Module I

Introduction to machine Learning :

→ Machine Learning : field of study interested in the development of computer algorithms to transform data → intelligent action

→ It is about predicting the future based on the past.

→ Algorithm enables computers to 'learn' from data

→ Related to many areas such as
- data mining
- statistics etc

→ Traditional programming,

Data → Computer → Output
Program →

Machine Learning

Data → Computer → Program
output →

→ Applications,
- identification of unwanted spam messages in email
- weather forecast & long-term climate change.
- development of self-driving cars & auto-piloting drones
- preventing fraudulent credit card transactions
- prediction of election outcomes
- optimization of energy use in homes and offices

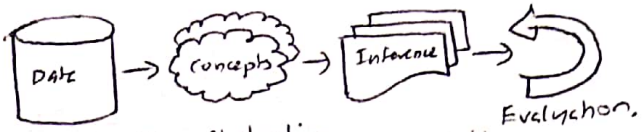
→ Limitations,
- recognize speech and handwriting accurately
- requires massive data to train on.
- High error-susceptibility.

How machines-Learn;

→ A formal definition proposed by computer scientist Tom. M. Mitchell states "A machine learns whenever it is able to utilize an experience, such that its performance improves on similar experiences in the future.

→ Human brains are naturally capable of learning from birth (computers ⇒ explicitly done)

→ The basic learning process is divided into 4 components,



Data storage Abstraction Generalization

(i) Data storage

[utilizes observation, memory and recall to provide a factual basis for further reasoning]

- all learning begins with data
- In human beings, brain uses electrochemical signals of biological cells to store & process observations for future call
- In computers, recall is done using HDD, flash memory and RAM in combination with a CPU

(ii) Abstraction

[involves translation of stored data into broader representations & concepts]

- process which assigns meaning to stored data (raw data comes to have a more abstract meaning)
- computer summarizes stored data using a model (explicit description of patterns within data)
- Training : process of fitting a model to a dataset.
- Different types of models,
  1) mathematical equations
  2) Relational diagrams (trees/graphs)
  3) Logical If/else rules
  4) Clusters (grouping of data)
- Eg:

Observation → Data → model

Distance	Time
4.9m	1s
19.6m	2s

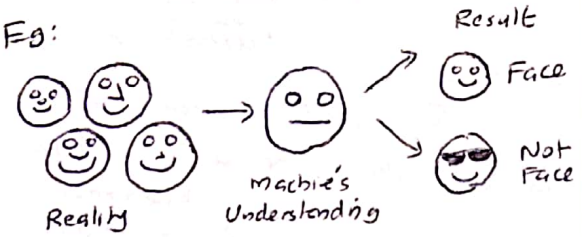
$g = 9.8 m/s^2$

(iii) Generalization

[uses abstracted data to create knowledge & inferences — that drive action in new contexts]

- turning abstract knowledge into a form that can be utilized for future action.
- It is a vague process (difficult to describe)
- involves reduction of hypothetical set in manageable number of important findings

- patterns are limited to only those that will be most relevant to future tasks
- it is not feasible to reduce no. of patterns by examining them one-by-one & ranking them
- so, algorithms employ shortcuts that reduce search space more quickly
- Towards end, algorithm will employ heuristics – (technique to solve problem more quickly)
- Eg:



(iv) **Evaluation**

[provides a feedback mechanism to measure utility of learned knowledge & inform potential improvements]

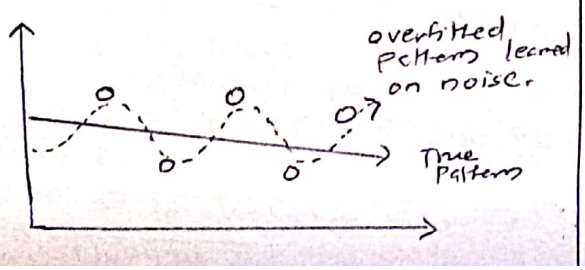
- evaluate/measure the learners success and use this info to inform additional training
- models fails to perfectly generalise due to problem of noise

(noise ⇒ unexplained variations of data)

- causes of noise,
  - → error due to sensors
  - → issue with human subjects (Eg: survey respondents)
  - → data quality problems (missing, corrupted values etc)
  - → complex phenomenas (unsystematic impact to data)
- Overfitting, ~~used to reduce~~ problem trying to model noise

solution to the problem of overfitting ⇒ are specific to particular machine learning approaches

- Eg:



Machine learning in practice (5 steps)

1) Data collection
2) Data exploration & preparation (checking quality of data)
3) model Training (selection of an appropriate algorithm to form a model)
4) model Evaluation (evaluating the accuracy of a model using test data)
5) model Improvement (using advanced strategies & switching model if required)

Types of Input Data.

→ Unit of analysis ⇒ smallest unit from which inference is made

→ Datasets (stores unit of observation and their properties)

- collection of data includes,
  - (i) Examples [instances of unit of observation]
  - (ii) Features [recorded properties of examples]

→ Eg: To build an algorithm to identify spam e-mail;

- Unit of observation ⇒ email messages
- Examples ⇒ specific messages
- Features ⇒ words used in emails

→ Examples and features are commonly gathered in Matrix format

Automobile

year	model	price	miles se	color

examples (row)

features (column)

→ various forms of feature,

1) Numerical (characteristic measured in numbers)
   Eg: no. of black pixels
2) Categorical (attribute consists of a set of categories)
   Eg: Gender.
3) Ordinal (follows an ordered list - special case of categorical)
   Eg: cloth size (S, M, L, XL)

Eg 2 – measurement of customer satisfaction

[not at all happy → very happy]

Types of machine Learning Algorithms

→ Machine Learning algorithms are divided into categories according to their purpose

→ They are,

1) **Predictive model**

- involve the prediction of one value using other values in the dataset.
- attempts to discover & model the relationship b/w target feature (feature being predicted) and other features
- Supervised Learning algo. used [attempts to optimize a model - to find combo of feature values that result in target output]

⇒ we have input variables (X), and an output variable (Y), and we use an algorithm to learn mapping function

$$Y = f(X)$$

⇒ Classification : predicting the category an example belongs to

⇒ Using classifier, we can predict, (whether an email is spam, whether a person has cancer etc)

2) **Descriptive model**

- no single feature is more important than the other.
- Here, there is no target to learn.
- Descriptive modelling tasks include,
  - (i) Pattern Discovery [identify useful associations within data] Eg: Market-Basket Analysis
  - (ii) clustering [dividing datasets to homogeneous groups] Eg: ads can be tailored for particular audience
- Unsupervised Learning algo. used [→ attempts to model the underlying structure in data, inorder to learn more about data]

⇒ we have input data (X), and no corresponding output variable (Y)

⇒ Eg: Clustering (K-means)
Association rules (Apriori algorithm)

Note ⇒ supervised learning algo ⇒
- Decision Tree — classification
- Linear Regression — Numeric prediction

Exploring & Understanding data

Numeric variables

1) **Measuring Central Tendency** - Mean, Median and Mode

→ A class of statistics used to identify a value that falls in the middle of a set of data

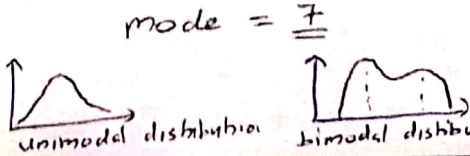
→ **Mean** :
- also known as average
- Def: $\left[\frac{\text{sum of all values}}{\text{number of values}}\right]$
- Eg: mean income of 3 people having salaries 10,000, 5000, 15000

$$\Rightarrow \frac{10000 + 5000 + 15000}{3}$$

$$\Rightarrow 10000$$

- In R, we use mean() function
- not always appropriate.

→ **Median** :
- Def: value that occurs halfway through an ordered list of values
- Eg: median of the salaries
  Ascending order (5000, 10,000, 15,000)
  ⇒ 10000
- To find median, first arrange the data in ascending / descending order
- If no. of observations (n) is odd, median value is at position $\Rightarrow \left(\frac{n+1}{2}\right)^{th}$ item
- If (n) is even, median value $\Rightarrow \left(\frac{\left(\frac{n}{2}\right)^{th} + \left(\frac{n+1}{2}\right)^{th}}{2}\right)$
- In R, we use median() function

→ **Mode** :
- Def: number in a set that appears most frequently.
- A set can have more than one mode
  - → single mode ⇒ Unimodal
  - → Two modes ⇒ Bimodal
- Eg: 12, 8, 7, 15, 7
  mode = 7
- 

2) Measuring Spread – Quartiles and 5-number summary

- used to measure Diversity
- denotes how highly / loosely the values are spaced.
- specifies whether most values are like or unlike mean & median.
- **5 number summary**
  - A set of five statistics that roughly depict the spread of a feature's values
  - All 5 are included in the output of summary() fn.

1) Minimum (Min.)
2) First quartile, or Q1 (1st Qu.)
3) median, or Q2 (median)
4) Third quartile, or Q2 (3rd Qu.)
5) Maximum (max)

- R provides min() & max() fn to calculate minimum and maximum
  - ⇒ range() ⇒ returns both min & max.
  - ⇒ quantile() ⇒ returns 5-number summary.

