Data Clustering

Dr. Hrishikesh Bhaumik
Associate Professor, Department of Information Technology,
RCC Institute of Information Technology, Kolkata

Classification

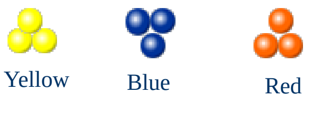
$$\begin{tikzcd}
 & \text{Classification} \arrow[dl] \arrow[dr] & \\
\text{Supervised Classification (Discriminant Analysis)} & & \text{Unsupervised Classification (Clustering)}
\end{tikzcd}$$

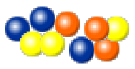
What is Supervised Classification?

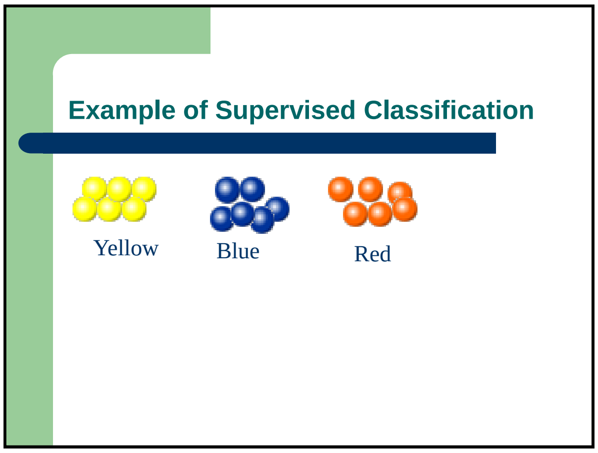
In Supervised classification:

- We are provided with a collection of labeled (pre-classified) patterns.
- The given labeled (training) patterns are used to learn the descriptions of classes which in turn are used to label a new pattern.
- The problem is to classify or label a newly encountered, yet unlabeled pattern.

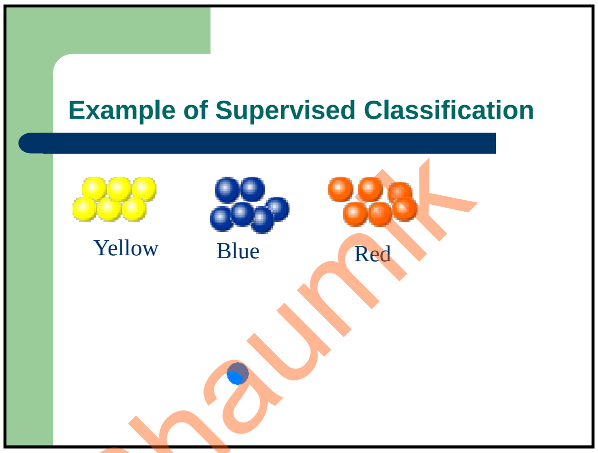
Example of Supervised Classification



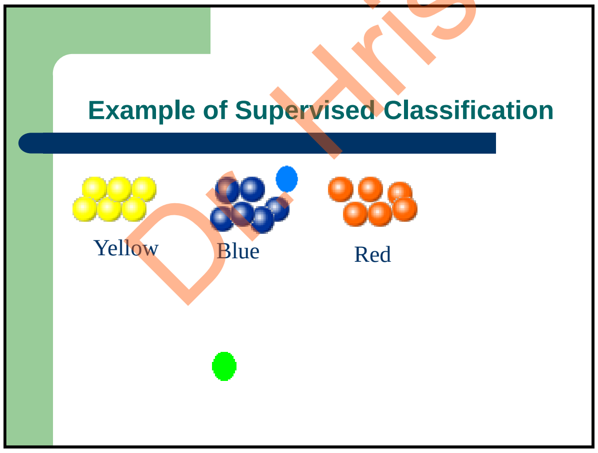




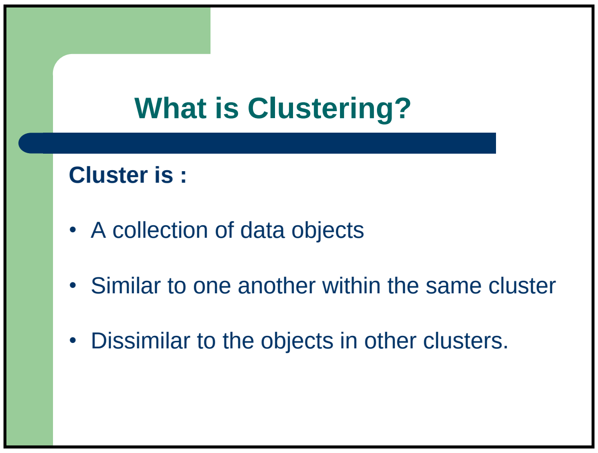
Example of Supervised Classification



Example of Supervised Classification



Example of Supervised Classification



What is Clustering?

