Module 3

Decision Tree & Clustering

Decision trees

- Decision tree learning is a method for approximating discrete valued target functions, in which the learned function is represented by a decision tree

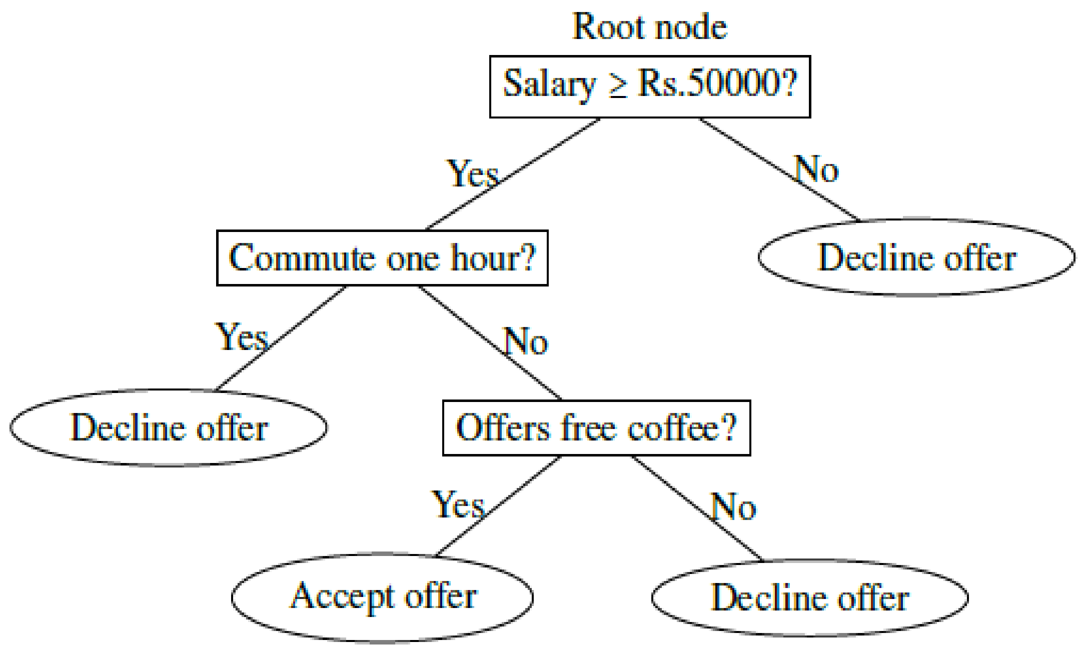


Figure 8.1: Example for a decision tree

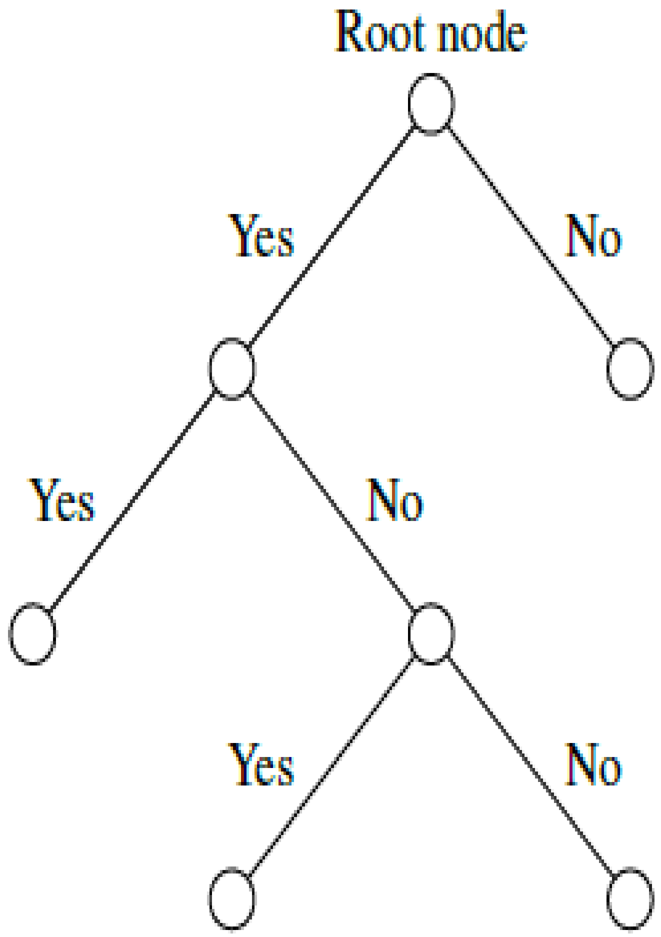
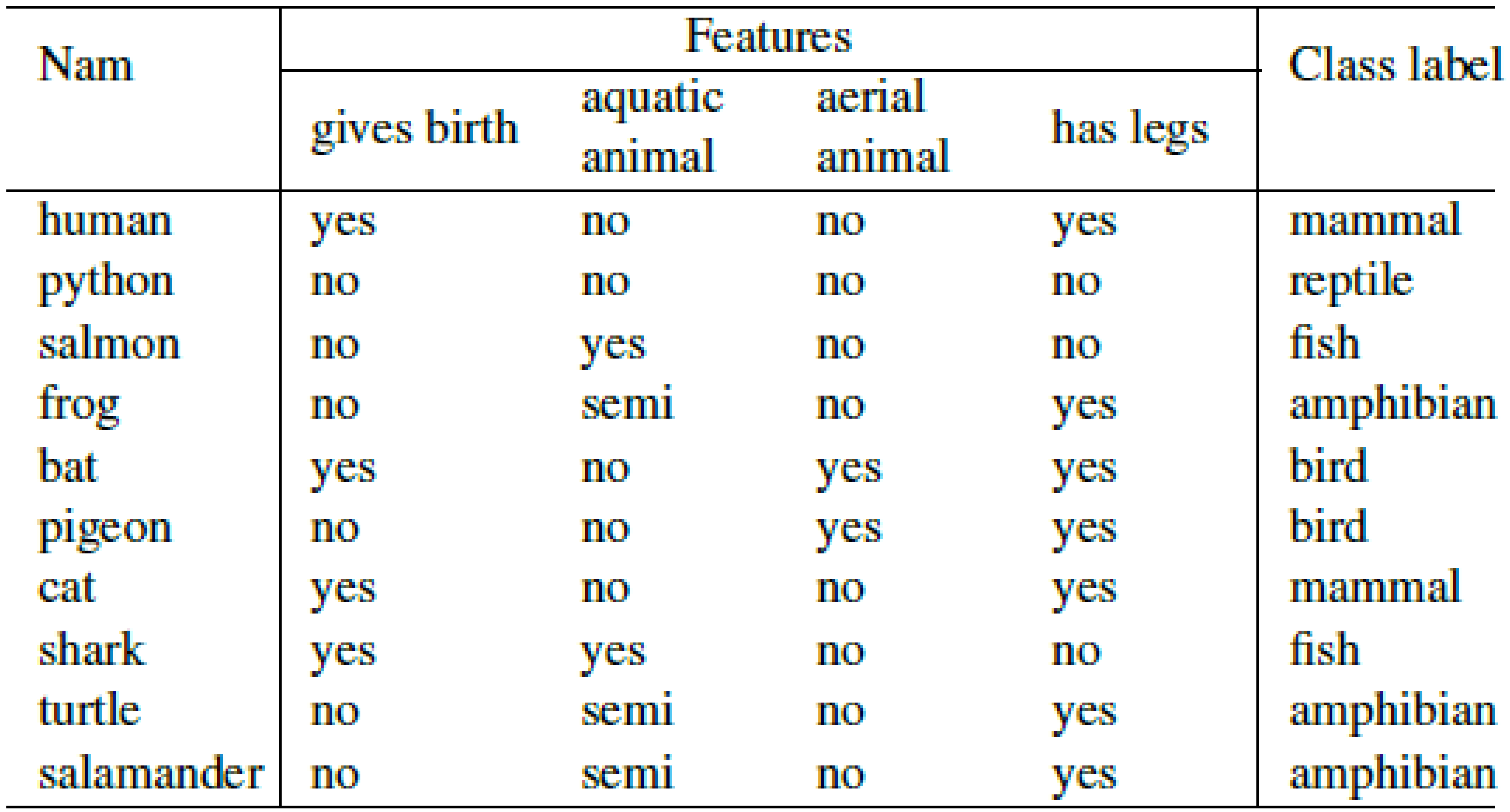


Figure 8.2: The graph-theoretical representation of the decision tree in Figure 8.6

Two types of decision trees

1. Classification trees: Tree models where the target variable can take a discrete set of values are called classification trees.
2. Regression Trees

Example



Nam	Features				Class label
	gives birth	aquatic animal	aerial animal	has legs	
human	yes	no	no	yes	mammal
python	no	no	no	no	reptile
salmon	no	yes	no	no	fish
frog	no	semi	no	yes	amphibian
bat	yes	no	yes	yes	bird
pigeon	no	no	yes	yes	bird
cat	yes	no	no	yes	mammal
shark	yes	yes	no	no	fish
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Table 8.1: The vertebrate data set