0%	25%	50%	75%	100%
x	y	z	a	b

  - ⇒ IQR() ⇒ returns Inter-Quartile Range ($Q_3 - Q_1$)

- Quartiles
  - ⇒ special case of quantiles (a type of statistics)
  - ⇒ quantiles: numbers that divide data into equally sized quantiles
  - ⇒ commonly used quantiles,
    a) Tertiles (3-parts)
    b) Quintiles (5-parts)
    c) Deciles (10-parts)
    d) Percentiles (100-parts)
  - ⇒ quartiles divide dataset into 4 portions
  - ⇒ Q_1 = middle number b/w (min) and median.
    Q_2 = median
    Q_3 = middle value b/w median and (max).

Eg 5, 7, 4, 4, 6, 2, 8

Ascending order ⇒ 2, 4, 4, 5, 6, 7, 8

Cut list into quarters: 2, 4, 4, 5, 6, 7, 8 (↑ Q_1, ↑ Q_2, ↑ Q_3)

Eg 2 3, 5, 7, 8, (12), 13, 14, 18, 21

lower half | median | upper half

$Q_1 = \frac{5+7}{2} = \frac{12}{2} = 6$ //

$Q_3 = \frac{14+18}{2} = \frac{32}{2} = 16$ //

$Q_2 = 12$ //

Variance and Std. Deviation

- Variance [average of squared differences between each value and the mean value]

$$Var(X) = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

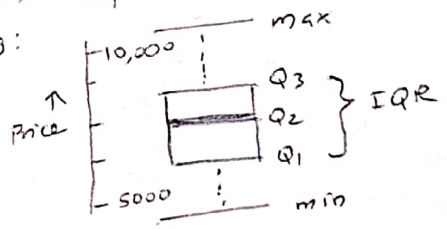
- Standard deviation [square root of variance denoted by sigma]

$$\text{Std Dev}(X) = \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

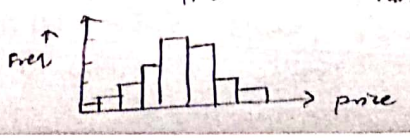
- var() and sd() functions in R can be used.

Visualising Numeric variables

1) Boxplot:
   - boxplot() used in R
   - depicts 5 number summary using horizontal lines & dots.
   - horizontal lines forming the box represents Q_1, Q_2, & Q_3
   - Eg:



2) Histograms
   - hist() used in R
   - divides values into predefined number of portions / bins
   - composed of series of bars.
     - ⇒ Height ⇒ freq. of values
     - ⇒ vertical line ⇒ start & end points of range of values.



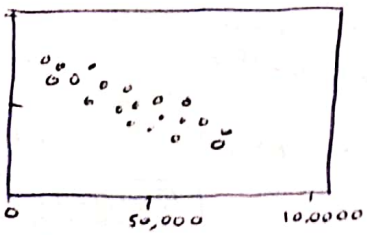
Understanding numeric data

1) Uniform Distribution
- all values are equally likely to occur
- Eg: fair 6-sided die rolled on a floor.
- easy to detect with a histogram. (bars have same height)
- Histogram

2) Normal Distribution
- some values are more likely to occur than others.
- Eg: Distribution of height/weight in general population.
- values are less likely to occur on both sides of central area.
- 
- Skewness ⇒ a) Right skew 
  b) No skew 
  c) Left skew 

Visualization Relationships

Scatter plot
- visualizes a bivariate relationship (relnship b/w 2 variables)
- dots are drawn on a coordinate plane (values of one feature ⇒ x, values of other feature = y)
- plot() fn. used in R.
- Negative association ⇒ pattern of dots sloping downward
  Positive association ⇒ pattern of dots sloping upward.
- plot(x = , y = , main = "Heading" xlab = " " ylab = " ")
- Eg:

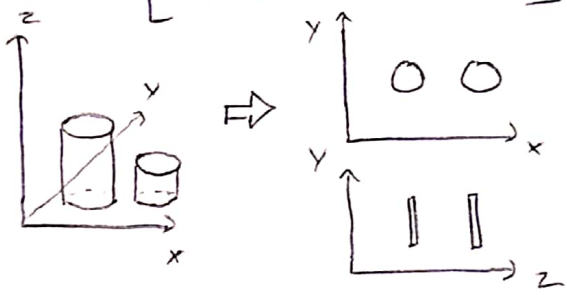


Categorical / Nominal Variables
- allows to assign categories
- Eg: used car dataset, 3 categorical variables [model, color, transmission]
- examined using table (one way table)
- table() in R.

Dimensionality Reduction
→ process of reducing the number of random variables, by obtaining a set of principal variables.
→ Divided into,
a) Feature selection
- find the subset of the original set of variables
- involves 3 ways,
  (i) Filter
  (ii) Wrapper
  (iii) Embedded

b) Feature extraction
- reduces data in a high dimension space → low dimension space

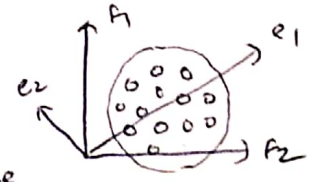
Eg:



→ Advantages,
- helps in data compression (ie, reduced storage space)
- reduces computation time
- removes redundant features.

Disadvantages
- some data loss
- linear correlation is undesirable in some cases.
- mean & covariance not enough to define datasets.

Method (Principal Component Analysis)
→ introduced by Karl Pearson
→ here variance of data in lower dimensional space ⇒ maximum
→ steps,



1) construct the covariance matrix
2) compute the eigenvalues of the matrix.
3) Largest eigenvalues are used to reconstruct a large fraction of variance of original data

Module II

Lazy Learning

- Lazy Learning methods simply store the data.
- generalizing beyond these data is postponed until an explicit request is made.
- Eg: Nearest Neighbor classifiers
  - defined by their characteristic of classifying unlabelled examples by assigning them the class of similar labelled examples
- Nearest Neighbor methods are simple & extremely powerful
- Applications,
  - Predicting whether a person will enjoy a music recommendation
  - facial recognition
  - identifying patterns in genetic data.

The K-NN Algorithm

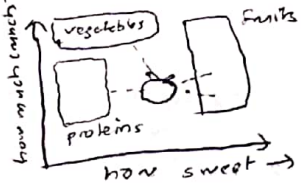
- stands for K-Nearest Neighbors algorithm (K-NN)
- simplest & widely used machine learning algorithm.
- strengths & weaknesses,

Strengths	Weaknesses
• simple & effective • makes no assumptions regarding data distribution • Fast training phase	• doesn't produce a model. • selection of an appropriate 'K' • slow classification phase. • Nominal features & missing data need additional processing

- k-NN algorithm uses information about an examples 'k' nearest-neighbours to classify unlabelled examples
- k ⇒ any number of nearest neighbors can be used.
- after choosing k, algorithm requires a training dataset, that have been classified into several categories (labeled by a nominal variable)
- then for each unlabeled record, k-NN identifies k records in the training data that are the "nearest" in similarity.
- The unlabeled test instance is assigned the class of majority of k neighbors
- 2 features of each ingredient is only taken (simplify, on 2D graph)
- k-NN algorithm treats the features as coordinates in a multidimensional feature space.

Example of K-NN Algorithm

Qsln:> Is tomato a fruit or vegetable?



Measuring similarity with distance

- Locating tomato's nearest neighbors requires a distance function/formula
- many ways to calculate distance
- k-NN uses Euclidean distance. (distance measured by a ruler to connect two points)
- Euclidean distance,

  p & q ⇒ examples to be compared, having n features

  p1 ⇒ value of first feature of p,

  q1 ⇒ value of first feature of q

$$dist(P,q) = \sqrt{(P_1-q_1)^2+(P_2-q_2)^2+\dots+(P_n-q_n)^2}$$

- we have the table, of ingredient and features (sweetness & crunchiness)

	Ingredient	Sweetness	Crunchiness
F	grape	8	5
V	green bean	3	7
P	nuts	3	6
F	orange	7	3

To calculate distance to tomato (6,4) from,

a) grape ⇒ $\sqrt{(6-8)^2+(4-5)^2} = 2.2$

b) green bean ⇒ $\sqrt{(6-3)^2+(4-7)^2} = 4.2$

c) nuts ⇒ $\sqrt{(6-3)^2+(4-6)^2} = 3.6$

d) orange ⇒ $\sqrt{(6-7)^2+(4-3)^2} = 1.4$

1-NN classification

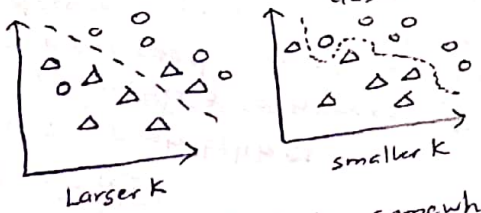
- To classify the tomato as a vegetable, protein or fruit, we'll begin by assigning the tomato to a single nearest neighbor
- Here orange is the nearest neighbor (distance = 1.4)
- As orange = a fruit, 1-NN would classify tomato also as a fruit

3-NN classification

- k=3 and performs vote against 3 nearest neighbours (orange, grape and nuts)
- Majority class ⇒ fruit (2 out of 3 votes)
- So tomato is again classified as a fruit

Choosing an appropriate k

- The decision of how many neighbors to use for k-NN ⇒ determines how well model will generalize future data.
- Bias variance Tradeoff: balance b/w overfitting & underfitting of training data
  - Overfitting ⇒ occurs when algo. captures noise of data
  - Underfitting ⇒ occurs when algo. neither model training data nor generalize to new data
- Very large k value ⇒ model always predict majority class
- single nearest neighbor ⇒ noisy data/ outliers influence classification



