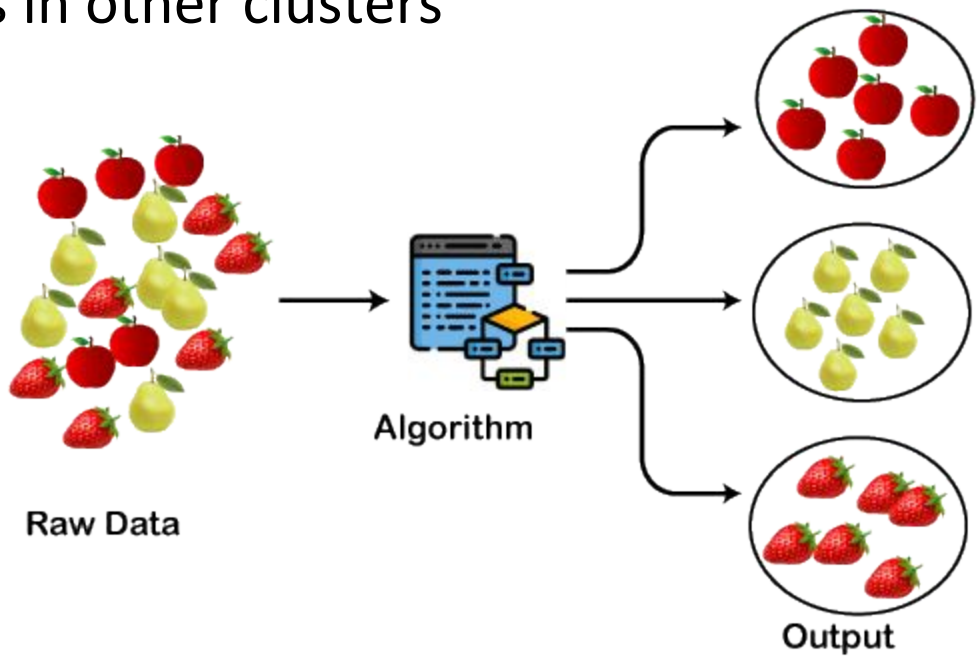
UNIT-6
Clustering

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University

Clustering

- Clustering is the process of grouping a set of data objects into multiple groups or clusters so that objects within a cluster have high similarity, but are very dissimilar to objects in other clusters



- Dissimilarities and similarities are assessed based on the attribute values describing the objects and often involve distance measure

Measuring Data Similarity and Dissimilarity

- In data mining applications, such as
  - clustering,
  - outlier analysis,
  - and nearest-neighbor classification

  we need ways to assess how alike or unalike objects are in comparison to one another.
- A cluster is a collection of data objects such that the objects within a cluster are similar to one another and dissimilar to the objects in other clusters.
- Similarity and dissimilarity measures, which are referred to as measures of **proximity**.
- Similarity and dissimilarity are related.

- We present **two data structures** that are commonly used to represent similarity and dis-similarity:
- **Data matrix** (used to store the data objects)
- **Dissimilarity matrix** (used to store dissimilarity values for pairs of objects).

- Suppose that we have n objects (e.g., persons, items, or courses) described by p attributes
- The objects are x1 = (x11,x12,...,x1p), x2 = (x21,x22,...,x2p), and so on, where xij is the value for object xi of the jth attribute.
- The objects may be tuples in a relational database, and are also referred to as data samples or feature vectors.

Data matrix (or *object-by-attribute structure*): This structure stores the n data objects in the form of a relational table, or n-by-p matrix (n objects $\times p$ attributes):

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}. \qquad (2.8)$$

Dissimilarity matrix (or *object-by-object structure*): This structure stores a collection of proximities that are available for all pairs of n objects. It is often represented by an n-by-n table:

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}, \qquad (2.9)$$

Where

- d(i, j) is the measured dissimilarity or "difference" between objects i and j.
- d(i, i) = 0; that is, the difference between an object and itself is 0.
- d(i, j) = d(j, i). (For readability, we do not show the d(j, i) entries;

Measures of similarity can often be expressed as a function of measures of dissimilarity. For example, for nominal data,

$$sim(i, j) = 1 - d(i, j), \tag{2.10}$$

- Many clustering and nearest-neighbor algorithms operate on a dissimilarity matrix.
- Data in the form of a data matrix can be transformed into a dissimilarity matrix before applying such algorithms.

Proximity Measures for Nominal Attributes

- Let the number of states of a nominal attribute be M.
- The states can be denoted by letters, symbols, or a set of integers, such as 1, 2,..., M.
- Notice that such integers are used just for data handling and do not represent any specific ordering

"How is dissimilarity computed between objects described by nominal attributes?"
The dissimilarity between two objects i and j can be computed based on the ratio of mismatches:

$$d(i, j) = \frac{p - m}{p}, \tag{2.11}$$

Where

- m is the number of matches (i.e., the number of attributes for which i and j are in the same state)
- p is the total number of attributes describing the objects.

Proximity Measures for Nominal Attributes

Calculate the dissimilarity Matrix

Object Identifier	test-1 (nominal)
1	code A
2	code B
3	code C
4	code A

Object Identifier	test-1 (nominal)
1	code A
2	code B
3	code C
4	code A

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix}.$$

Since here we have one nominal attribute, *test-1*, we set $p = 1$ in Eq. (2.11) so that $d(i, j)$ evaluates to 0 if objects i and j match, and 1 if the objects differ. Thus, we get

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

From this, we see that all objects are dissimilar except objects 1 and 4 (i.e., $d(4,1) = 0$).

Proximity Measures for Binary Attributes

- One approach involves computing a dissimilarity matrix from the given binary details we have the 2 × 2 contingency table

Contingency Table for Binary Attributes

		Object j		
		1	0	sum
	1	q	r	$q+r$
Object i	0	s	t	$s+t$
	sum	$q+s$	$r+t$	p

NOTE:-

- A binary variable is symmetric if both of its states are equally valuable, that is, there is no preference on which outcome should be coded as 1.
- A binary variable is asymmetric if the outcome of the states are not equally important, such as positive or negative outcomes of a disease

- If objects i and j are described by symmetric binary attributes, then the dissimilarity between i and j is

$$d(i, j) = \frac{r + s}{q + r + s + t}.$$

- If objects i and j are described by asymmetric binary attributes, then the dissimilarity between i and j is

$$d(i, j) = \frac{r + s}{q + r + s}.$$

Example 2.18 **Dissimilarity between binary attributes.** Suppose that a patient record table (Table 2.4) contains the attributes *name*, *gender*, *fever*, *cough*, *test-1*, *test-2*, *test-3*, and *test-4*, where *name* is an object identifier, *gender* is a symmetric attribute, and the remaining attributes are asymmetric binary.

