# ACDS Lecture Series

Lecture - 10

CSIR

# Supervised and Unsupervised Learning

G. N. Sastry and Team

ADVANCED COMPUTATION AND DATA SCIENCES (ACDS) DIVISION

CSIR-North East Institute of Science and Technology, Jorhat, Assam, India

# 10. Supervised and Unsupervised Learning

10.1 Introduction:

10.2 Supervised Learning

- 10.2.1 Classification
  - 10.2.1.1 Random Forest
  - 10.2.1.2 Decision Trees
  - 10.2.1.3 Logistic Regression
  - 10.2.1.4 Support Vector Machines
  - 10.2.1.5 KNN
  - 10.2.1.6 Naïve Bayes
  - 10.2.1.7 Examples
- 10.2.2 Regression
  - 10.2.2.1 Linear Regression
  - 10.2.2.2 Regularization Techniques (LASSO)
  - 10.2.2.3 Polynomial Regression
  - 10.2.2.4 Examples

10.3 Unsupervised Learning

- 10.3.1 Clustering
  - 10.3.1.1 K-Means
  - 10.3.1.2 K Nearest Neighbors
  - 10.3.1.3 Association Rule Learning
- 10.3.2 Dimensionality Reduction
  - 10.3.2.1 PCA
  - 10.3.2.2 LDA
  - 10.3.2.3 GDA
  - 10.3.2.4 Examples

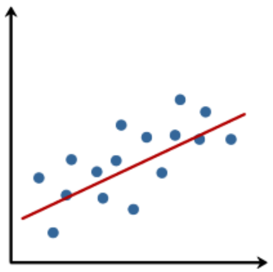
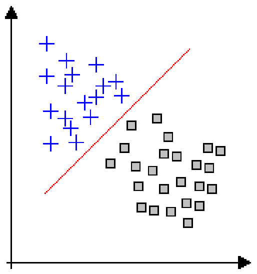
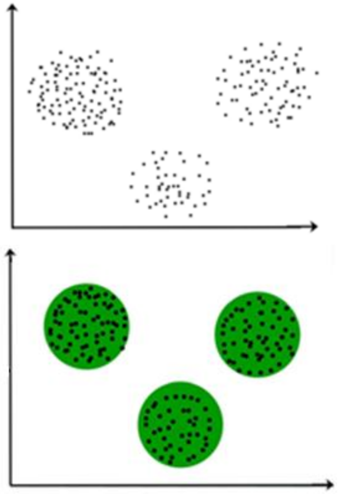
10.4 Exercises

10.5 References

- 10.5.1 Books
- 10.5.2 Video Lectures

# 10.1 Introduction

Machine Learning

| Supervised<br>(Predictive) | | Unsupervised<br>(Descriptive) | |
|---|---|---|---|
| Regression | Classification | Clustering | Association |
| Predicting a continuous output<br> | Predicting the discrete class<br> | Finding natural grouping<br> | Finding associations among variables<br>{X1, X2} → {Y1, Y2}<br>Antecedent Consequent<br>Itemset = {X1, X2, Y1, Y2} |

# 10.2 Supervised Learning

## The Topics Covered in This Section

10.2.1 Classification
- 10.2.1.1 Random Forest
- 10.2.1.2 Decision Trees
- 10.2.1.3 Logistic Regression
- 10.2.1.4 Support Vector Machines
- 10.2.1.5 KNN
- 10.2.1.6 Naïve Bayes
- 10.2.1.7 Examples

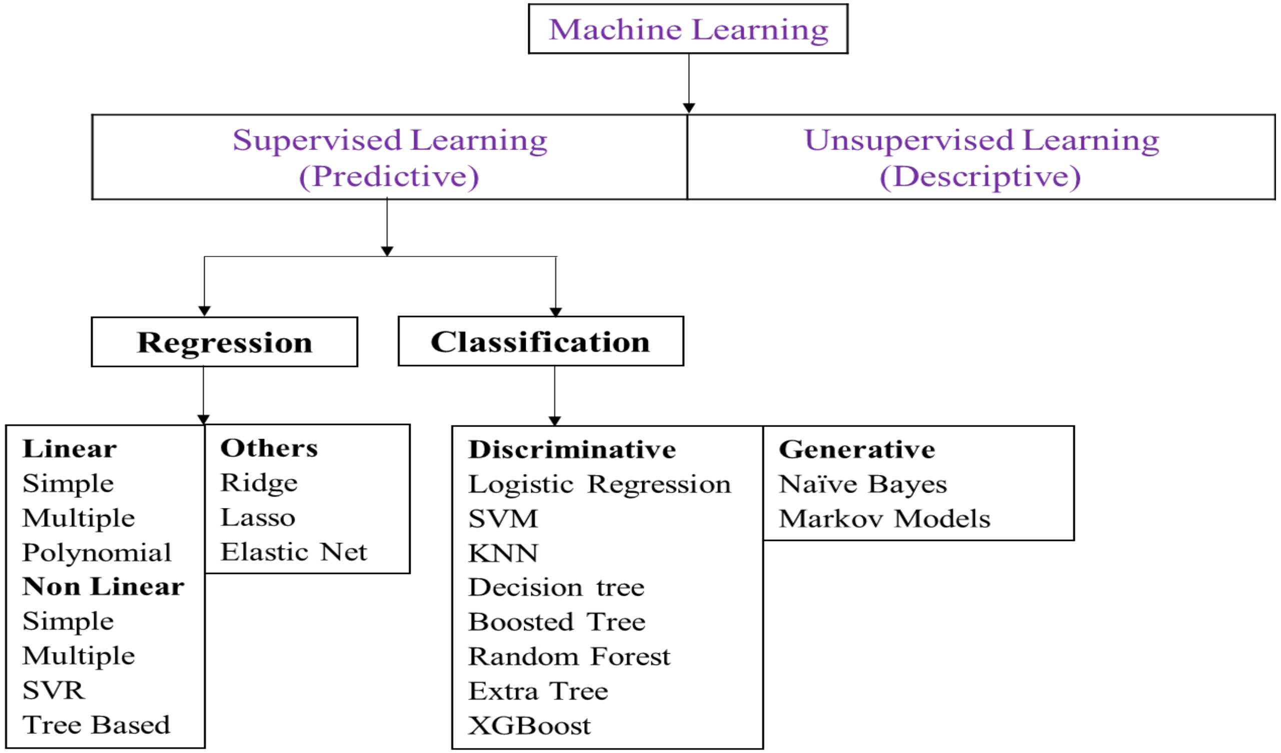
10.2.2 Regression
- 10.2.2.1 Linear Regression
- 10.2.2.2 Regularization Techniques (LASSO)
- 10.2.2.3 Polynomial Regression
- 10.2.2.4 Examples

# 10.2 Supervised Learning

Supervised learning is where there is an input variables (x) and an output variable (y) and an algorithm to learn the mapping function from the input to the output.

$$y = f(x)$$

The goal is to approximate the mapping function so well that when there is a new input data (x) that the model can predict the output variables (y) for that data.



Machine Learning

Supervised Learning (Predictive)

Unsupervised Learning (Descriptive)

**Regression**

**Classification**

**Linear**
Simple
Multiple
Polynomial
**Non Linear**
Simple
Multiple
SVR
Tree Based

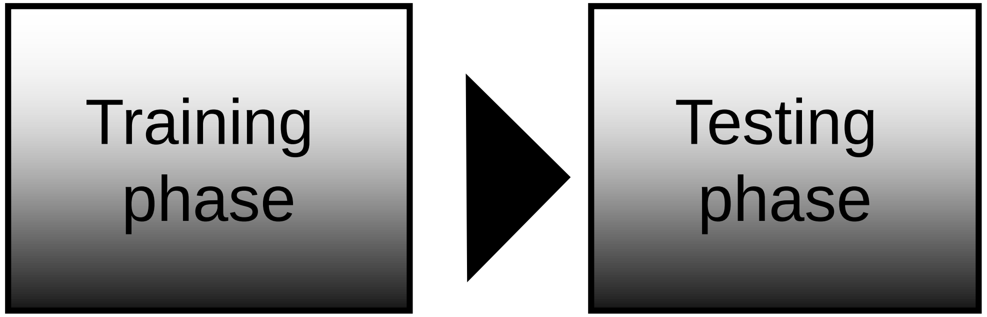
**Others**
Ridge
Lasso
Elastic Net

**Discriminative**
Logistic Regression
SVM
KNN
Decision tree
Boosted Tree
Random Forest
Extra Tree
XGBoost

**Generative**
Naïve Bayes
Markov Models

# 10.21 Classification

## Classification learning



Learning the classifier from the available data ‘Training set’ (Labeled)

Testing how well the classifier performs ‘Testing set’

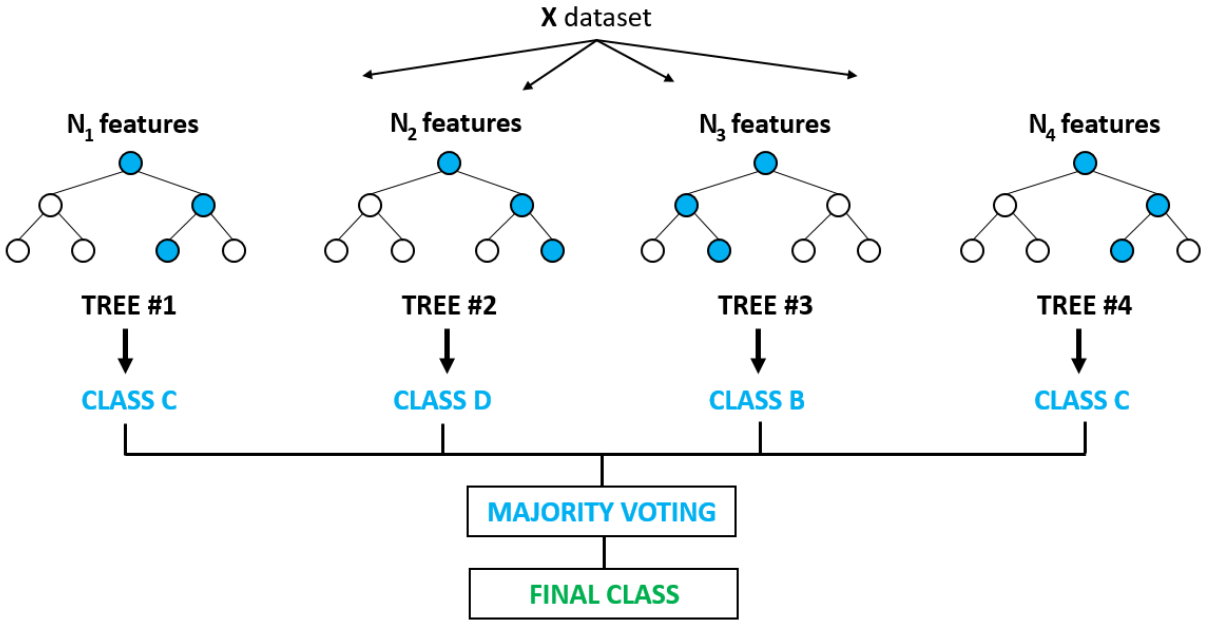
| Discriminative | Generative |
|---|---|
| Logistic Regression | Naïve Bayes |
| SVM | Hidden Markov Models |
| KNN | Gaussian Mixture Model |
| Decision tree | |
| Boosted Tree | |
| Random Forest | |
| Extra Tree | |
| Regularized Greedy Forest | |

## Generating datasets

- Methods:
  - Holdout (2/3rd training, 1/3rd testing)
  - Cross validation (n – fold)
    - Divide into n parts
    - Train on (n-1), test on last
    - Repeat for different combinations
  - Bootstrapping
    - Select random samples to form the training set

# 10.2.1.1 Random Forest

- An ensemble learning method for by constructing a multiple decision trees.
- The basic difference is that RF is a collection or ensemble model of numerous DT

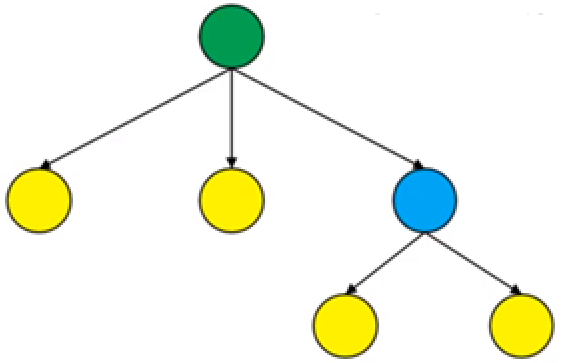


**Why Random Forest?**

- One single DT would lead to over-fit model if the dataset is huge, same way like a single person might have its own perspective on the complete population.
- However if we implement the voting system and ask different individuals to interpret the data then we would be able to cover the patterns in a much meticulous way.

# 10.21.2 Decision Trees

- A decision support tool that uses a tree like model of decisions and their possible consequences including chance of event outcomes.
- Decision tree uses divide and conquer technique for the basic learning strategy.



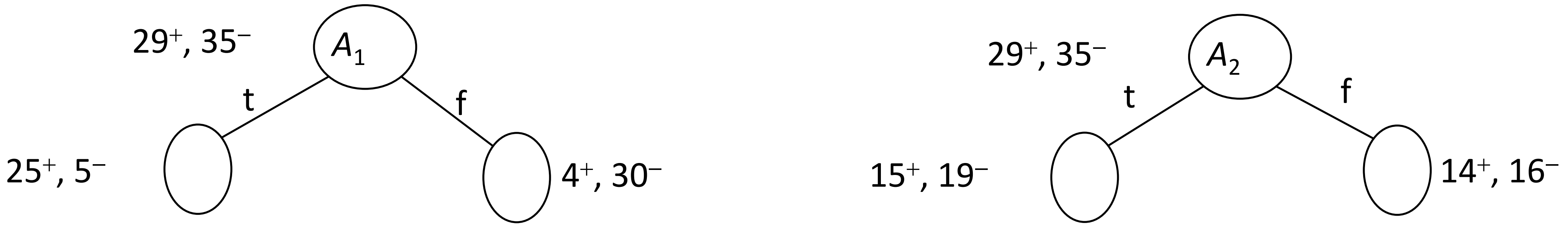
- Decision tree of types –
  - **Classification tree:** when the predicted outcome is the class to which the data belong.
  - **Regression tree:** when the predicted outcome is a continuous real number.
- Both the classification and regression trees have similarities as well as differences, such as procedure used to determine where to split.

Note: Classification and Regression Tree (CART) is an umbrella term used to refer both types.

# 10.21.2 Decision Trees

## Selecting Best Split

- Consider 64 examples, $29^{+}$ and $35^{-}$
- Which one is better?



**Methods**

- **Information Gain/Entropy Approach**
  - ID3 (Iterative Dichotomiser 3)
  - C4.5
  - CART (Classification and Regression Tree)
  - CHAID(CHi-squared Automatic Interaction Detector)
  - MARS
- **Gini Index Approach (*For Noisy, High Dimensional)**

# 10.21.2 Decision Trees

## Selecting Best Split

### Entropy Based Approach

- The Shannon Entropy, H(X) of a discrete variable X with possible values $X_1$ $X_2$….$X_n$ and probability mass function P(X) is defined as

$$H(X) = -\sum_{i=1}^{n} P(X_i) \ln P(Xi)$$

For Completely homogenous dataset: Entropy is 0
For Completely heterogonous dataset: Entropy is 1

- Entropy with respect to a given predictor/feature

$$E(T, X) = \sum_{i=1}^{n} P(X_i)\, \mathrm{H}(Xi)$$

A branch with Entropy more than 1 needs splitting.
Leaf nodes have entropy 0

- Information Gain

$$IG = H(X) – E(T,X)$$

Feature/ Attribute with highest IG be the Root node.
Root node has maximum information gain.

# 10.21.2 Decision Trees

## Selecting Best Split

**Gini Index Approach**

- If a dataset D contains examples from n classes, Gini index of D defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where, $p_j$ is the relative frequency of class j in D

- If a dataset D split on A into two subset D1 and D2, Gini index of D defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity $\Delta gini(A) = gini(D) - gini_A(D)$

# 10.21.2 Decision Trees

## Handling Overfitting

- A tree that classifies the training data perfectly may not lead to best generalization performance.
  - ➢ There may be noise in the training data the tree is fitting.
  - ➢ The algorithm might be making decisions based on very little data.

A hypothesis *h* is said to overfit the training data

- ➢ if there is another hypothesis, h', such that *h* has smaller error than h' on the training data but *h* has larger error on the test data than *h'*.

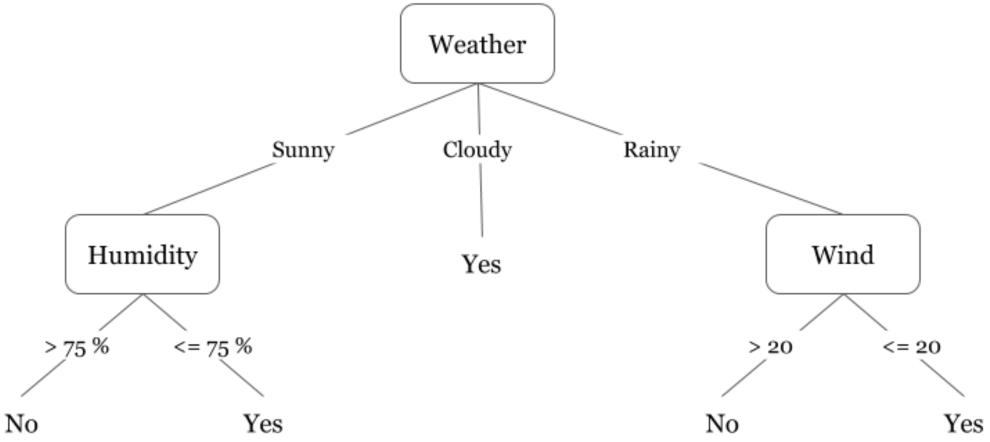
**Avoiding Overfitting**

- Two basic approaches
  - ➢ Prepruning: Stop growing the tree at some point when there is not enough data.
  - ➢ Postpruning: Grow the full tree and then remove nodes not having sufficient evidence.

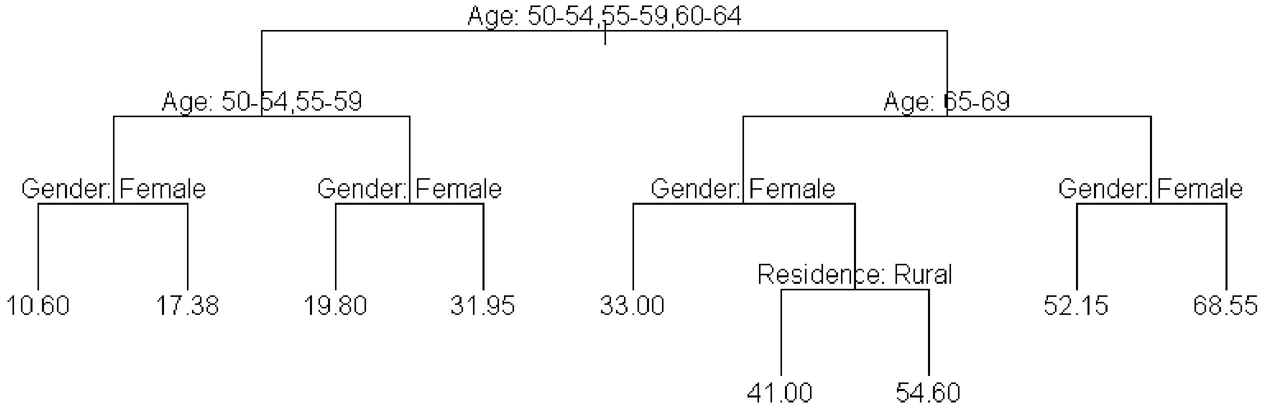
# 10.21.2 Decision Trees

## Advantages

- What if some examples are missing values of attribute *A*?
- Use training example anyway, sort through tree
    - If node *n* tests *A*, assign most common value of *A* among other examples sorted to node *n*
    - Assign most common value of *A* among other examples with same target value
    - Assign probability $p_i$ to each possible value $v_i$ of *A*
        - Assign fraction $p_i$ of example to each descendant in tree
- Classify test instances with missing values in same fashion.



Classification Tree



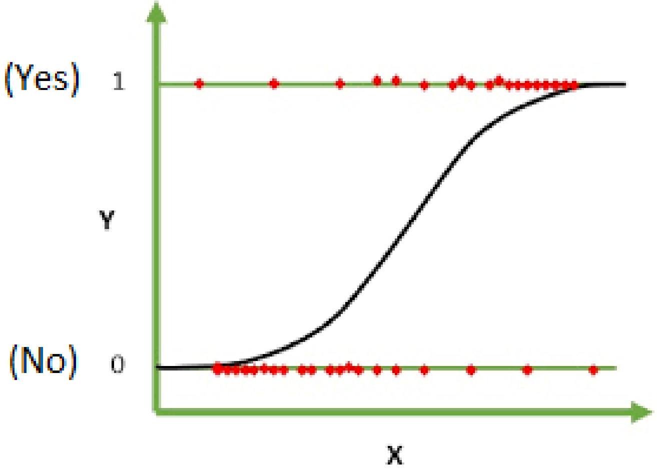
Regression Tree

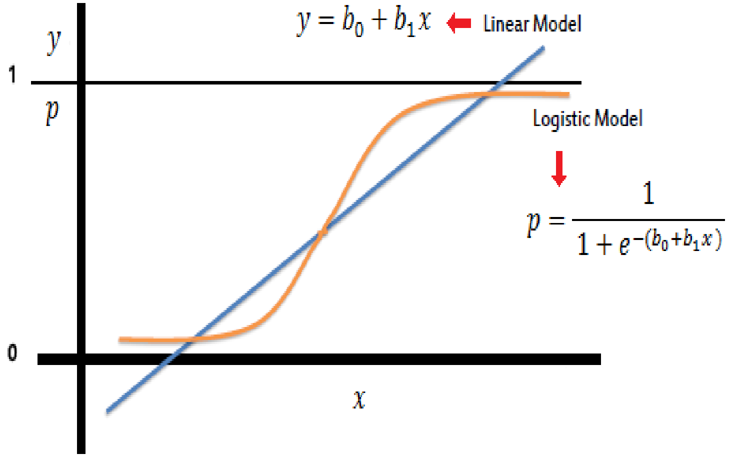
Instead of computing information gain, compute the RMS error

# 10.21.3 Logistic Regression

- Used in Classification (binary or multiclass) problem where *y* is a discrete value.
- Set threshold based on which classifier classifies.

| Size of Tumor | Malignant |
|---|---|
| 2 | N |
| 1 | N |
| 1 | N |
| 8 | Y |
| … | … |
| … | … |

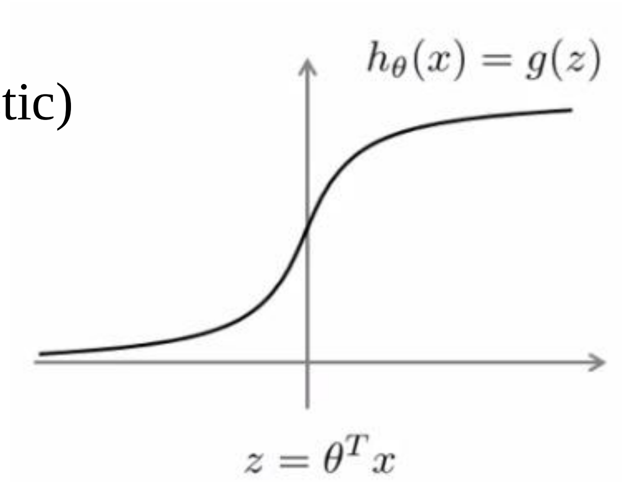




# 10.21.3 Logistic Regression

## Hypothesis

- Regression hypothesis can give values large than 1 or less than 0
- Logistic regression generates a value always between 0 or 1 (probabilistic)
- In linear regression we did $h_\theta(x) = (\theta^T x)$ $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$
- For classification hypothesis representation we do $h_\theta(x) = g((\theta^T x))$



- If we combine these equations we can write out the hypothesis as $g(z) = (z) = 1/(1 + e^{-z})$

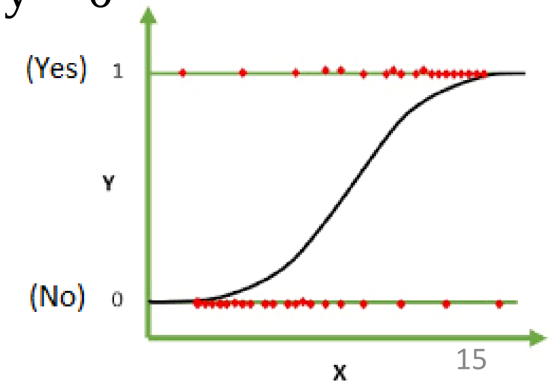
Where the **sigmoid** or **logistic function**

- When the probability of $y > 0.5$ then we can predict $y = 1$ Else we predict $y = 0$

$g(z) >= 0.5$ when $z\ (=\theta^T x) >= 0$

$g(z) < 0.5$ when $z\ (=\theta^T x) < 0$

- Multivariate logistic Regression $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
- Polynomial logistic Regression $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_3 x_1^2 + \theta_4 x_2^2)$



# 10.21.3 Logistic Regression

## Cost Function

- Linear regression uses the following function to determine θ

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

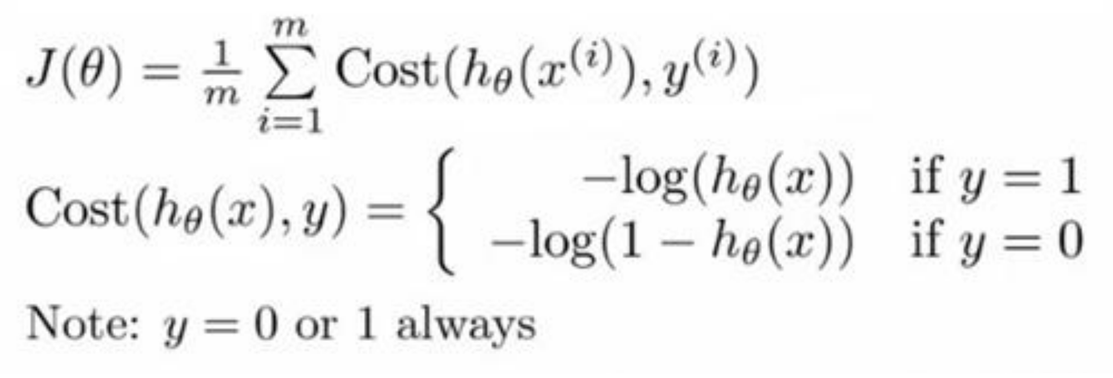
- Instead of writing the squared error term, we can write
  - If we define "cost()" as $\quad$ $\text{cost}(h_\theta(x^{(i)}), y) = ½ (h_\theta(x^{(i)}) - y^{(i)})^2$
  - We can **redefine J(θ) as**