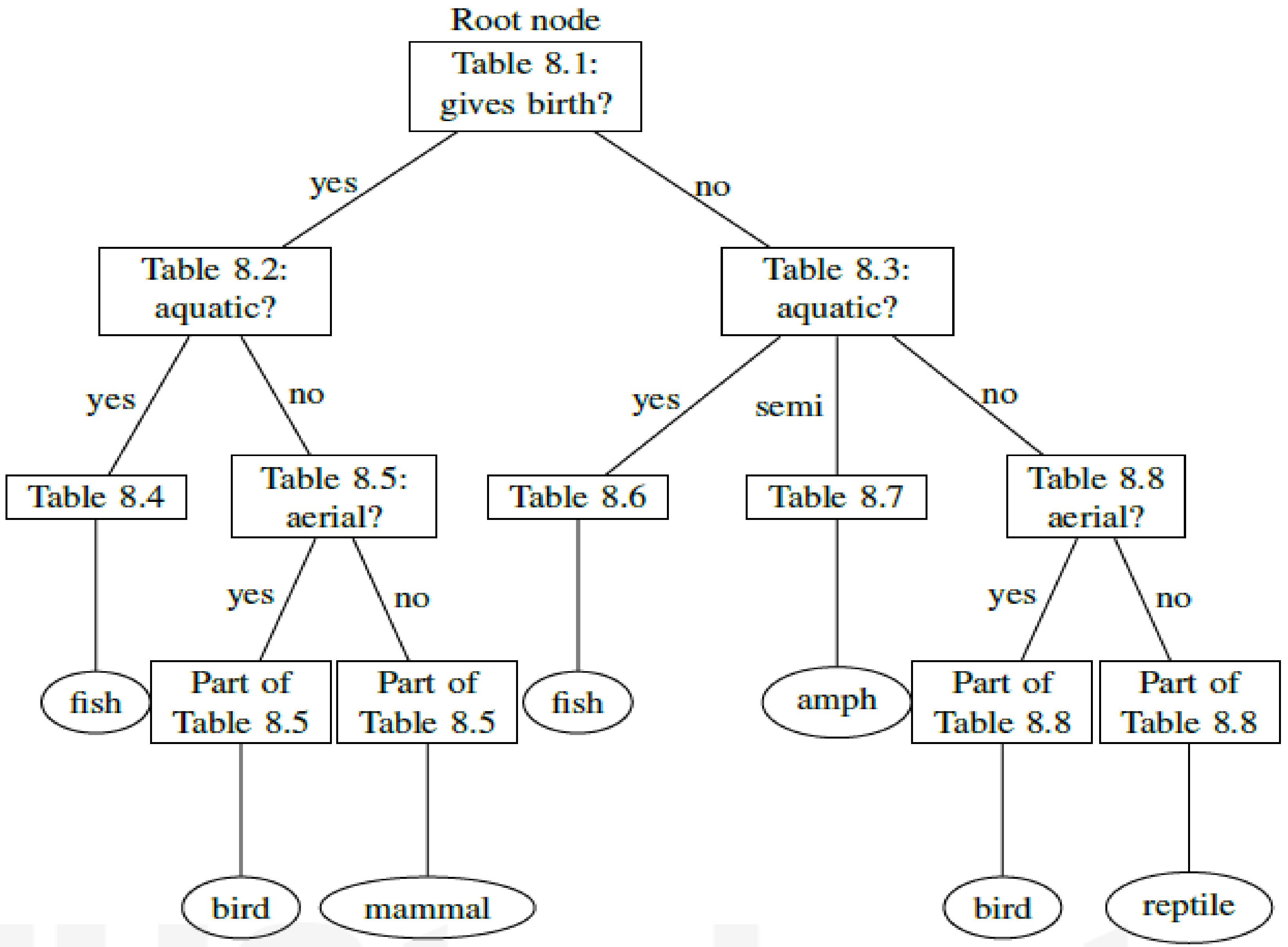


Figure 8.5: Classification tree

Splitting Indices: Feature selection measures

- If a dataset consists of n attributes then deciding which attribute is to be to placed at the root or at different levels of the tree as internal nodes is a complicated problem.
- These are called the feature selection measures.
- Two of the popular feature selection measures are **information gain and Gini index.**

Entropy

- The degree to which a subset of examples contains only a single class is known as purity, and any subset composed of only a single class is called a pure class.
- Informally, entropy is a measure of “impurity” in a dataset.
- Sets with high entropy are very diverse
- Entropy is measured in bits.
- If there are only two possible classes, entropy values can range from 0 to 1.
- For n classes, entropy ranges from 0 to log2(n).
- In each case, the minimum value indicates that the sample is completely homogeneous, while the maximum value indicates that the data are as diverse as possible.

Definition

- Consider a segment S of a dataset having c number of class labels. Let pi be the proportion of examples in S having the i class label.
- The entropy of S is defined as

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i).$$

Let “xxx” be some class label. We denote by p_{xxx} the proportion of examples with class label “xxx”.

1. **Entropy of data in Table 8.1**

   Let S be the data in Table 8.1. The class labels are ”amphi”, “bird”, ”fish”, ”mammal” and ”reptile”. In S we have the following numbers.

Number of examples with class label “amphi”	= 3
Number of examples with class label “bird”	= 2
Number of examples with class label “fish”	= 2
Number of examples with class label “mammal”	= 2
Number of examples with class label “reptile”	= 1
Total number of examples	= 10

Therefore, we have:

$$\text{Entropy}\ (S) = \sum_{\text{for all classes “xxx”}} -p_{\text{xxx}} \log_2(p_{\text{xxx}})$$

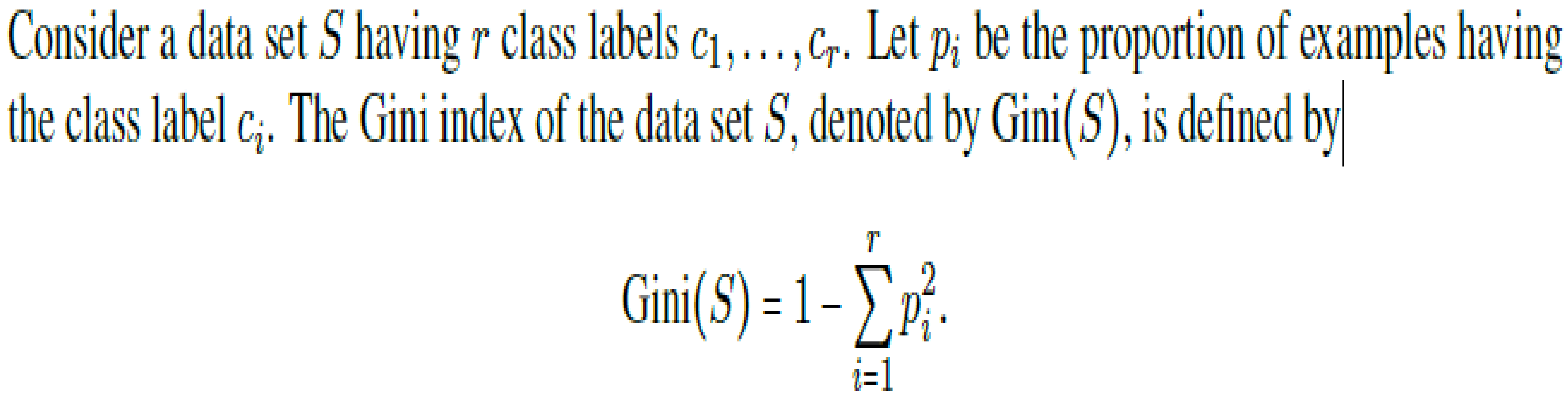
Information gain

- Let S be a set of examples, A be a feature (or, an attribute), Sv be the subset of S with A = v, and Values (A) be the set of all possible values of A. Then the information gain of an attribute A relative to the set S, denoted by Gain (S;A), is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

Gini index

- It is a measure of diversity
- Best splitter- Attribute with smallest Gini value



Consider a data set S having r class labels $c_1, \ldots, c_r$. Let p_i be the proportion of examples having the class label c_i. The Gini index of the data set S, denoted by $\text{Gini}(S)$, is defined by

$$\text{Gini}(S) = 1 - \sum_{i=1}^{r} p_i^2.$$

Example

Let S be the data in Table 8.1. There are four class labels "amphi", "bird", "fish", "mammal" and "reptile". The numbers of examples having these class labels are as follows:

Number of examples with class label "amphi"	= 3
Number of examples with class label "bird"	= 2
Number of examples with class label "fish"	= 2
Number of examples with class label "mammal"	= 2
Number of examples with class label "reptile"	= 1
Total number of examples	= 10

The Gini index of S is given by

$$\begin{aligned}\text{Gini}(S) &= 1 - \sum_{i=1}^{r} p_i^2\\ &= 1 - (3/10)^2 - (2/10)^2 - (2/10)^2 - (2/10)^2 - (1/10)^2\\ &= 0.78\end{aligned}$$

Gini Split Index

Let S be a set of examples, A be a feature (or, an attribute), S_v be the subset of S with $A = v$, and Values(A) be the set of all possible values of A. Then the *Gini split index of A relative to S*, denoted by $\text{Gini}_{\text{split}}(S, A)$, is defined as

$$\text{Gini}_{\text{split}}(S,A) = \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Gini}(S_v).$$

where $|S|$ denotes the number of elements in S.

The ID3 algorithm

- Developed by Ross Quinlan
- Iterative Dichotomiser 3
- Assumptions
  - The algorithm uses information gain to select the most useful attribute for classification.
  - Assume that there are only two class labels, namely, “+” and “−”. The examples with class labels “+” are called positive examples and others negative examples.

Notations

The following notations are used in the algorithm:

S	The set of examples
C	The set of class labels
F	The set of features
A	An arbitrary feature (attribute)
Values(A)	The set of values of the feature A
v	An arbitrary value of A
S_v	The set of examples with $A = v$
Root	The root node of a tree

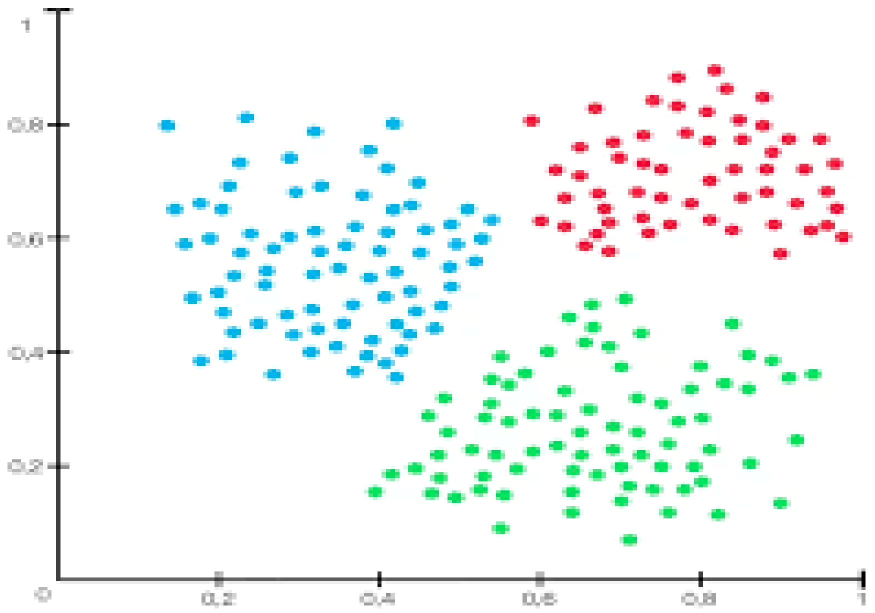
Algorithm ID3(S, F, C)