- Best value of k ⇒ somewhere b/w two extremes

Common practice, $k = \sqrt{\text{No of training examples}}$

Eg: for 15 example ingredients k=4, [since $\sqrt{15} = 3.87$]

Preparing data for use with K-NN

- one solution ⇒ rescale features by shrinking / expanding their range such that one contributes relatively equally to distance formula.
- Eg: if sweetness & crunchiness are both measured 1-10 scale, we would like to measure spiciness on (1-10) scale.
- several methods for scaling include,

1) Min-Max Normalization

- traditional method for rescaling
- transforms a feature such that all values fall b/w (0-1)
- Formula,

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

2) Z-score standardization

- $X_{new} = \frac{X - \mu}{\sigma}$ where $\mu = \text{mean}(X)$, $\sigma = \text{stdDev}(X)$
- resulting value is called z-score (falls on an unbound range of negative & posibe numbers)
- no predefined maximum/minimum

3) Dummy coding

- Euclidean distance not defined for nominal data.
- To calculate distance b/w nominal features ⇒ convert them to numeric format
- Eg: Dummy coding for gender,

$$\text{male} = \begin{cases} 1 & \text{if } x = \text{male} \\ 0 & \text{otherwise} \end{cases}$$

- Dummy coding of two category ⇒ results in a single new feature
- No need to construct a separate feature for female. (since two are exclusive mutually)
- for n category nominal feature ⇒ dummy code = (n-1) levels of feature.
- for temperature,

$$\text{hot} = \begin{cases} 1 & \text{if } x = \text{hot} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{medium} = \begin{cases} 1 & \text{if } x = \text{medium} \\ 0 & \text{otherwise} \end{cases}$$

cold if both hot & medium = 0

- distance b/w dummy coded feature ⇒ [0-1] (no additional transformation required) (values fall on same scale)

Why k-NN algo. is lazy

- here no abstraction occurs.
- abstraction and generalizations are skipped altogether ⇒ undermines def. of learning
- process of making predictions tends to be relatively slow
- Instance based / Rote learning.

Probabilistic Learning

Eg:- meteorologist says there is "70%" chance of rain in a weather forecast report (7/10)

- Use of past ~~data~~ events' data to predict future events

Probabilistic classifier

↳ a classifier that is able to predict, given a set of classes in probability distribution, a sample input

Naive Bayes

↳ uses probabilities
↳ mathematician Thomas Bayes - developed principles to describe probability of events
↳ these principles formed the foundation for Bayesian methods
↳ Probability (b/w 0 and 1, ie, 0% to 100%)

lower probability ⇒ event is to occur less likely
P(0) ⇒ definitely not occur
P(1) ⇒ event will occur with 100% certainity.

↳ Classifiers based on Bayesian methods ⇒ calculate observed probability from training data

Later classifier applied to unlabelled data ⇒ to predict most likely class

Bayesian classifiers

↳ estimated likelihood / or a potential outcome should be based on evidence across multiple trials

↳ Eg: Trial ⇒ coin flip
Event ⇒ Result = Head.

Understanding Probability

↳ Probability = (No. of trials in which event occured / Total no. of events)

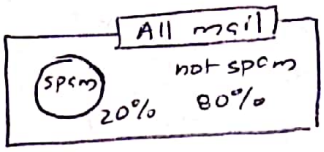
↳ Eg: if it rained 3 out of 10 days with similar conditions as today,
Probality of rain = 3/10 = 30%

↳ P(A): probability of event (A)
P(Rain) = 0.30

↳ sum of all possible outcomes is equal to 1

↳ complement of A = A^c / A'
P(¬A) ⇒ probability of A not occurs

P(spam) = 0.20
P(¬spam) = 0.80

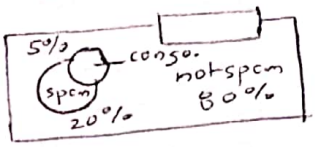


↳ ↑ event and their compliments

Joint Probability

↳ If certain events occur with event of interest, we will be able to make predictions.

↳ consider a second event ⇒ [an email - messege contains word "congratulations"]

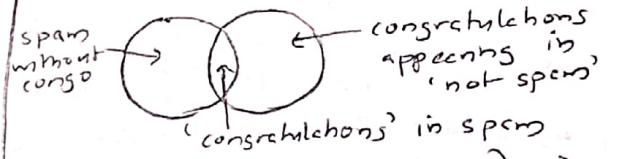


Not all spam messeges contain word congratulations. & not every email with word "congratulations" is spam.

↳ Probability of event Y occuring at the same time event X occurs

P(X ∩ Y) or P(X, Y)

↳ Venn Diagram (By John Venn)



P(spam ∩ congratulations) ⇒ how probability of one event is related to probability of other.

↳ If independent events,

P(A ∩ B) = P(A) * P(B)

P(A) = 0.2 ; P(B) = 0.05
∴ P(A ∩ B) = 0.05 × 20 = 0.01

↳ If dependent events,
- using predictive modelling
- P(spam) and P(congratulations) are likely to be highly dependent ⇒ calculation above may be incorrect
- more careful formulation of relationship b/w two events required

⇒ use of Advanced Bayesian methods

Conditional Probability – with Bayes Theorem

- The relationships b/w dependent events can be described using Bayes theorem.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- Conditional probability of event A is the probability that event will occur, given B is already occured.

[P(A) is dependent on what happened with even B]

- Bayes theorem tells that estimate of P(A|B) should be based on, P(A∩B) & P(B)

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \Rightarrow P(A \cap B) = P(A|B) * P(B)$$

Also $P(A \cap B) = P(B \cap A)$

Substituting,

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)\,P(A)}{P(B)}$$

- Prior probability ⇒ probability of an even computed before collection of new data

Eg: P(spam) = 20%

Likelihood ⇒ P(congo | spam)

Marginal Likelihood ⇒ P(congratulations)

Posterior Probability ⇒ P(spam | congo)

$$P(spam|congo) = \frac{P(congo|spam)\,P(spam)}{P(congo)}$$

if P(congo|spam) = 4/20,

Then $P(spam \cap congo) = (4/20) * (\frac{20}{100})$

$= 0.04$

To compute posterior probability,

$P(spam|congo)$

$\Rightarrow (4/20) * \frac{(20/100)}{(5/100)} = 0.80$

⇒ Probability is 80% that a message is spam, given that it contains word congratulations

The Naive Bayes algorithm

- simple method to apply Bayes theorem to classification problems
- strength & weaknesses,

Strength	Weakness
• simple, fast & effective • works well with noisy & missing data • Requires only few training examples • easy to obtain estimated probability.	• sometimes rely on faulty assumptions of features • not ideal for datasets with many numeric features • Estimated probabilities are less reliable than predicted classes

- it makes some "naive" assumptions about data. (assumes all features are equally important & independent)
- assumptions rarely true in most real world applications.
- To identify spam by monitoring email messages
  - sender maybe an important indicator of spam than text message
  - words are not independent etc.
  - other words like 'earn' & 'money' 'subscribe' will be here.

Classification with naive Bayes

- Suppose a message contains — congo & money, but no earn/subscribe
- using bayes', difficult to solve
- Easier in Naive Bayes' since it assumes independence among events

[cross conditional independence ⇒ conditioned on the same class value]

$\Rightarrow P(A \cap B) = P(A) * P(B)$

$\Rightarrow P(spam | w_1 \cap \neg w_2 \cap \neg w_3 \cap w_4) \propto P(w_1|spam)\,P(\neg w_2|spam)\,P(\neg w_3|spam)\,P(\neg w_4|spam)\,P(spam)$

8 terms

- Formula

$$P(C_L|F_1 \ldots F_n) = \frac{1}{Z} P(C_L) \prod_{i=1}^{n} P(F_i|C_L)$$

- Laplace estimator : if a message with all 4 terms occur, P(spam) becomes 0. ⇒ so add a small no. to each counts in frequency table to ensure non-zero probability (Typically set to 1)
- Note: To use Naive Bayes, features must be categorical, not numeric
  Solution ⇒ bins

Module III

Decision Trees

→ machine learning models that make complex decisions from sets of simple choices.

→ useful for business strategy.

→ Decision Tree Learners are powerful classifiers, which utilizes a tree structure to model relationships among features and potential outcomes.

→ It uses a structure of branching decisions which guides to a final predicted class value.

→ Types of Nodes;

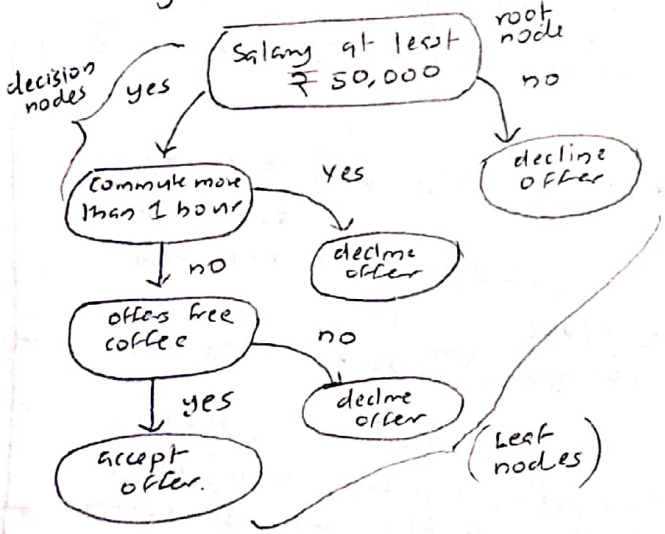
(i) Root Node [tree starts from here]

(ii) Decision Node [choices to be made based on the attributes]

(iii) Branches [potential outcomes of a decision (Yes/No)]

(iv) Leaf Nodes [final decision (also called terminal nodes)]

→ Eg: Should I accept a new job offer?



