Chapter 10

Cluster Analysis

Xike Xie

Slides are based on Prof. Ben Kao's work.

Cluster Analysis

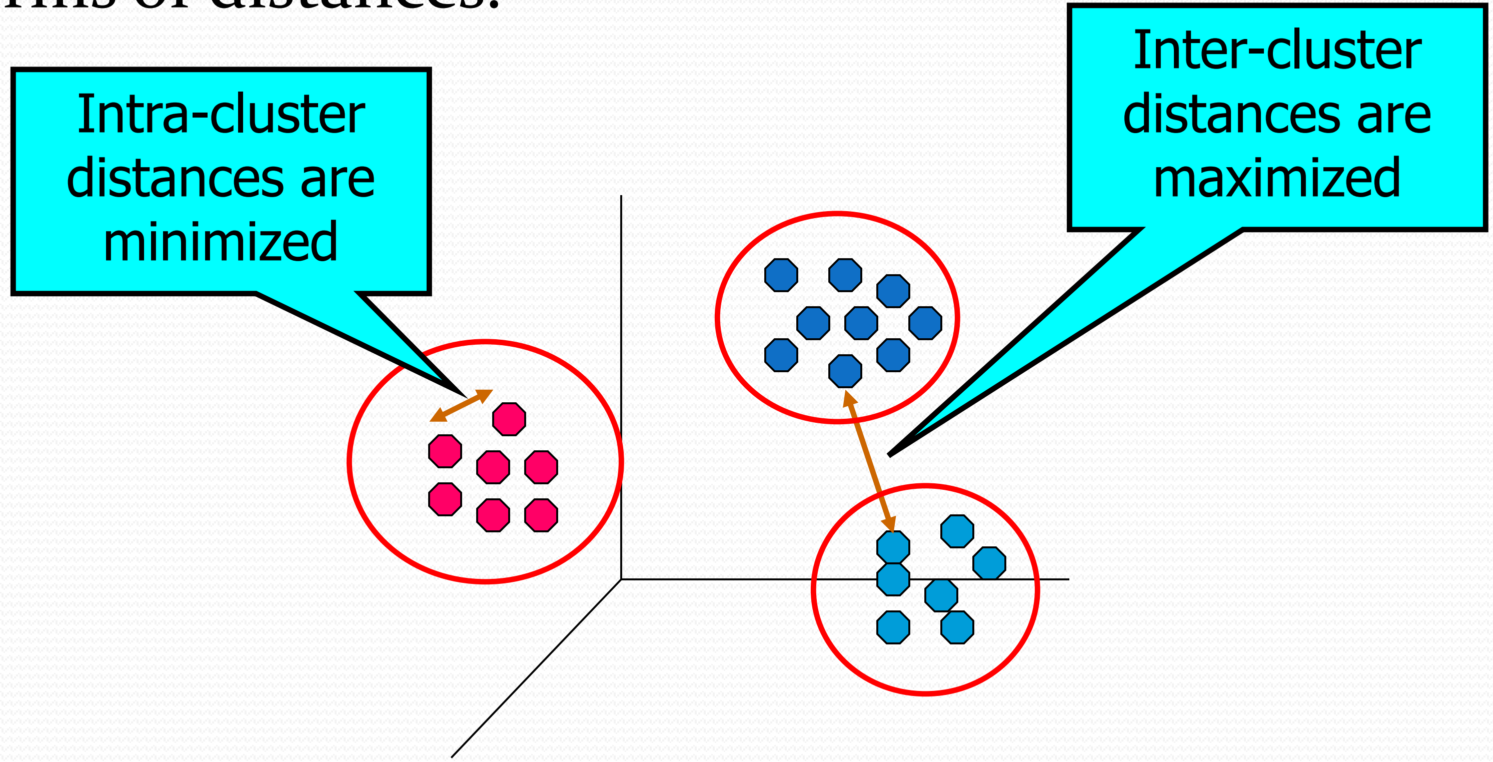
- What is clustering?
- Applications
- Types of clusters
- Partitioning methods
- Hierarchical methods

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group are similar (or related) to each another and different from (or unrelated to) the objects in other groups
- Sometimes called “unsupervised classification” (no pre-defined class labels)

What is Cluster Analysis

- An example where “object similarity” is measured in terms of distances.


Intra-cluster distances are minimized
Inter-cluster distances are maximized

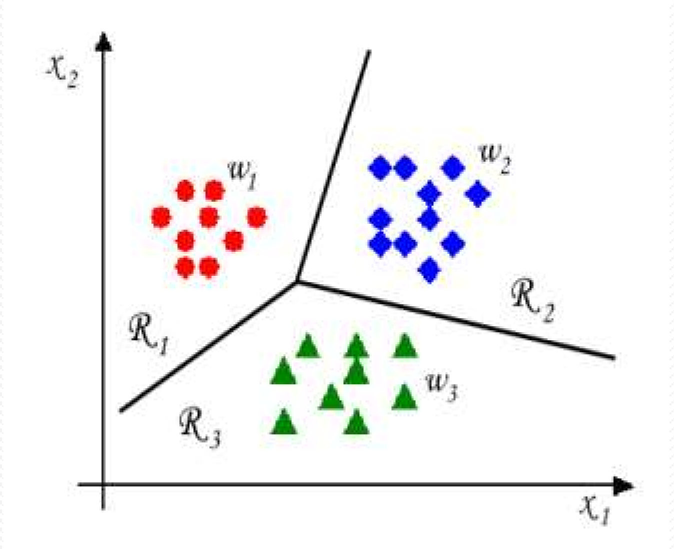
Supervised Learning (Classification)

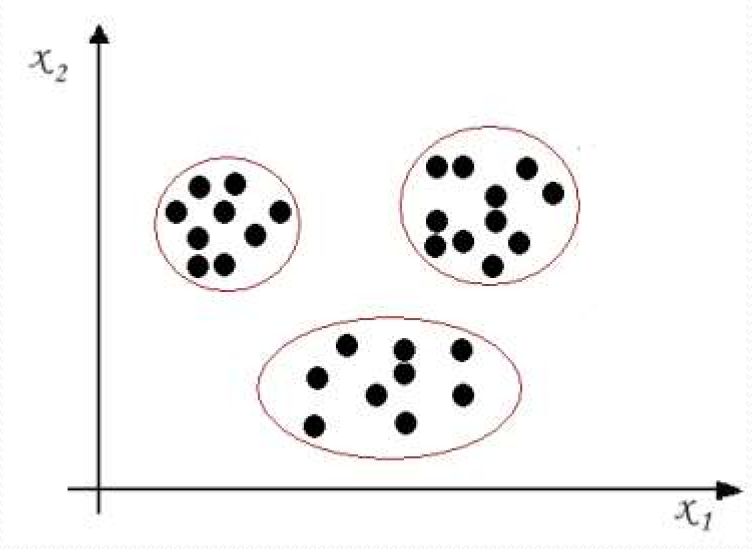
DOG

CAT

Unsupervised Learning (Clustering)

Classification and Clustering





Given training patterns from each class, goal is to construct decision boundaries or to partition the feature space

Given a collection of patterns, the goal is to discover the underlying structure (categories) in the data based on inter-pattern similarities

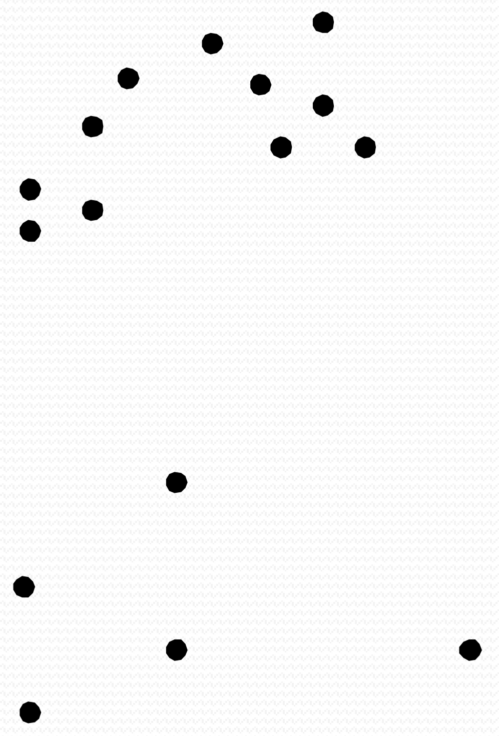
Why clustering?

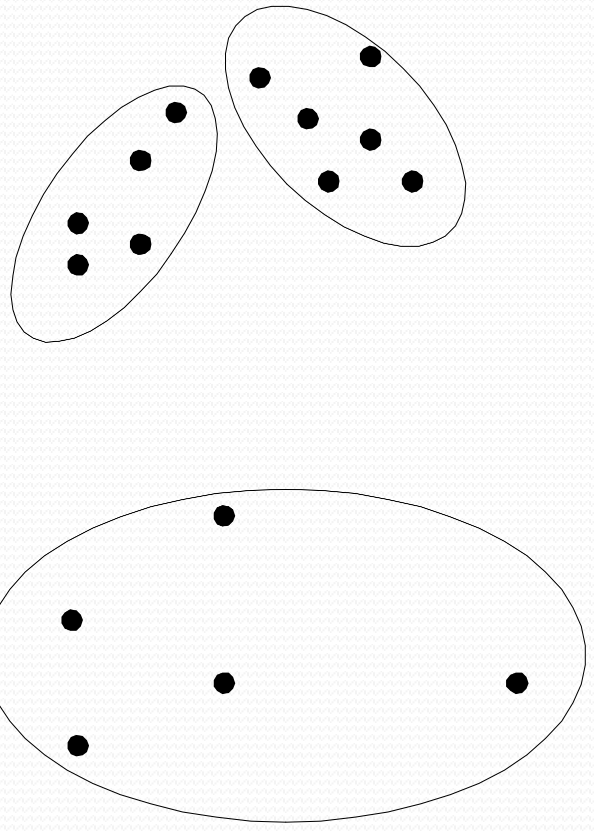
- Summarization
  - Greatly reduces the size of large data set by picking a representative from each cluster and present only those representatives.
- Outlier detection
  - Objects that do not belong to any cluster are outliers.
- Compression (e.g., vector quantization)
  - Each object O is represented by, say, the centroid (C) of its cluster and the deviation of O from C.
- Understanding key features of similar objects.

Types of Clusterings

- A *clustering* is a set of clusters
- *Partitional Clustering*
  - A division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- *Hierarchical clustering*
  - A set of nested clusters organized as a hierarchical tree

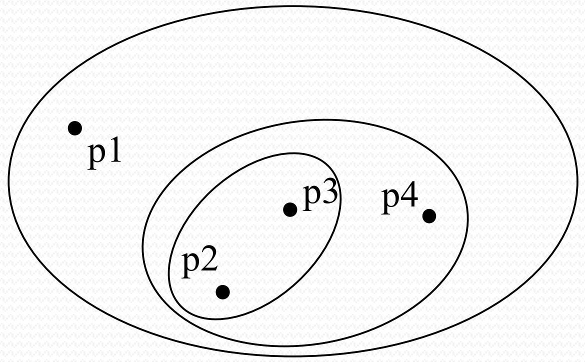
Partitional Clustering


Original Points

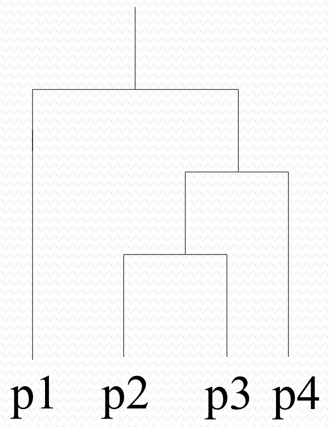

A Partitional Clustering

Hierarchical Clustering

Very loose proximity requirement



Hierarchical Clustering



Dendrogram

Very stringent proximity requirement

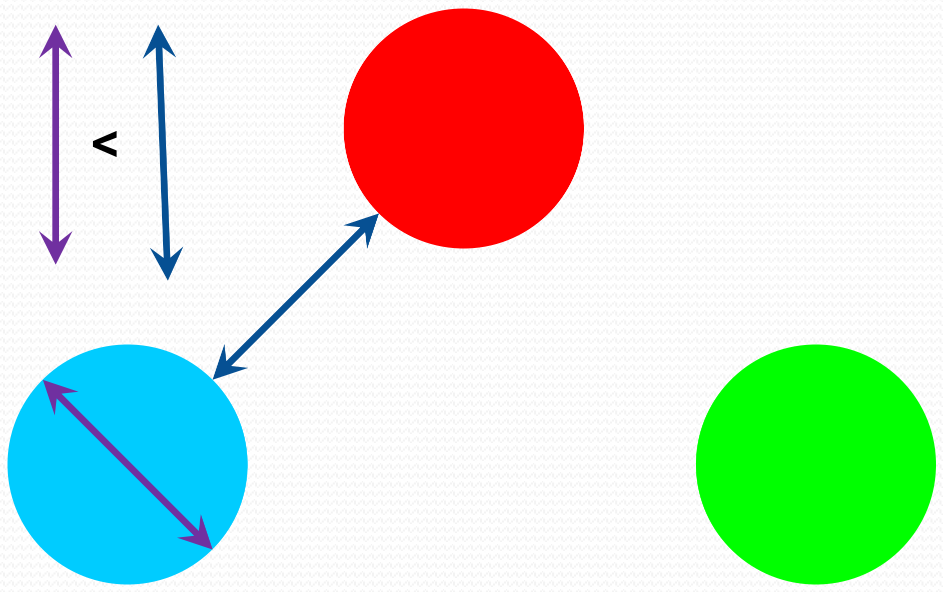
Types of Clusters

- Well-separated clusters
- Center-based clusters
- Contiguity-based clusters
- Density-based clusters

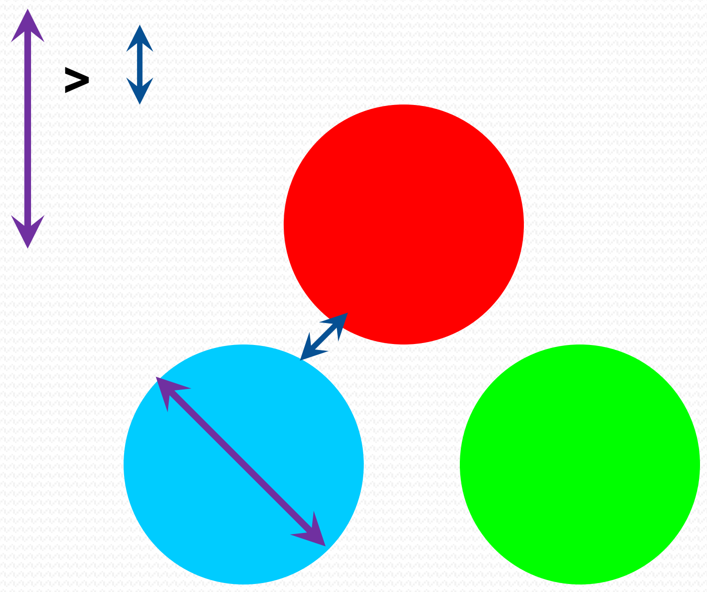
Types of Clusters: Well-Separated

- Well-Separated Clusters:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.

Well separated clusters



3 well-separated clusters

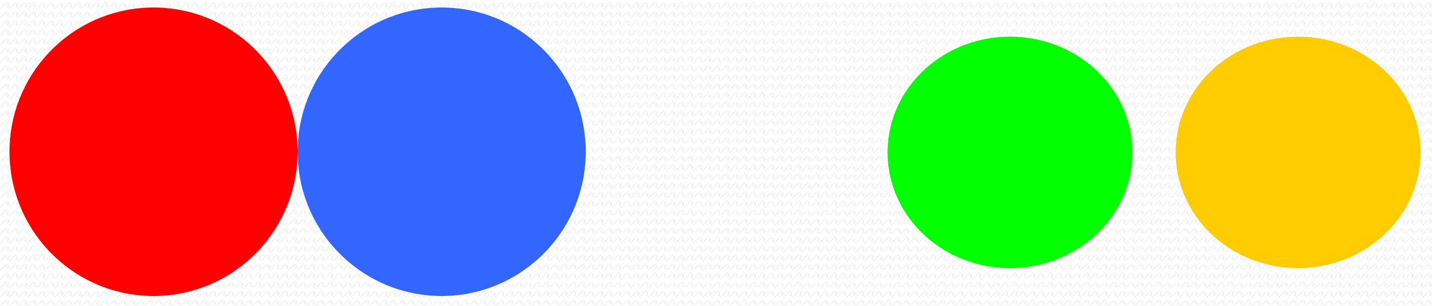


3 not well-separated clusters

Types of Clusters: Center-Based

- Center-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster
  - The *center* of a cluster is often a *centroid*, the average of all the points in the cluster, or a *medoid*, the most "representative" point of a cluster

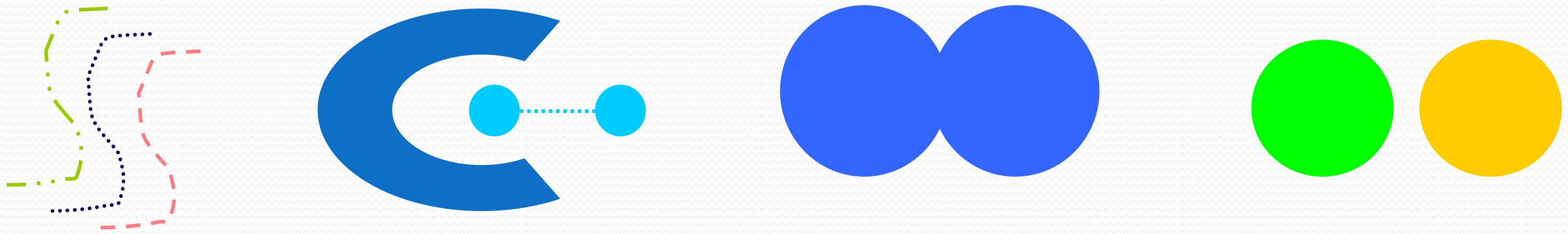
Center-based clusters



4 center-based clusters

Types of Clusters: Contiguity-Based

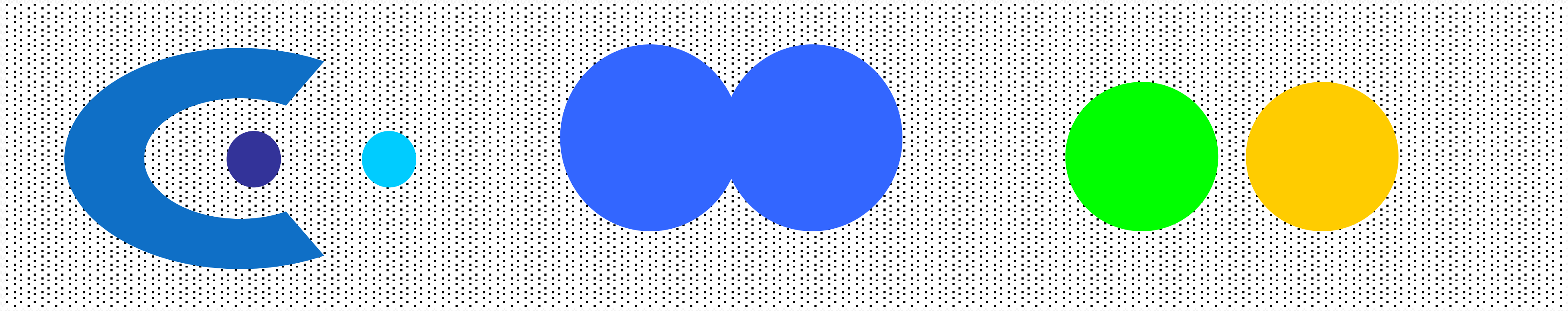
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or a set of other points in the cluster than to any point not in the cluster.



