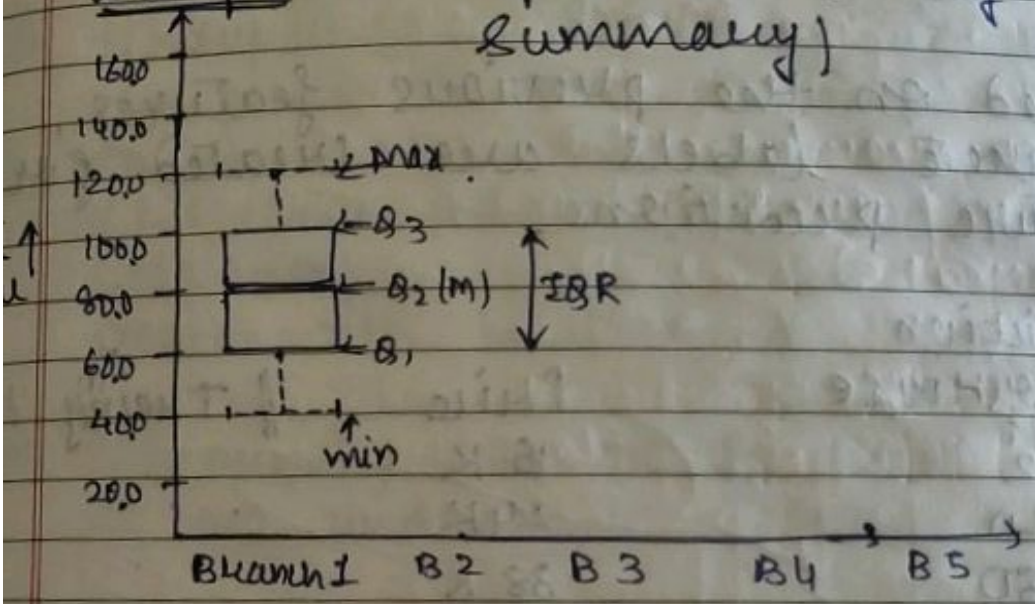
about the data residing at extremes. Thus, 5 ~~point~~ number summary is used.

Steps:

		Order
i)	Observe Q_1	(2)
ii)	" Median	(3)
iii)	" Q_3	(4)
iv)	" minimum	(1)
v)	" maximum	(5)

Box-plot (Representation of 5 number summary)



Classification and Prediction

Decision Tree
Bayesian
ANN (Backprop.)
Other
- Fuzzy Set
- Rough Set
- Genetic Algo
- KNN (Clustering)
- Case Base reasoning

Linear regression

Supervised learning
- class label given based on features

| | | | → classification → good / extraordinary

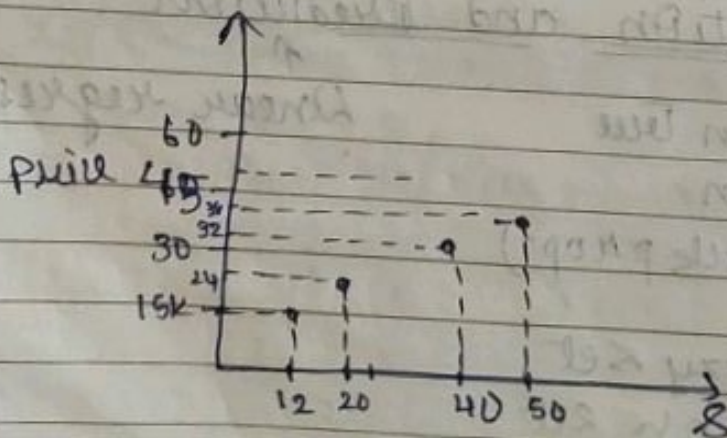
features (attributes) — (Label) // discrete

Ex- credit rating to predict whether person should be given loan or not.

Based on the previous features, discrete labels are created for future predictions.

Prediction

Size of House	Price // Training data
12	15 K
20	24K
50	38 K
40	32K



// curve

A best-fit line is evaluated and further analysis performed.

Prediction deals with continuous values [classification deals with discrete values]

When dealing with curves, the degree of polynomial (sophistication level) needs to be carefully chosen.
If almost all the points fall on the curve, the model may fail to correctly classify if some outliers is fed. (Analogous to memorization).

⇒ Unsupervised learning is where only the features are provided. We need to determine the labels as well. (No class label provided with feature vector).
Deals with clustering problems.

⇒ Determining the no. of clusters in advance is an undecidable problem.

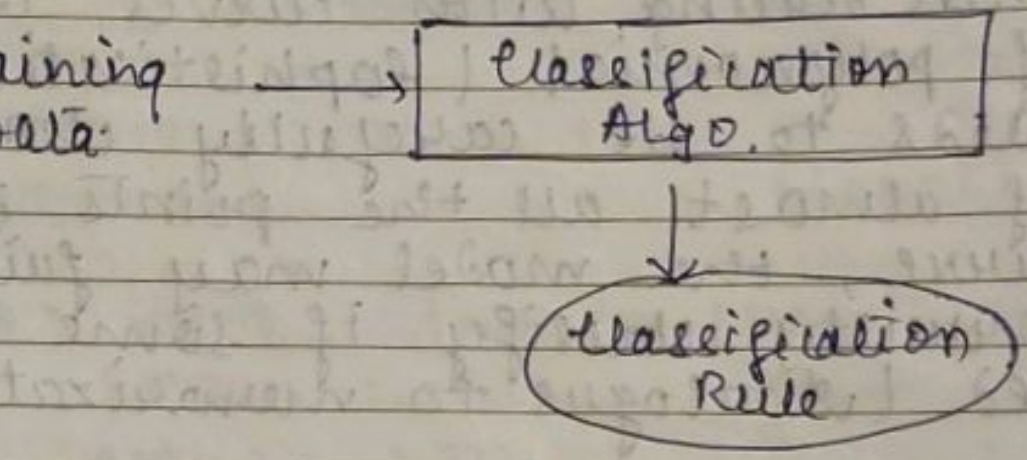
1) Model construction
↑ learning
Rule, based on training data