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

# 10.21.3 Logistic Regression

## Optimization

**Cost Function**

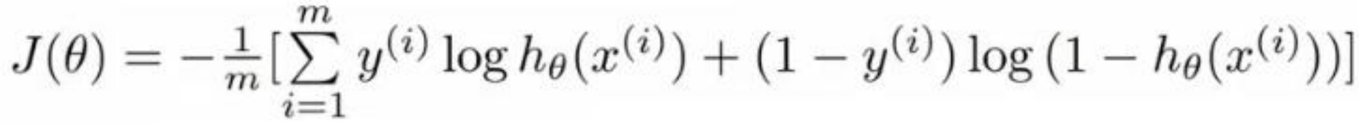


$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

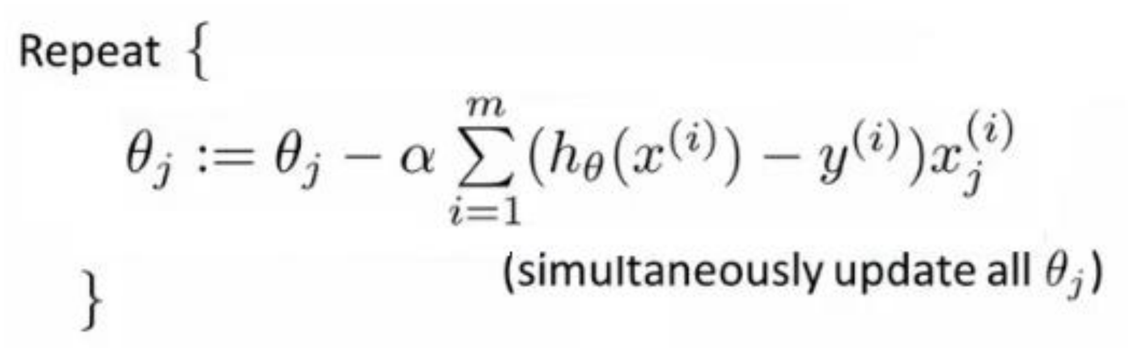
Note: $y = 0$ or $1$ always

Minimize cost function



$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

**Gradient Descent**



Repeat $\{$

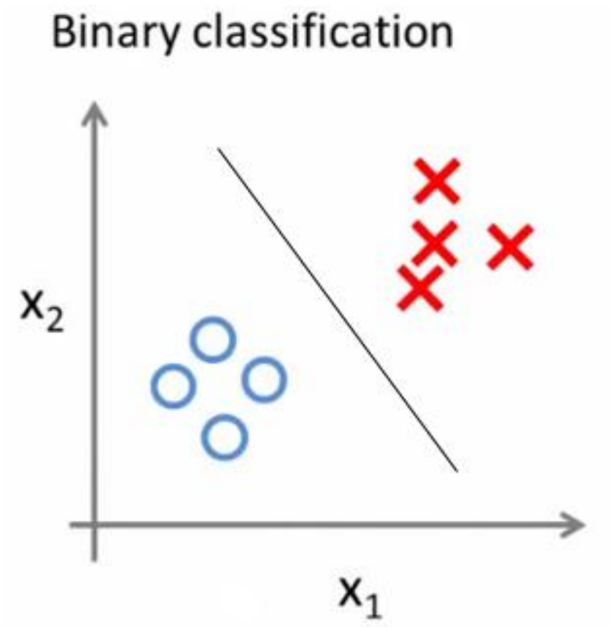
$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

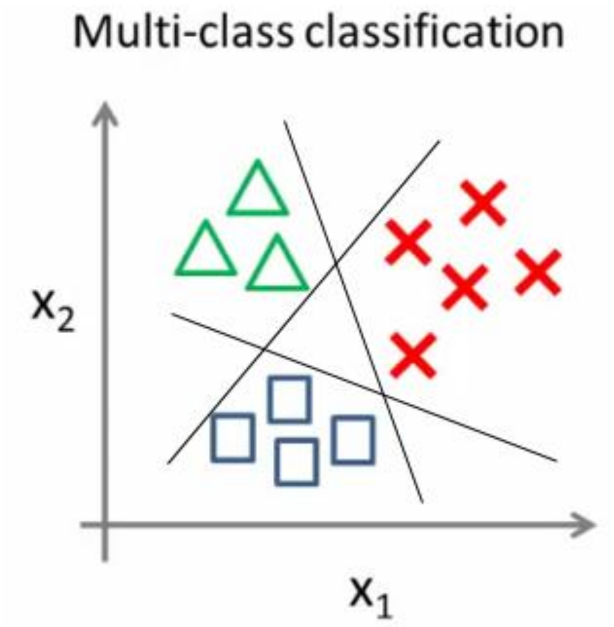
(simultaneously update all $\theta_j$)

$\}$

# 10.21.3 Logistic Regression

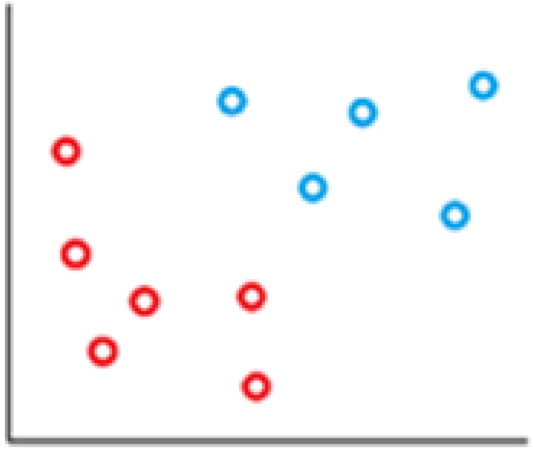
## Multiclass Classification



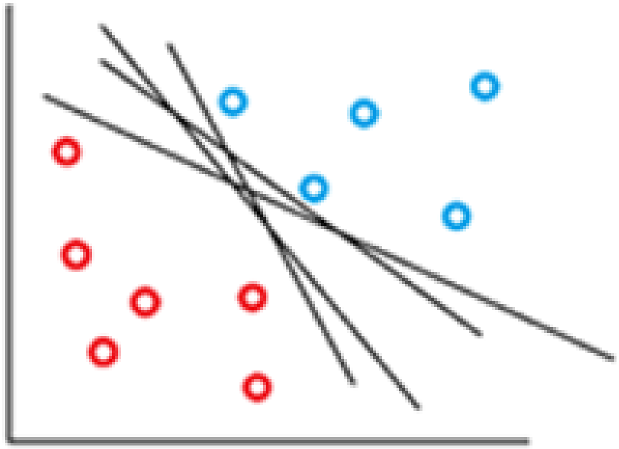


- Split the training set into three separate binary classification problems.
  - Triangle (1) vs crosses and squares (0) $h_\theta^1(x)$ $p(y=1 \mid x_1; \theta)$
  - Crosses (1) vs triangle and square (0) $h_\theta^2(x)$ $p(y=1 \mid x_2; \theta)$
  - Square (1) vs crosses and square (0) $h_\theta^3(x)$ $p(y=1 \mid x_3; \theta)$

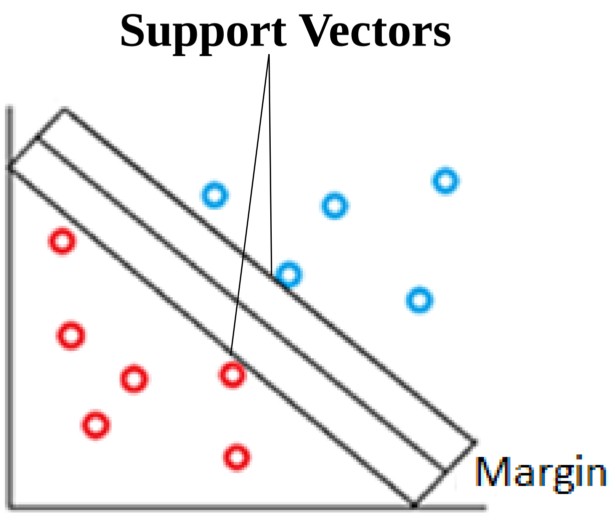
# 10.21.4 Support Vector Machine



Supervised data



Decision Boundary



Optimal Decision Boundary
(Large Margin)

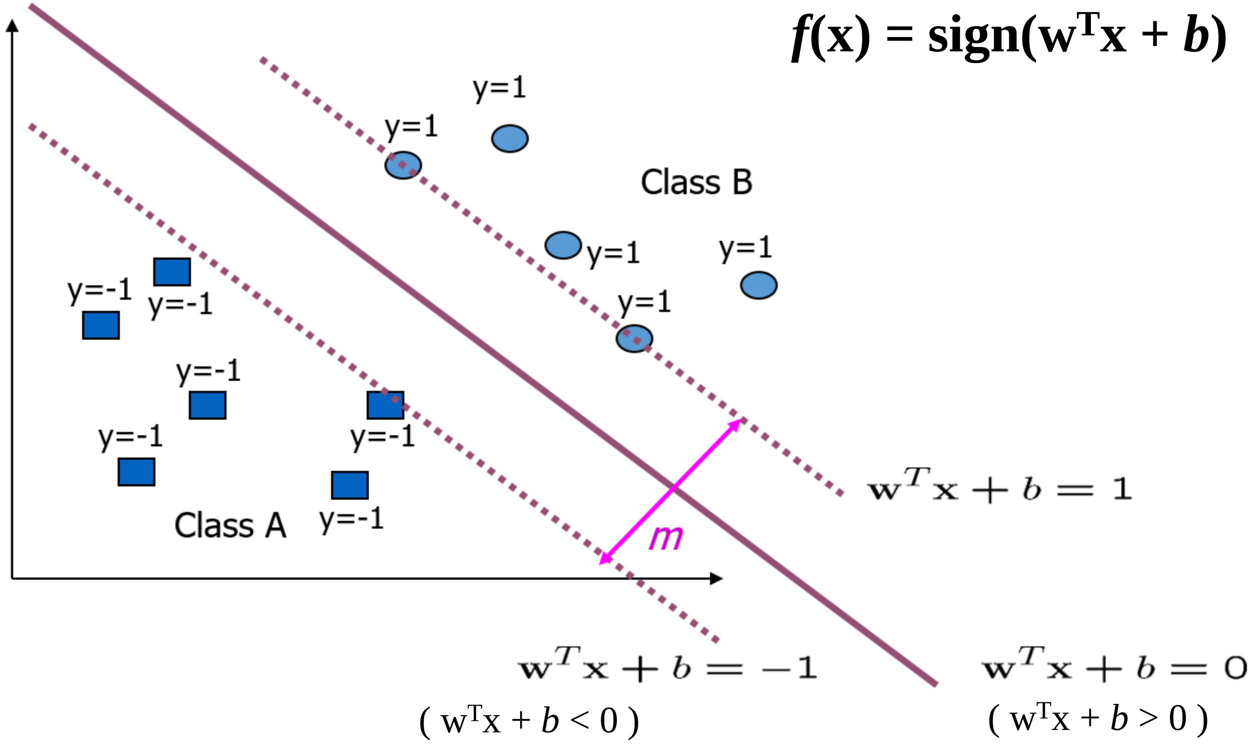
# 10.21.4 Support Vector Machine

## Decision Boundary

- Let {$x_1$, ..., $x_n$} be our data set and let $y_i$   {1,-1} be the class label of $x_i$

- For $y_i$=1   $w^T x_i + b \geq 1$
- For $y_i$=-1   $w^T x_i + b \leq -1$

$$y_i \cdot \left(w^T x_i + b\right) \geq 1, \forall (x_i, y_i)$$



$f$(**x**) = **sign(w$^T$x +** $b$**)**

y=1
y=1
Class B
y=1
y=1
y=1
y=-1
y=-1
y=-1
y=-1
y=-1
Class A
y=-1
m
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = -1$
( w$^T$x + $b$ < 0 )
$\mathbf{w}^T\mathbf{x} + b = 0$
( w$^T$x + $b$ > 0 )

**Margin**

- The decision boundary should be as far away from the data of both classes as possible.
- Goal is to maximize the margin ($m$)

$$\boxed{m = \frac{2}{||\mathbf{w}||}}$$

# 10.21.4 Support Vector Machine

## Optimization

- The decision boundary should classify all points correctly for $y_i$=1

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

- The decision boundary can be found by solving the following constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

- This is a constrained optimization problem.
- Solving it requires to use Lagrange multipliers.

# 10.21.4 Support Vector Machine

## Optimization

- **The optimization problem**

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0 \qquad \text{for } i = 1, \ldots, n$$

- **The Lagrangian is**

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

Note: $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

**Note:** Lagrange multipliers are used in multivariable calculus to find maxima and minima of a function subject to constraints (like "find the highest elevation along the given path" or "minimize the cost of materials for a box enclosing a given volume").

# 10.21.4 Support Vector Machine

## Optimization

**Gradient with respect to *w* and *b***

- Setting the gradient of L w.r.t. **w** and b to zero, we have

$$L = \frac{1}{2} w^T w + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(w^T x_i + b\right)\right)$$

$$= \frac{1}{2} \sum_{k=1}^{m} w^k w^k + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(\sum_{k=1}^{m} w^k x_i^k + b\right)\right)$$

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i (-y_i) \mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\begin{cases} \dfrac{\partial L}{\partial w^k} = 0, \forall k \\ \dfrac{\partial L}{\partial b} = 0 \end{cases}$$

- If we substitute $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ to Lagrange Multipliers, we have

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b\right)\right)$$

$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i - b \sum_{i=1}^{n} \alpha_i y_i$$

$$= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i$$

Therefore, this is a function of $\alpha_i$ only    Since $\sum_{i=1}^{n} \alpha_i y_i = 0$

# 10.21.4 Support Vector Machine

## Optimization

**Dual Problem**

- The new objective function is in terms of $\alpha_i$ only.
- It is known as the dual problem: if we know w, we know all $\alpha_i$ if we know all $\alpha_i$ we know w.
- The original problem is known as the primal problem.

- The objective function of the dual problem needs to be maximized.

- The dual problem is therefore:

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- This is a **quadratic programming (QP) problem**
  - A global maximum of $\alpha_i$ can always be found
- **w** can be recovered by $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$

## Optimization

**Quadratic Programming Problem**

- Many approaches have been proposed.
- Most are “interior-point” methods.
    - Start with an initial solution that can violate the constraints
    - Improve this solution by optimizing the objective function

- For SVM, sequential minimal optimization (SMO) seems to be the most popular.
    - A QP with two variables is trivial to solve
    - Each iteration of SMO picks a pair of ($\alpha_i$, $\alpha_j$) and solve the QP with these two variables; repeat until convergence

- In practice, we can just regard the QP solver as a “black-box” without bothering how it works.

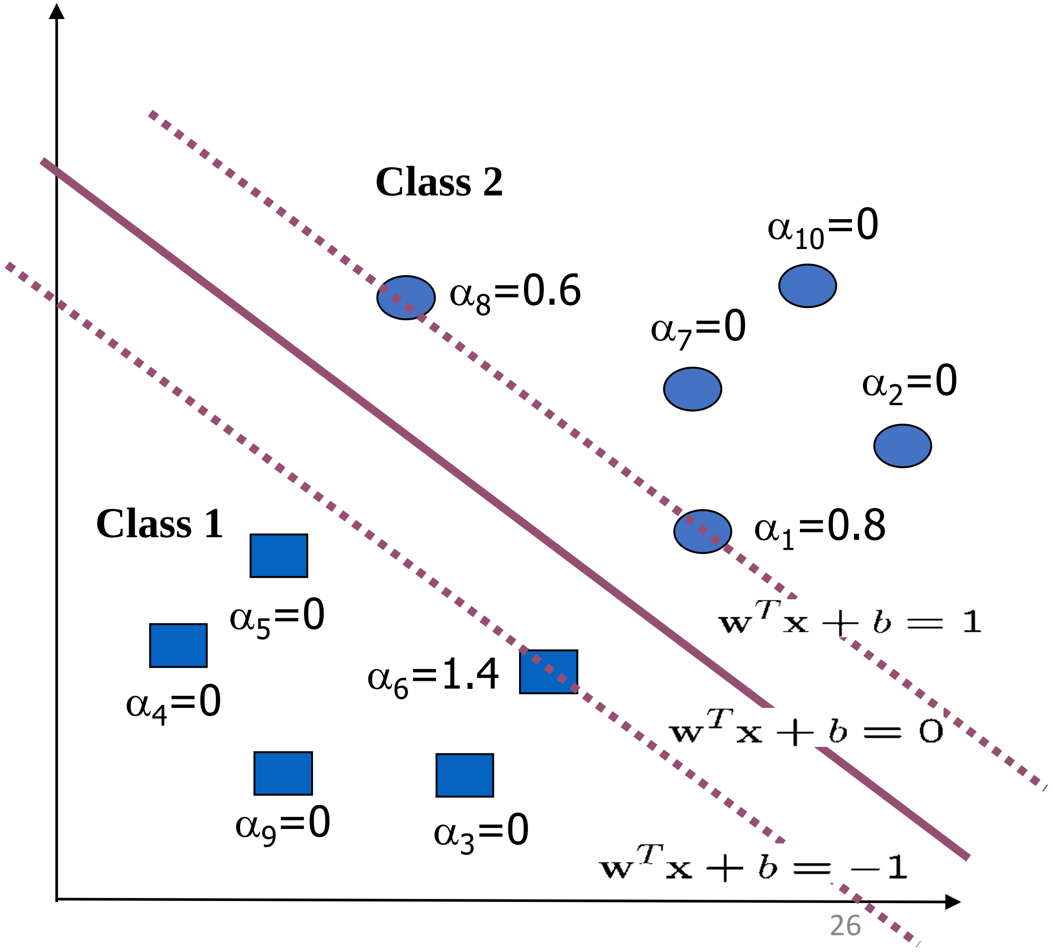
# 10.21.4 Support Vector Machine

## Characteristics of the Solution

- Many of the $\alpha_i$ are zero
- So **w** is a linear combination of a small number of data points

- $\mathbf{x}_i$ with non-zero $\alpha_i$ are called support vectors (SV)
  - The decision boundary is determined only by the SV
  - Let $t_j$ ($j$=1, ..., $s$) be the indices of the $s$ support vectors
  - We can write $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$

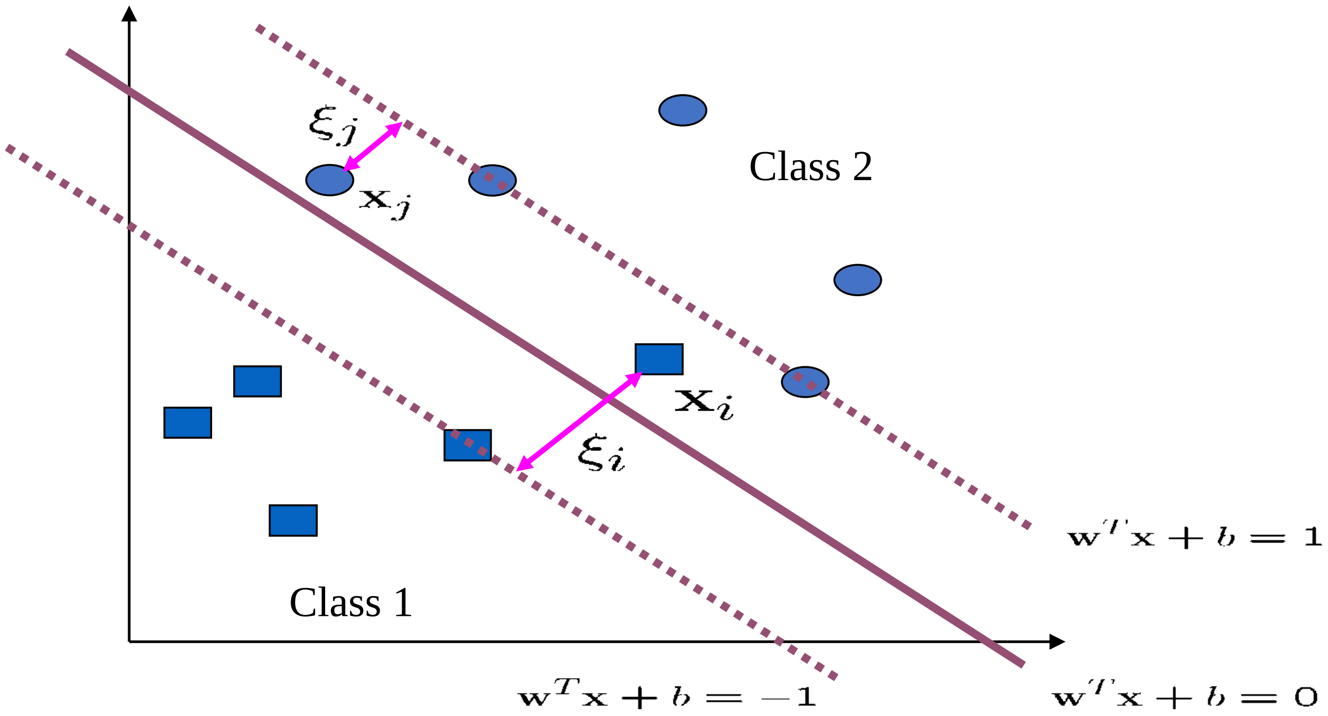
- For testing with a new data **z**
  - Compute $\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$
  - Classify **z** as class 1 if the sum is positive else class 2



# 10.21.4 Support Vector Machine

## Non Linearly Separable Problems

- We allow "error" $\xi_i$ in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}+b$
- $\xi_i$ approximates the number of misclassified samples

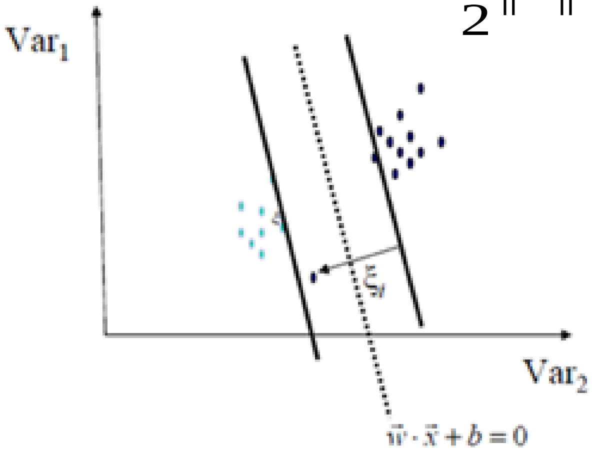

$\xi_j$
$\mathbf{x}_j$
Class 2
$\mathbf{x}_i$
$\xi_i$
$\mathbf{w}^T\mathbf{x}+b=1$
Class 1
$\mathbf{w}^T\mathbf{x}+b=-1$
$\mathbf{w}^T\mathbf{x}+b=0$

# 10.21.4 Support Vector Machine

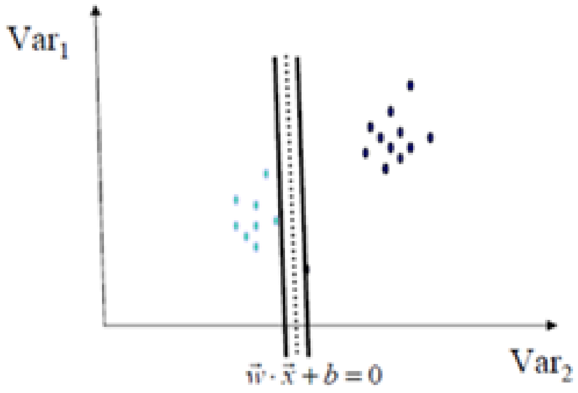
## Non Linearly Separable Problems

**Soft vs. Hard Margin SVM**

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$



Soft Margin SVM



Hard Margin SVM

- The algorithm try to keep ξ null, maximizing the margin.
- The algorithm does not minimize the number of error.
- Instead, it minimizes the sum of distances from the hyperplane.
- When C increases the number of errors tend to lower.
- At limit C → infinite, the solution tend to that given by hard margin formulation, with 0 errors.

# 10.21.4 Support Vector Machine

## Non Linearly Separable Problems

### Soft Margin Hyperplane

- The new conditions become

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

  - $x_i$ are “slack variables” in optimization
  - Note that $x_i$=0 if there is no error for $\mathbf{x}_i$
  - $x_i$ is an upper bound of the number of errors

- We want to minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$

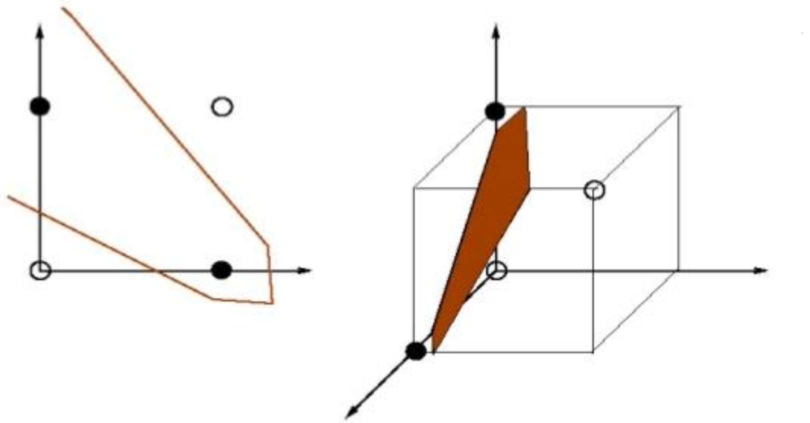
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

- $C$ : tradeoff parameter between error and margin

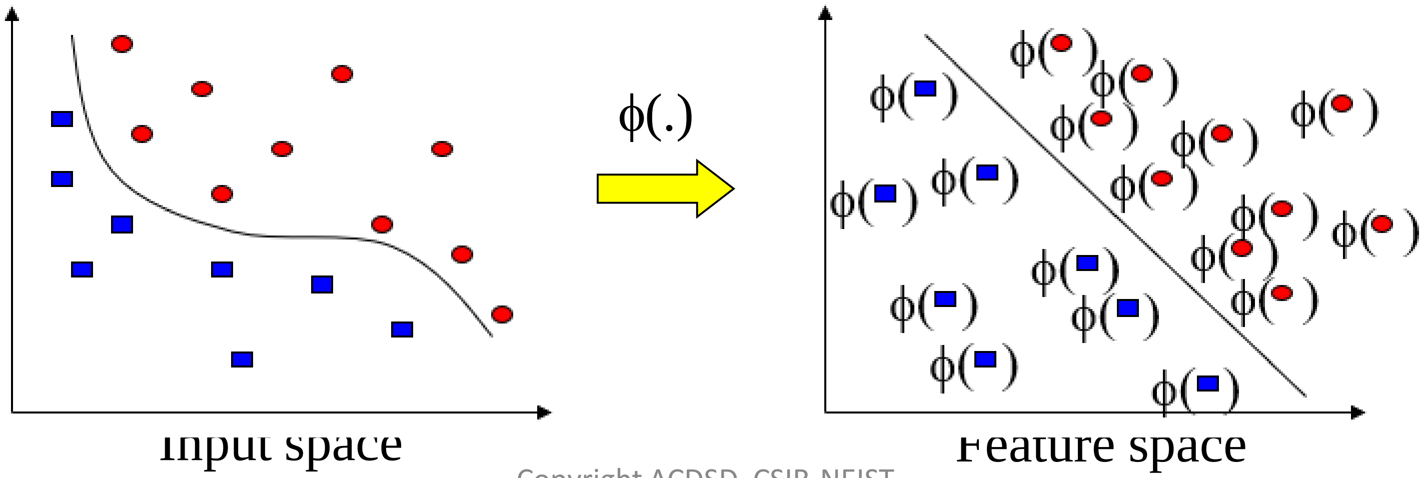
# 10.21.4 Support Vector Machine

## Transformation

- Generalizing large-margin classifier with non linear decision
- Idea is to transform $\mathbf{x}_i$ to a higher dimensional space
  - Input space: the space the point $\mathbf{x}_i$ are located
  - Feature space: the space of $f(\mathbf{x}_i)$ after transformation

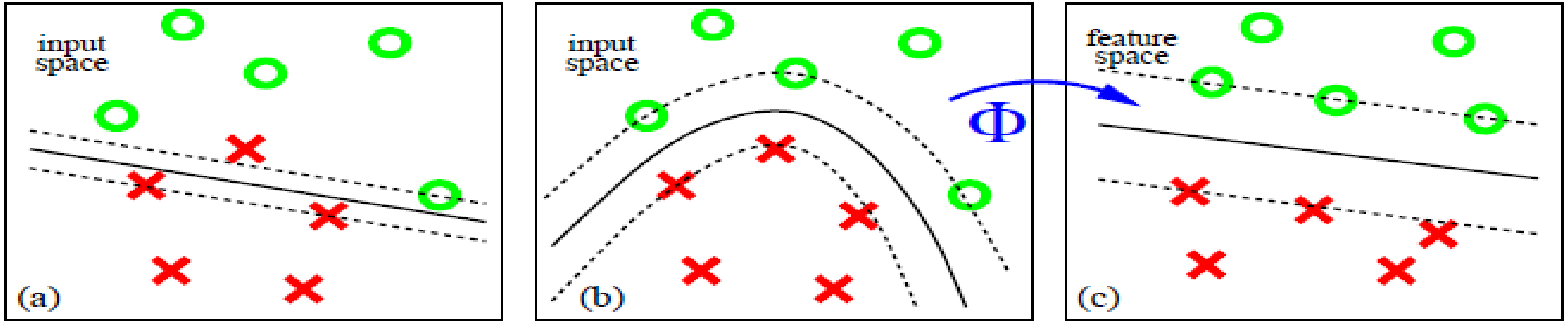


- Why transform?
  - Classification can become easier with a proper transformation.
  - In the XOR problem, for ex. adding a new feature of $x_1x_2$ make the problem linearly separable.