Cluster is :

- A collection of data objects
- Similar to one another within the same cluster
- Dissimilar to the objects in other clusters.

What is Clustering?

Clustering of data is a method by which large sets of data is grouped into clusters of smaller sets of similar data.It is a useful technique for the discovery of some knowledge from a dataset.

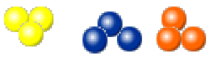
Clustering

In clustering, the problem is to group a given collection of unlabeled patterns into meaningful groups. Labels are associated with clusters also, but these category labels are data driven i.e. they are obtained from the data.

Example of Clustering

There are a total of 10 balls which are of three different colours. We are interested in making clusters of the balls having same colour.

The balls of same colour are clustered into a group as shown below :



Components of a Clustering Task

- Pattern Representation(including feature extraction and/or selection)
- Definition of a pattern proximity measure appropriate to the data domain
- Clustering or grouping
- Data Abstraction(if needed)
- Assessment of output(if needed)

Stages in Clustering

Pattern Representation refers to:

- Number of classes
- Number of available patterns
- Number, type, and the scale of the features available to the clustering algorithm.

Stages in Clustering

- Feature selection is the process of identifying the most effective subset of the original features to use in clustering.
- Feature extraction is the use of one or more transformations of the input features to produce new salient features.

Stages in Clustering

Pattern proximity is usually measured by a distance function defined on pairs of patterns.

Distance measures may be:

- Euclidean distance
- Manhattan distance
- Minkowski distance
- Mahalnobis distance
- Correlation measures etc.

Stages in Clustering

Grouping or clustering can be performed in a number of ways. The output clusters can be

- **hard** (a partition of the data into groups)
- **fuzzy** (where each pattern has a variable degree of membership in each of the output clusters)

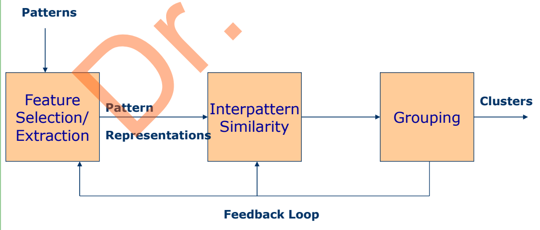
Stages in Clustering

Data abstraction is the process of extracting a simple and compact representation of a data set usually in terms of cluster prototypes or a representative patterns such as centroid.

Stages in Clustering

Cluster validity analysis is the assessment of a clustering procedure's output. Often this analysis uses a specific criterion of optimality. This criteria are usually arrived at subjectively. Validity assessments are objective and are performed to determine whether the output is meaningful.

Stages in Clustering



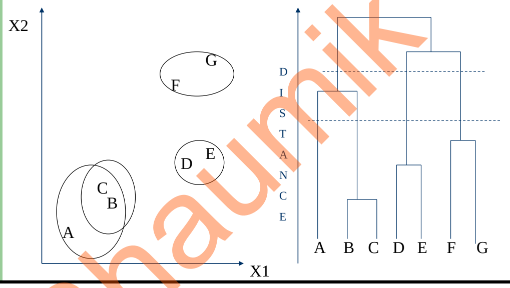
Terms related to Clustering

- **Measurement Space** (M): It consists of all the measurements to be taken or existing measurements of an object which is to be recognized.
- **Feature Space** (F): It is a subset of M and consists of the measurements (called features) which are required to correctly classify an object.
- **Feature Vector** : It is also called a pattern vector. It is a multi-dimensional vector whose dimensions equal the number of features corresponding to an object.
- **Decision Space** (D): When the feature space is divided into some regions where each region represents a particular class of the object, then such a space is called a decision space.

Clustering Techniques

Hierarchical clustering algorithms produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity.

Hierarchical Clustering



Hierarchical Clustering Algorithms

A hierarchical algorithm yields a *dendrogram* representing the nested grouping of patterns and similarity levels at which groupings change.

Hierarchical Clustering Algorithm

Algorithm

(1)Compute the proximity matrix containing the distance between each pair of patterns. Treat each pattern as a cluster.

(2)Find the most similar pair of clusters using the proximity matrix. Merge these two clusters into one cluster. Update the proximity matrix to reflect this merge operation.

(3)If all patterns are in one cluster, stop. Otherwise, go to step 2.