For asymmetric attribute values, let the values *Y* (*yes*) and *P* (*positive*) be set to 1, and the value *N* (*no* or *negative*) be set to 0. Suppose that the distance between objects

Table 2.4 Relational Table Where Patients Are Described by Binary Attributes

name	*gender*	*fever*	*cough*	*test-1*	*test-2*	*test-3*	*test-4*
Jack	M	Y	N	P	N	N	N
Jim	M	Y	Y	N	N	N	N
Mary	F	Y	N	P	N	P	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

(patients) is computed based only on the asymmetric attributes. According to Eq. (2.14), the distance between each pair of the three patients—Jack, Mary, and Jim—is

$$d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67,$$

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33,$$

$$d(Jim, Mary) = \frac{1+2}{1+1+2} = 0.75.$$

These measurements suggest that Jim and Mary are unlikely to have a similar disease because they have the highest dissimilarity value among the three pairs. Of the three patients, Jack and Mary are the most likely to have a similar disease. ■

Proximity Measures for Numeric Attribute

These measures include

- Euclidean
- Manhattan

- The most popular distance measure is Euclidean distance
- Let i = (xi1, xi2,..., xip) and j = (xj1, xj2,..., xjp) be two objects described by p numeric attributes.
- The **Euclidean distance** between objects i and j is defined as

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}.$$

- The **Manhattan distance**, between objects i and j is defined as

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|.$$

Euclidean distance and Manhattan distance. Let $\boldsymbol{x_1} = (1, 2)$ and $\boldsymbol{x_2} = (3, 5)$ represent two objects as shown in Figure 2.23. The Euclidean distance between the two is $\sqrt{2^2 + 3^2} = 3.61$. The Manhattan distance between the two is $2 + 3 = 5$. ■

Proximity Measures for Ordinal Attributes

- The values of an ordinal attribute have a meaningful order
- An example includes the sequence small, medium, large for a size attribute
- Ordinal attributes may also be obtained from the discretization of numeric attributes
- Let M represent the number of possible states that an ordinal attribute can have. These ordered states define the ranking 1,..., Mf .

1. The value of f for the ith object is x_{if}, and f has M_f ordered states, representing the ranking $1, \ldots, M_f$. Replace each x_{if} by its corresponding rank, $r_{if} \in \{1, \ldots, M_f\}$.

2. Since each ordinal attribute can have a different number of states, it is often necessary to map the range of each attribute onto [0.0, 1.0] so that each attribute has equal weight. We perform such data normalization by replacing the rank r_{if} of the ith object in the fth attribute by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}. \tag{2.21}$$

3. Dissimilarity can then be computed using any of the distance measures described in Section 2.4.4 for numeric attributes, using z_{if} to represent the f value for the ith object.

Example 2.21 **Dissimilarity between ordinal attributes.** Suppose that we have the sample data shown earlier in Table 2.2, except that this time only the *object-identifier* and the continuous ordinal attribute, *test-2*, are available. There are three states for *test-2*: *fair*, *good*, and *excellent*, that is, $M_f = 3$. For step 1, if we replace each value for *test-2* by its rank, the four objects are assigned the ranks 3, 1, 2, and 3, respectively. Step 2 normalizes the ranking by mapping rank 1 to 0.0, rank 2 to 0.5, and rank 3 to 1.0. For step 3, we can use, say, the Euclidean distance (Eq. 2.16), which results in the following dissimilarity matrix:

Object Identifier	test-1 (nominal)	test-2 (ordinal)
1	code A	excellent
2	code B	fair
3	code C	good
4	code A	excellent

$$\begin{bmatrix} 0 & & & \\ 1.0 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}.$$

Therefore, objects 1 and 2 are the most dissimilar, as are objects 2 and 4 (i.e., $d(2,1) = 1.0$ and $d(4,2) = 1.0$). This makes intuitive sense since objects 1 and 4 are both *excellent*. Object 2 is *fair*, which is at the opposite end of the range of values for *test-2*. ■

Similarity values for ordinal attributes can be interpreted from dissimilarity as $sim(i,j) = 1 - d(i,j)$.

Clustering

Unsupervised learning: no predefined classes (i.e., *learning by observations*)

Typical applications

- As a stand-alone tool to get insight into data distribution
- As a preprocessing step for other algorithms

- In general, the major fundamental clustering methods can be classified into the following categories,
- **Partitioning methods**
- **Hierarchical methods**
- **Density-based methods**
- **Grid-based methods**

Partitioning Method

- Formally, given a data set, D, of n objects, and k, the number of clusters to form, a partitioning algorithm organizes the objects into k partitions ($k \leq n$), where each partition represents a cluster
- The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters in terms of the data set attributes.
- commonly used partitioning methods—**k-means** and **k-medoids**

Method	General Characteristics
Partitioning methods	– Find mutually exclusive clusters of spherical shape – Distance-based – May use mean or medoid (etc.) to represent cluster center – Effective for small- to medium-size data sets

K-Means

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

Figure 10.2 The *k*-means partitioning algorithm.

Use the k-means algorithm and Euclidean distance to cluster the following 10 examples into 3 clusters

X1(2,10);
X2(2, 5);
X3(8, 4);
X4(9,4);
Y1(5, 8);
Y2(7,5);
Y3(6,4);
Zl(l,2);
Z2(4,9);
Z3(6,10).

Initial seed are Xl, X4, and Z2.

Run k-means algorithm . At the end of each iteration, show:

(i)The new clusters (ii)The centers of the new clusters.

Draw a 10 by 10 space with all the 10 points and show the clusters after each iteration.

Disadvantage

- The k-means algorithm is **sensitive to outliers** because such objects are far away from the majority of the data, and thus, when assigned to a cluster, they can dramatically distort the mean value of the cluster

Concept:-

- The difference between an object p $\in$ Ci and ci , the representative of the cluster, is measured by dist(p,ci), where dist(x,y) is the Euclidean distance between two points x and y. The quality of cluster Ci can be measured by **the within cluster variation**, which is the sum of squared error between all objects in Ci and the centroid ci , defined as

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(\boldsymbol{p}, \boldsymbol{c_i})^2,$$

A drawback of *k*-means. Consider six points in 1-D space having the values $1, 2, 3, 8, 9, 10$, and 25, respectively. Intuitively, by visual inspection we may imagine the points partitioned into the clusters $\{1, 2, 3\}$ and $\{8, 9, 10\}$, where point 25 is excluded because it appears to be an outlier. How would *k*-means partition the values? If we apply *k*-means using $k = 2$ and Eq. (10.1), the partitioning $\{\{1, 2, 3\}, \{8, 9, 10, 25\}\}$ has the within-cluster variation

$$(1-2)^2 + (2-2)^2 + (3-2)^2 + (8-13)^2 + (9-13)^2 + (10-13)^2 + (25-13)^2 = 196,$$

given that the mean of cluster $\{1, 2, 3\}$ is 2 and the mean of $\{8, 9, 10, 25\}$ is 13. Compare this to the partitioning $\{\{1, 2, 3, 8\}, \{9, 10, 25\}\}$, for which *k*-means computes the within-cluster variation as

$$(1-3.5)^2 + (2-3.5)^2 + (3-3.5)^2 + (8-3.5)^2 + (9-14.67)^2 + (10-14.67)^2 + (25-14.67)^2 = 189.67,$$

given that 3.5 is the mean of cluster $\{1, 2, 3, 8\}$ and 14.67 is the mean of cluster $\{9, 10, 25\}$. The latter partitioning has the lowest within-cluster variation; therefore, the k-means method assigns the value 8 to a cluster different from that containing 9 and 10 due to the outlier point 25. Moreover, the center of the second cluster, 14.67, is substantially far from all the members in the cluster. ∎

