# UNIT – IV LOGIC BASED AND ALGEBRAIC MODELS

- Distance Based Models: Neighbors and Exemplars, Nearest Neighbor Classification, Distance based clustering algorithms - K-means and K-medoids, Hierarchical clustering.
- Rule Based Models: Rule learning for subgroup discovery, Association rules mining – Apriori Algorithm, Confidence and Support parameters.
- Tree Based Models: Decision Trees, Minority Class, Impurity Measures – Gini Index and Entropy, Best Split.

# Minkowski distance

- *If* $\mathcal{X} = \mathbb{R}^d$ *, the Minkowski distance of order*
- *p > 0 is defined as*

$$\mathrm{Dis}_p(\mathbf{x}, \mathbf{y}) = \left( \sum_{j=1}^{d} |x_j - y_j|^p \right)^{1/p} = ||\mathbf{x} - \mathbf{y}||_p$$

- 1-*norm denotes Manhattan distance, also called cityblock distance:*

$$\mathrm{Dis}_1(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{d} |x_j - y_j|$$

- 2-*norm refers to Euclidean distance*

$$\mathrm{Dis}_2(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2} = \sqrt{(\mathbf{x}-\mathbf{y})^{\mathrm{T}}(\mathbf{x}-\mathbf{y})}$$

- *Given an instance space X, a distance metric is*
- *a function Dis :$\mathcal{X}$ X $\mathcal{X}$ → $\mathbb{R}$ such that for any x, y, z ∈ : $\mathcal{X}$*
  1. *distances between a point and itself are zero: Dis(x,x) = 0;*
  2. *all other distances are larger than zero: if x = y then Dis(x, y) > 0;*
  3. *distances are symmetric: Dis(y,x) = Dis(x, y);*
  4. *detours can not shorten the distance: Dis(x, z) ≤ Dis(x, y)+Dis(y, z).*
- *If the second condition is weakened to a non-strict inequality – i.e., Dis(x, y) may be zero even if x = y – the function Dis is called a pseudo-metric.*

- Exemplars :
  - centroids that find a centre of mass according to a chosen distance metric,

    or

  - medoids that find the most centrally located data point

1, 2, 3, 4, 5

Arithmetic Mean $\rightarrow$ $\mu = \frac{1}{|X|} \cdot \sum_{x \in X} x$

Geometric Mean $\rightarrow$

- Medoids
  - Always the members of data set
  - Can be selected by calculating distance metric
- Centroid
  - Mean
  - Imaginary data points
-

# The arithmetic mean minimises squared Euclidean distance

- *The arithmetic mean* ***μ of a set of data points D in a Euclidean space is the unique point*** *that minimises the sum of squared Euclidean distances to those data points.*

# Nearest neighbour Classification

- Most commonly used distance based classifier
- Uses each training instance as exemplar
- Can be used in two ways
  - Decision boundary
    - A perpendicular bisector of the line connecting the two exemplars
  - Decision rule

# K-Nearest Neighbour (kNN)

- Takes a vote between the k $\geq$ 1 nearest exemplars of the instance to be classified and predicts majority class.
- Well suited for classification task
- Supervised learning

# K-Nearest Neighbour (kNN)

- Takes a vote between the $k \geq 1$ nearest exemplars of the instance to be classified and predicts majority class.
- K value selection is most important
- Generally it is odd and set to the square root of no of training examples
- No standard procedure exists. K is selected after testing several k values on variety of test data sets.

# Working

- Training data set(labeled)
- Test data set(unlabeled)
- For each record in test data set kNN identifies k records in training data that are nearest in similarity
- Unlabeled test instance is assigned the class of majority of the k nearest neighbours

| | X | Y | Result |
|---|---|---|---|
| $D_1$ | 7 | 7 | Pass |
| $D_2$ | 7 | 4 | Pass |
| $D_3$ | 3 | 4 | Fail |
| $D_4$ | 1 | 4 | Fail |
| $D_n$ | 3 | 7 | ? — Fail |

$D_n(3, 7)$

$d_{1n} = \sqrt{(7-3)^2 + (7-7)^2} = 4$

$d_{2n} = \sqrt{(7-3)^2 + (4-7)^2} = 5$

$d_{3n} = \sqrt{(3-3)^2 + (4-7)^2} = 3$

$d_{4n} = \sqrt{(1-3)^2 + (4-7)^2} = \sqrt{13} \simeq 3.60$

$k = 3$

3, 3.60, 4, 5

F F P P

(8)

| No. of Good words | No. of bad words | |
|---|---|---|
| 3 | 104 | ~~spat~~ spam |
| 2 | 100 | spam |
| 1 | 81 | spam |
| 101 | 10 | Not spam |
| 99 | 5 | Not spam |
| 98 | 2 | Not spam |
| 18 | 98 | ? |

$k = 3 \rightarrow$

$k = 5 \rightarrow$

- Let k b e any integer generally odd and must not be multiple of no. of classes.
- Find n Euclidean Distances between Xnew and X1,X2,X3,……,Xn
- Arrange all distances in Ascending order
- Select K shortest distances
- Find K corresponding points
- Find their labels
- Find Maximum from the labels and assign to new data point.