- more than 2 outcomes is also possible
- leaf nodes ⇒ actions to be taken as the result of series of decisions

- Benefits ⇒ Model is created in a human readable format.
- provides insight into — why the model works or doesn't work well for a particular task (for future business practices)

Uses of decision trees

- Applicant rejection [credit scoring models should mention criteria in a doc free of bias]
- Customer Behaviour [whether satisfied/not — send to marketing agencies]
- Medical Diagnosis [based on symptoms, laboratory measurements etc]

Divide and Conquer

→ also called recursive partitioning

→ Decision trees are built using this heuristic

→ splits data into subsets ⇒ and are then split repeatedly into even smaller subsets and so on.

→ Algorithm stopped when subsets are sufficiently homogenous / stopping criterion has met.

→ In a decision tree,

- at root node ⇒ entire dataset, no splitting done.
- now choose a feature to split.
- (most predictive feature is used)
- examples are partioned into groups according to distinct values

→ Stopping Criteria;

(i) All the examples at the node have same class

(ii) No remaining feature to distinguish among the examples

(iii) Tree has grown to a predefined size limit.

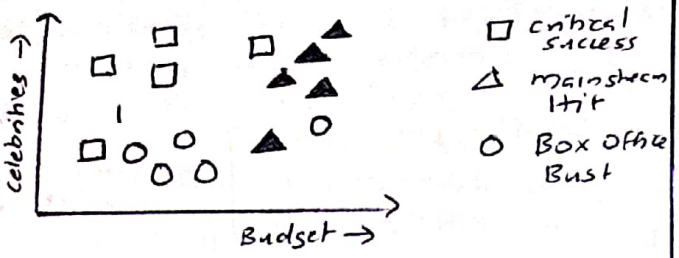
→ Eg: Decision Tree algorithm to predict whether a movie would fall into one of the three categories

- critical success
- mainstream hit
- Box office Bust (flop)

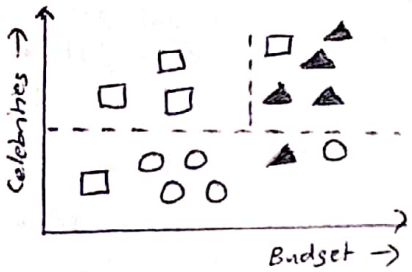
⇒ company's 15 most recent releases are analyzed (success/failure)

⇒ Relationship (budget & no. of major celebrities)

⇒ scatterplot

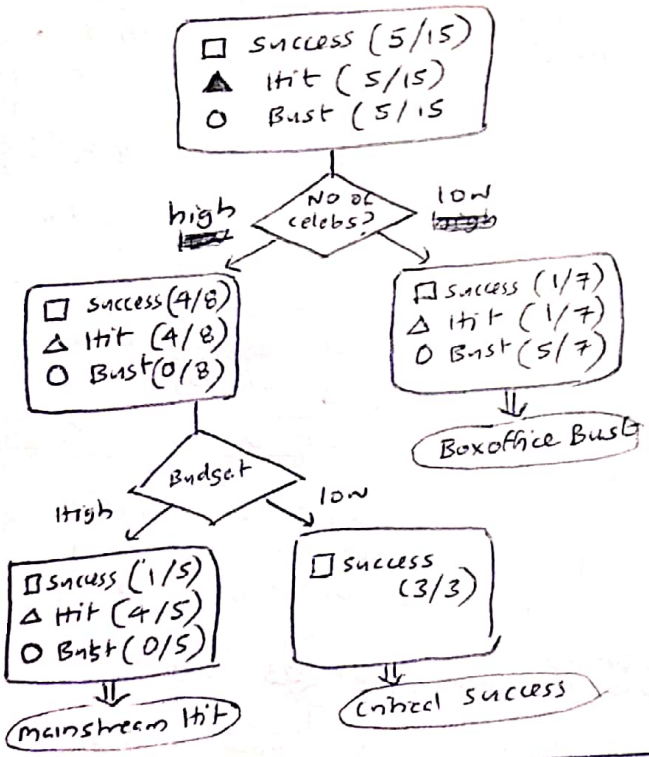


Firstly
- split the dataset using the feature (no. of celebrities) ⇒ create root node.
- second split based on the feature (budget of movie — high/low)



- Data is partitioned into 3 groups.

- Decision Tree : predicting the future success of movies



The C5.0 decision tree algorithm

→ popular algorithm to build decision tree models automatically.

→ well known implementation ⇒ C5.0 Algorithm (developed by Ross Quinlan)
- improved version of prior algo. C4.5, which itself is an improvement over ID3 algorithm
- easier to understand & deploy

Strengths	Weakness
• all purpose classifier (works well on most problems)	• biased towards splits (having a large no. of levels)
• highly automatic learning process (handle ⇒ missing, numerical / nominal)	• easy to overfit / underfit model
• Excludes unimportant features	• Trouble modeling relationships (reliance on axis-parallel splits)
• used on small / large datasets	• small changes in training data ⇒ large changes in decision logic
• results can be interpreted without mathematical background	• Large trees can be difficult to interpret
• more efficient	

Choosing the best split;
(to identify which feature to split upon)

→ Purity ⇒ degree to which a subset contains only a single class

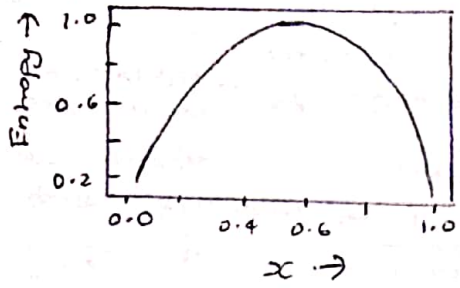
→ Pure subset ⇒ subset composed of only a single class.

→ Entropy ⇒ C5.0 uses entropy as a measurement of purity.
- quantifies randomness / disorder in a set of values.
- sets with high entropy ⇒ very diverse
- decision trees hope to find splits that reduce entropy.
- measured in bits (only 2 classes possible) [0 − 1]
- for n classes, Range (Entropy) ⇒ [0 − log2(n)]
- Entropy, min value ⇒ completely homogeneous
  max value ⇒ diverse as possible
- $$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

S − segment of data
C − no. of class levels
p_i − proportion of values falling into class level i.

- Eg: A partition of data with two classes
  → red (60%) → white (40%)
  → Entropy = −0.60 * log2(0.60) − 0.40 * log2(0.40)
  = 0.9709

- If proportion of examples in one class is x, then proportion (other class) $\Rightarrow (1-x)$

- 

peak entropy value $[x = 0.50]$

ie, 50-50 split creates maximum entropy.

- If one class increasingly dominates the other, entropy reduces to zero

Information Gain

(for a feature F is calculated as the difference b/w the entropy in the segment before split (S1) and partitions resulting from split (S2))

$$\text{Info Gain}(F) = \text{Entropy}(S1) - \text{Entropy}(S2)$$

- after a split, data is divided into more than one partition
- so E(S2) must consider total entropy (weighing each partitions entropy)
- ie, Total entropy,

$$\text{Entropy}(S) = \sum_{i=1}^{n} w_i \, \text{Entropy}(P_i)$$

- ie, total entropy resulting from a split is sum of the entropy of each of n partitions weighted by proportion.

- Higher Information Gain [a feature is better at creating homogeneous groups]
- Information Gain = 0 [no reduction in entropy for splitting on a particular feature]

Pruning the decision tree

→ involves reducing the size of decision tree

→ 2 types

a) **pre-pruning** (early stopping)
- stop growing tree further, before it classifies training set
- condition
  (i) reaching a certain no. of decisions
  (ii) decision node contains only small no. of examples.

b) **post-pruning**
- allows tree to perfectly classify training set and then post prune the tree.
- pruning leaf nodes to reduce the size of the tree to a more appropriate level
- more effective approach (difficult to determine the optimal dept without growing it first)

Regression methods

Regression

→ concerned with specifying the relationship b/w a single dependent variable and one/more numeric independent variables.

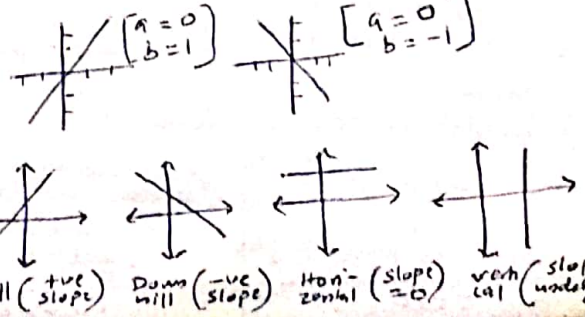
→ simplest form ⇒ follows a straight line

→ model data using a slope-intercept format.

$$y = a + bx$$

y – dependent
x – independent
b – slope
a – intercept

- slope (how much line rises for increase in x)
- intercept (value of y, when $x = 0$)
- positive values ⇒ slope upward
- negative values ⇒ slope downward

→ 

→ Regression analysis - Uses.

