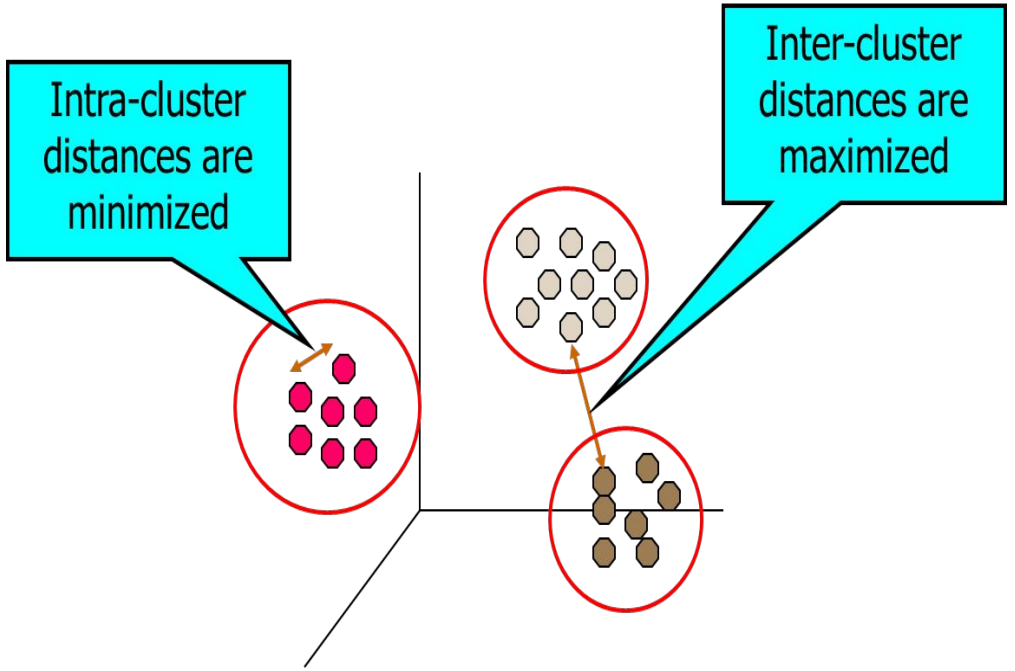
Clustering Techniques

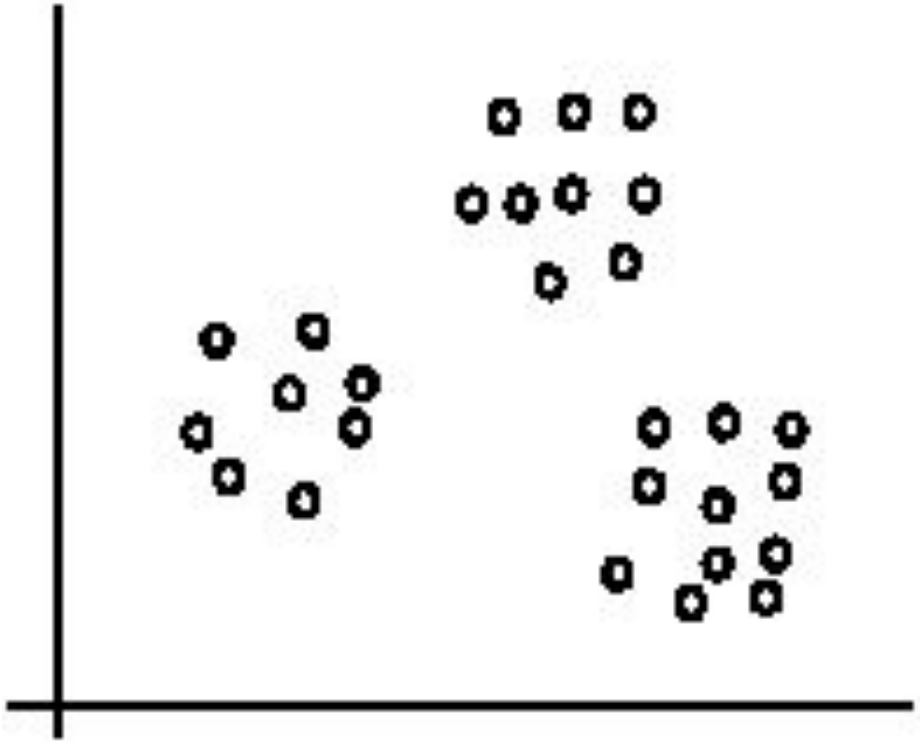
Clustering

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Clustering is a technique for finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.



An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



Notion of a Cluster can be Ambiguous

How many clusters?

Six Clusters

Two Clusters

Four Clusters

What Is Good Clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.

Examples of Clustering Applications

- **Marketing:** Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **Land use:** Identification of areas of similar land use in an earth observation database
- **Insurance:** Identifying groups of motor insurance policy holders with a high average claim cost
- **City-planning:** Identifying groups of houses according to their house type, value, and geographical location
- **Earth-quake studies:** Observed earth quake epicenters should be clustered along continent faults

Requirements of Clustering in Data Mining

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

Major Clustering Approaches

- **Partitioning algorithms:** Construct various partitions and then evaluate them by some criterion
- **Hierarchy algorithms:** Create a hierarchical decomposition of the set of data (or objects) using some criterion
- **Density-based:** A cluster is defined as a maximal set of density connected points
- **Grid-based:** These algorithms partition the data space into a finite number of cells to form a grid structure and then form clusters from the cells in the grid structure. Clusters correspond to regions that are more dense in data points than their surroundings

Partitioning Clustering Approach

- Partitioning algorithms construct partition of a database of N objects into a set of K clusters.
- The partitioning clustering algorithm usually adopts the Iterative Optimization paradigm.
  - It starts with an initial partition and uses an iterative control strategy.
  - It tries swapping data points to see if such a swapping improves the quality of clustering.
  - When swapping does not yield any improvements in clustering, it finds a locally optimal partitioning
- in principle, optimal partition achieved via minimizing the sum of squared distance to its "representative object" in each cluster

$$E = \Sigma_{k=1}^{K} \Sigma_{\mathbf{x} \in C_k} Distance(\mathbf{x}, \mathbf{m}_k)$$

e.g., Euclidean distance $$Distance(\mathbf{x}, \mathbf{m}_k) = \sum_{d=1}^{D} (x_d - m_{kd})^2$$

K-means algorithm

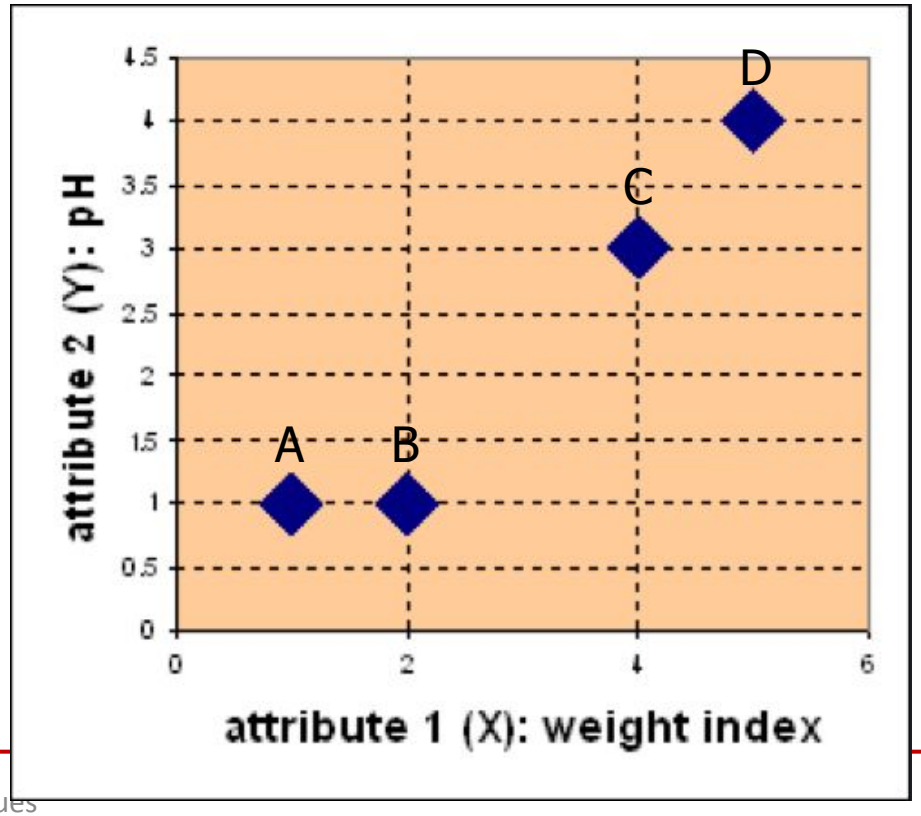
- Given the cluster number *K*, the *K-means* algorithm is carried out in three steps after initialization:
  - **Initialization: set seed points (randomly)**

1. **Assign each object to the cluster of the nearest seed point measured with a specific distance metric**
2. **Compute new seed points as the centroids of the clusters of the current partition (the centroid is the centre, i.e., *mean point*, of the cluster)**
3. **Go back to Step 1), stop when no more new assignment (i.e., membership in each cluster no longer changes)**

K-means - Example

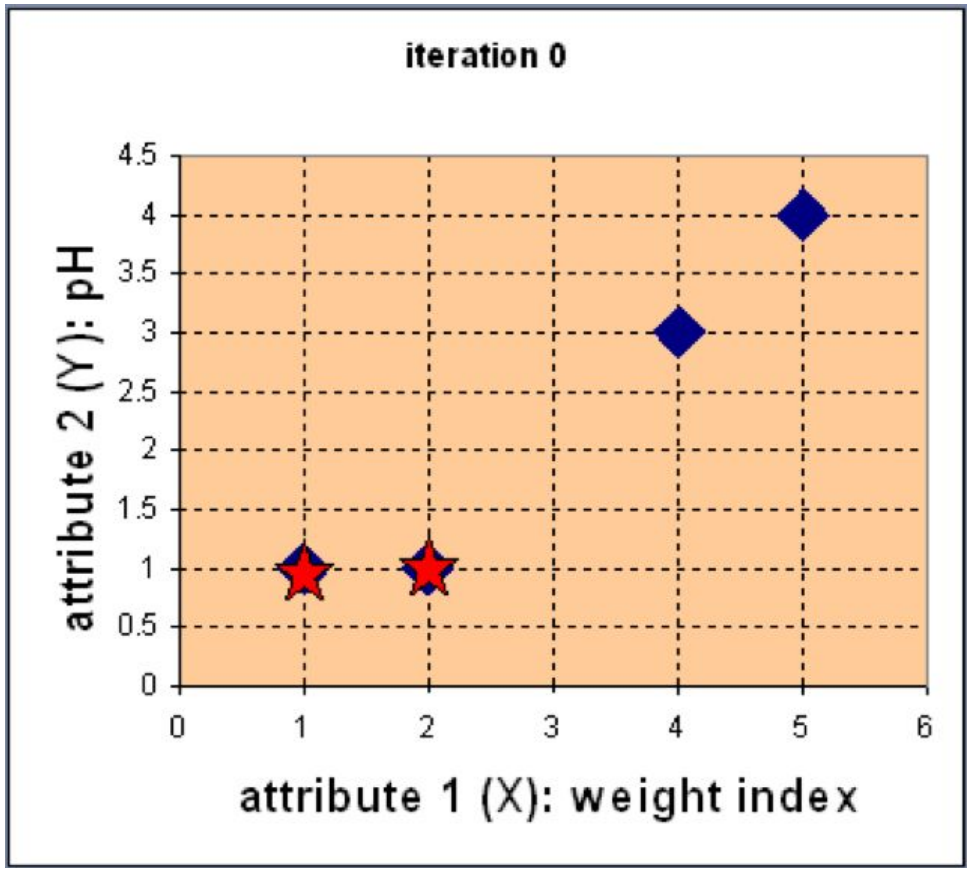
- **Problem:**
  - **Suppose we have 4 types of medicines and each has two attributes (pH and weight index). Our goal is to group these objects into *K=2* group of medicine.**

Medicine	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4



K-means - Example

- **Step 1: Use initial seed points for partitioning**



$$c_1 = A\,,\; c_2 = B$$

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{matrix} \mathbf{c}_1 = (1,1) & group-1 \\ \mathbf{c}_2 = (2,1) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

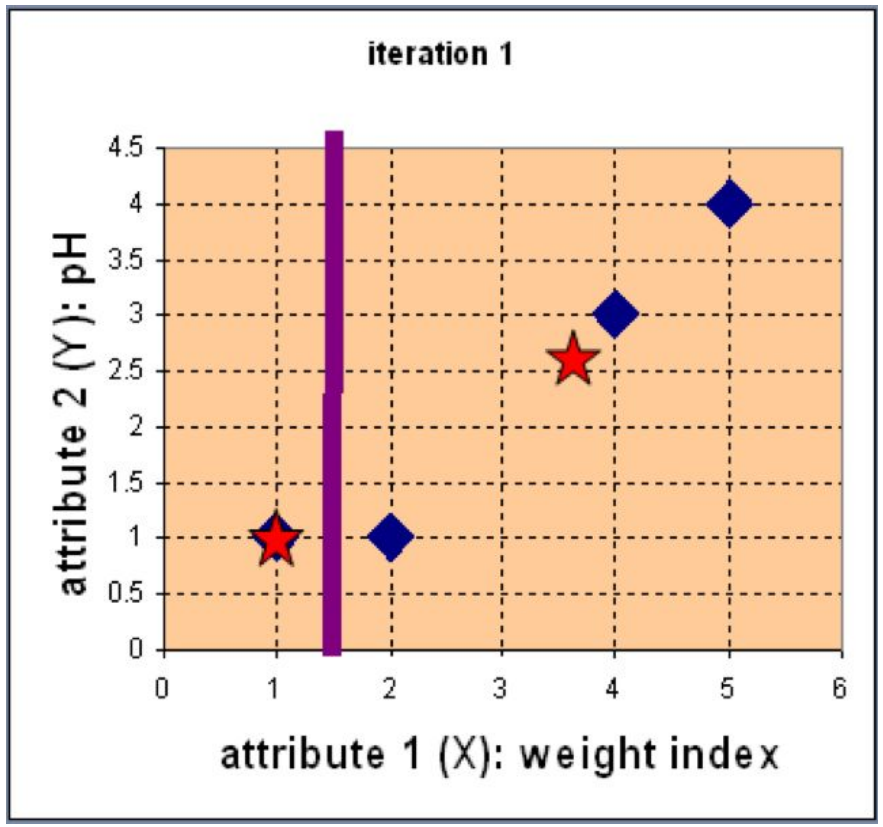
$$d(D, c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$d(D, c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Assign each object to the cluster with the nearest seed point

K-means - Example

- **Step 2: Compute new centroids of the current partition**



Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

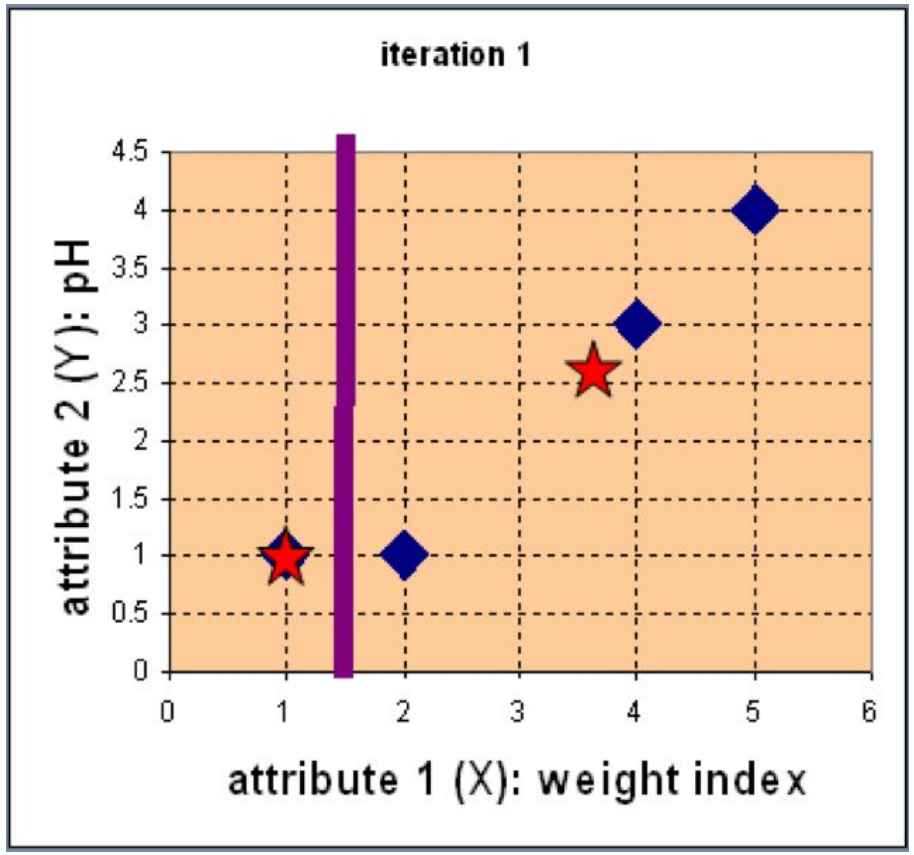
$$c_1 = (1,\ 1)$$

$$c_2 = \left(\frac{2+4+5}{3},\ \frac{1+3+4}{3}\right)$$

$$= \left(\frac{11}{3},\ \frac{8}{3}\right)$$

K-means - Example

- **Step 2: Renew membership based on new centroids**



Compute the distance of all objects to the new centroids

$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{matrix} c_1 = (1,1) & group-1 \\ c_2 = (\frac{11}{3}, \frac{8}{3}) & group-2 \end{matrix}$$

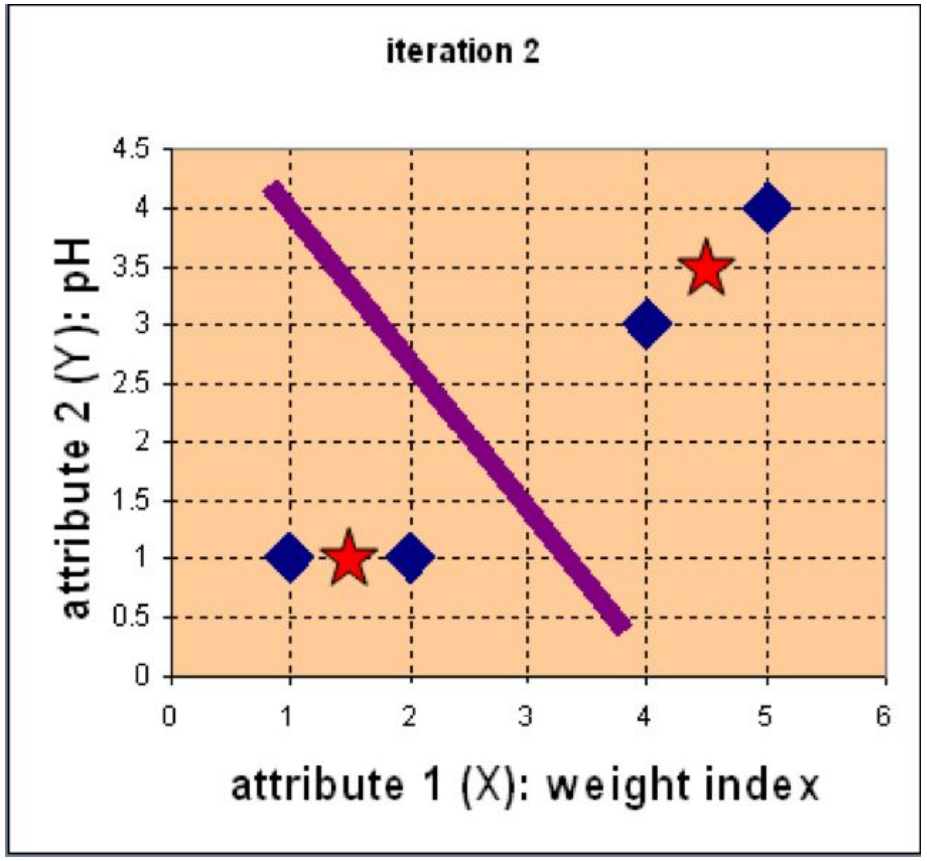
$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