# R implementation

- knn( train, test, class, k)  function
  - train  is a data frame containing training data
  - test  is a data frame containing testing data
  - Class vector with class for each row in training data
  - K an integer indicating nearest neighbours

- Returns the vector of predicted classes

# Distance based clustering

- Unsupervised Learning
- exemplar-based
  - Predictive
- not exemplar-based
  - Descriptive

① centroid
② medoid

- Predictive distance-based clustering methods use the same ingredients as distance based classifiers: a distance metric, a way to construct exemplars and a distance-based decision rule.

# KMeans

Lloyd's algo

- The *K-means problem is to find a* partition that minimises the total within-cluster scatter.
- The algorithm iterates between partitioning the data using the nearest-centroid decision rule, and recalculating centroids from a partition

# Steps

- Select initial centroids at random
- Calculate the distance of each instance from each centroid Eu
- Assign every instance to cluster represented by it's nearest centroid
- Recalculate the centroid for each k clusters
- Go to step 2 and keep on repeating until no change in centroids is observed

- Consider the following instances given as input to k-means clustering algorithm for k=3.Find the members of these 3 clusters.
- X = {(2,10),(2,5),(8,4),(5,8),(7,5),(6,4),(1,2),(4,9)}

1 2 3 4 5 6 7 8

$C_1$ $C_2$ $C_3$

X = (10, 2) (5, 2) (4, 8) (8, 4) (5, 7) (4, 6) (2, 1) (4, 9)

| | $C_1(2,10)$ | $C_2(5,8)$ | $C_3(1,2)$ |
|---|---|---|---|
| 1) (2,10) | 0 | $\sqrt{(2-5)^2+(10-8)^2} = \sqrt{13}$ | $\sqrt{(2-1)^2+(10-2)^2} = \sqrt{65}$ |
| 2) (2,5) | $\sqrt{25}$ | $\sqrt{(2-5)^2+(5-8)^2} = \sqrt{18}$ | $\sqrt{(2-1)^2+(5-2)^2} = \sqrt{10}$ |
| 3) (8,4) | $\sqrt{72}$ | $\sqrt{25}$ | $\sqrt{53}$ |
| 4) (5,8) | $\sqrt{13}$ | 0 | $\sqrt{32}$ |
| 5) (7,5) | $\sqrt{50}$ | $\sqrt{13}$ | $\sqrt{45}$ |
| 6) (6,4) | $\sqrt{52}$ | $\sqrt{17}$ | $\sqrt{29}$ |
| 7) (1,2) | $\sqrt{65}$ | $\sqrt{52}$ | 0 |
| 8) (4,9) | $\sqrt{5}$ | $\sqrt{2}$ | $\sqrt{58}$ |

centroids

$C_1$ (2,10) $\rightarrow$ (2,10)

$C_2$ (8,4), (5,8), (7,5) (6,4), (4,9)

$\frac{8+5+7+6+4}{5} = 6$

$4+8+5+4+9/5 = 6$

(6,6)

$C_3$ (2,5), (1,2)

$3/2,\ 7/2 = (1.5, 3.5)$

| | $C_1\ (2,10)$ | $C_2\ (6,6)$ | $C_3\ (1.5,3.5)$ |
|---|---|---|---|
| 1) 2,10 | 0 | $\sqrt{32}$ | $\sqrt{42.5}$ |
| 2 2,5 | $\sqrt{25}$ | $\sqrt{17}$ | $\sqrt{2.25}$ |
| 3 8,4 | $\sqrt{72}$ | $\sqrt{8}$ | $\sqrt{36.25}$ |
| 4 5,8 | $\sqrt{13}$ | $\sqrt{5}$ | $\sqrt{29.25}$ |
| 5 7,5 | $\sqrt{50}$ | $\sqrt{2}$ | $\sqrt{27.25}$ |
| 6 6,4 | $\sqrt{52}$ | $\sqrt{4}$ | $\sqrt{4.25}$ |
| 7 1,2 | $\sqrt{65}$ | $\sqrt{41}$ | $\sqrt{3.25}$ |
| 8 4,9 | $\sqrt{5}$ | $\sqrt{13}$ | $\sqrt{34.25}$ |

$C_1$ (2,10), (4,9) (3, 9.5)

$C_2$ (8,4) (5,8) (7,5), (6,4) (6.5, 5.25)

$C_3$ (2,5) (1,2) (1.5, 3.5)

3, 9.5 6.5, 5.25 (1.5, 3.5)

$C_1$ (2,10) (5,8), (4,9) $\rightarrow$ (3.66, 9)

$C_2$ (8,4), (7,5), (6,4) $\rightarrow$ (7, 4.33)

$C_3$ (2,5) (1,2) $\rightarrow$ (1.5)(3.5)

**Algorithm 8.1:** KMeans($D, K$) – $K$-means clustering using Euclidean distance $\mathrm{Dis}_2$.

**Input** : data $D \subseteq \mathbb{R}^d$; number of clusters $K \in \mathbb{N}$.

**Output** : $K$ cluster means $\mu_1, \ldots, \mu_K \in \mathbb{R}^d$.

randomly initialise $K$ vectors $\mu_1, \ldots, \mu_K \in \mathbb{R}^d$;

**repeat**

  assign each $\mathbf{x} \in D$ to $\arg\min_j \mathrm{Dis}_2(\mathbf{x}, \mu_j)$;

  **for** $j = 1$ **to** $K$ **do**

    $D_j \leftarrow \{\mathbf{x} \in D | \mathbf{x}$ assigned to cluster $j\}$;

    $\mu_j = \frac{1}{|D_j|} \sum_{\mathbf{x} \in D_j} \mathbf{x}$;

  **end**

**until** no change in $\mu_1, \ldots, \mu_K$;

**return** $\mu_1, \ldots, \mu_K$;

**Algorithm 8.2:** KMedoids($D, K$, Dis) – $K$-medoids clustering using arbitrary distance metric Dis.

**Input** : data $D \subseteq \mathscr{X}$; number of clusters $K \in \mathbb{N}$;
distance metric Dis : $\mathscr{X} \times \mathscr{X} \to \mathbb{R}$.