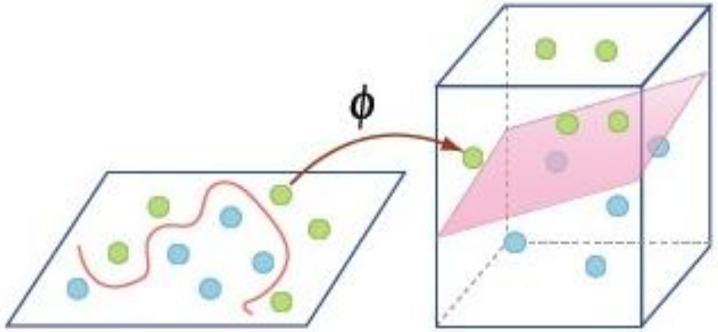


# 10.21.4 Support Vector Machine

## Transformation



- Feature space is of higher dimension than the input space in practice
- Computation in the feature space can be costly because it is high dimensional.
- The kernel trick comes to rescue.



Input space

(Hyperplane in $R^2$ is a line)

Feature space

(Hyperplane in $R^3$ is a plane)

# 10.21.4 Support Vector Machine

## The Kernel Trick

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Recall the SVM optimization problem
- The data points only appear as inner product
- As long as we can calculate the inner product in feature space, we don’t need mapping explicitly.
- Many common geometric operations (angles, distances) can be expressed by inner products.

- Suppose f(.) is given as follows $\phi(\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$
- An inner product in the feature space is $\langle \phi(\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]), \phi(\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right]) \rangle = (1 + x_1y_1 + x_2y_2)^2$
- So, if we define the kernel function as follows, there is no need to carry out f(.) explicitly

$$K(\mathrm{x}, \mathrm{y}) = (1 + x_1y_1 + x_2y_2)^2$$

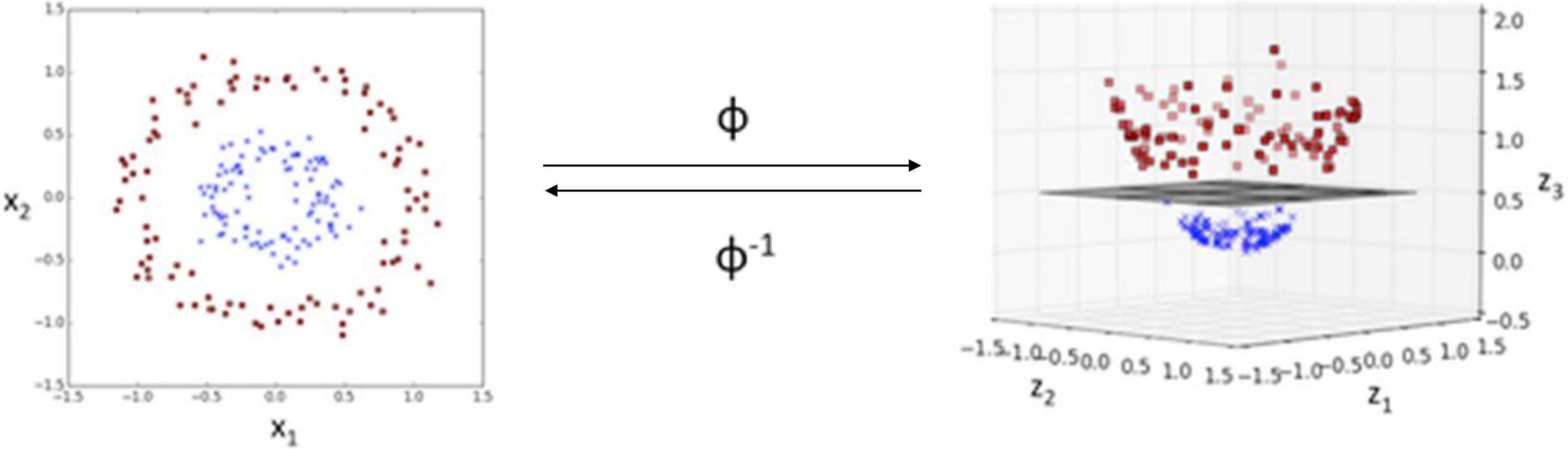
- This use of kernel function to avoid carrying out f(.) explicitly is known as the kernel trick.

- Define the kernel function $K$ by $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

# 10.21.4 Support Vector Machine

## The Kernel Trick

### Kernel Function



- In practical use of SVM, user specifies kernel function; transformation f(.) is not explicitly stated.
- Given a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, the transformation f(.) is given by its Eigen functions.
    - Eigen functions can be difficult to construct explicitly.
    - This is why people only specify kernel function without worrying about exact transformation.
- Another view: kernel function, being an inner product, is a similarity measure between the objects.

# 10.21.4 Support Vector Machine

## SVM Parameter Tuning

- **Kernel:** It specifies the kernel type to be used in the algorithm.
  - It can be ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’
  - The default value is ‘rbf’
- **Degree:** The degree of the polynomial kernel function (‘poly’)

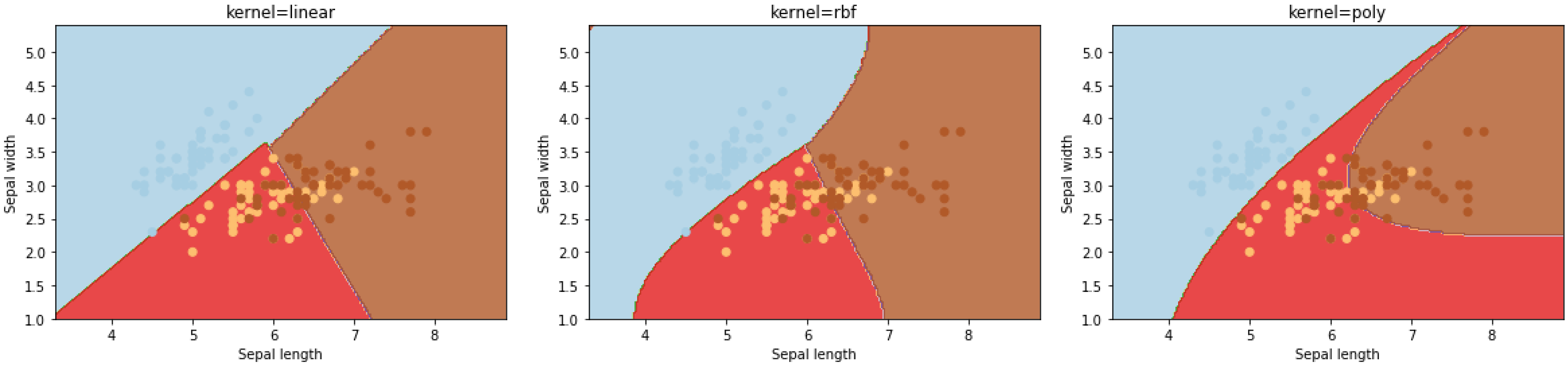
- **C:** It is the regularization parameter of the error term.
- **Gamma:** It is the kernel coefficient for ‘rbf’, ‘poly’, and ‘sigmoid’.

https://cs.stanford.edu/people/karpathy/svmjs/demo/

# 10.21.4 Support Vector Machine

## SVM Parameter Tuning

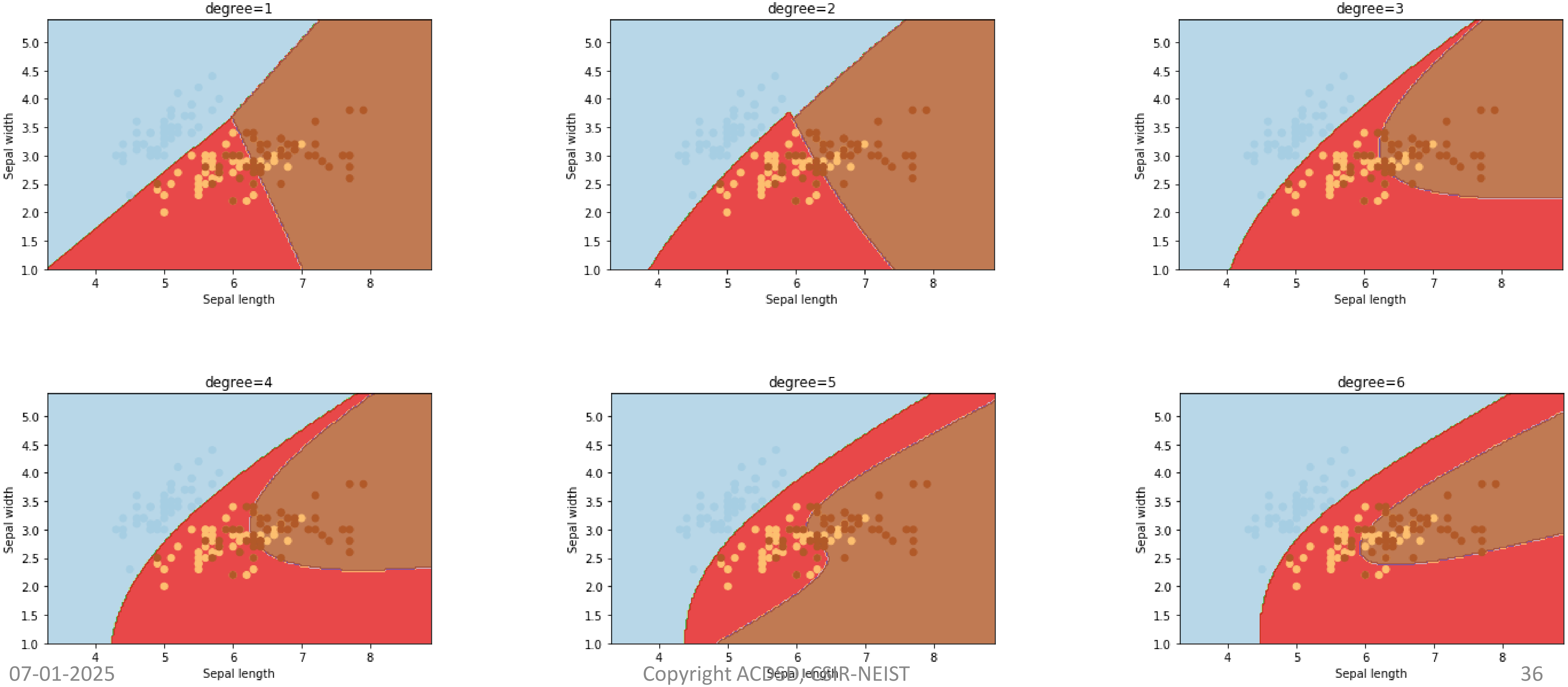
- kernel parameters selects the type of hyperplane used to separate the data.
- ‘linear’ use a linear hyperplane.
- ‘rbf’ and ‘poly’ uses a non linear hyper-plane.



# 10.21.4 Support Vector Machine

## SVM Parameter Tuning

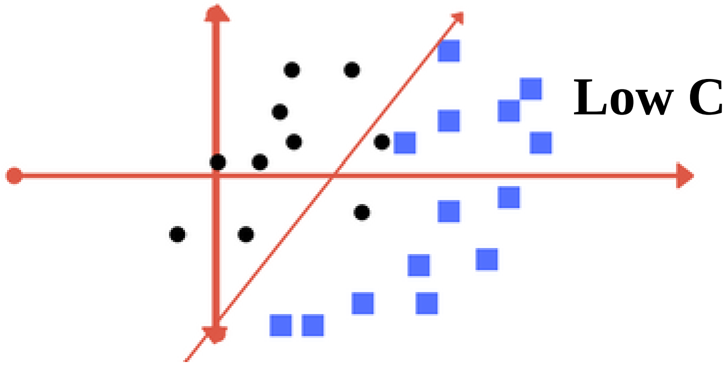
- Degree is a parameter used when kernel is set to ‘poly’.
- It’s basically the degree of the polynomial used to find the hyperplane to split the data.



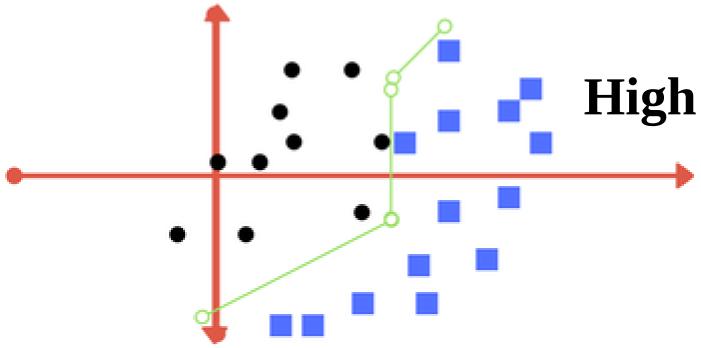
# 10.21.4 Support Vector Machine

## SVM Parameter Tuning

- C Value controls the trade off between smooth boundary and classifying training points correctly.
- Increasing C values may lead to getting more training points correctly.
- With that overfitting the training data.



**Low C values**



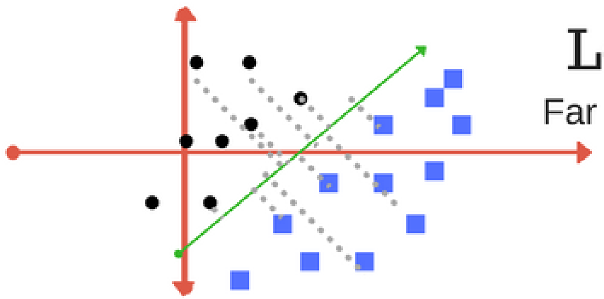
**High C values**

- Finding out smooth decision boundary vs correct (with overfitting) is part of artistry of ML.
- So try different values of c to get the perfectly balanced curve and avoid over fitting.

# 10.21.4 Support Vector Machine

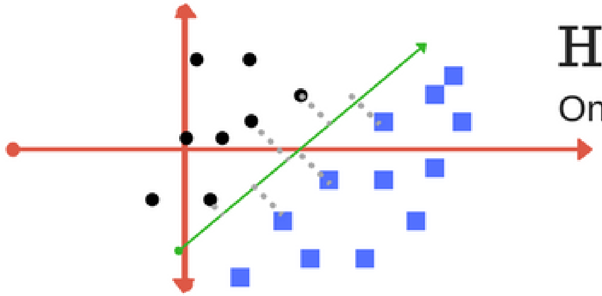
## SVM Parameter Tuning

- Gamma defines how far data points are considered in calculation of separation line.



Low Gamma

Far away points are also considered.



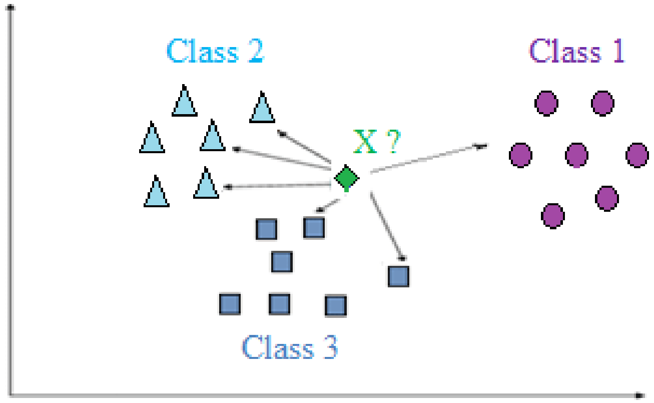
High Gamma

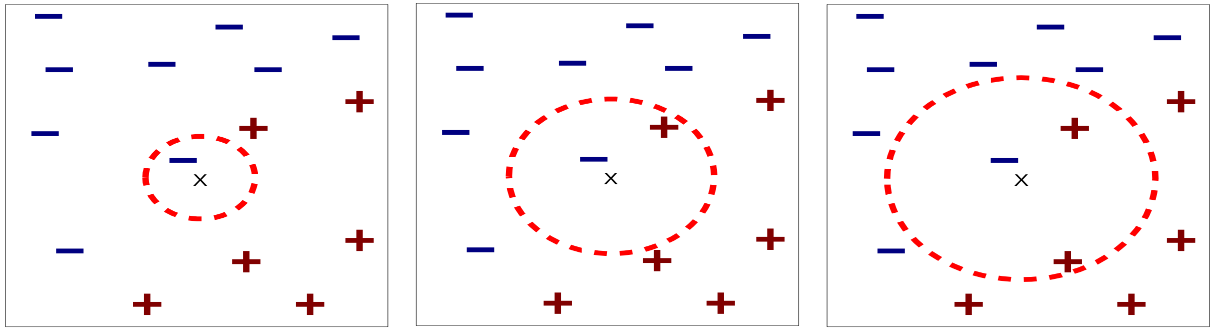
Only nearby points are considered.

- For low gamma even the far away points get considerable weight and it results more linear curve.
- For high gamma the closer points get more weight and it results in a wiggly curve.

# 10.21.5 KNN

- Explains a categorical value using the majority votes of (k) nearest neighbors.
- Select the tuning parameter k based on data and experiment.





(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

**Algorithm**

1. Load the data and initialize the value of k
2. For getting the predicted class, iterate from 1 to total number of training data points
   a) Calculate distance (metrics ex.- Euclidean) between test data and each row of training data.
   b) Sort the calculated distances in ascending order based on distance values
   c) Get top k rows from the sorted array
   d) Get the most frequent class of these rows
   e) Return the predicted class

## Choosing K

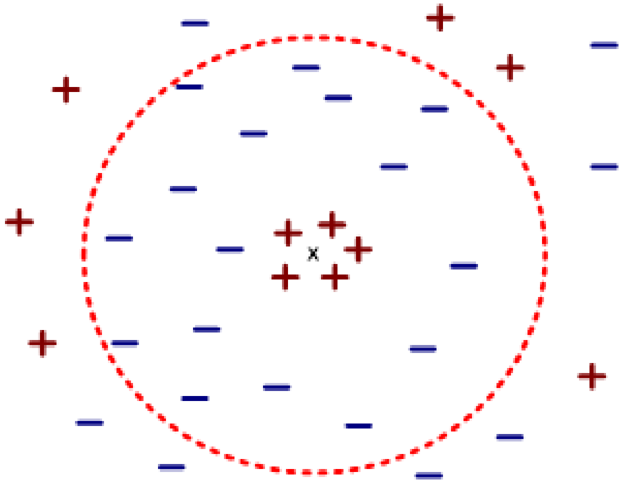
Choosing the value of k:

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes

**Rule of thumb:**

K = sqrt(N) where N: number of training points

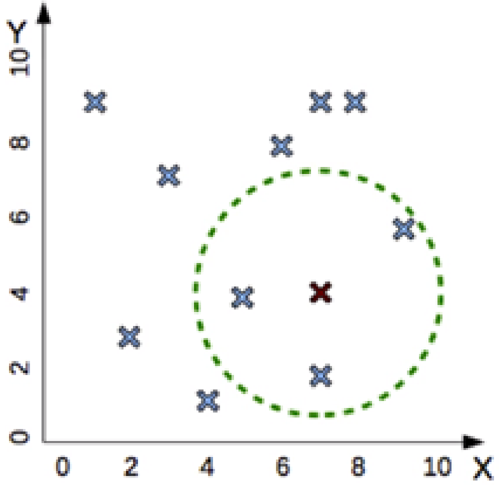
In practice, *k* is usually chosen to be odd, so as to avoid ties.



## Laziness

- **Training:** O(1)
- **Testing:** O(nd)

Training set: n sample
Testing instance: 1 sample
Nearest Neighbor: Compare One by one to each training instance
Complexity: N comparisons and each comparisons takes d operations

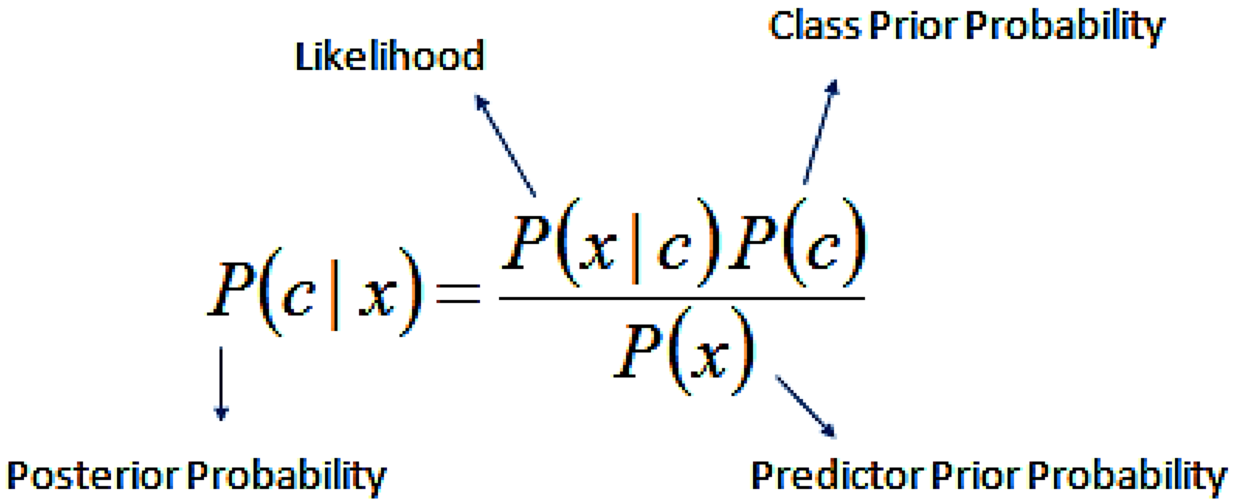


**Making Fast**

- **Reduce d:** dimensionality reduction
- **Reduce n:** do not compare to all training examples

# 10.21.6 Naïve Bayes

## Bayes' Theorem



Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)\,P(c)}{P(x)}$$

Posterior Probability

Predictor Prior Probability

$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

# 10.21.6 Naïve Bayes

- Probabilistic Classifier based on Baye’s theorem with strong (Naïve) independence assumptions among the features.

**Why Naïve?**

- It assumes the features that go into the model is independent of each other.
- That is changing the value of one feature, does not directly influence any other features.
- Because of independence nature it makes assumptions that may or may not turn out to be correct.

**Multivariant Bayesian Classifier**

- Compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, An|C)\, P(C)}{P(A_1, A_2, \ldots, An)}$$

- Choose value of C that maximizes $P(C \mid A_1, A_2, \ldots, A_n)$
- Equivalent to choosing value of C that maximizes $P(A_1, A_2, \ldots, A_n|C)\, P(C)$

**Note:** Other names "Simple Baye’s" and "Independence Baye’s"

# 10.21.6 Naïve Bayes

## Variants

Based on the probability distribution adopted

**Bernoulli:** Assumes that all features are binary such that they take only two values.

**Multinomial:** Assumes that features element represents frequency i.e. the number of times it appears.

**Gaussian:** Based on a continuous distribution and it’s suitable for more generic classification tasks.

# 10.21.7 Examples

## Logistic Regression.

**Example:** A patient is admitted with abdominal sepsis (blood poisoning). The case is severe enough to warrant surgery. The patient is wheeled to the operation theatre. Let us speculate what will happen after the surgery. Let

$$\boxed{\begin{aligned} Y &= 1 \text{ if death follows surgery,} \\ &= 0 \text{ if the patient survives} \end{aligned}}$$

The variables $X_1$, $X_2$, $X_3$, and $X_5$ are categorical covariates and $X_4$ is a continuous covariate. The categorical variables are binary with only two possible values. It is felt that the outcome Y depends on these covariates.

| Response variable | Covariates, predictors or independent variables |
|---|---|
| Y | $X_1$, $X_2$, $X_3$, $X_4$, $X_5$ |

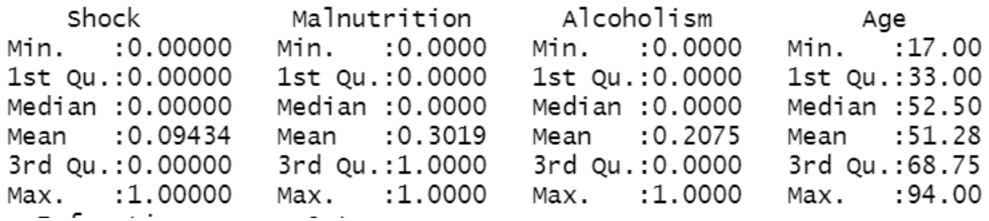
Basic question: Which of the predictors have an impact on the response?

# 10.21.7 Examples

```
dim(Sepsis)    # dimension of the data set
[1] 106     6
```

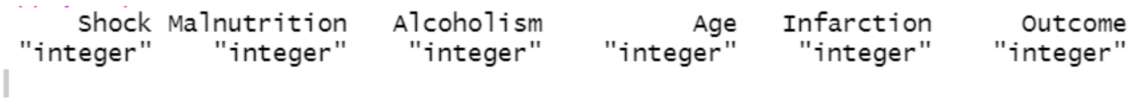
```
head(Sepsis)    # Display the head
                Shock    Malnutrition    Alcoholism    Age    Infarction    Death
  1               0               0             0     56         0            0
  2               0               0             0     80         0            0
  3               0               0             0     61         0            0
```

```
Summary(Sepsis)   # summary statistics of each variable
```



```
     Shock              Malnutrition       Alcoholism           Age       
 Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :17.00  
 1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:33.00  
 Median :0.00000   Median :0.0000   Median :0.0000   Median :52.50  
 Mean   :0.09434   Mean   :0.3019   Mean   :0.2075   Mean   :51.28  
 3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:68.75  
 Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :94.00  
```

```
apply(Sepsis, 2, class)    # class of each variable
```



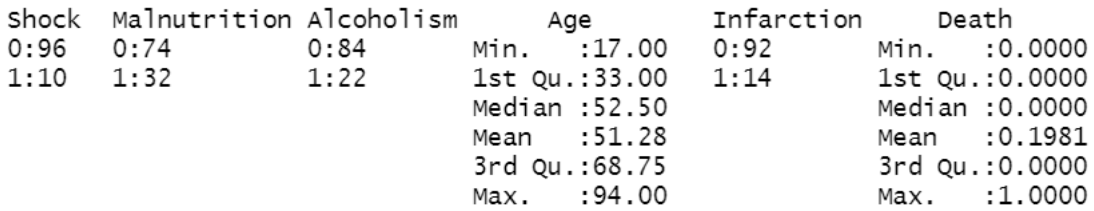
```
    Shock Malnutrition  Alcoholism         Age  Infarction     Outcome 
"integer"   "integer"   "integer"   "integer"   "integer"   "integer" 
```

# 10.21.7 Examples

Let us convert each binary variable into a factor (categorical variable with direction). Create another copy of Sepsis

```
Sepsis1 <- Sepsis
 > Sepsis1$Shock <- as.factor(Sepsis1$Shock)
> Sepsis1$Malnutrition <- as.factor(Sepsis1$Malnutrition)
> Sepsis1$Alcoholism <- as.factor(Sepsis1$Alcoholism)
> Sepsis1$Infarction <- as.factor(Sepsis1$Infarction)
> Sepsis1$Death <- as.factor(Sepsis1$Death
```

```
summary(Sepsis1)  # summary statistics
```



```
Shock   Malnutrition Alcoholism      Age          Infarction      Death       
0:96    0:74         0:84       Min.   :17.00   0:92        Min.   :0.0000  
1:10    1:32         1:22       1st Qu.:33.00   1:14        1st Qu.:0.0000  
                                Median :52.50               Median :0.0000  
                                Mean   :51.28               Mean   :0.1981  
                                3rd Qu.:68.75               3rd Qu.:0.0000  
                                Max.   :94.00               Max.   :1.0000  
```

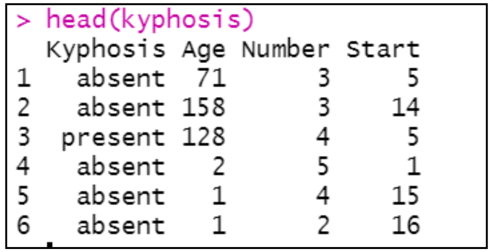
# 10.21.7 Examples

## Classification Trees

Kyphosis is a misalignment of vertebrate. Surgery is helpful to align the problematic vertebrate. Different parameters such as age, number of the topmost vertebrae surgery was done, outcome of surgery was taken into consideration.