Assign the membership to objects

K-means - Example

- **Step 3: Repeat the first two steps until its convergence**



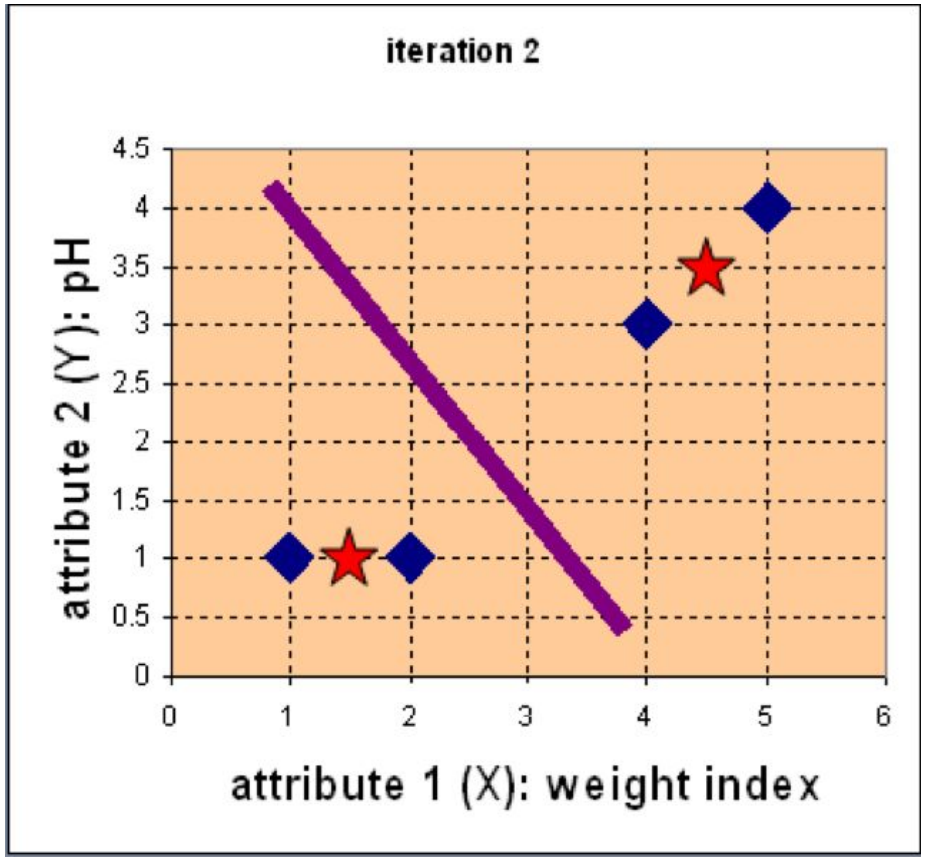
Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1\frac{1}{2}, 1)$$

$$c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4\frac{1}{2}, 3\frac{1}{2})$$

K-means - Example

- **Step 3: Repeat the first two steps until its convergence**



Compute the distance of all objects to the new centroids

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \begin{matrix} c_1 = (1\frac{1}{2}, 1) & group-1 \\ c_2 = (4\frac{1}{2}, 3\frac{1}{2}) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

Stop due to no new assignment
Membership in each cluster no longer change

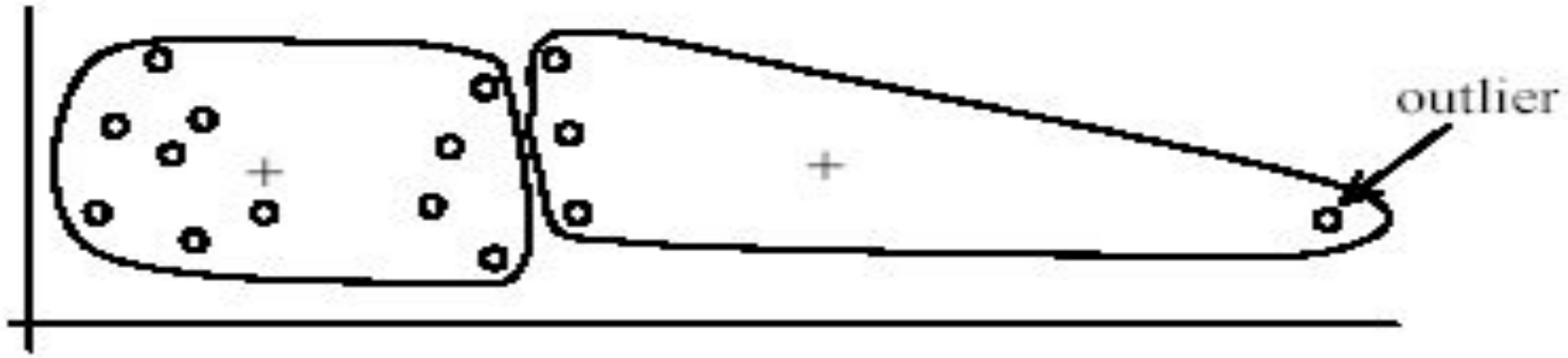
Strengths of k-means

- **Strengths:**
  - **Simple: easy to understand and to implement**
  - **Efficient: Time complexity: O(tkn), where n is the number of data points, k is the number of clusters, and t is the number of iterations.**

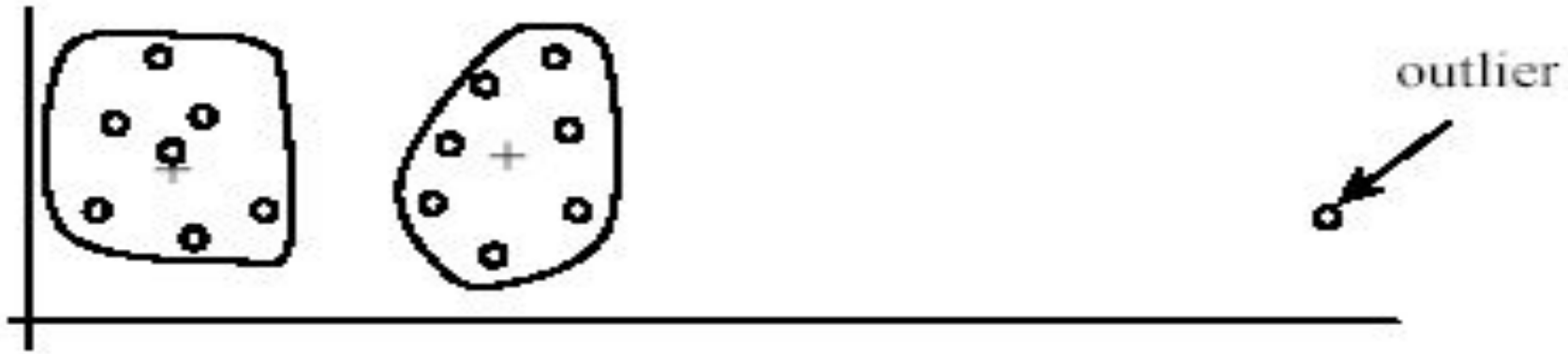
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined.
- The user needs to specify *k*.
- The algorithm is sensitive to outliers
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



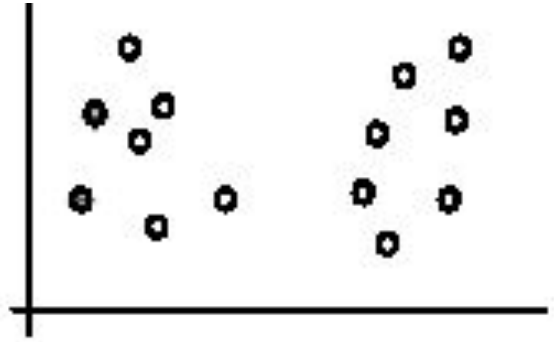
(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

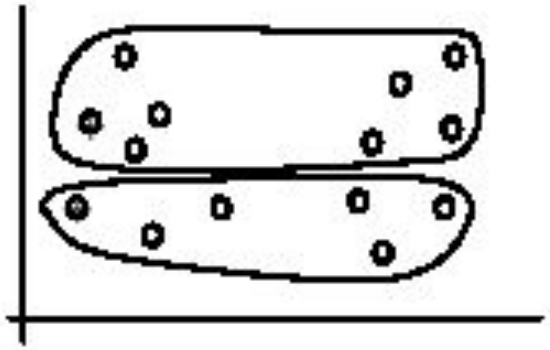
- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means

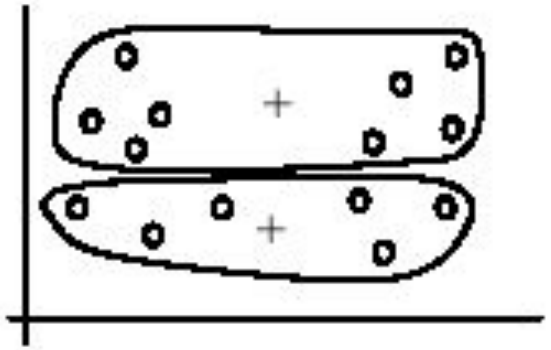
- The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



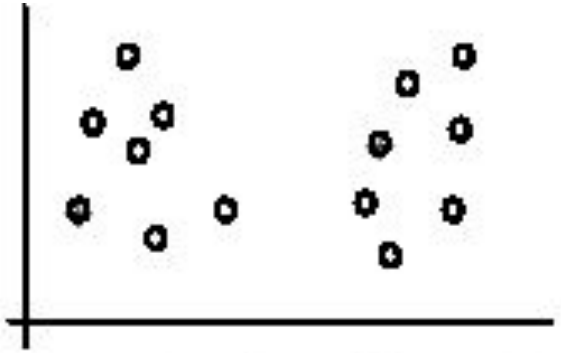
(B). Iteration 1



(C). Iteration 2

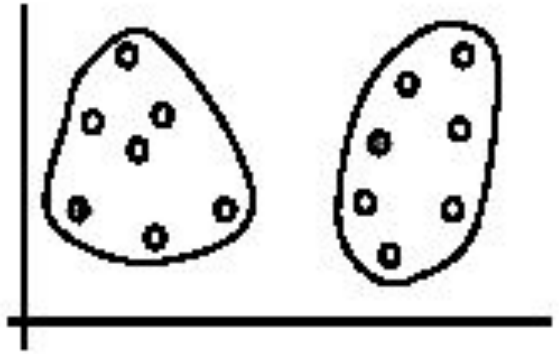
Weaknesses of k-means

- If we use different seeds: good results

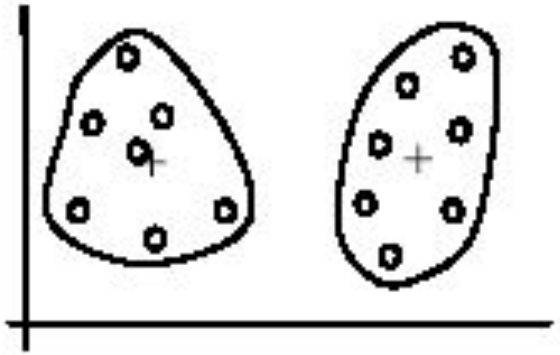


There are some methods to help choose good seeds

(A). Random selection of k seeds (centroids)



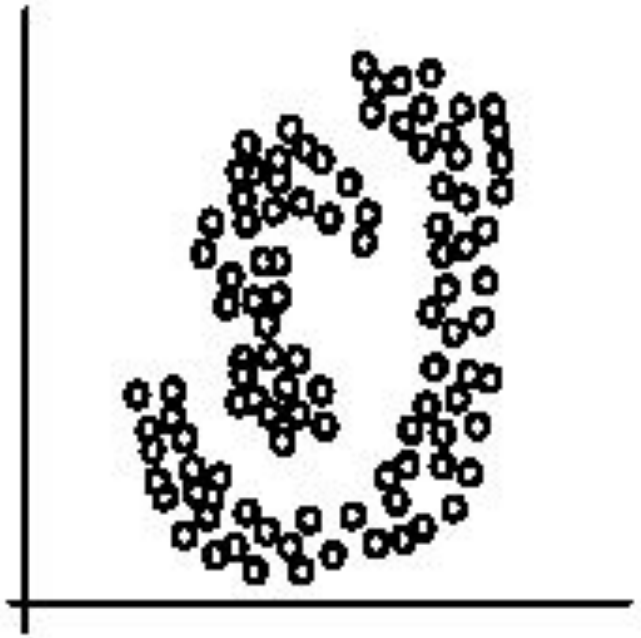
(B). Iteration 1



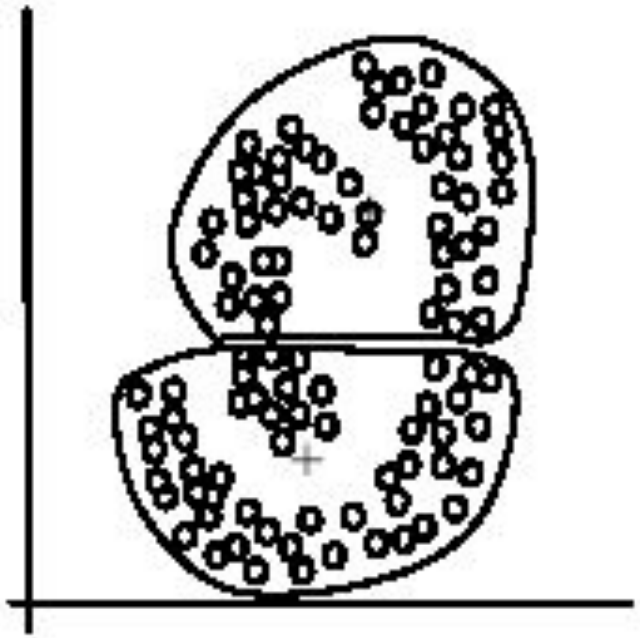
(C). Iteration 2

Weaknesses of k-means

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B): *k*-means clusters

The *K-Medoids* Clustering Method

- Find representative objects, called medoids, in clusters
- PAM (Partitioning Around Medoids)
  - The algorithm is intended to find a sequence of objects called medoids that are centrally located in clusters
  - The goal of the algorithm is to minimize the average dissimilarity of objects to their closest selected object.
  - PAM works effectively for small data sets, but does not scale well for large data sets
- CLARA
- CLARANS

PAM Partition Around Medoids

1) Pick a number, *k*, of random data items as medoids
2) Calculate

$$\arg\min_{(n,m)} TC_{mn}$$

The pair (*n*,*m*) of medoid/non-medoid with the smallest impact on clustering quality

3) If $TC_{mn} < 0$, replace *m* by *n* and go back to 2
4) Assign every item to its nearest medoid

Swapping Cost

- For each pair of a medoid *m* and a non-medoid object *h*, measure whether *h* is better than *m* as a medoid
- For example, we can use the squared-error criterion

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} d(p, m_i)^2$$

  - Compute E_h-E_m
  - Negative: swapping brings benefit
- Choose the minimum swapping cost

K-medoids Example

X1	2	6
X2	3	4
X3	3	8
X4	5	7
X5	6	2
X6	6	4
X7	7	3
X8	7	4
X9	8	5
x10	7	6

Assume k=2
Select X5 and X9 as medoids

Distances	to X5	to X9
X1	8	7
X2	5	6
X3	9	8
X4	7	6
X6	2	3
X7	2	3
X8	3	2
x10	5	2

Current clustering: {X2,X5,X6,X7},{X1,X3,X4,X8,X9,X10}

K-medoids Example

X1	2	6
X2	3	4
X3	3	8
X4	5	7
X5	6	2
X6	6	4
X7	7	3
X8	7	4
X9	8	5
x10	7	6

- **So, now let us choose some other point to be a medoid instead of X5 (6, 2). Let us randomly choose X1 (2, 6).**

- **Not the new medoid set is: (2, 6) and (8, 5). Now repeating the same task as earlier:**

Replace X5 by X1	Before	to X1	To X9	Change
X1	7	0	0	-7
X2	5	3	6	-2
X3	8	3	8	-5
X4	6	4	6	-2
X5	0	8	5	5
X6	2	6	3	1
X7	2	8	3	1
X8	2	7	2	0
X9	0	0	0	0
x10	2	5	2	0
				-9

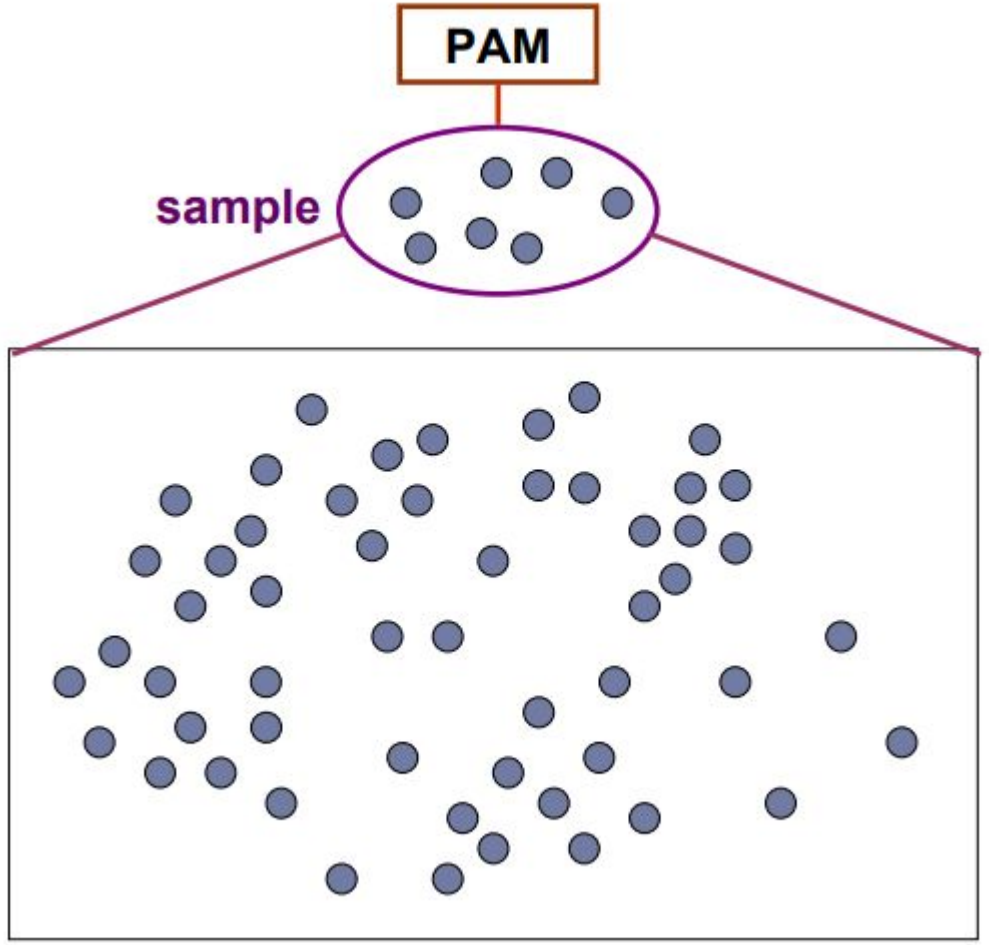
Current clustering: {**X1**,X2,X3,X4},{X5,X6,X7,X8,**X9**,X10}

K-medoids Properties

- The complexity of each iteration is $O(k(n-k)^2)$
- For large values of n and k, such computation becomes very costly
- Advantages
  - K-Medoids method is more robust than k-Means in the presence of noise and outliers
- Disadvantages
  - K-Medoids is more costly that the k-Means method
  - Like k-means, k-medoids requires the user to specify k
  - It does not scale well for large data sets

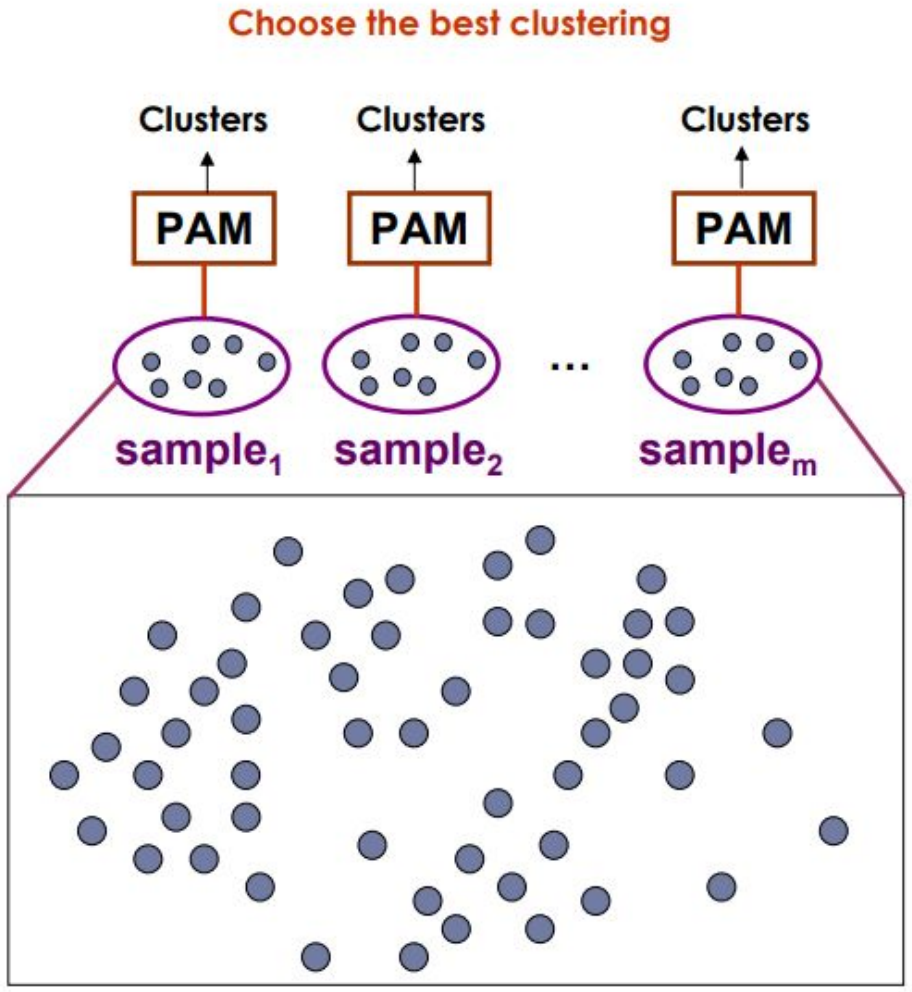
CLARA (Clustering Large Applications)