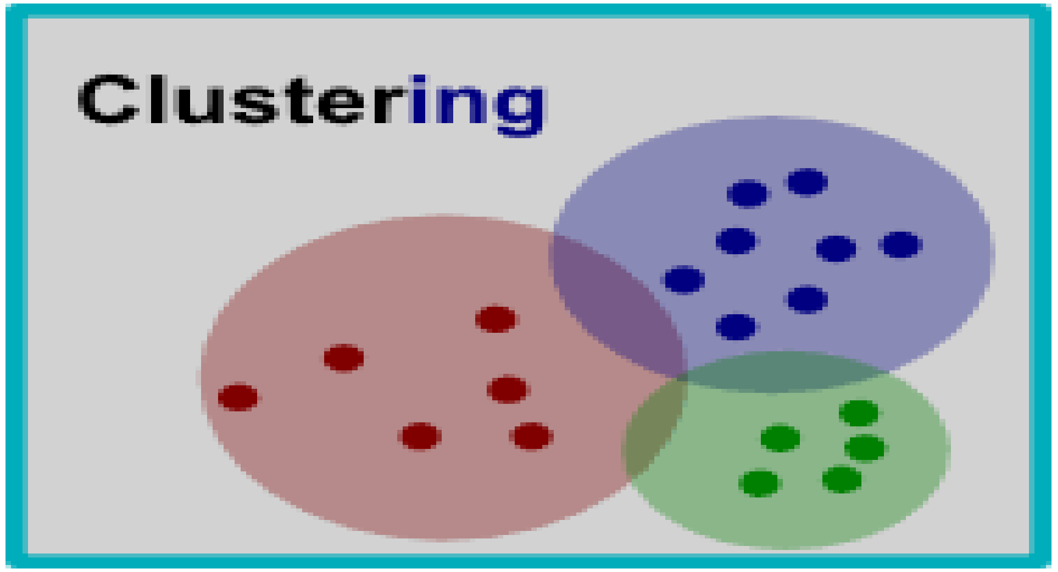
```
1. Create a root node for the tree.
2. if (all examples in $S$ are positive) then
3.         return  single node tree Root with label "+"
4. end if
5. if (all examples are negative) then
6.         return  single node tree Root with label "–"
7. end if
8. if (number of features is 0) then
9.         return  single node tree Root with label equal to the most common class label.
```

```
10. else
11.     Let $A$ be the feature in $F$ with the highest information gain.
12.     Assign $A$ to the Root node in decision tree.
13.     for all (values $v$ of $A$) do
14.         Add a new tree branch below Root corresponding to $v$.
15.         if ($S_v$ is empty) then
16.             Below this branch add a leaf node with label equal to the most common class
                label in the set $S$.
17.         else
18.             Below this branch add the subtree formed by applying the same algorithm ID3
                with the values ID3$(S_v, C, F - \{A\})$.
19.         end if
20.     end for
21. end if
```

Clustering

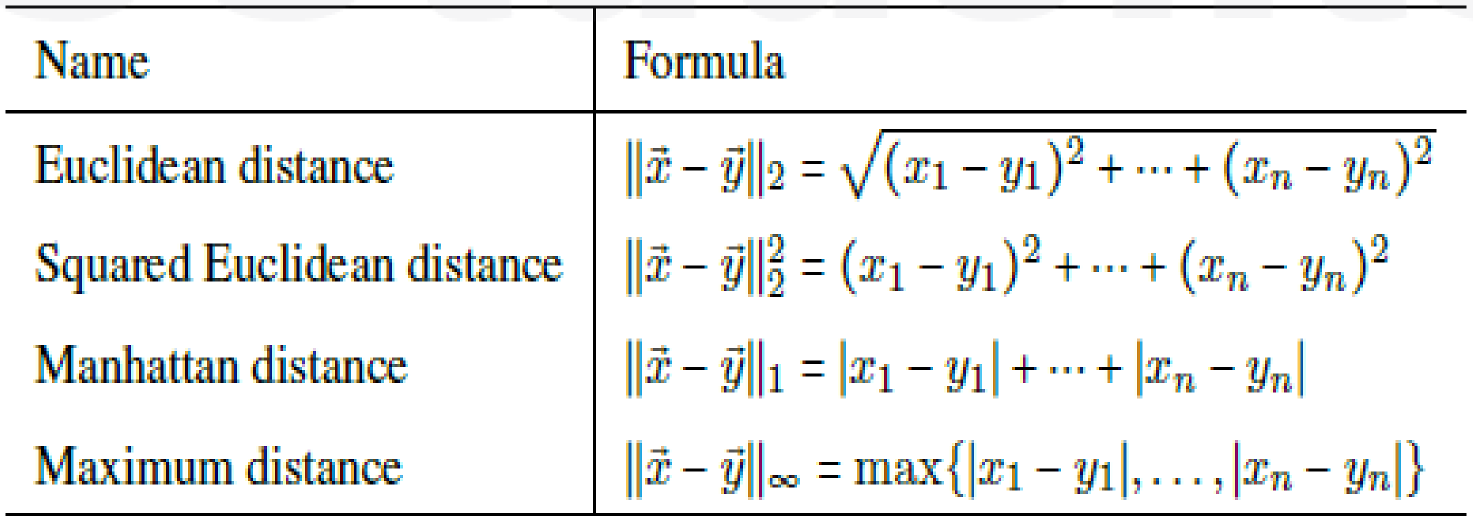
- Introduction to clustering
- Clustering or cluster analysis is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).





Measures of dissimilarity

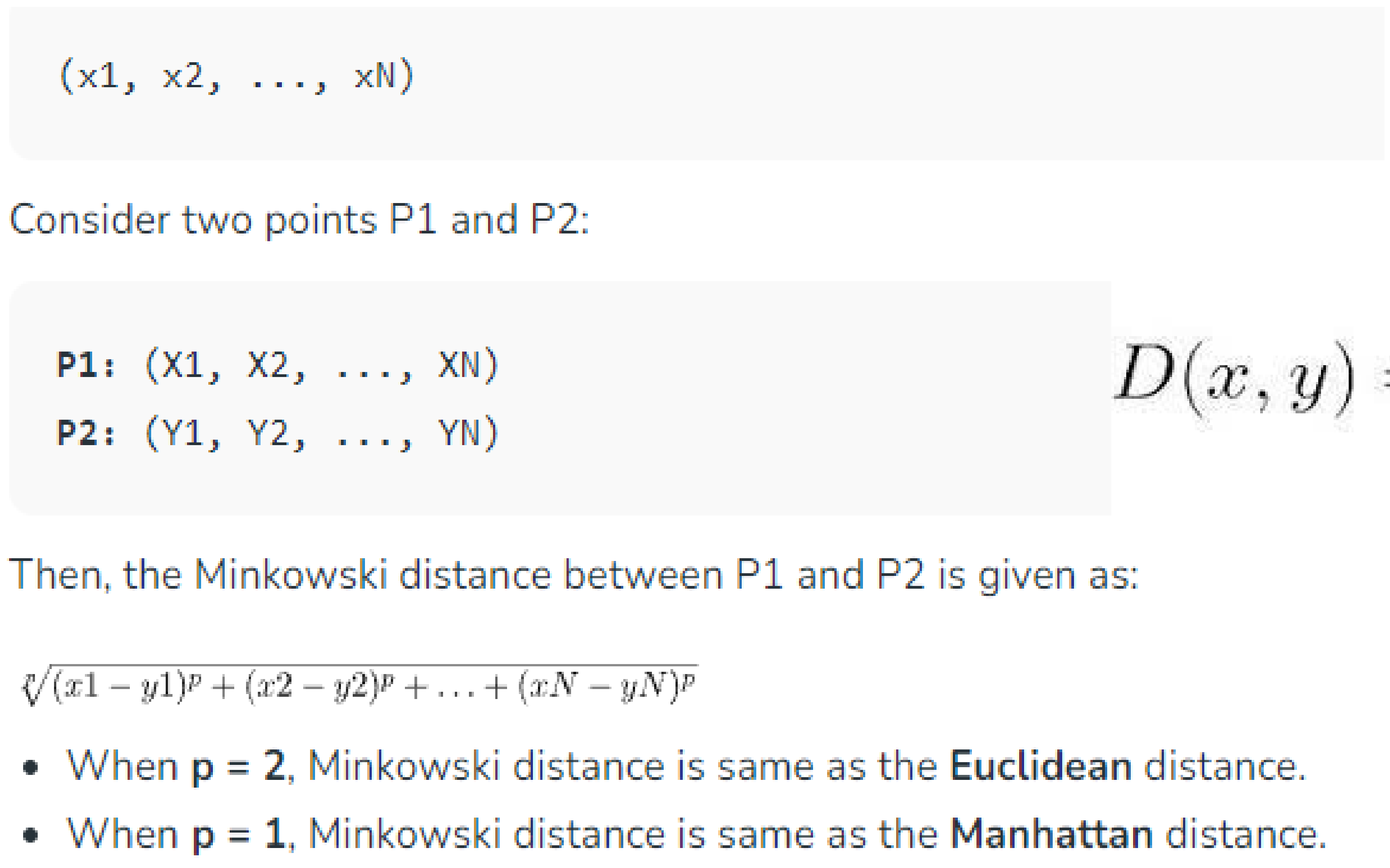
- In order to decide which clusters should be combined, or where a cluster should be split , a measure of dissimilarity between sets of observations is required.
- Measures of distance between data points
- Numeric Data