K-Medoid Clustering

- "How can we modify the k-means algorithm to diminish such sensitivity to outliers?"
- Instead of taking the mean value of the objects in a cluster as a reference point, we can **pick actual objects to represent the clusters**, using one representative object per cluster.
- Each remaining object is assigned to the cluster of which the representative object is the most similar.
- The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object p and its corresponding representative object
- That is, an absolute-error criterion is used, defined as

$$E = \sum_{i=1}^{k} \sum_{\boldsymbol{p} \in C_i} dist(\boldsymbol{p}, \boldsymbol{o}_i),$$

- The initial representative objects (called seeds) are chosen arbitrarily.
- We consider whether replacing a representative object by a nonrepresentative object would improve the clustering quality.
- All the possible replacements are tried out.
- The iterative process of replacing representative objects by other objects continues until the quality of the resulting clustering cannot be improved by any replacement.
- This quality is measured by a cost function of the average dissimilarity between an object and the representative object of its cluster

Algorithm: *k*-medoids. PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, *S*, of swapping representative object, o_j, with o_{random};
(6) **if** $S < 0$ **then** swap o_j with o_{random} to form the new set of *k* representative objects;
(7) **until** no change;

PAM, a *k*-medoids partitioning algorithm.

Use the k-Medoid algorithm and Manhatan distance to cluster the following 10 examples into 3 clusters X1(3,11); X2(3, 6); X3(9, 5); X4(10,5); Y1(6, 9); Y2(8,6); Y3(7,5); Zl(2,3); Z2(5,10); Z3(7,11). Initial seed are Xl, X4, and Z2. Run k-Medoid algorithm for 2 iteration. At the end of each iteration, show:

(i) The new clusters and the centers of the new clusters.

Draw a 10 by 10 space with all the 10 points and show the clusters after each iteration.

- However, the complexity of each iteration in the k-medoids algorithm is O(k(n − k) 2). For large values of n and k, such computation becomes very costly, and much more costly than the k-means method.
- Both methods require the user to specify k, the number of clusters

“How can we scale up the k-medoids method?”

- A typical k-medoids partitioning algorithm like PAM works effectively for small data sets, but does not scale well for large data sets.
- To deal with larger data sets, a sampling-based method called CLARA (Clustering LARge Applications) can be used.
- Instead of taking the whole data set into consideration, CLARA uses a random sample of the data set.
- The PAM algorithm is then applied to compute the best medoids from the sample
- Ideally, the sample should closely represent the original data set.

Hierarchical Clustering

- A hierarchical clustering method works by grouping data objects into a hierarchy or “tree” of clusters
- Representing data objects in the form of a hierarchy is useful for data summarization and visualization.
- A tree structure called a **dendrogram** is commonly used to represent the process of hierarchical clustering.

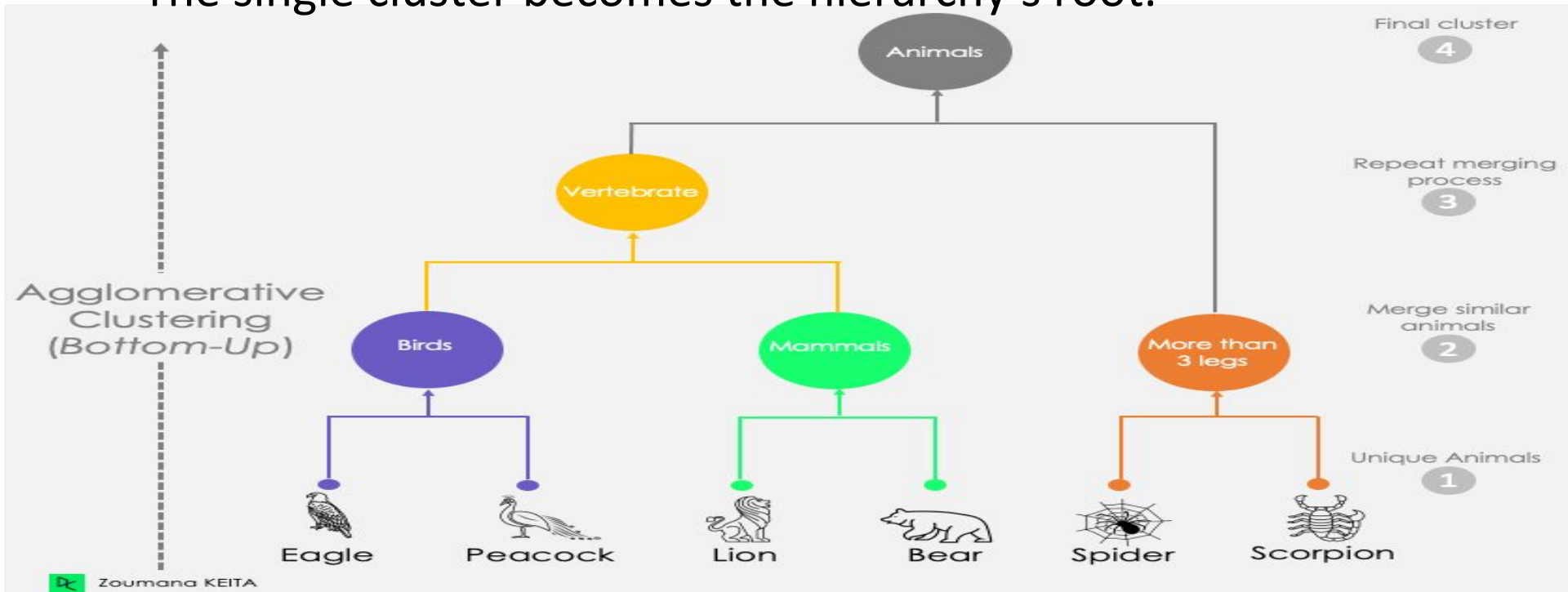
2 type Hierarchical Clustering

- **Agglomerative hierarchical clustering method**
- **Divisive hierarchical clustering method**

Hierarchical methods	– Clustering is a hierarchical decomposition (i.e., multiple levels) – Cannot correct erroneous merges or splits – May incorporate other techniques like microclustering or consider object “linkages”