**Output** : $K$ medoids $\mu_1, \ldots, \mu_K \in D$, representing a predictive clustering of $\mathscr{X}$.

```
randomly pick K data points μ_1, ..., μ_K ∈ D;
repeat
    assign each x ∈ D to argmin_j Dis(x, μ_j);
    for j = 1 to k do
        D_j ← {x ∈ D | x assigned to cluster j};
        μ_j = argmin_{x ∈ D_j} Σ_{x' ∈ D_j} Dis(x, x');
    end
until no change in μ_1, ..., μ_K;
return μ_1, ..., μ_K;
```

$\mu_1$

| $x$ | $d$ |
|---|---|
| $x_1$ | $d_1$ |
| $x_2$ | $d_2$ |
| $x_3$ | |

**Algorithm 8.3:** PAM($D, K$, Dis) – Partitioning around medoids clustering using arbitrary distance metric Dis.

**Input** : data $D \subseteq \mathscr{X}$; number of clusters $K \in \mathbb{N}$;
distance metric Dis : $\mathscr{X} \times \mathscr{X} \to \mathbb{R}$.

**Output** : $K$ medoids $\mu_1, \ldots, \mu_K \in D$, representing a predictive clustering of $\mathscr{X}$.

```
randomly pick K data points μ_1, ..., μ_K ∈ D;
repeat
    assign each x ∈ D to argmin_j Dis(x, μ_j);
    for j = 1 to k do
        D_j ← {x ∈ D | x assigned to cluster j};
    end
    Q ← Σ_j Σ_{x∈D_j} Dis(x, μ_j);
    for each medoid m and each non-medoid o do
        calculate the improvement in Q resulting from swapping m with o;
    end
    select the pair with maximum improvement and swap;
until no further improvement possible;
return μ_1, ..., μ_K;
```

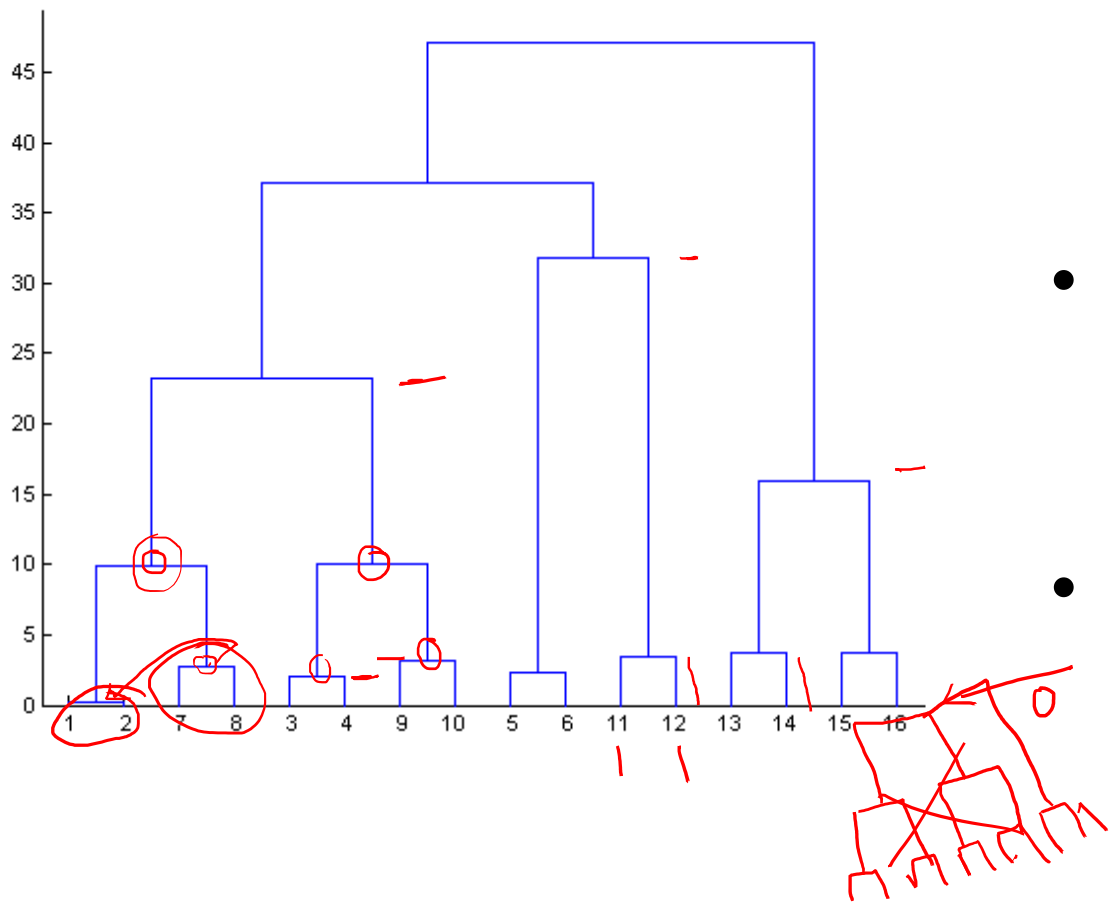
# Hierarchical clustering

- represent clusters using trees. → Dendogram
- trees use features to navigate the instance space similar to decision trees