- CLARA (Clustering Large Applications) uses a sampling-based method to deal with large data sets
- A random sample should closely represent the original data
- The chosen medoids will likely be similar to what would have been chosen from the whole data set



CLARA (Clustering Large Applications)

- Draw multiple samples of the data set
- Apply PAM to each sample
- Return the best clustering



CLARA Properties

- Complexity of each Iteration (PAM on sample) is:

$$O(tk(s-k)^2 + k(n-k))$$

  - s: the size of the sample
  - k: number of clusters
  - n: number of objects
  - t: maximum iteration required for PAM
- PAM finds the best k medoids among a given data, and CLARA finds the best k medoids among the selected samples
- Problems
  - The best k medoids may not be selected during the sampling process, in this case, CLARA will never find the best clustering
  - If the sampling is biased we cannot have a good clustering

CLARA - Algorithm

- Set mincost to MAXIMUM;
- Repeat *q* times // draws *q* samples
  - Create *S* by drawing *s* objects randomly from *D*;
  - Generate the set of medoids *K* from *S* by applying the PAM algorithm;
  - Compute cost(K,D)
  - If cost(*K*, *D*)<mincost

    Mincost = cost(*K*, *D*);

    Bestset = *K*;
  - Endif;
- Endrepeat;
- Return Bestset;

Complexity of CLARA

- Set mincost to MAXIMUM; ⟶ $O(1)$
- Repeat *q* times ⟶ $O(t(s-k)^2*k+(n-k)*k)$
  - Create *S* by drawing *s* objects randomly from *D*; ⟶ $O(1)$
  - Generate the set of medoids *K* from *S* by applying the PAM algorithm; ⟶ $O(t(s-k)^2*k)$
  - Compute cost(K,D) ⟶ $O((n-k)*k)$
  - If cost(*K*, *D*)<mincost ⟶ $O(1)$

    Mincost = cost(*K*, *D*);

    Bestset = *K*;

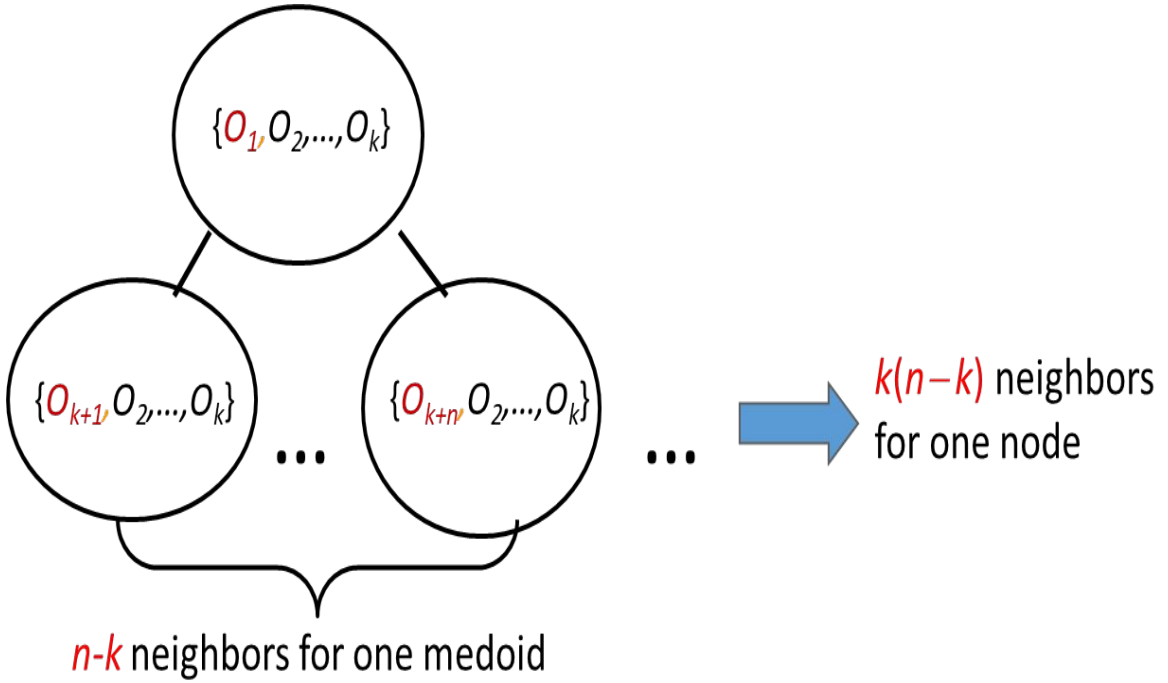
  Endif;
- Endrepeat;
- Return Bestset;

CLARANS ("Randomized" CLARA)

- CLARANS (A Clustering Algorithm based on Randomized Search)
- The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of k medoids
- Two nodes are neighbours if their sets differ by only one medoid
- Each node can be assigned a cost that is defined to be the total dissimilarity between every object and the medoid of its cluster
- The problem corresponds to search for a minimum on the graph
- At each step, all neighbours of current node node are searched; the neighbour which corresponds to the deepest descent in cost is chosen as the next solution

CLARANS ("Randomized" CLARA)

- CLARANS (A Clustering Algorithm based on Randomized Search)
- The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of k medoids
- Graph Abstraction
  - Every node is a potential solution (k-medoid)
  - Two nodes are adjacent if they differ by one medoid
  - Every node has $k(n-k)$ adjacent nodes

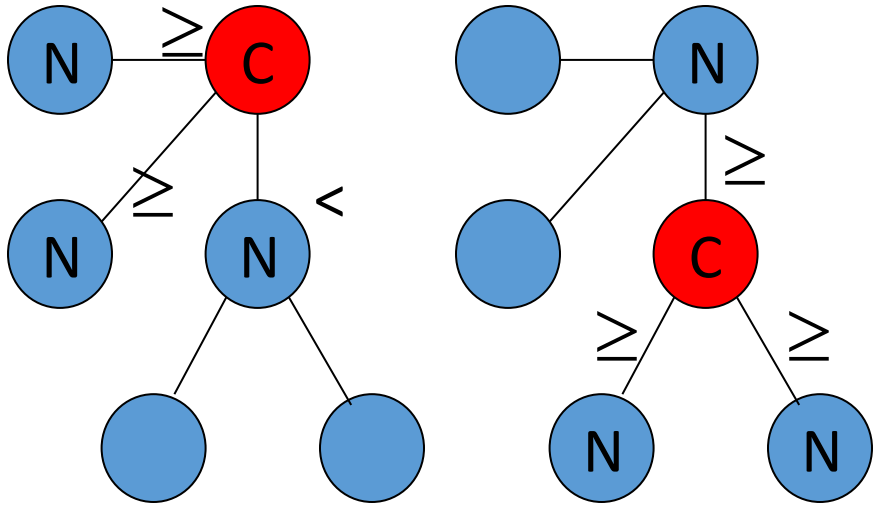


CLARANS ("Randomized" CLARA)

- For large values of n and k, examining k(n-k) neighbours is time consuming.
- At each step, CLARANS draws sample of neighbours to examine.
- Note that CLARA draws a sample of nodes at the beginning of search; therefore, CLARANS has the benefit of not confining the search to a restricted area.
- If the local optimum is found, CLARANS starts with a new randomly selected node in search for a new local optimum. The number of local optimums to search for is a parameter.
- It is more efficient and scalable than both PAM and CLARA; returns higher quality clusters.

CLARANS

Compare no more than *maxneighbor* times



... → Local minimum

... → Local minimum

... → Local minimum

... → Local minimum

numlocal

Best Node

CLARANS - Algorithm

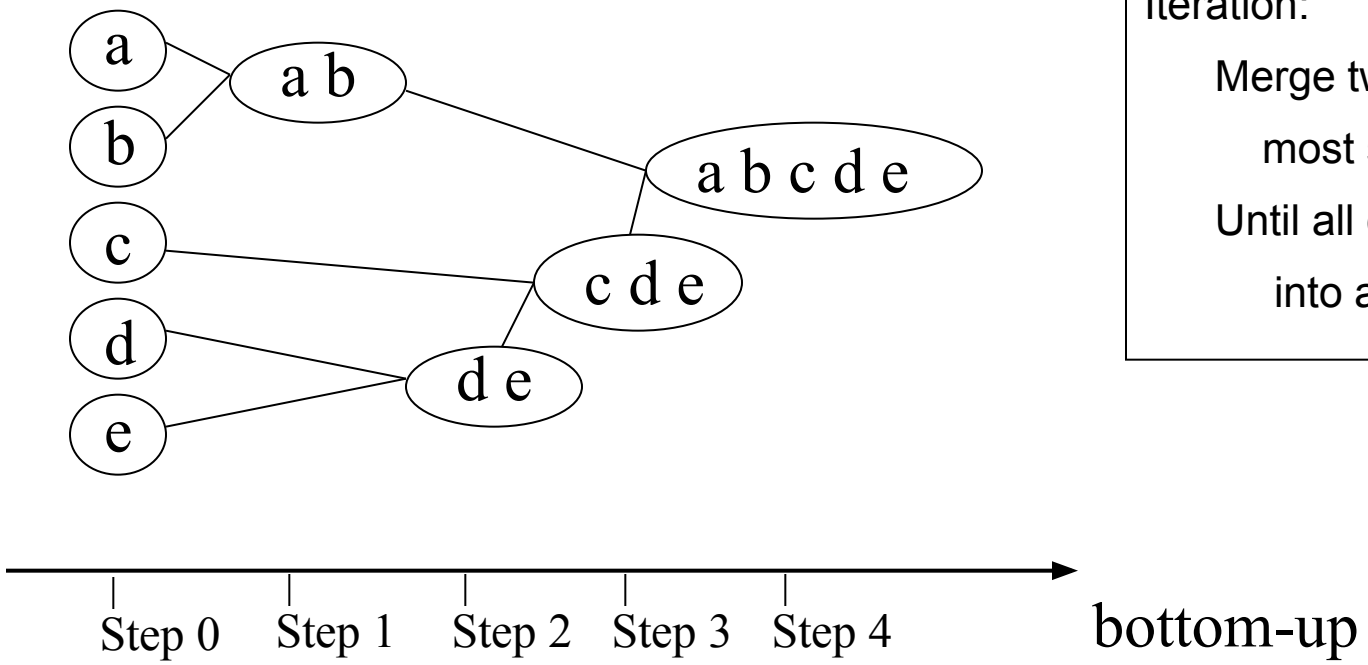
- Set mincost to MAXIMUM;
- For i=1 to *h* do // find h local optimum
  - Randomly select a node as the current node C in the graph;
  - J = 1; // counter of neighbors
  - Repeat
    Randomly select a neighbor N of C;
    If Cost(N,D)<Cost(C,D)
      Assign N as the current node C;
      J = 1;
    Else J++;
    Endif;
  - Until J > m
  - Update mincost with Cost(C,D) if applicableEnd for;
- End For
- Return bestnode;

Hierarchical Clustering

- Hierarchical Clustering Approach
  - A typical clustering analysis approach via partitioning data set sequentially
  - Construct nested partitions layer by layer via grouping objects into a tree of clusters (without the need to know the number of clusters in advance)
  - Use (generalised) distance matrix as clustering criteria
- Agglomerative vs Divisive
  - Two sequential clustering strategies for constructing a tree of clusters
  - Agglomerative: a bottom-up strategy
    - Initially each data object is in its own (atomic) cluster
    - Then merge these atomic clusters into larger and larger clusters
  - Divisive: a top-down strategy
    - Initially all objects are in one single cluster
    - Then the cluster is subdivided into smaller and smaller clusters

Hierarchical Clustering

- **Agglomerative approach**



Initialization:

Each object is a cluster

Iteration:

Merge two clusters which are most similar to each other;

Until all objects are merged into a single cluster

Hierarchical Clustering

- **Divisive Approaches**

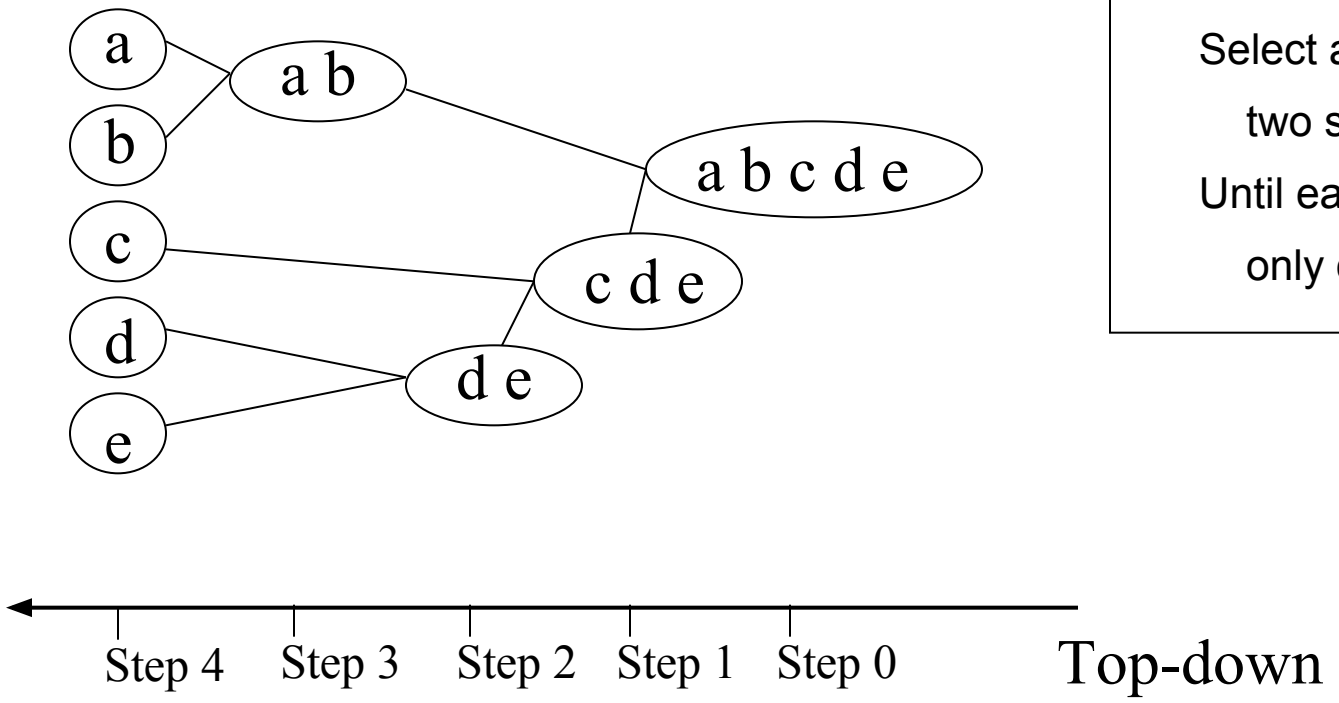
Initialization:

All objects stay in one cluster

Iteration:

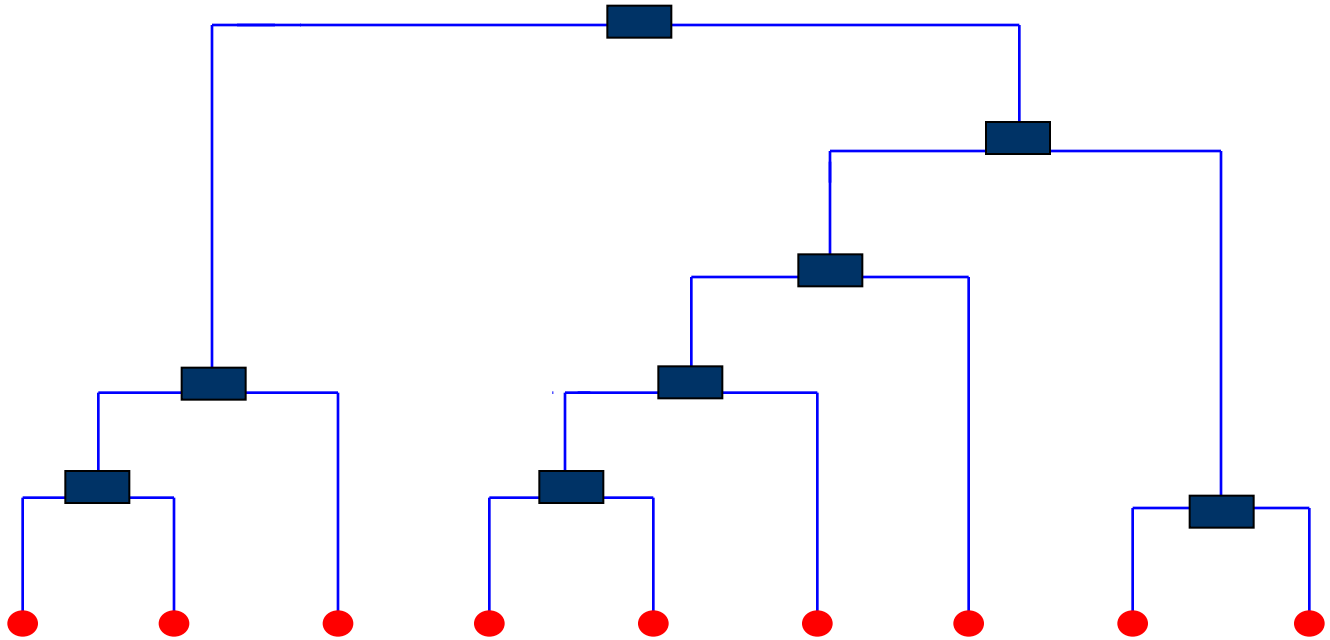
Select a cluster and split it into two sub clusters

Until each leaf cluster contains only one object



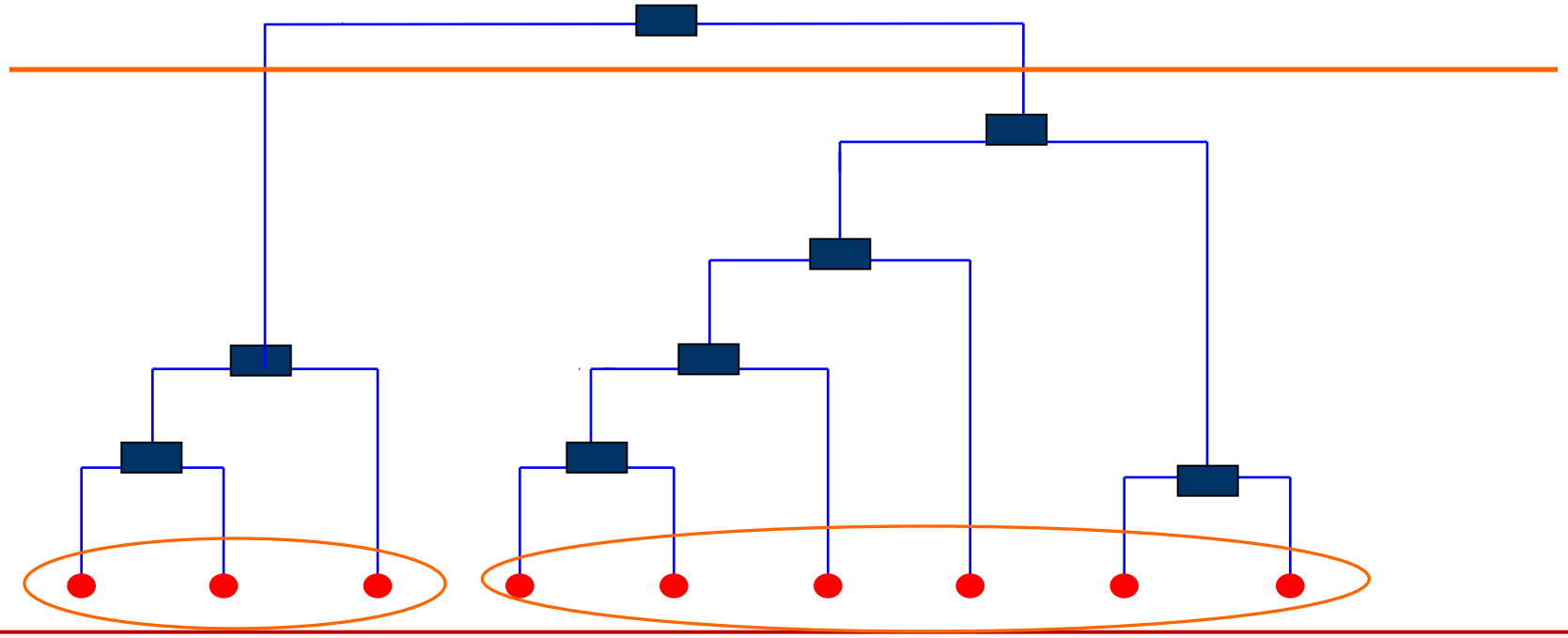
Dendrogram

- A binary tree that shows how clusters are merged/split hierarchically
- Each node on the tree is a cluster; each leaf node is a singleton cluster