Agglomerative hierarchical clustering method

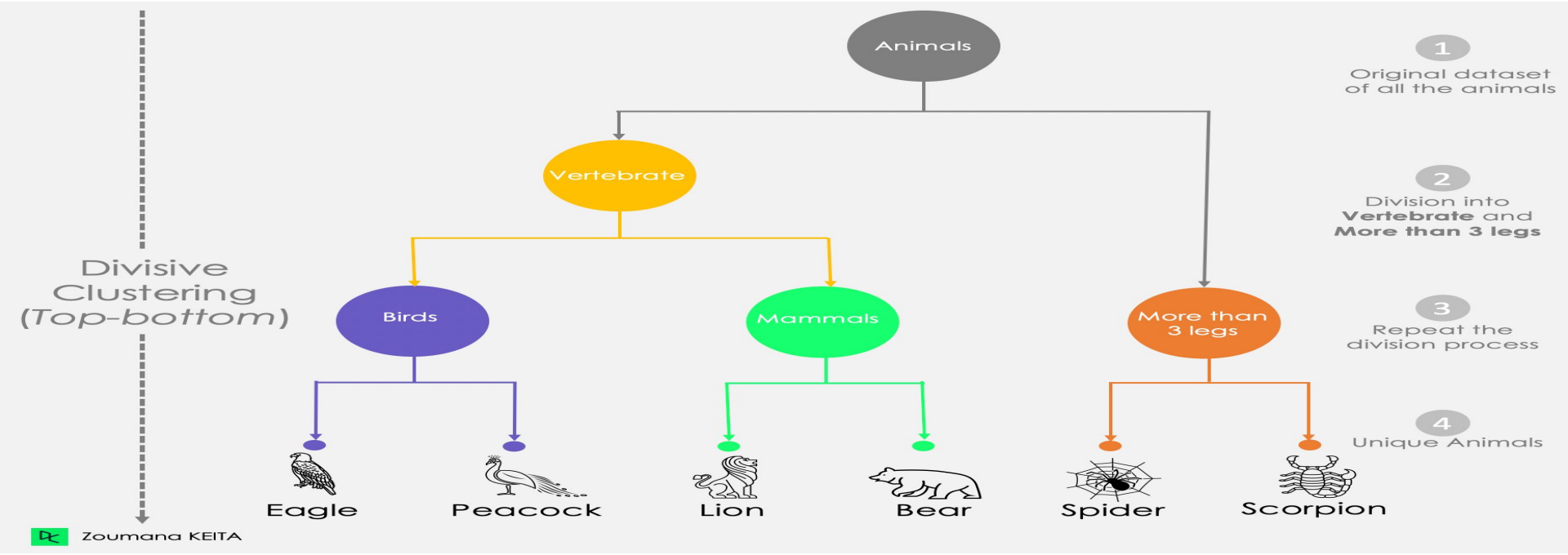
- An agglomerative hierarchical clustering method uses a bottom-up strategy.
- It typically starts by letting each object form its own cluster and iteratively merges clusters into larger and larger clusters, until all the objects are in a single cluster or certain termination conditions are satisfied.
- The single cluster becomes the hierarchy's root.



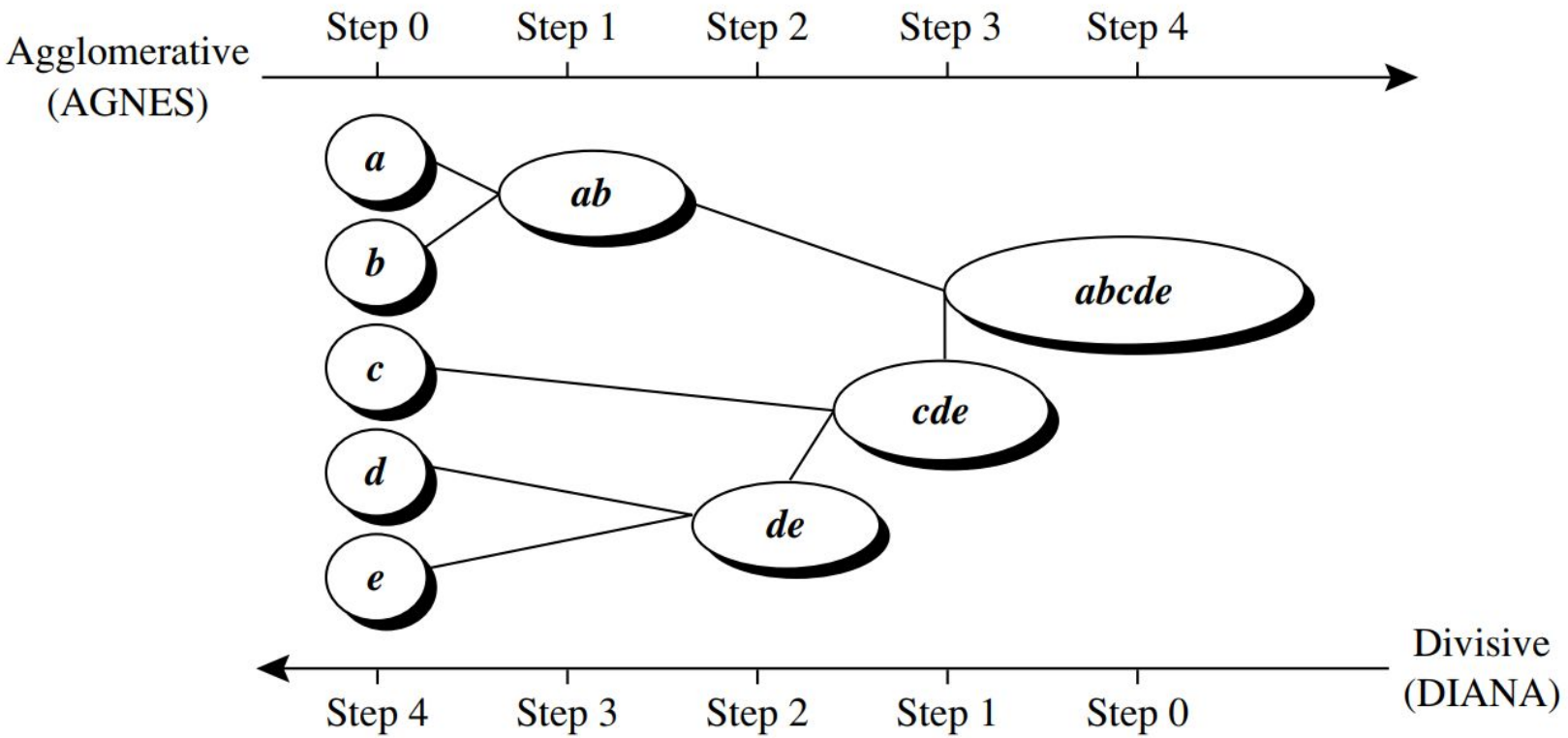
- For the merging step, it finds the two clusters that are closest to each other (according to some similarity measure),
- and combines the two to form one cluster. Because two clusters are merged per iteration, where each cluster contains at least one object,
- An agglomerative method requires at most n iterations.