- We start with every data point in a separate cluster
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left
- This is called a bottom-up or agglomerative method

# *Dendrogram*

- *Given a data set D, a dendrogram is a binary tree with the elements of D at its leaves.*
- *An internal node of the tree represents the subset of elements in the leaves of the subtree rooted at that node.*
- *The level of a node is the distance between the two clusters represented by the children of the node.*
- *Leaves have level 0.*



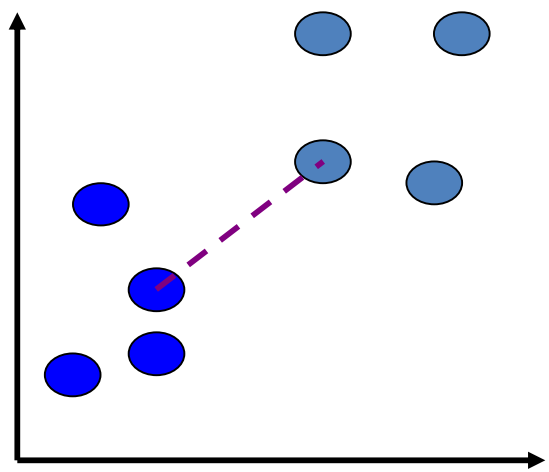

- This produces a binary tree or ***dendrogram***
- The final cluster is the root and each data item is a leaf
- The height of the bars indicate how close the items are

- We already know about distance measures between data items, but what about between a data item and a cluster or between two clusters?
- We just treat a data point as a cluster with a single item, so our only problem is to define a ***linkage*** method between clusters
- As usual, there are lots of choices…

# Single Linkage

$x_1$
$x_2$
$x_3$

- The minimum of all pairwise distances between points in the two clusters
- Tends to produce long, “loose” clusters

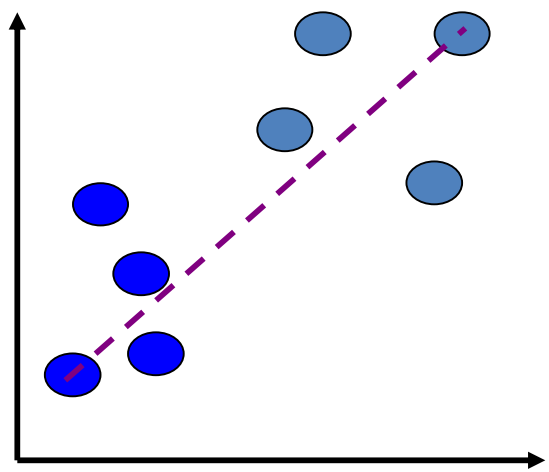


# Average Linkage

- Eisen's cluster program defines average linkage as follows:
  - Each cluster $c_i$ is associated with a mean vector $\mu_i$ which is the mean of all the data items in the cluster
  - The distance between two clusters $c_i$ and $c_j$ is then just $d(\mu_i, \mu_j)$
- This is somewhat non-standard – this method is usually referred to as centroid linkage and average linkage is defined as the average of all pairwise distances between points in the two clusters

# Complete Linkage

- The maximum of all pairwise distances between points in the two clusters
- Tends to produce very tight clusters



- Given Data instances

A B C D E F

- X = { (1,1),(1.5,1.5),(5,5),(3,4),(4,4),(3,3.5)}

| Distance | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 0 | 0.71 | 5.66 | 3.61 | 4.24 | 3.20 |
| B | 0.71 | 0 | | | | |
| C | 5.66 | | 0 | | | |
| D | 3.61 | | | 0 | | |
| E | 4.24 | | | | 0 | |
| F | 3.20 | | | | | 0 |