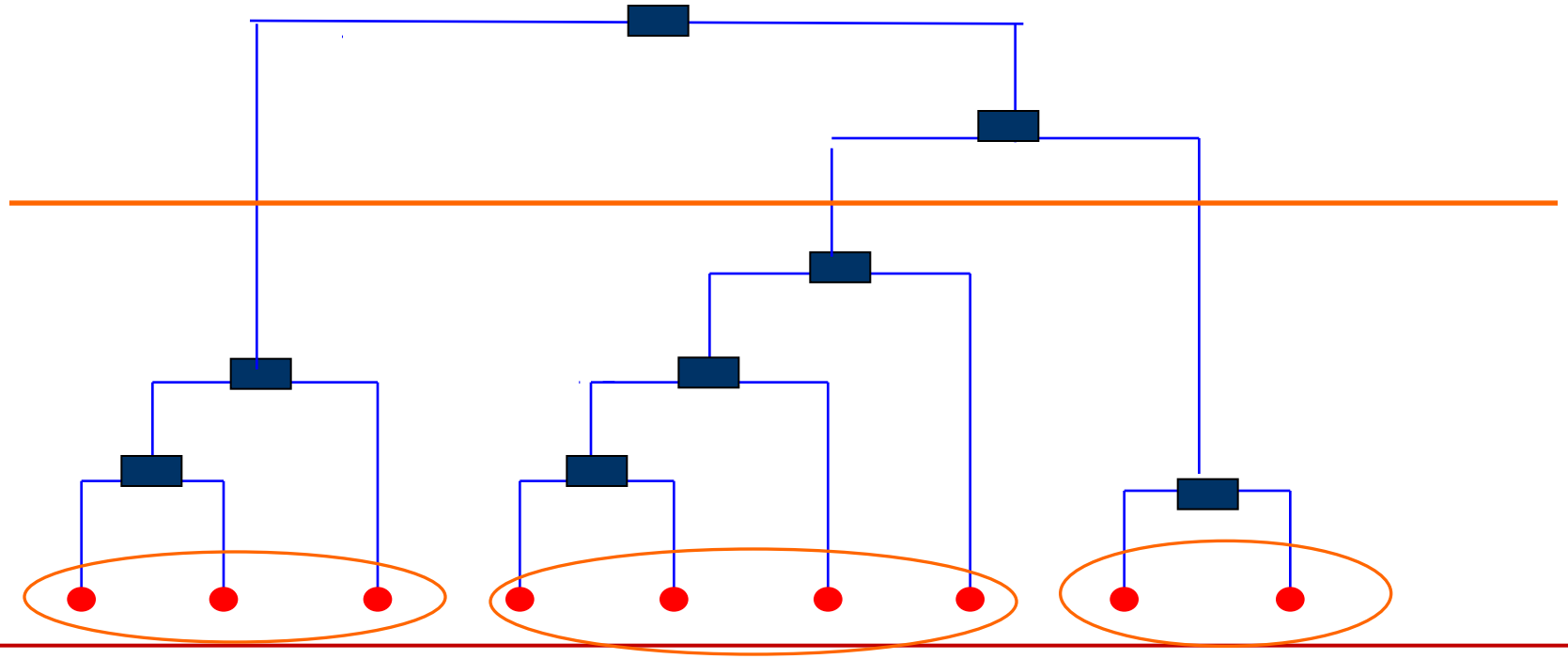
Dendrogram

- A clustering of the data objects is obtained by cutting the *dendrogram* at the desired level, then each connected component forms a cluster



Dendrogram

- A clustering of the data objects is obtained by cutting the *dendrogram* at the desired level, then each connected component forms a cluster



How to Merge Clusters?

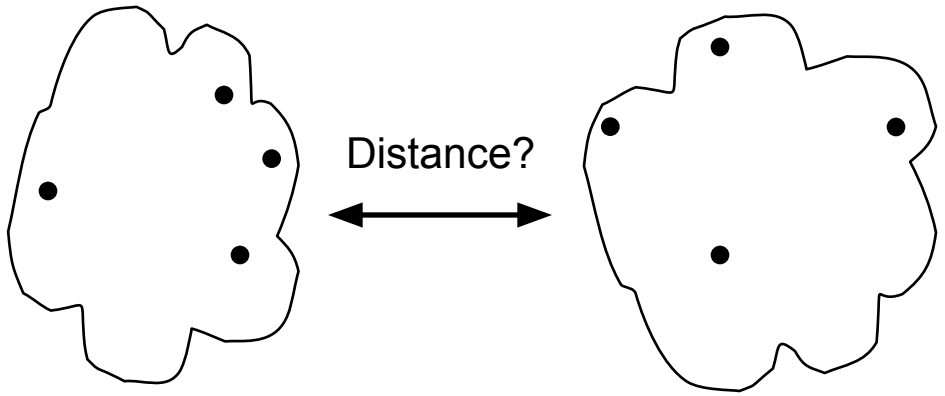
- How to measure the distance between clusters?

Single-link

Complete-link

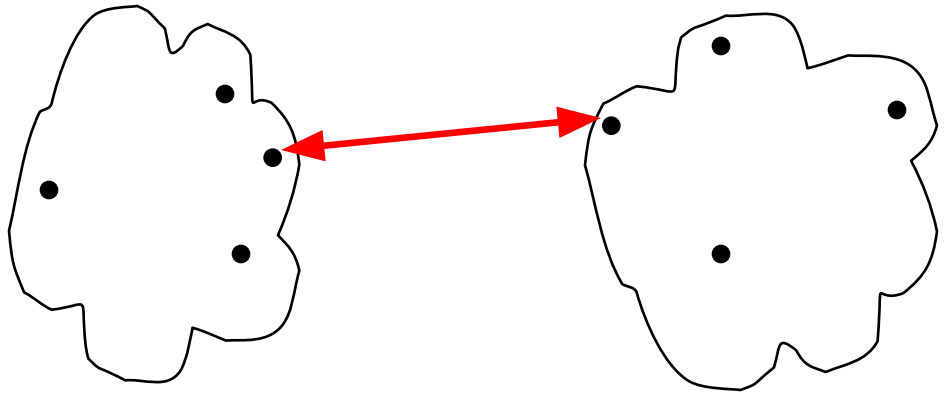
Average-link

Centroid distance



Hint: *Distance between clusters* is usually defined on the basis of *distance between objects.*

How to Define Inter-Cluster Distance



Single-link

Complete-link

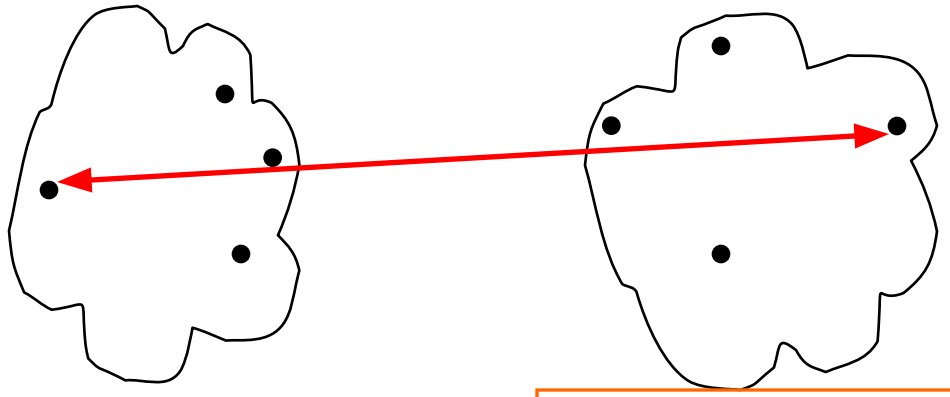
Average-link

Centroid distance

$$d_{\min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)$$

The distance between two clusters is represented by the distance of the *closest pair of data objects* belonging to different clusters.

How to Define Inter-Cluster Distance



Single-link

Complete-link

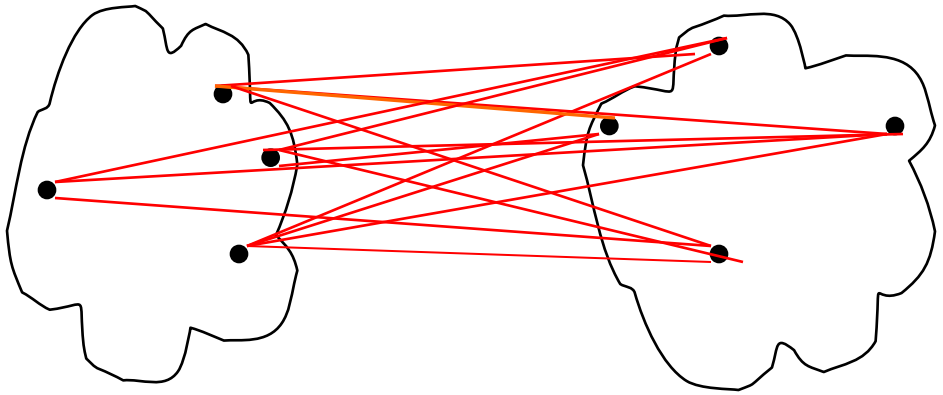
Average-link

Centroid distance

$$d_{\min}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)$$

The distance between two clusters is represented by the distance of the *farthest pair of data objects* belonging to different clusters.

How to Define Inter-Cluster Distance



Single-link

Complete-link

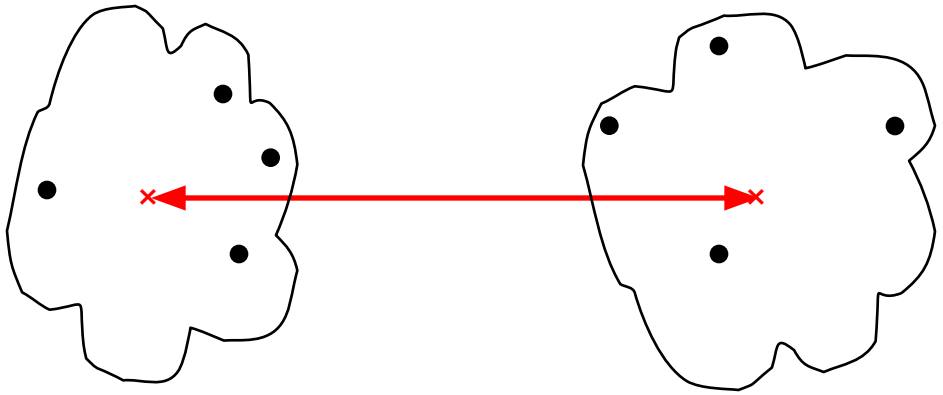
Average-link

Centroid distance

$$d_{\min}(C_i, C_j) = \underset{p \in C_i, q \in C_j}{avg} d(p, q)$$

The distance between two clusters is represented by the *average* distance of *all pairs of data objects* belonging to different clusters.

How to Define Inter-Cluster Distance



m_i,m_j are the means of C_i, C_j,

Single-link

Complete-link

Average-link

Centroid distance

$$d_{mean}(C_i, C_j) = d(m_i, m_j)$$

The distance between two clusters is represented by the distance between *the means of the cluters.*

Cluster Distance Measures

Example: Given a data set of five objects characterized by a single continuous feature, assume that there are two clusters: C_1: {a, b} and C_2: {c, d, e}.

a	b	c	d	e
1	2	4	5	6

1. Calculate the distance matrix.

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

2. Calculate three cluster distances between C_1 and C_2.

Single link

$$\text{dist}(C_1, C_2) = \min\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\}$$
$$= \min\{3, 4, 5, 2, 3, 4\} = 2$$

Complete link

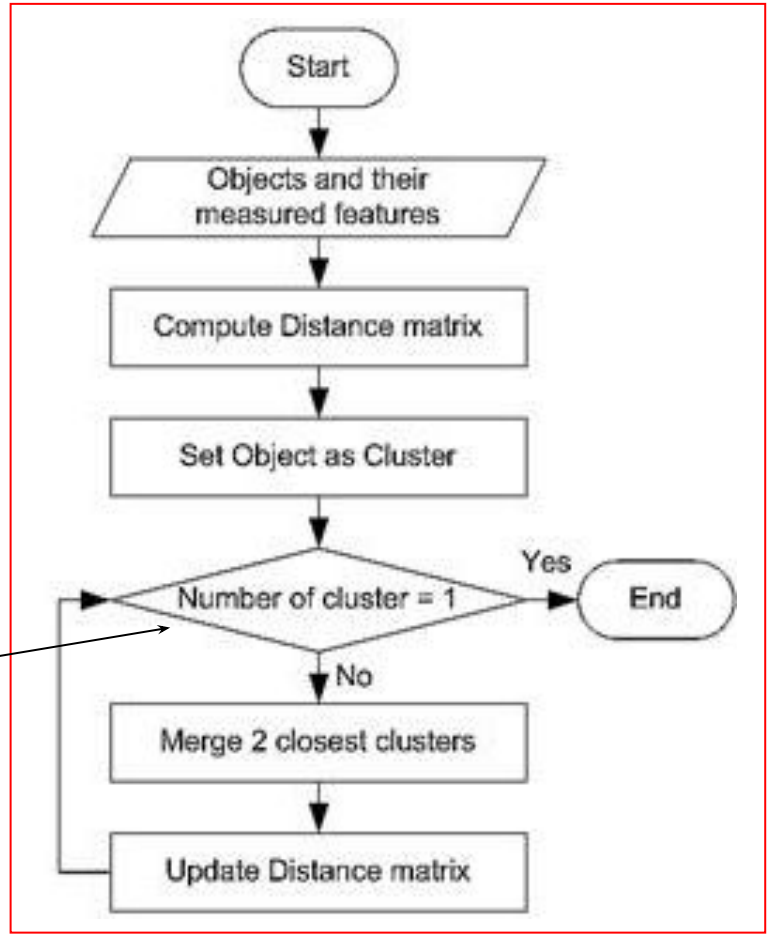
$$\text{dist}(C_1, C_2) = \max\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\}$$
$$= \max\{3, 4, 5, 2, 3, 4\} = 5$$

Average

$$\text{dist}(C_1, C_2) = \frac{d(a,c) + d(a,d) + d(a,e) + d(b,c) + d(b,d) + d(b,e)}{6}$$
$$= \frac{3+4+5+2+3+4}{6} = \frac{21}{6} = 3.5$$

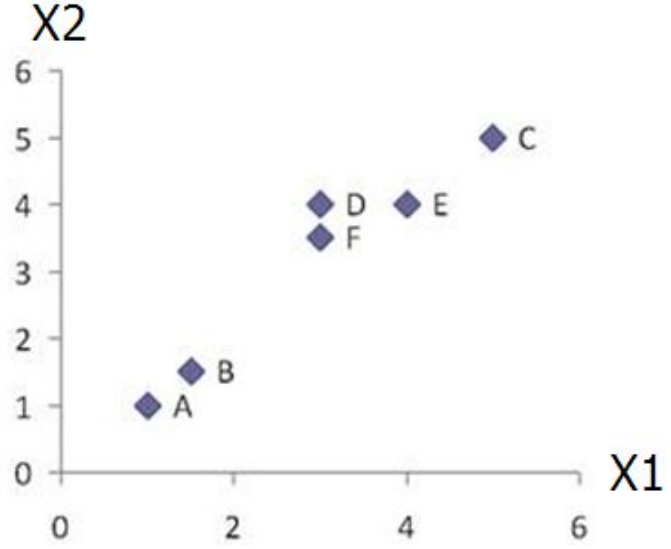
Agglomerative Algorithm

- **The *Agglomerative* algorithm is carried out in three steps:**
  1) Convert all object features into a distance matrix
  2) Set each object as a cluster (thus if we have *N* objects, we will have *N* clusters at the beginning)
  3) Repeat until number of cluster is one (or known # of clusters)
     - Merge two closest clusters
     - Update "distance matrix"



Example

- **Problem: clustering analysis with agglomerative algorithm**



	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

data matrix

$$d_{AB} = \left((1-1.5)^2 + (1-1.5)^2\right)^{\frac{1}{2}} = \sqrt{\tfrac{1}{2}} = 0.7071$$

$$d_{DF} = \left((3-3)^2 + (4-3.5)^2\right)^{\frac{1}{2}} = 0.5$$

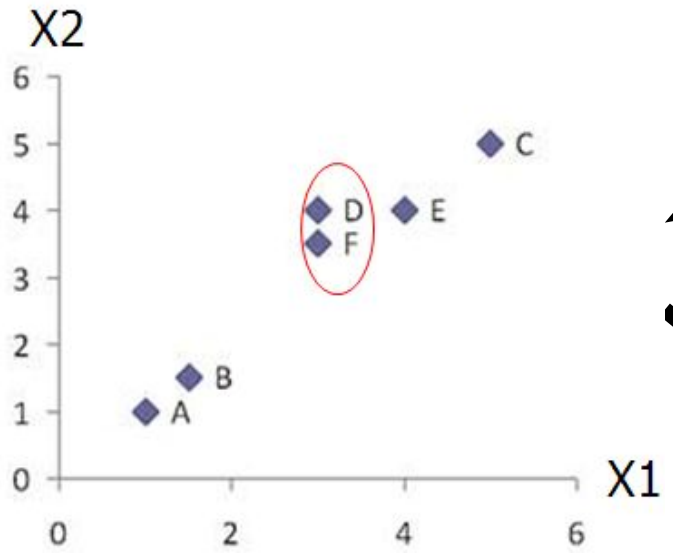
Euclidean distance

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

distance matrix

Example

- Merge two closest clusters (iteration 1)



Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

Example

- **Update distance matrix (iteration 1)**

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

$$d_{(D,F)\to A} = \min\left(d_{DA}, d_{FA}\right) = \min\left(3.61, 3.20\right) = 3.20$$

$$d_{(D,F)\to B} = \min\left(d_{DB}, d_{FB}\right) = \min\left(2.92, 2.50\right) = 2.50$$

$$d_{(D,F)\to C} = \min\left(d_{DC}, d_{FC}\right) = \min\left(2.24, 2.50\right) = 2.24$$

$$d_{E\to(D,F)} = \min\left(d_{ED}, d_{EF}\right) = \min\left(1.00, 1.12\right) = 1.00$$

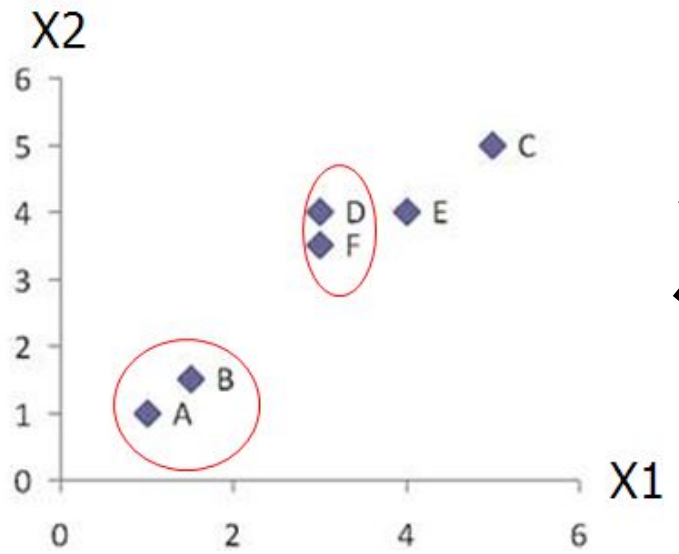
Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

Example

- Merge two closest clusters (iteration 2)



Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Example

- **Update distance matrix (iteration 2)**

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

$$d_{C\to(A,B)} = \min\left(d_{CA}, d_{CB}\right) = \min\left(5.66, 4.95\right) = 4.95$$

$$d_{(D,F)\to(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}\right) = \min\left(3.61, 2.92, 3.20, 2.50\right) = 2.50$$

$$d_{E\to(A,B)} = \min\left(d_{EA}, d_{EB}\right) = \min\left(4.24, 3.54\right) = 3.54$$