Ex- how we calculate the values of β_0 & β_1 in the model for:

$$Y = \beta_0 + \beta_1 x$$

Method (Basic): Hold out
- Divide data into training & testing
(80% of total) (20%)
hold 20% of the data
Should be random

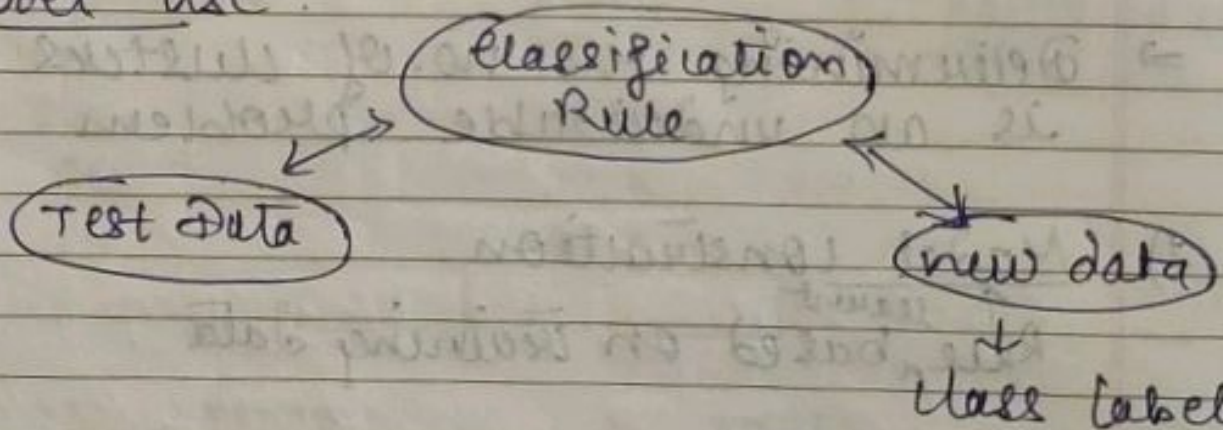
Train your model based on the 80% data & run error analysis on the held out 20% of data.



Model construction

After training the model, the model is 'repaired' based on the error generated by the built model.

2) Model use

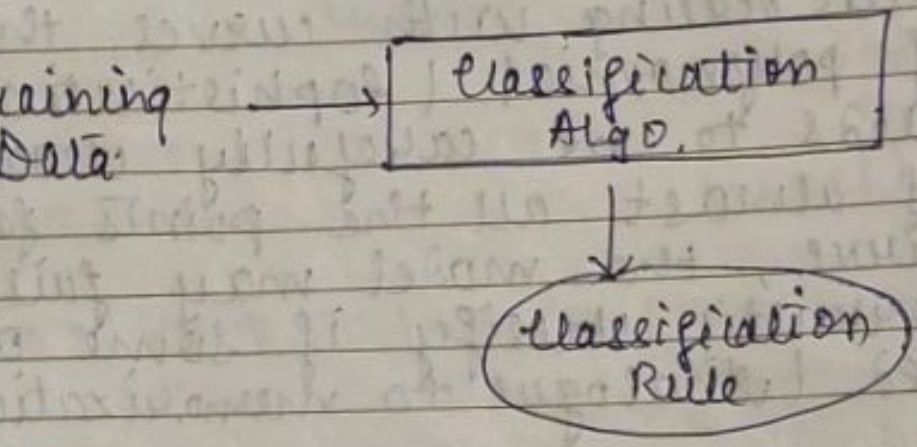


Hold out performed in this stage.

Before classification & prediction

1) Data cleaning
2) Relevance analysis
   - Irrelevant data may lead the model to deviate from the intended path.

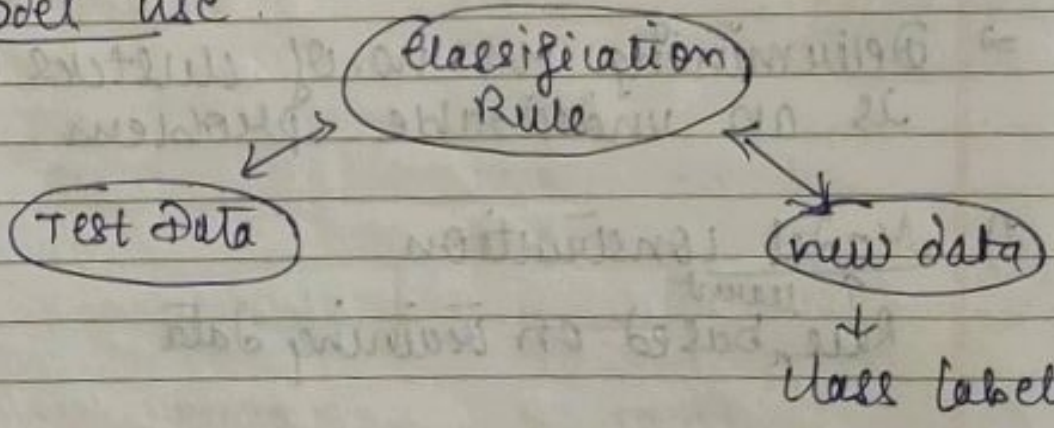
Train your model based on the 80% data & run error analysis on the held out 20% of data.



Model construction

After training the model, the model is 'repaired' based on the error generated by the built model.

2) Model use



Hold out performed in this stage.

Before classification & prediction

1) Data cleaning
2) Relevance analysis
   - Irrelevant data may lead the model to deviate from the intended path.

3) Data transformation
- especially when dealing with distance based algo. (for classification)

	Income	Age
will dominate over other.	20,000	03
	18,000	24.
	:	:

age if normalization not performed.

