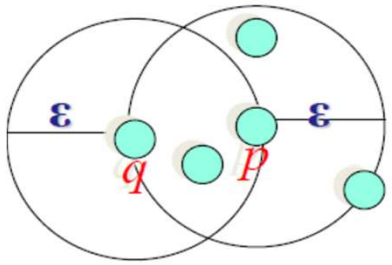
Point types: core,
border and outliers

$\varepsilon = 10$, MinPts = 4

Density-reachability

Directly density-reachable

- An object *q* is directly density-reachable from object *p* if *p* is a core object and *q* is in p's ε-neighborhood.

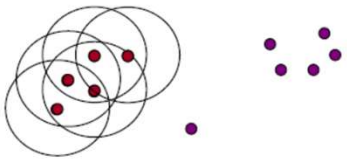


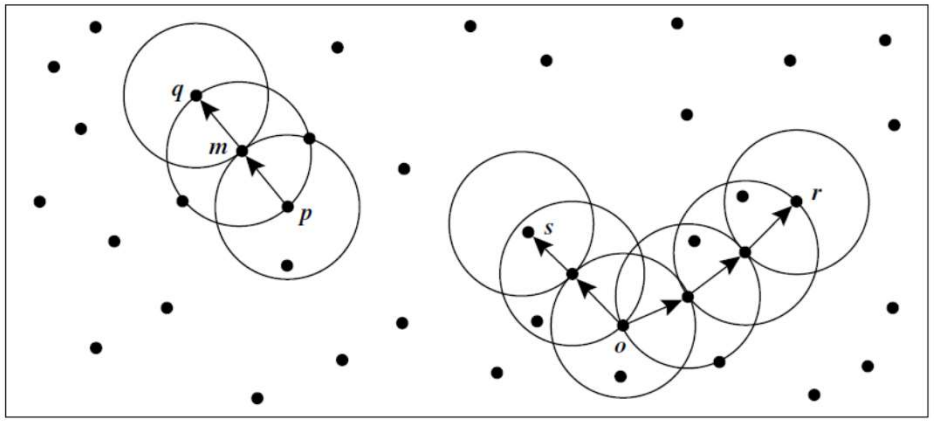
MinPts = 4

- *q* is directly density-reachable from *p*
- *p* is not directly density-reachable from *q*
- Density-reachability is asymmetric

Algorithm

- ε = 2 cm
- *MinPts* = 3






```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