- modeling complex relationships among data elements.
- estimating the impact of a treatment on an outcome.
- Quantifying relationship b/w event & response (clinical drug tests, engineering safety tests)
- Identifying patterns (used to forecast future behaviour given known criteria — natural disaster damage, election results)
- statistical hypothesis testing (whether a premise is likely to be true/false in light of the observed data)

→ Basic Linear Regression models

- simple Linear Regression (only single independent variable)
- multiple Linear Regression (Two or more independent variable)

Other Regression methods

- Logistic Regression (model a binary categorical outcome)
- Poisson Regression (models integer count data)
- multinominal Logistic Regression (models a categorical outcome)

Simple Linear Regression

→ relationship b/w a dependent variable and a single independent predictor variable.

→ line equation,

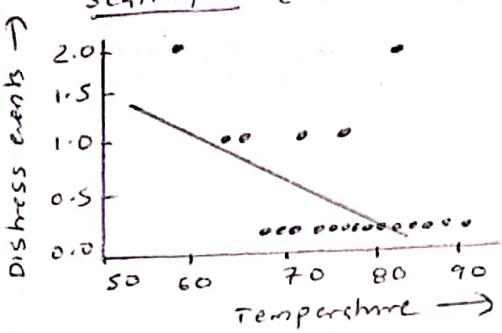
$$y = \alpha + \beta x$$

intercept, α → line crosses the y axis

slope, β → change in y given to increase in x

→ Example; (Rocket Failure)

- regression model demonstrated a link b/w temperature & O-ring failure
- could forecast the change of failure given expected temperature at launch.
- component distress (one of the two types of problem)
  - (i) Erosion [excessive heat burns up the O-ring]
  - (ii) Blow by [hot gases leak through poorly sealed O-ring]
- Scatterplot (23 launches)



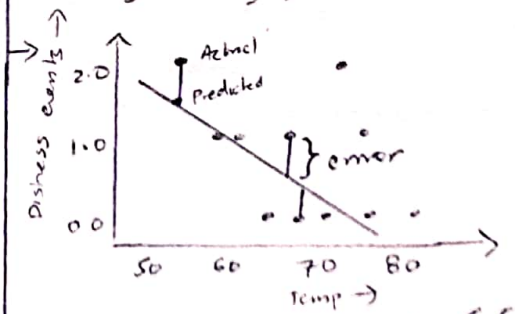
$a = 3.70$
$b = -0.048$

- at 60°F, just one O-ring distress
  70°F, expect around 0.3 failure
- Some interpretations;
  launch at 31°F ⇒ 3 times more risky than at 60°F
  ⇒ 8 times risky than at 70°F

Ordinary Least Squares (OLS) Estimation

→ to determine optimal estimates of α and β

→ slope & intercept chosen so that they minimize the sum of squared errors (vertical distance b/w predicted & actual y value).



→ Goal ⇒ minimizing $\sum (y_i - \hat{y}_i)^2$

→ solution for 'a' depends on value of b.

$$a = \bar{y} - b\bar{x}$$

→ value of b

$$b = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

→ break this eqn to simplify it a bit.

→ denominator for b ⇒ similar to variance of x

$$Var(x) = \frac{\Sigma(x_i - \bar{x})^2}{n}$$

average squared deviation from mean of x

→ numerator for b ⇒ similar to covariance for x & y

$$cov(x,y) = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{n}$$

→ 'n' gets cancelled as we divide covariance by variance fn.

$$\therefore b = \frac{cov(x,y)}{var(x)}$$

- easy to calculate in R language
- x = temp & y = distress event
- use cov() & var() of R. to estimate 'b'
- use mean() to estimate 'a'

Correlations

→ a number that indicates how closely their relationship follows in a straight line.

→ refers to Pearson's correlation coefficient (developed by Karl Pearson)

→ Ranges b/w [-1 & +1]

→ extreme values ⇒ perfectly linear relationship.

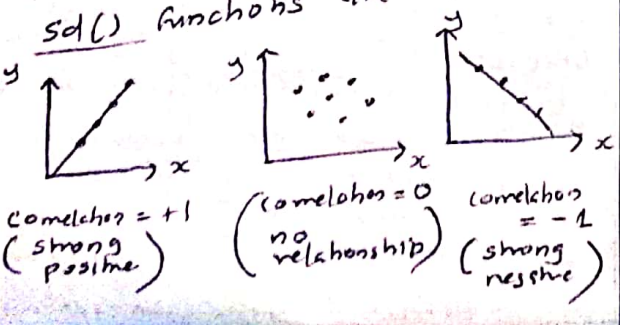
close to zero values ⇒ absence of linear relationship.

→ Pearson's correlation - Formula

$$Corr(x,y) = \frac{cov(x,y)}{\sigma_x \sigma_y}$$

→ Eg: To calculate correlation b/w launch temperature & no. of distress events (-0.51)

→ In R language, cov() and sd() functions are used



correlation = +1 (strong positive) | (correlation = 0 no relationship) | correlation = -1 (strong negative)

→ Interpretation of correlation strength

- weak ⇒ [0.1 to 0.3]
- moderate ⇒ [0.3 to 0.5]
- strong ⇒ above 0.5

for both positive & negative values.

→ context-based ⇒

0.5 for human beings [extremely high]

0.5 for mechanical processes [weak]

Multiple Linear Regression

→ more than one independent variable.

→ Extension of simple linear regression.

→ similar ⇒ goal is to minimize prediction error.

difference ⇒ additional independent variables involved.

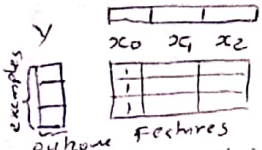
→ General form;

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 \cdots \beta_i x_i + \varepsilon$$

ε (epsilon) ⇒ added since predictions are not perfect.

ε: residual term

- y changes by amount β_i for each unit increase in x_i.
- when all independent variable = 0 ⇒ $y = \alpha$ (intercept) $= \beta_0$ [beta - naught] $= \beta_0 x_0$
- coefficients β_1 β_2 β_3



matrix form

$$Y = \beta X + \varepsilon$$

$$\hat{\beta} = (x^T x)^{-1} x^T Y$$

Strengths

- common approach for modelling numeric data
- can be adapted to any modelling task
- estimates strength & size of relationships among features & outcomes

Weakness

- makes strong assumptions on data.
- model's form ⇒ must be specified in advance
- doesn't handle missing data
- only works with numeric features.
- requires some knowledge of statistics to understand model.

Module IV

Neural Networks

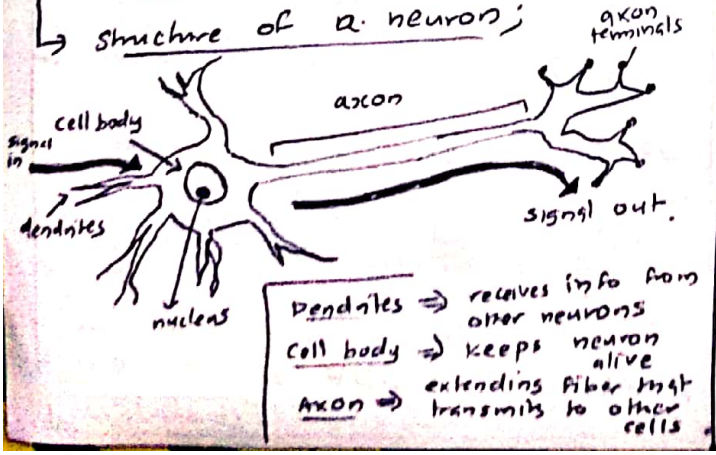
- mimics the structure of animal brains to model arbitrary functions.
- Artificial Neural Network (ANN)
- models the relationship b/w a set of input signals & an output signal.
- it uses a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs.
- brain uses a network of interconnected cells (neurons)
- ANN uses a network of artificial neurons to solve learning problems
- human brain is made up of 85 billion neurons

Applications

- speech and handwriting recognition programs (used in postal mail sorting machines)
- automation of smart devices (self driving cars & self-piloting drones)
- sophisticated models (weather and climate patterns)
- can be also applied to numeric prediction & unsupervised pattern recognition

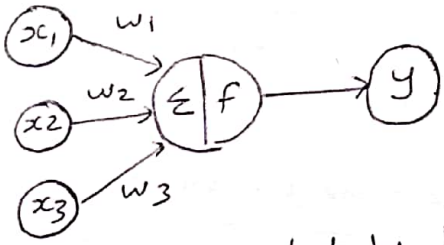
Biological motivation

- ANNs were designed as conceptual models of human brain activity.
- structure of a neuron;



Dendrites ⇒ receives info from other neurons
Cell body ⇒ keeps neuron alive
Axon ⇒ extending fiber that transmits to other cells

- incoming signals are received by dendrites (through bio-chemical process)
- impulses are weighted according to relative importance / freq
- A cell fires only when it reaches a threshold (output signal transmitted via electro-chemical process)
- Electric signal is again processed as chemical signal at axon's terminals (to be passed to the neighboring neurons across a tiny gap [synapse])

- Artificial neuron ⇒ similar to biological model
  - Directed Network Diagram (defines relationship b/w input signals [x variables] & output signals [y variables])
  - each dendrite's signal weighted (as in biological neuron)
  - signal is passed according to activation function (f).
  - 

can be represented by the formula

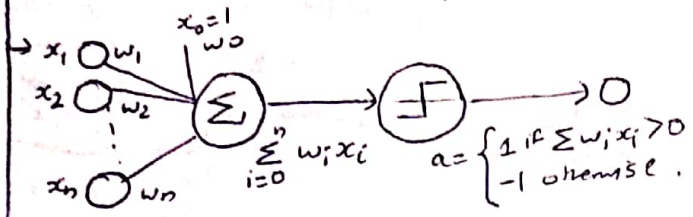
$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

- Characteristics of neural networks
  - (i) An activation function
  - (ii) Network topology (architecture)
  - (iii) Training algorithm.