Name	Formula
Euclidean distance	$\|\vec{x}-\vec{y}\|_2 = \sqrt{(x_1-y_1)^2+\cdots+(x_n-y_n)^2}$
Squared Euclidean distance	$\|\vec{x}-\vec{y}\|_2^2 = (x_1-y_1)^2+\cdots+(x_n-y_n)^2$
Manhattan distance	$\|\vec{x}-\vec{y}\|_1 = \lvert x_1-y_1\rvert+\cdots+\lvert x_n-y_n\rvert$
Maximum distance	$\|\vec{x}-\vec{y}\|_\infty = \max\{\lvert x_1-y_1\rvert,\ldots,\lvert x_n-y_n\rvert\}$

City Block Distance /

- **Minkowski distance:** It is the **generalized** form of the Euclidean and Manhattan Distance Measure. In an **N-dimensional space**, a point is represented as,



```
(x1, x2, ..., xN)
```

Consider two points P1 and P2:

```
P1: (X1, X2, ..., XN)
P2: (Y1, Y2, ..., YN)
```

Then, the Minkowski distance between P1 and P2 is given as:

$$\sqrt[p]{(x1-y1)^p+(x2-y2)^p+\ldots+(xN-yN)^p}$$

- When **p = 2**, Minkowski distance is same as the **Euclidean** distance.
- When **p = 1**, Minkowski distance is same as the **Manhattan** distance.

$$D(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

Non-numeric data

- The Levenshtein distance is a measure of the ”distance” between two words.
- The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.
- For example, the Levenshtein distance between “kitten” and “sitting” is 3, since the following
  - three edits change one into the other, and there is no way to do it with fewer than three edits:
- Kitten : sitten (substitution of “s” for “k”)
- sitten : sittin (substitution of “i” for “e”)
- sittin : sitting (insertion of ‘g” at the end)

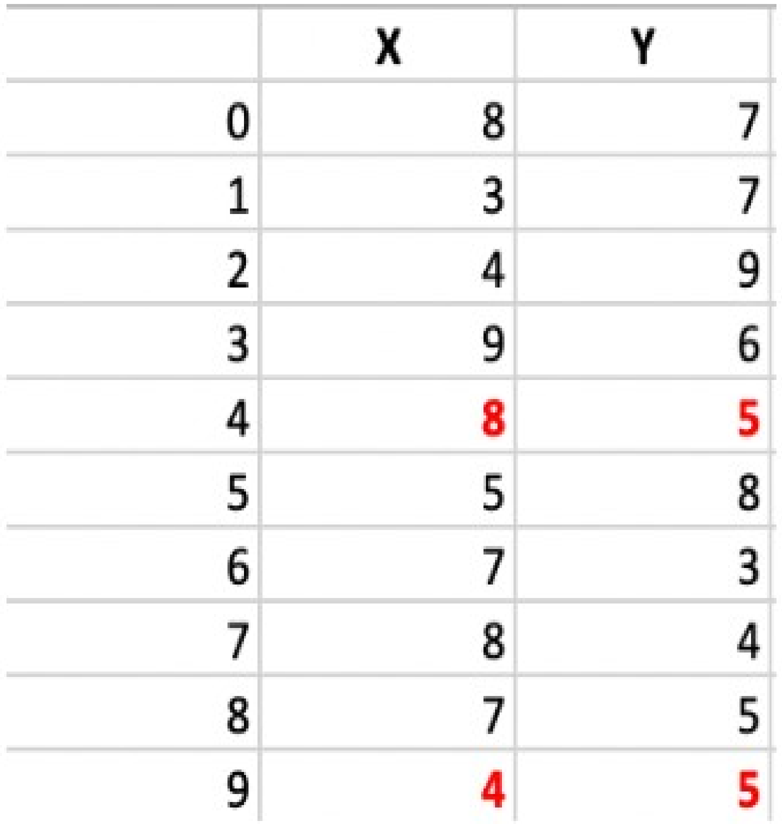
Clustering Methods

- Hierarchical vs Partitioning
  - Partitioning: K-means, K-medoid, k-mode
  - Hierarchical: Agglomerative, Divisive
- Numerical vs Categorical

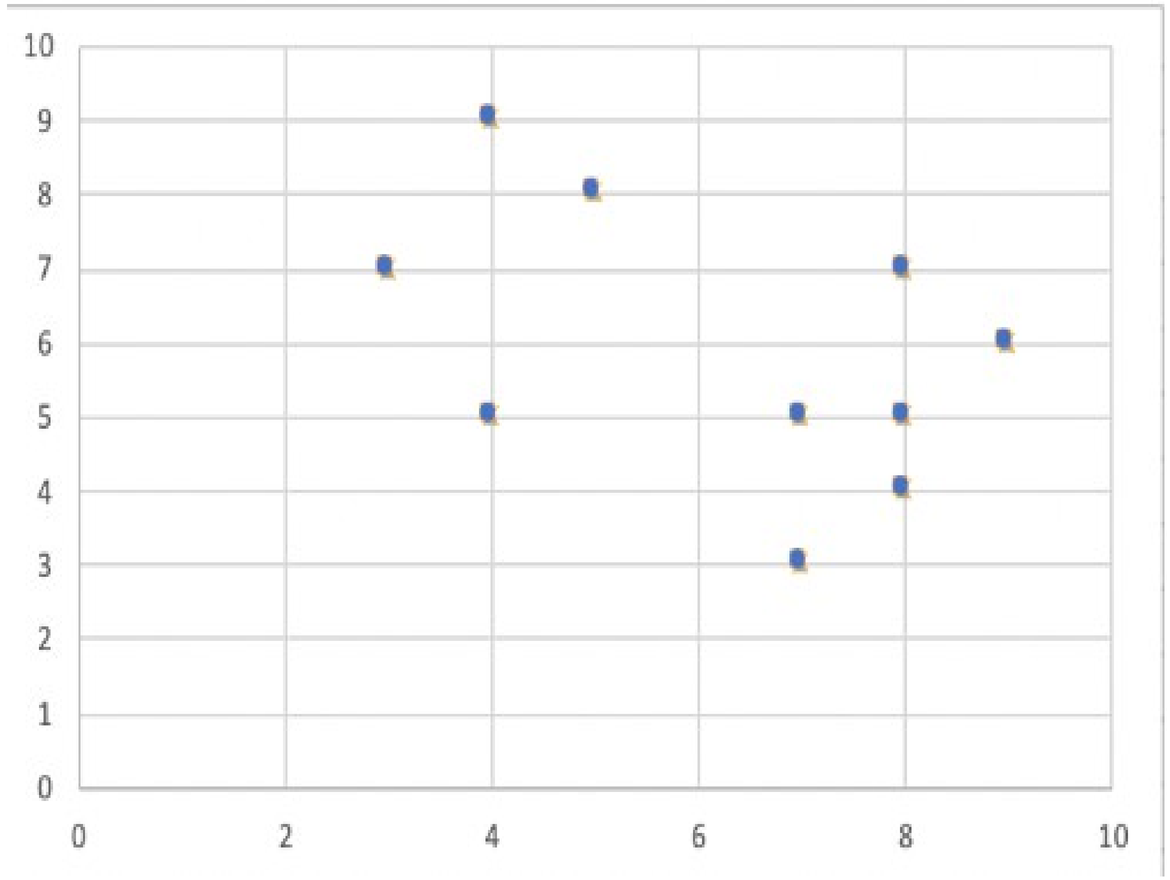
PAM- Partition Around Medoid

1. Initialize: select *k* random points out of the *n* data points as the medoids.
2. Associate each data point to the closest medoid by using any common distance metric methods.
3. While the cost decreases: For each medoid ‘m’, for each data point ‘o’ which is not a medoid:
    1. Swap m and o, associate each data point to the closest medoid, and recompute the cost.
    2. If the total cost is more than that in the previous step, undo the swap.