```
Function DBSCAN(X,ε,MinPts)
    C=0
    forall x_n ∈ X do
        if x_n has not been visited mark x_n as visited
        V_n= regionQuery(x_n,ε)
        if sizeof(V_n)≥MinPts then
            expandCluster(x_n,V_n,C++,ε,MinPts)
        else
            mark x_n as noise
        end
    end
Function expandCluster(x_n,V_n,C,ε,MinPts)
    Add x_n to cluster C
    forall x_i ∈ V_n do
        if x_i has not been visited then
            Mark x_i as visited
            V_i = regionQuery(x_i,ε)
            if sizeof(V_i) > MinPts then
                V_n+ = V_i
            end
        end
        if x_i does not belong to any cluster yet: Add x_i to cluster C
    end
Function regionQuery(x_n,ε)
    forall x_i ∈ X do
        if i ≠ n and d(x_n, x_i) ≤ ε: list.add({x_i})
    end
    return list
```

DBSCAN: sensitive to parameter

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

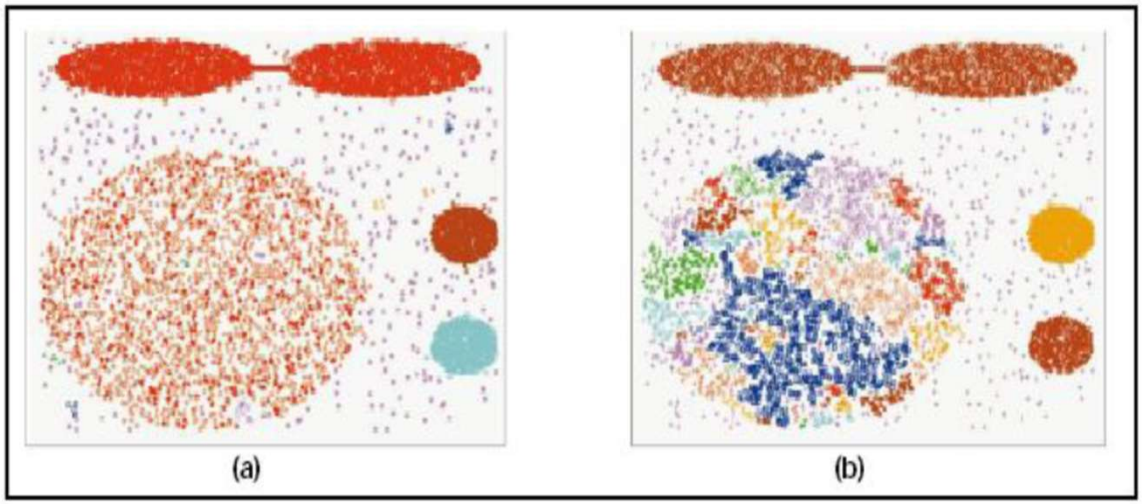
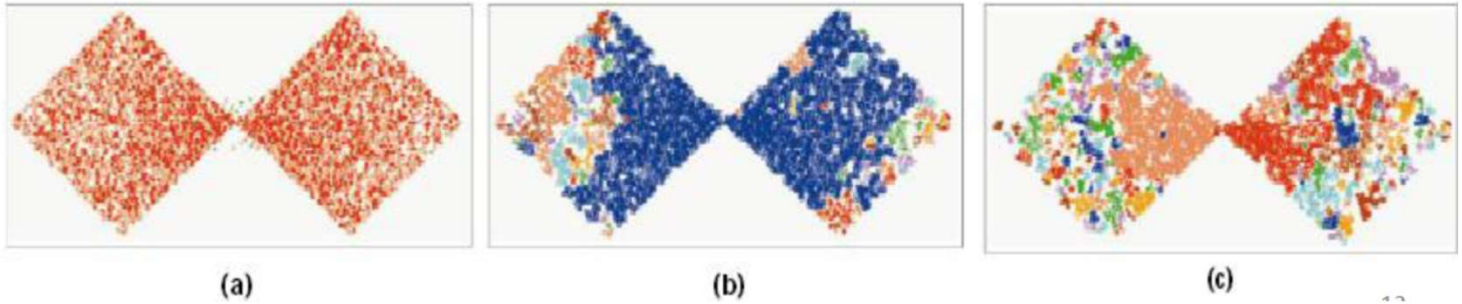
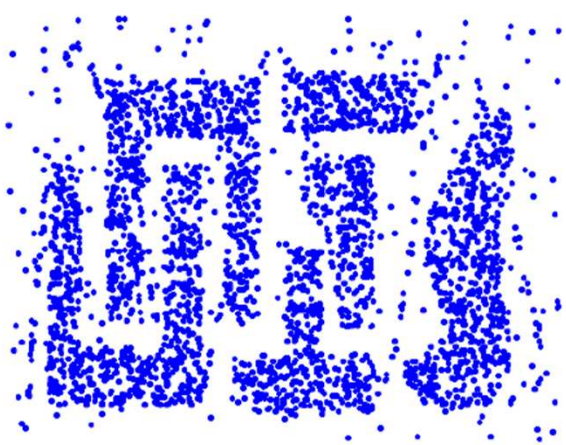


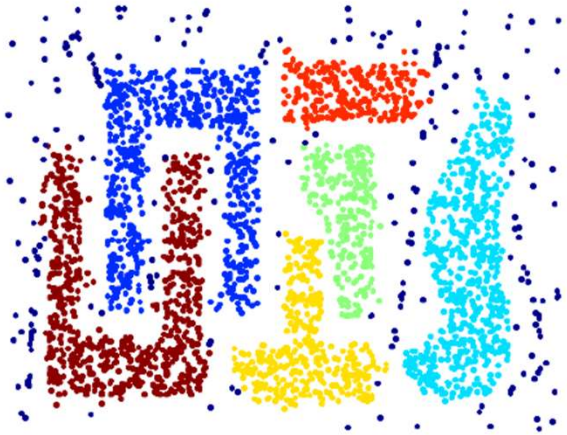
Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



When DBSCAN work well



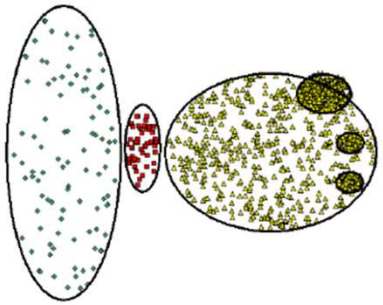
Original Points



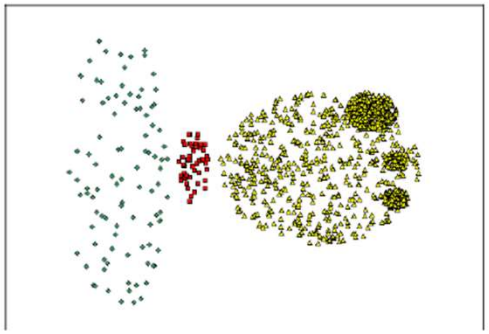
Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

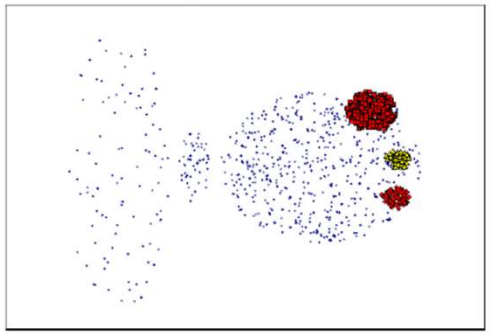
When DBSCAN does not work



Original Points



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

- Cannot handle varying densities
- sensitive to parameters—hard to determine the correct set of parameters

DBSCAN :determining EPS and MinPts

Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance

Noise points have the k^{th} nearest neighbor at farther distance

So, plot sorted distance of every point to its k^{th} nearest neighbor