Divisive hierarchical clustering

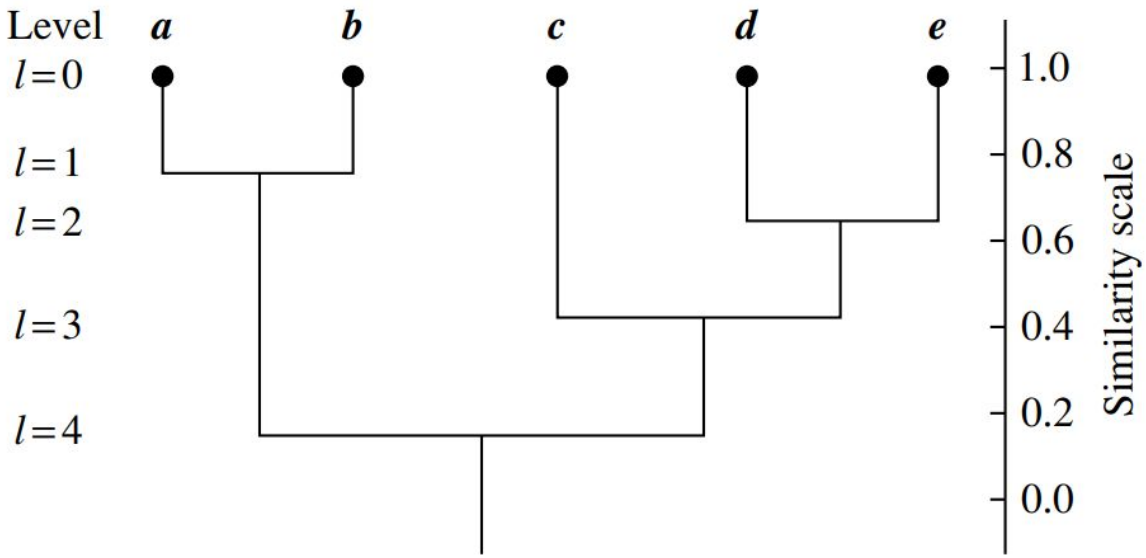
- A divisive hierarchical clustering method employs a top-down strategy.
- It starts by placing all objects in one cluster, which is the hierarchy's root.
- It then divides the root cluster into several smaller subclusters, and recursively partitions those clusters into smaller ones.



- The partitioning process continues until each cluster at the lowest level is coherent enough—either containing only one object, or the objects within a cluster are sufficiently similar to each other



0.6 Agglomerative and divisive hierarchical clustering on data objects $\{a, b, c, d, e\}$.



7 Dendrogram representation for hierarchical clustering of data objects $\{a, b, c, d, e\}$.

Distance Measures in cluster

Four widely used measures for distance between clusters are as follows, where $|\boldsymbol{p} - \boldsymbol{p}'|$ is the distance between two objects or points, $\boldsymbol{p}$ and $\boldsymbol{p}'$; $\boldsymbol{m_i}$ is the mean for cluster, C_i; and n_i is the number of objects in C_i. They are also known as *linkage measures.*

Minimum distance: $$dist_{min}(C_i, C_j) = \min_{\boldsymbol{p} \in C_i, \boldsymbol{p}' \in C_j} \{|\boldsymbol{p} - \boldsymbol{p}'|\} \quad (10.3)$$

Maximum distance: $$dist_{max}(C_i, C_j) = \max_{\boldsymbol{p} \in C_i, \boldsymbol{p}' \in C_j} \{|\boldsymbol{p} - \boldsymbol{p}'|\} \quad (10.4)$$

Mean distance: $$dist_{mean}(C_i, C_j) = |\boldsymbol{m_i} - \boldsymbol{m_j}| \quad (10.5)$$

Average distance: $$dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{\boldsymbol{p} \in C_i, \boldsymbol{p}' \in C_j} |\boldsymbol{p} - \boldsymbol{p}'| \quad (10.6)$$

HAC is represented by Three Technique

Single Linkage:
This is the distance between closest member of two cluster
it is sometimes called a nearest-neighbor clustering algorithm

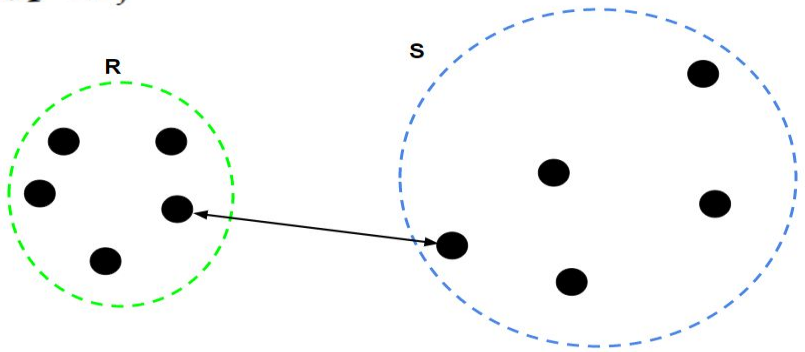
Complete Linkage
This is the distance between farthest member of two cluster
it is sometimes called a farthest-neighbor clustering algorithm

Average Linkage
This is the average distance between member of two cluster

Single Linkage

- The clusters are then merged step-by-step according to some criterion.
- For example, clusters C1 and C2 may be merged if an object in C1 and an object in C2 form the minimum Euclidean distance between any two objects from different clusters.
- This is a single-linkage approach in that the similarity between two clusters is measured by the similarity of the closest pair of data points belonging to different clusters.
- The cluster-merging process repeats until all the objects are eventually merged to form one cluster

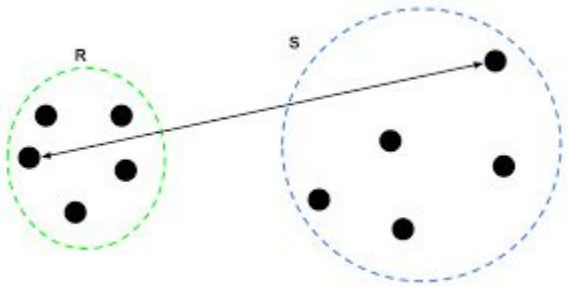
Minimum distance: $dist_{min}(C_i, C_j) = \min_{\boldsymbol{p} \in C_i, \boldsymbol{p}' \in C_j} \{|\boldsymbol{p} - \boldsymbol{p}'|\}$



*

Complete Linkage

- When an algorithm uses the maximum distance, dmax(Ci ,Cj), to measure the distance between clusters, it is sometimes called a **farthest-neighbor clustering algorithm**.
- If the clustering process is terminated when the maximum distance between nearest clusters exceeds a user-defined threshold, it is called a **complete-linkage algorithm**.
- The distance between two clusters is determined by the most distant nodes in the two clusters.