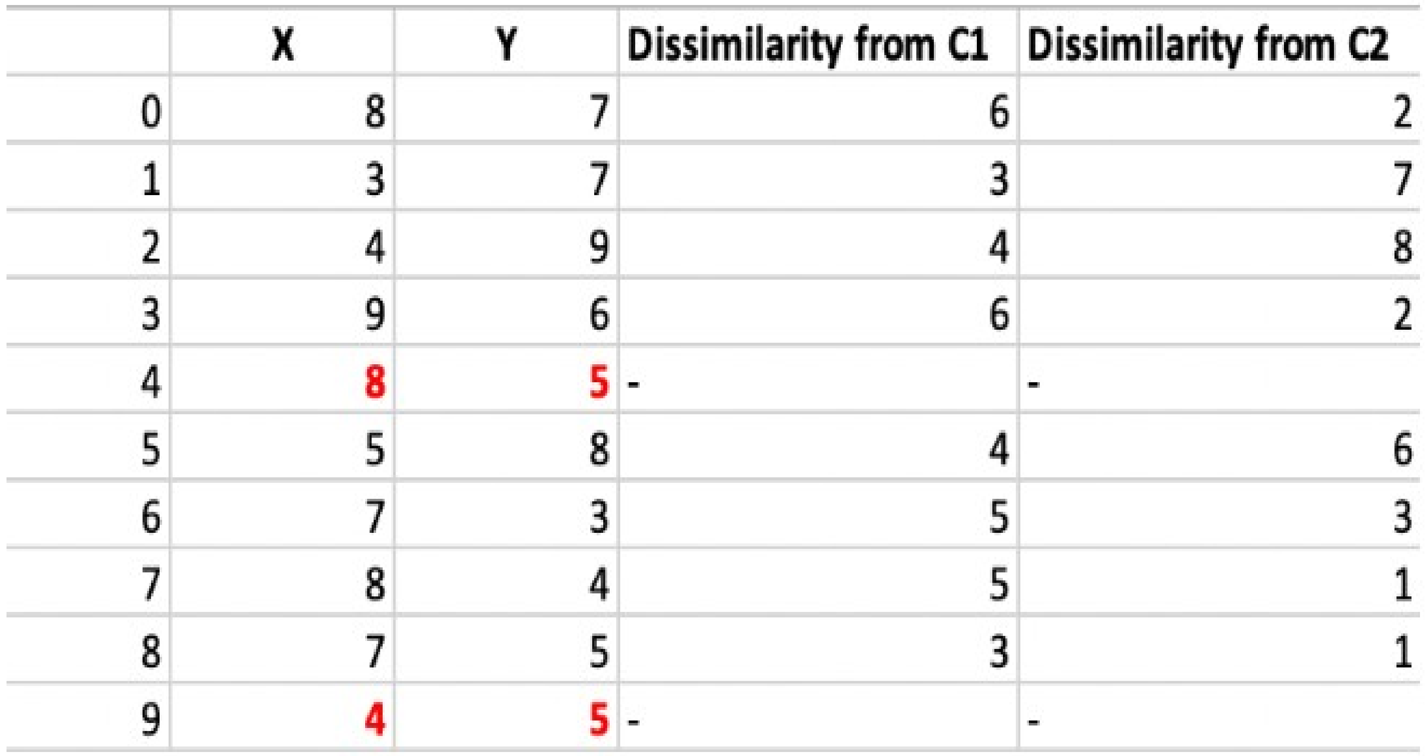
Example



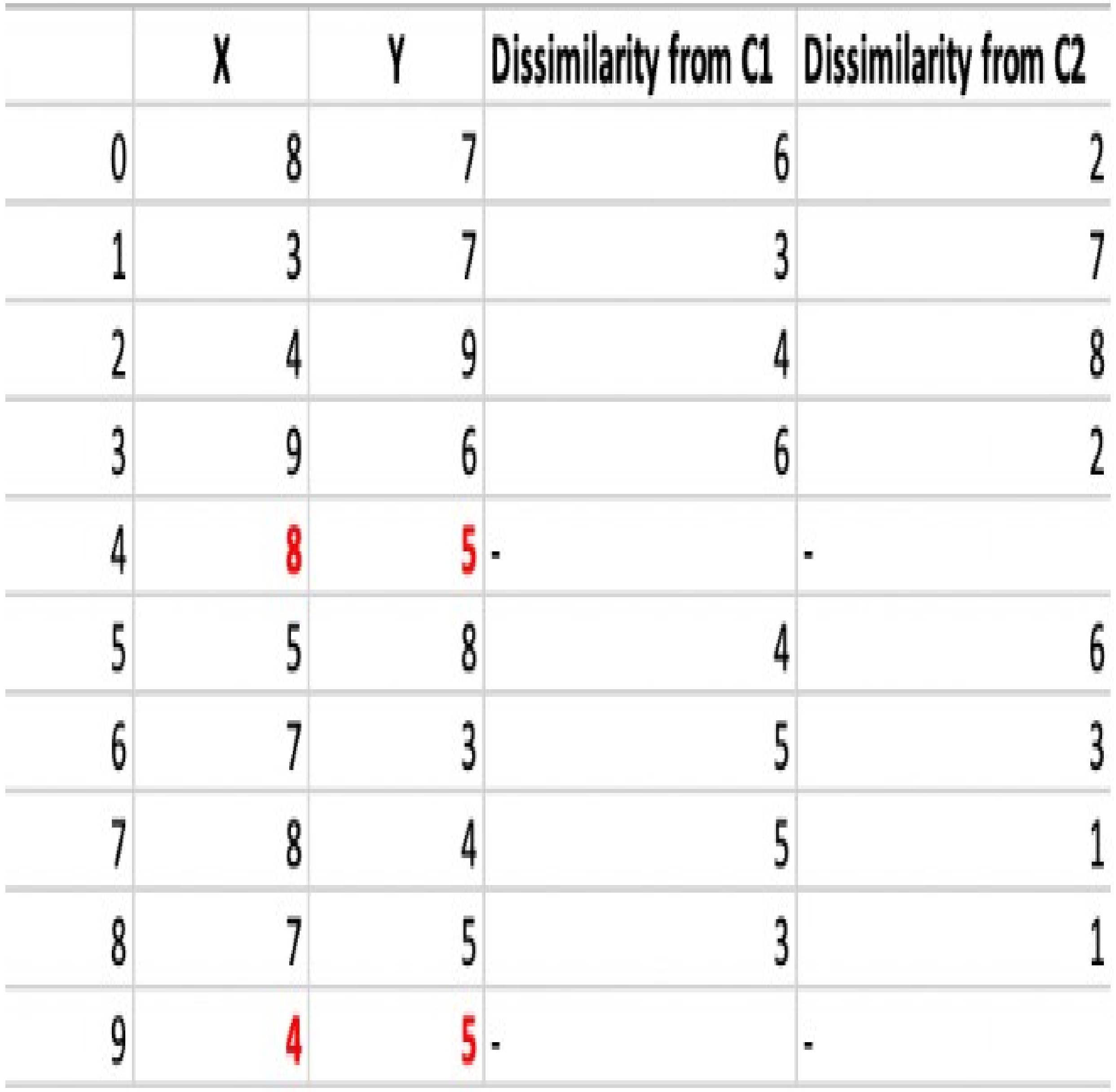
	X	Y
0	8	7
1	3	7
2	4	9
3	9	6
4	8	5
5	5	8
6	7	3
7	8	4
8	7	5
9	4	5



- **Step 1:** Let the randomly selected 2 medoids, so select k = 2, and let **C1 -(4, 5)** and **C2 -(8, 5)** are the two medoids.
- **Step 2: Calculating cost.** The dissimilarity of each non-medoid point with the medoids is calculated and

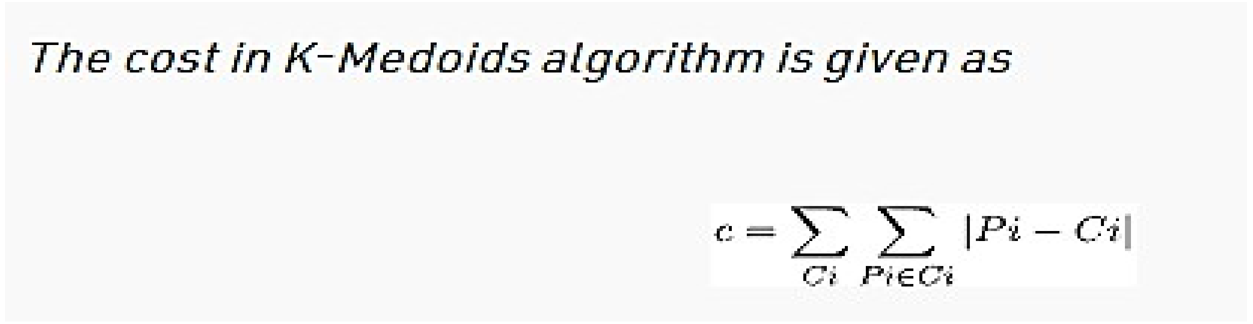


	X	Y	Dissimilarity from C1	Dissimilarity from C2
0	8	7	6	2
1	3	7	3	7
2	4	9	4	8
3	9	6	6	2
4	8	5	-	-
5	5	8	4	6
6	7	3	5	3
7	8	4	5	1
8	7	5	3	1
9	4	5	-	-



	X	Y	Dissimilarity from C1	Dissimilarity from C2
0	8	7	6	2
1	3	7	3	7
2	4	9	4	8
3	9	6	6	2
4	8	5	-	-
5	5	8	4	6
6	7	3	5	3
7	8	4	5	1
8	7	5	3	1
9	4	5	-	-

- Here we have used Manhattan distance formula.
- That formula tell that **Distance = |X1-X2| + |Y1-Y2|.**
- Each point is assigned to the cluster of that medoid whose dissimilarity is less.
- Points 1, 2, and 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The Cost = (3 + 4 + 4) + (3 + 1 + 1 + 2 + 2) = 20



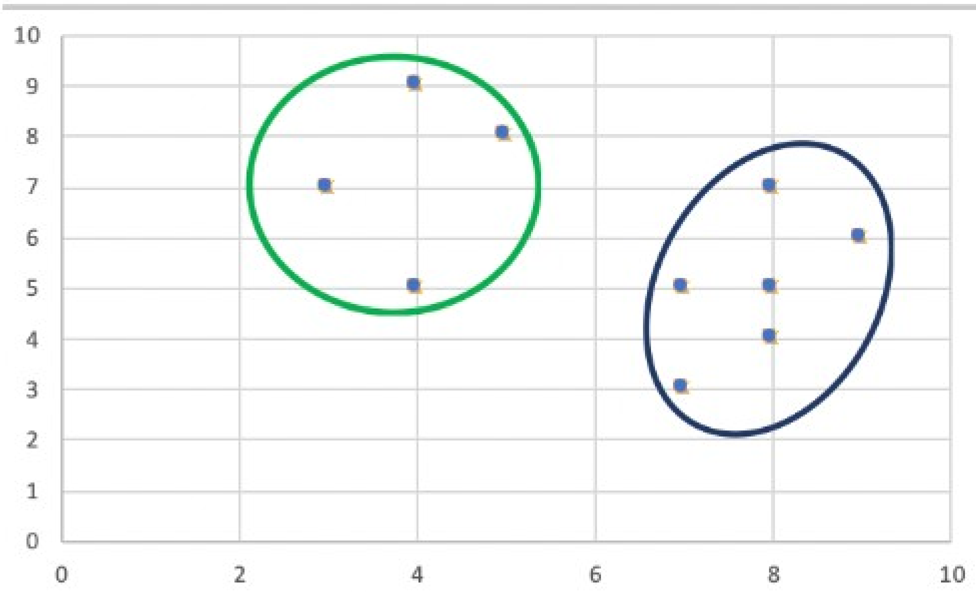
The cost in K-Medoids algorithm is given as

$$c = \sum_{Ci} \sum_{Pi \in Ci} |Pi - Ci|$$

- **Step 3: randomly select one non-medoid point and recalculate the cost.** Let the randomly selected point be (8, 4).
- The dissimilarity of each non-medoid point with the

	X	Y	Dissimilarity from C1	Dissimilarity from C2
0	8	7	6	3
1	3	7	3	8
2	4	9	4	9
3	9	6	6	3
4	8	5	4	1
5	5	8	4	7
6	7	3	5	2
7	**8**	**4**	-	-
8	7	5	3	2
9	**4**	**5**	-	-