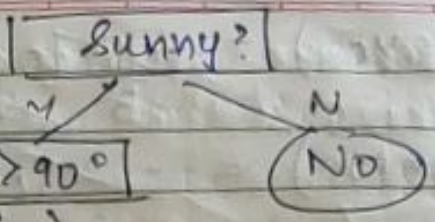
19/9 comparing classification methods
Parameters :

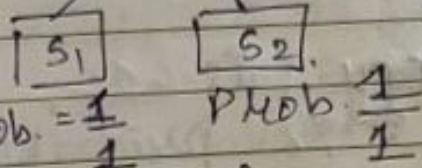
i) prediction accuracy.
- Accuracy on the new (testing) data. How the classifier works on data other than ~~on~~ the one on which it was trained.

ii) Speed.
- The computational power of the classifier.

iii) Robustness.
- How your classifier handles noisy data (or missing data). In the worst case, the classifier may give ambiguous results.

iv) Scalability.
- Ability to handle a large set of data This is interlinked with the speed parameter too
(Volume, variety, velocity of Big Data)

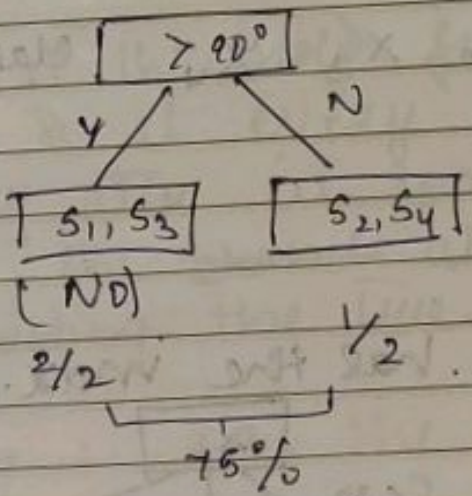




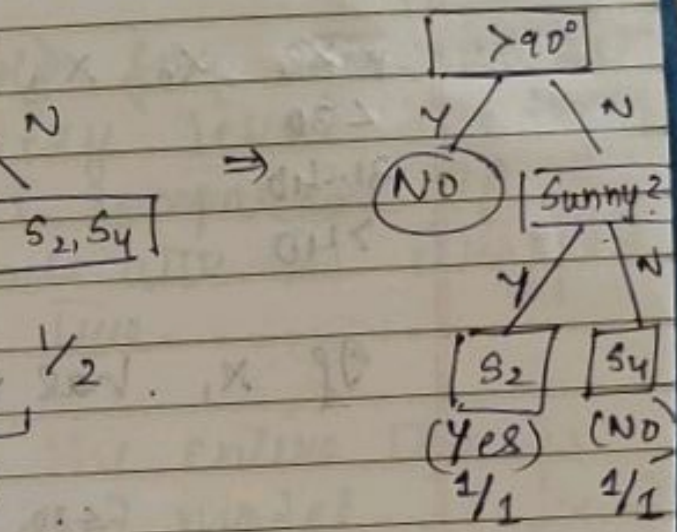
Prob. = 1/1 (NO). Prob 1/1 (Yes).

∴ All paths give accurate results (atleast on training data)

Conversely,



⇒



Since, training data here consists of all the possible combinations without any ambiguity, we get 100% results on the training data.

If we add the tuple (N, N, Y), the model / tree will not give a 100% output or accuracy.

Most relevant attribute is generally considered as the root node. → having higher info. gain

$$I(S_1, S_2) = -\frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{3}{4}\log_2\left(\frac{3}{4}\right)$$

(Yes, No)

=

$I(S=Yes) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right)$

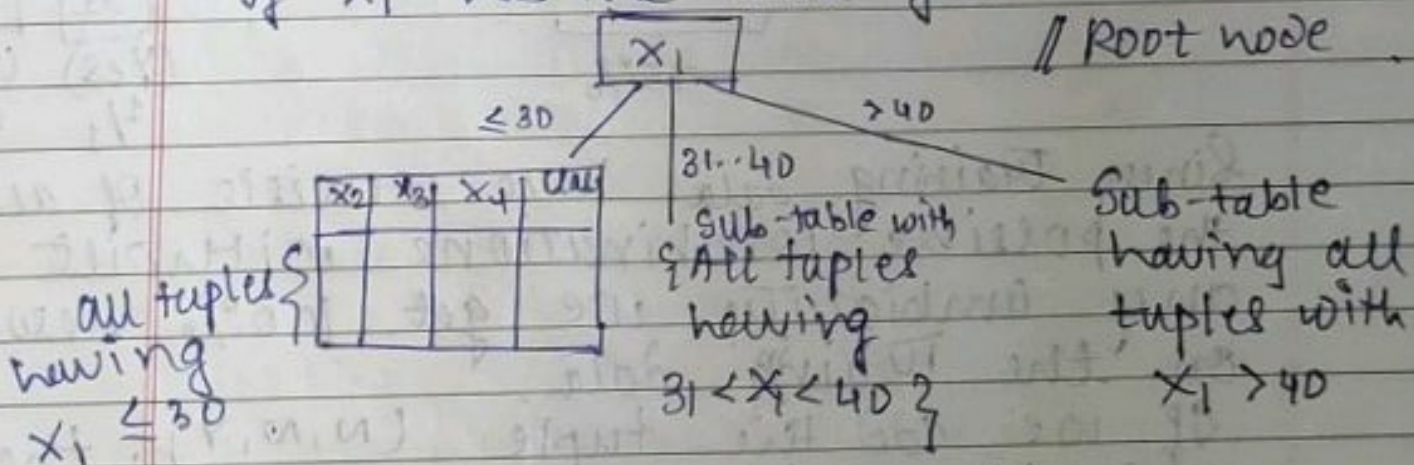
$I(S=NO) = -\frac{2}{3}\log_2\left(\frac{2}{3}\right)$

19/9/19

When entropy is 1, the result is uncertain (50% probability)
While if it is 0, the outcome is certain (completely, i.e. 100% probability)

X_1	X_2	X_3	X_4	Class Label Y
< 30				
31..40				
> 40				

If X_1 has the max. gain:



If all the tuples in the sub-table have the same class label, a leaf node is created with the class label and the process for that branch terminated.