Disadvantages of Hierarchical Clustering

- Low quality clusters may result if the merge and split decisions are not made properly
- In case of large number of data objects this method is not very practicable since it scales at least quadratically i.e. O(N^2)

Partitional Clustering

Partitional clustering algorithms begins with all patterns in a **single cluster** and performs splitting until a **stopping criterion** is met i.e. it identifies the partition that **optimizes** (usually locally) a **clustering criterion**.

Partitional Clustering

1. Obtains a **single partition** of the data instead of a clustering structure, such as the dendrogram produced by a hierarchical technique.
2. Advantageous for applications involving **large data sets** for which the construction of a **dendrogram** is **computationally very expensive**.

Problem: choice of the number of desired output clusters.

Partitional Clustering

The **partitional techniques** usually produce clusters by **optimizing a criterion function** defined either locally (on a subset of the patterns) or globally (defined over all of the patterns).

Combinatorial search of the set of possible labelings for an optimum value of a criterion is clearly **computationally prohibitive**.

In practice, therefore, the algorithm is typically run **multiple times with different starting states**, and the best configuration obtained from all of the runs is used as the output clustering.

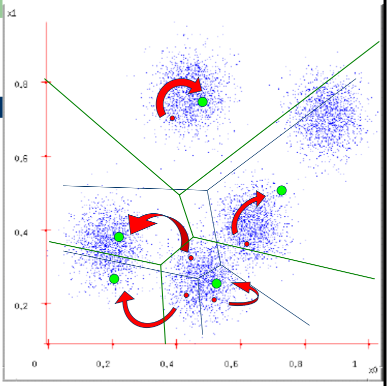
THE ALGORITHM

1. Ask User how many clusters they'd like. (e.g. k=5).
2. Random select k cluster centers locations

$$M_k = \frac{1}{N_k} \cdot \sum_{j=1}^{N_k} X_{jk}$$

d ints)

4. Each cluster fine the centroid of the points it owns.
5. …and jumps to there
6. …repeat step 3 to 5 until terminated



k-Means Clustering Algorithm

1. **Choose *k* cluster centers** to coincide with *k* randomly-chosen patterns or *k* randomly defined points inside the hypervolume containing the pattern set.
2. **Assign each pattern** to the closest cluster center.
3. **Recompute the cluster centers** using the current cluster memberships.
4. If a **convergence criterion** is not met, go to step 2.

Typical convergence criteria are: no (or minimal) reassignment of patterns to new cluster centers or minimal decrease in squared error.

Squared Error

$$e^2(X,L) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2$$

X is the pattern set (containing K clusters)

$x_i^{(j)}$ is the i^{th} pattern belonging to the j^{th} cluster

c_j is the centriod of the j^{th} cluster

Squared Error Clustering Method

(1) Select an initial partition of the patterns with a fixed number of clusters and cluster centers.

(2) Assign each pattern to its closest cluster center and compute the new cluster centers as the centroids of the clusters. Repeat this step until convergence is achieved, i.e., until the cluster membership is stable.

(3) Merge and split clusters based on some heuristic information, optionally repeating step 2.

K-Means Algorithm

The *k*-means algorithm is popular because it is easy to implement, and its time complexity is $O(n)$, where n is the number of patterns.

A major problem with this algorithm is that it is sensitive to the selection of the initial partition and may converge to a local minimum of the criterion function value if the initial partition is not properly chosen.

Divisive Clustering

A divisive method of clustering begins with all **patterns in a single cluster** and performs splitting until a **stopping criterion** is met.

Graph-Theoretic Clustering

The best-known graph-theoretic divisive clustering algorithm is based on construction of the ***minimal spanning tree*** (MST) of the data and then deleting the MST edges with the largest lengths to generate clusters.

Spanning Trees

Given (connected) graph G(V,E),

a spanning tree T(V',E'):

- Is a subgraph of G; that is, $V' \subseteq V$, $E' \subseteq E$.
- Spans the graph (V' = V)
- Forms a tree (no cycle);
- So, E' has |V| -1 edges

Minimum Spanning Trees

- Edges are weighted: find minimum cost spanning tree
- Applications
  - › Find cheapest way to wire your house
  - › Find minimum cost to send a message on the Internet

Strategy

- Strategy for construction:
  - › Add an edge of minimum cost that does not create a cycle (greedy algorithm)
  - › Repeat |V| -1 times
  - › Correct since if we could replace an edge with one of lower cost, the algorithm would have picked it up

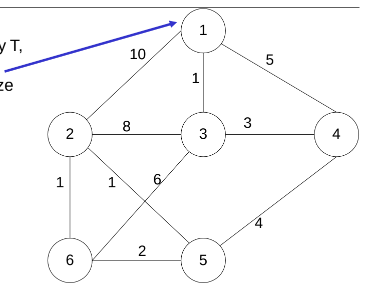
Two Algorithms

- Prim: (build tree incrementally)
  - › Pick lower cost edge connected to known (incomplete) spanning tree that does not create a cycle and expand to include it in the tree
- Kruskal: (build forest that will finish as a tree)
  - › Pick lowest cost edge not yet in a tree that does not create a cycle. Then expand the set of included edges to include it. (It will be somewhere in the forest.)