Maximum distance: $dist_{max}(C_i, C_j) = \max_{\boldsymbol{p} \in C_i, \boldsymbol{p}' \in C_j} \{|\boldsymbol{p} - \boldsymbol{p}'|\}$

Consider following 6 1-D data point
18,22,25,42,27,43
Solve by using single linkage and complete linkage hierarchical procedure and draw dendograms

The distance between five pair of cases given below:

	A	B	C	D	E
A	0				
B	8	0			
C	2	6	0		
D	5	4	8	0	
E	10	9	3	7	0

Cluster the five cases using below procedure and draw the Dendograms structure.

a) Single linkage hierarchical procedure.
b) Complete linkage hierarchical procedure.

Use single, complete, and average link agglomerative clustering to group the data described by the following distance metric. Produce the dendrograms.

	A	B	C	D
A	0	1	4	5
B		0	2	6
C			0	3
D				0

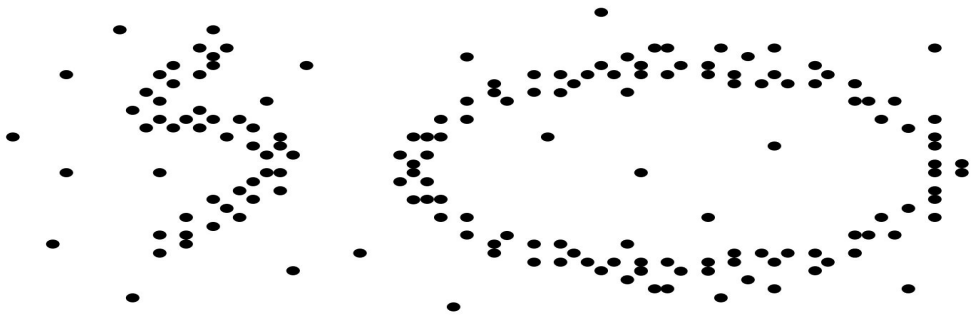
10(CO4)

Find the cluster using single, complete and average linkage . use Euclidian distance to find distance and draw dendrogram

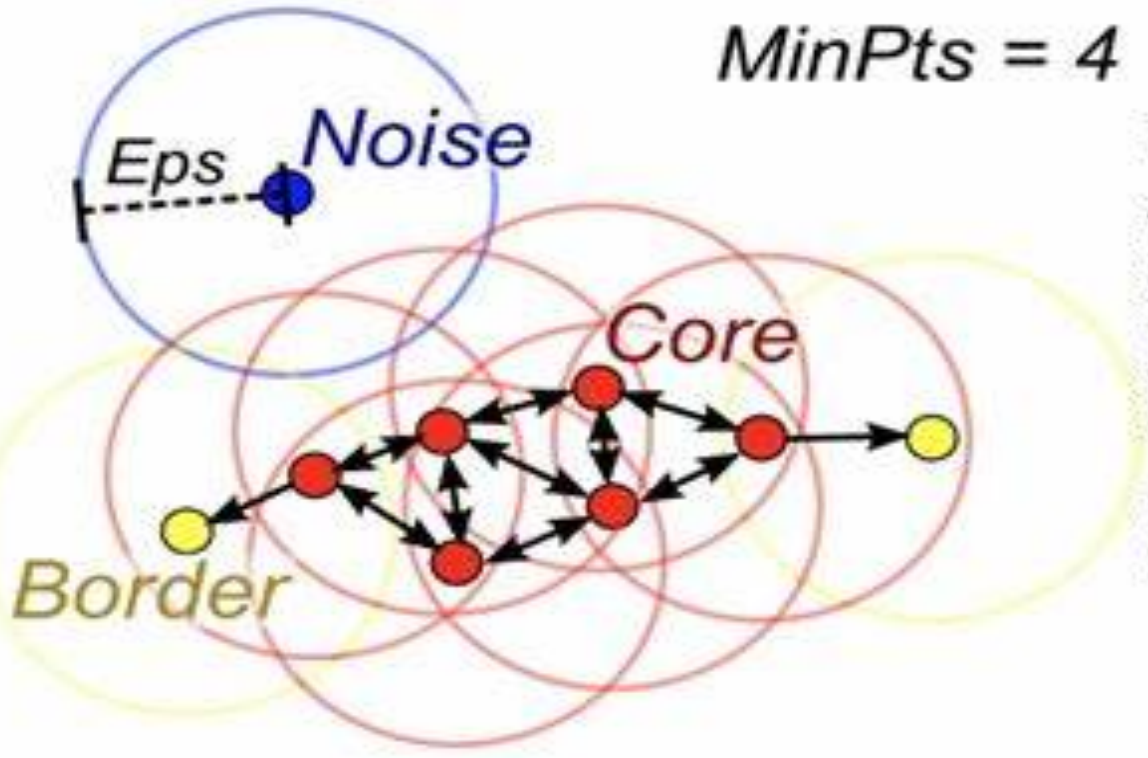
	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)

- Partitioning and hierarchical methods are designed to find spherical-shaped clusters.
- They have difficulty finding clusters of arbitrary shape such as **the "S" shape and oval clusters**
- To find clusters of arbitrary shape, alternatively, we can model clusters as dense regions in the data space, separated by sparse regions
- The density of an object **o** can be measured by the number of objects close to o
- A user-specified parameter $\epsilon > 0$ is used to specify the radius of a neighborhood we consider for every object.
- The -neighborhood of an object o is the space within a radius centered at o.



- To determine whether a neighborhood is dense or not, DBSCAN uses another user-specified parameter, **MinPts**, which specifies the density threshold of dense regions.
- An object is a core object if the -neighborhood of the object contains at least MinPts objects.
- Core objects are the pillars of dense regions



Red: Core Points

Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria

Blue: Noise point. Not assigned to a cluster

Density-based methods	– Can find arbitrarily shaped clusters – Clusters are dense regions of objects in space that are separated by low-density regions – Cluster density: Each point must have a minimum number of points within its "neighborhood" – May filter out outliers