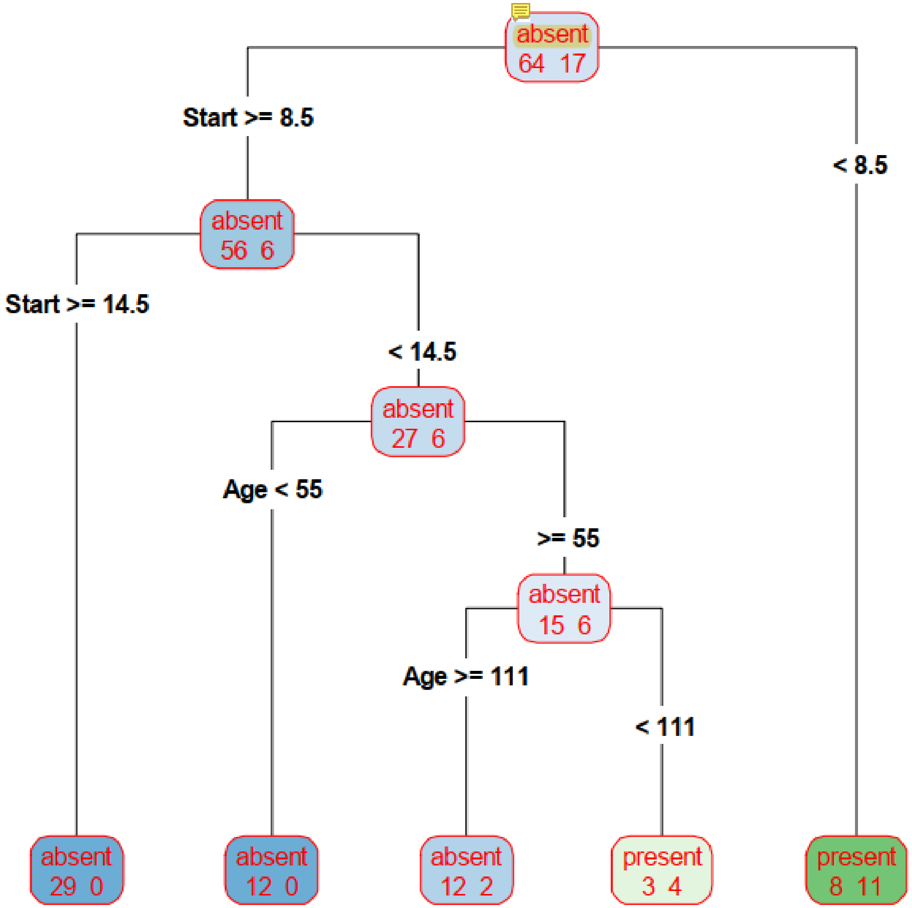
# 10.21.7 Examples

```
library(rpart)     #Activate the rpart library contains different dataset
library("rpart.plot")    #Activate the plot function
data(kyphosis)    # access kyphosis data set
dim(kyphosis)
[1] 81    4
head(kyphosis)
```



```
> head(kyphosis)
  Kyphosis Age Number Start
1   absent  71      3     5
2   absent 158      3    14
3  present 128      4     5
4   absent   2      5     1
5   absent   1      4    15
6   absent   1      2    16
```

```
MB <- rpart(Kyphosis ~ Age+Number+Start,data=kyphosis)
rpart.plot(MB, type = 4, extra = 1, digits = 3, col = “red”)
```



Description of the tree:

1. The root node has 81 subjects, for 64 of them kyphosis is absent and 17 present.
2. All those subjects with Start ≥ 8.5 go into the left node. Total number of subjects in the left node is 62, 56 of them have kyphosis absent.
3. All those subjects with Start < 8.5 go into the right node. Total number of subjects in the right node is 19, 8 of them have kyphosis absent.
4. This node is a terminal node. No further split is envisaged because the total number of observations is 19 ≤ 20. The command stops splitting a node if the size of the node is 20 or less (default). This is a pruning strategy. This node is declared ‘present’ as per the ‘majority rule’ paradigm.

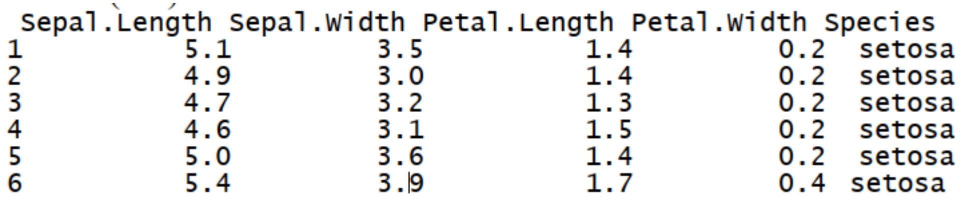
Misclassification rate = re-substitution error = 100*(8 + 0 + 2 + 3)/81 = 16%

# 10.21.7 Examples

## Random Forest

**Objective**: The response variable is ‘Species.’ The flower could be any one of the three types. I have a flower. I do not know to which species the flower belongs. I know its petal length, petal width, sepal length, and sepal width. Guess the species. This is a classification problem or a pattern recognition problem.

```
>data(iris)          # Download the data.
> head(iris)          # Understand the data. The top six rows of the data …
```



```
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4 setosa
```

```
➢dim(iris)
[1] 150 5
> table(iris$Species)
setosa     versicolor    virginica
50                50              50
```

Install and activate the package ‘randomForest.’

```
 install.packages(“randomForest”)
Library(randomForest)
MB <- randomForest(Species ~ ., data = iris, importance = T)
```

# 10.21.7 Examples

print(MB)      # print the output

```
Call:
 randomForest(formula = Species ~ ., data = iris, importance = T,
proximity = T)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2
(Here the number of covariates is 4 and m = 2.)
        OOB estimate of error rate: 4%
Confusion matrix:
              setosa versicolor virginica
class.error
setosa          50          0         0         0.00
versicolor       0         47         3         0.06
virginica        0          3        47         0.06
```

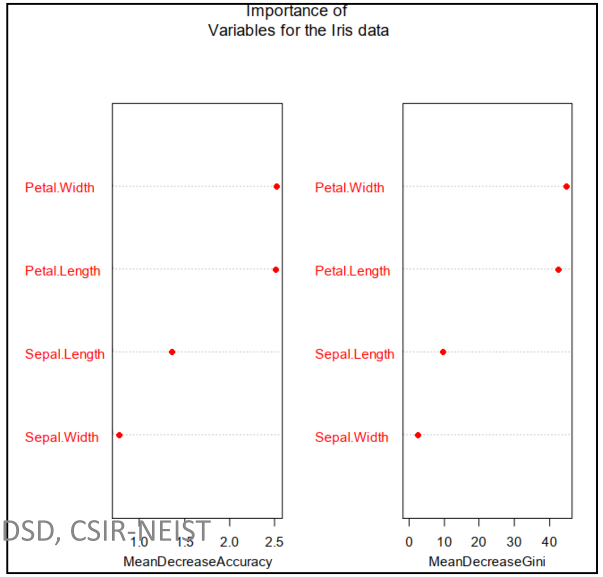
One can obtain a graph of the importance of the predictors in the classification problem.

> varImpPlot(MB, pch = 16, col = "red", n.var = 4, sort = T, main = "Importance of Variables for the Iris data")

***Understand the output:***

1. R says that this is a classification problem.
2. R produces a forest of 500 trees.
3. There are four predictors. At every node, it chose 2 predictors at random for splitting the node.
4. The given data were poured into the root node for classification. The acronym ‘OOB’ stands for ‘out-of-bag.’ The OOB (error) estimate is 4%.

The confusion matrix spells out where errors of misclassification occurred

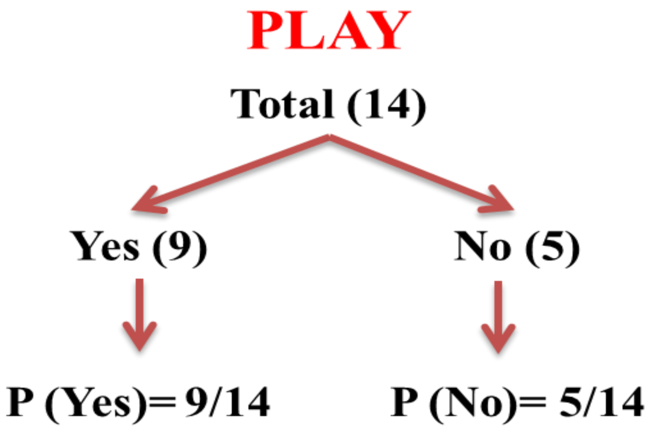


# 10.21.7 Examples

**Find the probability to play on 15th day where conditions are temp = cool, humidity = high wind = strong and outlook = sunny**

| Day | Outlook | Temp | Humidity | Wind | Play |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Sunny | Hot | High | Strong | No |
| 3 | Overcast | Hot | High | Weak | Yes |
| 4 | Rain | Mild | High | Weak | Yes |
| 5 | Rain | Cool | Normal | Weak | Yes |
| 6 | Rain | Cool | Normal | Strong | No |
| 7 | Overcast | Cool | Normal | Strong | Yes |
| 8 | Sunny | Mild | High | Weak | No |
| 9 | Sunny | Cool | Normal | Weak | Yes |
| 10 | Rain | Mild | Normal | Weak | Yes |
| 11 | Sunny | Mild | Normal | Strong | Yes |
| 12 | Overcast | Mild | High | Strong | Yes |
| 13 | Overcast | Hot | Normal | Weak | Yes |
| 14 | Rain | Mild | High | Strong | No |

# 10.2.1.7 Examples



PLAY

Total (14)

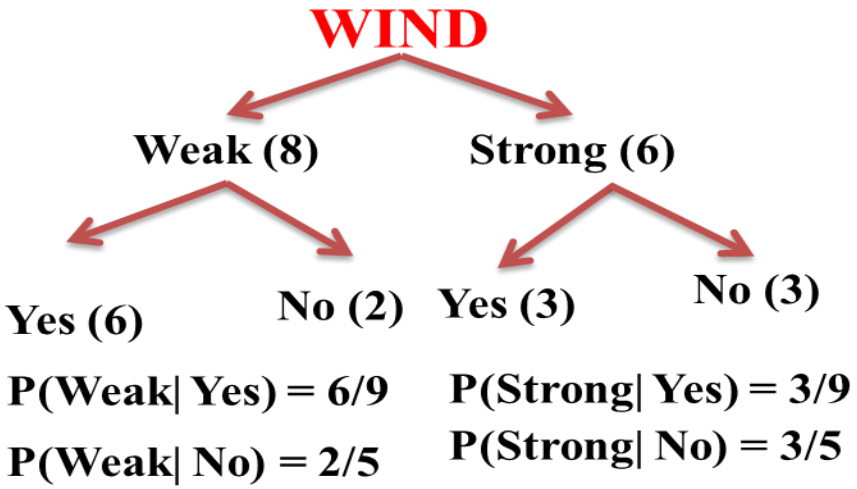
Yes (9)

No (5)

P (Yes)= 9/14

P (No)= 5/14

Expansion from right to left



WIND

Weak (8)

Strong (6)

Yes (6)

No (2)

Yes (3)

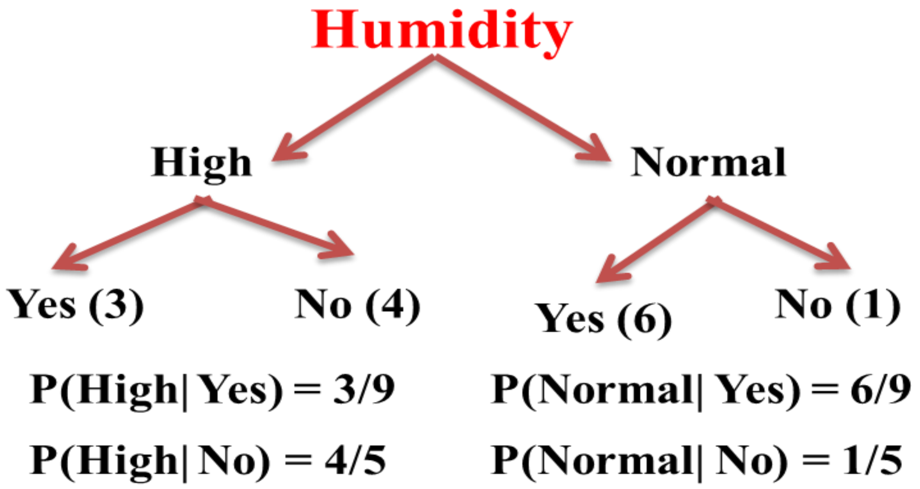
No (3)

P(Weak| Yes) = 6/9

P(Weak| No) = 2/5

P(Strong| Yes) = 3/9

P(Strong| No) = 3/5



Humidity

High

Normal

Yes (3)

No (4)

Yes (6)

No (1)

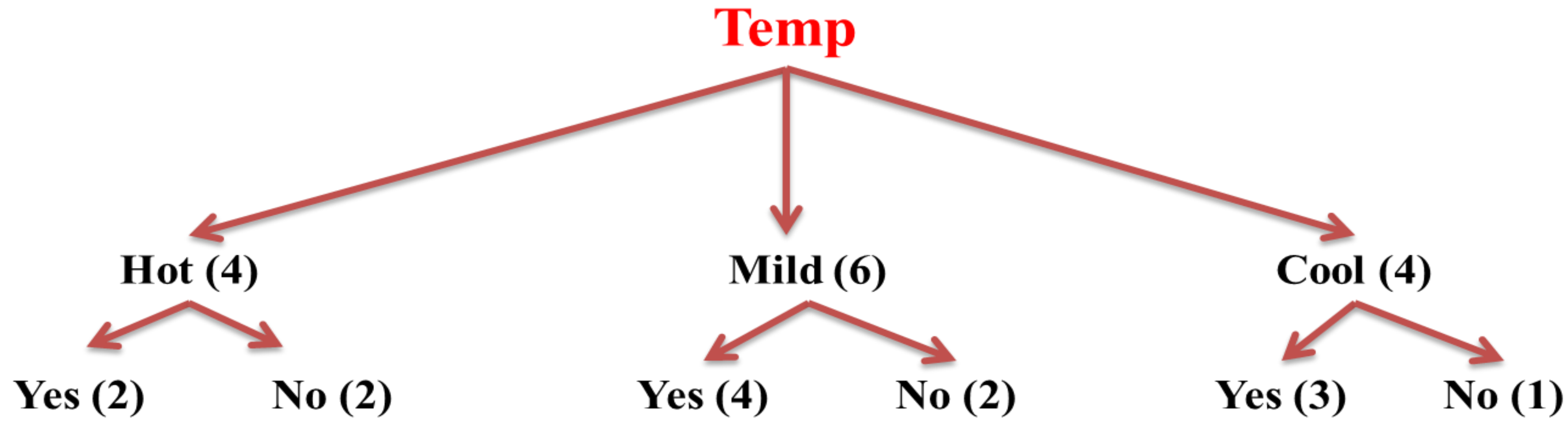
P(High| Yes) = 3/9

P(High| No) = 4/5

P(Normal| Yes) = 6/9

P(Normal| No) = 1/5

# 10.21.7 Examples



P(Hot| Yes) = 2/9

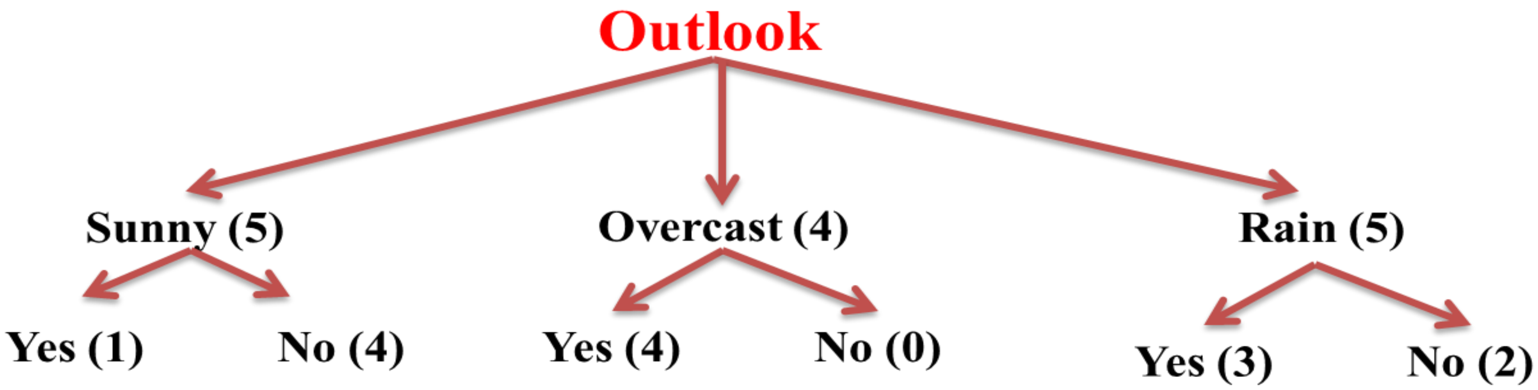
P(Hot| No) = 2/5

P(Mild| Yes) = 4/9

P(Mild| No) = 2/5

P(Cool| Yes) = 3/9

P(Cool| No) = 1/5



P(Sunny| Yes) = 1/9

P(Sunny| No) = 4/5

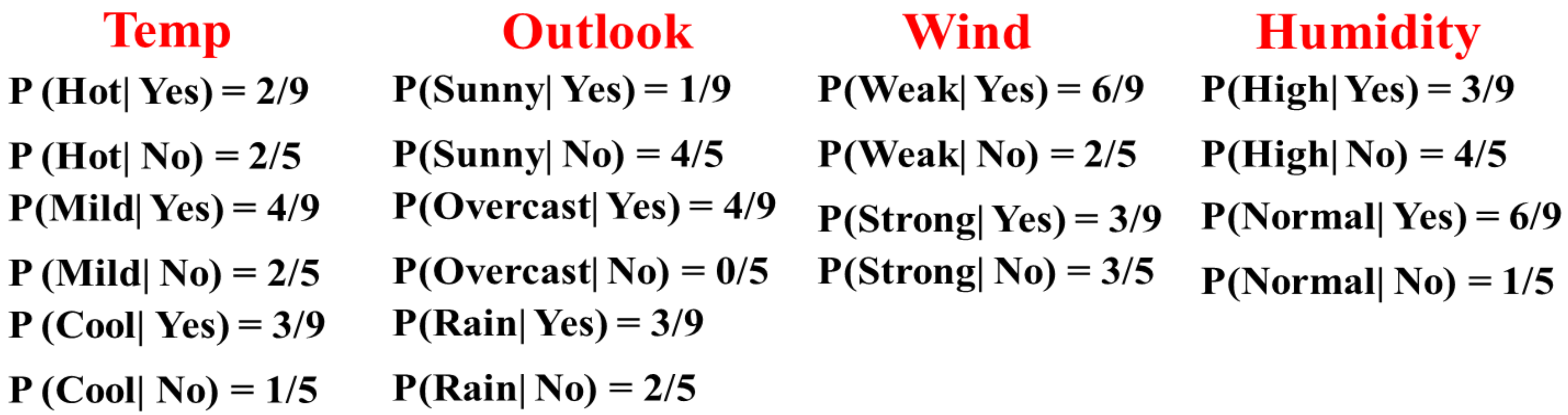
P(Overcast| Yes) = 4/9

P(Overcast| No) = 0/5

P(Rain| Yes) = 3/9

P(Rain| No) = 2/5

# 10.21.7 Examples



| Temp | Outlook | Wind | Humidity |
|---|---|---|---|
| P (Hot\| Yes) = 2/9 | P(Sunny\| Yes) = 1/9 | P(Weak\| Yes) = 6/9 | P(High\| Yes) = 3/9 |
| P (Hot\| No) = 2/5 | P(Sunny\| No) = 4/5 | P(Weak\| No) = 2/5 | P(High\| No) = 4/5 |
| P(Mild\| Yes) = 4/9 | P(Overcast\| Yes) = 4/9 | P(Strong\| Yes) = 3/9 | P(Normal\| Yes) = 6/9 |
| P (Mild\| No) = 2/5 | P(Overcast\| No) = 0/5 | P(Strong\| No) = 3/5 | P(Normal\| No) = 1/5 |
| P (Cool\| Yes) = 3/9 | P(Rain\| Yes) = 3/9 | | |
| P (Cool\| No) = 1/5 | P(Rain\| No) = 2/5 | | |

Let X = {Sunny, Cool, High, Strong}

P(X|Yes) = P(Yes)*P(Sunny|Yes)*P(Cool|Yes)*P(High|Yes)*P(Strong|Yes)

= 9/14*1/9*3/9*3/9*3/9

= 0.0053

P(X|No)= P(No)*P(Sunny|No)*P(Cool|No)*P(High|No)*P(Strong|No)

= 5/14*4/5*1/5*4/5*3/5

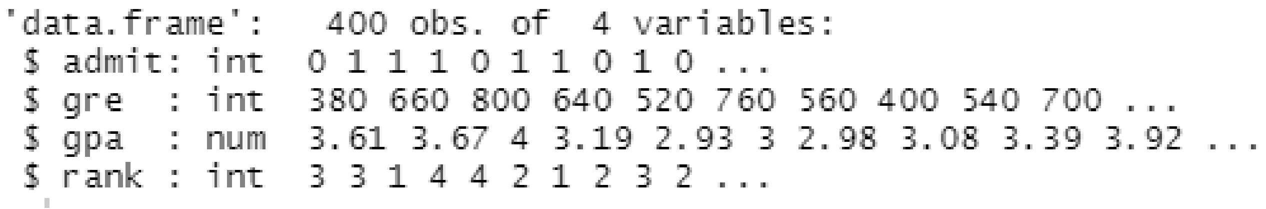
=0.0206

P(X|No) > P(X|Yes)

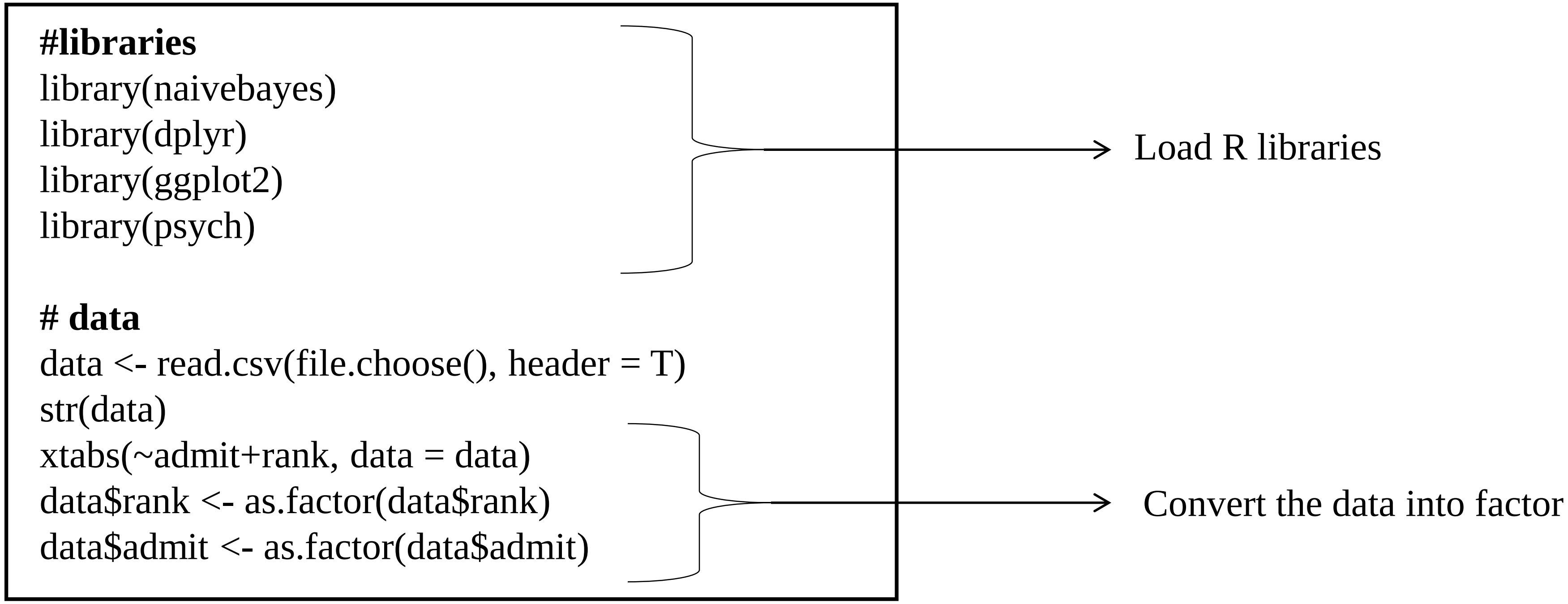
# 10.21.7 Examples

## Classification with Naïve Bayes

A dataset containing 400 observations with 4 variables GRE, GPA, rank of school where a candidate has studied and admission status. Using this dataset design a classification model with Naïve Bayes classifier in R.



```
'data.frame':   400 obs. of  4 variables:
 $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```



```
#libraries
library(naivebayes)
library(dplyr)
library(ggplot2)
library(psych)

# data
data <- read.csv(file.choose(), header = T)
str(data)
xtabs(~admit+rank, data = data)
data$rank <- as.factor(data$rank)
data$admit <- as.factor(data$admit)
```

Load R libraries

Convert the data into factor

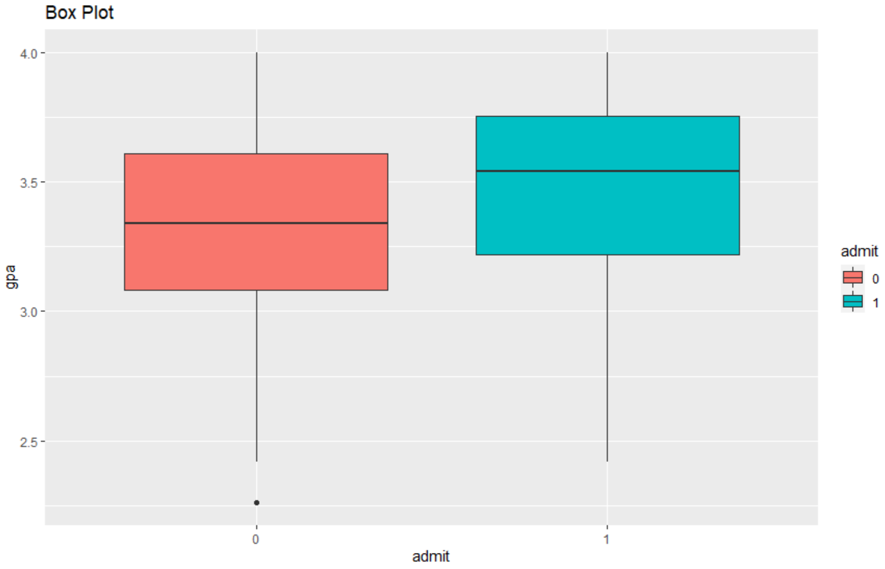
# 10.21.7 Examples

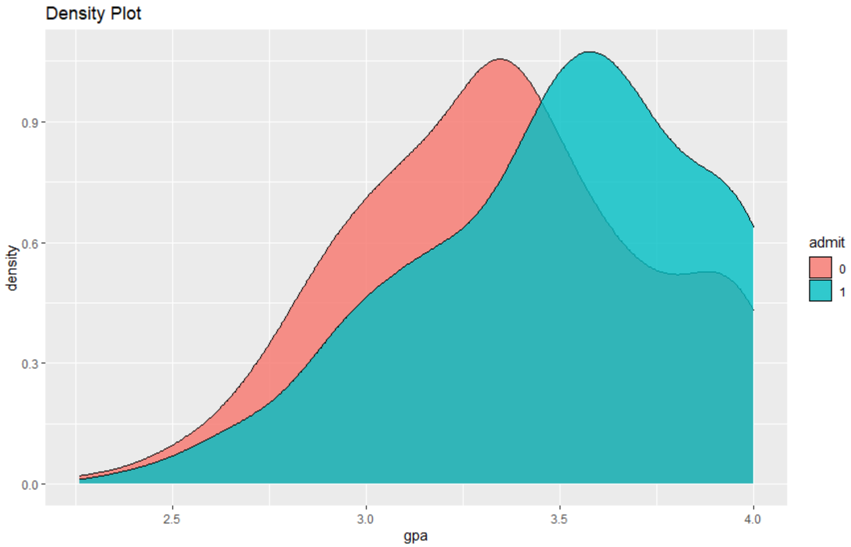
```
#visualization
pairs.panels(data[-1])
data %>%
  ggplot(aes(x=admit, y=gpa, fill = admit)) +
  geom_boxplot() +
  ggtitle("Box Plot")

data %>% ggplot(aes(x=gpa, fill = admit)) +
  geom_density(alpha=0.8, color= 'black') +
  ggtitle("Density Plot")