Prim's algorithm

Starting from empty T, choose a vertex at random and initialize

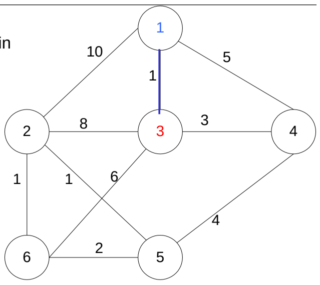
V = {1), E' ={ }



Prim’s algorithm

Choose the vertex u not in V such that edge weight from u to a vertex in V is minimal (greedy!)

V={1,3} E’= {(1,3) }



Prim’s algorithm

Repeat until all vertices have been chosen

Choose the vertex u not in V such that edge weight from v to a vertex in V is minimal (greedy!)

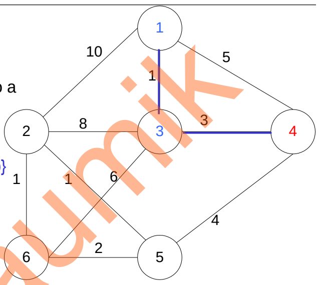
V= {1,3,4} E’= {(1,3),(3,4)}

V={1,3,4,5} E’={(1,3),(3,4),(4,5)}

….

V={1,3,4,5,2,6}

E’={(1,3),(3,4),(4,5),(5,2),(2,6)}



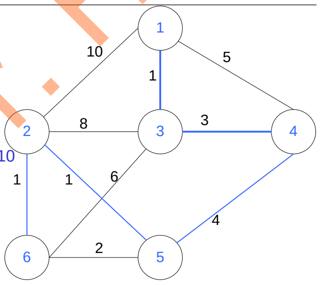
Prim’s algorithm

Repeat until all vertices have been chosen

V={1,3,4,5,2,6}

E’={(1,3),(3,4),(4,5),(5,2),(2,6)}

Final Cost: 1 + 3 + 4 + 1 + 1 = 10



Prim’s Algorithm Implementation

- Assume adjacency list representation

```
Initialize connection cost of each node to “inf” and “unmark” them
Choose one node, say v and set cost[v] = 0 and prev[v] =0
While they are unmarked nodes
    Select the unmarked node u with minimum cost; mark it
    For each unmarked node w adjacent to u
        if cost(u,w) < cost(w) then cost(w) := cost (u,w)
        prev[w] = u
```

Kruskal's Algorithm

- Select edges in order of increasing cost
- Accept an edge to expand tree or forest only if it does not cause a cycle
- Implementation using adjacency list, priority queues and disjoint sets

Detecting Cycles

- If the edge to be added (u,v) is such that vertices u and v belong to the same tree, then by adding (u,v) you would form a cycle
  - Therefore to check, Find(u) and Find(v). If they are the same discard (u,v)
  - If they are different Union(Find(u),Find(v))

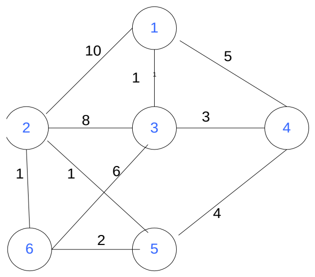
Properties of trees in K's algorithm

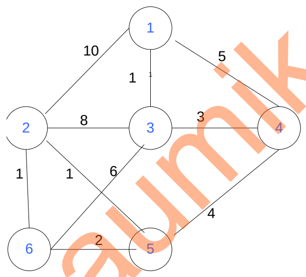
- Vertices in different trees are disjoint
  - True at initialization and Union won't modify the fact for remaining trees
- Trees form equivalent classes under the relation "is connected to"
  - u connected to u (reflexivity)
  - u connected to v implies v connected to u (symmetry)
  - u connected to v and v connected to w implies a path from u to w so u connected to w (transitivity)

K's Algorithm Data Structures

- Adjacency list for the graph
  - To perform the initialization of the data structures below
- Disjoint Set ADT's for the trees (recall Up tree implementation of Union-Find)
- Binary heap for edges