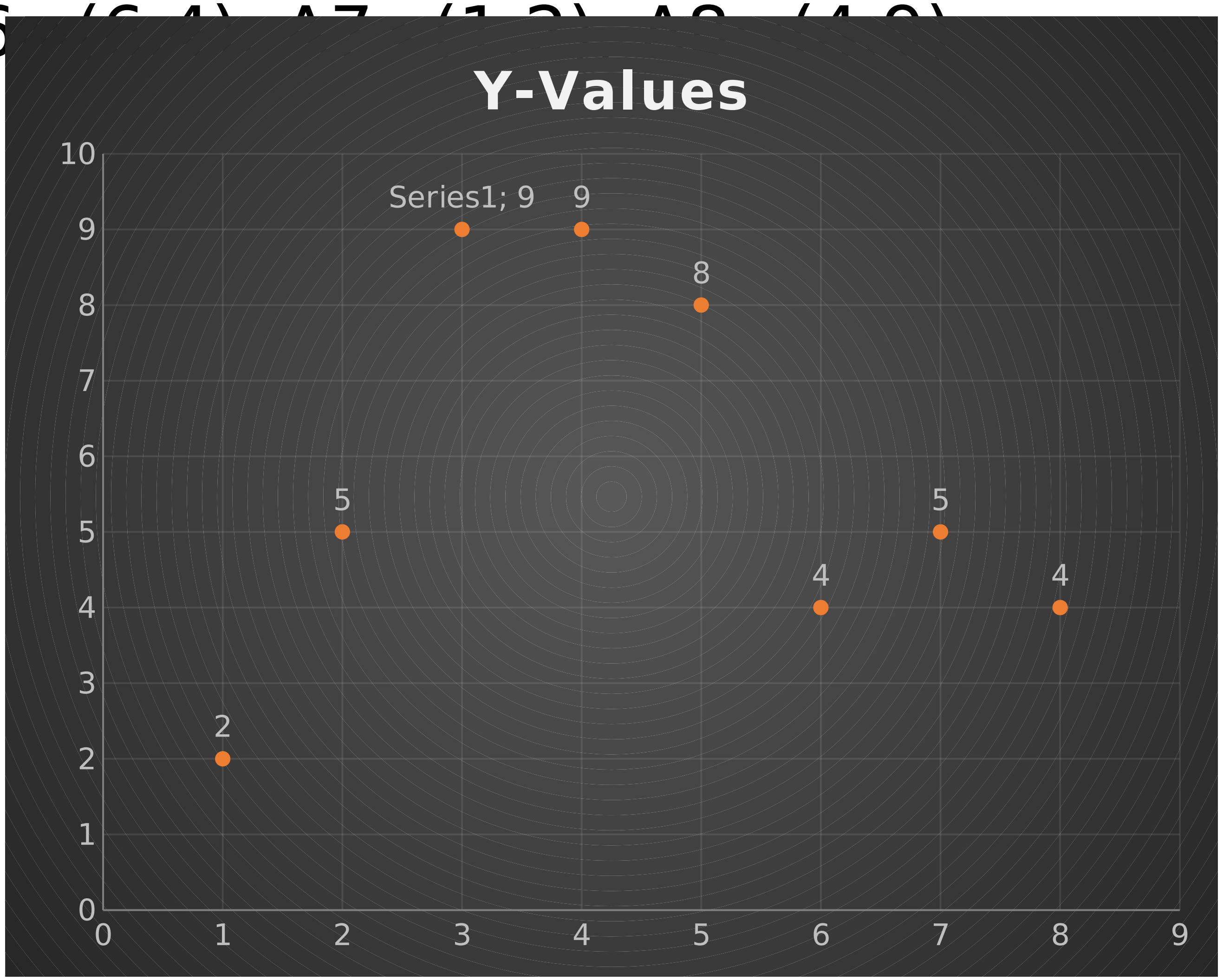
- Each point is assigned to that cluster whose dissimilarity is less. So, points 1, 2, and 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2.
- The New cost = (3 + 4 + 4) + (2 + 2 + 1 + 3 + 3) = 22 Swap Cost = New Cost – Previous Cost = 22 – 20 and **2 >0**
- As the swap cost is not less than zero, we undo the swap.
- Hence (4, 5) [illegible] ds.



Problem

- Illustrate the working of K medoid algorithm for the given dataset. A1=(3,9), A2=(2,5),   A3=(8,4), A4=(5,8), A5=(7,5), A6

A1	3	9
A2	2	5
A3	8	4
A4	5	8
A5	7	5
A6	6	4
A7	1	2
A8	4	9



Data	X	Y	Dist from C1	Dist from C2
A1	3	9	5	8
A2	2	5	-	-
A3	8	4	7	2
A4	5	8	6	5
A5	7	5	-	-
A6	6	4	5	2
A7	1	2	4	9
A8	4	9	6	7

Let C1= (2,5) and C2= (7,5)

Groups- G1=(A1, A7, A8)
G2=(A3, A4, A6)
Cost(G1)= 5+4+6=15
Cost(G2)=2+5+2=9
Total Cost= 24

Swap/ Change medoid

Data	X	Y	Dist from C1	Dist from C2
A1	3	9	-	-
A2	2	5	5	5
A3	8	4	10	2
A4	5	8	3	5
A5	7	5	-	-
A6	6	4	8	2
A7	1	2	9	9
A8	4	9	1	7

Let C1= (3,9) and C2= (7,5)

Groups- G1=(A4, A7, A8)
G2=(A2, A3, A6)
Cost(G1)= 3+9+8=20
Cost(G2)=5+2+2=9
New_Total Cost= 29

New_total cost> old total

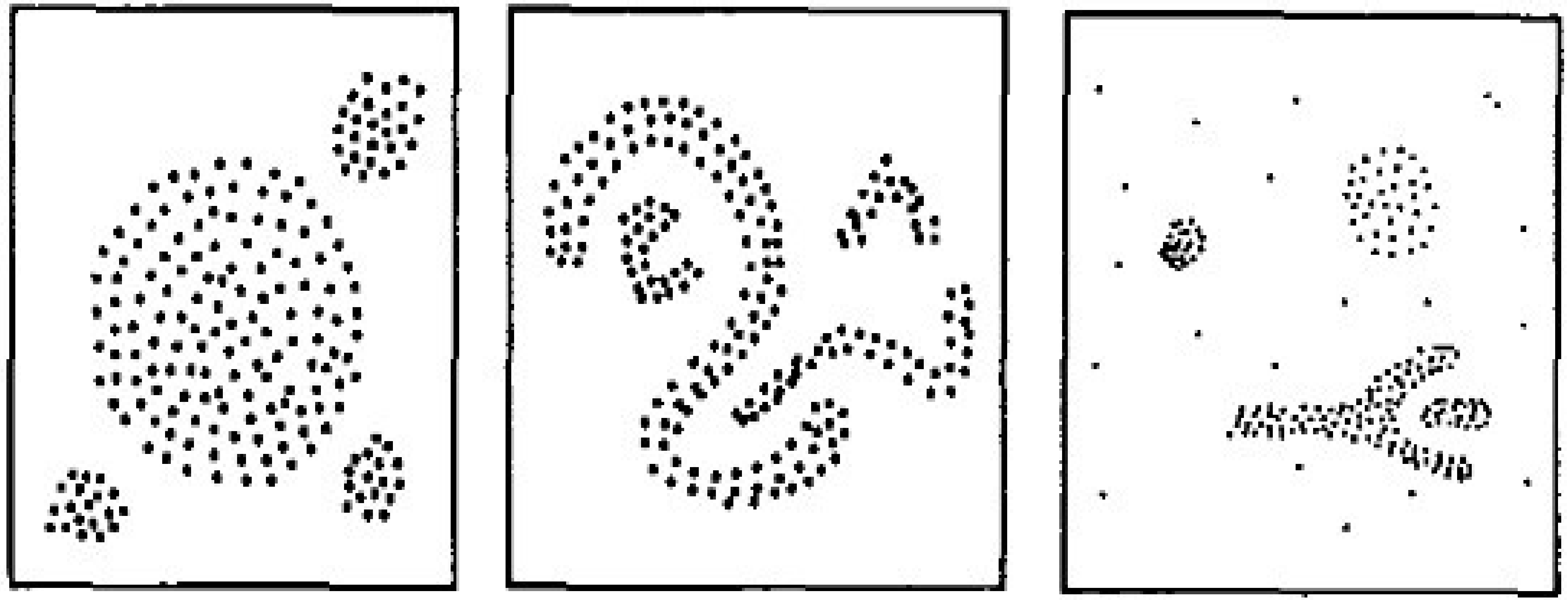
- Final Medoid Groups are

  G1=(A1, A7, A8)

  G2=(A3, A4, A6)

DBSCAN

- Density-Based Spatial Clustering of Applications with Noise.
- In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set.
- Objects in these sparse areas - that are required to separate clusters - are usually considered to be noise

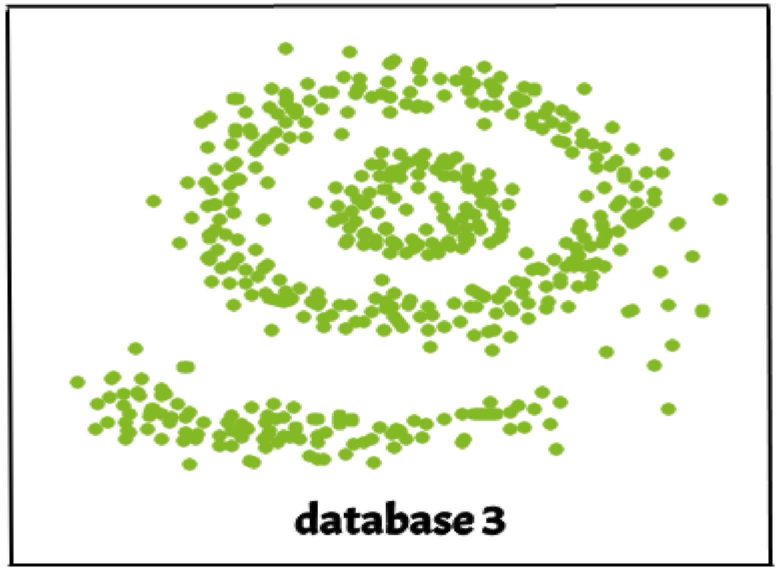


Clusters of points and noise points

Why DBSCAN?