8 contiguous clusters

Types of Clusters: Density-Based

- Density-based
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.



6 density-based clusters

Clustering Algorithms

- K-means and K-medoid
- Hierarchical clustering

Distance-based

- Density-based clustering (DBSCAN)

K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

```
1: Select $K$ points as the initial centroids.
2: repeat
3:     Form $K$ clusters by assigning all points to the closest centroid.
4:     Recompute the centroid of each cluster.
5: until The centroids don't change
```

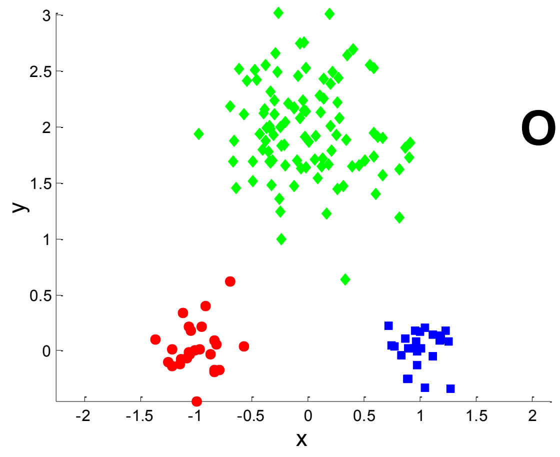
K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced depend on the choice of the initial centroids.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ depends on the type of data: measured by Euclidean distance, cosine similarity, etc.

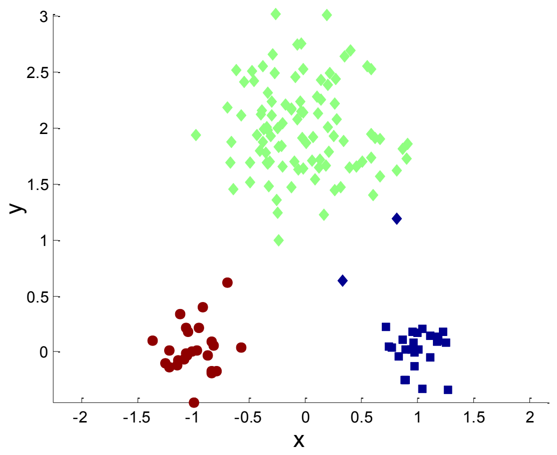
K-means Clustering – Details

- K-means will converge for common similarity measures such as Euclidean distance.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes

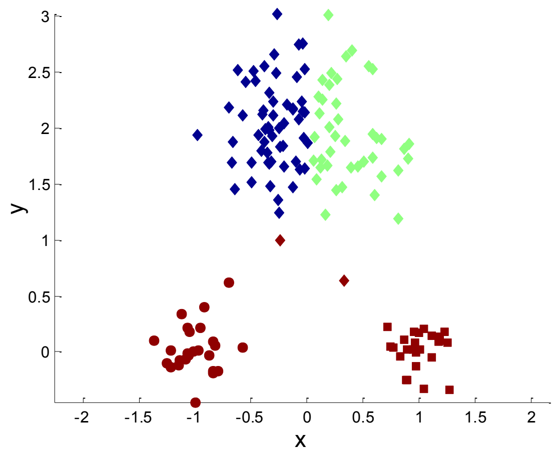
Two different K-means Clusterings (K=3)



Original Points

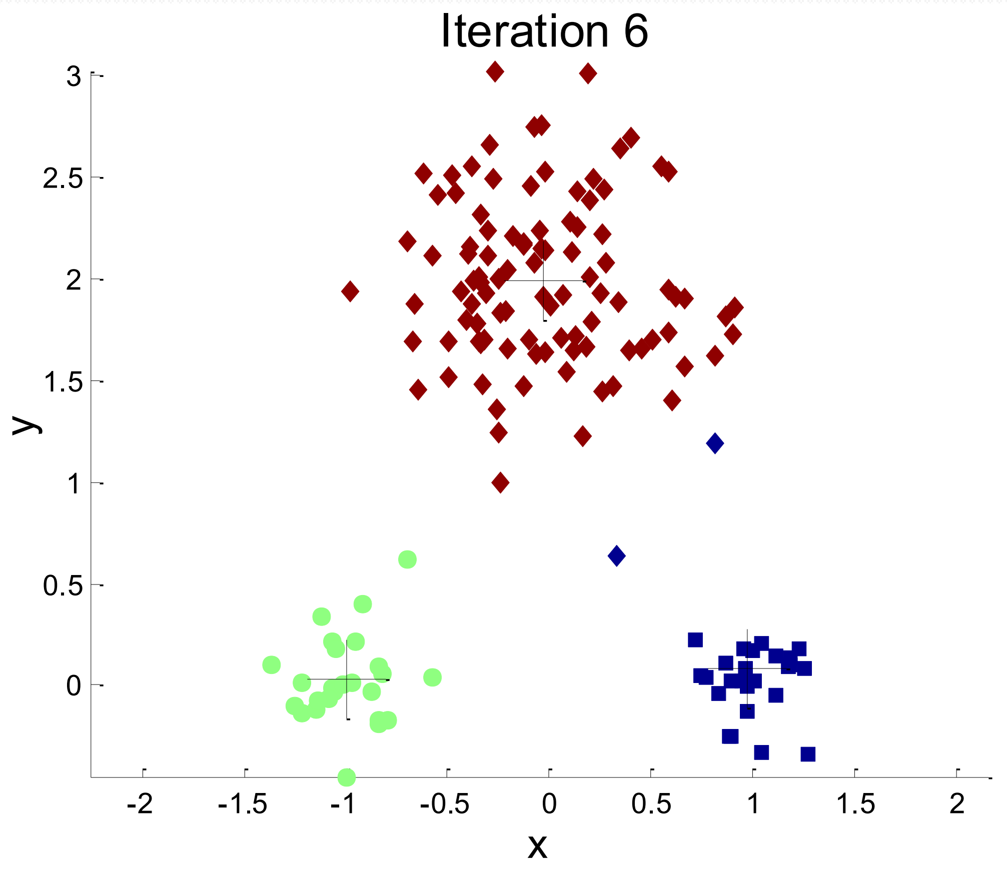


Optimal Clustering

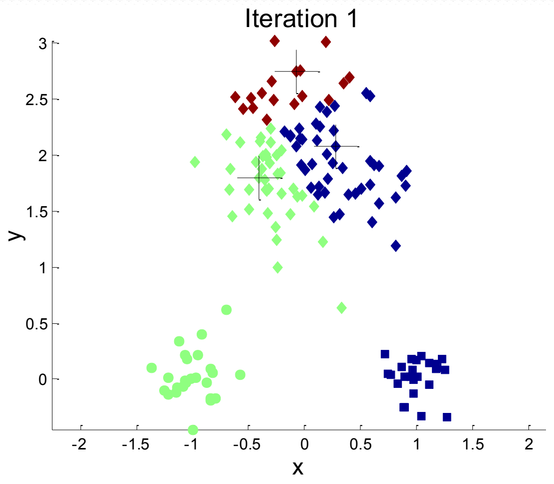


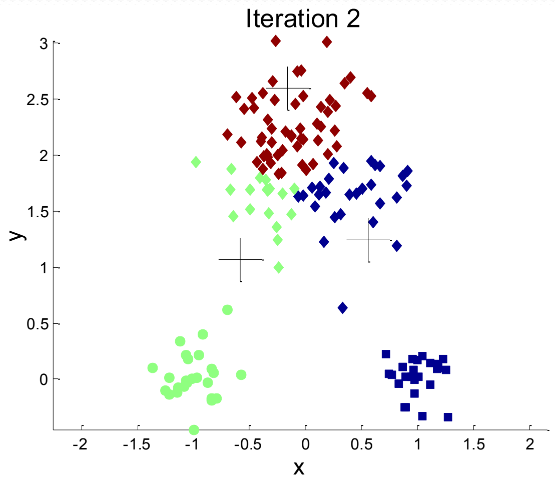
Sub-optimal Clustering

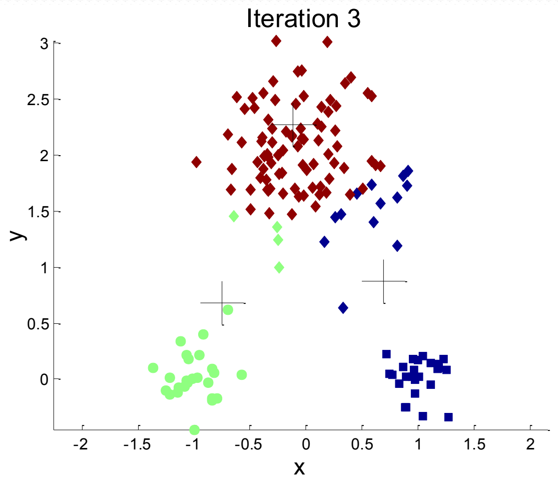
Importance of Choosing Initial Centroids

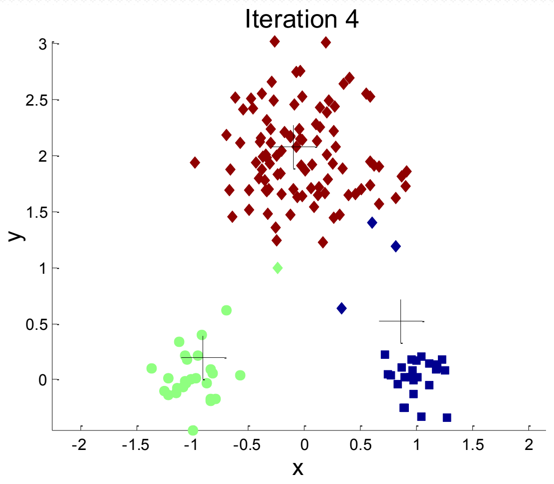


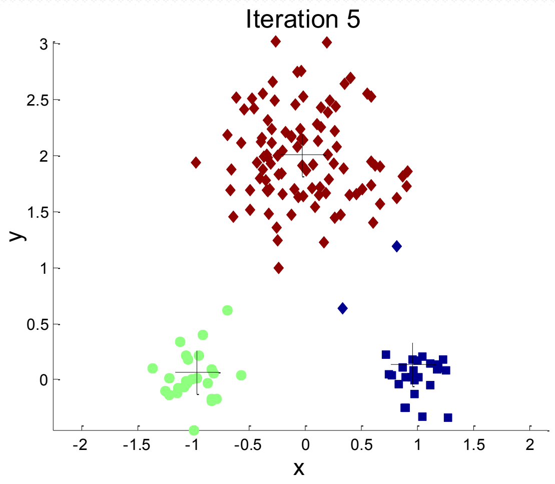
Importance of Choosing Initial Centroids

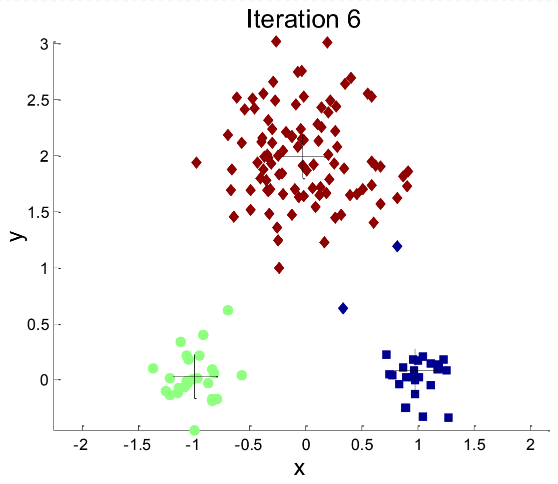




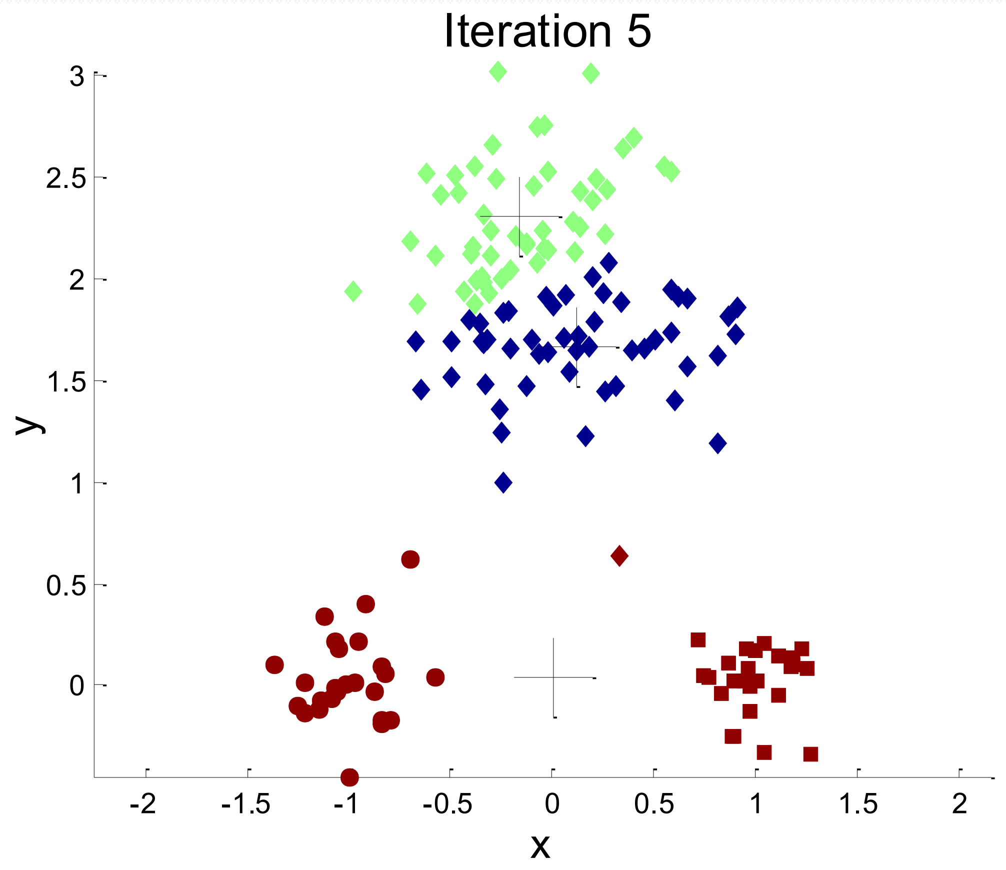




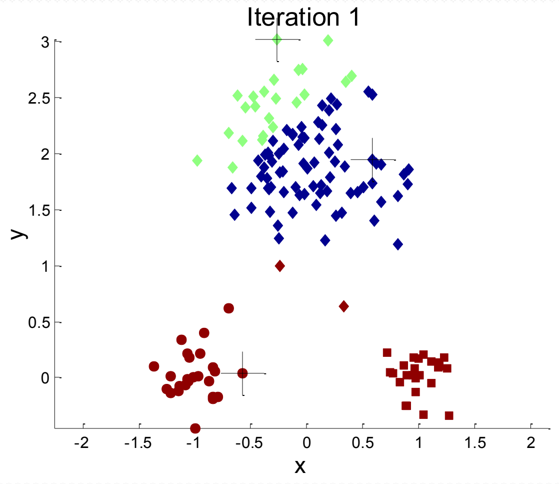


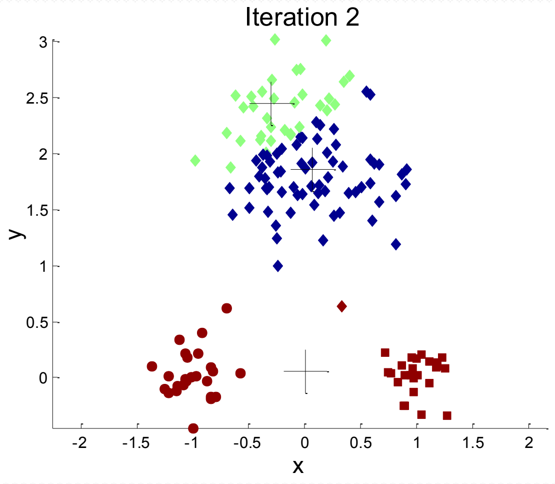


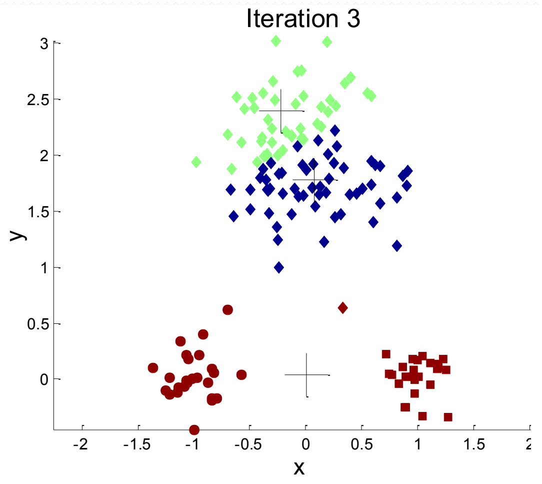
Importance of Choosing Initial Centroids …

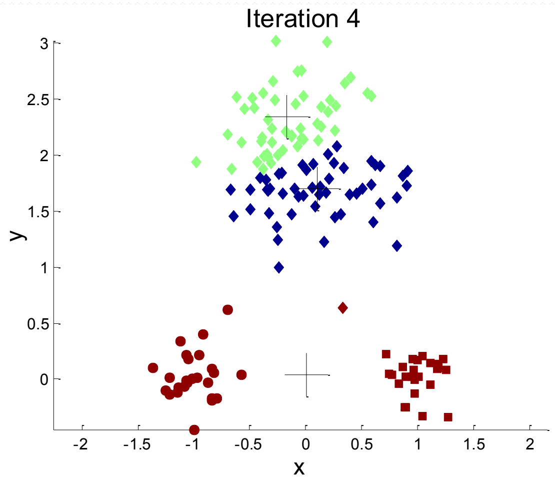


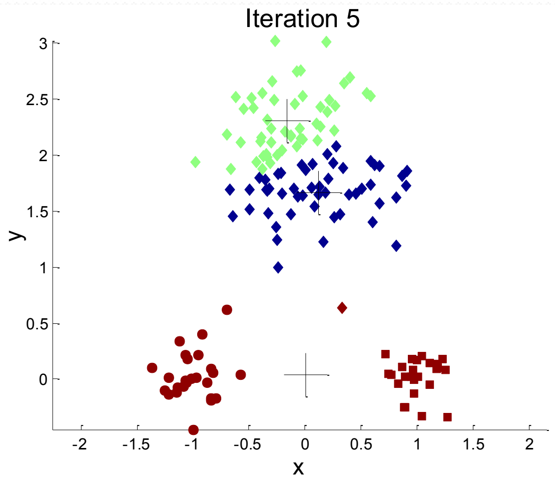
Importance of Choosing Initial Centroids …











Challenges

- Solutions to Randomness of Initial Centroids Selection
  - Multiple runs for data (or data sample)
  - Select most widely separated objects
  - Bisecting K-means

- Choosing parameter K
  - Elbow method
  - Gap statistics

Evaluating K-means Clusters

- Most common measure is *Sum of Squared Error* (SSE)
  - For each point, the error is the distance to its cluster representative
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