Example



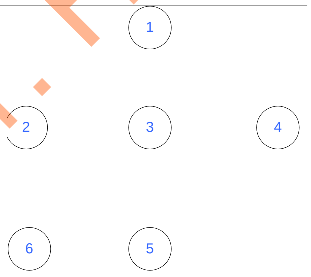


Initialization

Initially, Forest of 6 trees

F= {{1},{2},{3},{4},{5},{6}}

Edges in a heap (not shown)



Step 1

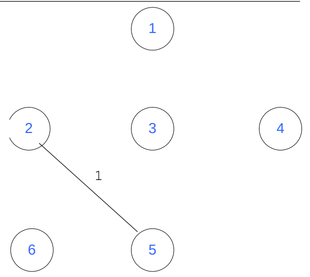
Select edge with lowest cost (2,5)

Find(2) = 2, Find (5) = 5

Union(2,5)

F= {{1},{2,5},{3},{4},{6}}

1 edge accepted



Step 2

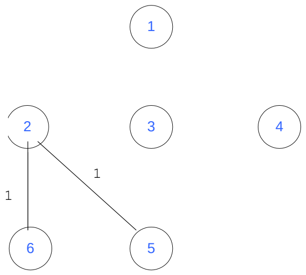
Select edge with lowest cost (2,6)

Find(2) = 2, Find (6) = 6

Union(2,6)

F= {{1},{2,5,6},{3},{4}}

2 edges accepted



Step 3

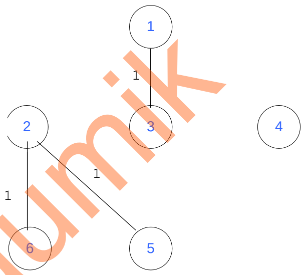
Select edge with lowest cost (1,3)

Find(1) = 1, Find (3) = 3

Union(1,3)

F= {{1,3},{2,5,6},{4}}

3 edges accepted



Step 4

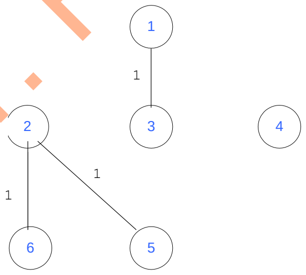
Select edge with lowest cost (5,6)

Find(5) = 2, Find (6) = 2

Do nothing

F= {{1,3},{2,5,6},{4}}

3 edges accepted



Step 5

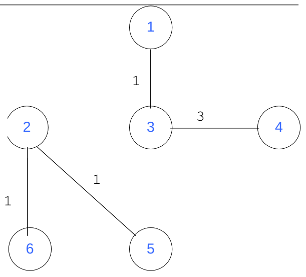
Select edge with lowest cost (3,4)

Find(3) = 1, Find (4) = 4

Union(1,4)

F= {{1,3,4},{2,5,6}}

4 edges accepted



Step 6

Select edge with lowest cost (4,5)

Find(4) = 1, Find (5) = 2

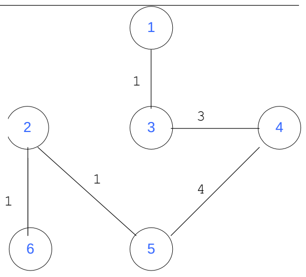
Union(1,2)

F= {{1,3,4,2,5,6}}

5 edges accepted : end

Total cost = 10

Although there is a unique spanning tree in this example, this is not generally the case

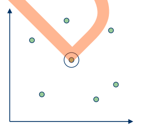


Cluster Representation

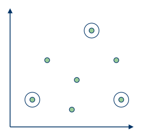
The notion of cluster representation was introduced by **Duran** and **Odell** [1974] which suggests several representation schemes.

Cluster Representation

1) Represent a cluster of points by their **centroid** or by a **set of distant points** in the cluster.



By the CENTROID



By Three Distant Points

Cluster Representation

(2) Represent clusters using **nodes in a classification tree**.

(3) Represent clusters by using **conjunctive logical expressions**.

Example:

The expression [*X*1 > 3] [*X*2 <2] stands for the logical statement

'***X*1 is greater than 3**' ***and*** '***X*2 is less than 2**'

Cluster Representation…

Centroid representation of a cluster is the most popular scheme. It works well when the clusters are **compact** or **isotropic**.

When the clusters are **elongated** or **non-isotropic**, this **scheme fails** to represent them properly. In such a case, the use of a **collection of boundary points** in a cluster **captures its shape** well. The number of points used to represent a cluster should increase as the complexity of its shape increases.

ANY ?

Thank You