For all other sub-tables, repeat the entire procedure of calculating info. gain. (discard the previous table)

Limitations:

→ Outliers can create a serious problem in case of decision tree.
→ Not scalable (In-memory processing thus, mem. usage limitations)
→ If let's say, > 40 branch has only one tuple and based on that we terminate the branch with leaf label 'NO' it is overfitting. May lead to misclassification.

Pre-pruning: A threshold for attr. is set & at every level, the info gain of every attr. is compared. If found less, the ~~branch~~ attr. is dropped. Done while building the tree.

Post-pruning: After the entire tree is built and model is found to misbehave, a branch is selected and cut out temporarily. The tree left is tested with the testing data again and if the error is compatible with the error obtained before the branch is permanently cut (pruned).

Bayes Rule: $P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$

- $P(Y|X)$: Posteriori prob.
- $P(X|Y)$: likelihood
- $P(Y)$: Prior
- $P(X)$: normalization factor (prob. of obs. samples) [Evidence]

23/9/19

X - feature vector $\{X = (X_1, X_2 \cdots X_n)\}$

Y - class label (target class)

Ex: Y → Yes / No

If $P(Y/X_{new})$ ← new sample

$P(Y = Yes / X_{new})$ — (i)

$P(Y = No / X_{new})$ — (ii)

If (i) > (ii) then, outcome of $P(Y/X_{new})$ is predicted as 'Yes'

If (ii) > (i) then outcome → 'No'

$$P\left(\frac{Y}{X}\right) = \begin{cases} \dfrac{P(X_1, X_2, X_3 \cdots X_n | Y) \cdot P(Y)}{P(X_1, X_2 \cdots X_n)} \\ P(X_1 | X_2, X_3 \cdots X_n, Y) \cdot P(X_2 | X_3, X_4 \cdots X_n, Y) \cdot \\ \quad \cdots \cdot P(X_{n-1} | X_n, Y) \cdot P(X_n | Y) \end{cases}$$

What makes this theorem 'Naive' Bayes is the assumption that all the attr $(X_1, X_2, \cdots X_n)$ are independent of each other. Also, all the attr are equally contributing to the target label.

But all are dependent on target class Y.

Thus, the above formula becomes:

$$P(Y/X) = \frac{P(X_1|Y) \cdot P(X_2|Y) \cdots P(X_n|Y) \cdot P(Y)}{P(X_1) \cdot P(X_2) \cdot P(X_3) \cdots P(X_n)}$$

OR

$$P(Y|X) = \frac{P(Y) \cdot \prod_{i=1}^{n} P(X_i|Y)}{\prod_{i=1}^{n} P(X_i)}$$ ← Evidence (constant) irrespective of target class

Ex: X_{new} = (age = "<= 30", income = "medium", student = "yes", credit_rating = "fair")

Y_{new} = Buy_computer = ?

↳ not present in given data. Thus, we need to predict

Now, we don't need to calculate the prob. of age >30 or 40-50 etc. because we don't have to predict any value for it.

∵ $\prod_{i=1}^{n} P(X_i)$ is going to be same for all values (distinct) of "Buy_computer," we may skip its calculation.

∴ $P(Y|X) \propto P(Y) \cdot \prod_{i=1}^{n} P(X_i|Y)$

$$y = \text{argmax}\left(P(y) \cdot \prod_{i=1}^{n} P(X_i|y) \right)$$

Here, from data given,

$P(\text{Buy\_computer} = \text{Yes}) = \frac{9}{14}$

$P(\text{Buy\_computer} = \text{No}) = \frac{5}{14}$

Now calculate:

2/9 P(age = "≤30" / Buy-computer = "Yes")
3/5 P(" | " = "No")
4/9 P(income = "medium" / Buy-computer = "Yes")
2/5 P(" " = "No")
6/9 P(Student = "Yes" / " = "Yes")
1/5 P(" " = "No")
6/9 P(credit-rating = "fair" / Buy-computer = "Yes")
2/5 P(" " = "No")

Now,

$$P(\text{Buy-computer} = \text{"Yes"} \mid X) = \frac{9}{14} \cdot \frac{2}{9} \cdot \frac{4}{9} \cdot \frac{6}{9} \cdot \frac{6}{9}$$

$$= 0.028$$

$$P(\text{Buy-computer} = \text{"No"} \mid X) = \frac{5}{14} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5}$$