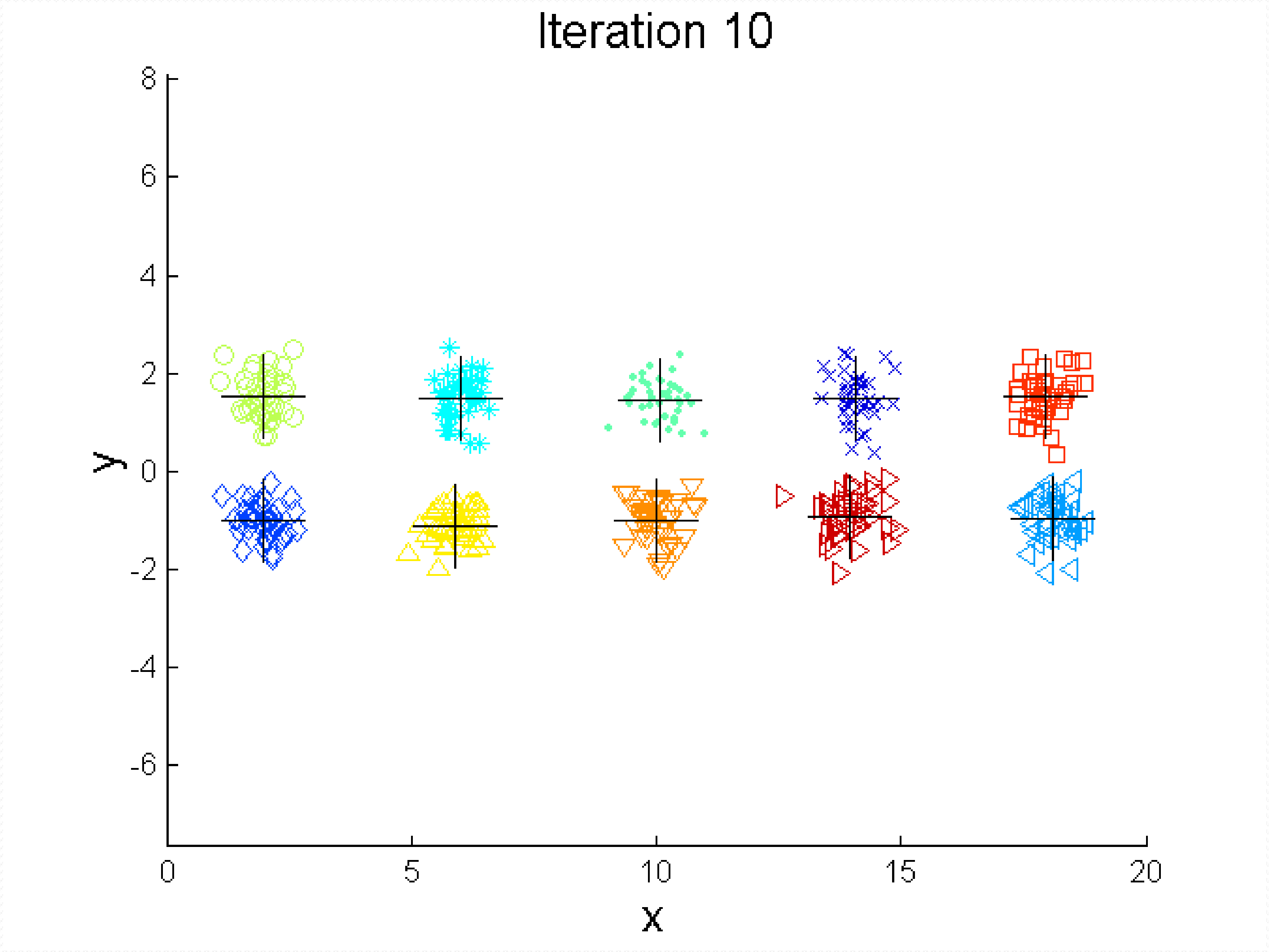
  - *x* is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - Given a clustering, if each m_i is set as the center (mean) of its cluster, then SSE is minimized.
  - Given two cluster results, we can choose the one with the smallest error

Bisecting K-means

- Bisecting K-means algorithm
  - Variant of K-means that can produce a partitional or a hierarchical clustering
  - Each time pick a cluster to split
    - based on size, SSE, or both

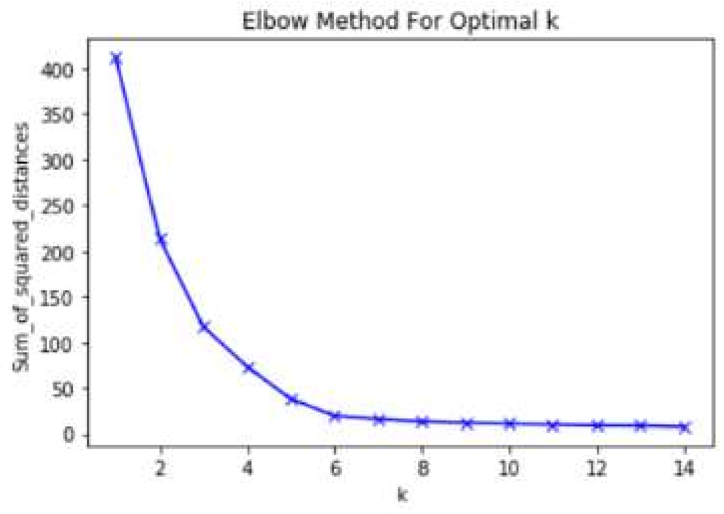
-
```
1: Initialize the list of clusters to contain one cluster with all points.
2: repeat
3:     Remove a cluster from the list of clusters.
4:     {Perform several "trial" bisections of the chosen cluster.}
5:     for i = 1 to number of trials do
6:             Bisect the selected cluster using basic K-means.
7:     Select the two clusters from the bisection with the lowest total SSE.
8:     Add these two clusters to the list of clusters.
9: until the list of clusters contains K clusters.
```

Bisecting K-means Example



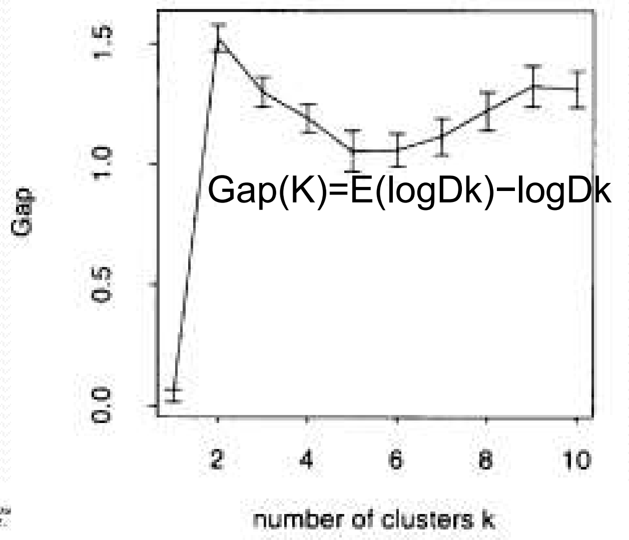
Choosing Parameter K

Elbow Method

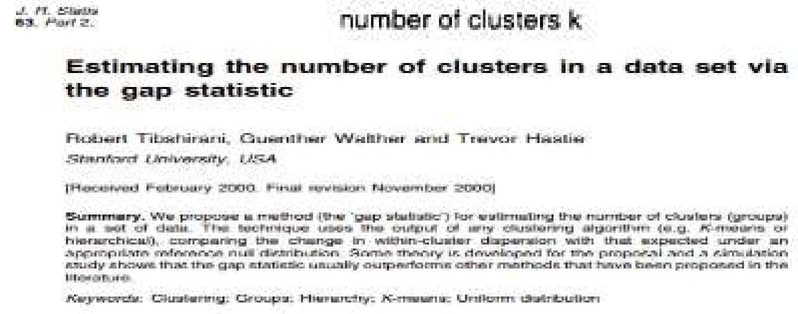


$$D_K = \sum_{i=1}^{K} \sum_{X \in C_i} \|X - M_i\|$$

Gap Statistics



Gap(K)=E(logDk)−logDk



J. R. Statis
63, Part 2,

Estimating the number of clusters in a data set via the gap statistic

Robert Tibshirani, Guenther Walther and Trevor Hastie
Stanford University, USA

[Received February 2000. Final revision November 2000]

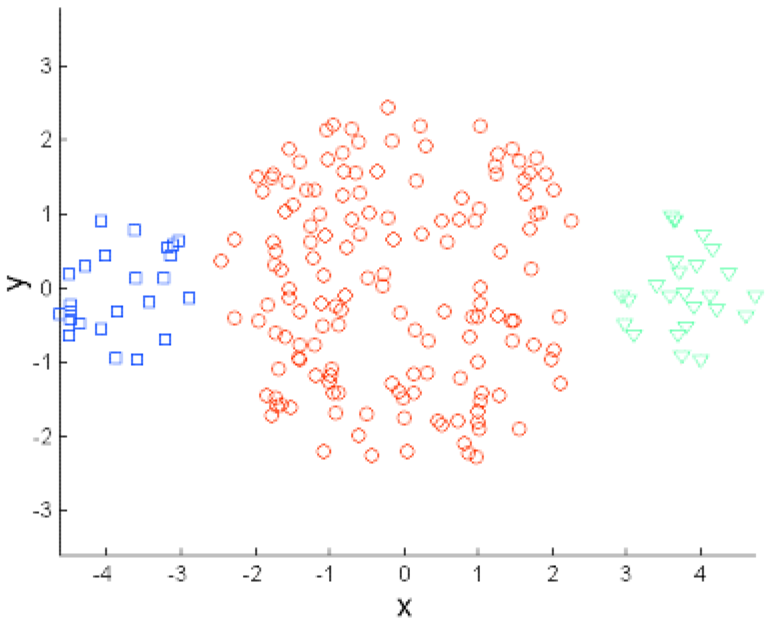
Summary. We propose a method (the 'gap statistic') for estimating the number of clusters (groups) in a set of data. The technique uses the output of any clustering algorithm (e.g. *K*-means or hierarchical), comparing the change in within-cluster dispersion with that expected under an appropriate reference null distribution. Some theory is developed for the proposal and a simulation study shows that the gap statistic usually outperforms other methods that have been proposed in the literature.

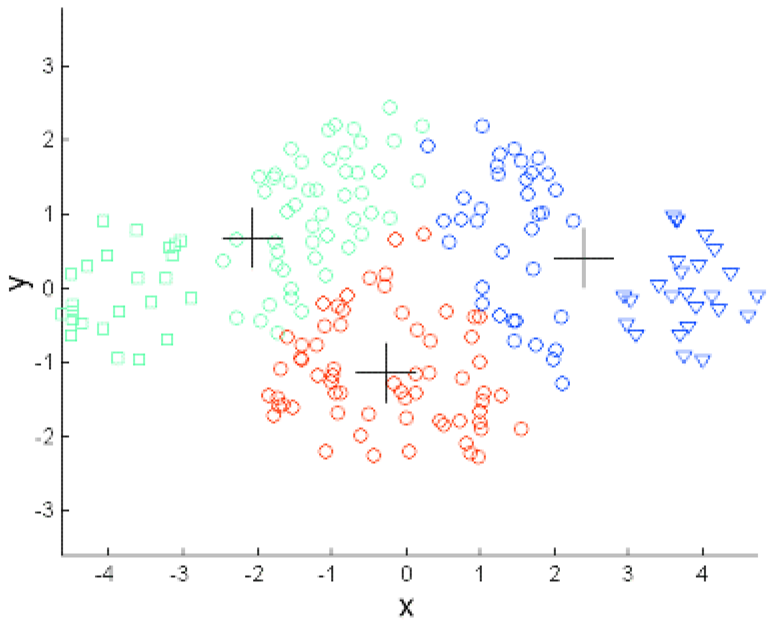
Keywords: Clustering; Groups; Hierarchy; *K*-means; Uniform distribution

Limitations of K-means

- K-means has limits when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has problems when the data contains outliers.

Limitations of K-means: Differing Sizes

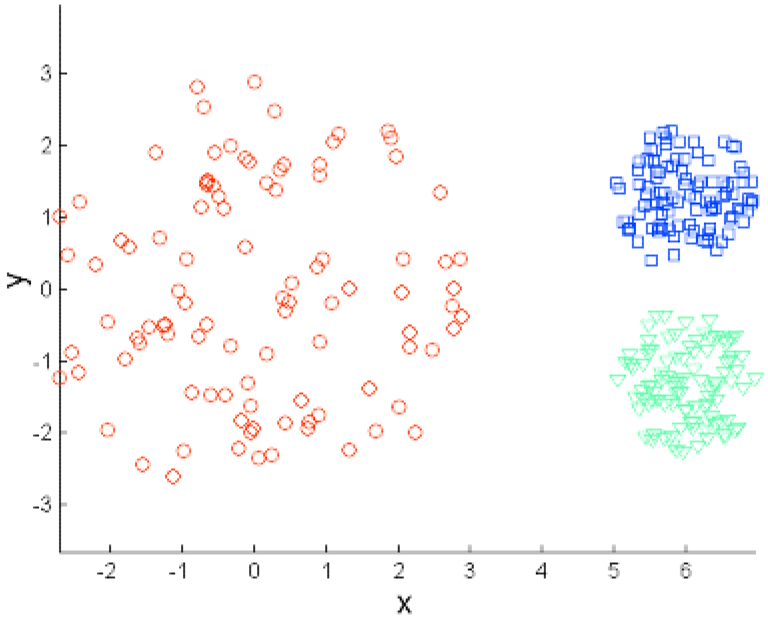




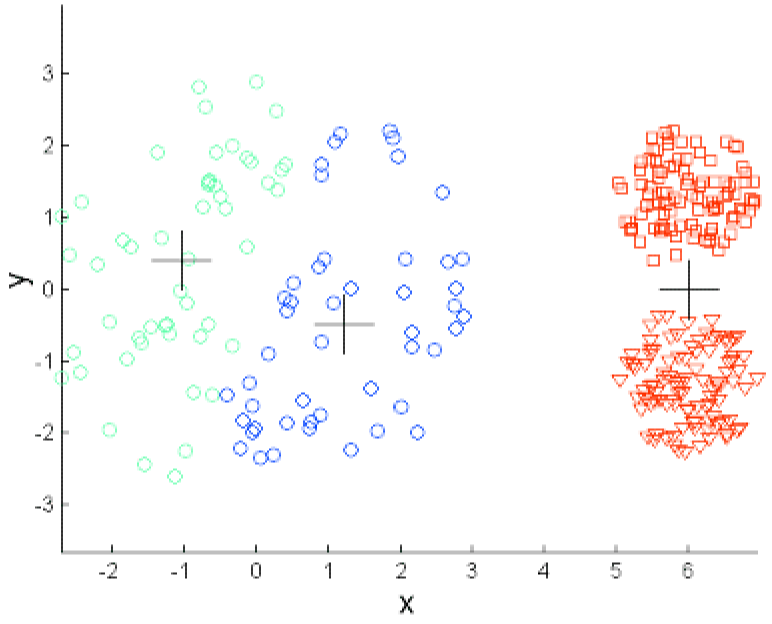
Original Points

K-means (3 Clusters)

Limitations of K-means: Differing Density

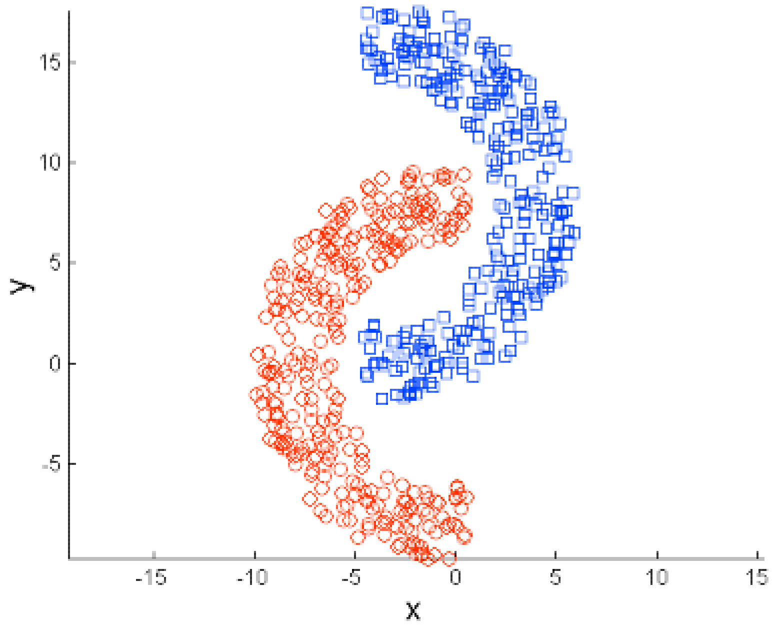


Original Points

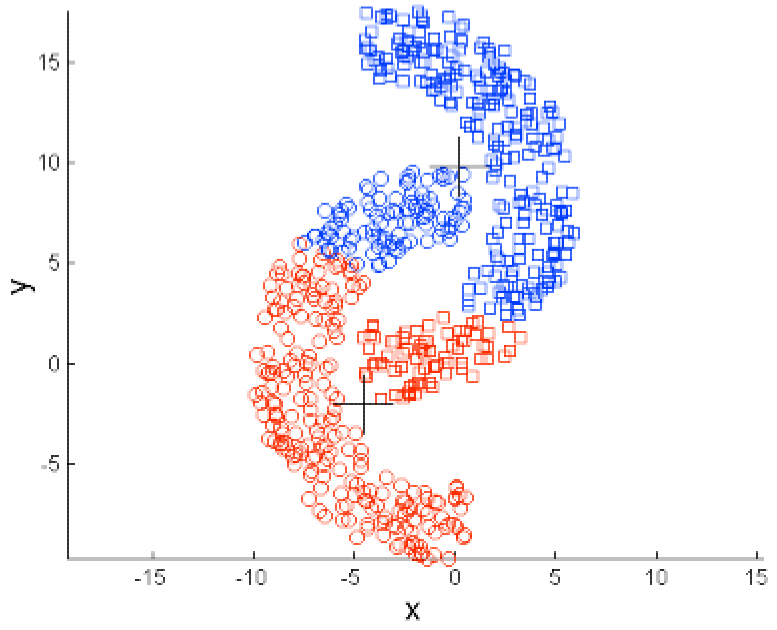


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes



Original Points



K-means (2 Clusters)

k-medoids

- disadvantages of K-means:
  - **Mean**: averages might not make much sense for categorical/binary attributes
  - **Mean:** averages could be distorted by noise
- A medoid in a cluster *C* is a data record *x* in *C* such that the total distance from *x* to all other objects in *C* is the minimum

k-medoids

- example:
  - $x_1 = [0.3, 0.6]$, $x_2 = [0.1, 0.1]$, $x_3 = [0.5, 0.3]$, and $x_4 = [0.4, 0.1]$
  - $d(x_1,x_2) + d(x_1,x_3) + d(x_1,x_4) = 0.539 + 0.361 + 0.510 = 1.410$
  - $d(x_2,x_1) + d(x_2,x_3) + d(x_2,x_4) = 0.539 + 0.447 + 0.300 = 1.286$
  - $d(x_3,x_1) + d(x_3,x_2) + d(x_3,x_4) = 0.361 + 0.447 + 0.224 = 1.032$
  - $d(x_4,x_1) + d(x_4,x_2) + d(x_4,x_3) = 0.510 + 0.300 + 0.224 = 1.034$
  - medoid = x_3

k-medoids

- The K-medoids method is **exactly the same as** the K-means method **except that** the "representative" used for a cluster is its medoid (instead of its mean)
- The K-medoids method, however, is **more computationally demanding** than the k-means method

k-medoids

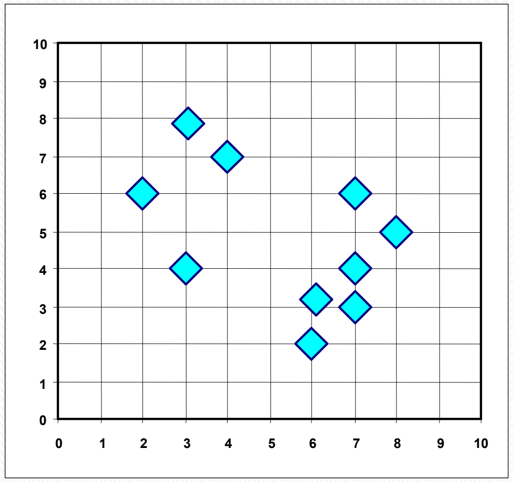
- Alternative strategy:
  - randomly pick k points as medoids, $m_1, \ldots, m_k$
  - partition all other points into clusters represented by the k medoids
  - calculate the error:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} d(p, m_i)$$