- Partitioning methods are suitable only for compact and well-separated clusters.
- Moreover, they are also severely affected by the presence of noise and outliers in the data.
- Real life data may contain irregularities, like:

1. Clusters can be of arbitrary shape such as those shown in the figure below.
2. Data may contain noise.

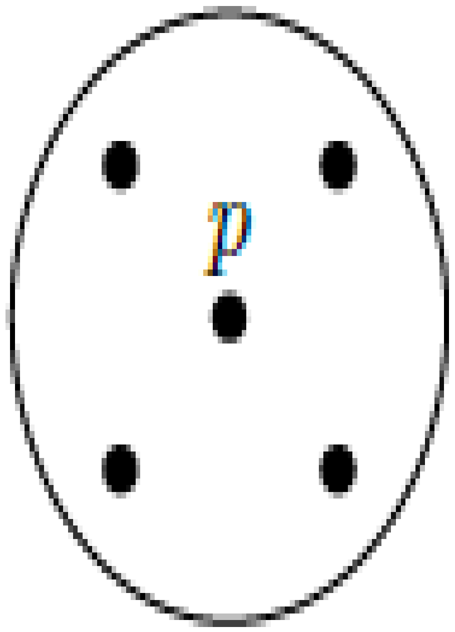


- **DBSCAN algorithm requires two parameters:**

1. **eps (Є**): It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to ‘eps’ then they are considered neighbors.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. The minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

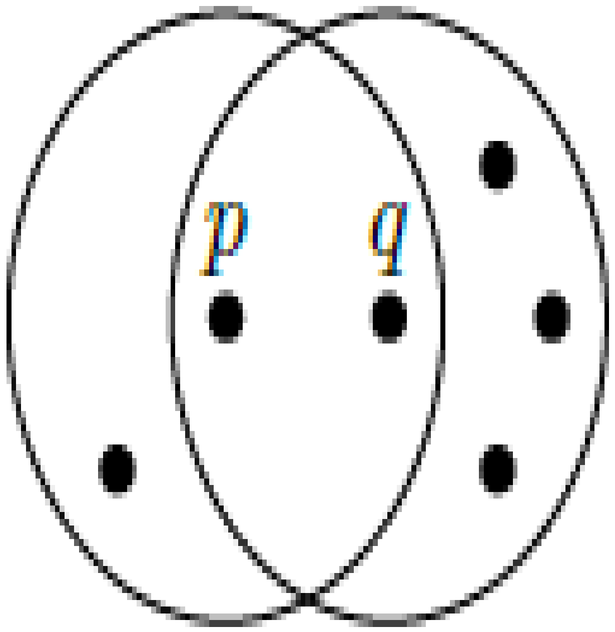
In this algorithm, we have 3 types of data points.

- **Core Point**: A point is a core point if it has more than MinPts points within Є distance.
- **Border Point**: A point which has fewer than MinPts within Є but it is in the neighborhood of a core point.
- **Noise or outlier**: A point which is not a core point or border point.