$$= 0.007$$

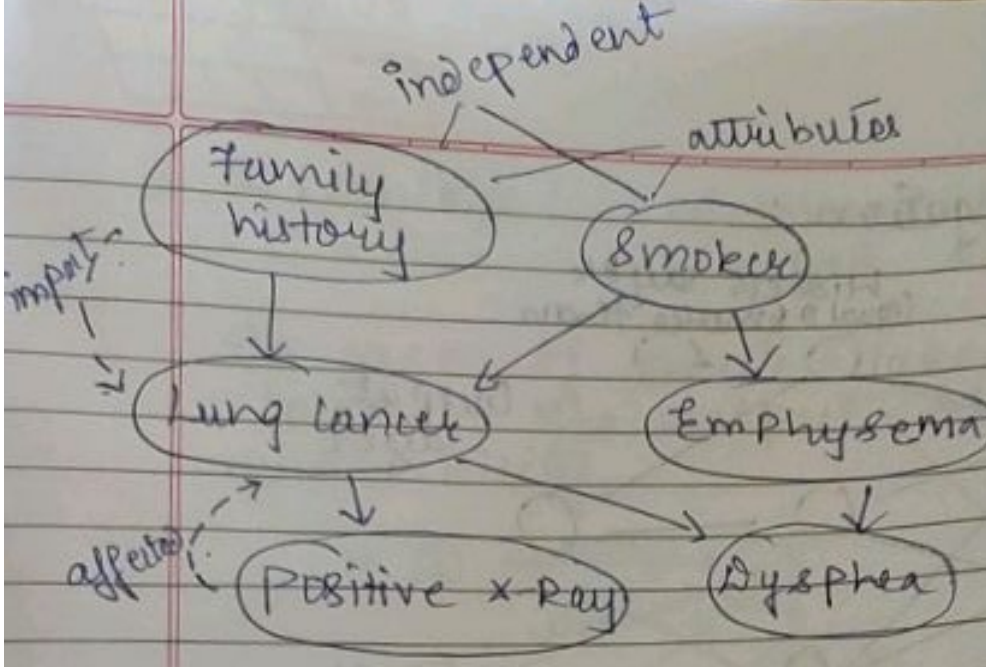
∴ Predicted value = Yes.

In case of continuous values:
Assume variable as Gaussian

$$P(X_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \cdot e^{-\left(\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)}$$

where, μ_y = mean of value of y
σ_y = Standard deviation

If we remove the assumption of attr being independent, Bayesian Belief Network comes into picture.



Representation of a belief network

Other way is conditional prob. table:

(Lung cancer)	FH, S	FH, $\bar{S}$	$\bar{FH}$, S	$\bar{FH}$, $\bar{S}$
LC	0.8	0.5	0.7	0.1
$\bar{LC}$	0.2	0.5	0.3	0.9

$$P(z_1, z_2 \cdots z_n) = \prod_{i=1}^{n} P(z_i \mid Parents(z_i))$$

26/9/19 Q. A murder takes place
witness 1 : 90% sure that X is murderer
Total 20 people present at the place.

→ $P(X = guilty) = \frac{1}{20} = 0.05$

$P(Y = eyewitness\_identifies \mid X = guilty) = 0.9$

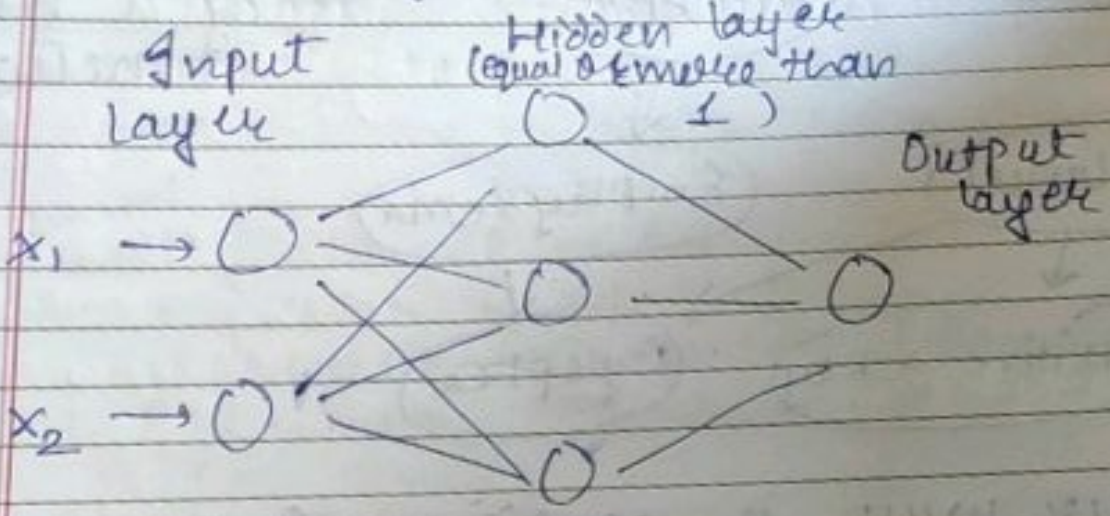
$P(Y = eyewitness\_identifies \mid X = not\_guilty) = 0.1$

$\therefore P(X = guilty \mid Y = eye\_witness\_identifies) = \frac{0.9 \times 0.05}{(0.9 \times 0.05) + (0.1 \times 0.95)}$

$= 0.3213$

$= 32\%$

Backpropagation :



$x = (x_1, x_2 \dots x_n)$

x_1 : height x_2 : weight

y : Adult or child.

Every ~~like~~ link is a weighted edge to the next node / neuron. The weight determines the attribute importance / relevance towards the particular node.

Neural network ≅ function approximator

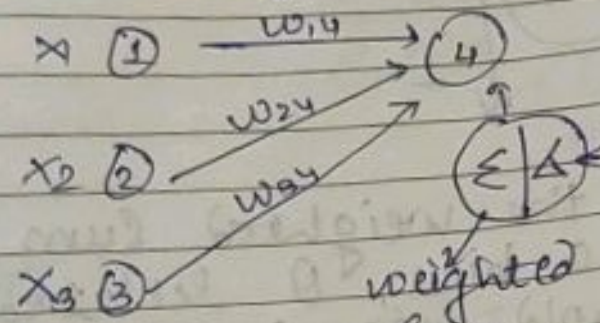
We go for NN only if we have a large amount of training data.

Disadvantage : Interpretability, because it involves learning weight matrix which may not always be straightforward.

Output layer — 1 node ⇒ Yes / No type of O/P.

Advantage : Accomodates non-linear shaped data.

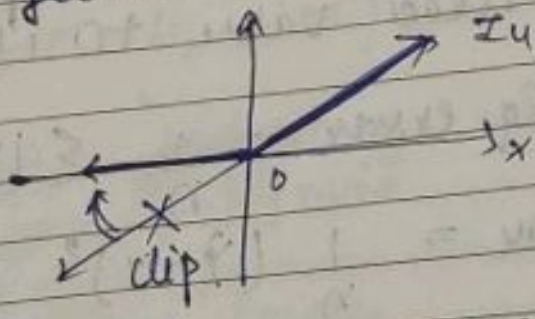
In case of multi-layer, the output obtained is probability.