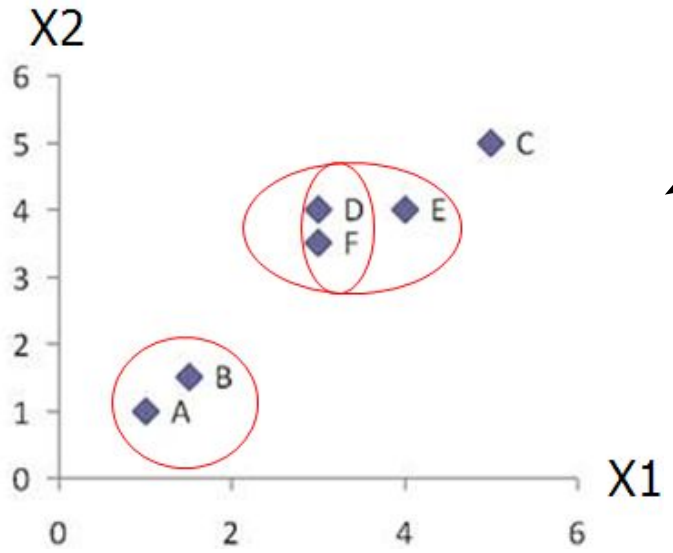
Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	**4.95**	0	2.24	1.41
(D, F)	**2.50**	2.24	0	1.00
E	**3.54**	1.41	1.00	0

Example

- Merge two closest clusters/update distance matrix (iteration 3)



Min Distance (Single Linkage)

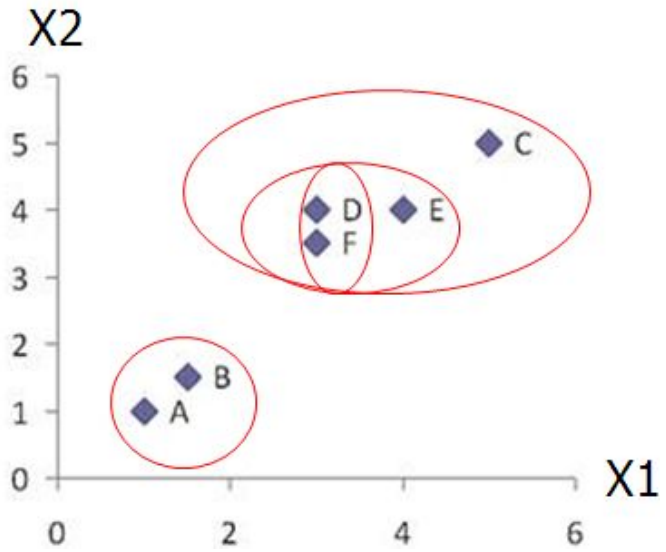
Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	**4.95**	0	2.24	1.41
(D, F)	**2.50**	2.24	0	1.00
E	**3.54**	1.41	1.00	0

Min Distance (Single Linkage)

Dist	(A,B)	C	(D, F), E
(A,B)	0.00	4.95	2.50
C	4.95	0.00	1.41
(D, F), E	**2.50**	**1.41**	0.00

Example

- Merge two closest clusters/update distance matrix (iteration 4)



Min Distance (Single Linkage)

Dist	(A,B)	C	(D, F), E
(A,B)	0.00	4.95	2.50
C	4.95	0.00	1.41
(D, F), E	**2.50**	**1.41**	0.00

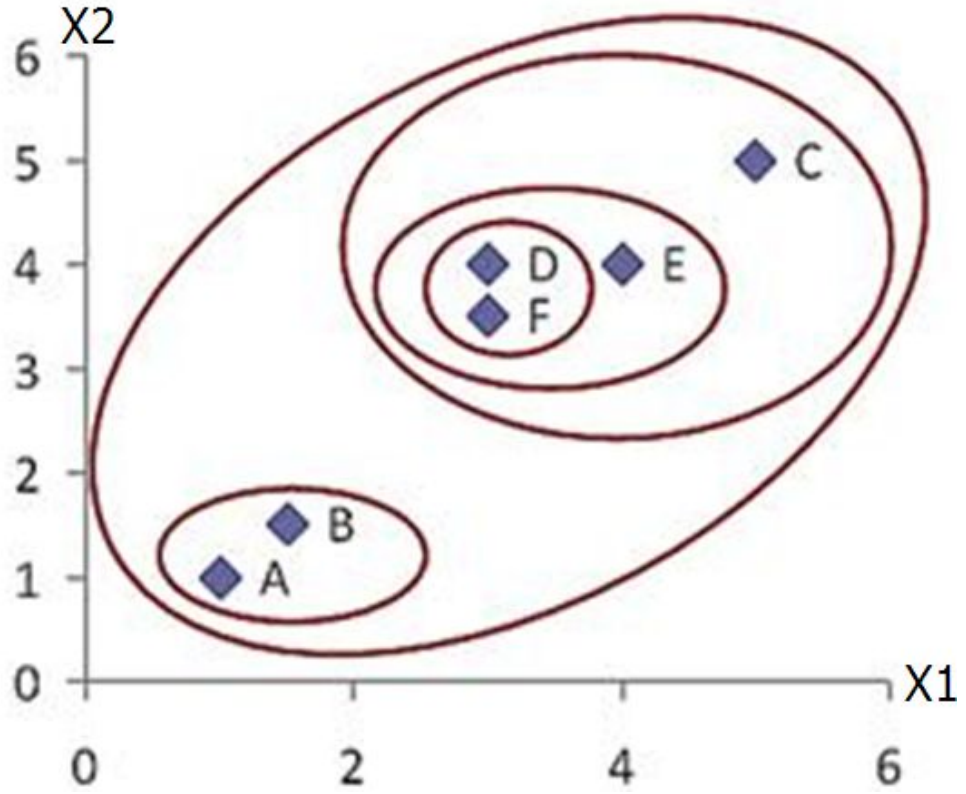
Min Distance (Single Linkage)

Dist	(A,B)	((D, F), E),C
(A,B)	0.00	2.50
((D, F), E),C	**2.50**	0.00

Example

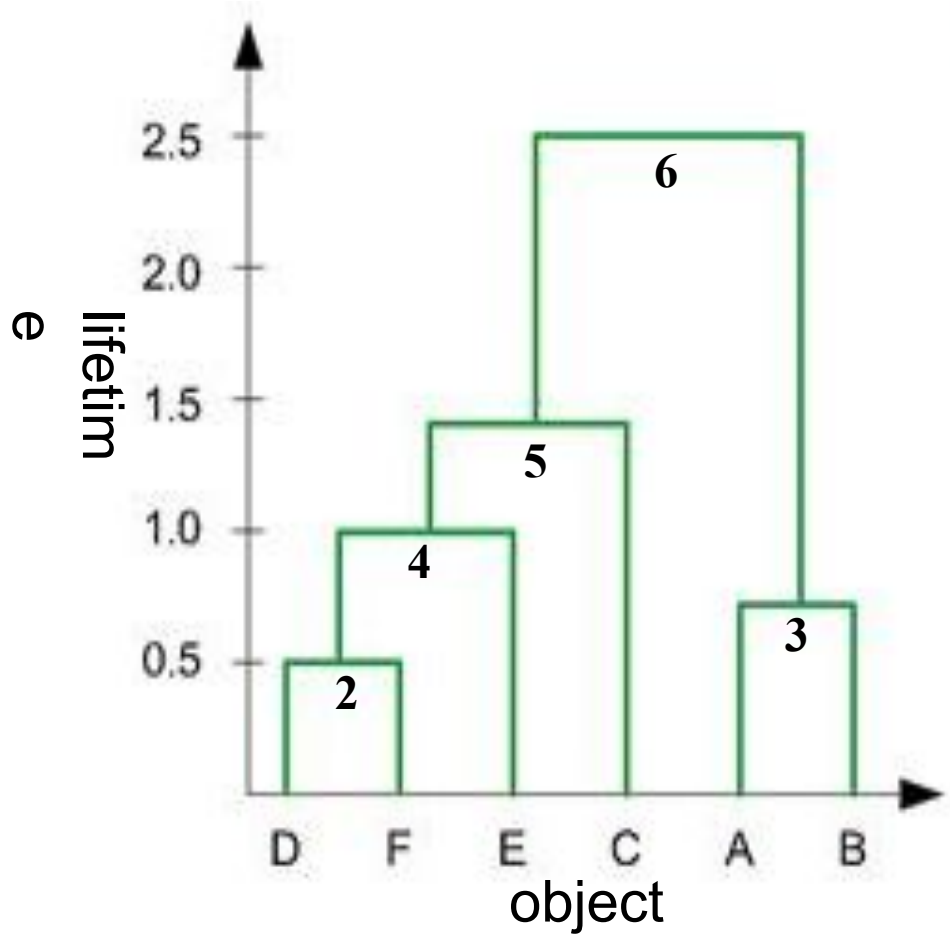
- **Final result (meeting termination condition)**

	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5



Example

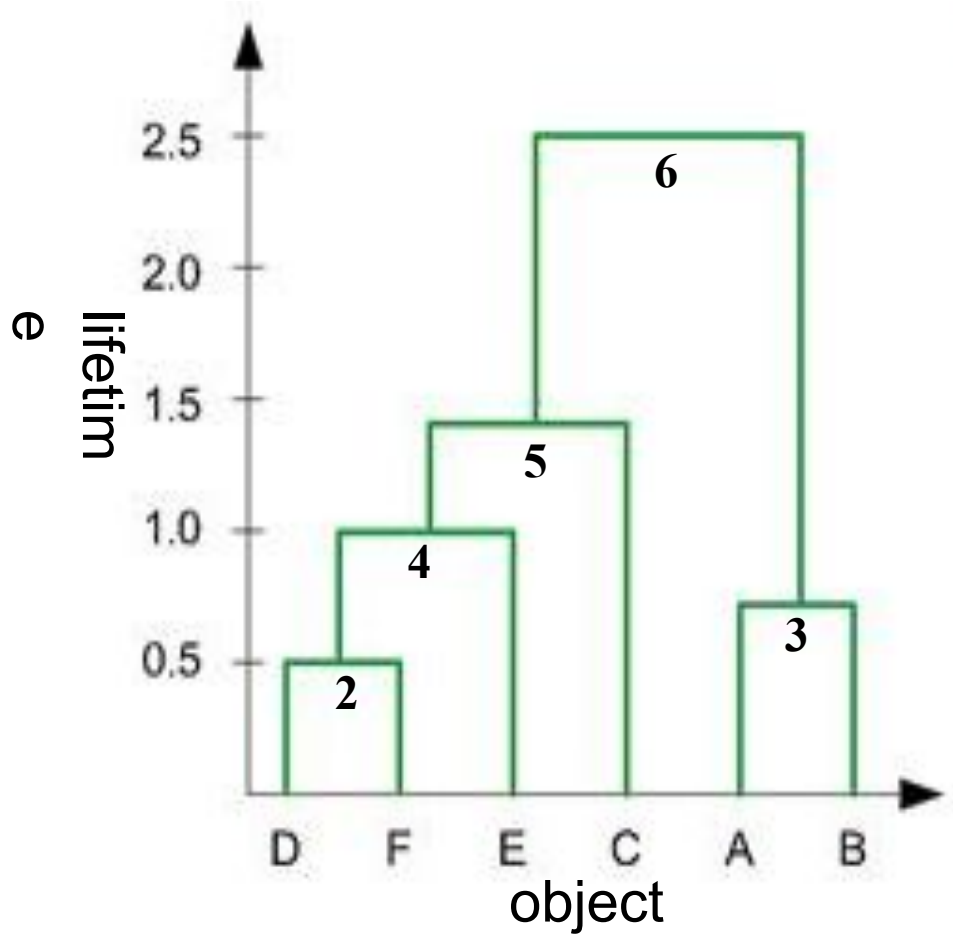
- **Dendrogram tree representation**



1. In the beginning we have 6 clusters: A, B, C, D, E and F
2. We merge clusters D and F into cluster (D, F) at distance 0.50
3. We merge cluster A and cluster B into (A, B) at distance 0.71
4. We merge clusters E and (D, F) into ((D, F), E) at distance 1.00
5. We merge clusters ((D, F), E) and C into (((D, F), E), C) at distance 1.41
6. We merge clusters (((D, F), E), C) and (A, B) into ((((D, F), E), C), (A, B)) at distance 2.50
7. The last cluster contain all the objects, thus conclude the computation

Example

- **Dendrogram tree representation**



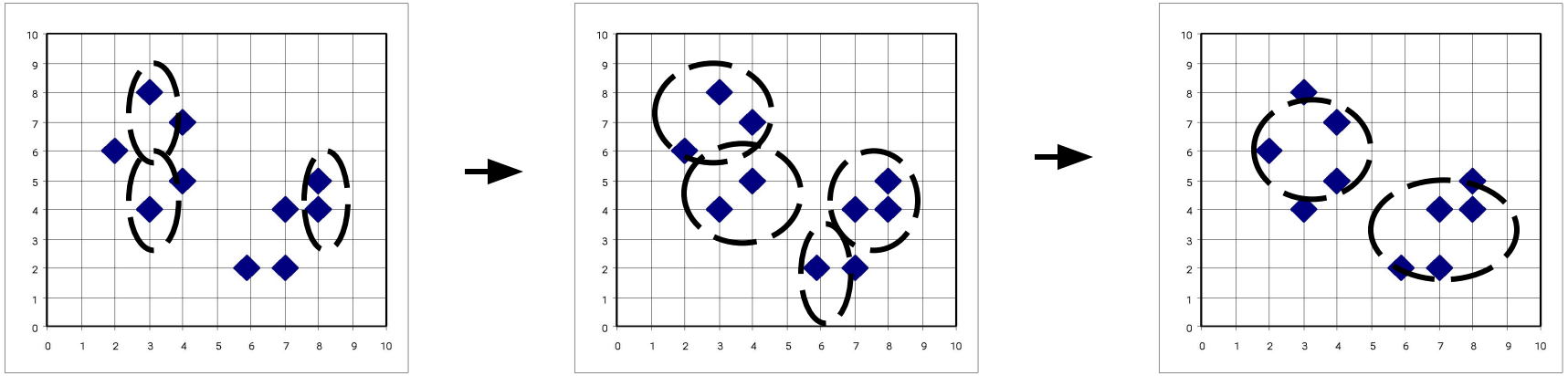
- For a dendrogram tree, its horizontal axis indexes all objects in a given data set, while its vertical axis expresses the lifetime of all possible cluster formation.
- The lifetime of a cluster (individual cluster) in the dendrogram is defined as a distance interval from the moment that the cluster is created to the moment that it disappears by merging with other clusters.

Which Distance Measure is Better?

- Each method has both advantages and disadvantages; application-dependent, single-link and complete-link are the most common methods
- Single-link
  - Can find irregular-shaped clusters
  - Sensitive to outliers, suffers the so-called chaining effects
    - In order to merge two groups, only need one pair of points to be close, irrespective of all others. Therefore clusters can be too spread out, and not compact enough
- Average-link, and Centroid distance
  - Robust to outliers
  - Tend to break large clusters

AGNES

- AGNES : Agglomerative Nesting
- Use *single-link method*
- Merge nodes that have *the least* dissimilarity
- Eventually all objects belong to the same cluster

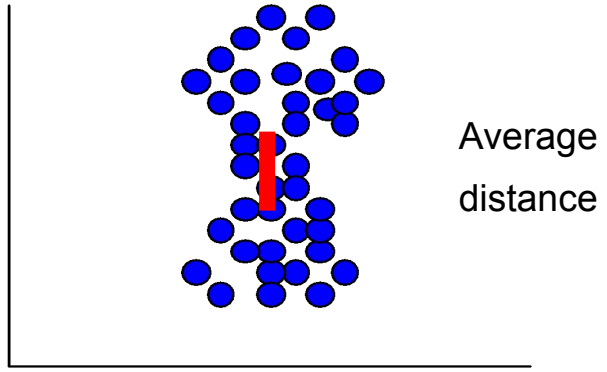


UPGMA

- UPGMA: Un-weighted Pair-Group Method Average.
- Merge Strategy:
  - Average-link approach
  - The distance between two clusters is measured by the average distance between two objects belonging to different clusters.

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{q \in C_j} d(p, q)$$

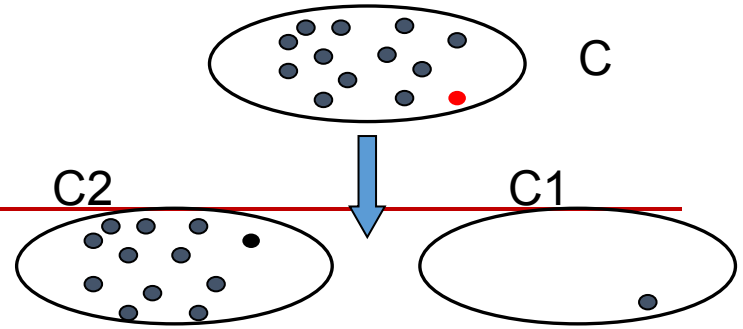
n_i,n_j: the number of objects in cluster C_i, C_j.



DIANA

- DIANA: Divisive Analysis
- First, all of the objects form one cluster.
- The cluster is split according to some principle, such as the minimum Euclidean distance between the closest neighboring objects in the cluster.
- The cluster splitting process repeats until, eventually, each new cluster contains a single object, or a termination condition is met.

Splitting Process of DIANA

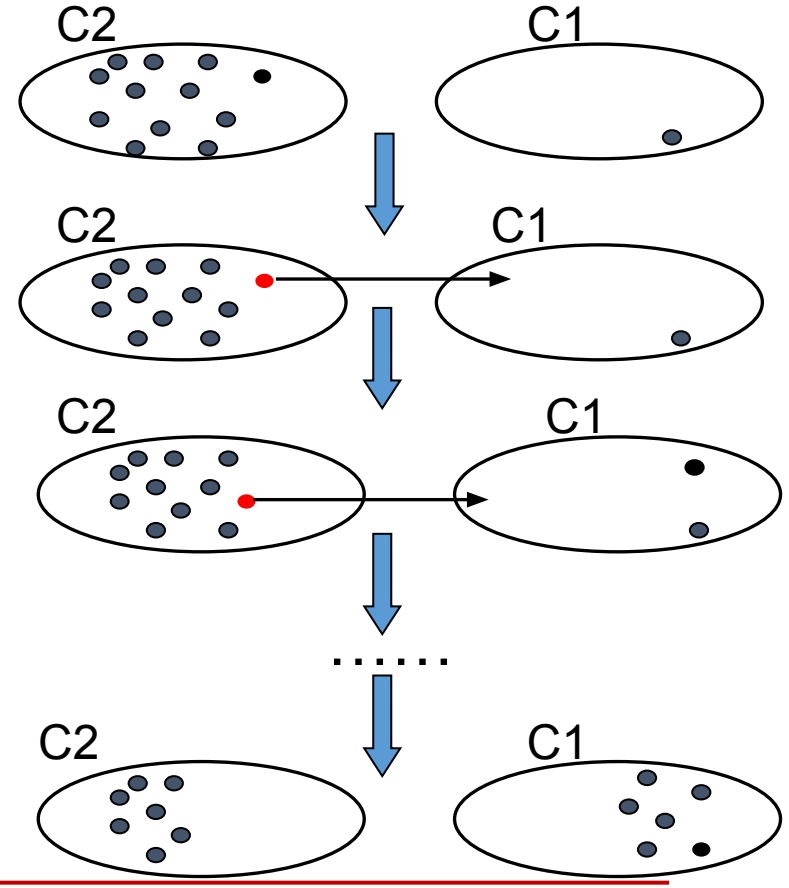


Initialization:

1. Choose the object O_h which is most dissimilar to other objects in C.
2. Let C1={O_h}, C2=C-C1.
3. For each object Oi in C2, tell whether it is more close to C1 or to other objects in C2

$$D_i = \underset{j \in C_2}{avg}\, d(O_i, O_j) - \underset{j \in C_1}{avg}\, d(O_i, O_j)$$

4. Choose the object O_k with greatest D score.
5. If D_k>0, move Ok from C2 to C1, and repeat 3-5.
6. Otherwise, stop splitting process.



Balanced Iterative Reducing and Clustering Using Hierarchies (BIRCH)

- Agglomerative Clustering designed for clustering a large amount of numerical data
- It introduces two concepts :
  - Clustering feature
  - Clustering feature tree (CF tree)
- These structures help the clustering method achieve good speed and scalability in large databases.

- What Birch algorithm tries to solve?
  - Most of the existing algorithms DO NOT consider the case that datasets can be too large to fit in main memory
  - They DO NOT concentrate on minimizing the number of scans of the dataset
  - I/O costs are very high
- The complexity of BIRCH is O(n) where n is the number of objects to be clustered.