  - pick a point r and replace a medoid m_j by r
  - compute error E again and see if E is reduce
    - if not, undo the replacement

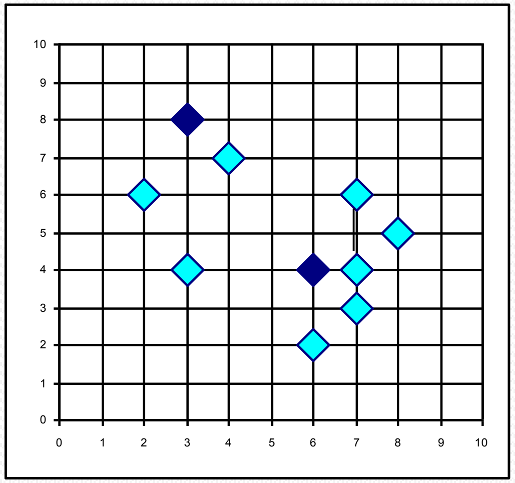
PAM: A Typical K-Medoids Algorithm

Total Cost = 20

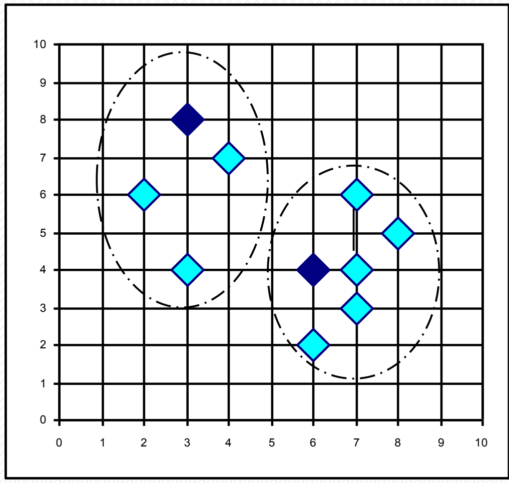


K=2

Arbitrary choose k object as initial medoids

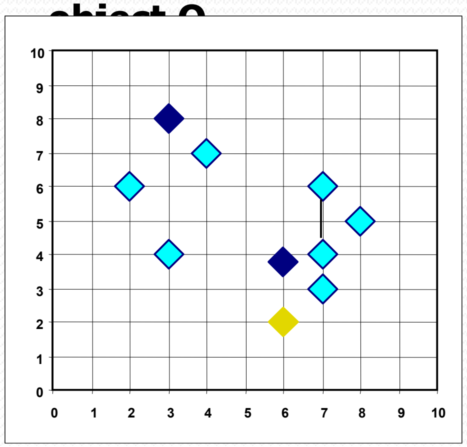


Assign each remaining object to nearest medoids

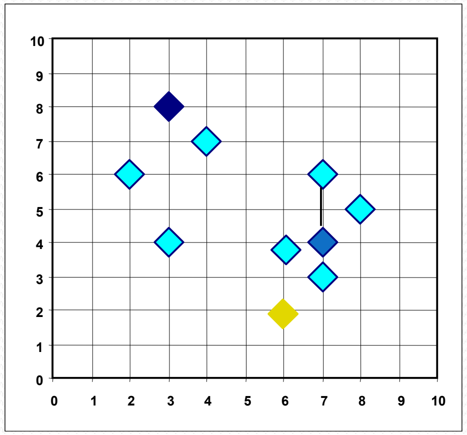


Randomly select a nonmedoid object,O

Total Cost = 26



Compute total cost of swapping



Swapping O and O_{ramdom}

If quality is improved.

Do loop

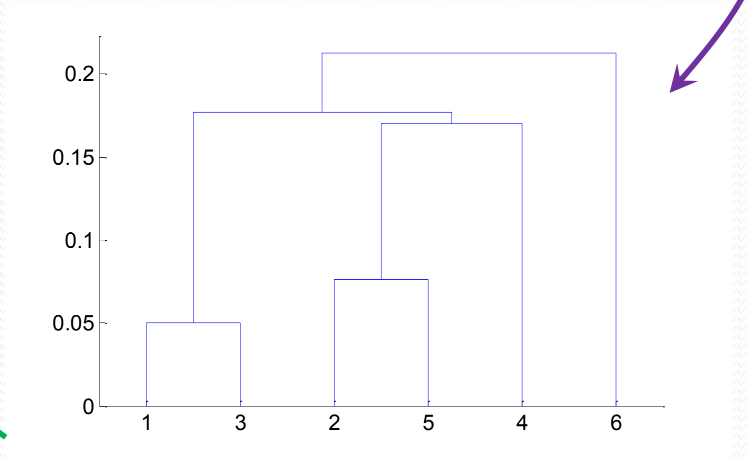
Until no change

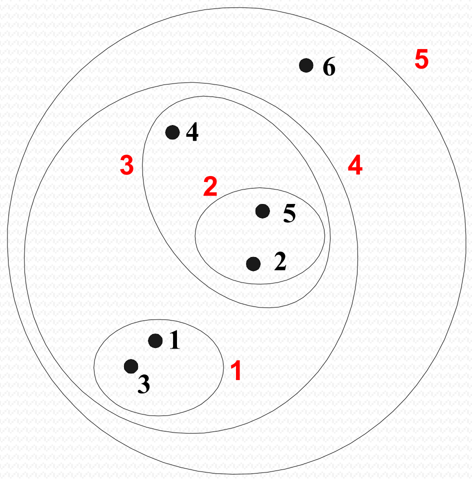
The K-Medoid Clustering Method

- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
  - *PAM* (Partitioning Around Medoids, Kaufmann & Rousseeuw 1987)
    - Starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
    - *PAM* works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity) **O(k(n-k))**
- Efficiency improvement on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a *dendrogram*
  - A tree like diagram that records the sequences of merges or splits

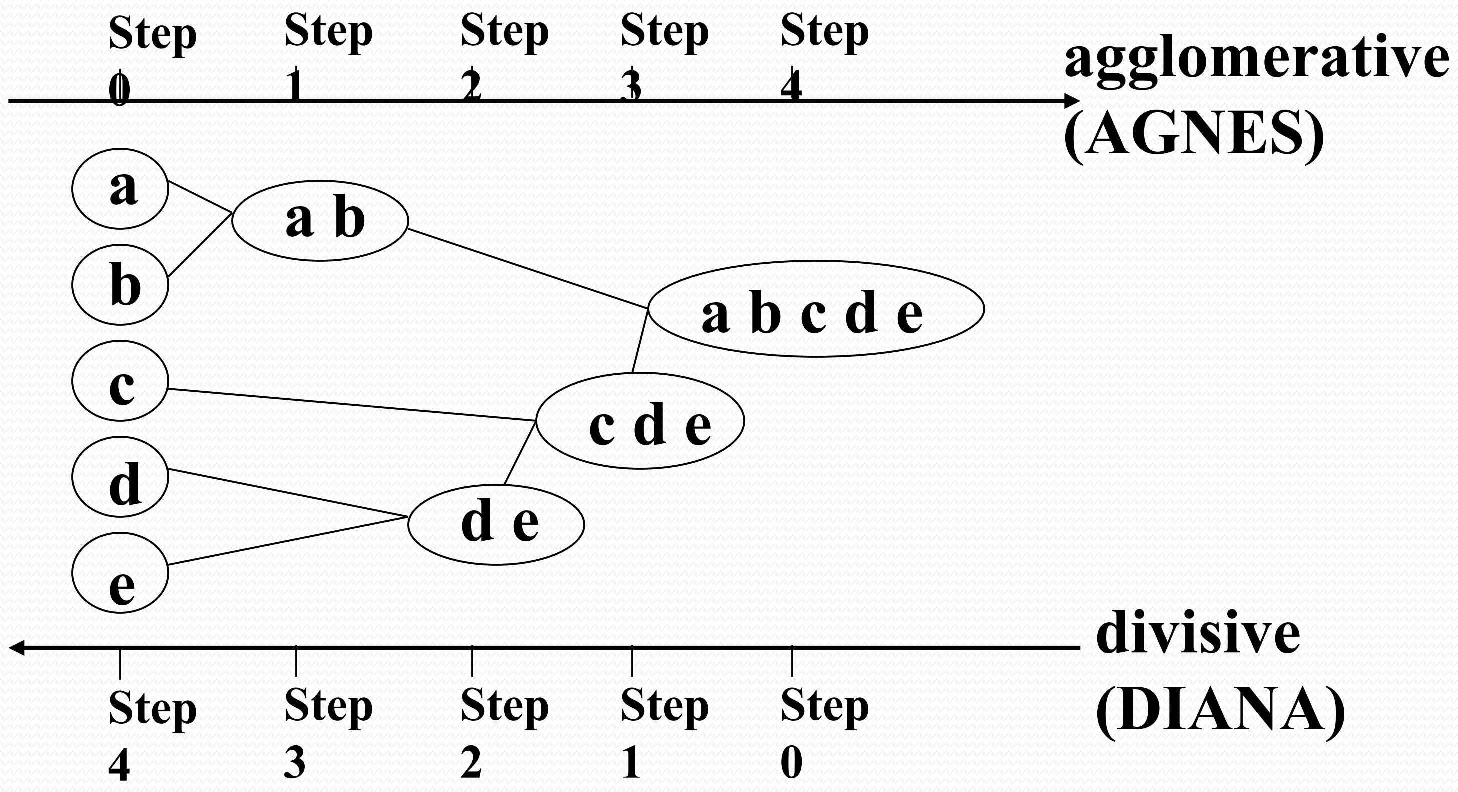




Proximity of each merge

Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

Hierarchical Clustering

- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

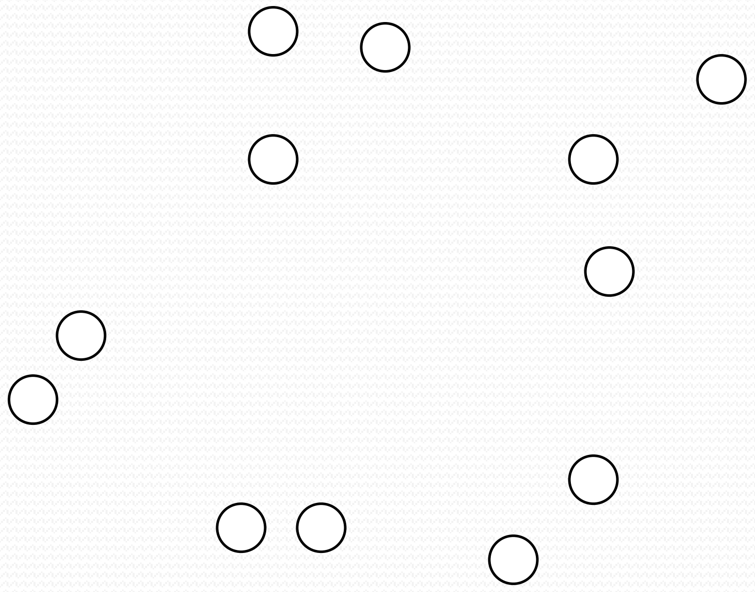
Agglomerative Clustering

- Most popular hierarchical clustering technique
- Basic algorithm is straightforward

```
Compute the proximity matrix
Let each data point be a cluster
Repeat
    Merge the two closest clusters
    Update the proximity matrix
Until only a single cluster remains
```

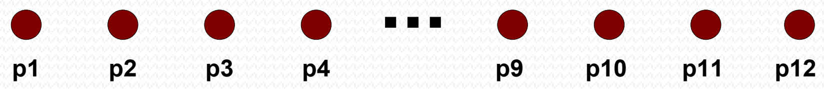
- Key operation is the computation of the proximity of two clusters

- Start with clusters of individual points and a proximity matrix

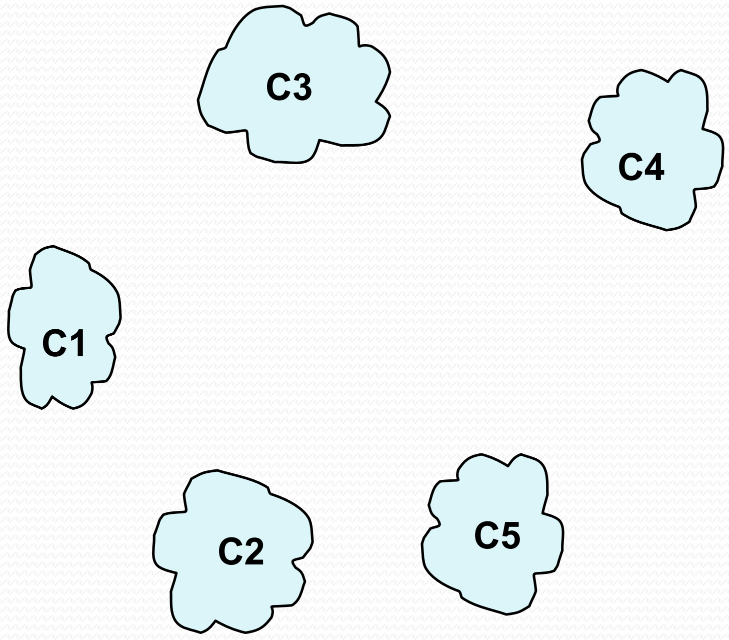


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

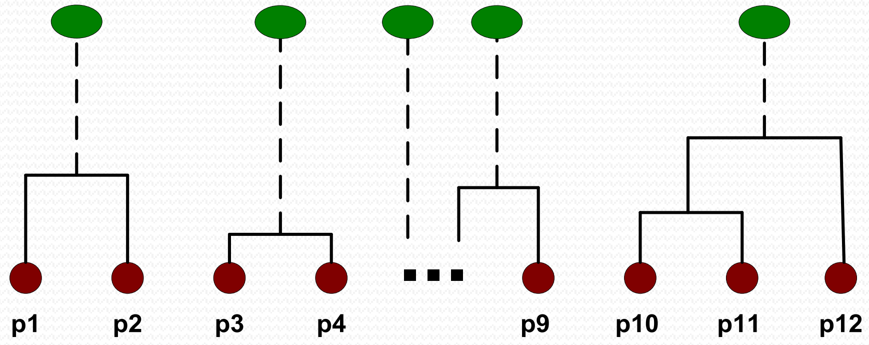


- After some merging steps, we have some clusters

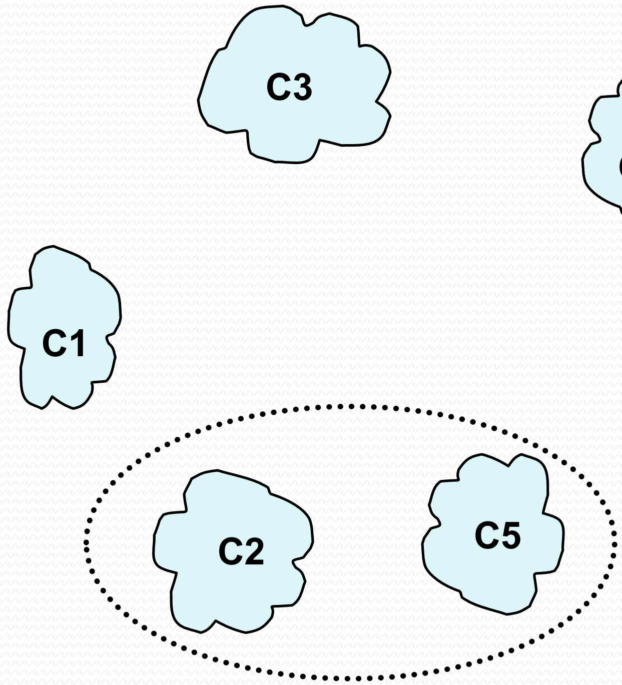


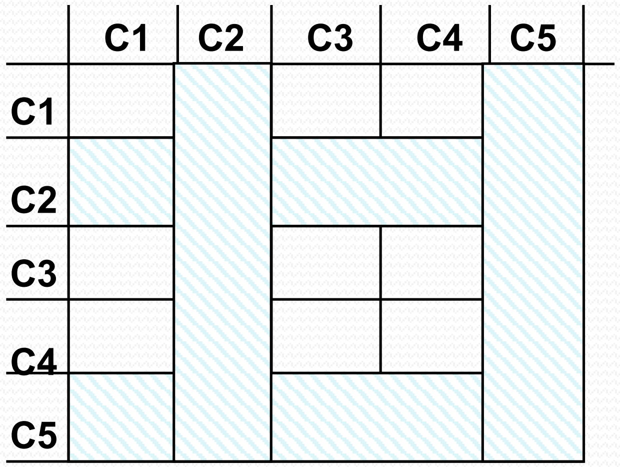
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

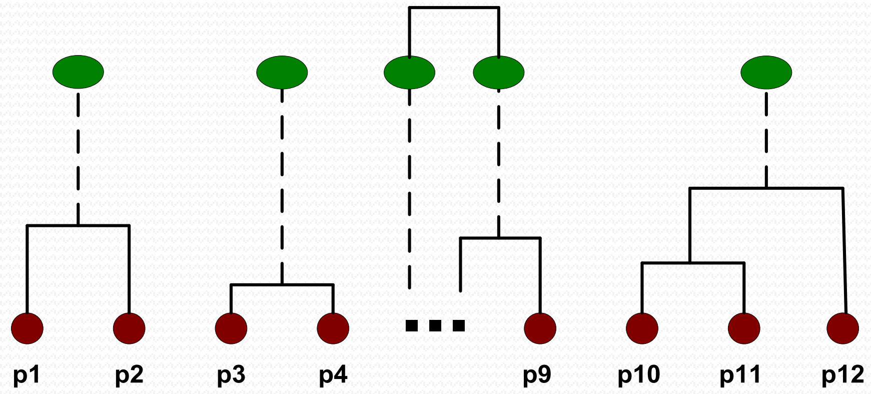


- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.