- Activation Function [transforms combined input signals into a single output signal to be broadcasted]
- Network Topology [describes no. of neurons and no. of layers in which they are connected]
- Training algorithm [specifies how weights are set inorder to inhibit/excite neurons in proportion to input signal]

Perceptrons

→ one type of ANN system is based on unit called Perceptron



→ Takes a vector of real valued inputs, calculates a linear combination of these inputs.

$$\text{Output } o(x_1 \dots x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + \dots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

→ Another form of function;

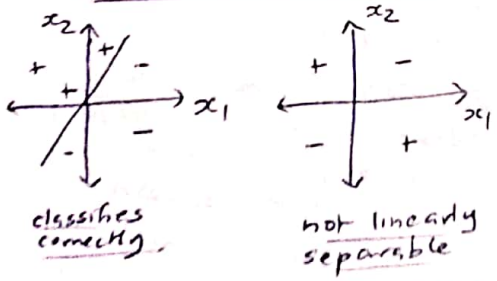
$$o(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$$

$$\text{sgn}(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

→ A single perceptron can be used to represent many boolean functions

- Assume value of 1 (True) & -1 (False)
- To implement AND function ⇒ $w_0 = -0.8$, $w_1 = w_2 = 0.5$
- OR function ⇒ $w_0 = -0.3$, $w_1 = w_2 = 0.5$
- ie can represent all primitive boolean functions (AND, OR, NOR, NAND)
- XOR can't be implemented using single perceptron.

→ decision surface represented by two-input perceptron;



→ Perceptron Training Rule:

- Two well known algorithms are,
  (i) The perceptron rule &
  (ii) The delta rule
- provide the basis for learning networks of many units.

(i) Perceptron Training Rule

(revises weight w_i associated with input x_i)

$$w_i \leftarrow w_i + \Delta w_i$$

where $\Delta w_i = \eta (t - o) x_i$

t = target output
o = output gen. by perceptron.
η = +ve constant (learning rate)

- conditions ⇒
  a) training examples are linearly separable.
  b) sufficiently small η used.

(ii) Delta Rule (for examples not linearly separable)

- key idea ⇒ gradient descent (to find weights that best fits examples)
  (basis of Back Propagation algorithm)
- output for first stage of perceptron,

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

- measure of training error of a weight vector,

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

D = set of training examples
t_d = target output of d
o_d = output of linear unit d.

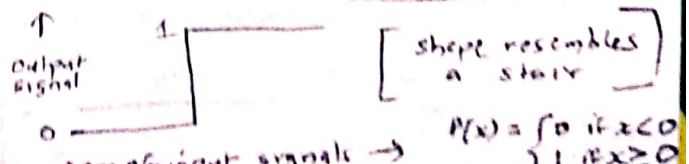
Activation functions

→ mechanism by which the artificial neuron processes incoming information & passes it throughout the network

→ Biological cases;
summing process of total input & determining whether it meets firing threshold

→ In ANN;
Threshold activation function:
results in an output signal only once a specified input threshold has been attained.

→ Also called Unit step activation function



$$f(x) = \begin{cases} 0 \text{ if } x < 0 \\ 1 \text{ if } x \geq 0 \end{cases}$$

- rarely used in ANN

Sigmoid Activation Function

- most commonly used.
- $F(x) = \frac{1}{1+e^{-x}}$ $e \approx 2.72$ (base of natural log)
- step / "S" shape
- output signal is not binary
- values can fallu anywhere in the range [0 to 1].
- differentiable (ie, calculate the derivative [crucial to create efficient ANN algorithms]
- used as default activation fn.

Linear Activation function

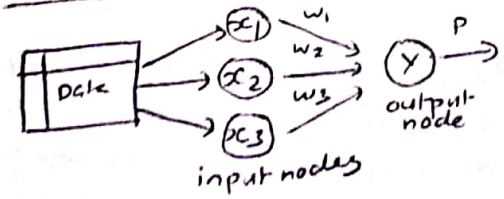
- results in neural network very similar to a linear regression model.

Gaussian Activation function

- results in a model called a Radial Basis Function (RBF) network

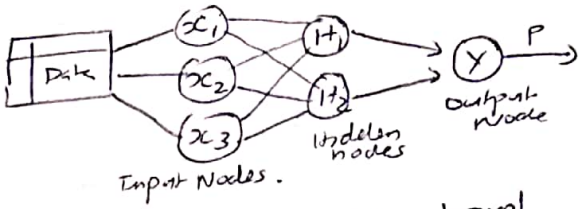
Network Topology

- The patterns and structures of interconnected neurons.
- The three key characteristics:
  1) The number of layers
  2) whether info in network is allowed to travel backward
  3) The number of nodes within each layer of network.
- complex networks are capable of identifying more subtle patterns.



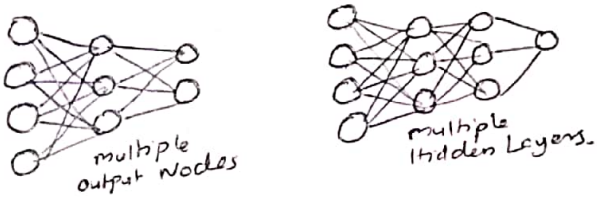
(i) No of Layers

- single Layer Network [→ only one set of connection weights → process incoming data exactly as received → used for basic pattern classification]
- multi Layer Network [→ adds one/more hidden layers → generally fully connected. → hidden layers process input signals prior reaching output node.



(ii) direction of information travel

- feedforward networks [form connection-to-connection until reaching the output layer]
- multiple outcomes can be modelled
- multiple hidden layers can be applied.
- feedback network [allows signals to travel in both directions] Use: weather forecasting



(iii) No. of nodes in each layer.

- No of input nodes ⇒ no. of features in the input data.
- No. of output nodes ⇒ no. of outcomes to be modeled.
- No. of hidden nodes ⇒ user decides.

Cost Function

- The closer the predicted values to corresponding real values the better the model.
- cost function used to measure how close predicted values are to their corresponding real values
- weights of model ⇒ responsible for predicting new values
- Eg: model ⇒ $y = 0.9 + x + 0.1$
  Cost Function measures ⇒ how close $(0.9 + x + 0.1)$ is to actual Y
- we are responsible for finding better weight for our model (0.9 and 0.1) to come up with a lowest cost.

Back Propogation Algorithm

→ ANN must be trained with experience.
→ an algorithm which uses strategy of back propogating errors
→ relatively slow (compared to other ML algos)
→ common in the field of determining
→ iterates through many cycles of two processes (each iteration ⇒ epoch)
→ algorithm cycles until a stopping criterion is reached.
→ 2 process cycle include;
- A forward phase.
- A backward phase.

Forward phase
- neurons are activated in sequence from input layer to output layer.
- applying each neuron's weights and activation function along the way.
- upon reaching final layer ⇒ output signal is produced.

Backward phase
- result of output signal from forward phase compared to the true target value.
- The difference ⇒ results in an error
- error is propogated backwards in the network ⇒ to modify the connection weights (to reduce future errors)

→ uses derivative of each neuron's AF to identify gradient (suggests how deeply error will be reduced / increased)
→ learning rate ⇒ reduction in error by an amount

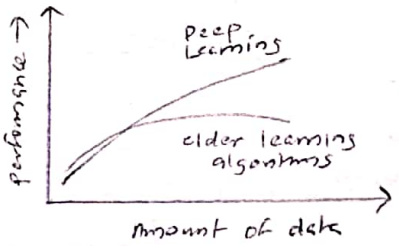
Backpropogation Algorithm
① Initialize Network with random weights.
② For all training cases (examples)
a) Give training inputs to network and calculate output.
b) For all layers (output layer → input layer)
(i) compare network output with correct output
(ii) update weights in current layer.

Strengths	Weaknesses
• adapted to classification or numeric prediction problems • capable of modeling more complex patterns. • makes few assumptions about data's relationship	• computationally intensive • slow to train • prone to overfitting data • difficult to interpret.

Deep Learning

→ branch of machine learning
→ set of algorithms that attempt to model high level abstractions in data.
→ deep network ⇒ many layers b/w input & output.
→ Def: Deep learning is a collection of statistical machine learning techniques used to learn feature hierarchies.
→ 

Deep learning is a specialized type of machine learning.
→ Deep Learning means using neural network with several layers of nodes b/w input & output.
→ Why Important?



Advantages	Disadvantages
• reduces the need for feature engineering. • architecture can be adapted to new problems • best-in-class performance on problems (speech, language, vision etc).	• Requires large amount of data • computationally expensive to train • no strong theoretical foundation. • no heavy to guide topology etc. • what is learned is not easy to comprehend.