BIRCH: Key Components

- Clustering Feature (CF)
  - Summary of the statistics for a given cluster: the 0-th, 1st and 2nd moments of the cluster from the statistical point of view
  - A CF entry has sufficient information to calculate the centroid, radius, diameter and many other distance measures
  - Additively theorem allows us to merge sub-clusters incrementally
- CF-Tree
  - height-balanced tree
  - two parameters:
    - number of entries in each node
    - The diameter of all entries in a leaf node
  - Leaf nodes are connected via prev and next pointers

Clustering Feature

- Clustering Feature (CF): CF = (N, LS, SS)
  - N: Number of data points
  - LS: linear sum of N points: $\sum_{i=1}^{N} X_i$
  - SS: square sum of N points: $\sum_{i=1}^{N} X_i^2$

Distance Measures

- Given a cluster with data points ,
  - Centroid:

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

  - Radius: average distance from member points to centroid

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

  - Diameter: average pair-wise distance within a cluster

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

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

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

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

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

$$= \left(\frac{SS - 2\frac{LS}{N}\cdot LS + N(\frac{LS}{N})^2}{N}\right)^{1/2}$$

Clustering Feature

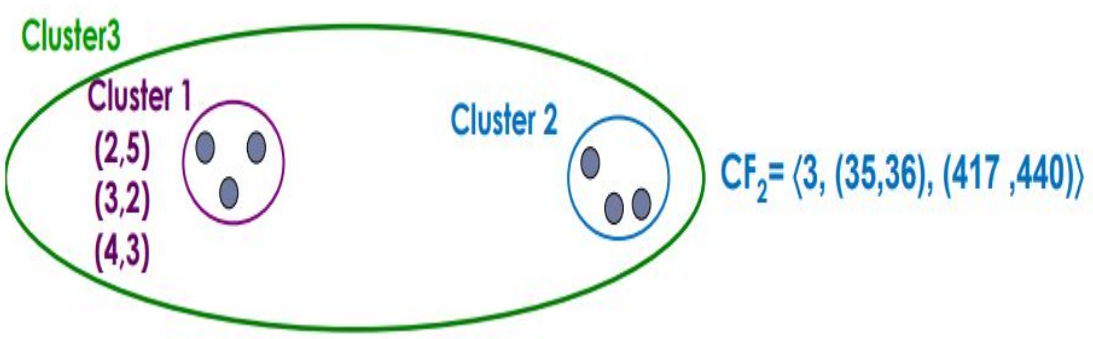
- Suppose cluster C_1 has $CF_1=(N_1, LS_1, SS_1)$, cluster C_2 has $CF_2=(N_2, LS_2, SS_2)$
- If we merge C_1 with C_2, the CF for the merged cluster C is

$$CF = CF_1 + CF_2$$

$$= (N_1 + N_2, \overrightarrow{LS_1} + \overrightarrow{LS_2}, \overrightarrow{SS_1} + \overrightarrow{SS_2})$$

- Why CF?
  - Summarized info for single cluster
  - Summarized info for two clusters
  - Additive theorem

CF1= $\langle 3, (2+3+4, 5+2+3), (2^2+3^2+4^2, 5^2+2^2+3^2) \rangle$
= $\langle 3, (9,10), (29, 38) \rangle$

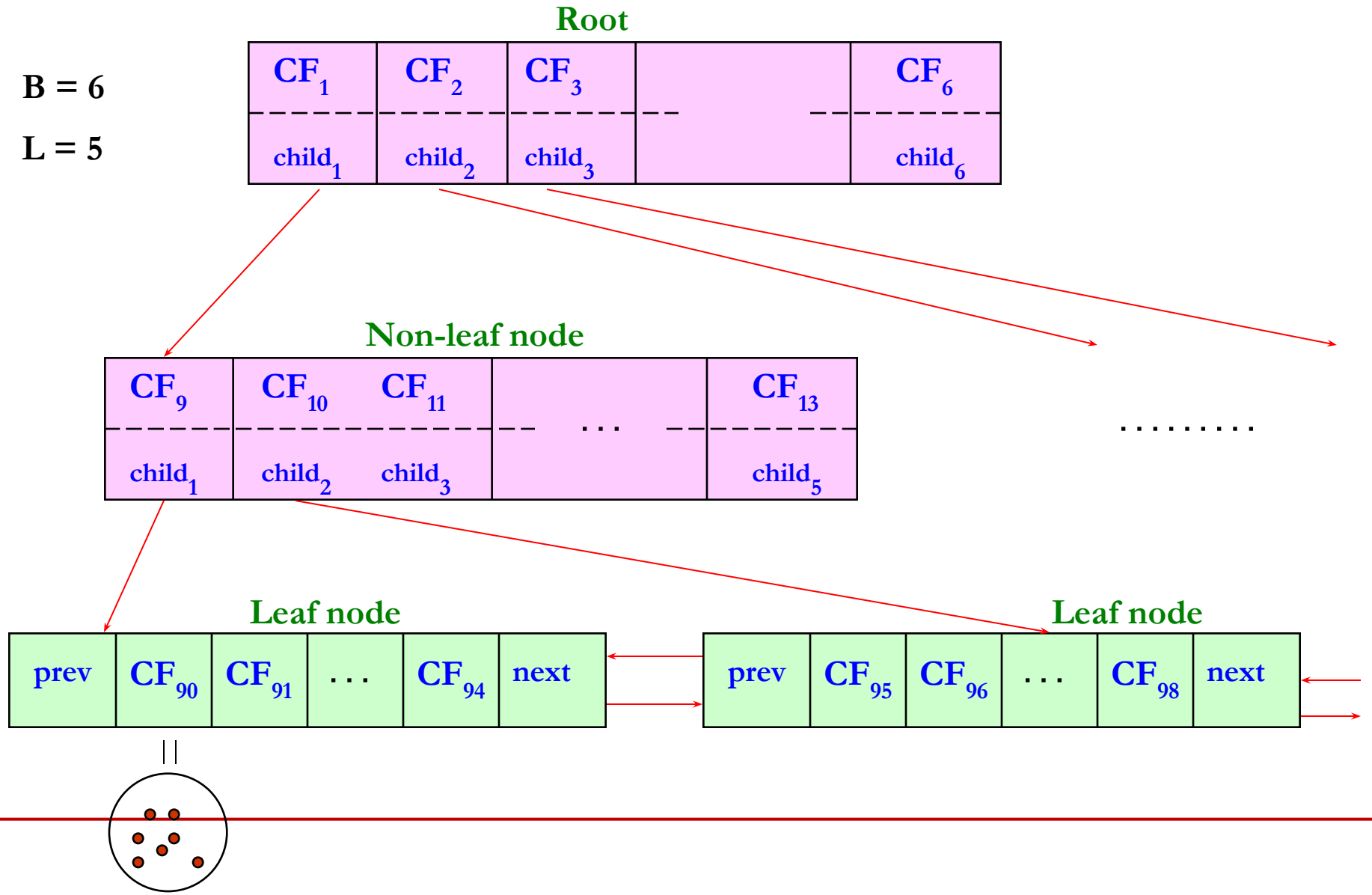


CF3=CF1+CF2= $\langle 3+3, (9+35, 10+36), (29+417, 38+440) \rangle$
= $\langle 6, (44,46), (446, 478) \rangle$

CF-Tree

- A CF tree is a height-balanced tree with two parameters: branching factor B and threshold T.
  - Branching factor
    - B : Each non-leaf node contains at most B entries of the form [CF_i, $child_i$] , where I = 1, 2, …, B, and $child_i$ is a pointer to its i-th child node. CF_i Is the CF of the sub-cluster represented by the $child_i$.
    - L : A leaf node contains at most L entries, each of the form [CF_i], where I = 1, 2, …, L. In addition, each leaf node has two pointers, "prev" and "next" which are used to chain all leaf node together
    - A leaf node also represent a cluster made up of all the sub-clusters represented by its entries.
  - Threshold T
    - The diameter (alternatively, the radius) of all entries in a leaf node is at most T
- Leaf nodes are connected via *prev* and *next* pointers

Example of CF Tree



CF Tree Insertion

- Given entry "Ent", it proceeds as below:
  - Identifying the appropriate leaf
    - recursively descending the CF tree and choosing the closest child node according to a chosen distance metric
  - Modifying the leaf:
    - When it reaches a leaf node, it finds the closest leaf entry, say L_i, and then test whether L_i can absorb the node without violating the threshold. If so, the CF vector for L_i is updated to reflect this.
    - If not, a new entry for "Ent" is added to the leaf. If there is no room, split the node
  - Modifying the path:
    - update CF information up the path.

Example

0.5

0.25

0

0.65

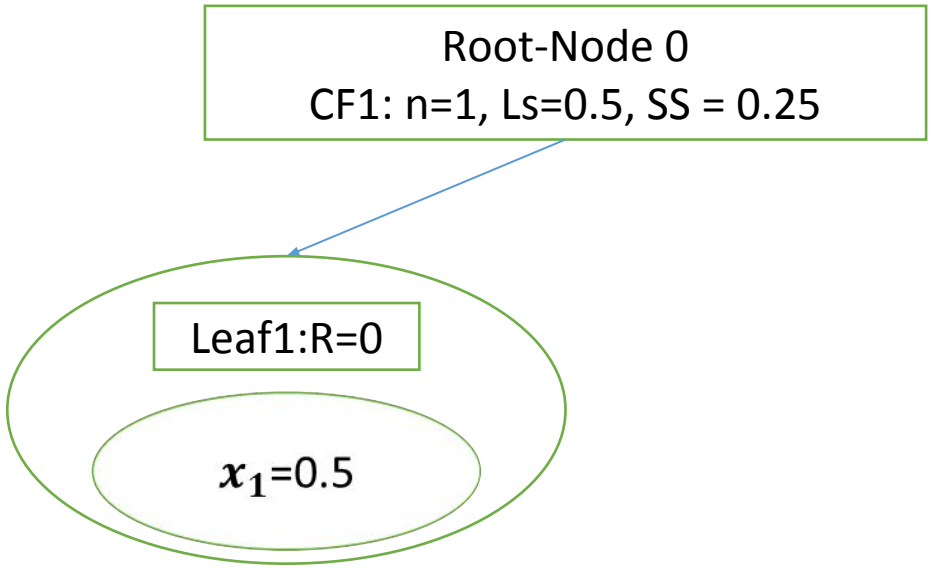
1

1.4

1.1

- T = 0.15
- L = 2
- B = 2

- Enter the first data x_1
- The root node is initialized with the CF values of the first data value
- A new leaf Leaf1 is created, and BIRCH assigns the first record x_1 to Leaf1
- Leaf1 contains only one record and hence the radius is zero and thus less than T=0.15



Example

0.5

0.25

0

0.65

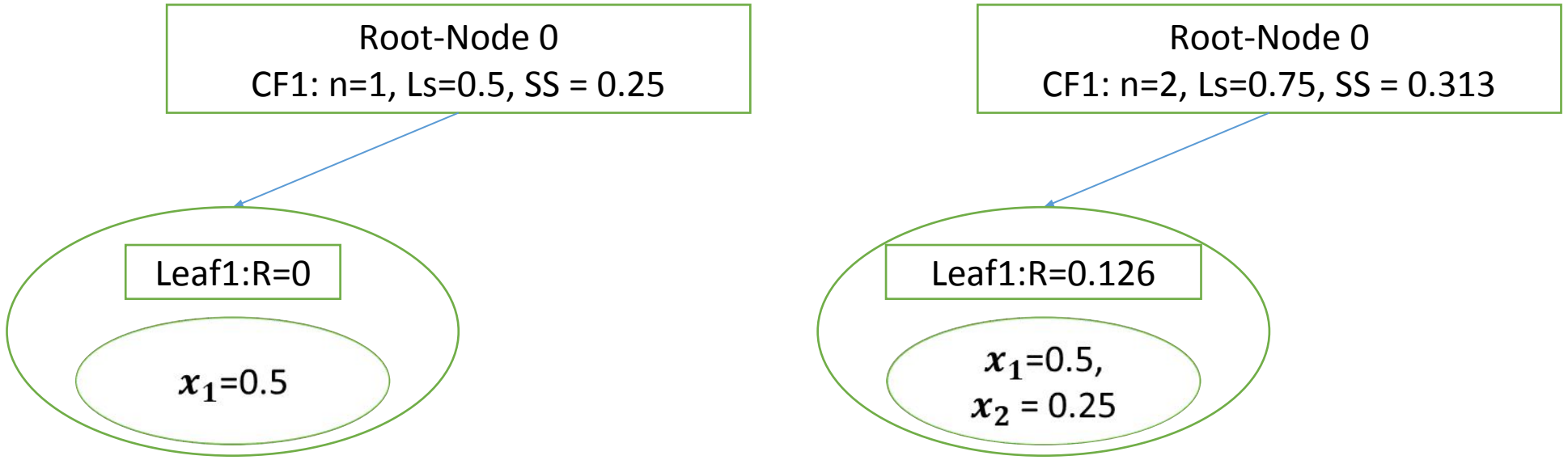
1

1.4

1.1

- T = 0.15
- L = 2
- B = 2

- The second data value x_2 =0.25 is entered.
- BIRCH tentatively passes x_2 =0.25 to Leaf1.
- The radius of Leaf1 is now R = 0.126 < T = 0.15, so x_2 is assigned to Leaf1.

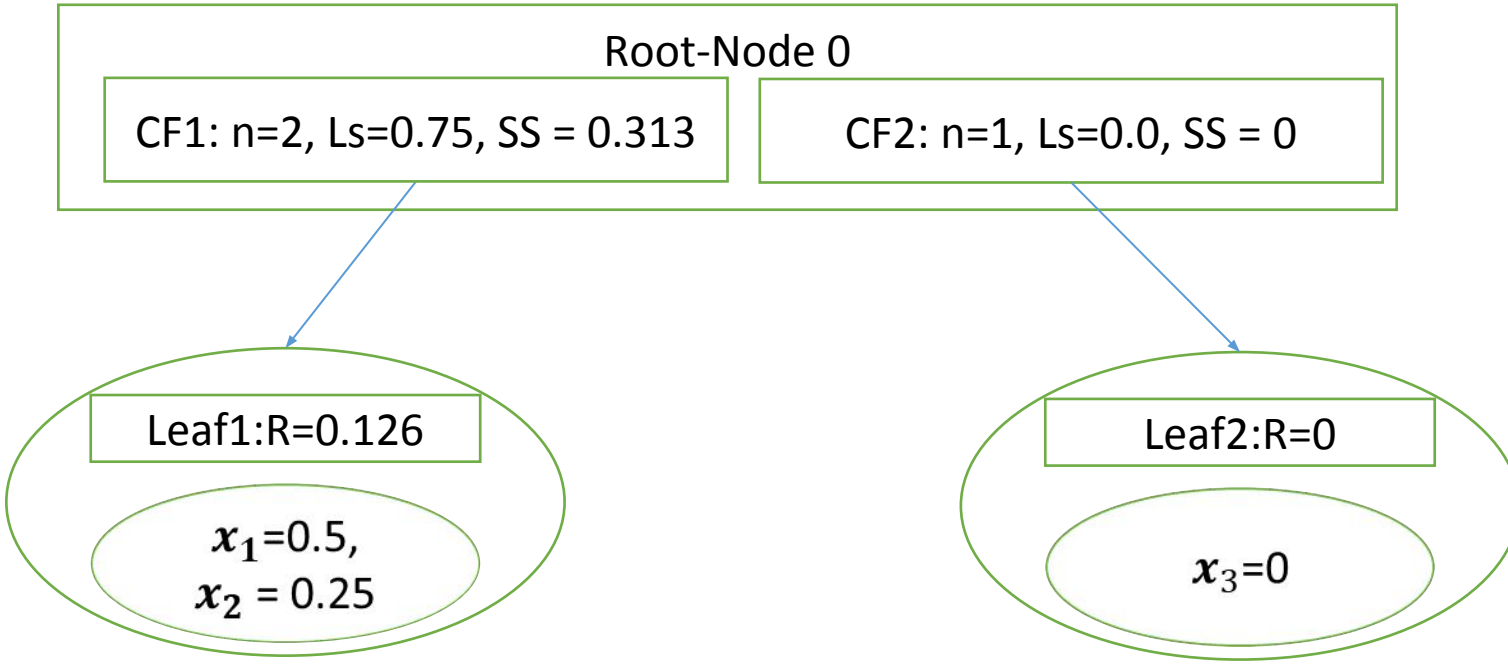


Example

- The third data value $x_3 = 0$ is entered.
- BIRCH tentatively passes $x_3 = 0$ to Leaf1.
- However, the radius of Leaf1 now increases to $R = 0.205 > T = 0.15$. The Threshold value T=0.15 is exceeded, so x3 is not assigned to Leaf1.
- Instead, a new leaf is initialized, called Leaf2, containing x3 only. Node splitting is done by choosing the farthest pair of entries as seeds, and redistributing the remaining entries based on the closest criteria

0.5
0.25
0
0.65
1
1.4
1.1

- T = 0.15
- L = 2
- B = 2



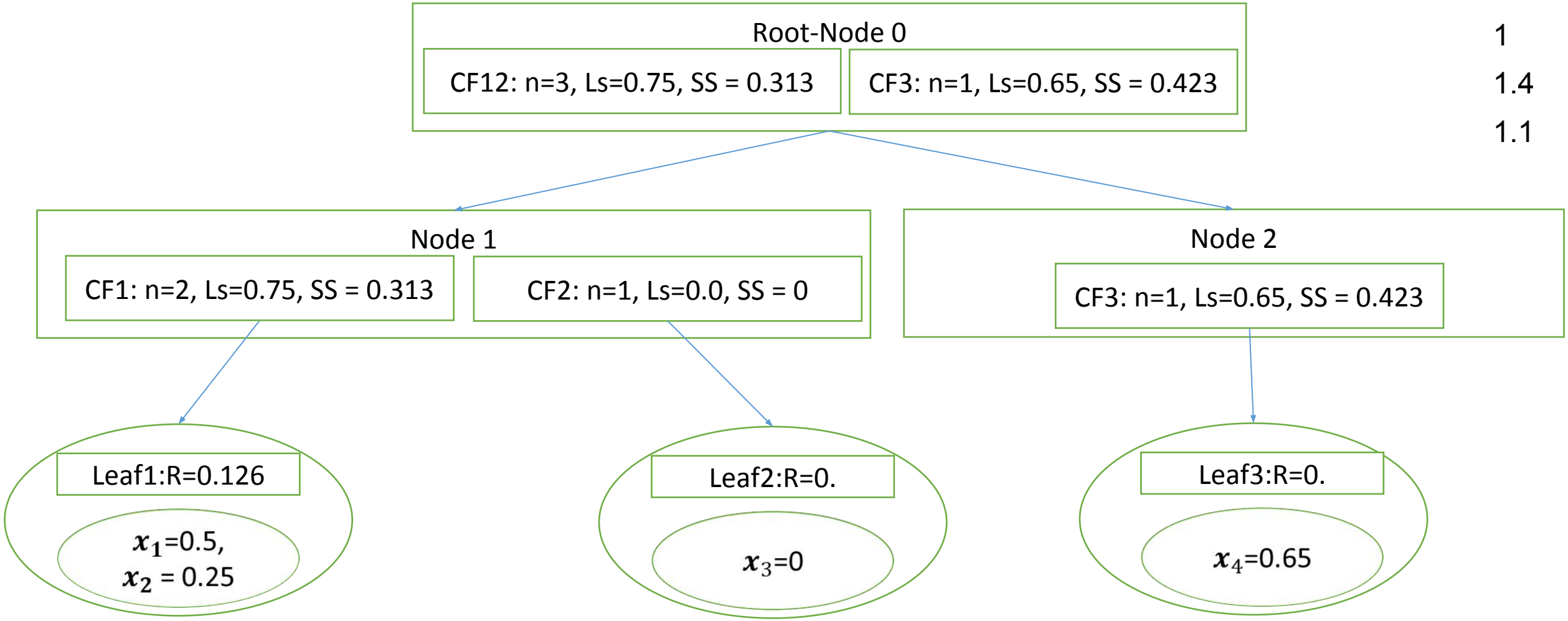
Example

- The fourth data value $x_4 = 0.65$ is entered. BIRCH compares x_4 to the locations of CF1 and CF2.
- The location is measured by $\overline{x} = \frac{LS}{n}$. We have $\overline{x}_{CF_1} = \frac{0.75}{2} = 0.375$ and $\overline{x}_{CF_2} = \frac{0}{1} = 0$
- The data point $x_4 = 0.65$ is thus closer to CF1 than to CF2. Thus, BIRCH tentatively passes x4 to CF1.
- The radius of CF1 now increases to R=0.166>T=0.15. The Threshold value T=0.15 is exceeded, so x_4 is not assigned to CF1. Instead, we would like to initialize a new leaf.
- However, B=2 means that we cannot have three leaf's in a leaf node. We must therefore split the root node into (i) Node1, which has as its children Leaf1 and Leaf2, and (ii) Node2, whose only leaf Leaf3 contains only x_4