# Data Partition
set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.8,
0.2))
train <- data[ind == 1,]
test <- data[ind == 2,]
```

Box plot on the gpa data

Density plot on the gpa data

Partitioning data for training and test

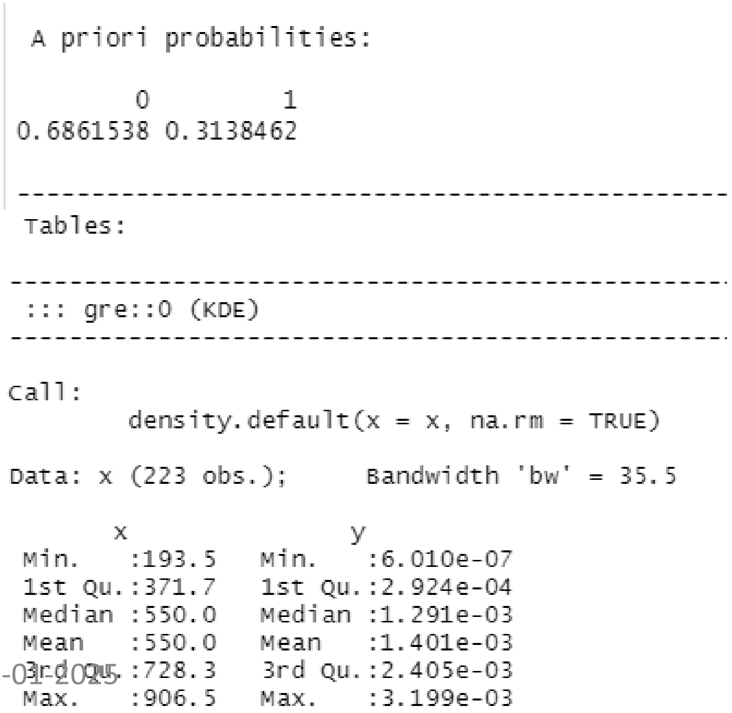




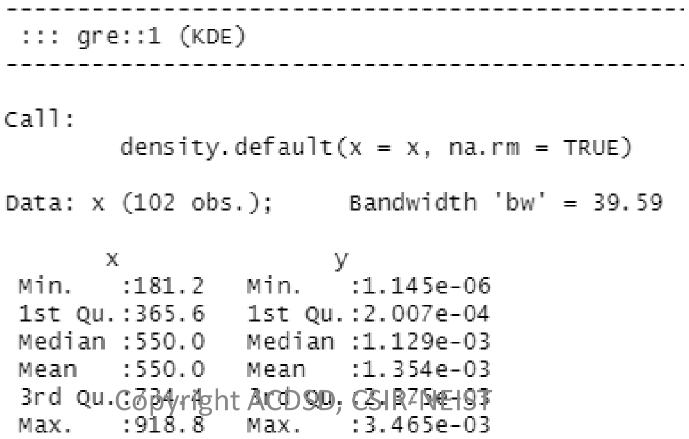
# 10.21.7 Examples

```
# Naive Bayes Model
model <- naive_bayes(admit ~ ., data = train, ,
usekernel = T)
model

train %>%
  filter(admit == "0") %>%
  summarise(mean(gre), sd(gre))
```

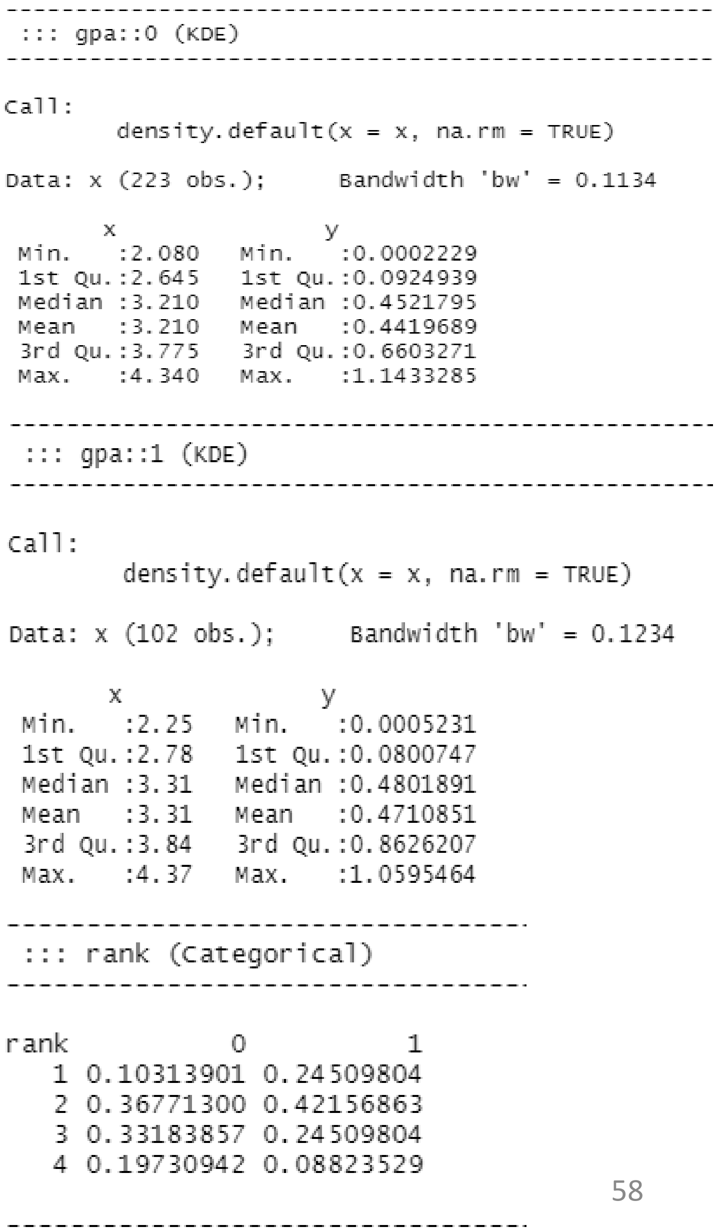
Naïve Bayes model for training



```
 A priori probabilities:

        0         1 
0.6861538 0.3138462 

-------------------------------------------------
 Tables: 

-------------------------------------------------
 ::: gre::0 (KDE)
-------------------------------------------------

Call:
	density.default(x = x, na.rm = TRUE)

Data: x (223 obs.);	Bandwidth 'bw' = 35.5

       x               y            
 Min.   :193.5   Min.   :6.010e-07  
 1st Qu.:371.7   1st Qu.:2.924e-04  
 Median :550.0   Median :1.291e-03  
 Mean   :550.0   Mean   :1.401e-03  
 3rd Qu.:728.3   3rd Qu.:2.405e-03  
 Max.   :906.5   Max.   :3.199e-03  
```



```
-------------------------------------------------
 ::: gre::1 (KDE)
-------------------------------------------------

Call:
	density.default(x = x, na.rm = TRUE)

Data: x (102 obs.);	Bandwidth 'bw' = 39.59

       x               y            
 Min.   :181.2   Min.   :1.145e-06  
 1st Qu.:365.6   1st Qu.:2.007e-04  
 Median :550.0   Median :1.129e-03  
 Mean   :550.0   Mean   :1.354e-03  
 3rd Qu.:734.4   3rd Qu.:2.375e-03  
 Max.   :918.8   Max.   :3.465e-03  
```



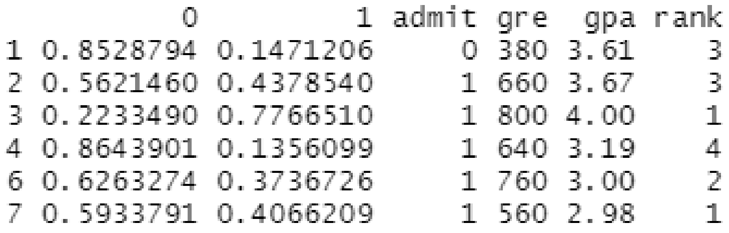
```
-------------------------------------------------
 ::: gpa::0 (KDE)
-------------------------------------------------

Call:
	density.default(x = x, na.rm = TRUE)

Data: x (223 obs.);	Bandwidth 'bw' = 0.1134

       x               y            
 Min.   :2.080   Min.   :0.0002229  
 1st Qu.:2.645   1st Qu.:0.0924939  
 Median :3.210   Median :0.4521795  
 Mean   :3.210   Mean   :0.4419689  
 3rd Qu.:3.775   3rd Qu.:0.6603271  
 Max.   :4.340   Max.   :1.1433285  

-------------------------------------------------
 ::: gpa::1 (KDE)
-------------------------------------------------

Call:
	density.default(x = x, na.rm = TRUE)

Data: x (102 obs.);	Bandwidth 'bw' = 0.1234

       x              y            
 Min.   :2.25   Min.   :0.0005231  
 1st Qu.:2.78   1st Qu.:0.0800747  
 Median :3.31   Median :0.4801891  
 Mean   :3.31   Mean   :0.4710851  
 3rd Qu.:3.84   3rd Qu.:0.8626207  
 Max.   :4.37   Max.   :1.0595464  

------------------------------------
 ::: rank (Categorical) 
------------------------------------

rank          0          1
   1 0.10313901 0.24509804
   2 0.36771300 0.42156863
   3 0.33183857 0.24509804
   4 0.19730942 0.08823529

------------------------------------
```

# 10.21.7 Examples

```
# Prediction
p <- predict(model, train, type = 'prob')
head(cbind(p, train))

# Confusion Matrix - train data
p1 <- predict(model, train)
(tab1 <- table(p1, train$admit))
1 - sum(diag(tab1)) / sum(tab1)

# Confusion Matrix - test data
p2 <- predict(model, test)
(tab2 <- table(p2, test$admit))
1 - sum(diag(tab2)) / sum(tab2)
```

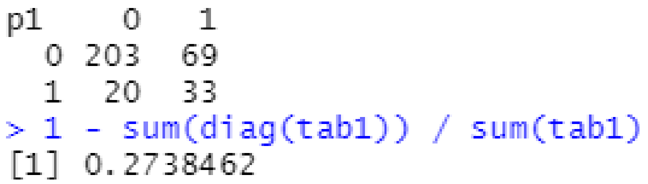
Prediction

Confusion matrix of train and test data



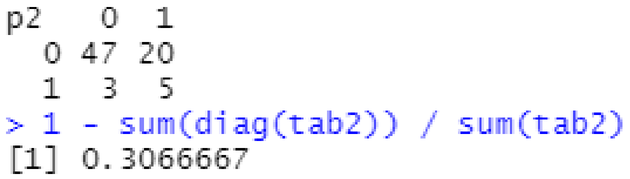
```
          0         1 admit gre  gpa rank
1 0.8528794 0.1471206     0 380 3.61    3
2 0.5621460 0.4378540     1 660 3.67    3
3 0.2233490 0.7766510     1 800 4.00    1
4 0.8643901 0.1356099     1 640 3.19    4
6 0.6263274 0.3736726     1 760 3.00    2
7 0.5933791 0.4066209     1 560 2.98    1
```

Confusion matrix of train data



```
p1     0   1
  0 203  69
  1  20  33
> 1 - sum(diag(tab1)) / sum(tab1)
[1] 0.2738462
```

Confusion matrix of test data



```
p2   0  1
  0 47 20
  1  3  5
> 1 - sum(diag(tab2)) / sum(tab2)
[1] 0.3066667
```

# 10.2.2 Regression

**Regression can of many types based on –**

- Parameters
  - Linear Regression (parameters are linear)
  - Non-linear (parameters are non linear)
- Number of independent Variables
  - Simple
  - Multiple
  - Polynomial

| Linear | Non Linear | Others |
|---|---|---|
| Simple | Simple | SVR |
| Multiple | Multiple | Regression Tree |
| Polynomial (special case) | | |

# 10.2.2 Regression

- Regression is a statistical technique used to establish a relationship model between two variables.
- Originally invented by Francis Galton to study the relationships between parents and children.
- One of these variable is called predictor variable whose value is gathered through experiments.
- The other variable is called response variable whose value is derived from the predictor variable.

*Dependent/Measured/Outcome/Experimental/Response - what we are trying to predict*
*Independent/Covariate/Feature/Explanatory/Predictor - what we are using to predict*

- Gather a sample (input) of response and corresponding predictor variable by experimentation.
- Find out a relationship between the response variables and corresponding output variable.
- Based on the fitted relationship/line (model) we predict the unknown predictor.

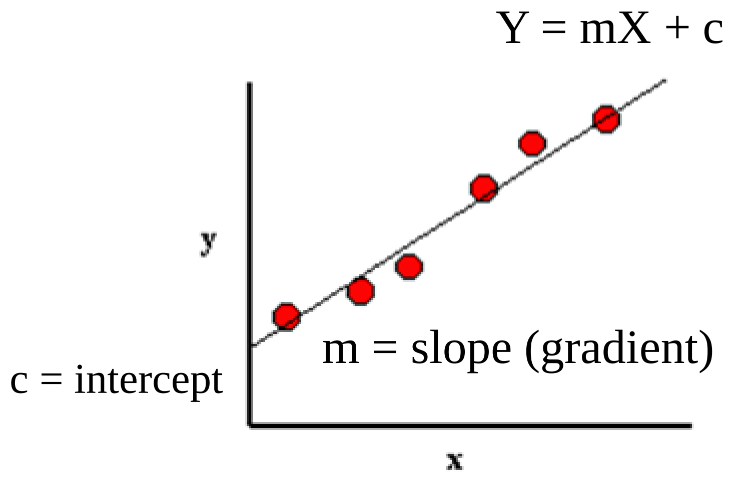
**Regression can of many types –**

- Linear Regression (parameters are linear)
- Non-linear (parameters are non linear)
- Simple
- Multiple
- Polynomial (Curvy linear – a special case of multiple regression)

# 10.2.2.1 Linear Regression

- Training set

| X | Y |
|---|---|
| 2.1 | 4 |
| 1.4 | 2.5 |
| 1.7 | 3.5 |
| .85 | 1 |
| … | … |
| … | … |



- Notation
  - m = number of **training examples**
  - x's = input variables / features
  - y's = output variable "target" variables
  - (x, y) - single training example
  - ($x^i$, $y^j$) - specific example (ith training example)

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$c = \bar{y} - m\bar{x}$$

# 10.2.2.1 Linear Regression

## Hypothesis

- Take training set
- Pass into a learning algorithm
  - Algorithm outputs a function (denoted *h* ) (h = **hypothesis**)

$$\mathbf{h_\theta(x) = \theta_0 + \theta_1 x}$$

Y is a linear function of x
$\theta_i$ are **parameters -** $\theta_0$ is zero condition or slope $\theta_1$ is gradient

- This function (hypothesis) takes an input X (e.g. size of new house)
- Uses parameters determined by a learning system
- Tries to output the estimated value of Y based on that input

- This kind of linear regression with one variable is called **univariate linear regression.**

# 10.2.2.1 Linear Regression

## Error Evaluation Metrics

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

$\hat{y}$ – *predicted value of y*
$\bar{y}$ – *mean value of y*

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

$$R^2_{adj} = 1 - (1 - R^2)\frac{N-1}{N-M-1}$$

***M – no of features***

# 10.2.2.1 Linear Regression

## Cost Function

- Choosing values for $\theta_i$ (parameters) - Different values give you different hypothesis.
- Based on training set generate parameters which give best hypothesis (h)

- **The squared error:** $\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} (\boldsymbol{h}_{\theta}(\boldsymbol{x}^{(\boldsymbol{i})}) - \boldsymbol{y}^{(\boldsymbol{i})})^{\mathbf{2}}$

- **Cost function:** $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} (\boldsymbol{h}_{\theta}(\boldsymbol{x}^{(\boldsymbol{i})}) - \boldsymbol{y}^{(\boldsymbol{i})})^{\mathbf{2}}$

1/m means we determine the average
The 2 makes the math a bit easier

Helps out how to fit the best parameters/hypothesis by minimizing squared error.

# 10.2.2.1 Linear Regression

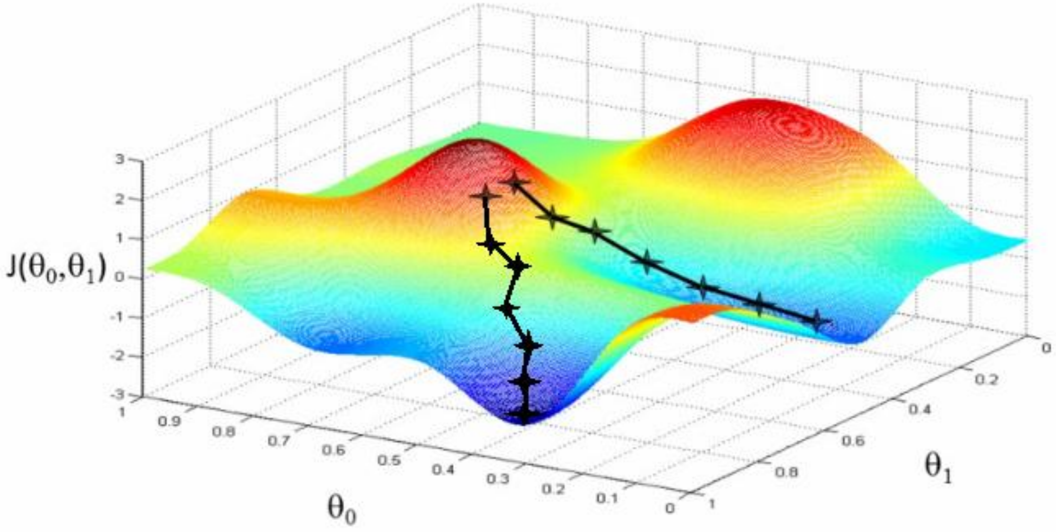
## Optimization

### Gradient Descent Algorithm

- GD Algorithm to minimize cost function used all over machine learning for minimization.
- Minimizing J means minimizing $\theta_0/\theta_1$ i.e. we get the values of $\theta_0$ and $\theta_1$ which find on average the minimal deviation of x from y

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \left(\boldsymbol{h}_\theta(\boldsymbol{x}^{(\boldsymbol{i})}) - \boldsymbol{y}^{(\boldsymbol{i})}\right)^{\boldsymbol{2}}$$

- Problem
  - We have $J(\theta_0, \theta_1)$
  - We want to get **min $\boldsymbol{J(\theta_0, \theta_1)}$**
- Gradient descent applies to more general functions
  - $J(\theta_0, \theta_1, \theta_2 \ldots. \theta_n)$
  - min $J(\theta_0, \theta_1, \theta_2 \ldots. \theta_n)$



# 10.2.2.2 Regularization Techniques (LASSO)

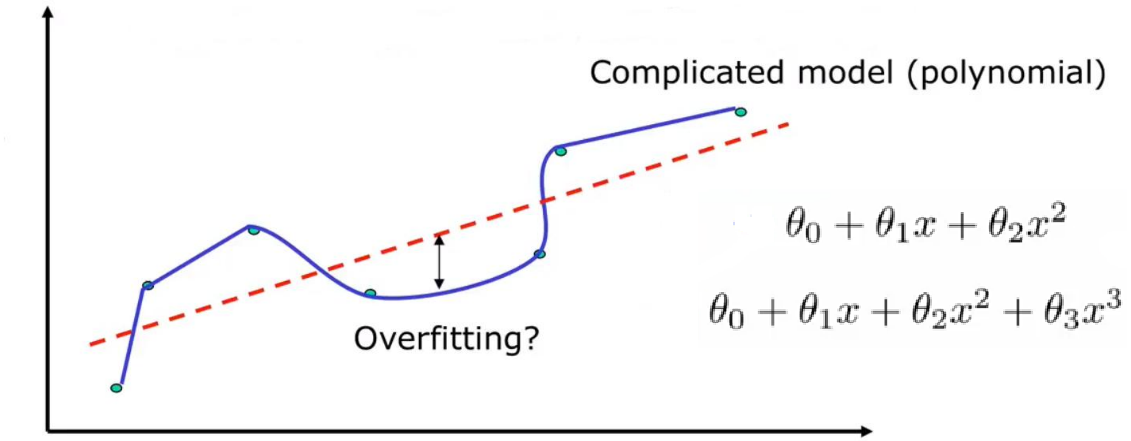
- Linear models are frequently favorable due to their interpretability and often good predictive performance.
- Ordinary Least Squares (OLS) estimation faces challenges: *Overfitting* and *Interpretability*
- Fitting techniques alternative to OLS:
  - Subset selection
  - Dimension reduction
  - Shrinkage methods
- Fit linear models with least squares but impose constraints on the coefficients
- Instead, alternative formulations add a penalty in the OLS formula
- Best known are ridge regression and LASSO (least absolute shrinkage operator)
  - Ridge regression can shrink parameters close to zero
  - LASSO models can shrink some parameters exactly to zero
  - Performs implicit variable selection

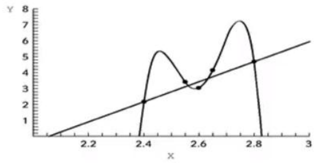
*OLS Estimation:* $$\boldsymbol{\beta}_{\text{OLS}} = \min_{\boldsymbol{\beta}} RSS = \min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

*Ridge Regression:* $$\boldsymbol{\beta}_{\text{ridge}} = \min_{\boldsymbol{\beta}} \underbrace{\sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2}_{\text{RSS}} + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{\text{shrinkage penalty}}$$

*Least Absolute Shrinkage and Selection Operator (LASSO):* $$\boldsymbol{\beta}_{\text{LASSO}} = \min_{\boldsymbol{\beta}} \underbrace{\sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2}_{\text{RSS}} + \underbrace{\lambda \sum_{j=1}^{p} |\beta|_j}_{\text{shrinkage penalty}}$$

# 10.2.2.3 Polynomial Regression


Complicated model (polynomial)
$\theta_0 + \theta_1 x + \theta_2 x^2$
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$
Overfitting?



The exact fit is not necessarily the best fit -overfitting

- How do we fit the model to this data
- To map our old linear hypothesis and cost functions to these polynomial descriptions set

$x_1 = x$

$x_2 = x^2$

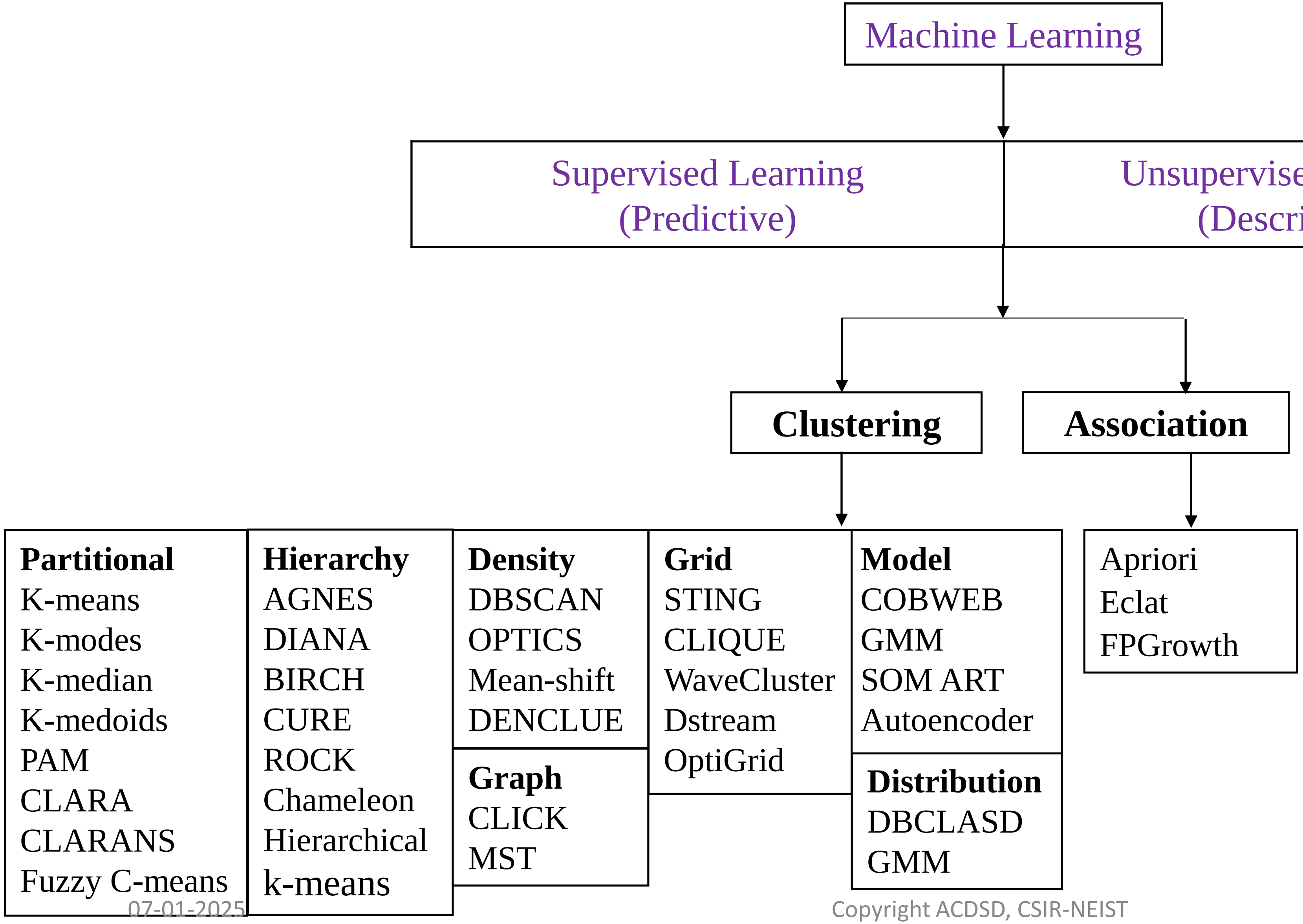
$x_3 = x^3$

# 10.3 Unsupervised Learning

## The Topics Covered in This Section

- 10.3.1 Clustering
  - 10.3.1.1 K-Means
  - 10.3.1.2 K Nearest Neighbors
  - 10.3.1.3 Association Rule Learning
- 10.3.2 Dimensionality Reduction
  - 10.3.2.1 PCA
  - 10.3.2.2 LDA
  - 10.3.2.3 GDA
  - 10.3.2.4 Examples

# 10.3 Unsupervised Learning



Machine Learning

Supervised Learning (Predictive)

Unsupervised Learning (Descriptive)

**Clustering**

**Association**

**Partitional**
K-means
K-modes
K-median
K-medoids
PAM
CLARA
CLARANS
Fuzzy C-means

**Hierarchy**
AGNES
DIANA
BIRCH
CURE
ROCK
Chameleon
Hierarchical k-means

**Density**
DBSCAN
OPTICS
Mean-shift
DENCLUE

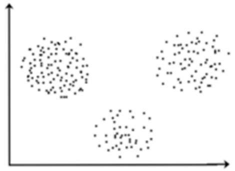
**Graph**
CLICK
MST

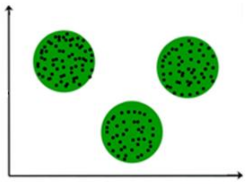
**Grid**
STING
CLIQUE
WaveCluster
Dstream
OptiGrid

**Model**
COBWEB
GMM
SOM ART
Autoencoder

**Distribution**
DBCLASD
GMM

Apriori
Eclat
FPGrowth





{X1, X2} → {Y1, Y2}
Antecedent Consequent

Itemset = {X1, X2, Y1, Y2}

# 10.3.1 Clustering

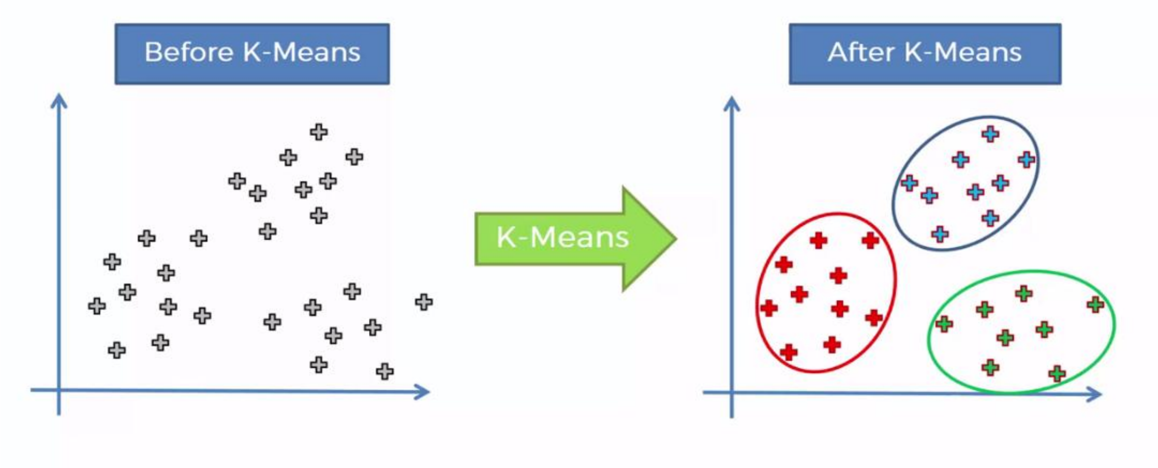
- Clustering involves grouping data into categories based on some measure of similarity or distance.
- Thus clustering requires some methods for computing (dis)**similarity** or **distance** between each pair of observations.
- The result of this computation is known as a dissimilarity or **distance metrics**.

**Similarity among points is defined using –**

- Some inter-observation distance measures
- Correlation-based distance measures

# 10.3.1.1 K-Means

- Clustering algorithm for partitioning a given data set into a set of k groups (i.e. *k clusters*), where k represents the number of groups pre-specified by the analyst.
- K-means algorithm allows that each one of the data belong to only a cluster.
- Therefore, this algorithm is a definite clustering and hard clustering algorithm.



Divides the observations into discrete groups based on some distance metric

# 10.3.1.1 K-Means

**ALGORITHM: K-Means**

**Input:** $\{x^1, x^2, x^3 \ldots, x^n\}$ //Training Set

K // Number of desired clusters

**Output:** K // Set of Clusters