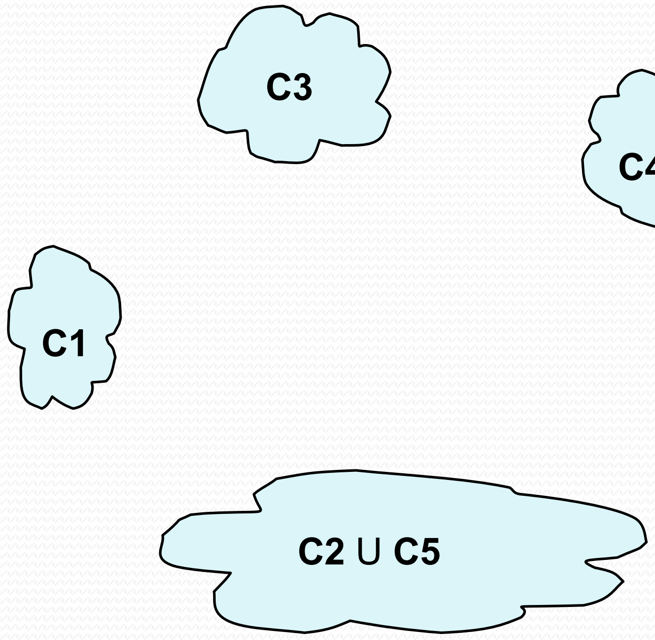




Proximity Matrix

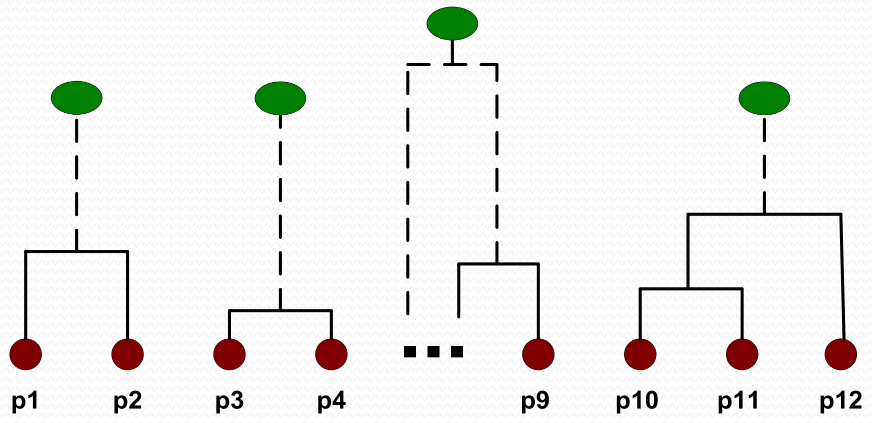


- The question is "How do we update the proximity matrix?"

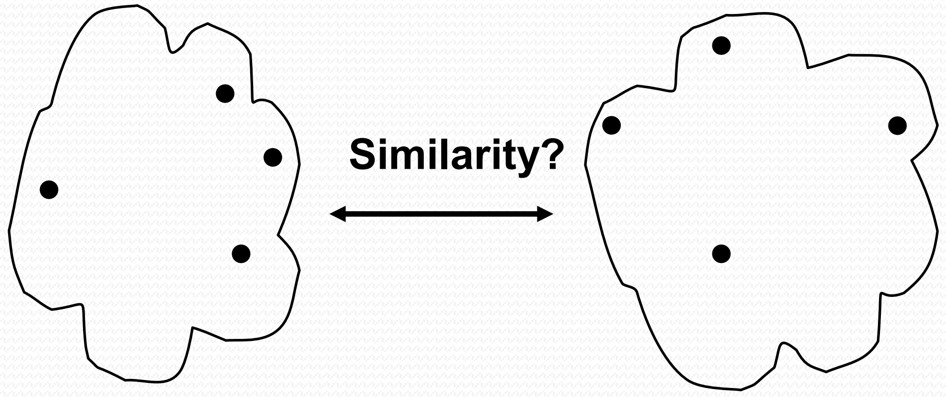


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

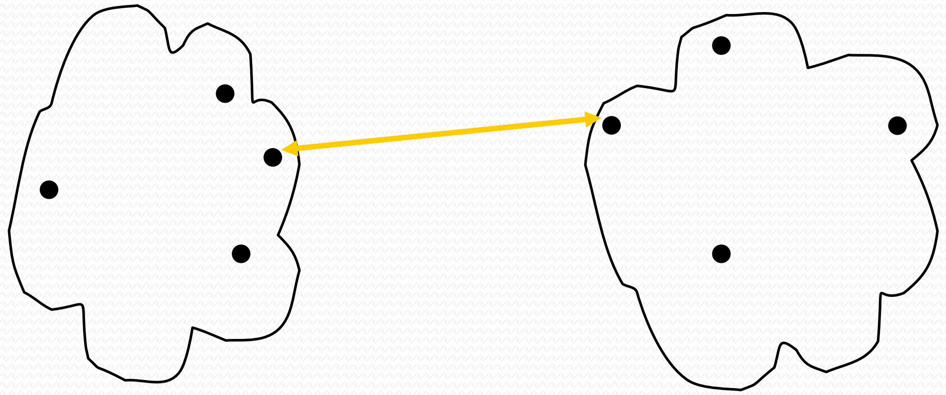


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids

How to Define Inter-Cluster Similarity

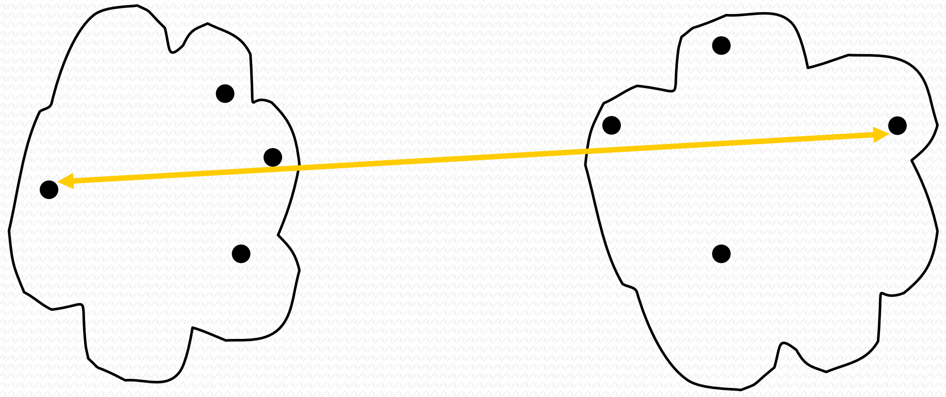


- MIN
- MAX
- Group Average
- Distance Between Centroids

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

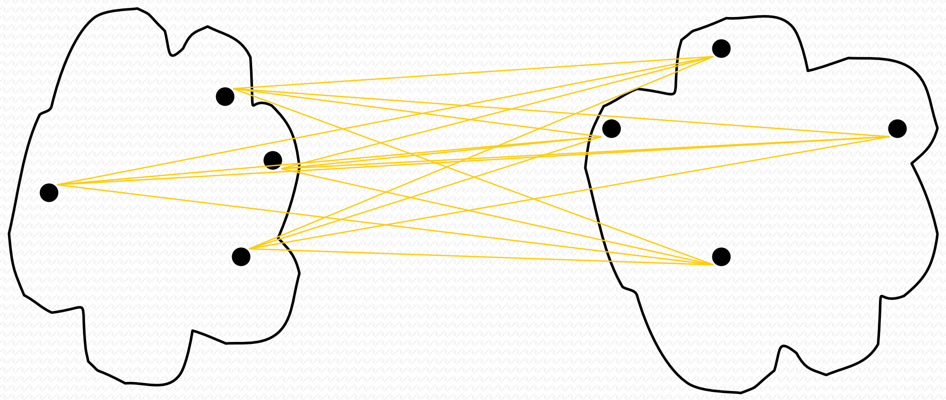


- MIN
- MAX
- Group Average
- Distance Between Centroids

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

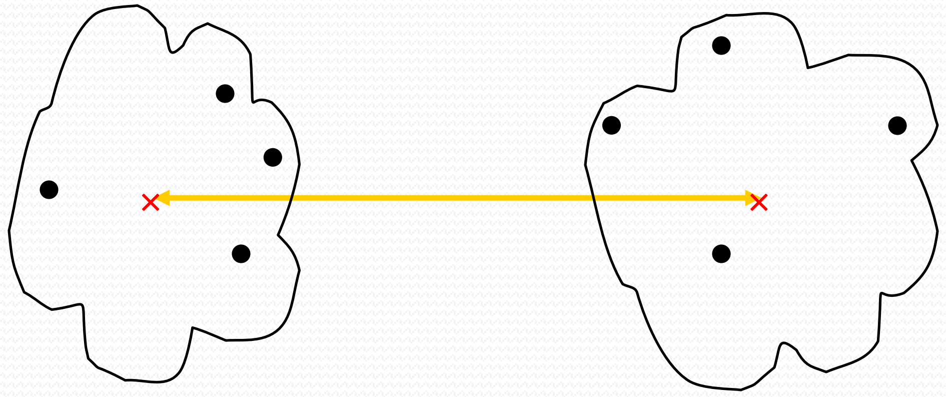


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

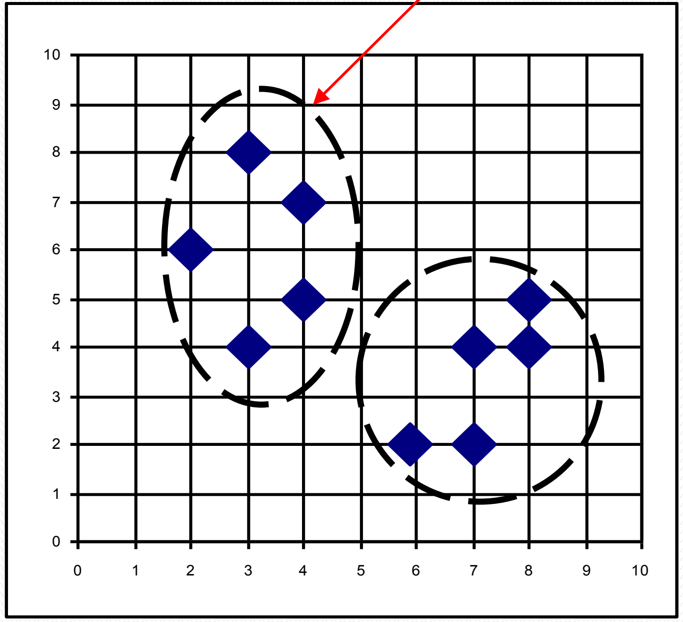
- MIN
- MAX
- Group Average
- Distance Between Centroids

Birch: Balanced Iterative Reducing and Clustering using Hierarchies

- Weakness of agglomerative clustering methods
  - time complexity of at least O(n^2)
  - Can never undo what was done previously
- Birch: Balanced Iterative Reducing and Clustering using Hierarchies
  - Incrementally construct a *CF* (*Clustering Feature*) tree
  - Get a good clustering with a single scan O(n)

Example of Clustering Feature Vector

- **Clustering Feature: CF=(n, LS, SS)**
- *n*: Number of data points $SS : \sum_{i=1}^{N} \vec{X}_i^2$ $LS : \sum_{i=1}^{N} \vec{X}_i$



CF = (5, (16, 30), (54, 190))

n	LS	SS

(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

Effect of CF on Single-cluster

- Given a cluster with N objects
  - Centroid

$$\overrightarrow{X_0} = \frac{\sum_{i=1}^{N} \overrightarrow{X_i}}{N}$$

LS/N

  - Radius

$$R = \left(\frac{\sum_{i=1}^{N} (\overrightarrow{X_i} - \overrightarrow{X_0})^2}{N}\right)^{1/2}$$

  - Diameter

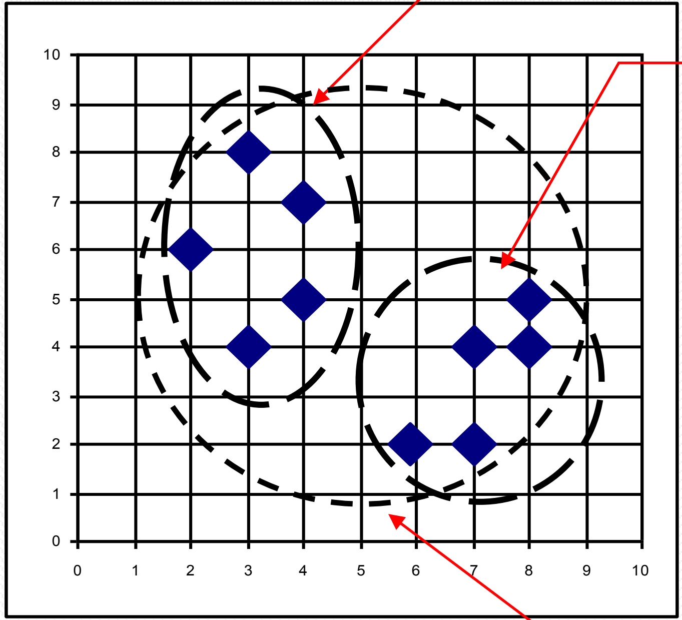
[2NSS-2LS*LS] $^{1/2}$

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\overrightarrow{X_i} - \overrightarrow{X_j})^2}{N(N-1)}\right)^{1/2}$$

CF Additive Theorem

$$CF = (\vec{N}, LS, SS)$$

- Suppose cluster C_1 has $CF_1=(N_1, LS_1, SS_1)$, cluster C_2 has $CF_2 =(N_2,LS_2,SS_2)$, if we merge C_1 with C_2, the CF for the merged cluster C is: $CF = (CF_1 + CF_2) = (N_1+ N_2, LS_1 + LS_2, SS_1+ SS_2)$



CF_1 = (5, (16, 30), (54, 190))

CF_2 = (5, (36, 17), (262, 61))

(3,4)	(6,2)
(2,6)	(7,2)
(4,5)	(7,4)
(4,7)	(8,4)
(3,8)	(8,5)

CF = (10, (52, 47), (316, 251))

Clustering Feature Tree (CFT)

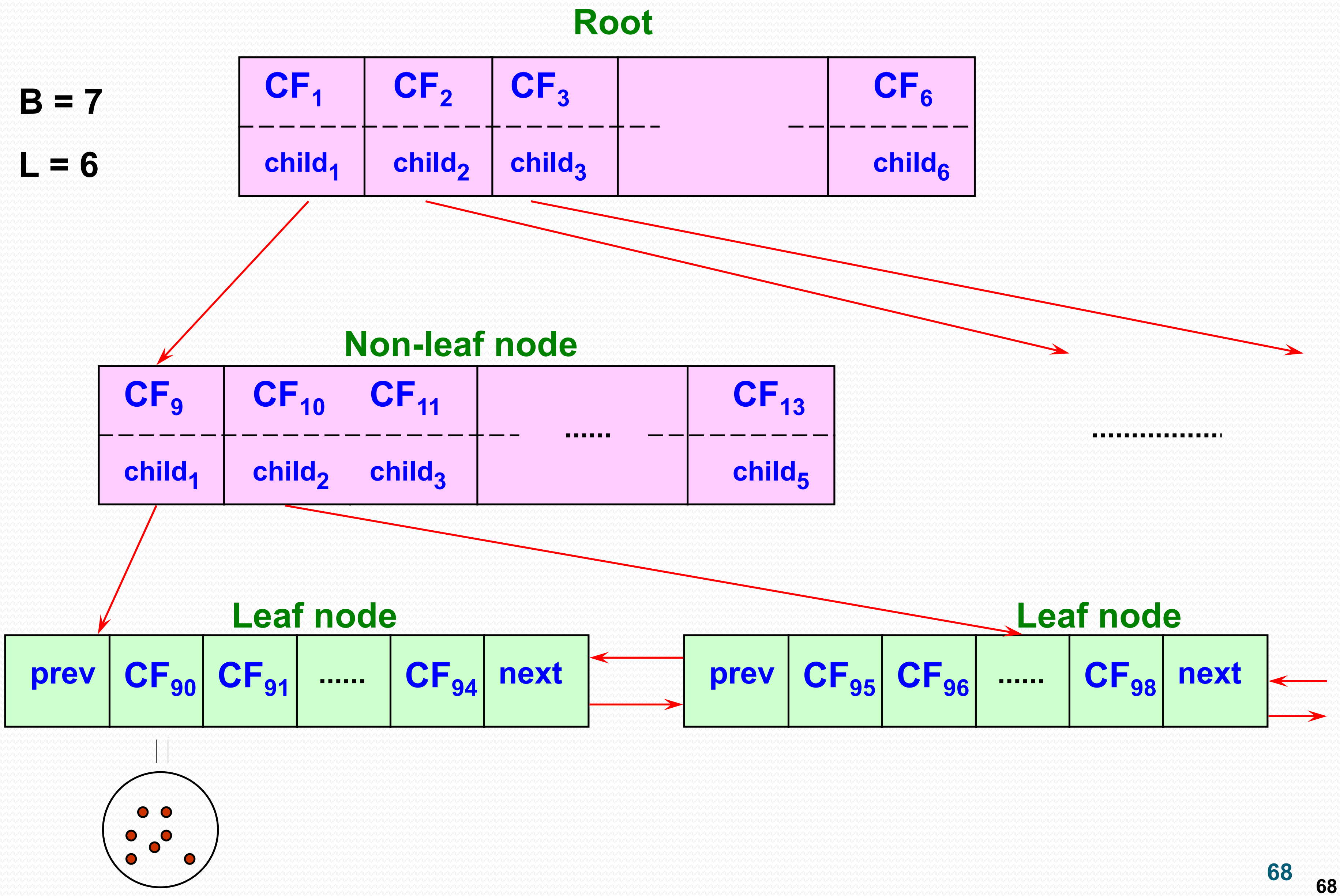
- Why CF and Additive theorem?
- Answer: CF-tree

- Clustering feature tree (*CFT*) is an alternative representation of data set:
  - Each non-leaf node is a cluster comprising sub-clusters corresponding to entries (at most ***B***) in non-leaf node
  - Each leaf node is a cluster comprising sub-clusters corresponding to entries (at most ***L***) in leaf node
  - Each sub-cluster's diameter is at most *T*, when *T* is larger, *CFT* is smaller
  - Each node must fit a memory page