1) Single linkage

$d(A,B)$

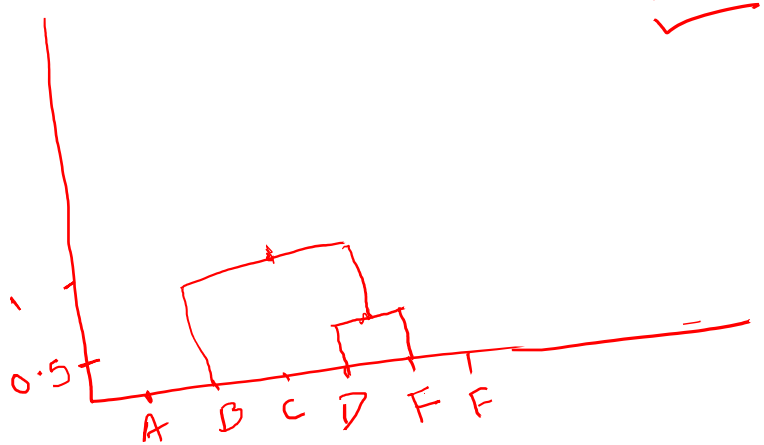
$= \sqrt{(1-1.5)^2+(1-1.5)^2}$

$= \sqrt{(-0.5)^2+(-0.5)^2}$

$= \sqrt{0.25+0.25} = \sqrt{0.5}$

$= \underline{\underline{0.71}}$

| 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 |




(B, D, F) ×

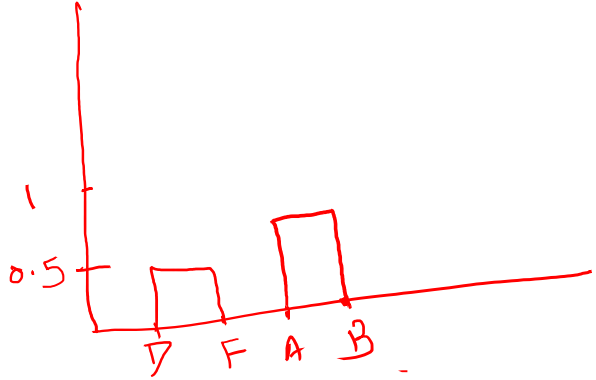
{B, {D, F}}

## 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 |

min((A,D), (A,F))

min(3.61) (3.20)




## 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 | ? 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 |

((D,F),E)

## 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 |

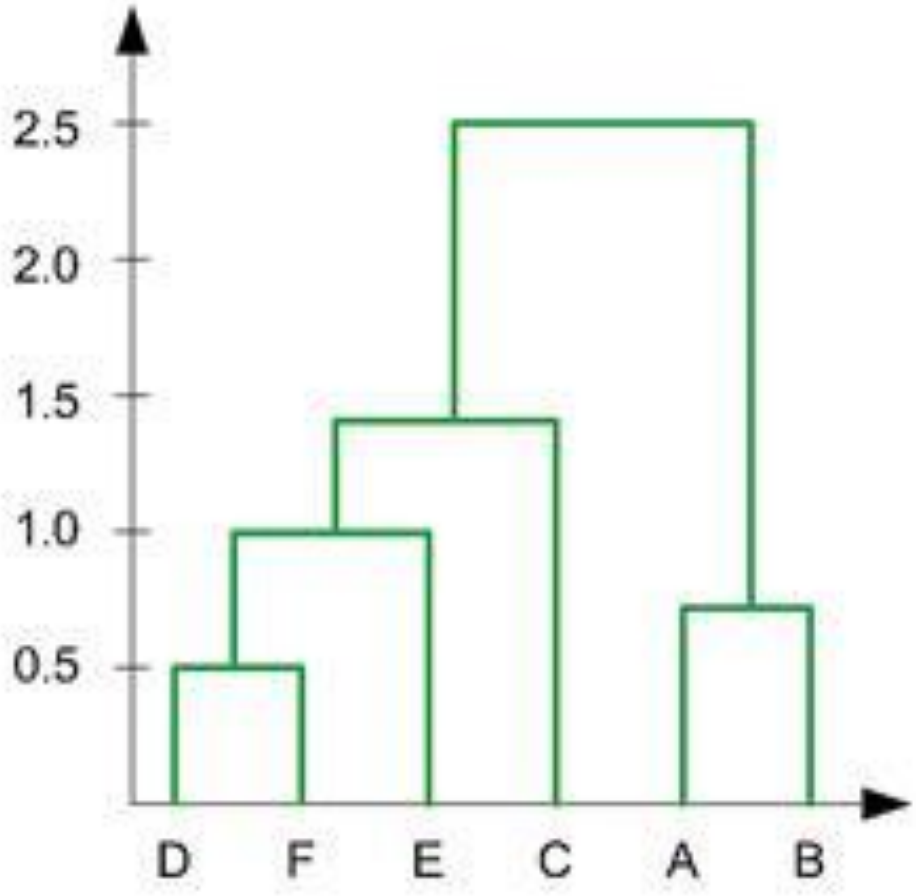
# Min Distance (Single Linkage)

| 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 |

(((D, F), E), C)

# 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 |


2.5
2.0
1.5
1.0
0.5
D
F
E
C
A
B

- **Rule Based Models:**
  - **Rule learning for subgroup discovery, Association rule mining.**

# Rule Based models

- Logical Models
- Rule has two parts
  - Body
  - head

$X \rightarrow Y$

Transactions

- Most rule learning algorithms work on the strategy called *separate-and-conquer*
- They search for a rule that explains some part of the data, separate these examples and recursively conquer the remaining examples.

# Rule Based Models

- Rule
  - A notation that represents which item is brought with what items.
- Itemset A => itemsetB
  - Items on the right were frequently bought along with items on the left
- e.g {milk, bread} => {butter}

# Definitions

1) Milk Bread, Butter
2) Milk eggs
3) Bread, eggs, salt
4) Milk eggs, salt

| | | |
|---|---|---|
| Milk | 3 | 3/4 ×100 75% |
| Bread | 2 | 50% |
| Butter | 1 | 25% |
| eggs | 3 | 75% |
| salt | 2 | 50% |

- Item
- Item set
- Transaction
- Support count
  - Sup(I) = no of transactions t that support (Contains) I
    - I is an Item set
- Frequent Item set is one with at least the minimum support count(minsup)

- Association Rule R
  - X => Y
- X and Y are Disjoint
- Y is non empty set

# Rule Support and Confidence

- Support
  - Proportion of transactions in the data set which contain Itemset

- Confidence of a rule
- Conf(X =>Y) = supp(X U Y)/supp(X)

# Association Rule mining task

- Given a set of Transactions T,the goal of association rule mining is to find all rules having
  - Support >= minsup
  - Confidence >= minconf