Example

- T = 0.15
- L = 2
- B = 2

0.5
0.25
0
0.65
1
1.4
1.1



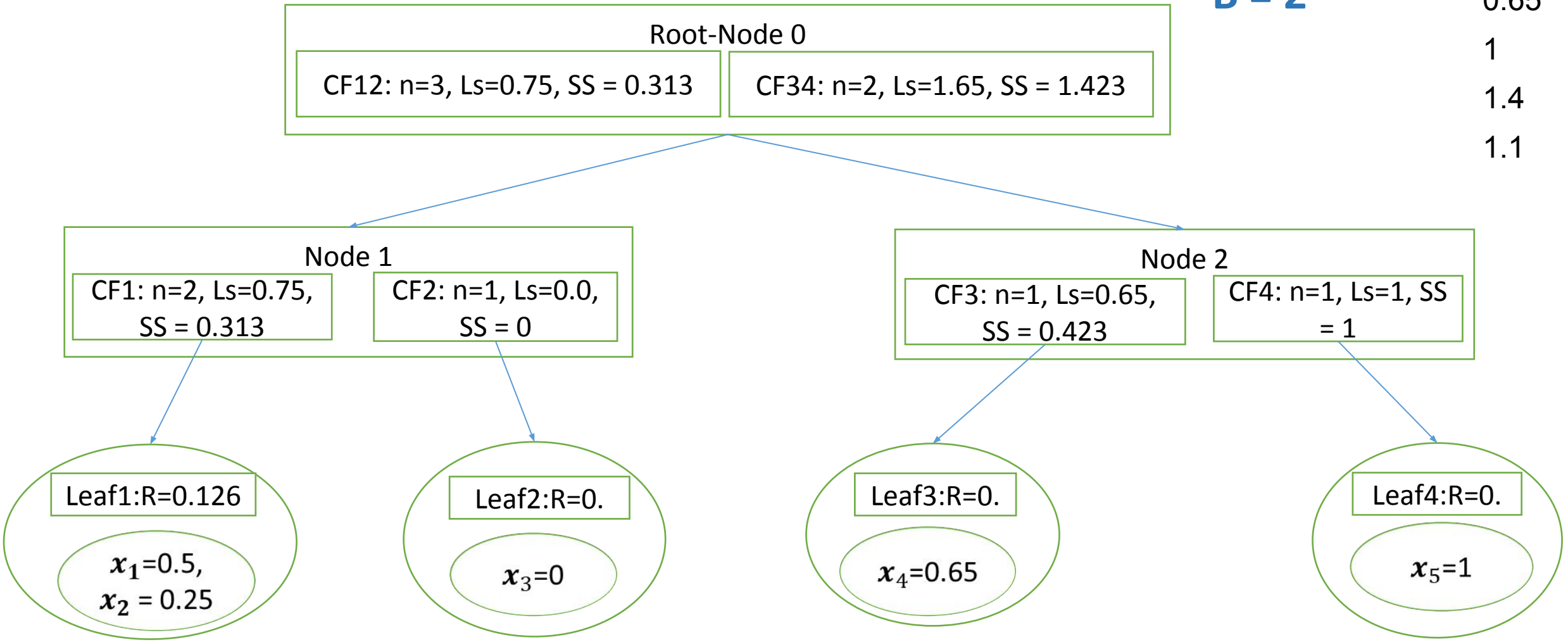
Example

- The fifth data value $x_5 = 1$ is entered.
- BIRCH compares $x_5 = 1$ with the location of CF_{12} and CF_3.
- We have $x_{CF_{12}} = 0.75/3 = \mathbf{0.25}$ and $x_{CF_3} = 0.65$ / $1 = 0.65$. The data point $x_5 = 1$ is thus closer to CF_3 than to CF_{12}.
- BIRCH passes x_5 to CF_3. The radius of CF_3 now increases to $R = 0.175 > T = 0.15$, so x_5 cannot be assigned to CF_3.
- Instead, a new leaf in leaf node Leaf4 is initialized, with CF, CF_4, containing x_5 only. The summary statistics for CF_{34} are updated.

Example

- T = 0.15
- L = 2
- B = 2

0.5
0.25
0
0.65
1
1.4
1.1



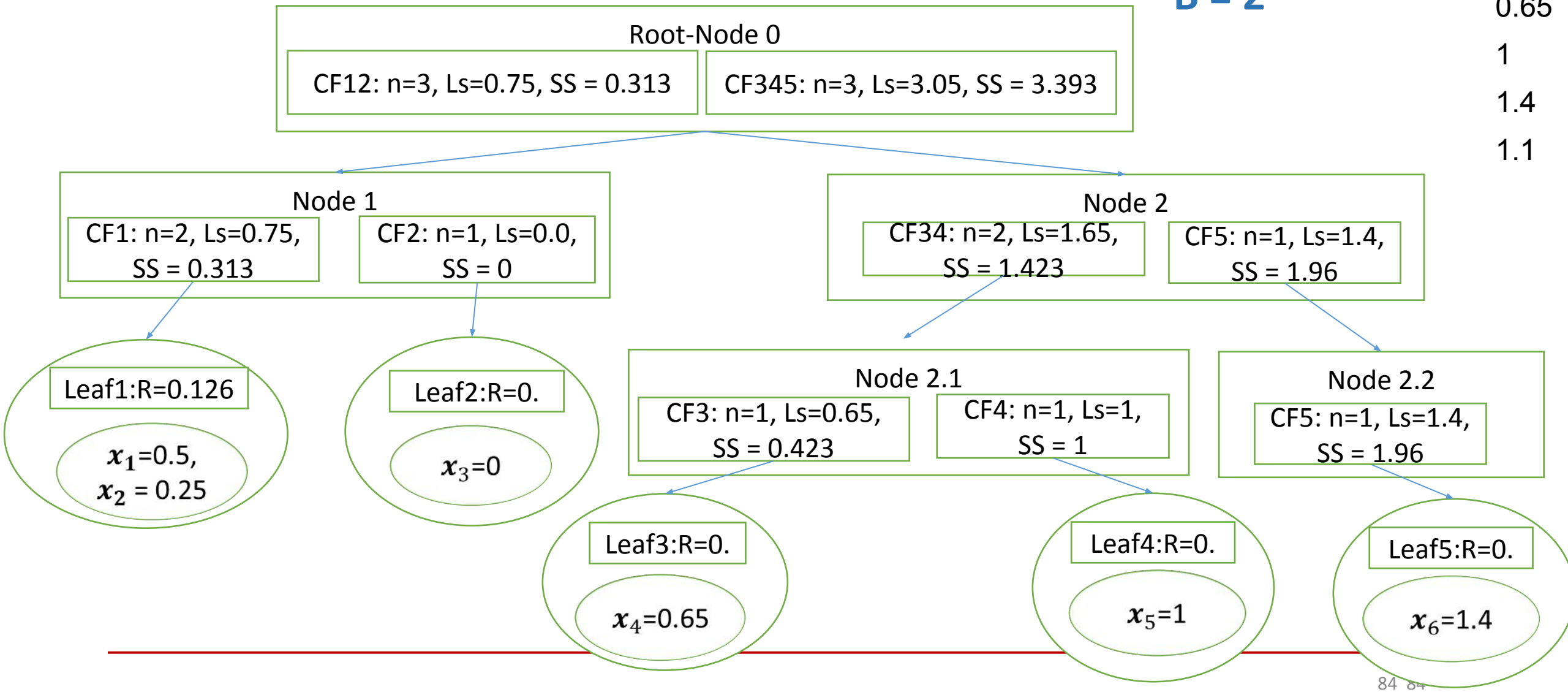
Example

- The sixth data value $x_6 = 1.4$ is entered. At the root node, BIRCH compares $x_6 = 1.4$ with the location of CF_{12} and CF_{34}.
- We have $x_{CF_{12}} = 0.75 / 3 = 0.25$ and $x_{CF_{34}} = 1.65 / 2 = 0.825$. The data point $x_6 = 1.4$ is thus closer to CF_{34}, and BIRCH passes x_6 to CF_{34}.
- The record descends to Node2, and BIRCH compares $x_6 = 1.4$ with the location of CF_3 and CF_4.
- We have $x_{CF_3} = 0.65$ and $x_{CF_4} = 1$. The data point $x_6 = 1.4$ is thus closer to CF_4 than to CF_3. BIRCH tentatively passes x_6 to CF_4.
- The radius of CF_4 now increases to $R = 0.2 > T = 0.15$. The Threshold value T=0.15 is exceeded, so x_6 is not assigned to CF_4. But the branching factor B=2 means that we may have at most two leaf nodes branching off of any non-leaf node. Therefore, we will need a new set of non-leaf nodes, Node2.1 and Node2.2, branching off from Node2.
- Node2.1 contains CF_3 and CF_4, while Node2.2 contains the desired new CF_5 and the new leaf node Leaf5 as its only child, containing only the information from x_6.

Example

- T = 0.15
- L = 2
- B = 2

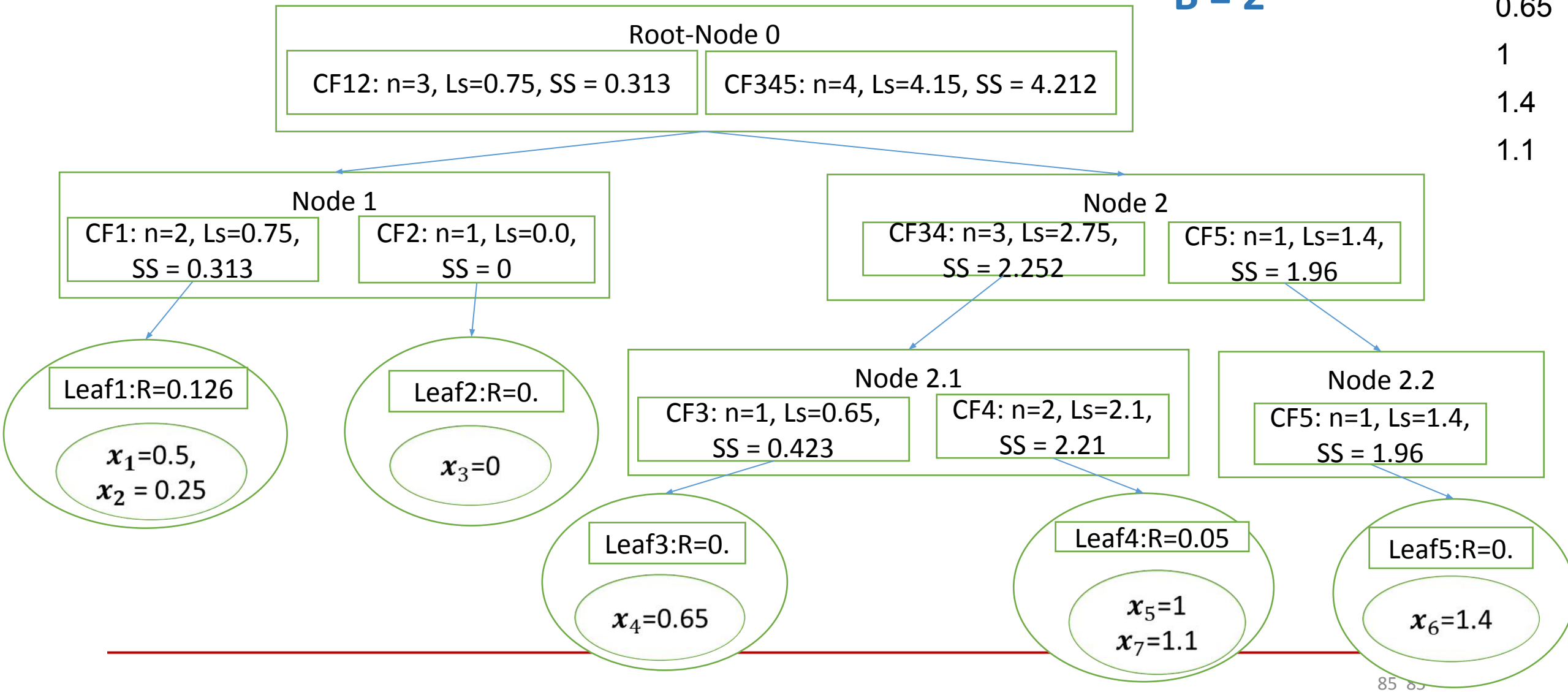
0.5
0.25
0
0.65
1
1.4
1.1



Example

- T = 0.15
- L = 2
- B = 2

0.5
0.25
0
0.65
1
1.4
1.1



CLUSTERING THE SUB-CLUSTERS

- **AGGLOMERATIVE CLUSTERING OF CLUSTER FEATURES**
- **Let *kmax* represent the total number of distinct CFs found by BIRCH at leaf node.**
  - **For $k = k_{max}$ to 2, perform the following:**
  - **Find the two sub-cluster centers that are closest together.**
    - **Merge the indicated clusters. Update the summary statistics.**
    - **Report evaluative measures.**

Clustering Using Representative (CURE)

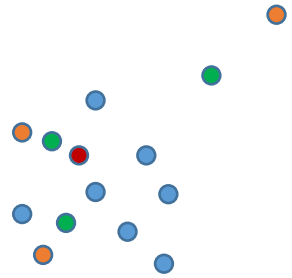
- **Drawbacks of Traditional Clustering Algorithms**
  - **Centroid-based approach (using d_{mean}) considers only one point as representative of a cluster - the cluster centroid.**
  - **All-points approach (based on d_{min}) makes the clustering algorithm extremely sensitive to outliers.**
  - **Both of them can't work well for non-spherical or arbitrary shaped clusters.**

CURE: Approach

- CURE is positioned between centroid based and all point extremes.
- A constant number of *well scattered* points is used to capture the shape and extend of a cluster.
- The points are *shrunk* towards the centroid of the cluster by a factor α.
- These *well scattered* and *shrunk* points are used as representative of the cluster.

CURE: Approach

- Scattered points approach alleviates shortcomings of centroid based and all point based methods.
  - Since multiple representatives are used, the splitting of large clusters is avoided.
  - Multiple representatives allow for discovery of non spherical clusters.
  - The shrinking phase will affect outliers more than other points since their distance from the centroid will be decreased more than that of regular points.

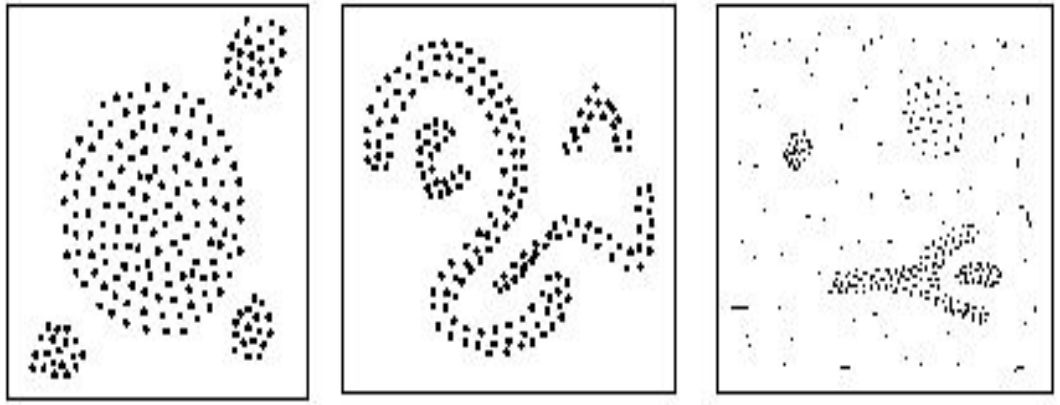


CURE: Approach

- Initially since all points are in separate clusters, each cluster is defined by the point in the cluster.
- Clusters are merged until they contain at least c points.
- The first scattered point in a cluster in one which is farthest away from the clusters centroid.
- Other scattered points are chosen so that their distance from previously chosen scattered points in maximal.
- When c well scattered points are calculated they are shrunk by some factor α (r = p + α*(mean-p)).
- After clusters have c representatives the distance between two clusters is the distance between two of the closest representatives of each cluster
- Every time two clusters are merged their representatives are re-calculated.

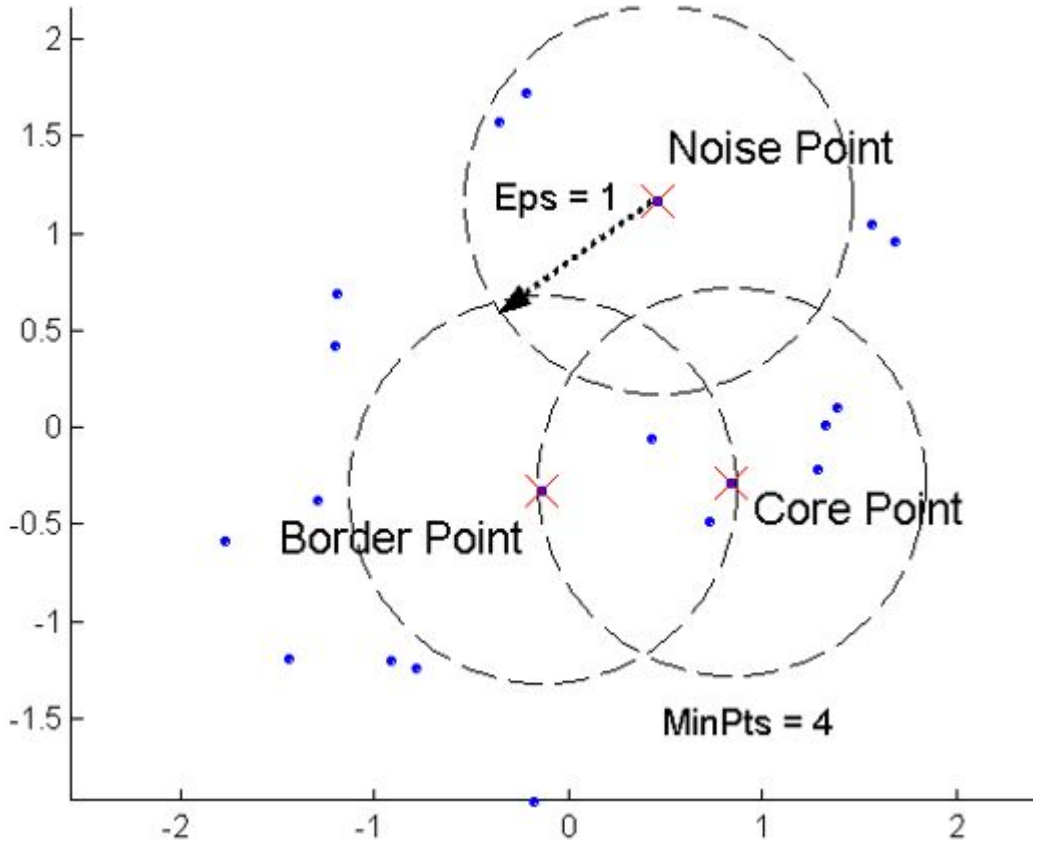
Density-Based Clustering Methods

- Clustering based on local connectivity and density functions
- 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
- Each cluster has a considerable higher density of points than outside of the cluster
- Major features:
  - Discover clusters of arbitrary shape
  - One scan



Density Based Spatial Clustering of Application of Noise (DBSCAN)

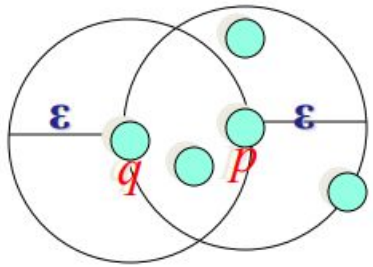
- Two global parameters:
  - Eps: Maximum radius of the neighbourhood
  - MinPts: Minimum number of points in an Eps-neighbourhood of that point
- Density = number of points within a specified radius r (Eps)
- A point is a core point if it has more than a specified number of points (MinPts) within Eps
  - 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 or a border point.