Example of CF Tree

B = 7

L = 6

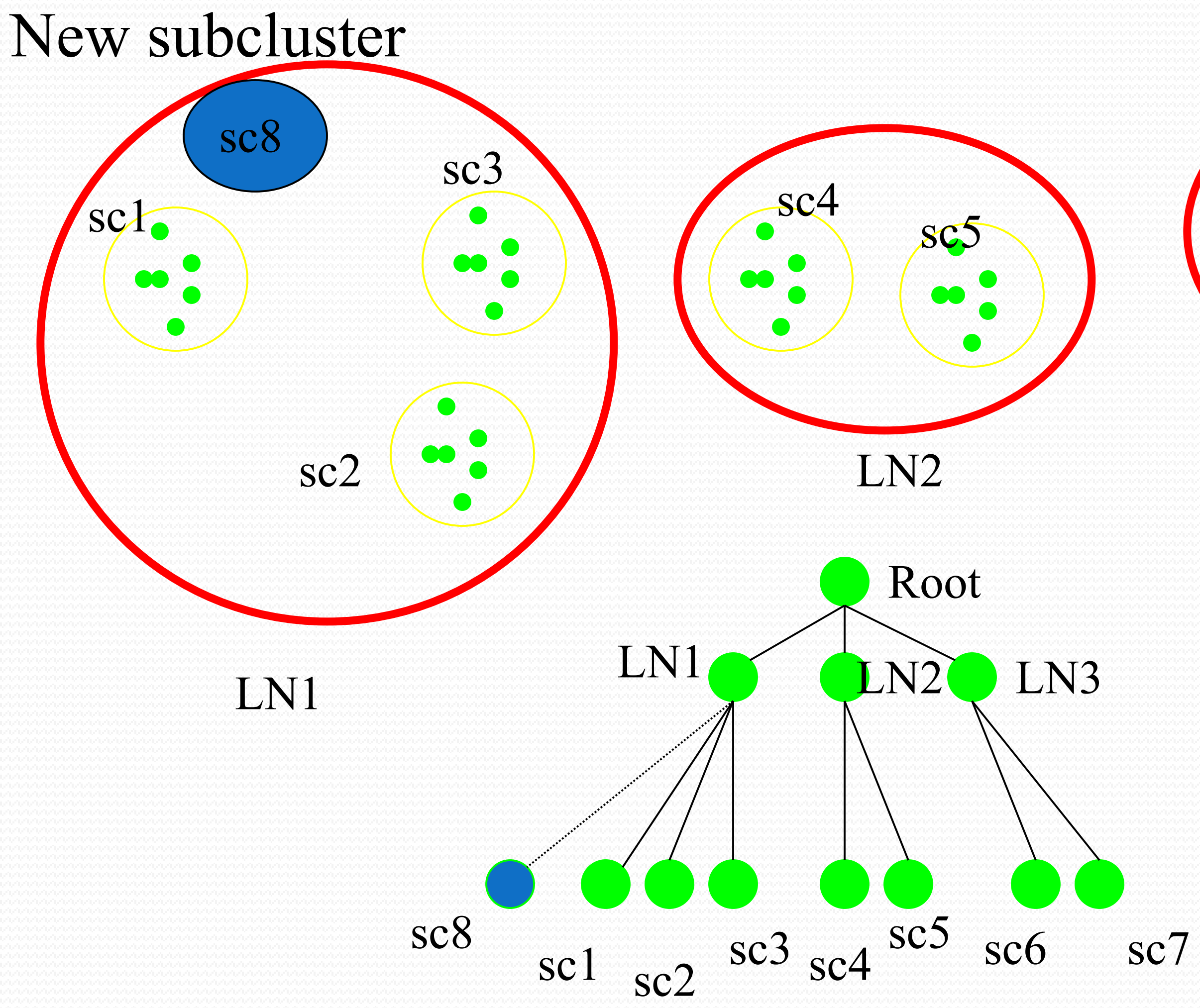


Root
CF_1 CF_2 CF_3 CF_6
$child_1$ $child_2$ $child_3$ $child_6$
Non-leaf node
CF_9 CF_{10} CF_{11} ...... CF_{13}
$child_1$ $child_2$ $child_3$ $child_5$
....
Leaf node
prev CF_{90} CF_{91} ...... CF_{94} next
Leaf node
prev CF_{95} CF_{96} ...... CF_{98} next

BIRCH Phase 1

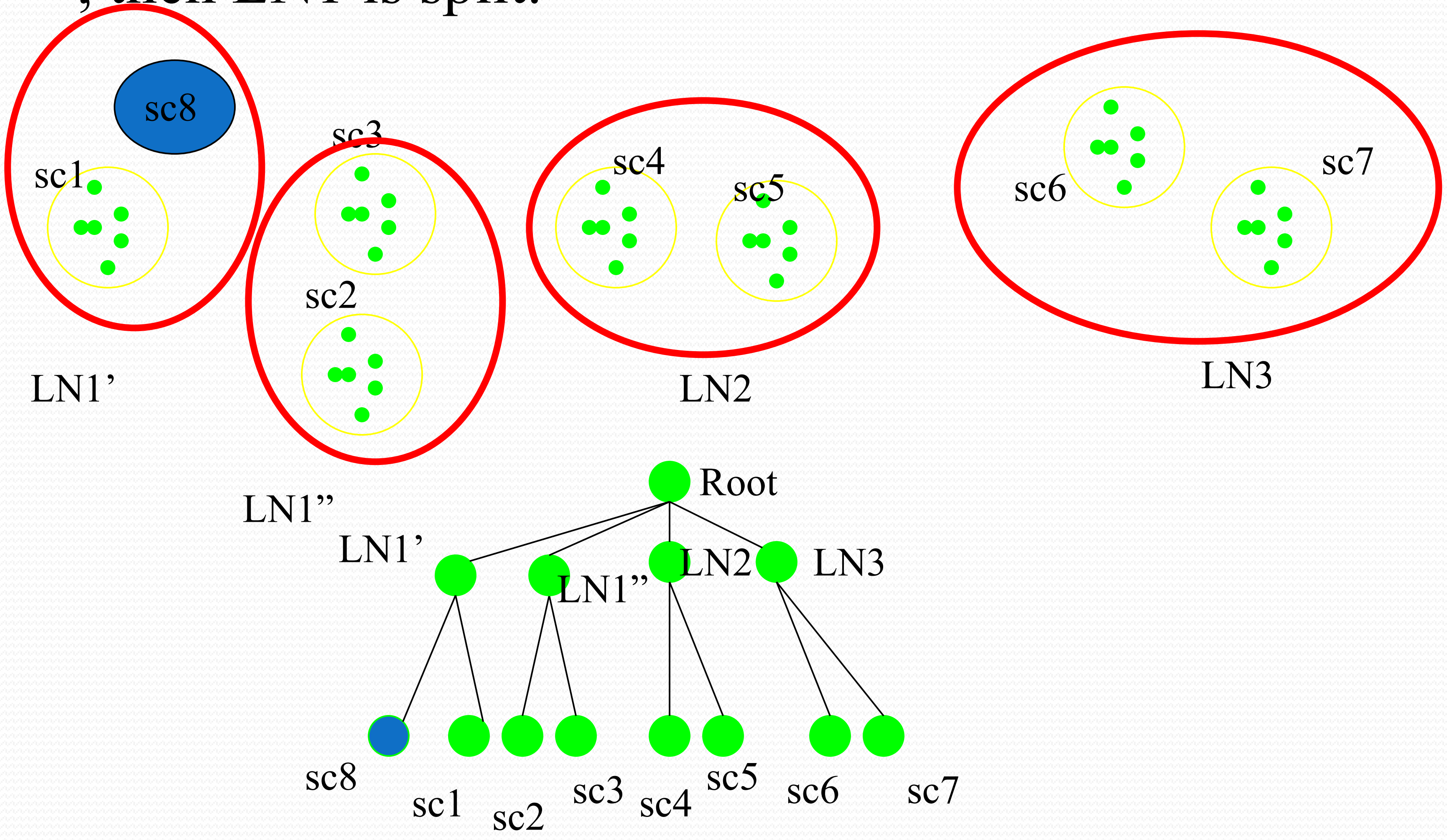
- Phase 1 scans data points and build in-memory CFT;
- Start from root, traverse down tree to choose closest leaf node for *d*
- Search for closest entry L_i in leaf node
- If *d* can be inserted in L_i, then update CF vector of L_i
- Else if node has space to insert new entry, insert; else split node
- Once inserted, update nodes along path to the root; if there is splitting, need to insert new entry in parent node (which may result in further splitting)

Example of the BIRCH Algorithm



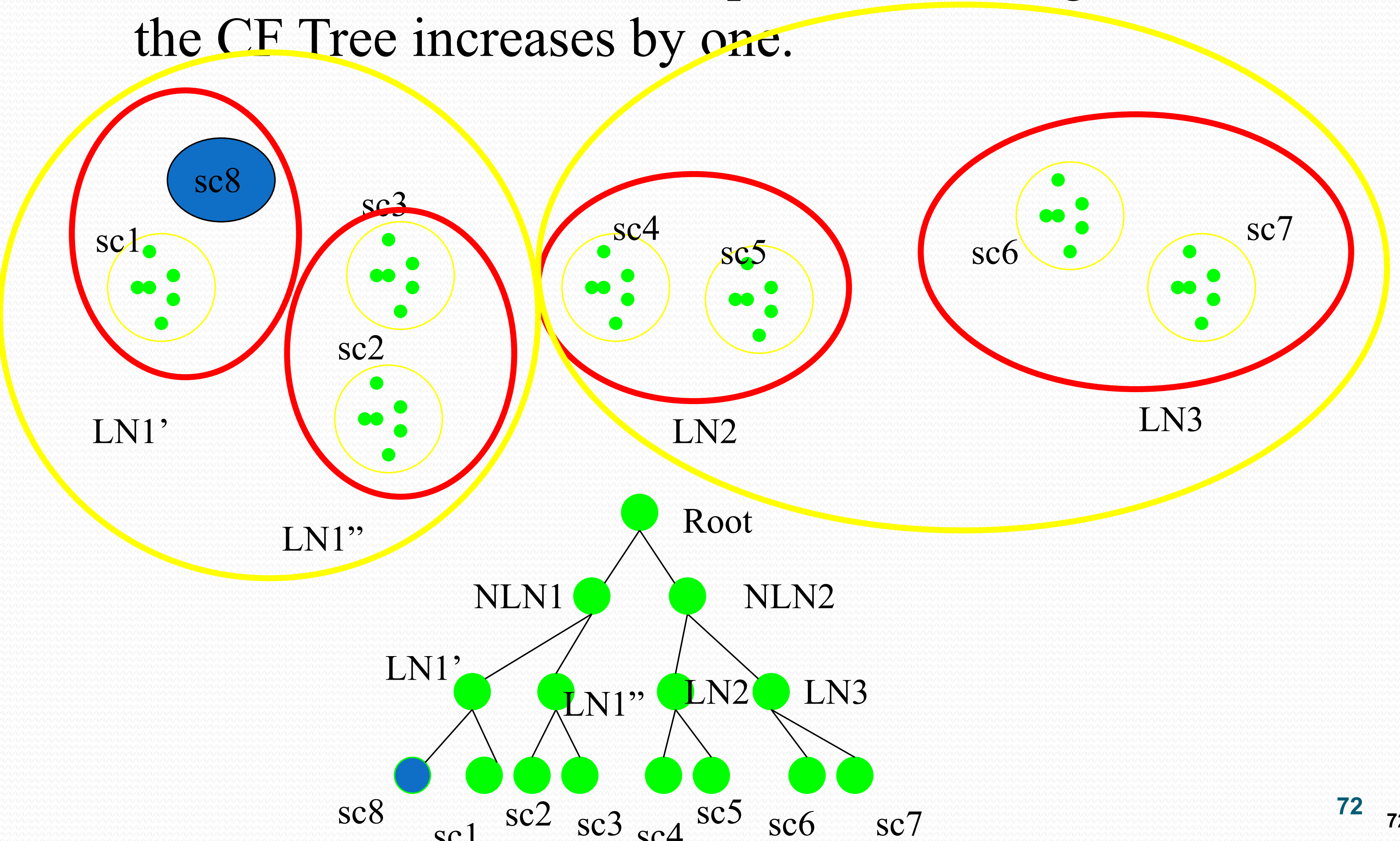
Merge Operation in BIRCH

If the branching factor of a leaf node can not exceed 3 , then LN1 is split.



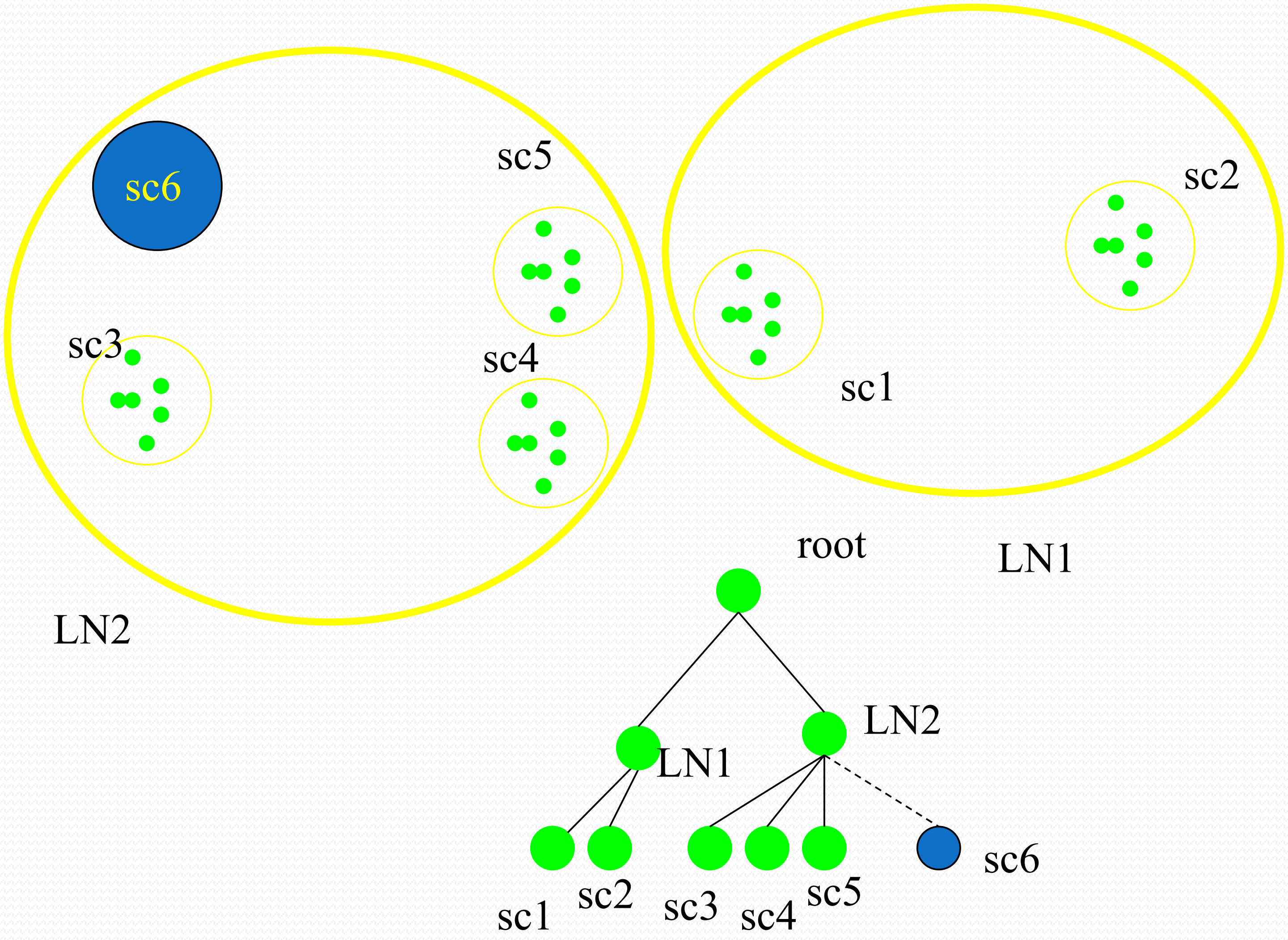
Merge Operation in BIRCH

If the branching factor of a non-leaf node can not exceed 3, then the root is split and the height of the CF Tree increases by one.