- Apporaches
  - Brute force

# Apriori Algorithm

- K=1
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - Generate length(k+1) candidate itemsets from length k frequent itemsets
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate
  - Eliminate candidates that are infrequent

- For the following given Transaction Data-set, Generate Rules using Apriori Algorithm. Consider the values as Support=50% and Confidence=75%

| Transaction ID | Items Purchased |
|---|---|
| 1 | Bread, Cheese, Egg, Juice |
| 2 | Bread, Cheese, Juice |
| 3 | Bread, Milk, Yogurt |
| 4 | Bread, Juice, Milk |
| 5 | Cheese, Juice, Milk |

| Item | ~~Support~~ frequency | Support (%) |
|---|---|---|
| Bread | 4 | 4/5 (80%) |
| Cheese | 3 | 3/5 (60%) |
| ~~Egg~~ → | ~~1~~ | ~~1/25~~ ~~20%~~ |
| Juice | 4 | 4/5 80% |
| Milk | 3 | 60% |
| ~~Yogurt~~ → | ~~1~~ | ~~20%~~ |

| | | |
|---|---|---|
| Bread | 4 | 80% |
| Cheese | 3 | 60% |
| Juice | 4 | 80% |
| mik | 3 | 60% |
| 2) Bread, cheese | 2 | 40 |
| Bread, Juice | 3 | 60 |
| Bread, Milk | 2 | 40 |
| Cheese, Juice | 3 | 60 |
| cheese, Milk | 1 | 20 |
| Juice Milk | 2 | 40 |

$$conf(X \rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}$$

3

| | | |
|---|---|---|
| Bread Juice | 3 | 60% |
| ~~Bread~~ Juice Cheese | 3 | 60% |
| | * | |
| | | |
| Bread → Juice | 3/4 3/5/4/5 | 75% |
| Juice → Bread | 3/4 | 75% |
| Juice → cheese | ~~3/10~~ 3/3 | ~~100~~ 100 |
| cheese → Juice | 3/4 | 75% |
| | | |

| L2 | L3 | L4 |
|---|---|---|
| Bread, Juice | Bread, Juice, cheese | Bread, Juice, cheese |
| Bread, cheese | ~~Bread, Juice, egg~~ | egg |
| Juice milk | ~~Bread~~ cheese egg | |
| Juice cheese | Juice, milk, ~~cheese~~ | |
| Bread, egg | Bread Juice cheese | |
| | Bread cheese egg | |
| | | |

minsup = 3 ✓ %

conf. 0.5 ✓ %

3/4

= 0.75

| | |
|---|---|
| 1 | B, C |
| 2 | B, C |
| 3 | A, C, D |
| 4 | A, B, C, D |
| 5 | B, D |

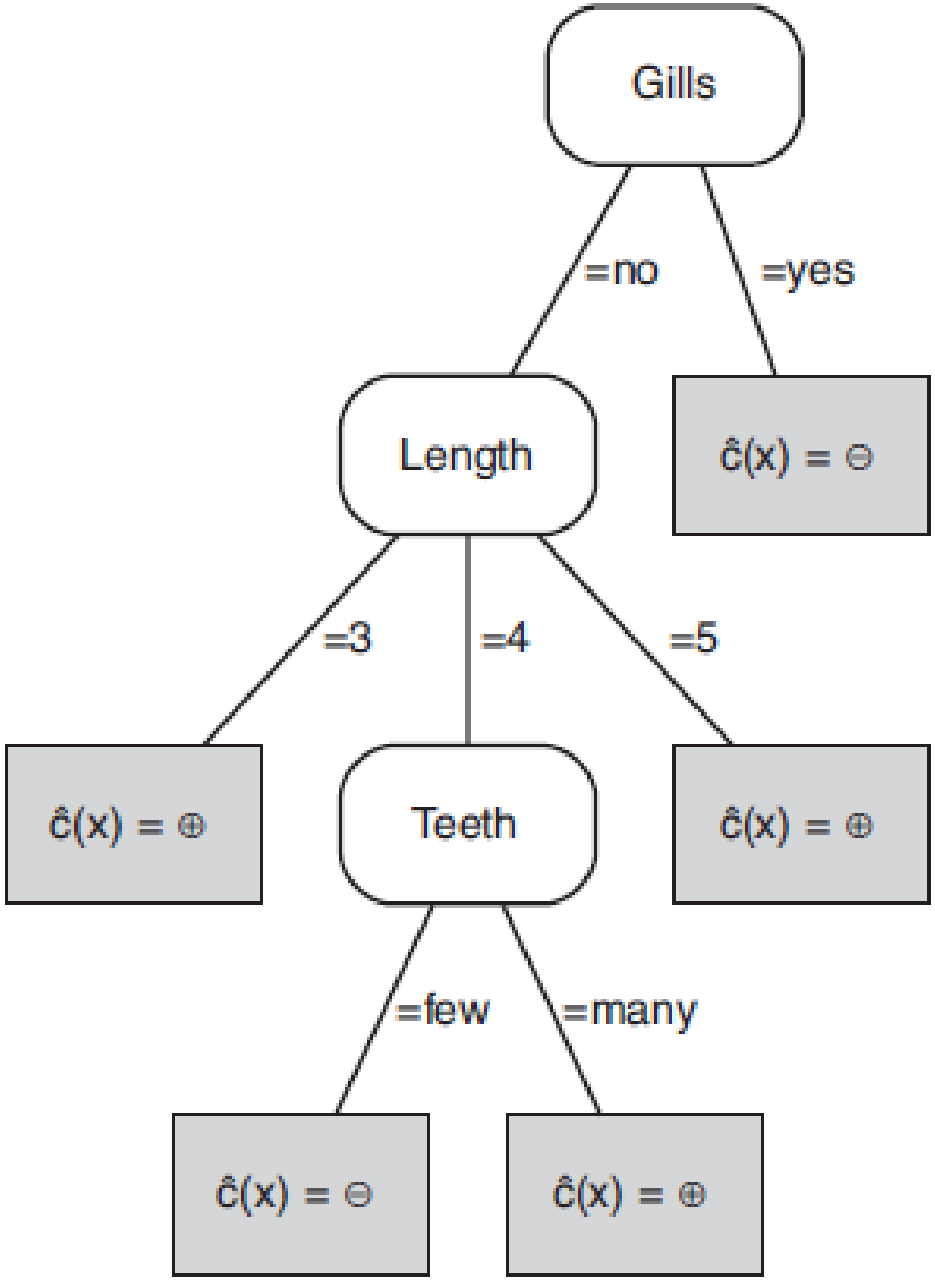
| | |
|---|---|
| **Notebook** | |
| Inkpen nInk bottle | |
| Notebook ,DVD | |
| DVD | |
| DVD,Inkbottle | |
| DVD,Notebook,Inkpen,Inkbottle | |
| Ink bottle | |