$(I_4 = x_1 \cdot w_{14} + x_2 \cdot w_{24} + x_3 \cdot w_{34})$

Without Act. func., the node can produce a result between $(-\infty, \infty)$.
∴ To make it more interpretable, act. func. is applied.

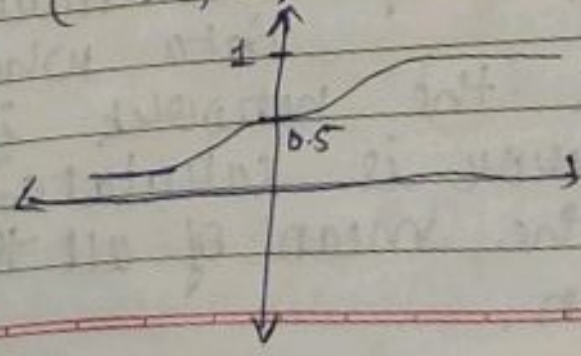
Ex:



$O_{I_4} = \max.(0, I_4)$

Other ex. is sigmoid func.

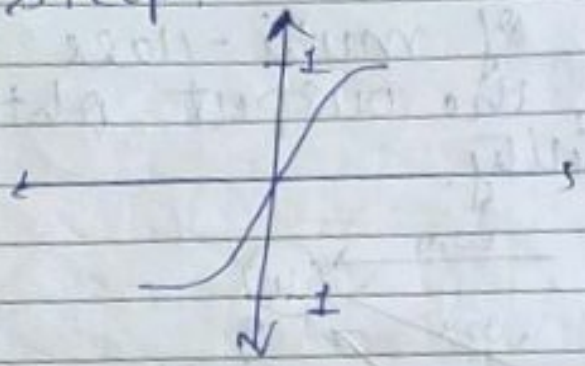
$(-\infty, \infty) \rightarrow [0, 1]$.



$$O_{I_4} = \frac{1}{1 + e^{-I_4}}$$

$$\boxed{y = \frac{1}{1 + e^{-x}}}$$

Yet another func. is hyperbolic tangent. Used when derivation is very steep.



Apart from the weighted sum, every node has a bias, θ related to it. The threshold associated to every node is generally considered as the bias.

bias.

Ex: $w_{12} \cdot x_1 + w_{22} \cdot x_2 - T > 0$

The output is then compared with the predicted value to check for errors.

Mean Sq. error $= \frac{1}{n} \Sigma (\hat{Y} - Y)^2$

Error $= \frac{1}{n} (\hat{Y} - Y)^2$

Epoch updating - The entire training data when iterated through the network is 1 epoch. The error is calculated. Then, the mean of all the Epoch's is taken.

Case updating - Error is calculated for every sample passed through the network and the attributes adjusted.

Weights are adjusted based on: (rate of change of error)

$\frac{\partial \text{Error}}{\partial w_{ij}}$ → (rate of change of weight b/w node i & j)

Steps:

1) Initialize weight
2) Propogate input forward (common bias for layer) [first layer $O_j = x_j$ i/p o/p]

$$I_j = \sum_i w_{ij} \cdot O_i + \theta_j$$

3) $$O_j = \frac{1}{1 + e^{-I_j}}$$

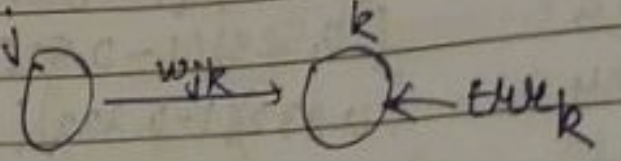
4) $$Err_j = \underbrace{O_j(1-O_j)}_{\text{derivative of sigmoid}} (\overbrace{T_j}^{\text{True o/p}} - \overbrace{O_j}^{\text{Actual o/p}})$$

for Output layer.

Predicted / actual

5) For hidden layer:

$$Err_j = O_j(1-O_j) \sum_k Err_k \cdot w_{jk}$$



6) $\Delta w_{ij} = (l)\, Err_j * O_i$ (first $O_i = x_i$)

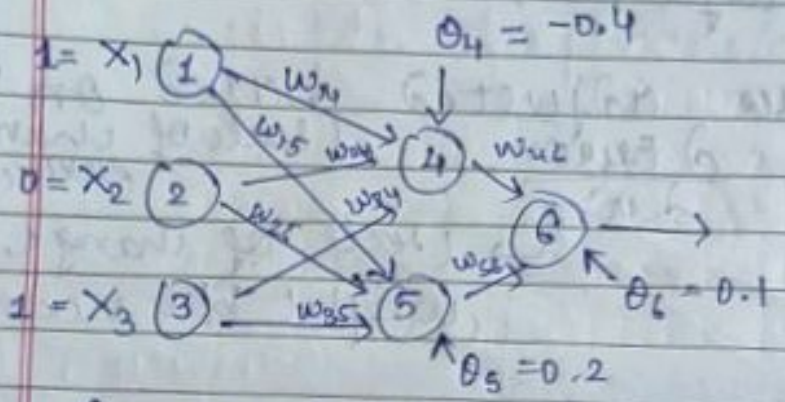
$w_{ij}^1 = w_{ij} + \Delta w_{ij} O_i$

learning rate (H/P parameter)

$\Delta\theta_j = (l)\ Err_j$

$\theta_j = \theta_j + \Delta\theta_j$

30/9/19 Ex:



$w_{14} = 0.2$
$w_{15} = -0.3$
$w_{24} = 0.4$
$w_{25} = 0.1$
$w_{34} = -0.5$
$w_{35} = 0.2$
$w_{46} = -0.3$)
$w_{56} = (-0.2)$

Correct O/P : T = 1

→

Unit / j	Net I/P, I_j	Output, O_j
4	$[0.2+0+(-0.5)]+(-0.4)$ $= -0.7$	$1/(1+e^{-(-0.7)}) = 0.332$
5	0.1	0.525
6	$[(-0.3)(0.332)+(-0.2)(0.525)]$ $+0.1 = -0.105$	0.474

$I_j = \sum_i w_{ij} O_i + \theta_j$

There is a huge difference between obtained O/P & expected output. Thus we adjust the values by back propagation.