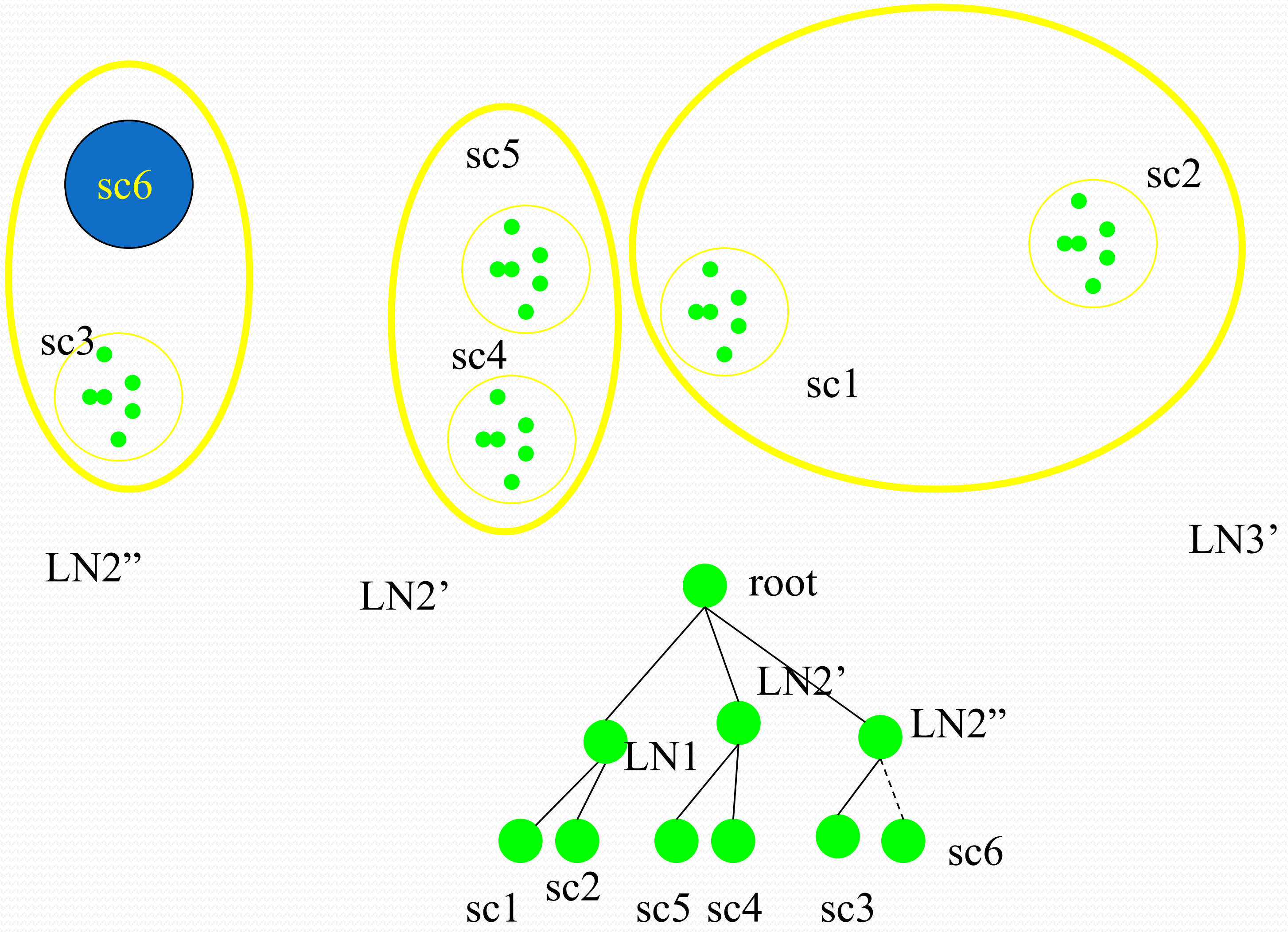


Merge Operation in BIRCH

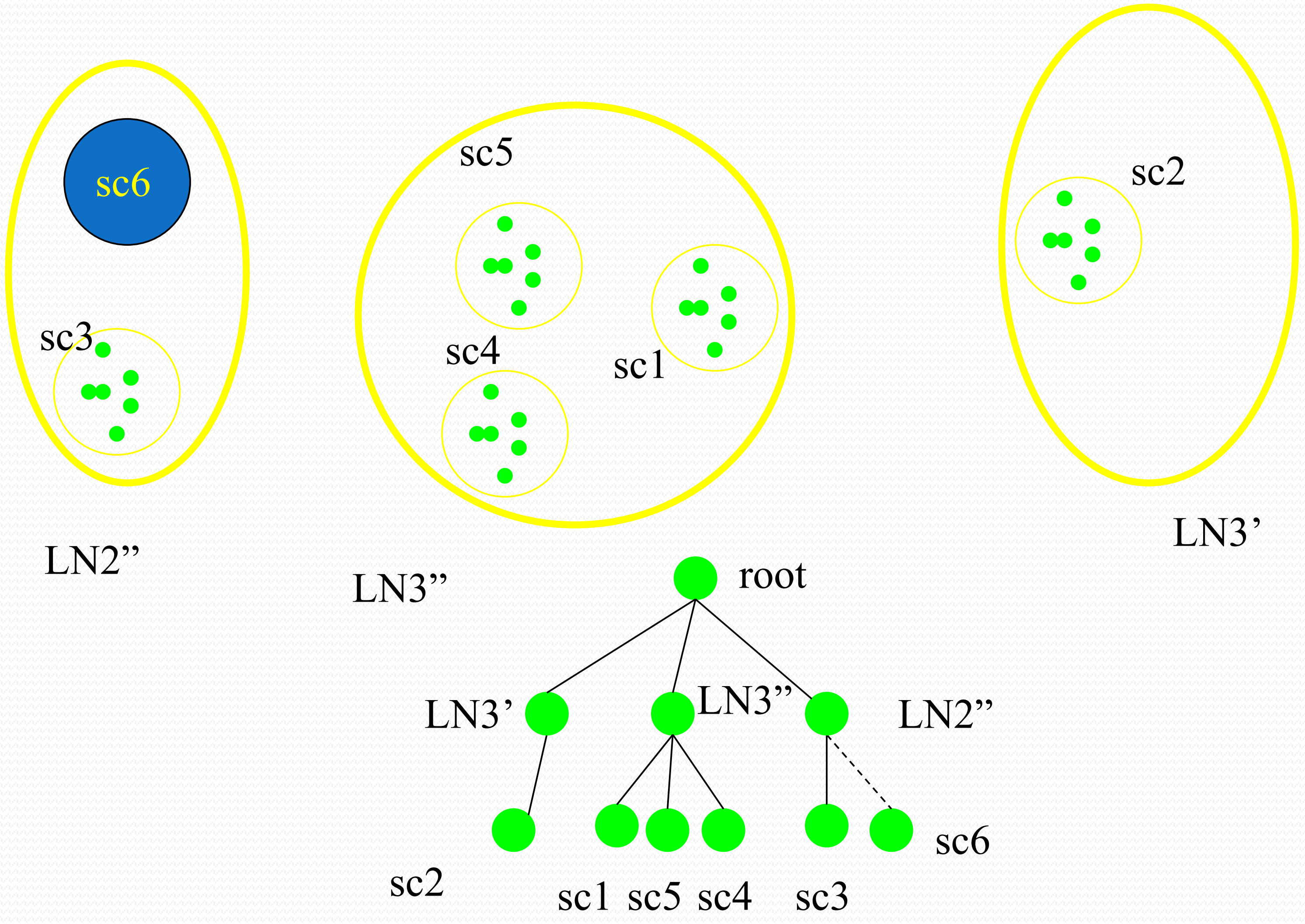
Assume that the subclusters are numbered according to the order of formation.



If the branching factor of a leaf node can not exceed 3, Then LN2 is split.



LN2’ and LN1 will be merged, and the newly formed Node will be split immediately.



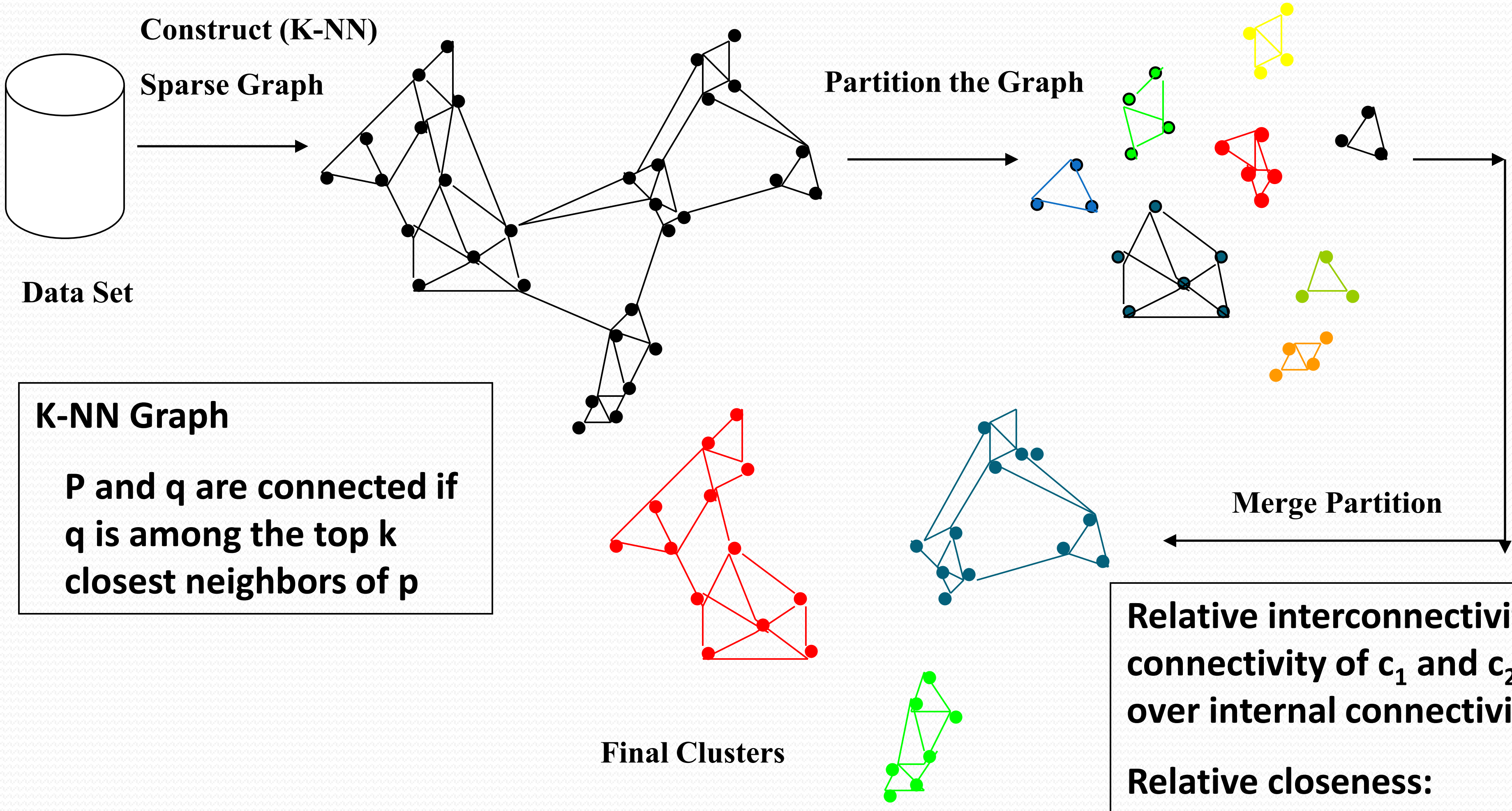
BIRCH Phases X

- Phase: Global clustering
  - Use existing clustering (e.g., AGNES) algorithm on sub-clusters at leaf nodes
  - May treat each sub-cluster as a point (its centroid) and perform clustering on these points
  - Clusters produced closer to data distribution pattern
- Phase: Cluster refinement
  - Redistribute (re-label) all data points w.r.t. clusters produced by global clustering
  - This phase may be repeated to improve cluster quality

BIRCH: Pros and Cons

- Pros:
  - Faster, Memory-efficiency, outlier detection
- Cons:
  - Threshold for each nodes may bias the cluster derived
  - Distance-based clustering, not good for concave shaped clusters

Overall Framework of CHAMELEON



K-NN Graph

P and q are connected if q is among the top k closest neighbors of p

Relative interconnectivity: connectivity of c_1 and c_2 over internal connectivity

Relative closeness: closeness of c_1 and c_2 over internal closeness

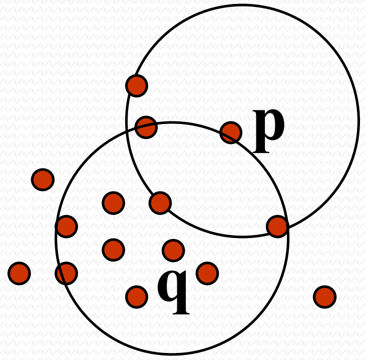
Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

Density-Based Clustering: Basic Concepts

- Two parameters:
  - *Eps*: Maximum radius of the neighbourhood
  - *MinPts*: Minimum number of points in an Eps-neighbourhood of that point
- $N_{Eps}(p)$: {q belongs to D | dist(p,q) ≤ Eps}
- Directly density-reachable: A point *p* is directly density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if
  - *p* belongs to $N_{Eps}(q)$
  - core point condition:

$$|N_{Eps}(q)| \geq MinPts$$

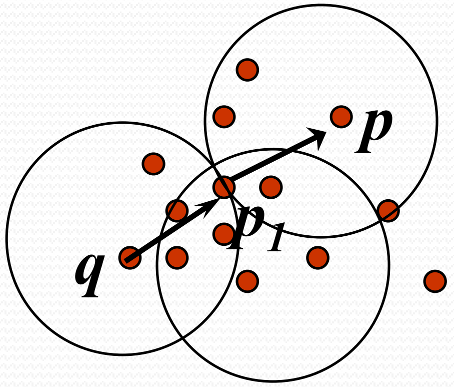


MinPts = 5

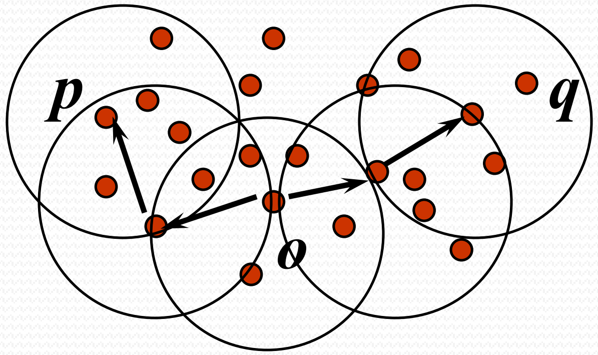
Eps = 1 cm

Density-Reachable and Density-Connected

- Density-reachable:
  - A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i

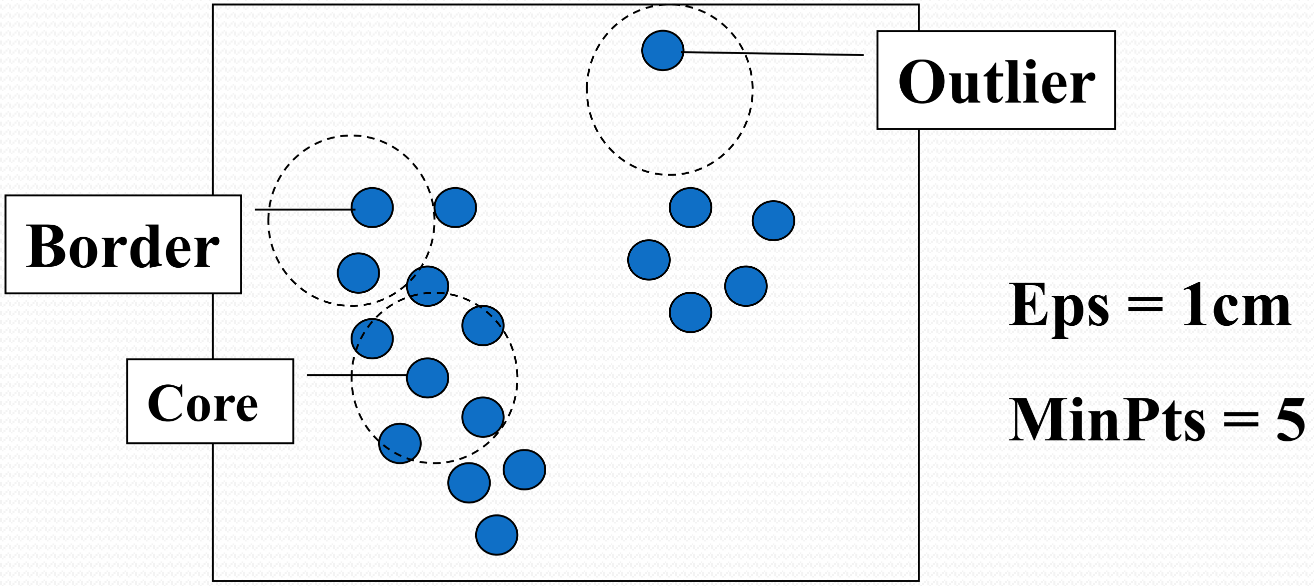


- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point o such that both, *p* and *q* are density-reachable from o w.r.t. *Eps* and *MinPts*



DBSCAN: Density-Based Spatial Clustering of Applications with Noise

- Relies on a *density-based* notion of cluster: A ***cluster*** is defined as a **maximal set of density-connected points**
- Discovers clusters of **arbitrary shape** in spatial databases with noise



DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*
- If *p* is a core point, a cluster is formed
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed

DBSCAN: Sensitive to Parameters

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

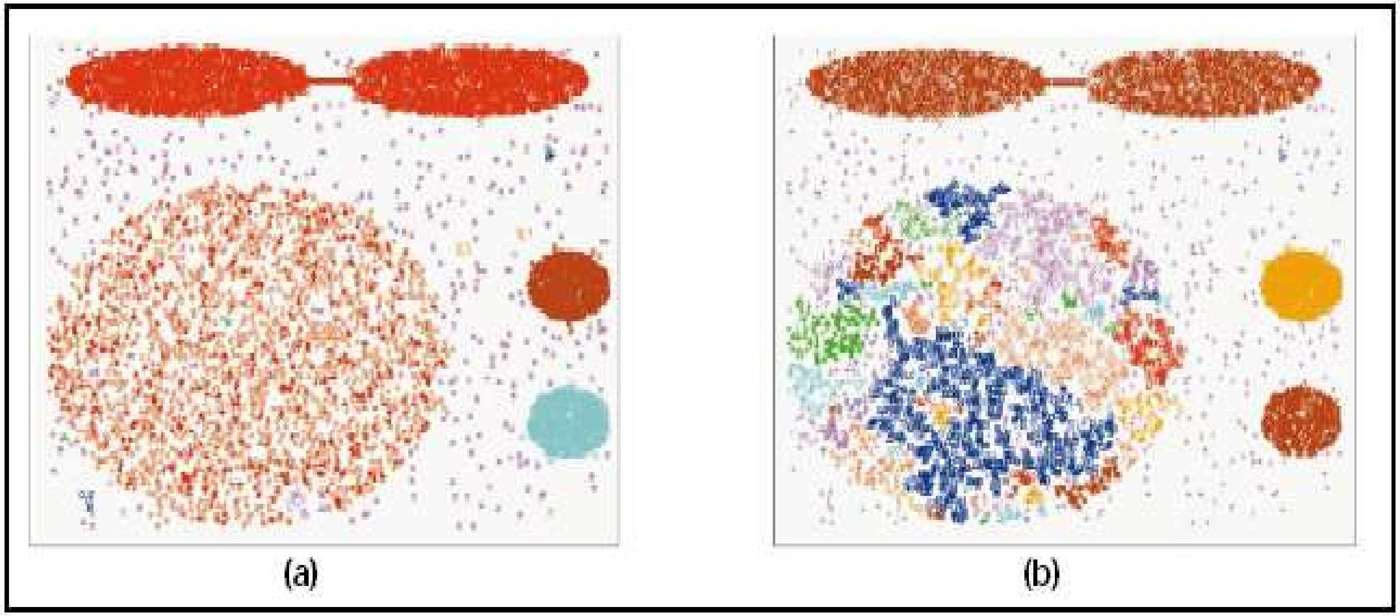
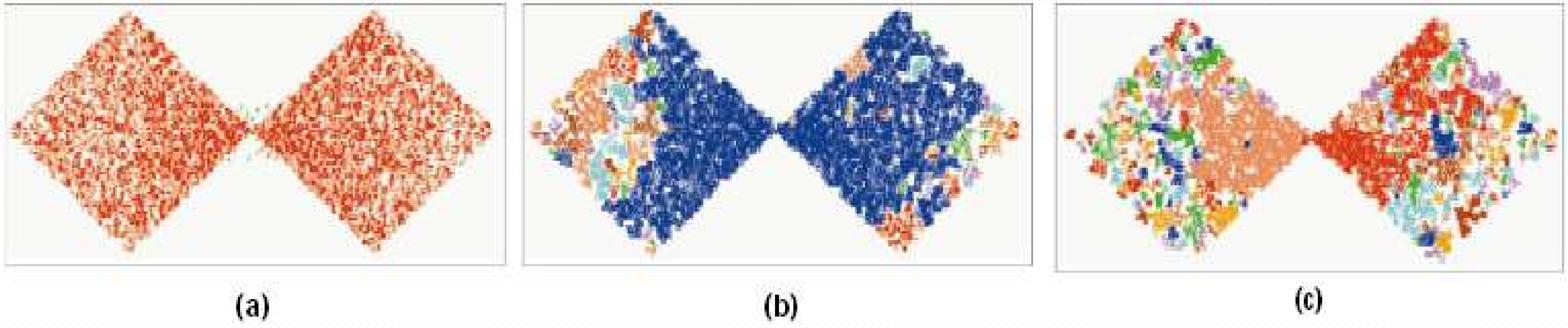


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.