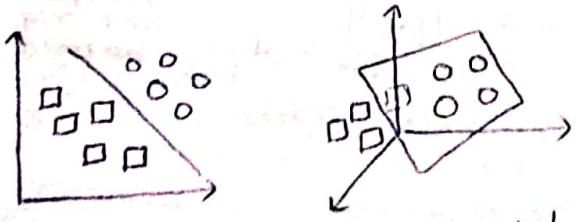
Module V

Support Vector Machines SVM

- A supervised machine learning algorithm
- used for both classification & regression
- can be imagined as a surface that creates a boundary b/w points of data plotted & their feature values.
- Goal ⇒ To create a flat boundary (hyperplane) — to divide space to create fairly homogeneous partitions.
- SVM Learning combines,
  - instance-based nearest neighbor learning.
  - linear regression modeling.
- extremely powerful & can model complex relationships.
- Applications;
  (i) Gene Expression Data Classification [to identify cancer or other genetic diseases]
  (ii) Text categorization [identification of language used in a doc]
  (iii) Detection of rare, but important events [Earthquakes, security breaches etc]

Hyper Planes

- boundary which partitions the data into groups of similar class values.
- SVMs use hyperplanes to partition data into groups of similar class values.
- Eg: seperate groups of circles & squares in 2D & 3D

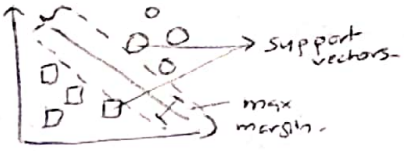


Note ⇒ linearly separable (separated perfectly by a straight line/ flat surface)

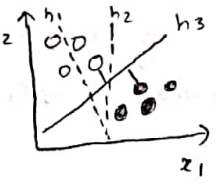
- There can be more than one choice of dividing line b/w groups of circles and squares.
  - possibilities ⇒ a, b, c
  - maximum margin hyperplane ⇒ chosed by algorithm from possibilities

Maximum Margin Hyperplane / MMH

- creates greatest separation b/w two classes
- generalize best to future data.
- improve chance that in case of random noise, points will remain correct side of boundary.
- Eg:



- support vectors:
  - points from each class that are closest to MMH.
  - Each class must have atleast one support vector
  - support vectors alone are possible to define MMH. (key feature of SVMs)



H_1 ⇒ doesn't separate classes
h_2 ⇒ separates only by a small margin.
h_3 ⇒ separates with max margin

MMH - In case of linearly separable data

- in case of n-dimensional space MMH is denoted by eqtn.

$$\vec{w} \cdot \vec{x} + b = 0$$

→ implies vectors

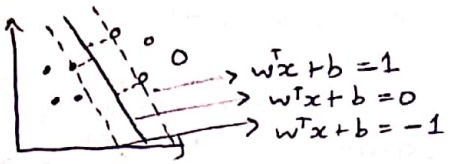
$\vec{w}$ ⇒ vector of 'n' weights $\{w_1, w_2 \cdots\}$

b ⇒ a single no. (bias) ⇒ intercept term.

- The goal of the process is to find set of weights that specifies two hyperplanes,

$$\vec{w} \cdot \vec{x} + b \geq +1$$
$$\vec{w} \cdot \vec{x} + b \leq -1$$

w → weight vector
x → input vector
b → bias.



→ All the points of one class fall above first hyperplane & all the points of other class fall below second hyperplane.

→ Distance b/w two planes, $D = \frac{2}{||\vec{w}||}$

- $||\vec{w}||$ ⇒ Euclidean Norm (distance from origin to vector w)
- for max distance, we need to minimize $||\vec{w}||$

→ Task represented as a set of constraints

1) $\min \frac{1}{2} ||\vec{w}||^2$
2) $s.t.\ y_i (\vec{w} . \vec{x_i} - b) \geq 1, \forall \vec{x_i}$

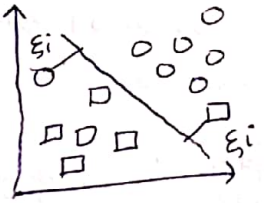
① ⇒ to minimize Euclidean norm
② ⇒ subject to condition that each y_i data points is correctly classified.

∀ ⇒ for all.
y ⇒ class value (+1/−1)

MMH - In case of non-linearly seperable data.

→ use of slack variable (ξ_i)
[creates a soft margin that allows some points to fall on incorrect side of the margin]

ie, allows misclassification of difficult/noisy examples.



- 2 points falling on wrong side of line
- slack variables represented by ξ_i (x_i).

→ A cost value (C) is applied to all points that violate constraints

→ Rather than finding maximum margin, algorithm attempts to minimize total cost.

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

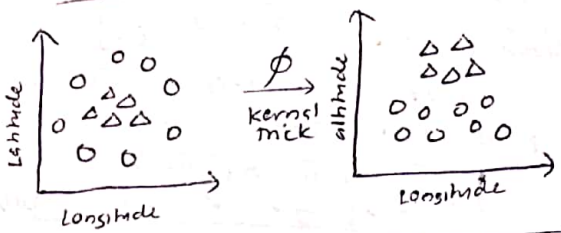
$$s.t.\ y_i (\vec{w} . \vec{x_i} - b) \geq 1 - \xi_i, \quad \forall \vec{x_i}, \xi_i \geq 0$$

C ⇒ cost parameter | Greater C ⇒ harder the optimization
Lower C ⇒ wider overall margin.

- balance b/w these 2 preferred to generalize future data.
- Real world applications ⇒ relationship b/w variables are non linear

Kernal Trick

→ kernals used to map the problem into a higher dimension space. (ie, a nonlinear relationship may suddenly appear to be quite linear)



Input space | Higher Dimension space

○ sunny △ snowy

- 'left' scatter plot [non linear relationship b/w a weather class — 2 features — latitude & longitude]
- add additional dimensions inorder to create separation

→ Kernal Trick ⇒ process of constructing new features, that express mathematical relationships between measured characteristics.

→ Altitude feature can be expressed mathematically as an interaction b/w latitude & longitude.

→ Strengths & Weakness (svms with nonlinear kernels)

Strengths	Weaknesses
• used for numeric prediction problems • not influenced by noisy data. • not prone to overfitting. • easier to use than neural networks • high accuracy in data mining	• various combinations of kernals and model parameters needed testing. • slow to train if dataset is large. • Results in a complex black box model — difficult to interpret.

Kernal functions – general form

- denoted by phi ($\psi(x)$) ⇒ mapping of data into another space.
- applies some transformation to the feature vectors x_i and x_j and combines using dot product ⇒ returns a single number.

$$K(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$$

- All SVM software packages will include these kernals.
- Most commonly used,

(i) Linear Kernal

- simplest kernal function
- doesn't transform data at all
- simply expressed as the dot product of the features.

$$K(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$$

(ii) Polynomial kernal

- adds a simple non-linear transformation of the data.

$$K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d$$

d = degree of polynomial.

(iii) Sigmoid Kernal

- analogous to neural network

$$K(\vec{x_i}, \vec{x_j}) = \tanh(K, \vec{x_i} \cdot \vec{x_j} - \delta)$$

tanh → hyperbolic tangent function

(iv) Gaussian RBF kernal

- general purpose kernal
- similar to RBF neural network.
- performs well on many types of data.
- reasonable starting point for many learning tasks

$$K(\vec{x_i}, \vec{x_j}) = e^{\frac{-\|\vec{x_i} - \vec{x_j}\|^2}{2\sigma^2}}$$

sigma, σ ⇒ adjustable parameter.

Note: No reliable rule to choose a kernal.
Choice of kernal is arbitrary
⇒ performance may vary slightly.

Multiclass SVM

- classification with more than two classes.
- depends on whether classes are mutually exclusive or not.
- 2 methods;

(i) Any of classifications (multilabel/ multivalue)

- classification not mutually exclusive
- The decision of one classifier has no influence on decision of other classes.
- Eg: Text classification
- Formal definition ⇒

$$\gamma_j(d) \in \{c_j, \bar{c_j}\}$$

- Eg: A doc about '2008 olympics' should be a member of 2 classes (china class & sports class)

- Steps:
  1) Build a classifier for each class
  2) Apply classifier seperately.

(ii) One of classification

- classification mutually exclusive
- each document belongs to exactly one of the classes.
- called multinominal, multiclass functions
- Formal definition

$$\gamma(d) \in \{c_1, \ldots c_j\}$$

- Eg: KNN is a nonlinear one of classifier.
  [doc member of only one class ie, china.]

- Steps:
  1) Build a classifier for each class
  2) Apply classifier seperately.
  3) Assign doc to the class with
     → maximum score.
     → maximum confidence value.
     → maximum probability.

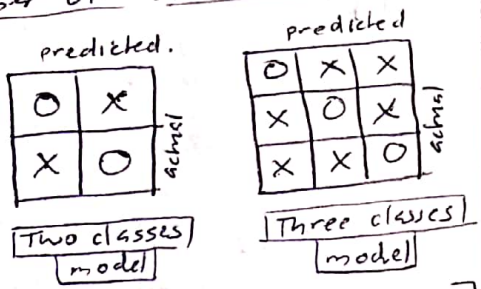
Module VI

Evaluching Model Performance

- process of evaluchng ML algorithms
- Algorithms have varying strengths and weaknesses.
- Classifiers are evaluated ⇒ reflects type of data.
  - Actual class values. [key to evaluchon]
  - Predicted class values,
  - Eshmated probability of the predichon.
- Goal ⇒ maintain two vectors (Actual & Predicted class values)
  - must have same no. of values.