Err	Err$_j$ (T_j ↓, O_j ↓)
6	$(0.474)(1-0.474)(1-0.474) = 0.1311$
5	$(0.525)(1-0.525)(0.1311 * (-0.2)) =$ -0.0065
4	$(0.332)(1-0.332)(0.1311 * (-0.3))$ $= -0.0087$

Now, update the weights & biases:

Given, learning rate, $l = 0.9$

weight & bias	New value
w_{46}	$(-0.3) + [(0.9)(0.1311 * 0.332)] = -0.261$
w_{56}	$(-0.2) + [(0.9)(0.1311 * 0.525)] = -0.138$
$w_{14}, w_{24}, w_{34}, w_{15}, w_{25}, w_{35}$	
θ_6	$(0.1) + [(0.9)(0.1311)] = 0.218$
θ_5	
θ_4	

⇒ Neural network is not interpretable.
⇒ Disadvantage: Representation is not easy.

⇒ Drop-out (ignoring or removing some neurons) is used to increase the accuracy of neural network. It also avoids over-fitting.

Other classification methods: (Practically not adaptable)

1) K-Nearest Neighbour
Euclidean distance is used to find the 'k' nearest nodes to a point and are classified as one.
Type of lazy learning. (Until & unless new point is not added, the model does not learn about any new knowledge)
While, decision tree is an eager

document classifier as it can build its model based on training data itself. It does not need to wait for testing data.

If it is a distance classifier, the max. value among the points is considered while if it is deployed as a real classifier, the average is considered.

2) Case Based Reasoning
Uses CBR (complex symbolic descriptions).
For any new input, it searches for the exact same case in the previous gained knowledge. If found, that exact output is given.

If not then, the most identical case is considered.

Ultimately the output depends on the importance or weightage of an attribute.

10/19 If cases are represented as graphs, the new sample graph should resemble a subgraph.

3) Genetic Algorithm
- Based on neural evolution.
- Rules are represented as string of bits

- Initially generate population randomly.

Attr. A_1 & A_2 — Target values C_1 & C_2

Rule: If A_1 (1) & not A_2 (0) then C_2 (0)

if it ← every attr is represented by 0/1.
ad only
value Target class: C1 : 1
Ex-M/F] C2 : 0

Rule: If not A1 (0) and not A2 (0) then C1 (1)

If attr has k values, ($k > 2$) every attr. is represented by k bits.

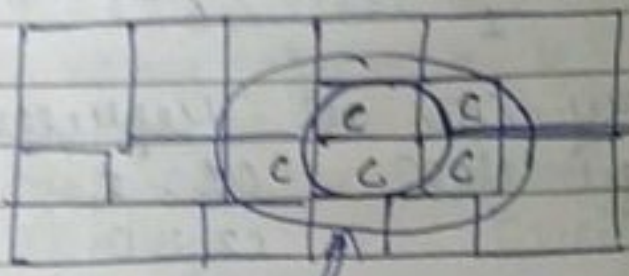
- Fitness of Rule / crossover and mutation
- New population is generated on the basis of fitness of rule.

4) Rough set approach
- Useful in handling imprecise and noisy data
- Discrete values
  ↓
  If continuous values → discretization required

- Based on equivalence class.
- Data tuples in one class should be indistinguishable w.r.t attr.
- In real world, however, complete

distinguishability may not be possible. Therefore, approximation may be used

↳ Lower approx.
↳ Upper approx



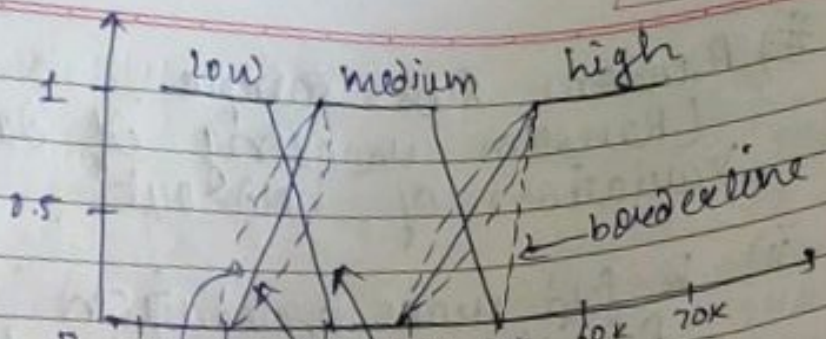
Upper approx.
(cannot guarantee whether tuple does not belong to the class)

) Fuzzy Set approach
- Rule based system have a sharp cut-off.
- To avoid it, some fuzziness is introduced.

If (year completed >= 2) ∧ (income ≥ 50K) then credit = "Approved"

Here if income = 49999 then also his/her credit is not approved which is very harsh.

Thus, instead of having 0/1 discrete values, we consider all the real values between 0 & 1.



somewhat lower

depends on the user how much weightage is to be given to these lines.