Density-reachability

- An object q is directly density-reachable from object p, if p is a core object and q is in p's Eps-neighborhood

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

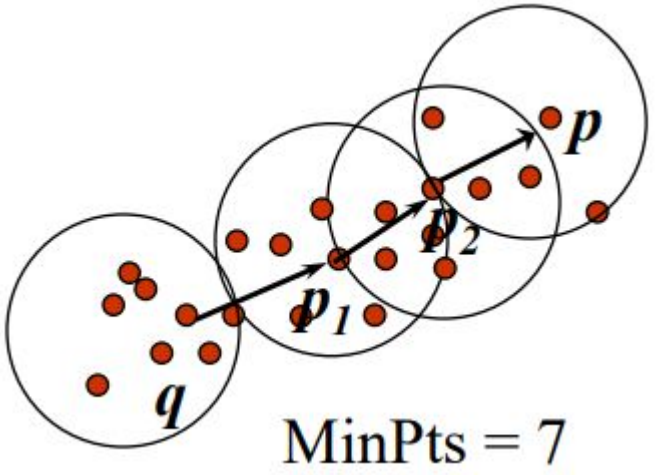


MinPts = 4

Density-reachability

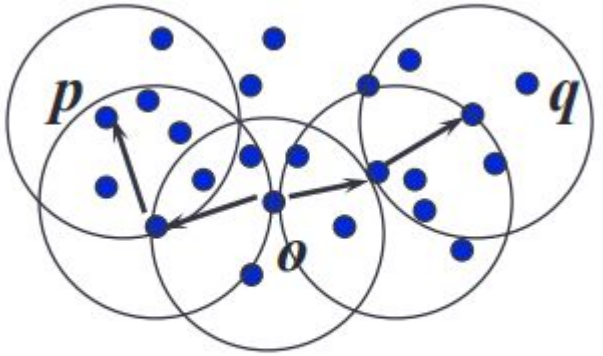
- Density-Reachable (directly and indirectly):
- A point p is directly density-reachable from p2
- p2 is directly density-reachable from p1
- p1 is directly density-reachable from q
- p ← p2 ← p1 ←q form a chain

- p is (indirectly) density-reachable from q
- q is not density-reachable from p



Density-Connectivity

- A pair of points p and q are density-connected, If they are commonly density-reachable from a point o
- Density-connectivity is symmetric

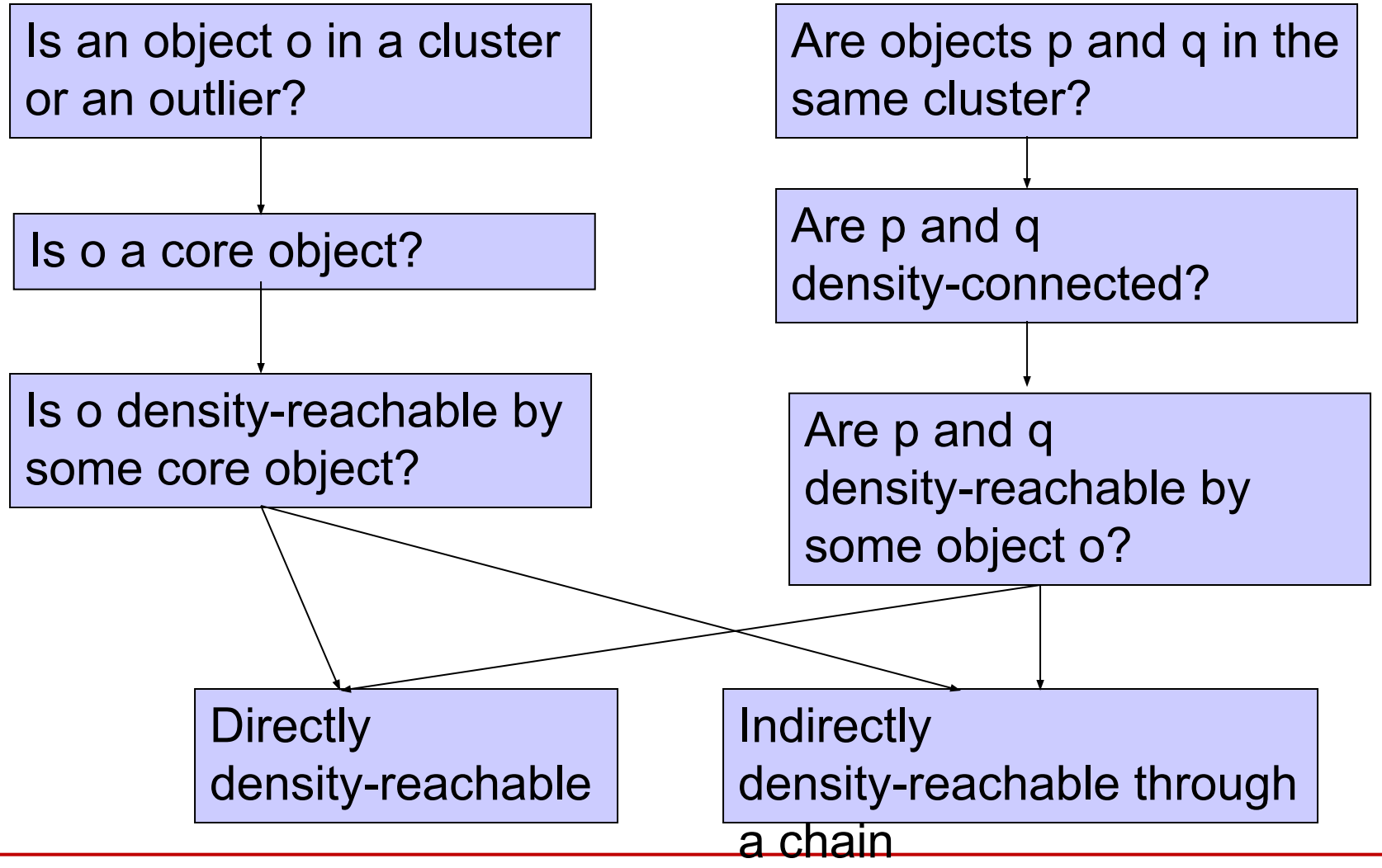


MinPts = 7

DBSCAN

- Cluster : Given a data set D, parameter Eps and MinPts, a cluster C is a subset of D satisfying the following conditions
  - For all points $p, q \in C$, if $p \in C$ and q is density-reachable from p with respect to Eps and MinPts, then $q \in C$
  - For all points $p, q \in C$, p is density connected to q with respect to Eps and Minpts (Connectivity)
- Note: cluster contains core points as well as border points

Review of Concepts

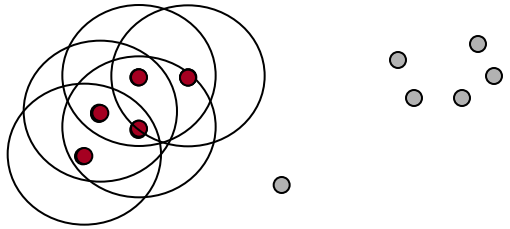


DBSCAN-Algorithm

```
Input: The data set D

Parameter: ε, MinPts

For each object p in D
    if p is a core object and not processed then
        C = retrieve all objects density-reachable from p
        mark all objects in C as processed
        report C as a cluster
    else mark p as outlier
    end if

End For
```

DBSCAN-Example

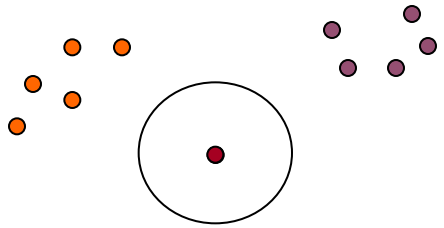
- **Parameter**
  - ε = 2 cm
  - *MinPts* = 3



for each $o \in D$ **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

DBSCAN-Example

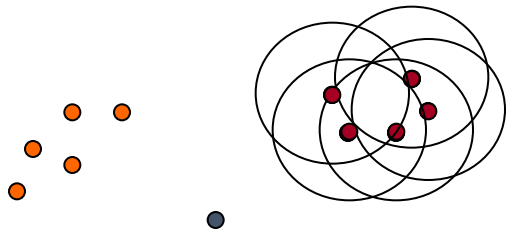
- **Parameter**
  - ε = 2 cm
  - *MinPts* = 3



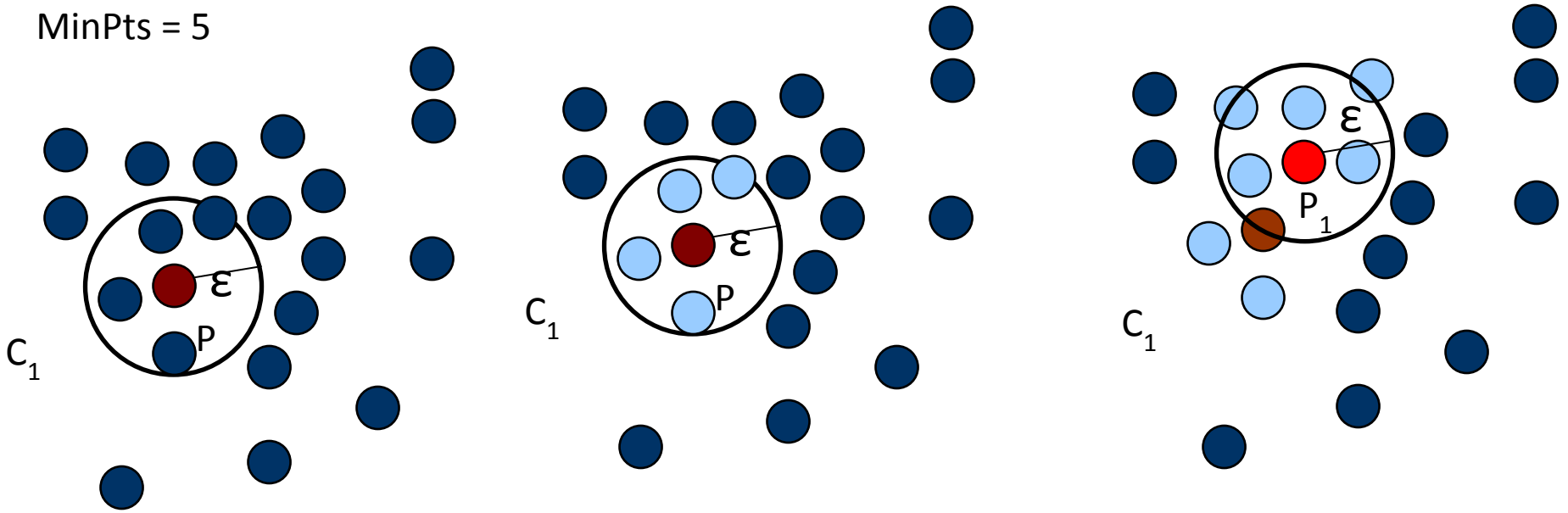
for each $o \in D$ **do**
    if o is not yet classified **then**
        if o is a core-object **then**
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE

DBSCAN-Example

- **Parameter**
  - ε = 2 cm
  - *MinPts* = 3

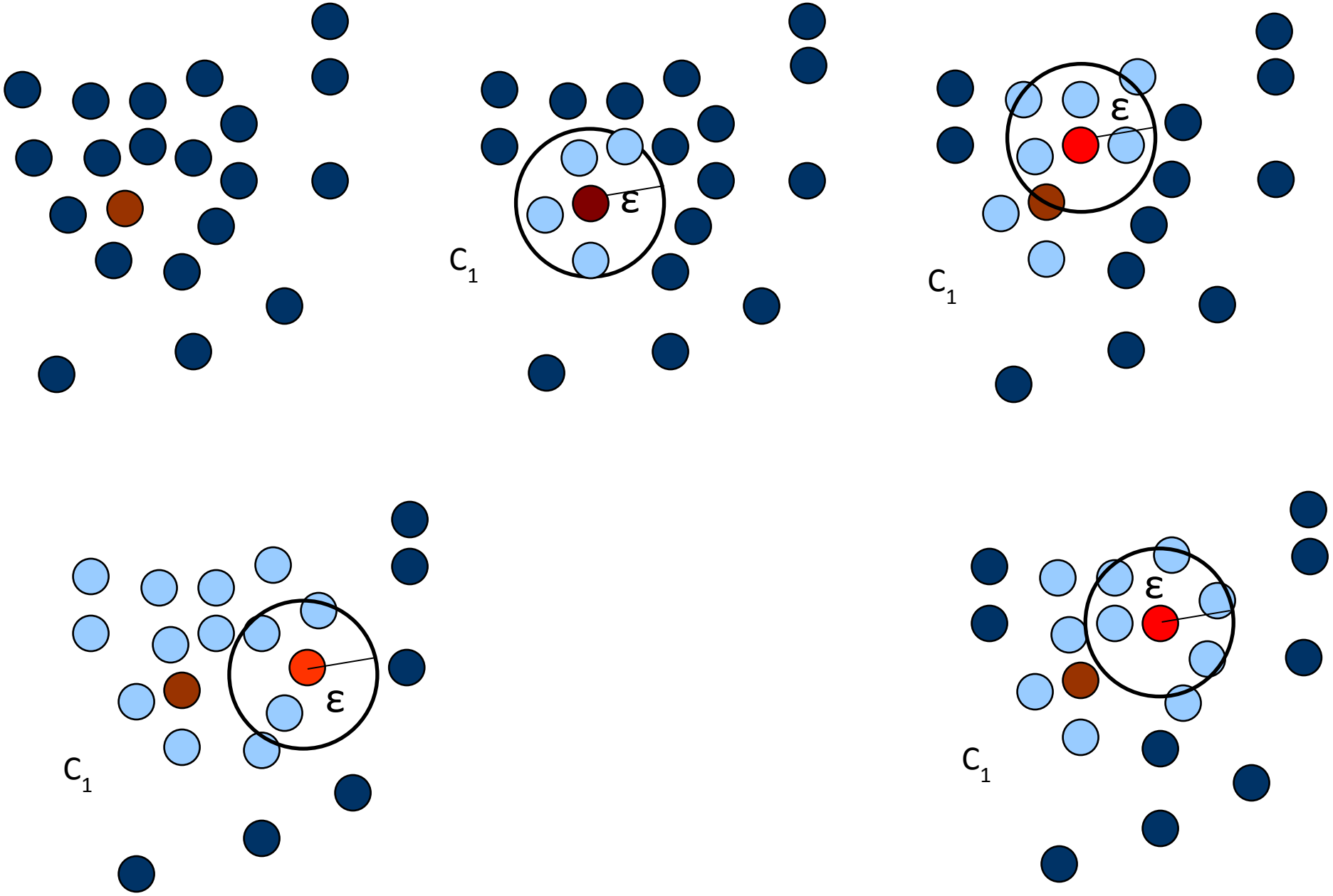


for each $o \in D$ **do**
    if o is not yet classified **then**
        if o is a core-object **then**
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE

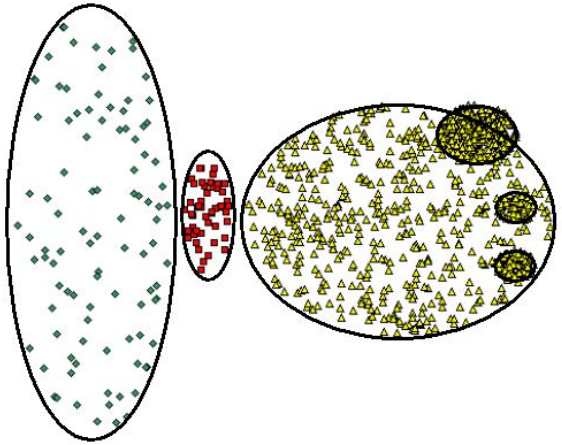


1. Check the ε-neighborhood of p;
2. If p has less than MinPts neighbors then mark p as outlier and continue with the next object
3. Otherwise mark p as processed and put all the neighbors in cluster C

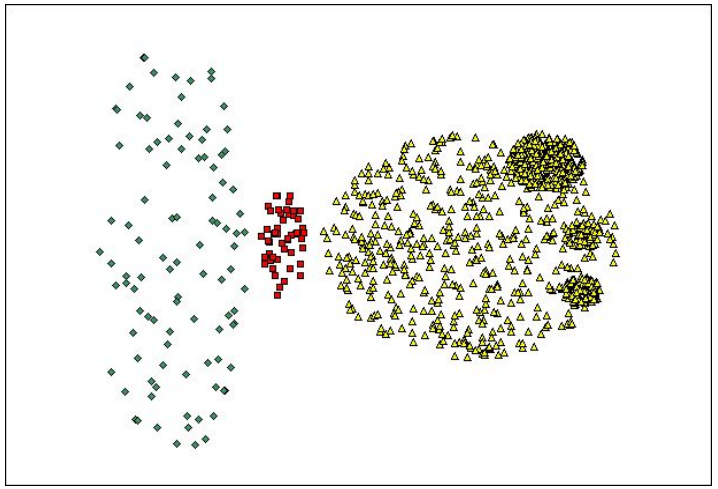
1. Check the unprocessed objects in C
2. If no core object, return C
3. Otherwise, randomly pick up one core object p_1, mark p_1 as processed, and put all unprocessed neighbors of p_1 in cluster C


ε
C1
ε
C1
ε
C1
ε
C1

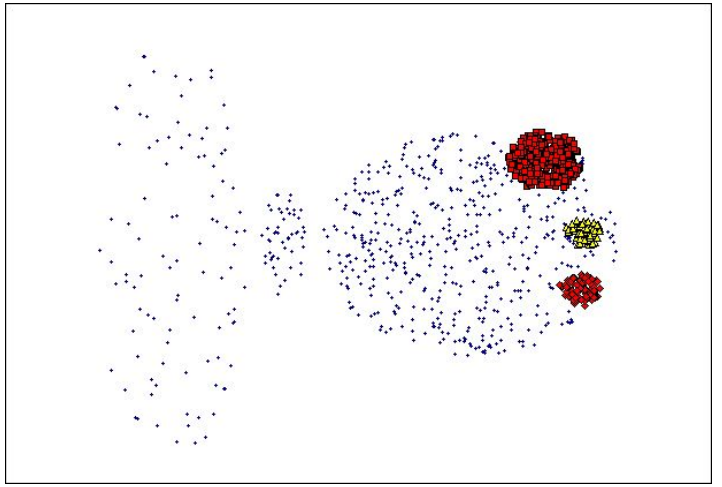
When DBSCAN Does NOT Work Well



Original Points

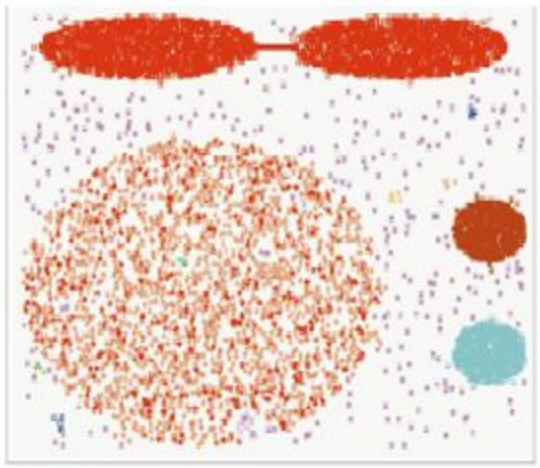


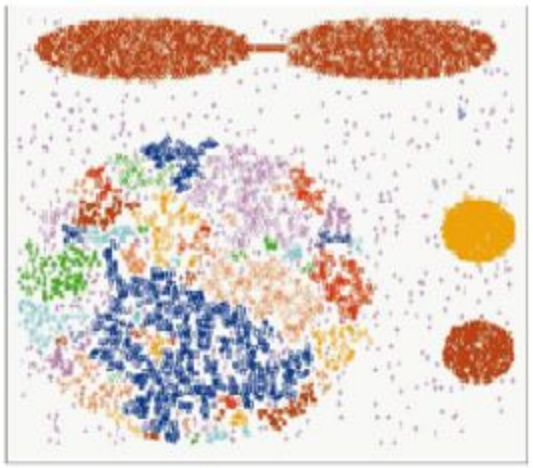
- Cannot handle Varying densities
- sensitive to parameters

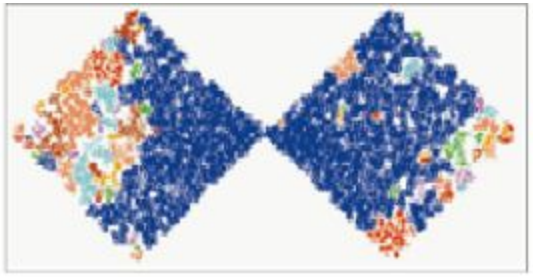


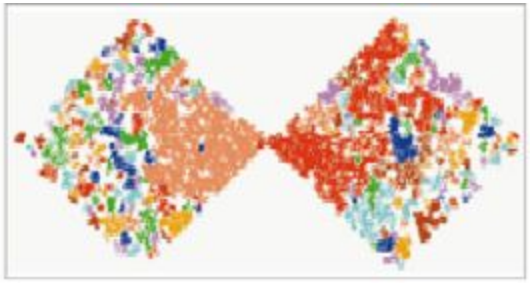
(MinPts=4, Eps=9.75)

DBSCAN: Sensitive to Parameters









Density Based Clustering: Discussion

- Advantages
  - **Clusters can have arbitrary shape and size**
  - **Number of clusters is determined automatically**
  - **Can separate clusters from surrounding noise**
- Disadvantages
  - **Input parameters may be difficult to determine**
  - **In some situations very sensitive to input parameter setting**