```
Randomly initialize K cluster centroids as $\{\mu_1, \mu_2, \mu_3 \ldots \mu_K\}$
Repeat

    for i=1 to n
        $c^{(i)}$ = index of cluster centroid (from 1 to K) closest to $x^{(i)}$

    for k=1 to K
        $\mu_K$ = average(mean) of points assigned to cluster k

Until convergence criteria is meet
```

# 10.3.1.1 K-Means

- Optimization objective:

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

where,

- $\mu_k$ is the cluster associated with centroid *k*
- $c^i$ is the index of clusters {1,2, ..., K} to which $x^i$ is currently assigned.
- So, $\mu_c{}^i$ is the cluster centroid of the cluster to which example $x^i$ has been assigned to.

- Minimizing cost:

$$\min_{\substack{c^{(1)}, \dots, c^{(m)}, \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

where,

- The **cluster assigned step** is minimizing J(...) with respect to $c^1$, $c^2$ ... $c^i$
- The **move centroid step** is minimizing J(...) with respect to μ

# 10.3.1.1 K-Means

**Initializing the centroids – Random Initialization**

- Randomly pick K training examples
- Set $\mu_1$ up to $\mu_K$ to these example's values

**Effects of Random Initialization**

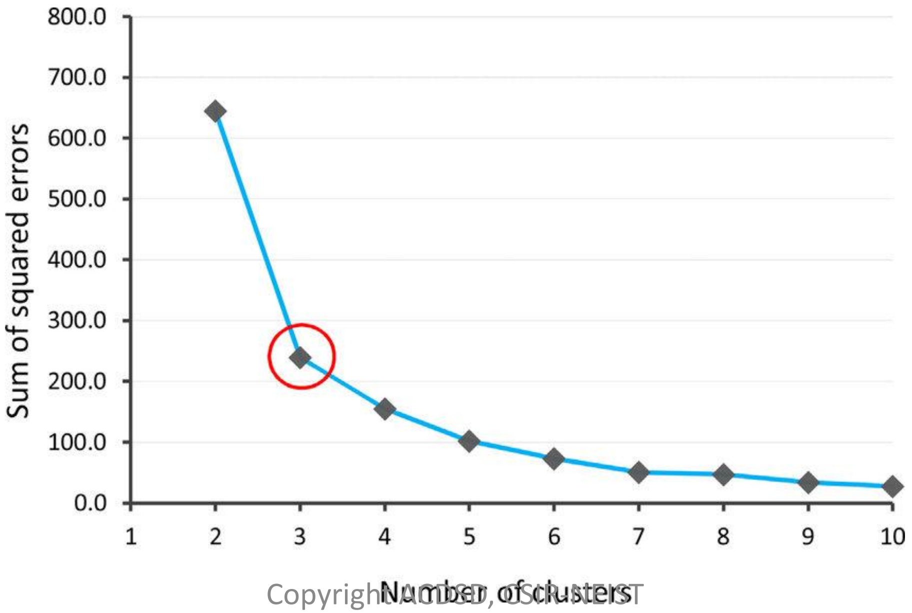
- Kmeans can converge to different solutions depending on the initialization setup
- That means risk of local optimum.
- The local optimum are valid convergence, but local optimum are not global ones
- If this is a concern go for multiple random initializations.

**Deciding K - Elbow method**

- Compute cost function at a range of K values (As K increases J(...) minimum value decreases)
- Plot (K vs J())
- Look for the "elbow" on the graph and Chose the "elbow" number of clusters.

**Risks:**

- Normally we don't get a nice line – no clear elbow on curve
- Not really that helpful



# 10.3.1.1 K-Means

**Strength:**

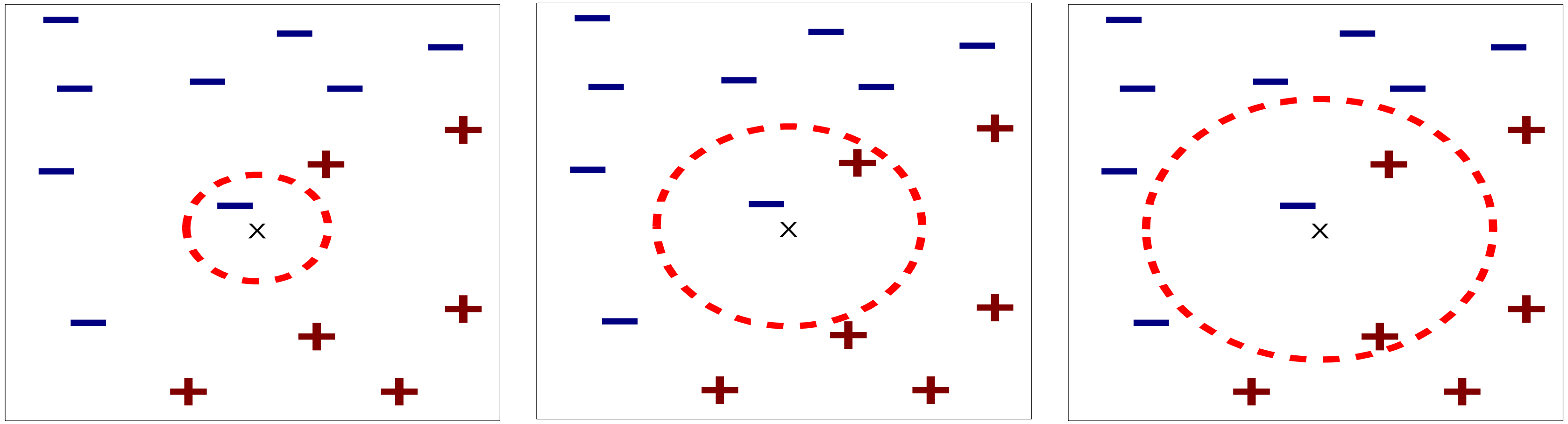
- Easy to implement.
- Efficient: O(tkn) where n is number of objects, k is number of clusters, and t is number of iterations.
- k-Means produce tighter clusters than hierarchical clustering.
- K-means is scalable and efficient for large dataset with complexity O(nkt).

**Weakness:**

- Need to specify k, the number of clusters, in advance.
- Initial seeds have a strong impact on the final results.
- Applicable only when mean is defined - Categorical data.
- Unable to handle noisy data and outliers.
- Not suitable to discover clusters with non-convex shapes.

# 10.3.1.2 K Nearest Neighbor

- Input features – can be both quantitative and qualitative
- Output feature – always categorical values
- Explains a categorical value using the majority votes of (k) nearest neighbors.
- Select the tuning parameter k based on data and experiment.
- Requires distance metric to define proximity between any two data points –
    - Manhattan Distance
    - Euclidean Distance
    - Minkowski Distance
    - Hamming Distance
    - Cosine Distance
    - Mahalanobis Distance



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

# 10.3.1.2 K Nearest Neighbor

## Distance Metrics

Chebyshev Distance: $\max_i(|x_i - y_i|)$

Manhattan/City Block Distance ($L_\infty$): $\sum_{i=1}^{k} |x_i - y_i|$

Euclidean Distance: $\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$

Minkowski Distance: $\left( \sum_{i=1}^{k} (|x_i - y_i|)^q \right)^{1/q}$

Hamming Distance: $D_H = \sum_{i=1}^{k} |x_i - y_i|$

Cosine Distance: $\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$

Mahalanobis Distance: $d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}$ where $S^{-1}$ is covariance matrix of the input data

Canberra: $\sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$

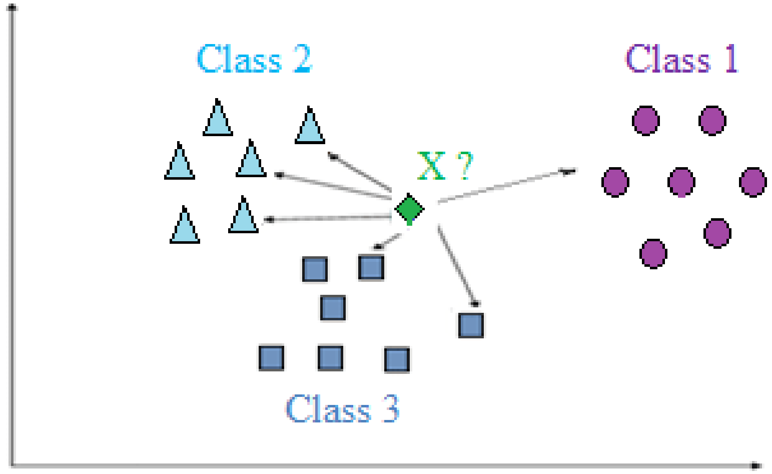
Correlation: $\frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$

Chi· $\sum_{i=1}^{m} \frac{1}{sum_i} \left( \frac{x_i}{size_x} - \frac{y_i}{size_y} \right)^2$

# 10.3.1.2 K Nearest Neighbor

## KNN Algorithm

1. Load the data
2. Initialize the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
    a) Calculate distance (metrics ex.- Euclidean) between test data and each row of training data.
    b) Sort the calculated distances in ascending order based on distance values
    c) Get top k rows from the sorted array
    d) Get the most frequent class of these rows
    e) Return the predicted class

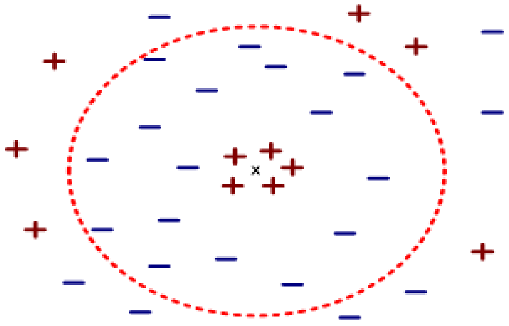


# 10.3.1.2 K Nearest Neighbor

## What is k = ?

Choosing the value of k:

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes



**Rule of thumb:**

K = sqrt(N) where N: number of training point

In practice, *k* is usually chosen to be odd, so as to avoid ties.

- **Training:** O(1)
- **Testing:** O(nd)

Training set: n sample
Testing instance: 1 sample
Nearest Neighbor:
Compare One by one to each training instance
Complexity: N comparisons and each comparisons takes d operations

**Making Fast**

- **Reduce d:** dimensionality reduction
- **Reduce n:** do not compare to all training examples

# 10.3.1.3 Association Rule Learning

**What is Association Mining:** Finding/mining frequent patterns and rules.

**Stages:**

- Find frequent patterns.
- Derive associations from frequent patterns.

**Example (Applications):**

- Graphs - Social Network Analysis
- Transactions - Market Basket Analysis
- Sequences - Time Series Analysis
- Topic Identification - Co-occurrence of words, Plagiarism Detection
- Bio-maskers - Genes or proteins vs. disease

# 10.3.1.3 Association Rule Learning

**Association Rule**

- An implication expression of the form X → Y, where X and Y are itemsets
- Example: {Milk, Diaper} → {Beer}

**Rule Evaluation Metrics:**

- Support (s): Fraction of transactions that contain both X and Y
- Confidence (c): Measures how often items in Y appear in transactions that contain X

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

| ***TID*** | ***Items*** |
|---|---|
| **1** | **Bread, Milk** |
| **2** | **Bread, Diaper, Beer, Eggs** |
| **3** | **Milk, Diaper, Beer, Coke** |
| **4** | **Bread, Milk, Diaper, Beer** |
| **5** | **Bread, Milk, Diaper, Coke** |

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

# 10.3.1.3 Association Rule Learning

- The goal of association rule mining is to find frequent itemset i.e. all rules having
    - Support ≥ *minsup* threshold
    - Confidence ≥ *minconf* threshold

**Method**

- Brute-force approach
- Apriori
- Frequency Pattern Growth

# 10.3.1.3 Association Rule Learning

## Brute Force Approach

1. Frequent Itemset Generation
   - Generate all combinatorically possible itemsets whose support ≥ minsup
2. Rule Generation
   - List all possible association rules.
   - Compute the support and confidence for each rule.
   - Prune rules that fail the *minsup* and *minconf* thresholds.

**Note:** Frequent itemset generation is still computationally expensive

**Factors Affecting Complexity:**

- Choice of minimum support threshold
- Dimensionality (number of items) of the data set
- Size of database (number of transaction)
- Average transaction width

# 10.3.1.3 Association Rule Learning

## Brute Force Approach

**Rule Generation**

- How to efficiently generate rules from frequent itemsets?
- In general, confidence does not have an anti-monotone property

  c(ABC →D) can be larger or smaller than c(AB →D)

- But confidence of rules generated from the same itemset has an anti-monotone property

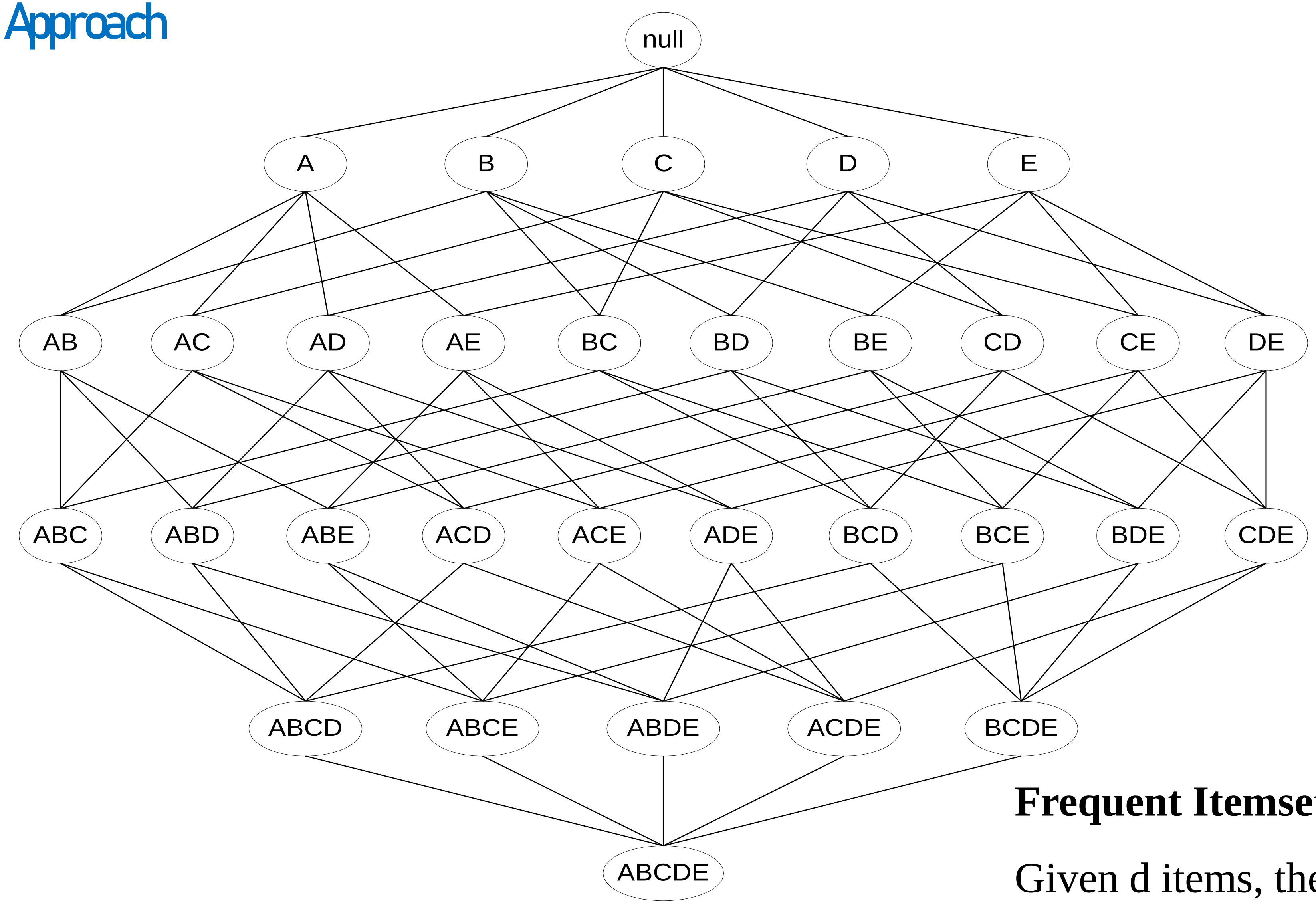
  e.g., L = {A,B,C,D}:

  $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$

- Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

# 10.3.1.3 Association Rule Learning

## Brute Force Approach



**Frequent Itemset Generation**

Given d items, there are $2^d$ possible candidate itemsets

# 10.3.1.3 Association Rule Learning

## Apriori Algorithm

**Goal**

- Reducing Number of Candidates.

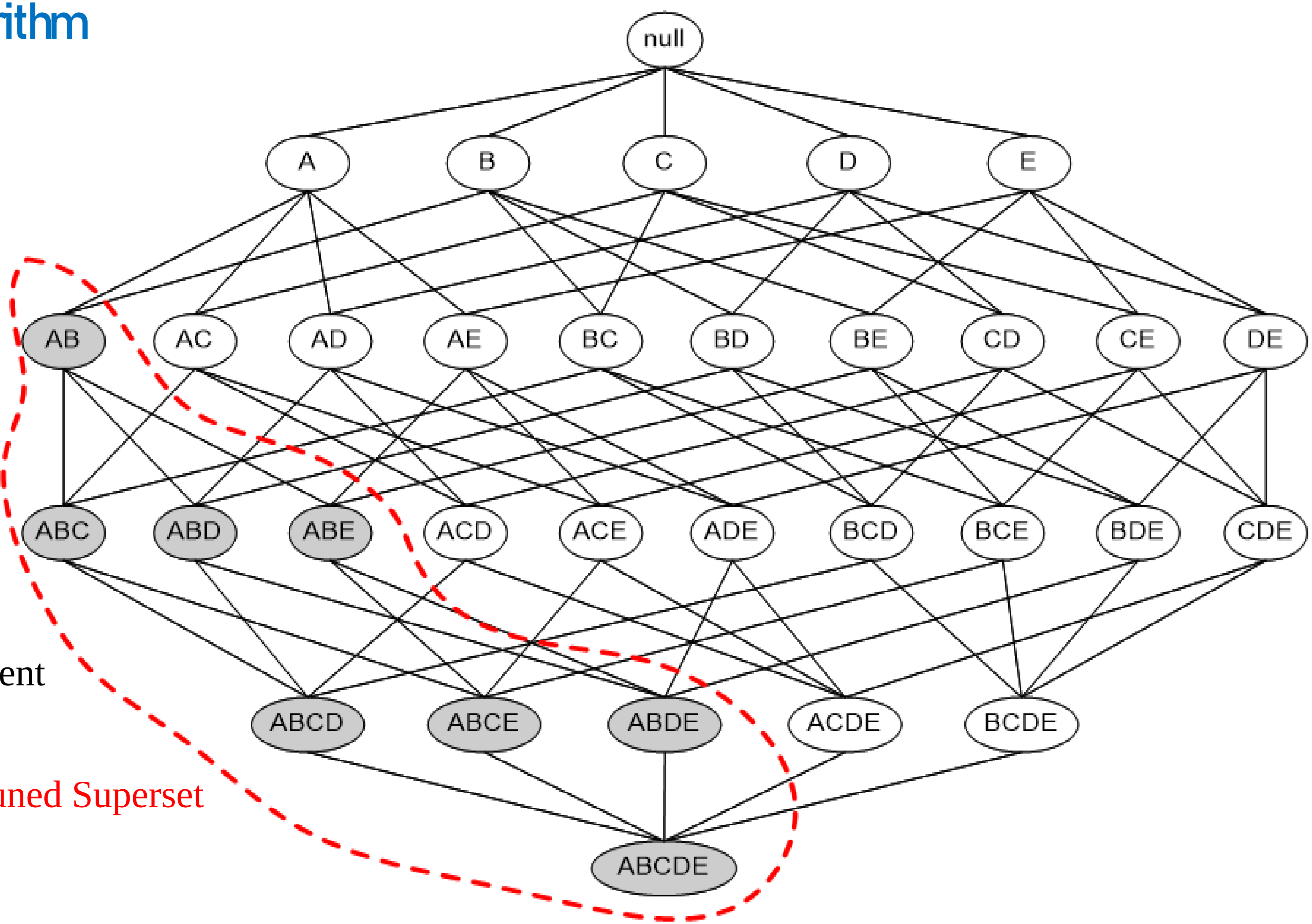
**Apriori principle**

- If an itemset is frequent, then all of its subsets must also be frequent.
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- Support of an itemset never exceeds the support of its subsets (anti-monotone property of support)

## Apriori Algorithm



Found to be infrequent

Pruned Superset

## Apriori Algorithm

### Pattern Evaluation

- Association rule algorithms tend to produce too many rules
  - Many of them are uninteresting or redundant
  - Redundant if {A,B,C} → {D} and {A,B} → {D} have same support & confidence

- Interestingness measures (Statistical-based) can be used to prune/rank the derived patterns.

- In the original formulation of association rules, support & confidence are the only measures used.

# 10.3.1.3 Association Rule Learning

## Apriori Algorithm

**Statistical Independence**

- Population of 1000 students
  - 600 students know how to swim (S)
  - 700 students know how to bike (B)
  - 420 students know how to swim and bike (S,B)

  - $P(S \wedge B) = 420/1000 = 0.42$
  - $P(S) \times P(B) = 0.6 \times 0.7 = 0.42$

  - $P(S \wedge B) = P(S) \times P(B)$ => Statistical independence
  - $P(S \wedge B) > P(S) \times P(B)$ => Positively correlated
  - $P(S \wedge B) < P(S) \times P(B)$ => Negatively correlated

# 10.3.1.3 Association Rule Learning

## Apriori Algorithm

**Statistical-based Measures**

- Take into account statistical dependence

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$

$$Interest = \frac{P(X,Y)}{P(X)P(Y)}$$

$$PS = P(X,Y) - P(X)P(Y)$$

$$\phi - coefficient = \frac{P(X,Y) - P(X)P(Y)}{\sqrt{P(X)[1-P(X)]P(Y)[1-P(Y)]}}$$

Contingency table for supports X → Y

| | Coffee | $\overline{\text{Coffee}}$ | |
|---|---|---|---|
| Tea | 15 | 5 | 20 |
| $\overline{\text{Tea}}$ | 75 | 5 | 80 |
| | 90 | 10 | 100 |

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Lift = 0.75/0.9= 0.8333 (< 1, so negatively associated)

# 10.3.1.3 Association Rule Learning

## FP Growth

- An efficient and scalable method to complete set of frequent patterns.
- It allow frequent item set discovery without candidate item set generation.
- Two step approach:
    1. Build a compact data structure called the FP-Tree.
    2. Extracts frequent item set directly from the FP-Tree.

**Steps**

1. Calculate minimum support
2. Find frequency of occurrence of each item
3. Prioritize the items according to the frequencies
4. Order the transactions according to priority of items
5. Build FP Tree (row by row for each ordered transaction)
6. Validation of the FP Tree
7. Extract Information

# 10.3.1.3 Association Rule Learning

## Example on Frequent Patterns

Suppose we have the following transaction

| TID | Items |
|---|---|
| 1 | E, A, D, B |
| 2 | D, A, C, E, B |
| 3 | C, A, B, E |
| 4 | B, A, D |
| 5 | D |
| 6 | D, B |
| 7 | A, D, E |
| 8 | B, C |

Step 1

Step 2

Step 3

| Items | Freq |
|---|---|
| A | 5 |
| B | 6 |
| C | 3 |
| D | 6 |
| E | 4 |

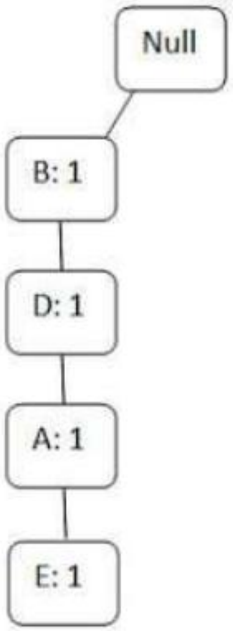
| Items | Frequency | Priority |
|---|---|---|
| A | 5 | 3 |
| B | 6 | 1 |
| C | 3 | 5 |
| D | 6 | 2 |
| E | 4 | 4 |

# 10.3.1.3 Association Rule Learning

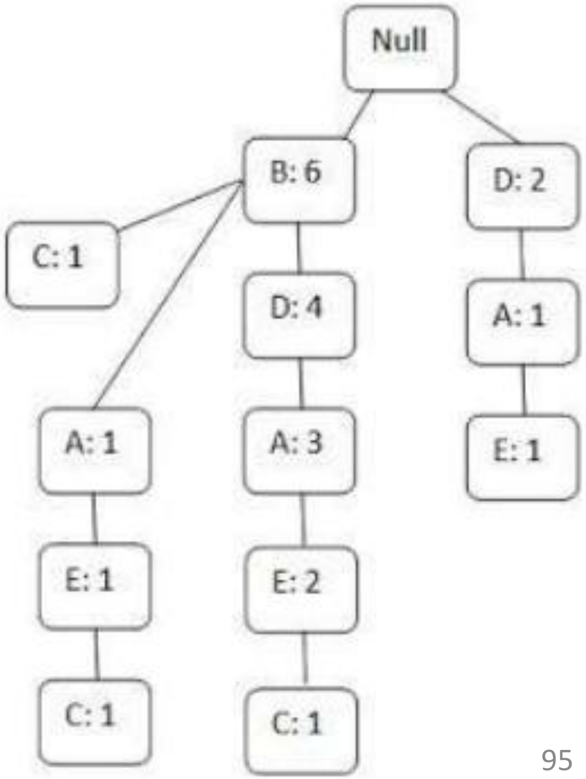
## Example on Frequent Patterns

Step 4

| TID | Items | Ordered |
|---|---|---|
| 1 | E, A, D, B | B, D, A, E |
| 2 | D, A, C, E, B | B, D, A, E, C |
| 3 | C, A, B, E | B, A, E, C |
| 4 | B, A, D | B, D, A |
| 5 | D | D |
| 6 | D, B | B, D |
| 7 | A, D, E | D, A, E |
| 8 | B, C | B, C |



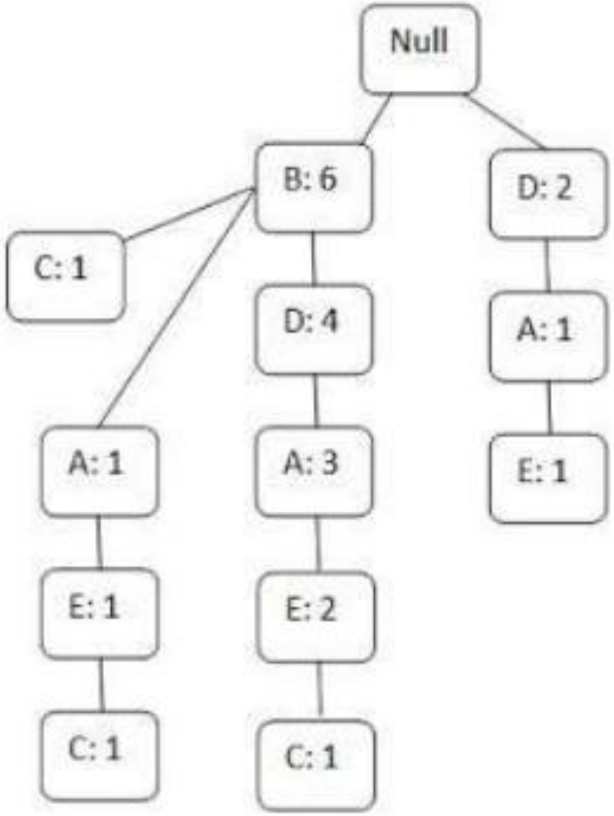
Step 5



# 10.3.1.3 Association Rule Learning

## Example on Frequent Patterns

Step 6

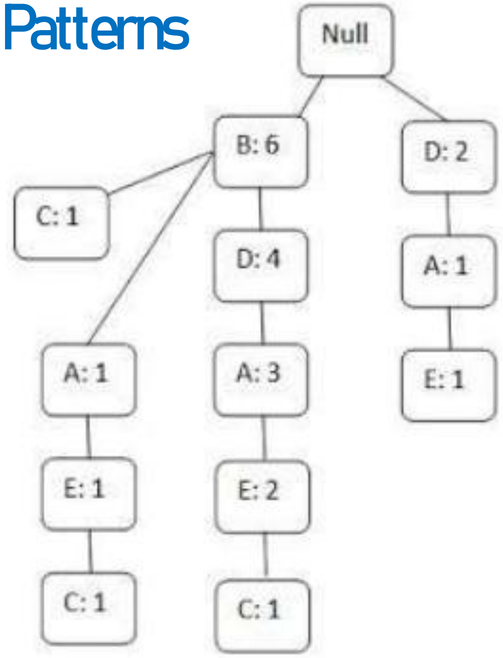


| Items | Frequency |
|---|---|
| A | 5 |
| B | 6 |
| C | 3 |
| D | 6 |
| E | 4 |

# 10.3.1.3 Association Rule Learning

## Example on Frequent Patterns

Step 7



| Priority item | Item | Conditional pattern base |
|---|---|---|
| B (1) | C | { B: 1} {BAE:1} {BDAE:1} |
| D (2) | E | {BA:1} {BDA:2} {DA:1} |
| A (3) | A | {B:1} {BD:3} {D:1} |
| E (4) | D | {B:4} |
| C (5) | B | - |

| Priority item | Item | Conditional pattern base | | Conditional FP tree (>minsupp) | Frequent pattern |
|---|---|---|---|---|---|
| B (1) | C | { B: 1} {BAE:1} {BDAE:1} | {B:3, A: 2, E:2, D:1} | {B:3} | {BC:3} |
| D (2) | E | {BA:1} {BDA:2} {DA:1} | {A:3, B:3,D:2}, {D: 1, A:1} | {A:3, B:3} | {AE: 3} {BE:3} {ABE:3} |
| A (3) | A | {B:1} {BD:3} {D:1} | {B:4, D:3} {D:1} | {B:4, D:3} | {BA:4} {DA: 3} (BAD:3) |
| E (4) | D | {B:4} | {B:4} | {B:4} | {BD:4} |
| C (5) | B | - | - | - | |

# 10.3.2 Dimensionality Reduction

- **Purpose**
  - Avoid curse of dimensionality.
  - Allow data to be more easily visualized.
  - May help to eliminate irrelevant (unimportant, dummy, redundant) features or reduce noise.
  - Reduce amount of time and memory required by data mining algorithms.
- **Techniques** (Reducing feature set with minimal loss of information)
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - Generalized Discriminant Analysis (GDA)
- **Curse of Dimensionality**
  - Issue due to high dimension data
  - More data is good, more detailed (dimension) data may not be

# 10.3.2.1 Principle Component Analysis (PCA)

- PCA is an unsupervised dimensionality reduction algorithm as it ignores the class labels.

- PCA does not select a set of features and discard other features.
- PCA infers some new features, which best describe the type of class from the existing features.

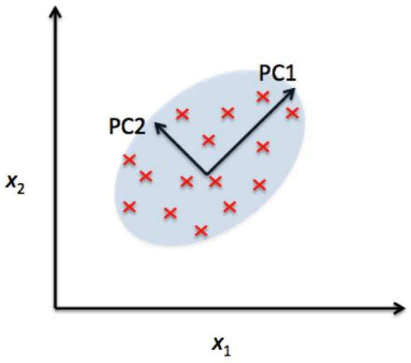
- PCA finds the lines of maximum variance in the dataset known as principal components.
  - PC1 having the maximum variance
  - PC2 having second maximum variance and so on.
- The components are orthogonal and hence highly uncorrelated with each other.

- PCA works on eigenvectors and eigenvalues of the covariance matrix (linear combination of variables), which is the equivalent of fitting those straight PC lines to the variance of the data.

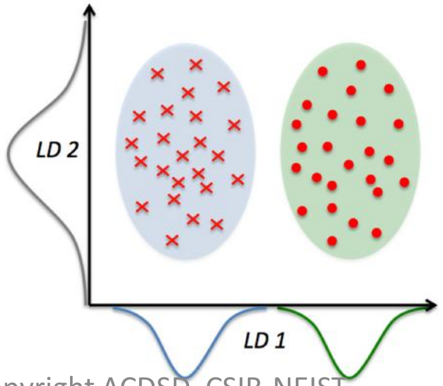
# 10.3.2.2 Linear Discriminant Analysis (LDA)

- LDA is supervised dimension reduction algorithm as it considers ignores the class labels.
- LDA is not necessarily a classifier, but can be used as one as it maintains the class separability.
- LDA finds components (linear combination of the observed variables) that maximize class separation (between-class variance) when such prior information is available (i.e. supervised).
- LDA finds a centroid of each class datapoints.
- Based on this, it determines a new dimension which should satisfy two criteria:
  - Maximize the distance between the centroid of each class.
  - Minimize the variation within each category.
- If the variation of data is minimum, then less overlapping between the classes will be there and maximum separation will be maintained between the different classes.

***PCA vs LDA***



PCA



LDA

**Note:** PC values are between -1 and +1

# 10.3.2.3 Generalized Discriminant Analysis (GDA)

- Generalized Discriminant Analysis (GDA) has provided an extremely powerful approach to extracting non-linear features
- The Generalized Discriminant Analysis is used for multi-class classification problems.
- GDA is employed to reduce the high dimensional data vectors and identification is handled in a low dimensional space with high efficiency and low use of system resources

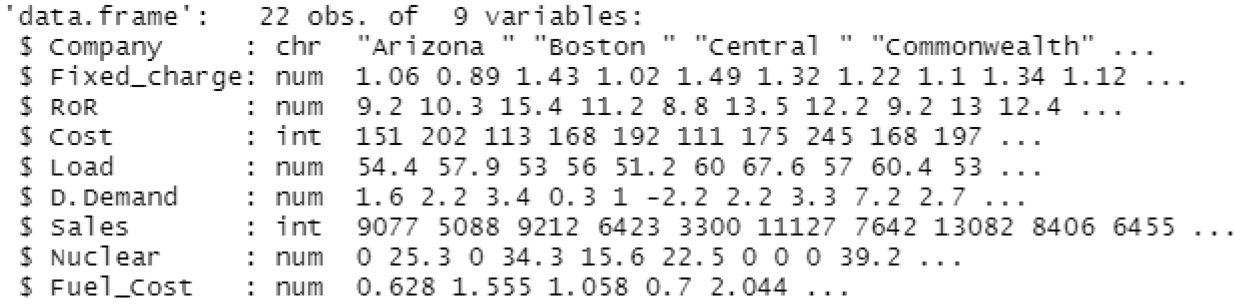
- ***Virtues where LDA fails:***
  - Linear boundaries fail to separate data
  - May want to model non linear data
  - Too many parameters estimated with high variance
  - Need of regularization

- Generalization of LDA to solve the issues:
  - ❑ Flexible Discriminant Analysis (FDA): LDA in enlarged space of predictors
  - ❑ Penalized Discriminant Analysis (PDA) : Can be used with large number of predictors
  - ❑ Mixture Discriminant Analysis (MDA): Model by mixture of two or more Gaussians sharing same covariance matrix

# 10.3.2.4 Examples

## Cluster Analysis with Herarchical & K-Means

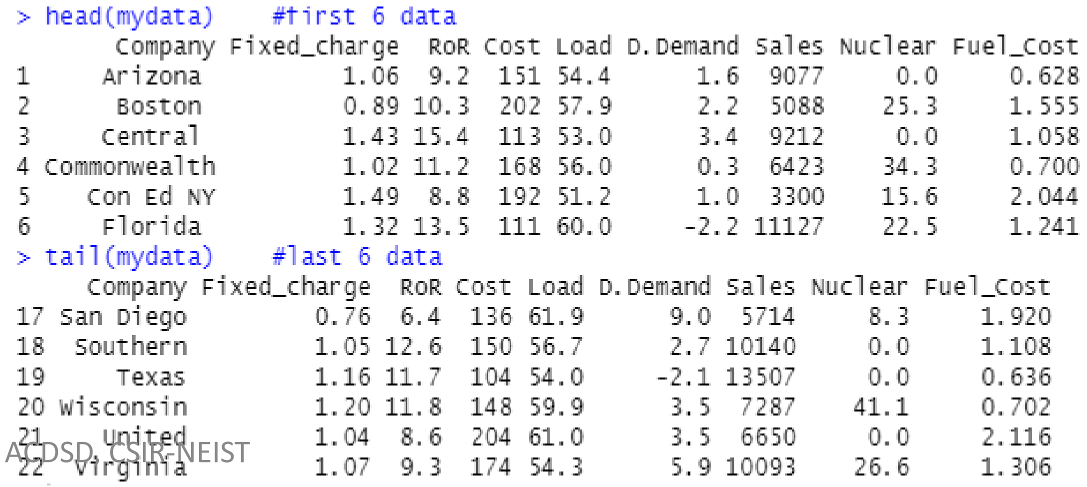
A dataset containing 22 observations with 9 variables for different companies has been given. Using this dataset design clustering models with Hierarchical & K-Means clustering approaches in R.