*

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- *MinPts*: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the ε-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the ε-neighborhood of p;
(8)          for each point p' in N
(9)              if p' is unvisited
(10)                 mark p' as visited;
(11)                 if the ε-neighborhood of p' has at least MinPts points,
                     add those points to N;
(12)             if p' is not yet a member of any cluster, add p' to C;
(13)         end for
(14)         output C;
(15)     else mark p as noise;
(16) until no object is unvisited;
```

Form clusters with $\epsilon = 3.5$ and MinPts=3

S1	5	7
S2	8	4
S3	3	3
S4	4	4
S5	3	7
S6	6	7
S7	6	1
S8	5	5

Run DBScan algorithm for following 10 examples
X1(2,10);
X2(2, 5);
X3(8, 4);
X4(9,4);
Y1(5, 8);
Y2(7,5);
Y3(6,4);
Zl(l,2);
Z2(4,9);
Z3(6,10)
with minimum points=2 and epsilon=2

Extra Topic

Types of Fact(measure)

- The numeric measures in a fact table fall into three categories.
- Additive Fact
- Semi Additive Fact
- Non-Additive fact

Additive Fact

- The most flexible and useful facts are fully *additive*
- additive measures can be summed across any of the dimensions associated with the fact table.

Semi Additive Fact

- *Semi-additive* measures can be summed across some dimensions, but not all;
- balance amounts are common semi-additive facts because they are additive across all dimensions except time.
-

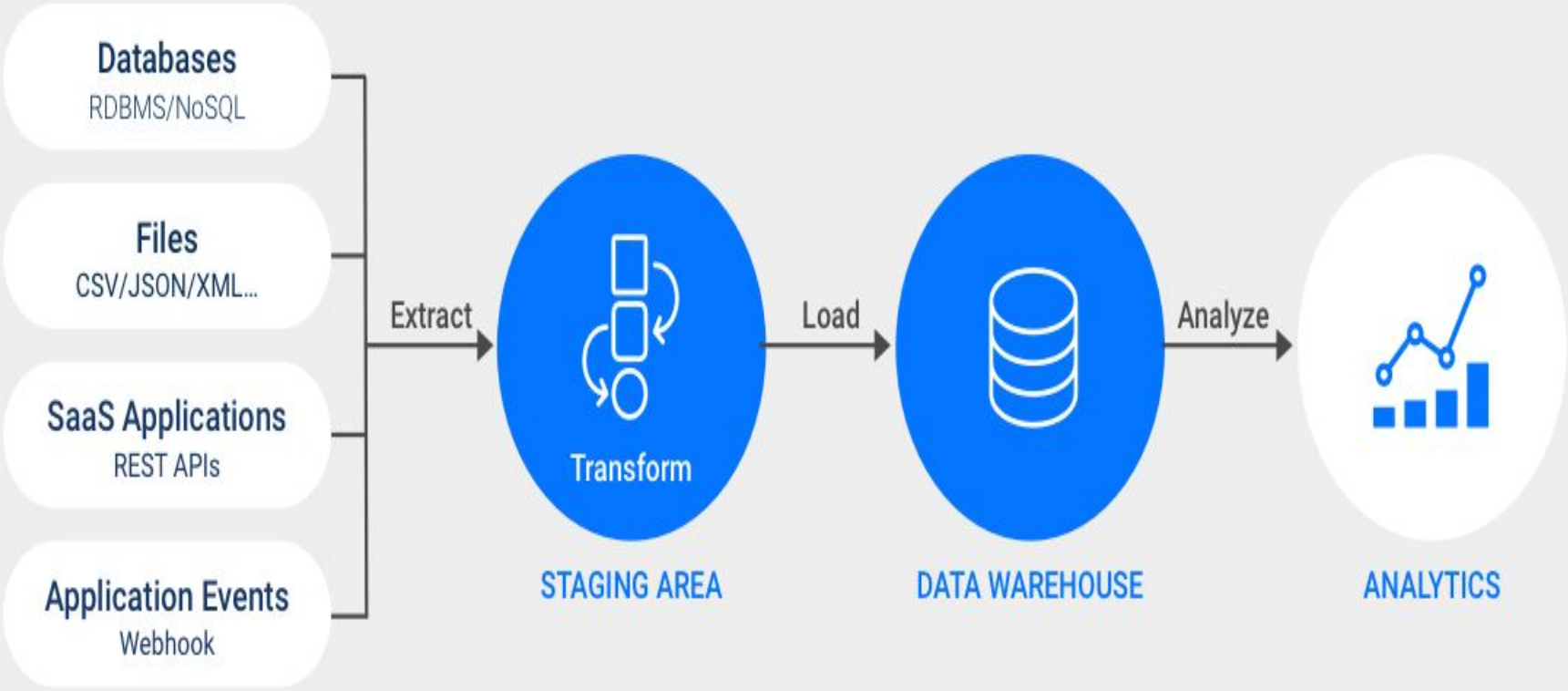
Non-Additive Fact:-

- some measures are completely *non-additive*, such as ratios.

Date	Store	Product	Transaction #	Quantity Sold	Unit price	Sales Amount
3/5/2017	Miami	A	1001	10	100	1000
3/5/2017	Miami	B	1003	20	120	2400
3/5/2017	Miami	C	1005	30	90	2700
3/6/2017	Miami	A	1007	20	100	2000
3/6/2017	Miami	B	1009	21	120	2520
3/5/2017	NYC	C	1002	9	90	810
3/5/2017	NYC	A	1004	15	100	1500
3/5/2017	NYC	B	1006	10	120	1200

Date	Store	Product	Transaction #	Quantity	Unit price	Sales Amount	Quantity in Hand	Sales Tax
3/5/2017	Miami	A	1001	10	100	1000	50	5.00%
3/5/2017	Miami	B	1003	20	120	2400	60	5.00%
3/5/2017	Miami	C	1005	30	90	2700	40	5.00%
3/6/2017	Miami	A	1007	20	100	2000	30	5.00%
3/6/2017	Miami	B	1009	21	120	2520	39	5.00%
3/5/2017	NYC	C	1002	9	90	810	49	5.00%
3/5/2017	NYC	A	1004	15	100	1500	55	5.00%
3/5/2017	NYC	B	1006	10	120	1200	45	5.00%
3/6/2017	NYC	C	1008	20	90	1800	40	5.00%

ETL PROCESS


Databases
RDBMS/NoSQL
Files
CSV/JSON/XML...
SaaS Applications
REST APIs
Application Events
Webhook
Extract
Transform
STAGING AREA
Load
DATA WAREHOUSE
Analyze
ANALYTICS

Basic Statistical Descriptions of Data

- For data preprocessing to be successful, it is essential to have an overall picture of your data
- Basic statistical descriptions can be used to identify properties of the data and highlight which data values should be treated as noise or outliers

Three areas of basic statistical descriptions

1. Measures of central tendency

- Which measure the location of the middle or center of a data distribution.

Example:- mean, median, mode, and midrange.

2. Dispersion of the data.

- That is, how are the data spread out?

Example:-The most common data dispersion measures are the range, quartiles, and interquartile range; the five-number summary and boxplots; and the variance and standard deviation

3. Graphic displays of basic statistical descriptions

Example:-Displays of data summaries and distributions include quantile plots, quantile–quantile plots, histograms, and scatter plots.

Example 2.6 **Mean.** Suppose we have the following values for *salary* (in thousands of dollars), shown in increasing order: 30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110. Using Eq. (2.1), we have

$$\bar{x} = \frac{30 + 36 + 47 + 50 + 52 + 52 + 56 + 60 + 63 + 70 + 70 + 110}{12}$$

$$= \frac{696}{12} = 58.$$

Thus, the mean salary is \$58,000. ■

Example 2.7 **Median.** Let's find the median of the data from Example 2.6. The data are already sorted in increasing order. There is an even number of observations (i.e., 12); therefore, the median is not unique. It can be any value within the two middlemost values of 52 and 56 (that is, within the sixth and seventh values in the list). By convention, we assign the average of the two middlemost values as the median; that is, $\frac{52+56}{2} = \frac{108}{2} = 54$. Thus, the median is \$54,000.

Suppose that we had only the first 11 values in the list. Given an odd number of values, the median is the middlemost value. This is the sixth value in this list, which has a value of \$52,000. ■

Mode:-

- The mode is another measure of central tendency.
- The mode for a set of data is the value that occurs most frequently in the set.
- Data sets with one, two, or three modes are respectively called unimodal, bimodal, and trimodal.