Confusion Matrix (imp. measure of model perbmance)

- A table that categorizes predichons ⇒ whether they match actual value.
- one dimension ⇒ possible categories
  other dimension ⇒ actual values.
- matrix can be created for models that predict any number of class values.
- Eg:



Two classes model | Three classes model

- correct predichons [fall on the diagonal in confusion matrix (denoted by O)]
- Incorrect predichons [off diagonal matrix cells (denoted by X)]
- Performance [based on counts of predichons falling on/off the diagonal]

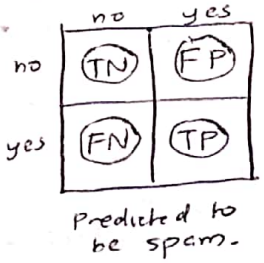
Positive class [class of interest]

negative class [other except class of interest]

- The relationship b/w positive & negative class predichons can be depicted as a 2×2 confusion matrix.
- 4 categories where predichon fall,
  1) True Positive (TP)
  2) True Negative (TN)
  3) False Positive (FP)
  4) False Negative (FN)

- TP [correctly classified as the class of interest]
- TN [correctly classified as not class of interest]
- FP [Incorrectly classified as class of interest]
- FN [Incorrectly classified as not class of interest]

- Eg: spam classifier.



Measuring Performance [using 2×2 confusion matrix]

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Error Rate} = \frac{FP+FN}{TP+TN+FP+FN}$$

or $\text{Error rate} = 1 - \text{accuracy}$

ie, if a model is 95% correct, it is 5% incorrect.

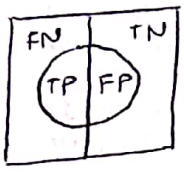
Precision and Recall

Precision [proportion of positive examples that are truely positive]

- trustworthy
- precise model will only predict positive classes
- $\text{Precision} = \frac{TP}{TP+FP}$

→ **Recall** [measure of how complete the results are.]

- **high recall** ⇒ large portion of positive examples.
- Eg: An SMS spam filter has a high recall ⇒ if majority of spam msgs are identified correctly.
- $\text{Recall} = \frac{TP}{TP + FN}$



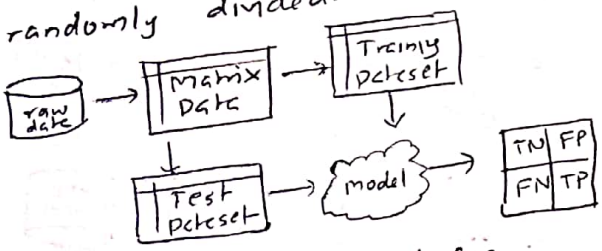
$\text{Precision} = \frac{a}{\square}$

$\text{Recall} = \frac{a}{\square}$

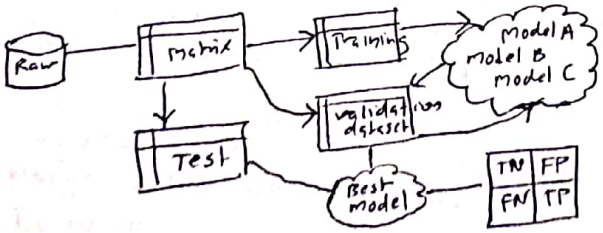
Holdout method

→ procedure of partitioning data into training & test datasets

→ Training dataset (2/3) ⇒ used to gen. model applied to,
test dataset (1/3) ⇒ used to gen. predictions
(proportion can vary)
randomly divided.



→ validation dataset: used for itering & refining model.
(Typical split ⇒ Training (50%)
Test (25%)
validation (25%))



- **Problem**: chance of a class being omitted from training dataset.
- **Solution**: [Stratified Random Sampling] guarantees random partitions have nearly same proportion of each class (as the full dataset)
- create DataPartition() in R studio.
- **Problem**: substantial amount of data cannot be used for training

Cross Validation

→ based on repeated holdout
→ known as k-fold cross validation
→ industry standard for estimating model performance.
→ randomly divides the data into 'k' random partitions called folds. (common = 10 fold CV)
→ For 10 folds, each fold comprises of 10% of data
- folds matching 10% sample used for model evaluation.
- process of training / evaluating done 10 times (10 diff. combinations) and average performance is reported

→ same class balance is maintained
→ createFolds() in R studio

Bootstrap Sampling

→ alternative to k-fold CV
→ less freq. used
→ method ⇒ creation of several randomly selected training & test datasets.
- Results of various datasets are then averaged to obtain final estimate

→ **Difference**
- In CV, each example can only appear once.
- In Bootstrap, examples can be selected multiple times
process: [sampling with replacement]

→ $P\left(\begin{matrix}\text{given instance included in} \\ \text{training dataset}\end{matrix}\right) = 63.2\%$
ie, test dataset = 36.8%

- Bootstrap sample ⇒ less representative of full dataset
- CV ⇒ 90% for training

→ **Error Rate**: $0.632 \times \text{error}_{\text{test}} + 0.368 \times \text{error}_{\text{train}}$

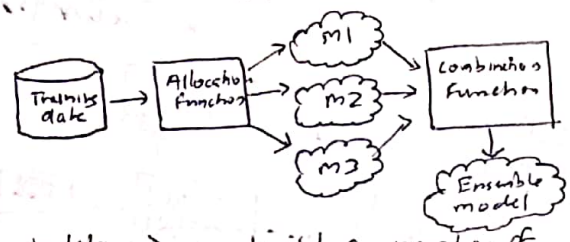
→ **Advantage**:
(i) works better on small datasets
(ii) ~~Appli~~ BS has applications beyond performance measurement
(iii) used to improve model performance.

Improving Model Performance

→ **Meta Learning**: Technique of combining & managing predictions from multiple models to form powerful model

→ **Ensemble methods**: (uses meta-learning approach)

- uses multiple learning algorithms to obtain better performance
- helps improving machine learning results by combining several models
- Idea ⇒ combining multiple weaker learners ⇒ a stronger learner is created.
- Ideal ensemble ⇒ includes diverse set of models.



Input data ⇒ to build a number of models

Allocation Function ⇒ dictates how much of training data each model receives.

Combination Function ⇒ how disagreements are reconciled

- Advantages (over single models)

(i) Better generalizability of future problems.

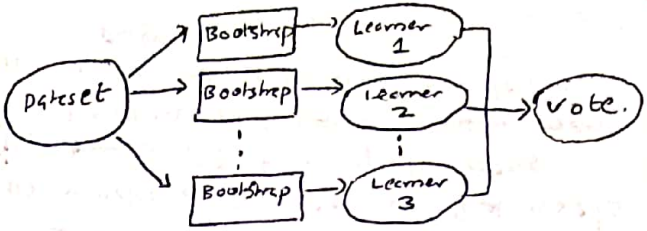
(ii) Improved performance on massive or miniscule datasets.

(iii) Ability to synthesize data from distinct domains

(iv) more understanding of difficult learning tasks.

Bagging

→ one of the first ensemble method.
→ widespread acceptance.
→ technique ⇒ bootstrap aggregating
→ • generates a no. of training datasets
• datasets used to generate models.
• models predictions are combined using voting (for classification) averaging (numeric prediction)



→ Bagging is often used with decision trees (tendency to vary)
→ ie, used with unstable learners. (one changes substantially when input changes slightly)
→ bagging() ⇒ to train model in R.
ipred package ⇒ for bagged decision trees
nbag parameter ⇒ used to control no. of decision trees voting (default = 25)

Boosting

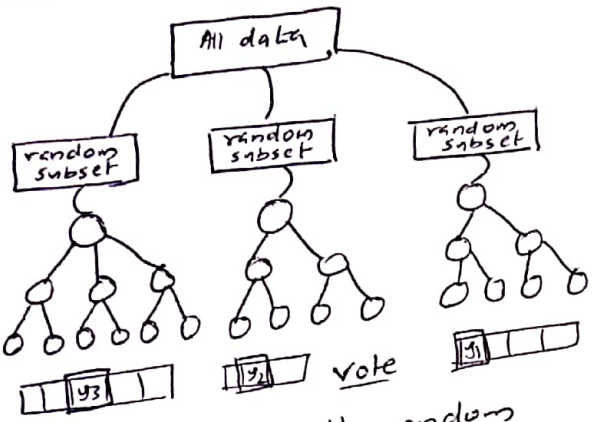
→ another ensemble-based method
→ boosts performance of weak learners
→ similar to bagging [uses models trained on resampled data & vote to determine final prediction]
→ key differences to bagging
- resampled datasets are constructed specifically to generate complementary learners.
- each learner's vote is given a weight (based on past performance)

→ possible to increase performance ⇒ by adding additional classifiers
→ Adaboost/Adaptive boosting:
- A boosting algorithm
- generates weak learners that learn a large portion by paying attention to misclassified examples

Random Forests

→ another ensemble-based method.
→ focuses only on ensembles of decision trees.
→ designed by Leo Breiman and Adele Cutler.

- combines (base principles of baggins) with (random feature selection)
  ⇒ to add additional diversity to decision tree models.
- After ensemble of trees (the forest) is generated, model uses a vote to combine the tree's predictions



- uses only a small, random portion of full feature set
- can handle extremely LARGE datasets
- error rates similar to any other method.
- easier to use, less prone to overfitting
  powerful & versatile.

Strengths	Weaknesses
• all purpose - model that performs well on most models • can handle noisy / missing data as well as categorical / continuous features. • selects only the most important features. • can be used on data with large no. of features	• not easily interpretable • require time & work to tune the model

- randomforest used in R studio.
  package
- randomForest() creates 500 trees by default
  - considers sqrt(no of features) at each split.
- Out of Bag (OOB) error rate ⇒ Total error rate of predictions.