```
'data.frame':	22 obs. of  9 variables:
 $ Company     : chr  "Arizona " "Boston " "Central " "Commonwealth" ...
 $ Fixed_charge: num  1.06 0.89 1.43 1.02 1.49 1.32 1.22 1.1 1.34 1.12 ...
 $ RoR         : num  9.2 10.3 15.4 11.2 8.8 13.5 12.2 9.2 13 12.4 ...
 $ Cost        : int  151 202 113 168 192 111 175 245 168 197 ...
 $ Load        : num  54.4 57.9 53 56 51.2 60 67.6 57 60.4 53 ...
 $ D.Demand    : num  1.6 2.2 3.4 0.3 1 -2.2 2.2 3.3 7.2 2.7 ...
 $ Sales       : int  9077 5088 9212 6423 3300 11127 7642 13082 8406 6455 ...
 $ Nuclear     : num  0 25.3 0 34.3 15.6 22.5 0 0 0 39.2 ...
 $ Fuel_Cost   : num  0.628 1.555 1.058 0.7 2.044 ...
```

```
# reading data from dataset
mydata <- read.csv(file.choose(), header=T)
str(mydata)
head(mydata)
tail(mydata)
```

Read first and
last 6 entries

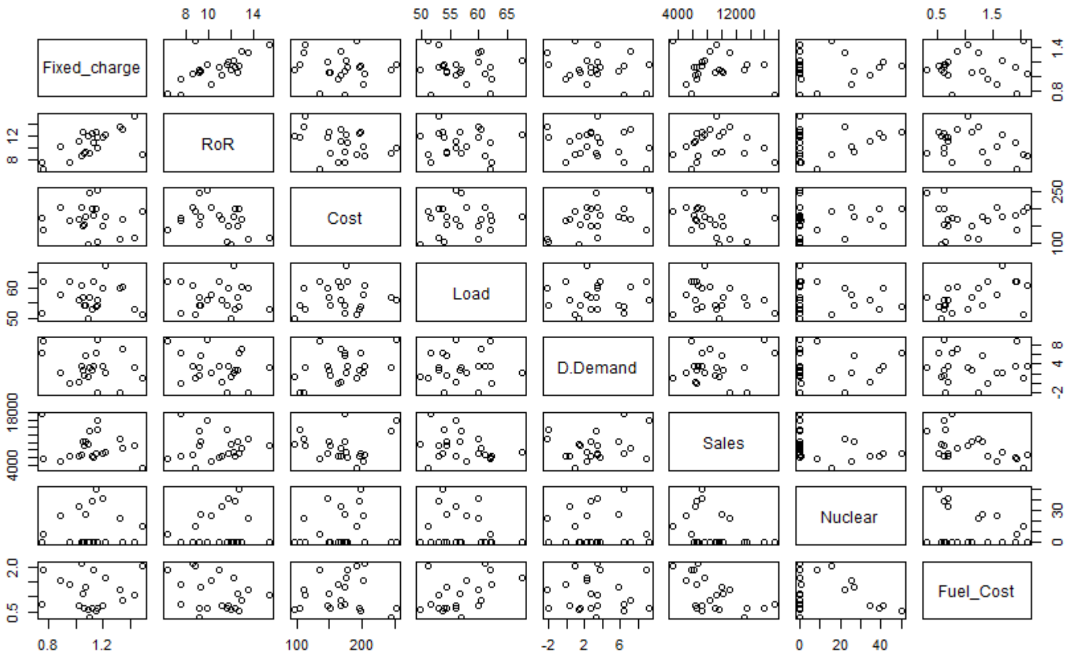


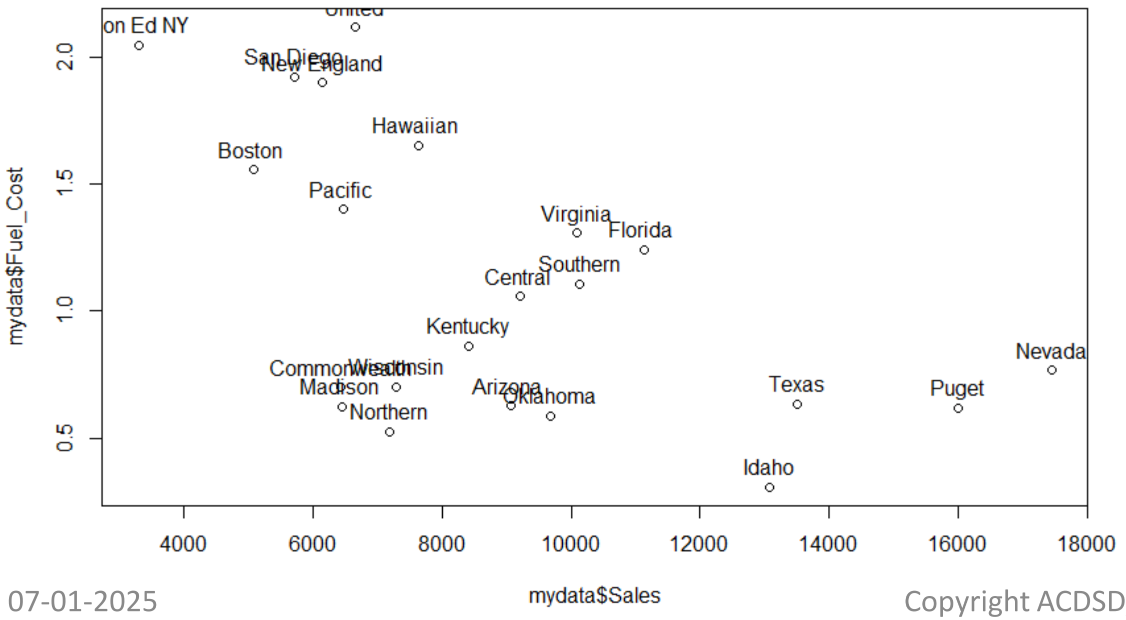
```
> head(mydata)    #first 6 data
       Company Fixed_charge  RoR Cost Load D.Demand Sales Nuclear Fuel_Cost
1      Arizona         1.06  9.2  151 54.4      1.6  9077     0.0     0.628
2       Boston         0.89 10.3  202 57.9      2.2  5088    25.3     1.555
3      Central         1.43 15.4  113 53.0      3.4  9212     0.0     1.058
4 Commonwealth         1.02 11.2  168 56.0      0.3  6423    34.3     0.700
5    Con Ed NY         1.49  8.8  192 51.2      1.0  3300    15.6     2.044
6      Florida         1.32 13.5  111 60.0     -2.2 11127    22.5     1.241
> tail(mydata)    #last 6 data
     Company Fixed_charge  RoR Cost Load D.Demand Sales Nuclear Fuel_Cost
17 San Diego         0.76  6.4  136 61.9      9.0  5714     8.3     1.920
18  Southern         1.05 12.6  150 56.7      2.7 10140     0.0     1.108
19     Texas         1.16 11.7  104 54.0     -2.1 13507     0.0     0.636
20 Wisconsin         1.20 11.8  148 59.9      3.5  7287    41.1     0.702
21    United         1.04  8.6  204 61.0      3.5  6650     0.0     2.116
22  Virginia         1.07  9.3  174 54.3      5.9 10093    26.6     1.306
```

# 10.3.2.4 Examples

```
# Visualization
pairs(mydata[2:9])


# Scatter plot
plot(mydata$Fuel_Cost~ mydata$Sales, data = mydata)
with(mydata,text(mydata$Fuel_Cost ~ mydata$Sales,
labels=mydata$Company,pos=3))
```

Scatter plot between all possible combinations of two variables



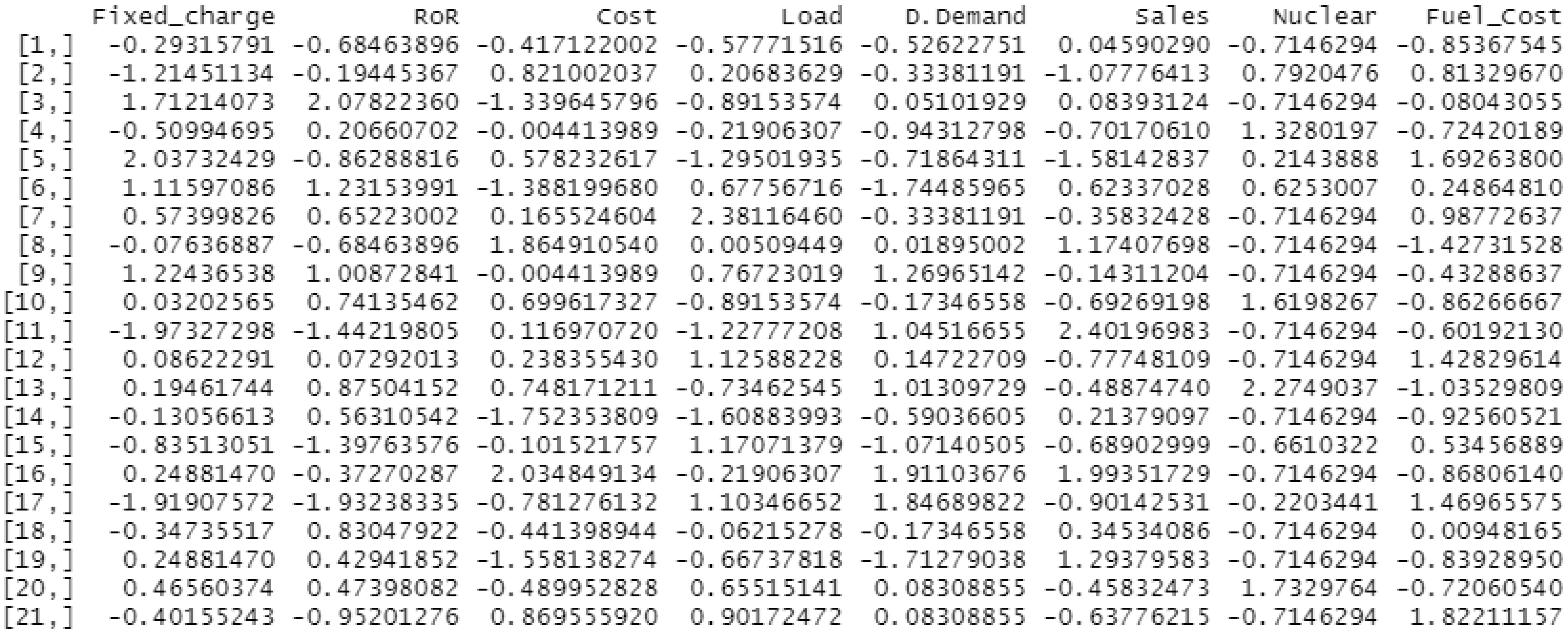


# 10.3.2.4 Examples

```
# Normalization
z <- mydata[,-c(1,1)]

means <- apply(z,2,mean)
sds <- apply(z,2,sd)
nor <- scale(z,center=means,scale=sds)
```

Remove first column

Normalize

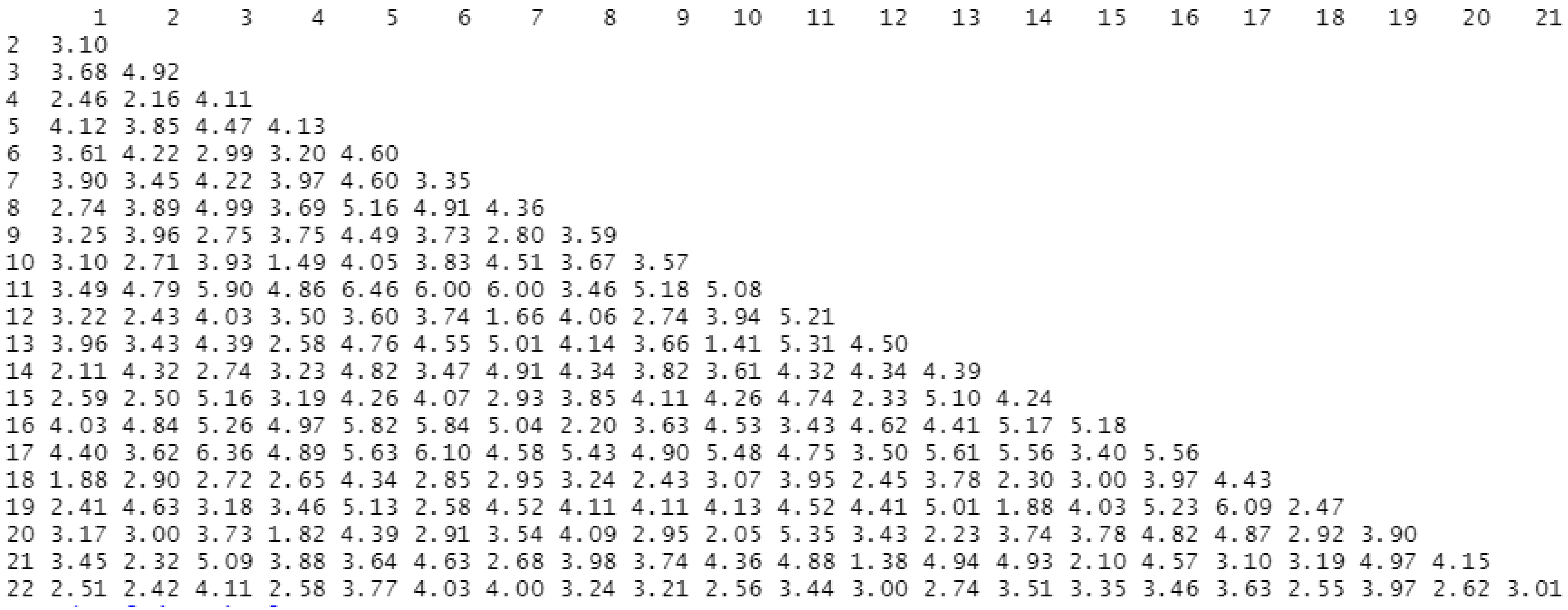