# Tree Based Models

- **Decision Trees**
- **Ranking and Probability estimation Trees**
- **Regression trees**
- **Clustering Trees.**

# Decision Tree

- The most popular models in machine learning.
- Trees are expressive and easy to understand
- recursive ‘divide-and-conquer’ nature.
- This tree can be turned into a logical expression
- Tree models are not limited to classification but can be employed to solve almost any machine learning task, including ranking and probability estimation, regression and clustering.


Gills
=no
=yes
Length
ĉ(x) = ⊖
=3
=4
=5
ĉ(x) = ⊕
Teeth
ĉ(x) = ⊕
=few
=many
ĉ(x) = ⊖
ĉ(x) = ⊕

- **A feature tree is a tree such that each internal node(the nodes that are not leaves) is labeled with a feature, and each edge emanating from an internal node is labeled with a literal.**
- **The set of literals at a node is called a split.**
- **Each leaf of the tree represents a logical expression, which is the conjunction of literals encountered on the path from the root of the tree to the leaf.**
- **The extension of that conjunction (the set of instances covered by it) is called the instance space segment associated with the leaf.**

# Important Definitions

- Pure
- Impurity
- Entropy : measure of Purity
  - Indicates how mixed the class values are
  - Given by formula

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

- Weighted Average impurity is calculated by

$$\mathrm{Imp}(\{D_1, \ldots, D_l\}) = \sum_{j=1}^{l} \frac{|D_j|}{|D|} \mathrm{Imp}(D_j)$$

**Homogeneous**($D$) returns true if the instances in $D$ are homogeneous enough to be labelled with a single label, and false otherwise;

**Label**($D$) returns the most appropriate label for a set of instances $D$;

**BestSplit**($D$,$F$) returns the best set of literals to be put at the root of the tree.

**Algorithm 5.1:** GrowTree($D, F$) – grow a feature tree from training data.

**Input** : data $D$; set of features $F$.

**Output** : feature tree $T$ with labelled leaves.

```
if Homogeneous(D) then return Label(D);          // Homogeneous, Label: see text
S ← BestSplit(D, F);                             // e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i do
    if D_i ≠ ∅ then T_i ← GrowTree(D_i, F) else T_i is a leaf labelled with Label(D);
end
return a tree whose root is labelled with S and whose children are T_i
```

**Algorithm 5.2:** BestSplit-Class($D, F$) – find the best split for a decision tree.

**Input** : data $D$; set of features $F$.

**Output** : feature $f$ to split on.

```
I_min ← 1;
for each f ∈ F do
    split D into subsets D_1, ..., D_l according to the values v_j of f;
    if Imp({D_1, ..., D_l}) < I_min then
        I_min ← Imp({D_1, ..., D_l});
        f_best ← f;
    end
end
return f_best
```

p1: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many

p2: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many

p3: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

p4: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many

p5: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

n1: Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many

n2: Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many

n3: Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many

n4: Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many

n5: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

| | |
|---|---|
| Length = [3, 4, 5] | [2+, 0−][1+, 3−][2+, 2−] |
| Gills = [yes, no] | [0+, 4−][5+, 1−] |
| Beak = [yes, no] | [5+, 3−][0+, 2−] |
| Teeth = [many, few] | [3+, 4−][2+, 1−] |