- In general, a data set with two or more modes is multimodal. At the other extreme, if each data value occurs only once, then there is no mode.

Example 2.8 **Mode.** The data from Example 2.6 are bimodal. The two modes are \$52,000 and \$70,000. ■

Midrange :-

- It can also be used to assess the central tendency of a numeric data set.
- It is the average of the largest and smallest values in the set.
- This measure is easy to compute using the SQL aggregate functions, max() and min().

Example 2.9 Midrange. The midrange of the data of Example 2.6 is $\frac{30{,}000+110{,}000}{2} = \$70{,}000.$

Quartile

- Quartiles are a type of **quantile**.
- The values which divide dataset into four equal parts are called quartile
- Each part contain equal no of observation
- There are three quartiles Q1,Q2,Q3
- The quartiles give an indication of a distribution's center, spread, and shape.
- The first quartile, denoted by Q1, is the 25th percentile. This is the number halfway between the lowest number and the middle number.
- The third quartile, denoted by Q3, is the 75th percentile. This is the number halfway between the middle number and the highest number.
- The second quartile is the 50th percentile. As the median, it gives the center of the data distribution.

- The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. This distance is called the interquartile range (IQR) and is defined as

IQR = Q3 – Q1

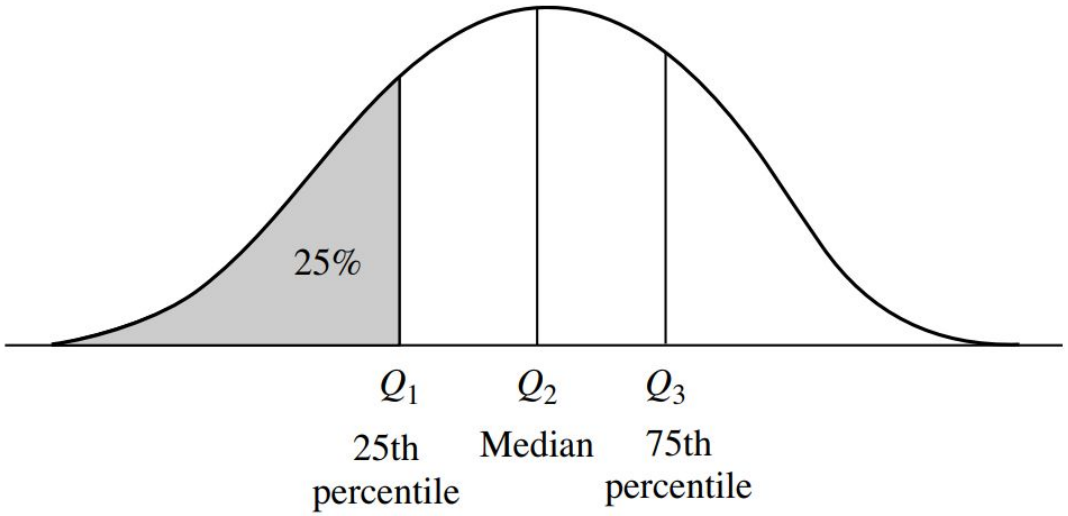


Figure 2.2 A plot of the data distribution for some attribute X. The quantiles plotted are quartiles. The three quartiles divide the distribution into four equal-size consecutive subsets. The second quartile corresponds to the median.

Identifying outliers

- The interquartile range (IQR) can be used to identify **outliers**.
- Outliers are observations that are extremely high or low.
- One definition of an outlier is any observation that is more than 1.5 IQR away from the first or third quartile.

How to find quartiles

Find the second quartile:

1. Count the number of observations in the dataset (*n*).
2. Sort the observations from smallest to largest.
3. Find the first quartile:
    1. Calculate *n* * (1 / 4).
    2. If *n* * (1 / 4) is an integer, then the first quartile is the mean of the numbers at positions *n* * (1 / 4) and *n* * (1 / 4) + 1.
    3. If *n* * (1 / 4) is **not** an integer, then round it up. The number at this position is the first quartile.

Find the second quartile:

4. Calculate $n * (2 / 4)$.
5. If $n * (2 / 4)$ is an integer, the second quartile is the mean of the numbers at positions $n * (2 / 4)$ and $n * (2 / 4) + 1$.
6. If $n * (2 / 4)$ is **not** an integer, then round it up. The number at this position is the second quartile.

Find the third quartile:

7. Calculate $n * (3 / 4)$.
8. If $n * (3 / 4)$ is an integer, then the third quartile is the mean of the numbers at positions $n * (3 / 4)$ and $n * (3 / 4) + 1$.
9. If $n * (3 / 4)$ is **not** an integer, then round it up. The number at this position is the third quartile.

Five number Summary

- Q1, the median, and Q3 together contain no information about the endpoints (e.g., tails) of the data,
- A fuller summary of the shape of a distribution can be obtained by providing the lowest and highest data values as well.
- This is known as the five-number summary.
- The five-number summary of a distribution consists of the median (Q2), the quartiles Q1 and Q3, and the smallest and largest individual observations, written in the order of Minimum, Q1, Median, Q3, Maximum.
- Boxplots are a popular way of visualizing a distribution.

A boxplot incorporates the five-number summary as follows:

- Typically, the ends of the box are at the quartiles so that the box length is the interquartile range.
- The median is marked by a line within the box.
- Two lines (called whiskers) outside the box extend to the smallest (Minimum) and largest (Maximum) observations

Slowly changing and rapidly changing dimension

Data warehouses store historical data from an online transaction processing (OLTP) system. As new data is extracted into the data warehouse from the source OLTP system, some records may change. When the attributes of a given dimension table change, this is called a *slowly changing dimension.*

There are three types of slowly changing dimensions:

- *Type 1 Slowly Changing Dimension:* This method overwrites the existing value with the new value and does not retain history.
- *Type 2 Slowly Changing Dimension:* This method adds a new row for the new value and maintains the existing row for historical and reporting purposes.
- *Type 3 Slowly Changing Dimension:* This method creates a new *current value* column in the existing record but also retains the original column.

Rapidly Changing Attribute

https://www.gability.com/en/courses/data-modeling/02-dimension-types/08-fast-changing-dimension/