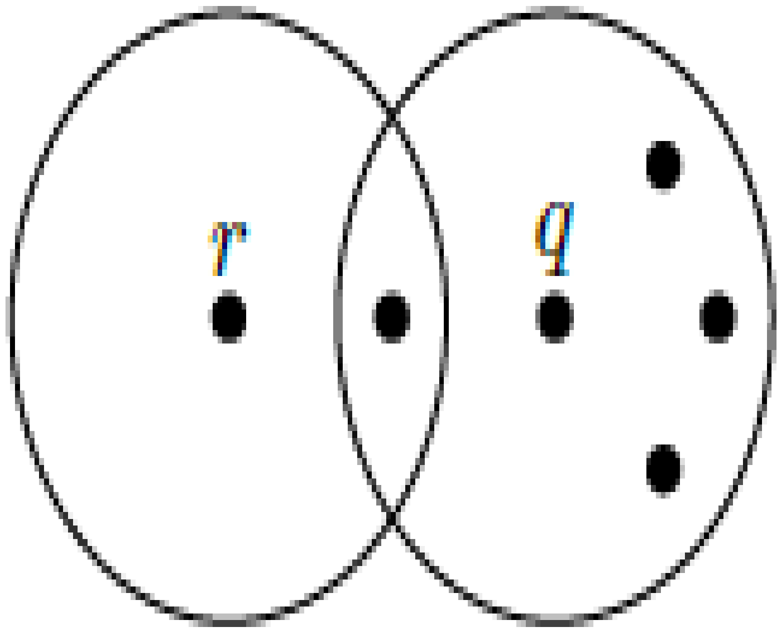
Let MinPts = 4

P- core point



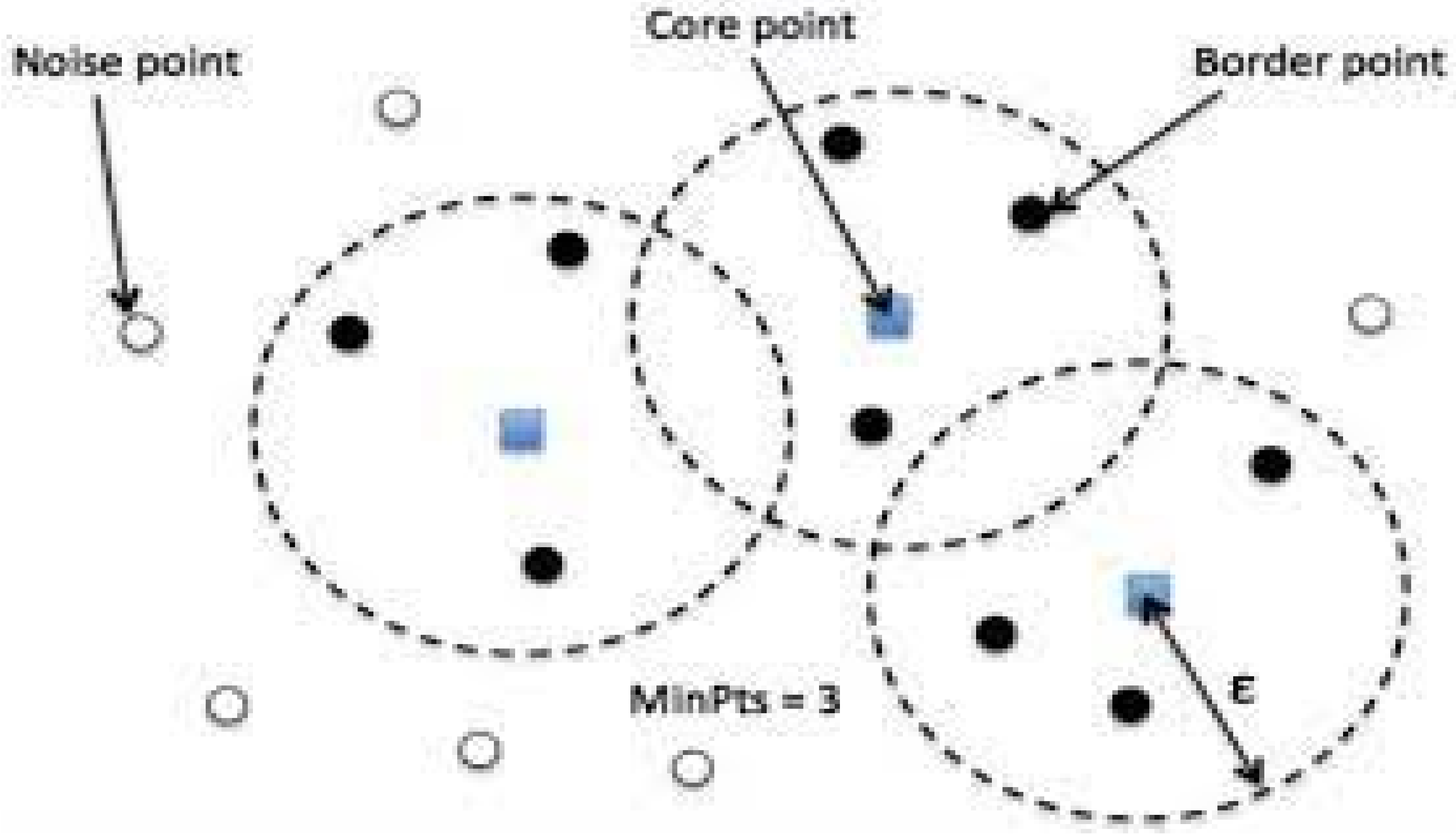
Let MinPts = 4

P- border point



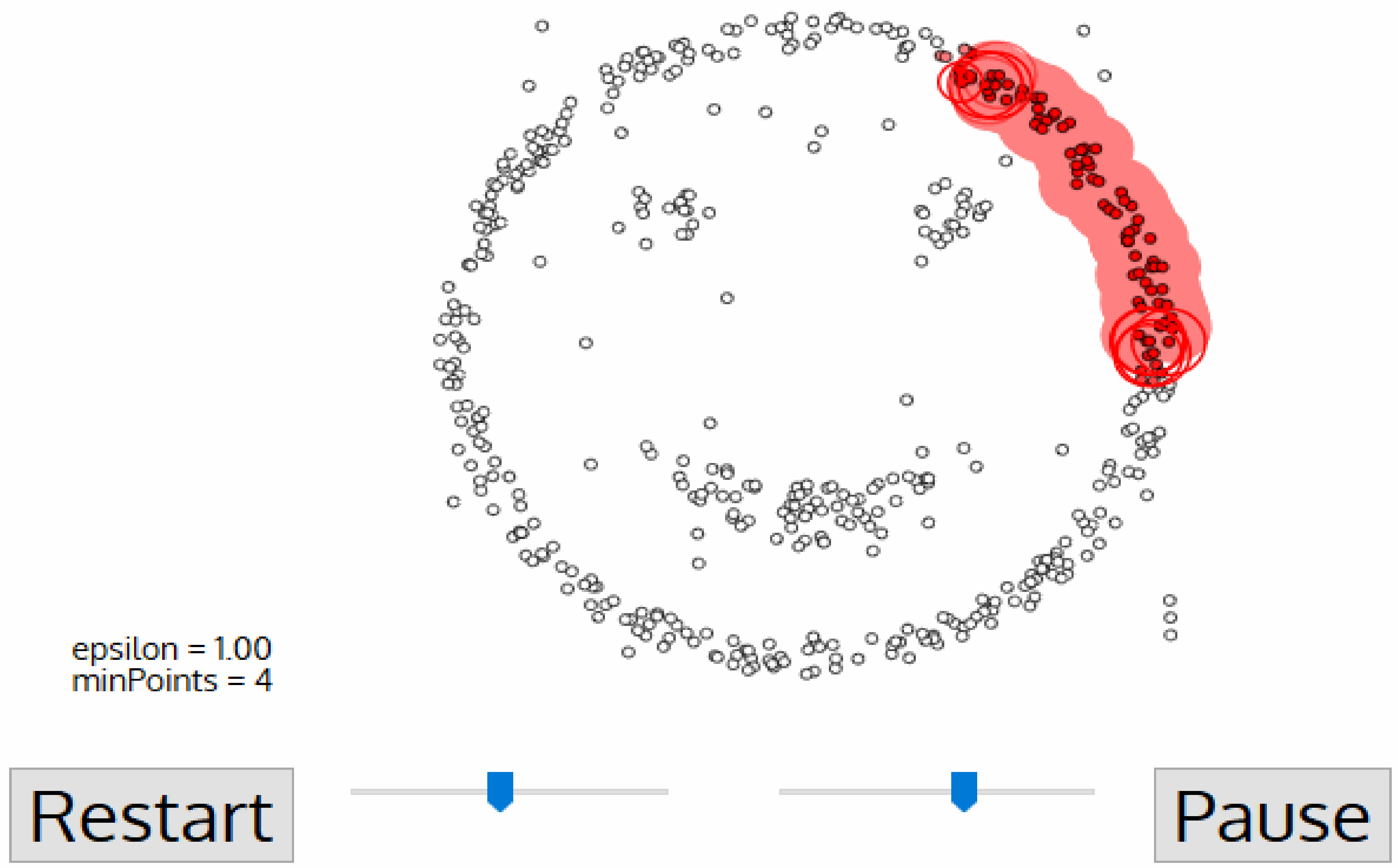
Let MinPts = 4

r- noise point


Noise point
Core point
Border point
MinPts = 3
ε

Algorithmic steps for DBSCAN clustering

- The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).
- If there are at least ‘minPoint’ points within a radius of ‘ε’ to the point then we consider all these points to be part of the same cluster.
- The clusters are then expanded by recursively repeating the neighborhood calculation for each neighboring point

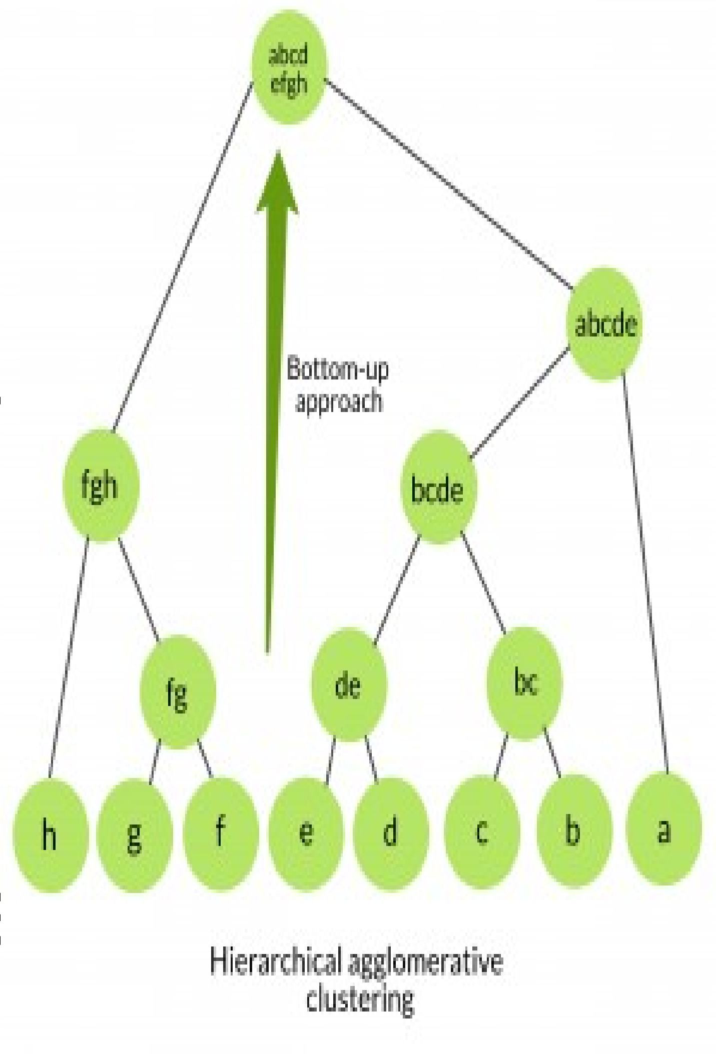


epsilon = 1.00
minPoints = 4

Categorical Clustering-ROCK

- Robust clustering Using Links
- ROCK belongs to the class of Agglomerative Hierarchical clusterir Algorithm

- ROCK works for categorical attribute



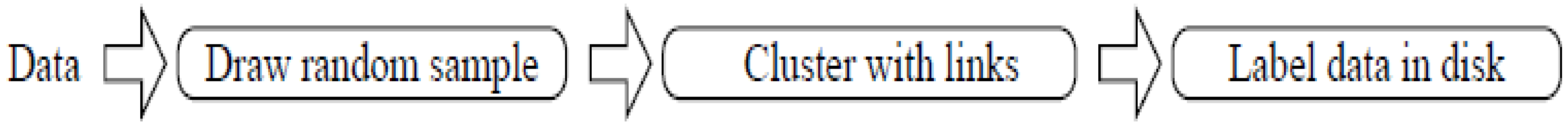


Fig. 2: Overview of ROCK

- The steps involved in clustering using ROCK are described in Figure. After drawing a random sample from the database, a hierarchical clustering algorithm that employs links is applied to the sampled points. Finally, the clusters involving only the sampled points are used to assign the remaining data points on disk to the appropriate clusters.

Measures used

- Neighbors : Given a threshold ϴ between 0 and 1, a pair of points pi and pj are defined to be neighbors if the following holds:

$$sim(p_i, p_j) \geq \theta$$

- Jaccard coefficient, for sim(T1; T2),

$$sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

- Links: *link(pi, pj)* to be the number of common neighbors between pi and pj
- Goodness Measure: For a pair of clusters Ci; Cj , let link[Ci, Cj] store the number of cross links between clusters Ci and Cj, Then, we define the goodness measure g(Ci, Cj) for merging clusters Ci, Cj as follows.

$$g(C_i, C_j) = \frac{link[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

where

```
procedure cluster($S, k$)
begin
1.  $link$ := compute_links($S$)
2.  for each $s \in S$ do
3.      $q[s]$ := build_local_heap($link, s$)
4.  $Q$ := build_global_heap($S, q$)
5.  while size($Q$) $> k$ do {
6.      $u$ := extract_max($Q$)
7.      $v$ := max($q[u]$)
8.      delete($Q, v$)
9.      $w$ := merge($u, v$)
10.     for each $x \in q[u] \cup q[v]$ do {
11.         $link[x, w] := link[x, u] + link[x, v]$
12.         delete($q[x], u$); delete($q[x], v$)
13.         insert($q[x], w, g(x, w)$); insert($q[w], x, g(x, w)$)
14.         update($Q, x, q[x]$)
15.     }
16.     insert($Q, w, q[w]$)
17.     deallocate($q[u]$); deallocate($q[v]$)
18. }
end
```

```
procedure compute_links($S$)
begin
1.  Compute $nbrlist[i]$ for every point $i$ in S
2.  Set $link[i, j]$ to be zero for all $i, j$
3.  for $i := 1$ to $n$ do {
4.      $N := nbrlist[i]$
5.      for $j := 1$ to $|N| - 1$ do
6.          for $l := j + 1$ to $|N|$ do
7.              $link[N[j], N[l]] := link[N[j], N[l]] + 1$
8.  }
end
```

Example

P1= {A,B,C,D}

P2={E,B,C}

P3={D,E,B}

P4={E,C,F}

Similarity threshold= 0.3

No of clusters=2

1) Similarity Table

	P1	P2	P3	P4
P1	1			
P2		1		
P3			1	
P4				1

,

Example

P1= {A,B,C,D}
P2={E,B,C}
P3={D,E,B}
P4={E,C,F}
Similarity threshold= 0.3
No of clusters=2

2) Adjacency Table

	P1	P2	P3	P4
P1	1	1	1	0
P2		1	1	1
P3			1	0
P4				1

,
,

Example

P1= {A,B,C,D}

P2={E,B,C}

P3={D,E,B}

P4={E,C,F}

Similarity threshold= 0.3

No of clusters=2

3) No. of Links/ common neighbours

$$No\,of\,Links(P_i, P_j) = Adj\,mat * Adj\,mat$$

	P1	P2	P3	P4
P1	-			
P2		-		
P3			-	
P4				-

4) Goodness Measure

$$g(C_i, C_j) = \frac{link[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

$$f(\Theta) = \frac{1-\theta}{1+\theta} = \frac{1-0.3}{1+0.3} = 0.57$$

Pair	Goodness Measure
,	1.35
,	1.35
,	0.45
,	1.35
,	0.90
,	0.45

Highest Equal value for 3 pairs .
Take any one pair and cluster them

Cluster
New Link table

	P1, P2	P3	P4
P1, P2	-		
P3		-	
P4			

	P1	P2	P3	P4
P1	-	3	3	1
P2		-	3	2
P3			-	1
P4				-

New Link Matrix

	P1, P2	P3	P4
P1, P2	-	6	3
P3		-	1
P4			

Goodness of Measure

Pair	Goodness Measure
,) ,	1.31
,) ,	0.66
,	0.22

Highest Value = 1.31
Cluster ,) , to new cluster , ,)

Final clusters
cluster 1: , ,)
cluster 2:

Algorithm stops because no. of desired clusters=2

THANK YOU