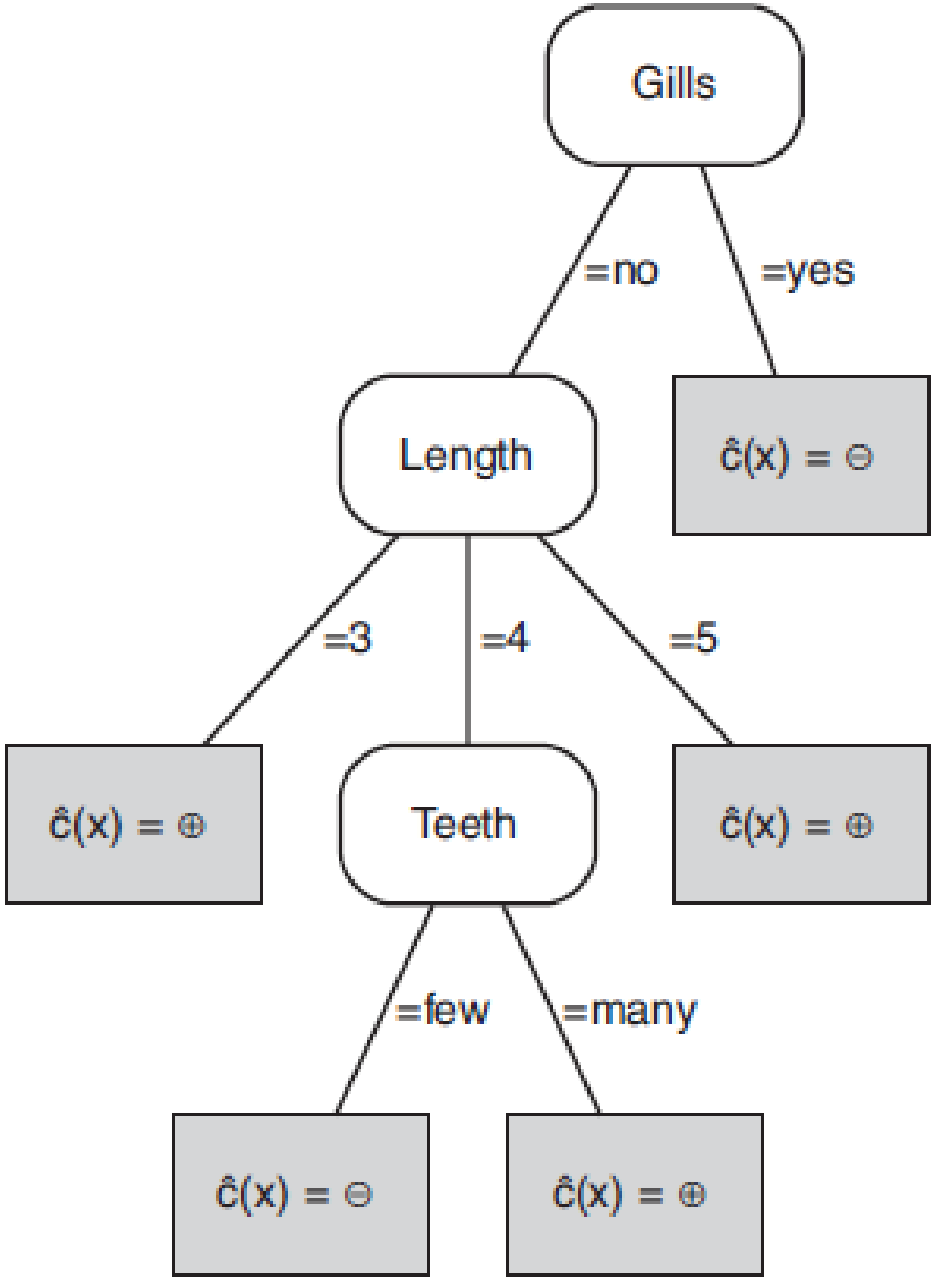
**Length 2/10.0 +4/10.0.81+4/10.1 = 0.72**

Gills $4/10 \cdot 0 + 6/10 \cdot \left(-(5/6)\log_2(5/6) - (1/6)\log_2(1/6)\right) = 0.39;$

Beak $8/10 \cdot \left(-(5/8)\log_2(5/8) - (3/8)\log_2(3/8)\right) + 2/10 \cdot 0 = 0.76;$

Teeth $7/10 \cdot \left(-(3/7)\log_2(3/7) - (4/7)\log_2(4/7)\right)$

$+3/10 \cdot \left(-(2/3)\log_2(2/3) - (1/3)\log_2(1/3)\right) = 0.97.$


Gills
=no
=yes
Length
ĉ(x) = ⊖
=3
=4
=5
ĉ(x) = ⊕
Teeth
ĉ(x) = ⊕
=few
=many
ĉ(x) = ⊖
ĉ(x) = ⊕

- Gini Index $(1-\dot{p})\,\dot{p}$
- Minority class $1/2-\left|\dot{p}-1/2\right|$
- Square root of Gini Index

# Rank and Probability Estimation Tree

- Decision tress whose leaves represents ranks or probabilities is called as Ranking or probability estimation Tree.
- Aim in these trees is not classifying training instances but to find instances which are most likely to belong particular class.
- Input to this model is training instances and out put is probability estimation tree

- To convert Decision tree into ranking tree
  - Order Leaf nodes in non increasing empirical probabilities

- To convert Decision tree into Probability Estimation Tree
  - Find probability for each tree node with Laplace estimator.

# Regression Trees

- Here we replace impurity measure by Var

$$\mathrm{Var}(Y) = \frac{1}{|Y|} \sum_{y \in Y} (y - \overline{y})^2$$

where $\overline{y} = \frac{1}{|Y|} \sum_{y \in Y} y$ is the mean of the target values in $Y$;

# Clustering Trees

- Dissimilarity is calculated to calculate the split

$$\mathrm{Dis}(D) = \frac{1}{|D|^2} \sum_{x \in D} \sum_{x' \in D} \mathrm{Dis}(x, x')$$