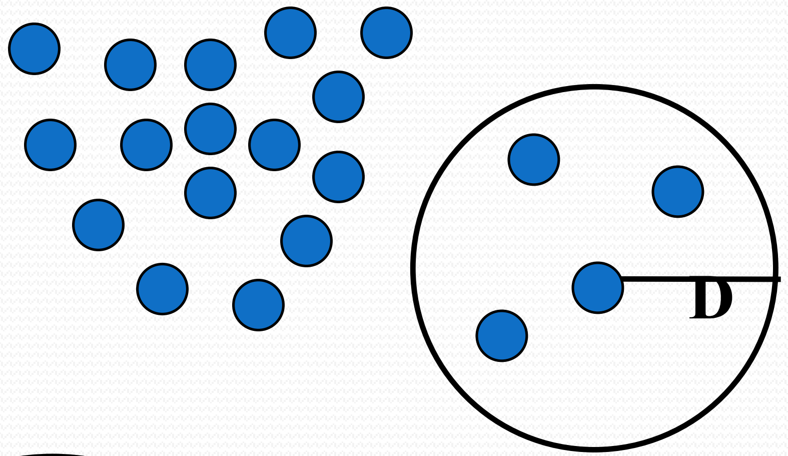


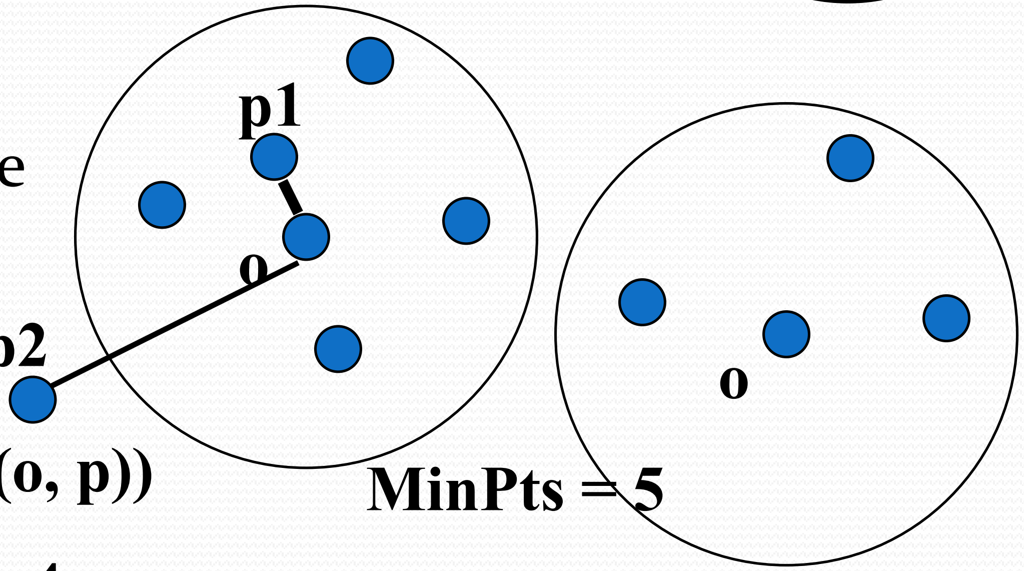
OPTICS: A Cluster-Ordering Method (1999)

- OPTICS: Ordering Points To Identify the Clustering Structure
  - Ankerst, Breunig, Kriegel, and Sander (SIGMOD'99)
  - Produces a special order of the database wrt its density-based clustering structure
  - This cluster-ordering contains info equiv to the density-based clusterings corresponding to a broad range of parameter settings
  - Good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure
  - Can be represented graphically or using visualization techniques

OPTICS: Some Extension from DBSCAN

- Index-based:
    - k = number of dimensions
    - N = 20
    - p = 75%
    - M = N(1-p) = 5
  - Complexity: O(*NlogN*)
- Core Distance:
  - min eps s.t. point is core
- Reachability Distance





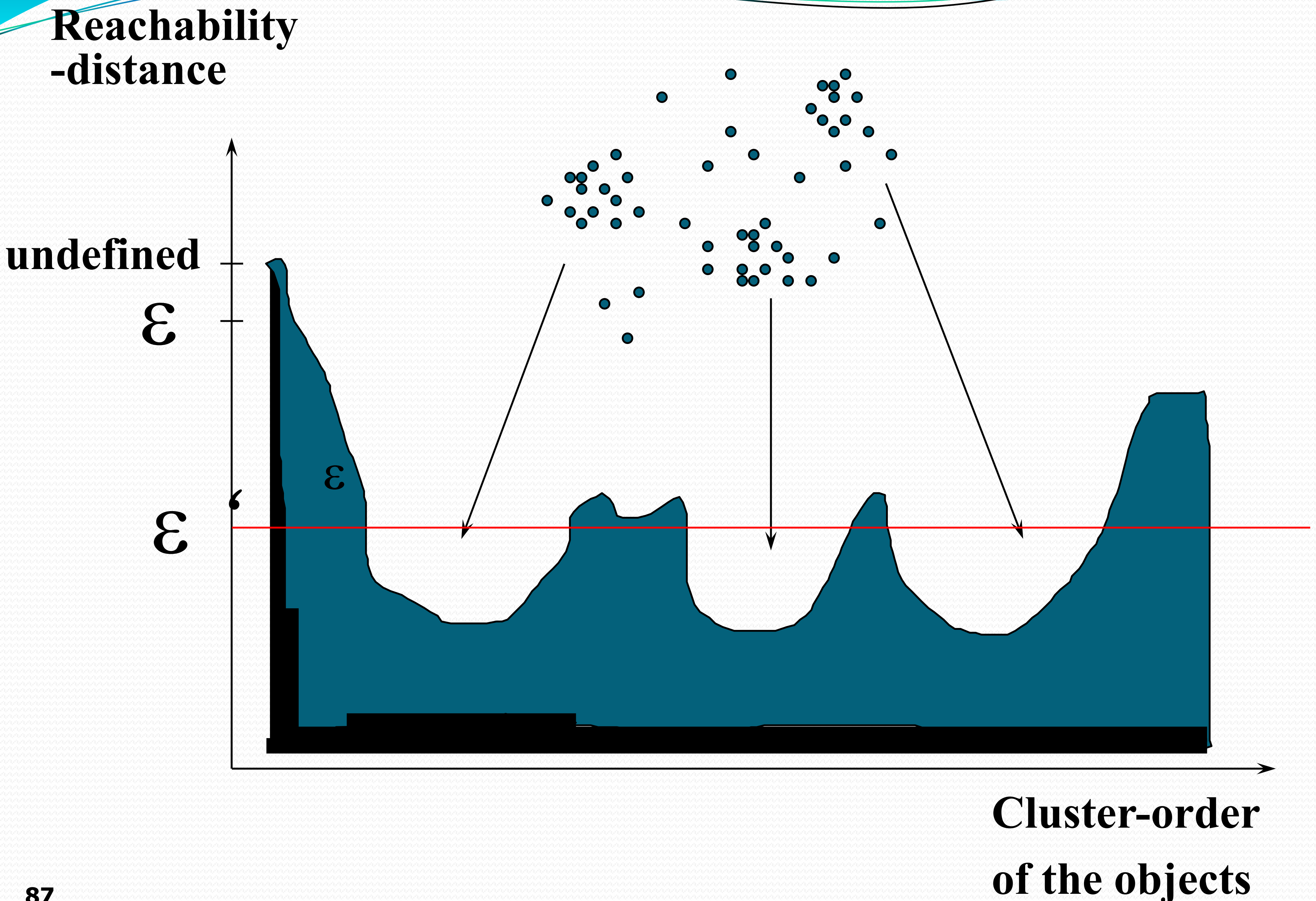
Max (core-distance (o), d (o, p))

r(p1, o) = 2.8cm. r(p2,o) = 4cm

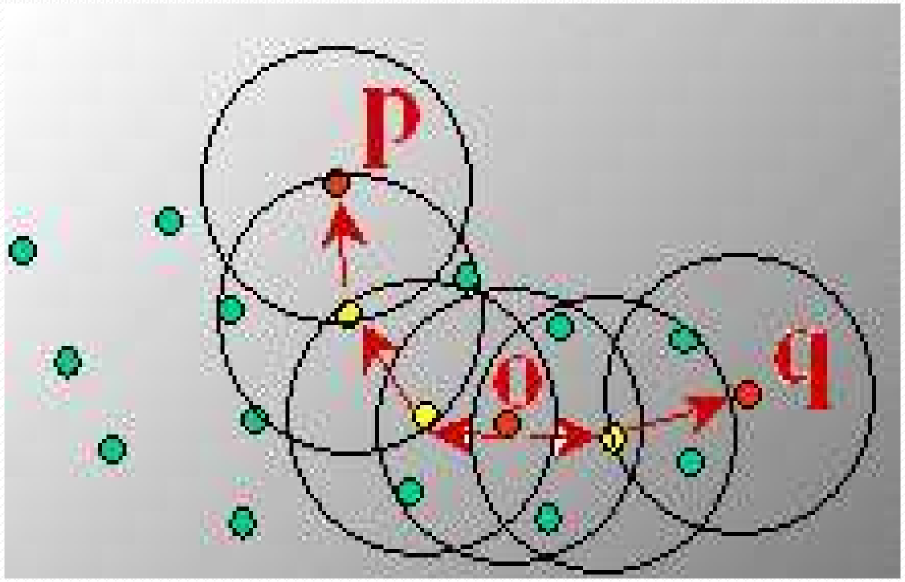
MinPts = 5

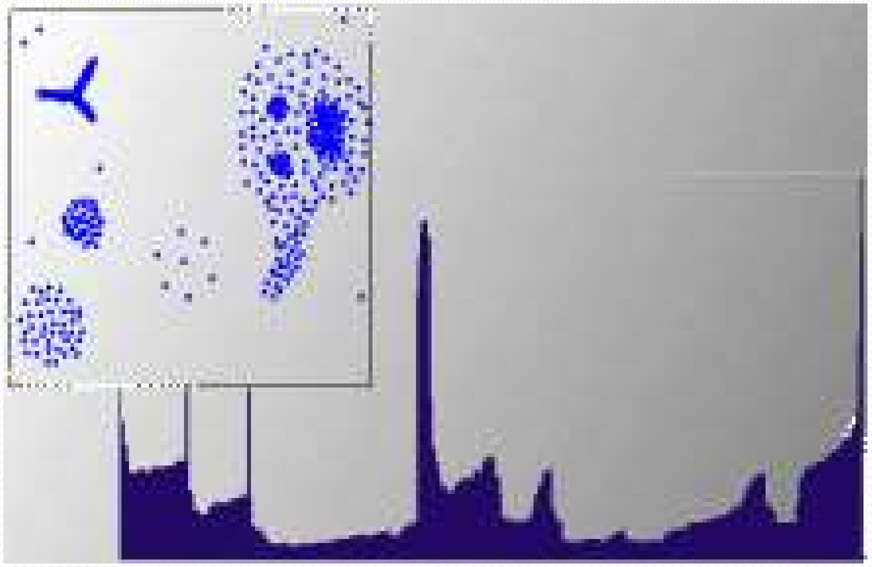
ε = 3 cm

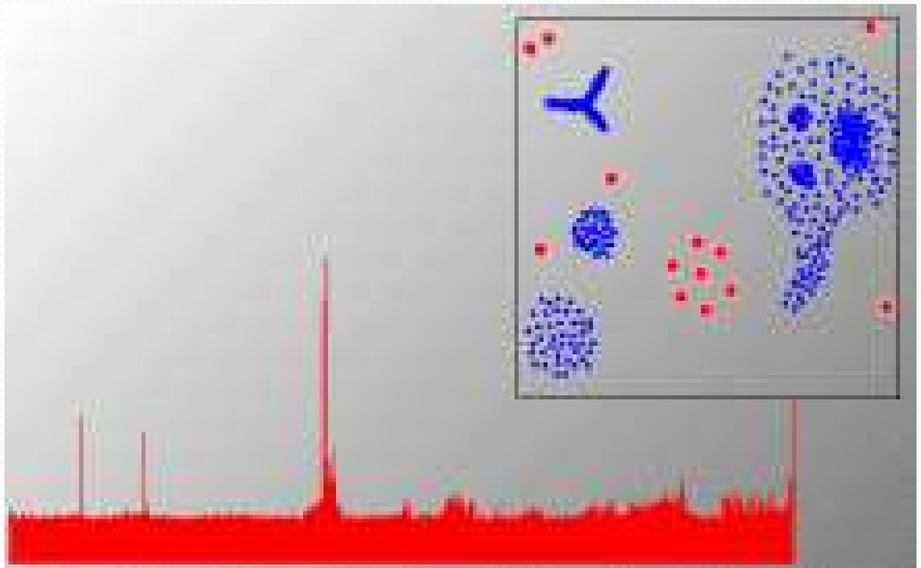
86

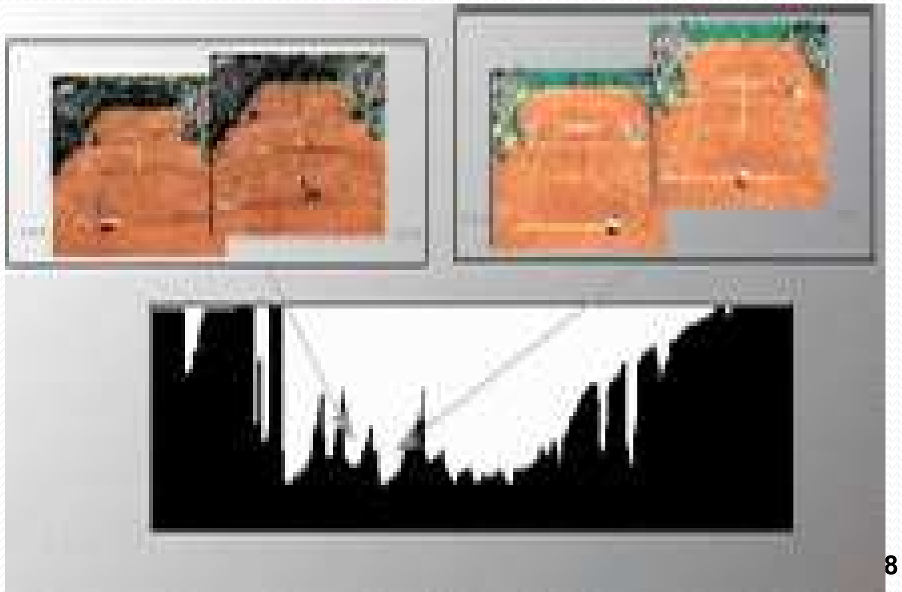

Reachability
-distance
undefined
ε
ε
ε'
Cluster-order
of the objects

Density-Based Clustering: OPTICS & Its Applications









DENCLUE: Using Statistical Density Functions

- DENsity-based CLUstEring by Hinneburg & Keim (KDD'98)
- Using statistical density functions:

$$f_{Gaussian}(x,y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

influence of y on x

$$f^{D}_{Gaussian}(x) = \sum_{i=1}^{N} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

total influence on x

$$\nabla f^{D}_{Gaussian}(x, x_i) = \sum_{i=1}^{N} (x_i - x) \cdot e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

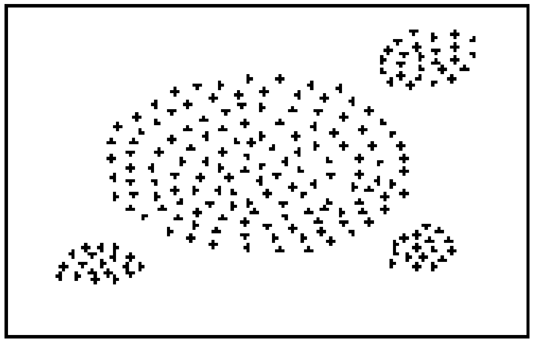
gradient of x in the direction of x_i

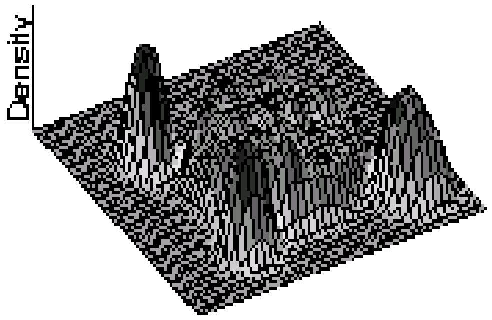
- Major features
  - Solid mathematical foundation
  - Good for data sets with large amounts of noise
  - Allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets
  - Significant faster than existing algorithm (e.g., DBSCAN)
  - But needs a large number of parameters

Denclue: Technical Essence

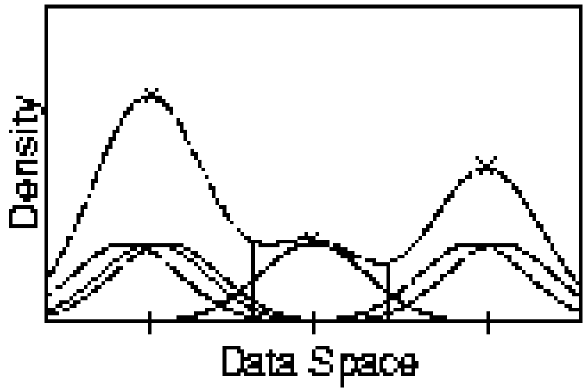
- Uses grid cells but only keeps information about grid cells that do actually contain data points and manages these cells in a tree-based access structure
- Influence function: describes the impact of a data point within its neighborhood
- Overall density of the data space can be calculated as the sum of the influence function of all data points
- Clusters can be determined mathematically by identifying density attractors
- Density attractors are local maximal of the overall density function
- Center defined clusters: assign to each density attractor the points density attracted to it
- Arbitrary shaped cluster: merge density attractors that are connected through paths of high density (> threshold)

Density Attractor


(a) Data Set



(c) Gaussian



Center-Defined and Arbitrary

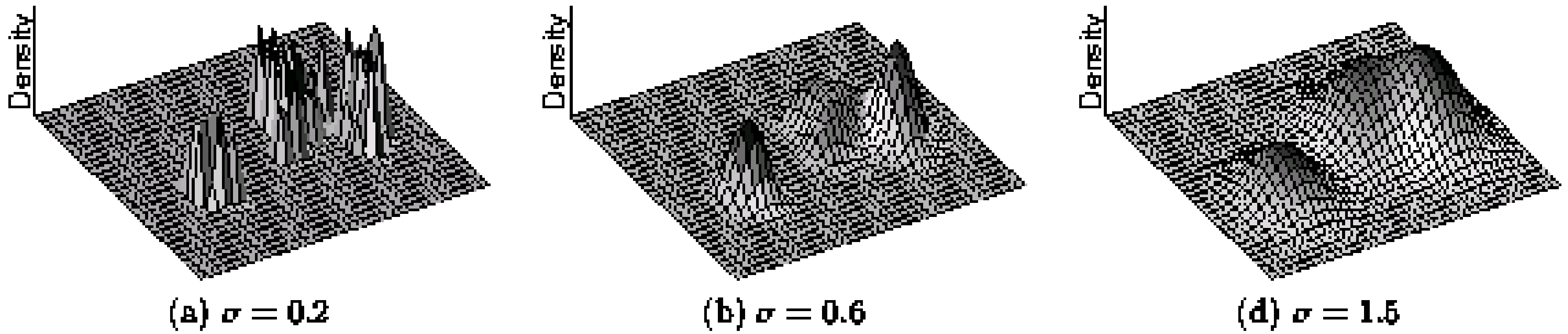


Figure 3: Example of Center-Defined Clusters for different σ

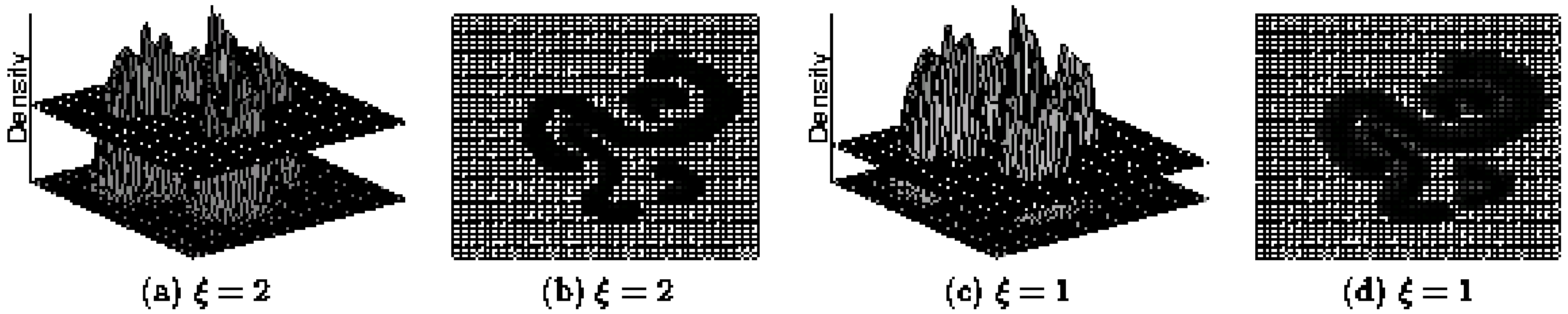


Figure 4: Example of Arbitray-Shape Clusters for different ξ