```
      Fixed_charge         RoR         Cost        Load    D.Demand       Sales    Nuclear   Fuel_Cost
 [1,]  -0.29315791 -0.68463896 -0.417122002 -0.57771516 -0.52622751  0.04590290 -0.7146294 -0.85367545
 [2,]  -1.21451134 -0.19445367  0.821002037  0.20683629 -0.33381191 -1.07776413  0.7920476  0.81329670
 [3,]   1.71214073  2.07822360 -1.339645796 -0.89153574  0.05101929  0.08393124 -0.7146294 -0.08043055
 [4,]  -0.50994695  0.20660702 -0.004413989 -0.21906307 -0.94312798 -0.70170610  1.3280197 -0.72420189
 [5,]   2.03732429 -0.86288816  0.578232617 -1.29501935 -0.71864311 -1.58142837  0.2143888  1.69263800
 [6,]   1.11597086  1.23153991 -1.388199680  0.67756716 -1.74485965  0.62337028  0.6253007  0.24864810
 [7,]   0.57399826  0.65223002  0.165524604  2.38116460 -0.33381191 -0.35832428 -0.7146294  0.98772637
 [8,]  -0.07636887 -0.68463896  1.864910540  0.00509449  0.01895002  1.17407698 -0.7146294 -1.42731528
 [9,]   1.22436538  1.00872841 -0.004413989  0.76723019  1.26965142 -0.14311204 -0.7146294 -0.43288637
[10,]   0.03202565  0.74135462  0.699617327 -0.89153574 -0.17346558 -0.69269198  1.6198267 -0.86266667
[11,]  -1.97327298 -1.44219805  0.116970720 -1.22777208  1.04516655  2.40196983 -0.7146294 -0.60192130
[12,]   0.08622291  0.07292013  0.238355430  1.12588228  0.14722709 -0.77748109 -0.7146294  1.42829614
[13,]   0.19461744  0.87504152  0.748171211 -0.73462545  1.01309729 -0.48874740  2.2749037 -1.03529809
[14,]  -0.13056613  0.56310542 -1.752353809 -1.60883993 -0.59036605  0.21379097 -0.7146294 -0.92560521
[15,]  -0.83513051 -1.39763576 -0.101521757  1.17071379 -1.07140505 -0.68902999 -0.6610322  0.53456889
[16,]   0.24881470 -0.37270287  2.034849134 -0.21906307  1.91103676  1.99351729 -0.7146294 -0.86806140
[17,]  -1.91907572 -1.93238335 -0.781276132  1.10346652  1.84689822 -0.90142531 -0.2203441  1.46965575
[18,]  -0.34735517  0.83047922 -0.441398944 -0.06215278 -0.17346558  0.34534086 -0.7146294  0.00948165
[19,]   0.24881470  0.42941852 -1.558138274 -0.66737818 -1.71279038  1.29379583 -0.7146294 -0.83928950
[20,]   0.46560374  0.47398082 -0.489952828  0.65515141  0.08308855 -0.45832473  1.7329764 -0.72060540
[21,]  -0.40155243 -0.95201276  0.869555920  0.90172472  0.08308855 -0.63776215 -0.7146294  1.82211157
```

# 10.3.2.4 Examples

```
# Calculate distance
distance = dist(nor)
print(distance,digits = 3)
mydata[c(5,11), ]
nor
```

Calculate
Euclidean distance

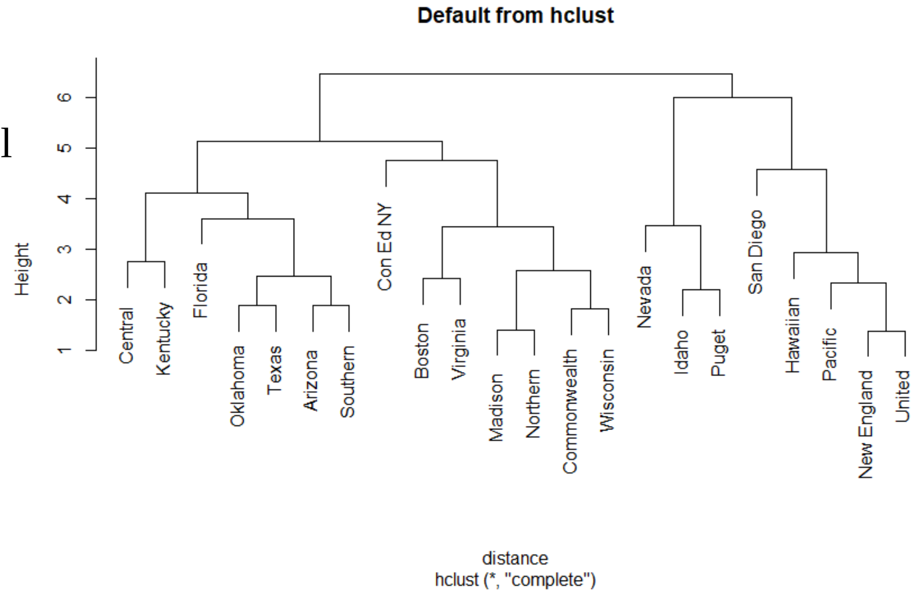


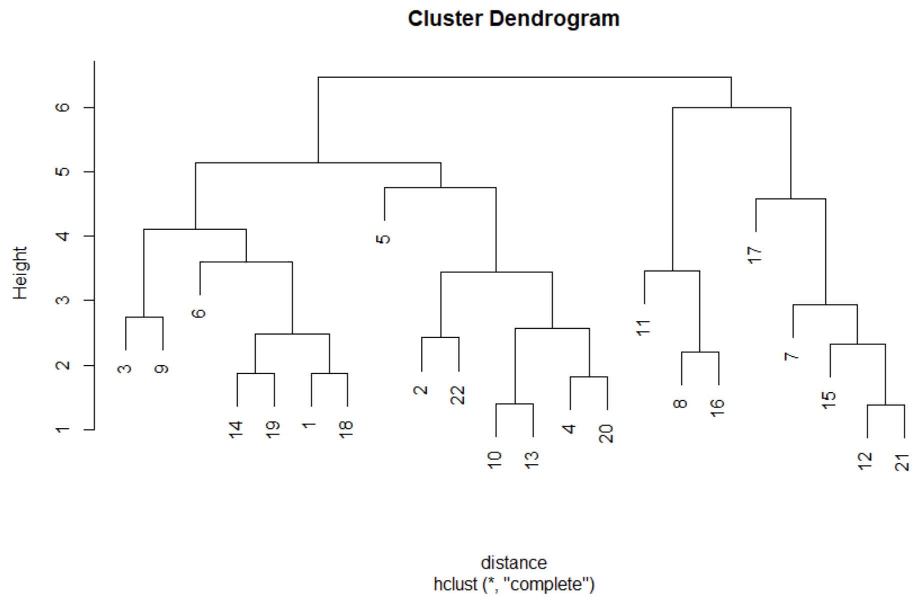
```
      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21
2  3.10
3  3.68 4.92
4  2.46 2.16 4.11
5  4.12 3.85 4.47 4.13
6  3.61 4.22 2.99 3.20 4.60
7  3.90 3.45 4.22 3.97 4.60 3.35
8  2.74 3.89 4.99 3.69 5.16 4.91 4.36
9  3.25 3.96 2.75 3.75 4.49 3.73 2.80 3.59
10 3.10 2.71 3.93 1.49 4.05 3.83 4.51 3.67 3.57
11 3.49 4.79 5.90 4.86 6.46 6.00 6.00 3.46 5.18 5.08
12 3.22 2.43 4.03 3.50 3.60 3.74 1.66 4.06 2.74 3.94 5.21
13 3.96 3.43 4.39 2.58 4.76 4.55 5.01 4.14 3.66 1.41 5.31 4.50
14 2.11 4.32 2.74 3.23 4.82 3.47 4.91 4.34 3.82 3.61 4.32 4.34 4.39
15 2.59 2.50 5.16 3.19 4.26 4.07 2.93 3.85 4.11 4.26 4.74 2.33 5.10 4.24
16 4.03 4.84 5.26 4.97 5.82 5.84 5.04 2.20 3.63 4.53 3.43 4.62 4.41 5.17 5.18
17 4.40 3.62 6.36 4.89 5.63 6.10 4.58 5.43 4.90 5.48 4.75 3.50 5.61 5.56 3.40 5.56
18 1.88 2.90 2.72 2.65 4.34 2.85 2.95 3.24 2.43 3.07 3.95 2.45 3.78 2.30 3.00 3.97 4.43
19 2.41 4.63 3.18 3.46 5.13 2.58 4.52 4.11 4.11 4.13 4.52 4.41 5.01 1.88 4.03 5.23 6.09 2.47
20 3.17 3.00 3.73 1.82 4.39 2.91 3.54 4.09 2.95 2.05 5.35 3.43 2.23 3.74 3.78 4.82 4.87 2.92 3.90
21 3.45 2.32 5.09 3.88 3.64 4.63 2.68 3.98 3.74 4.36 4.88 1.38 4.94 4.93 2.10 4.57 3.10 3.19 4.97 4.15
22 2.51 2.42 4.11 2.58 3.77 4.03 4.00 3.24 3.21 2.56 3.44 3.00 2.74 3.51 3.35 3.46 3.63 2.55 3.97 2.62 3.01
```

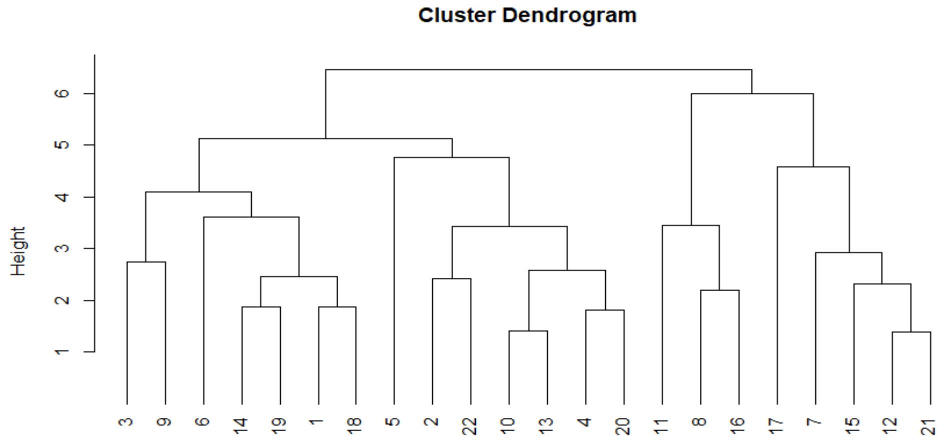
# 10.3.2.4 Examples

```
# Hierarchical clustering
mydata.hclust = hclust(distance)
plot(mydata.hclust)
plot(mydata.hclust,labels=mydata$Company,main='Default from hclust')
plot(mydata.hclust,hang=-1)
```

Calculate Hierarchical clustering



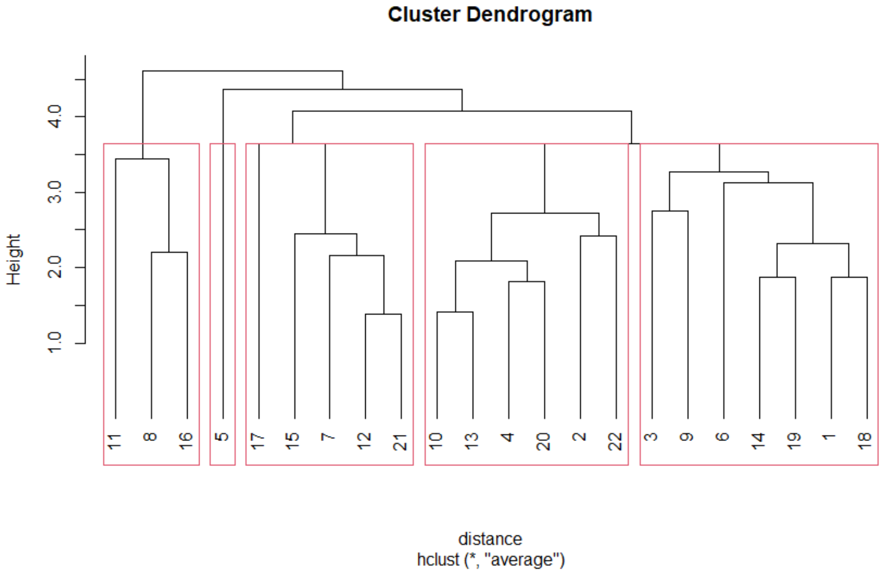




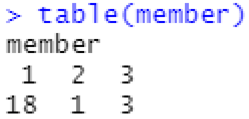
# 10.3.2.4 Examples

**# Hierarchical clustering**
```
mydata.hclust<-hclust(distance,method="average")
plot(mydata.hclust,hang=-1)
rect.hclust(mydata.hclust, k=5)
```

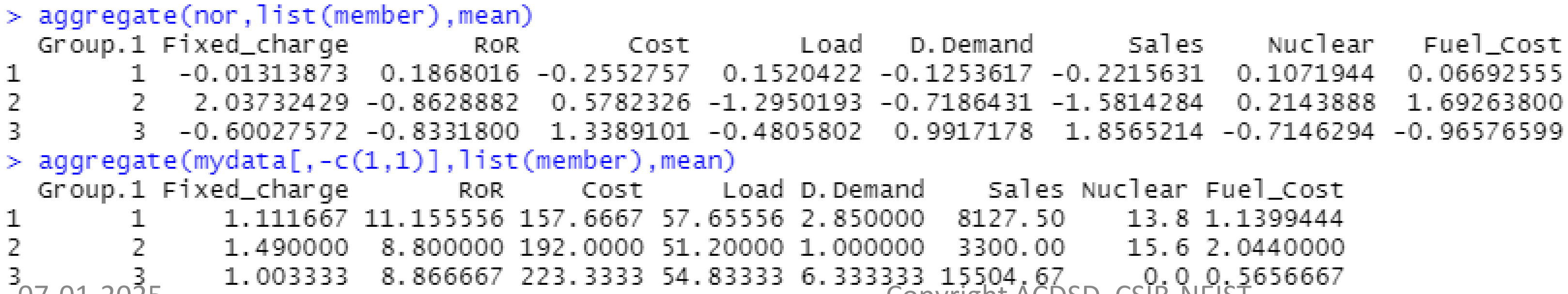
Hierarchical clustering



**# Identify clusters from tree**
```
member = cutree(mydata.hclust,3)
table(member)
```


```
> table(member)
member
 1  2  3
18  1  3
```

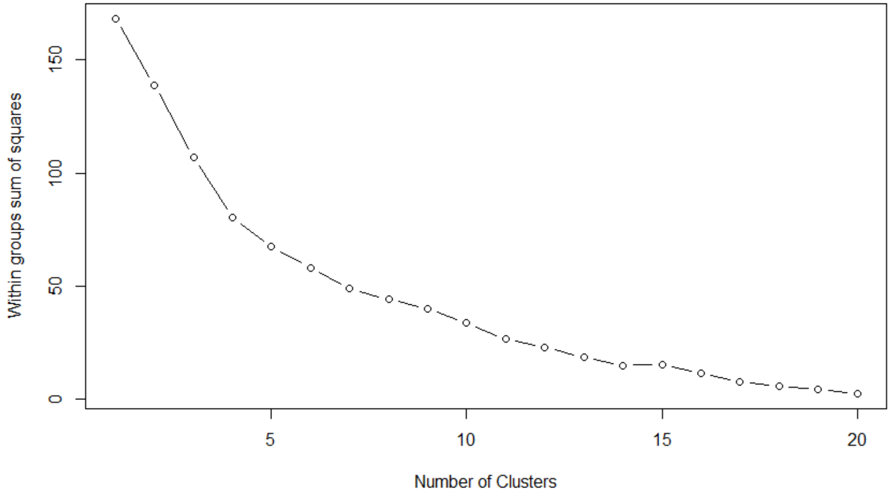
**# Characterizing clusters**
```
aggregate(nor,list(member),mean)
aggregate(mydata[,-c(1,1)],list(member),mean)
```


```
> aggregate(nor,list(member),mean)
  Group.1 Fixed_charge        RoR       Cost       Load   D.Demand      Sales    Nuclear   Fuel_Cost
1       1  -0.01313873  0.1868016 -0.2552757  0.1520422 -0.1253617 -0.2215631  0.1071944  0.06692555
2       2   2.03732429 -0.8628882  0.5782326 -1.2950193 -0.7186431 -1.5814284  0.2143888  1.69263800
3       3  -0.60027572 -0.8331800  1.3389101 -0.4805802  0.9917178  1.8565214 -0.7146294 -0.96576599
> aggregate(mydata[,-c(1,1)],list(member),mean)
  Group.1 Fixed_charge       RoR     Cost     Load D.Demand    Sales Nuclear Fuel_Cost
1       1     1.111667 11.155556 157.6667 57.65556 2.850000  8127.50    13.8 1.1399444
2       2     1.490000  8.800000 192.0000 51.20000 1.000000  3300.00    15.6 2.0440000
3       3     1.003333  8.866667 223.3333 54.83333 6.333333 15504.67     0.0 0.5656667
```

# 10.3.2.4 Examples

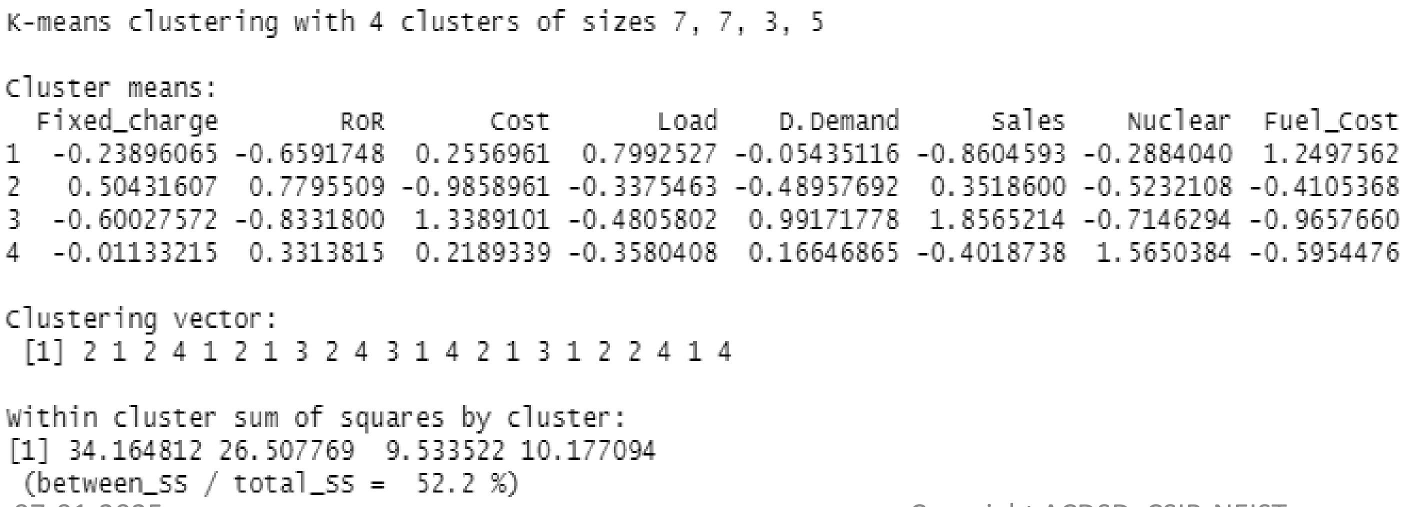
```
# Scree plot with sum of square
wss <- (nrow(nor)-1)*sum(apply(nor,2,var))
for (i in 2:20)
wss[i] <- sum(kmeans(nor, centers=i)$withinss)
plot(1:20, wss, type="b", xlab="Number of Clusters",
ylab="Within groups sum of squares")
```

WSS to find the variability, lesser clusters higher variability



```
# K-means clustering
# K-means clustering
set.seed(123) kc<-kmeans(nor,4)
kc
```

K-Means clustering with k=4

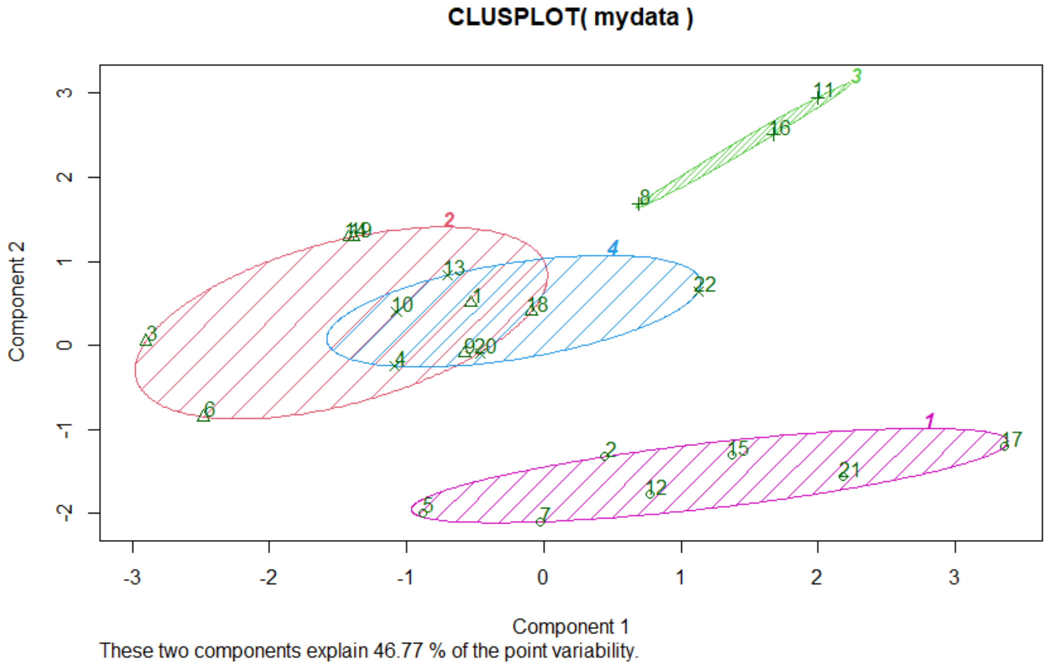


```
K-means clustering with 4 clusters of sizes 7, 7, 3, 5

Cluster means:
  Fixed_charge        RoR       Cost       Load    D.Demand      Sales    Nuclear  Fuel_Cost
1  -0.23896065 -0.6591748  0.2556961  0.7992527 -0.05435116 -0.8604593 -0.2884040  1.2497562
2   0.50431607  0.7795509 -0.9858961 -0.3375463 -0.48957692  0.3518600 -0.5232108 -0.4105368
3  -0.60027572 -0.8331800  1.3389101 -0.4805802  0.99171778  1.8565214 -0.7146294 -0.9657660
4  -0.01133215  0.3313815  0.2189339 -0.3580408  0.16646865 -0.4018738  1.5650384 -0.5954476

Clustering vector:
 [1] 2 1 2 4 1 2 1 3 2 4 3 1 4 2 1 3 1 2 2 4 1 4

Within cluster sum of squares by cluster:
[1] 34.164812 26.507769  9.533522 10.177094
 (between_SS / total_SS =  52.2 %)
```

# 10.3.2.4 Examples

```
# Visualization

clusplot(mydata,
        kc$cluster,
        color = T,
        shade = T,
        labels = 2,
        lines = 0)
```

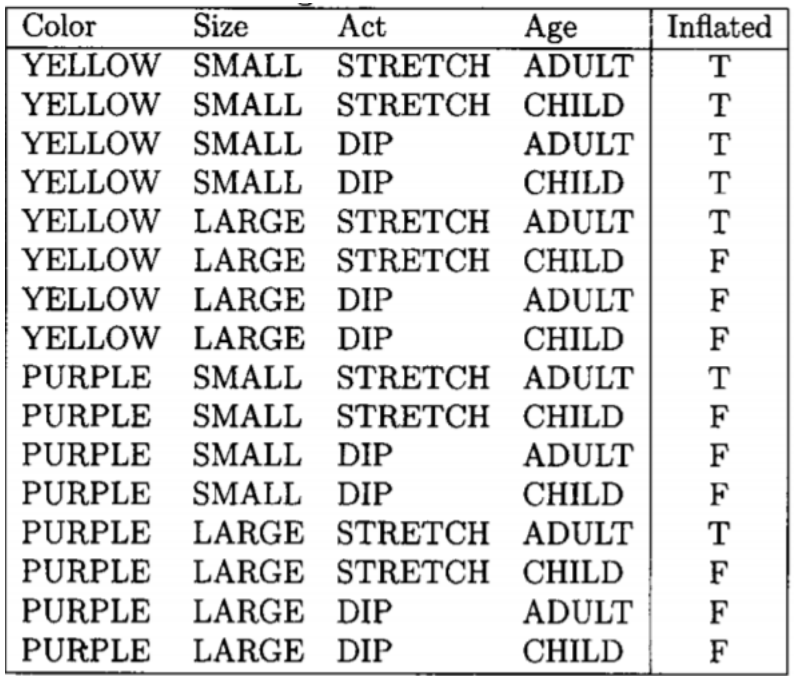
clusplot uses PCA to reduce the dimensions into a single component



These two components explain 46.77 % of the point variability.

# 10.4 Exercises

1. Find a decision tree of depth 2 that attains zero training error.
2. Consider the dataset given along. Compute the parameters of Naïve Bayes classifier for predicting **inflated.**



| Color | Size | Act | Age | Inflated |
|---|---|---|---|---|
| YELLOW | SMALL | STRETCH | ADULT | T |
| YELLOW | SMALL | STRETCH | CHILD | T |
| YELLOW | SMALL | DIP | ADULT | T |
| YELLOW | SMALL | DIP | CHILD | T |
| YELLOW | LARGE | STRETCH | ADULT | T |
| YELLOW | LARGE | STRETCH | CHILD | F |
| YELLOW | LARGE | DIP | ADULT | F |
| YELLOW | LARGE | DIP | CHILD | F |
| PURPLE | SMALL | STRETCH | ADULT | T |
| PURPLE | SMALL | STRETCH | CHILD | F |
| PURPLE | SMALL | DIP | ADULT | F |
| PURPLE | SMALL | DIP | CHILD | F |
| PURPLE | LARGE | STRETCH | ADULT | T |
| PURPLE | LARGE | STRETCH | CHILD | F |
| PURPLE | LARGE | DIP | ADULT | F |
| PURPLE | LARGE | DIP | CHILD | F |

3. In context of logistic regression, define prediction and loss functions.
4. Discuss the application of Neural network which is used for learning to steer an autonomous vehicle.

# 10.5 References

## The Topics Covered in This Section

10.5.1 Books
10.5.2 Video Lectures

# 10.5.1 Books

- Tang, P.N., Steibach, M., and Kumar, V. (2016). *Introduction to Data Mining*. Pearson.

➢ Provides a theoretical and practical coverage of all data mining techniques with examples. Covers topics like include classification, association analysis, clustering, anomaly detection ★ ★★ ★

- Mitchell, T.M. (1997). *Machine Learning*. McGraw Hill.

➢ Covers the field of machine learning, which is the study of algorithms that allow computer programs to automatically improve through experience. ★ ★★ ★

- Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer.

➢ This book reflects these developments while providing a grounding in the basic concepts of pattern recognition and machine learning.★ ★★

- Hastie, T., Tibshirani, R. Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.

➢ Describes the important ideas in a variety of fields such as medicine, biology, finance, and marketing in a common conceptual framework. While the approach is statistical, the emphasis is on concepts rather than mathematics. ★ ★★ ★

- James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.

Provides tools for Statistical Learning that are essential for practitioners in science, industry and other fields. Analyses and methods are presented in R. Covers topics like linear regression, classification, resampling methods, shrinkage approaches, tree-based methods, support vector machines, and clustering.★ ★★

# 10.5.1 Books

- Zaki, M.J., and Jr., W.B. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithm.* Cambridge University Press.

➢ Fundamental algorithms in data mining, automated methods to analyze patterns and models for all kinds of data, with applications ranging from scientific discovery to business intelligence and analytics are covered. Provides a broad yet in-depth overview of data mining, integrating related concepts from machine learning and statistics including exploratory data analysis, pattern mining, clustering, and classification. ★ ★★ ★

- Shwartz, S.S., and Davis, S.B. (2014). *Understanding Machine Learning: From Theory to Algorithm.* Cambridge University Press.

➢ Introduces machine learning and provides a theoretical account of the fundamentals underlying machine learning and the mathematical derivations that transform these principles into practical algorithms. ★ ★★ ★

# 10.5.2 Video Lectures

- Machine Learning | Stanford University

https://www.youtube.com/watch?v=PPLop4L2eGk&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN

- Introduction to Machine Learning | University of Toronto

https://www.youtube.com/watch?v=FvAibtlARQ8&list=PL-Mfq5QSs8iS9XqKuApPE1TSlnZblFHF

- Machine Learning for Intelligent Systems | Cornell

https://www.youtube.com/watch?v=MrLPzBxG95I&list=PLl8OlHZGYOQ7bkVbuRthEsaLr7bONzbXS

- Machine Learning Course | Caltech

https://www.youtube.com/watch?v=mbyG85GZ0PI&list=PLD63A284B7615313A

# Thank You