Prediction

1) Linear regression (example solved ahead)

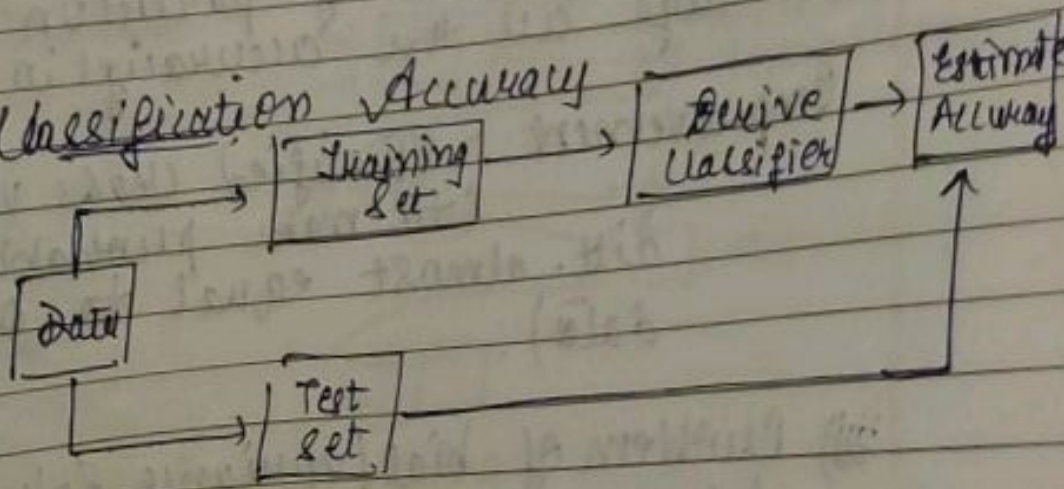
2) Non-linear

$y = \alpha_1 X + \alpha_2 X^2 + \cdots$ (based on shape)

$\downarrow$

$X_1, X_2, \cdots$ (if more dimensions (≥ 3))

Classification Accuracy



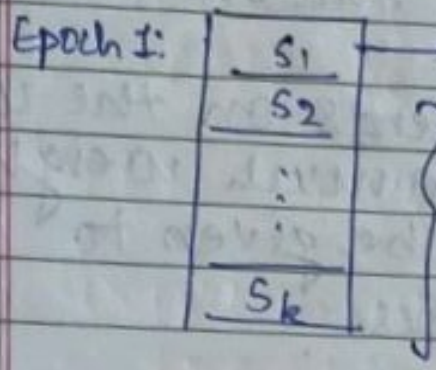
How to ~~estimate~~ improve accuracy?

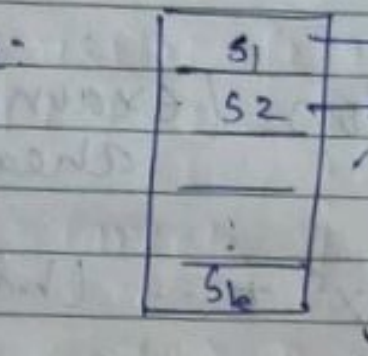
i) Hold Out → Train - Build classifier
→ Test - Test accuracy on

randomly

ii) Random sub-sample
(Random choosing of data)
variation of hold-out

iii) k-fold cross validation
- Partition dataset into k partitions





Repeated for every partition
Average all the accuracies in the end

Improvement - Stratified cross validation
(to make probability dist. almost equal to original data)

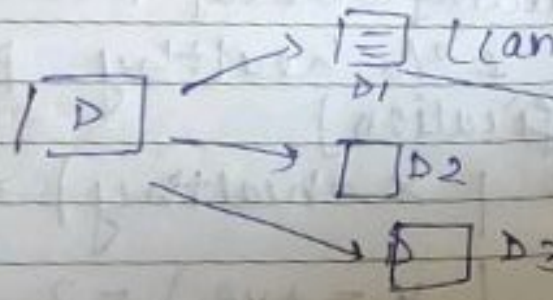
iv) Problem of bias / variance solved upto some extent.

Increase classification accuracy:
i) Pruning is a common solution

Other methods:

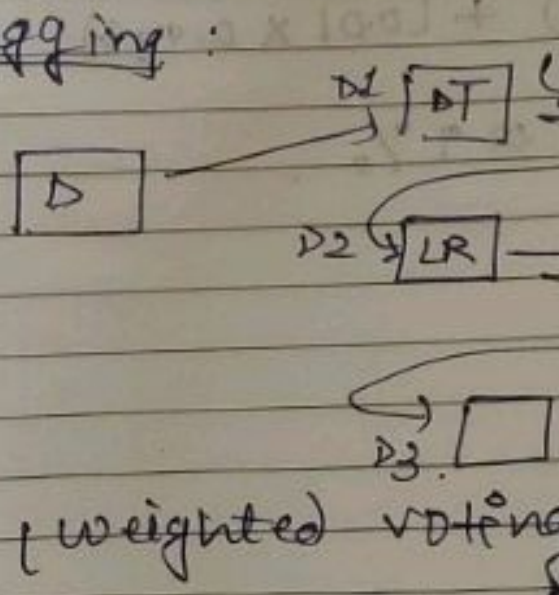
Bagging (Bootstrap aggregator)
- Train your dataset onto multiple classifiers and a model is built
Testing data is fed to each model
After this, voting is carried out and a majority opinion (result) is taken

Boosting
- Random sampling with replacement



Later, voting and majority.

Bagging:



(weighted voting)

⇒ Sensitivity = $\frac{\text{total-positive}}{\text{positive.}}$ → identified correct
(actual)

Specificity = $\frac{t\text{-neg}}{neg}$.

Precision = $\frac{t\text{-pos}}{(t\text{-pos} + f\text{-pos})}$.

⇒ Cancer is detected (assume)
0.1% is injected.

↳ P(X = cancer) = 0.001

P(Y = +ve | X = cancer) = 1
[Report is ~~correcting~~ correctly predicting]

- +ve for 1% healthy people (misclassification)

P(Y = +ve | X = healthy) = 0.01

P(X = cancer | Y = +ve) = ?

Ans. $\frac{1 \times 0.001}{(1 \times 0.001) + (0.01 \times 0.999)}$

= 0.091 ≈ 9%.