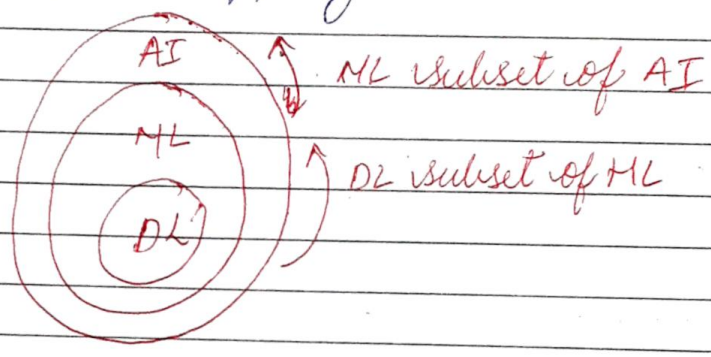
MACHINE LEARNING.

What is ML?

→ A field of study that gives computers a capability to learn without being explicitly programmed.
→ machine adapts to the user based on data
→ Eg:- amazon shopping



Well posed learning problem:- (Learning through experience)

→ An agent solves a problem or Task (T) performance (P) and again some experience E
→ If P is measured at T, it can improve (E). (learning by experience)...
→ Eg:- 1) playing checkers problem.
2) Hand written recognition problem.
3) Robo driving learning "

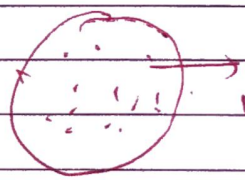
problem	Task (T)	Perf (P)	Exp (E)
1	playing against opponents to win	makes perfect moves to win game	It plays itself to improve

2. Handwriting recognition learning
Task :- classifying the images & text
Performance (P) :- Better classification
experience (E) :- A data-base of handwrit text

3. Robot driving learning problem
Task (T) :- Drive a car in 4 lane highway
Performance (P) :- Source to destination, the avg dist travelled (long & safe)
Experience (E) :- Images, vehicles on road

PERSPECTIVES OF ML.

→ It involves searching very large space of possible hypotheses to determine one that best fits the observed data & any prior knowledge held by learner.
Ex :-



Sorting Algos

ISSUES OF ML :-
1) What algos should be used?
2) Which algo is best for which type of problem
3) How much training data is sufficient? & testing data.
4) What kinds of methods should be used

5) What methods should be used to reduce learning overhead.
6) for which type of data which methods should be used?

DESIGNING A LEARNING SYSTEM:-

To get a successful learning system it should be designed for a proper design, several steps should be followed.

why? for perfect & efficient system.

4 steps:-

1) choosing the training experience
2) " " target function
3) " a representation for target function
4) " " learning algo for approximating the target function

final design is obtained.

STEP:- 1) Choosing a training experience:-
In choosing a training experience, 3 attributes are taken.
i) Type of feedback
(ii) Degree
(iii) distribution of examples

i) Type of feedback

(T) → (P) good → (E) good
using
Di Indi

__/__/__

Whether the training experience provides direct or indirect feedback regarding the choices made by performance system.
Direct feedback
Indirect feedback.

Examples:- chekers game, learning driving
→ you want to go with direct / Indirect
Its your choice

(ii) Degree to which learner will control the sequence of ~~learning~~ training.
Ex:- learning driving

with trainers help.	with trainers partial help	completely on your own.

(iii) How well it represents the distribution of examples over which the performance of final system is measured.
- more possible combinations
- more situations, more examples
- learning over distributed range of examples.

Ex:- learn driving

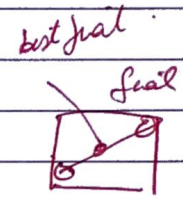
STEP: 2) CHOOSING A TARGET FUNCTION:-
What type of knowlege is learnt & how it is used by performance system.
Ex:- chekers game

__/__/__

In checkers game
- Travel only in forward dir.
- Only one move per chance.
- Only in diagnol dir.
- Jump over opponent.

→ while moving diagnolly set of all possible moves is called legal moves.
↓ one move.
Target move

Let's denote:-
Target function ⇒ $\boxed{v(b)}$
board state ⇒ $\boxed{b}$
legal moves set ⇒ $\boxed{B}$



How do you select a target func?
(i) b is the final board state that is won, then $\boxed{v(b) = 100}$
(ii) b is the final board state that is lost, then $\boxed{v(b) = -100}$
(iii) b is the final board state that is $\boxed{\text{draw}}$, then $\boxed{v(b) = 0}$
(iv) If b is not in final state then
$v(b) = v(b')$
b' = best final state

STEP-3 Choosing a representation for Target function
for any board state, we calculate function 'c' as linear combination of the foll board features i.e $c(b)$

x_1 : no of black pieces on board.
x_2 : no of red pieces on board.
x_3 : no of black kings on board.
x_4 : no of red kings on board.
x_5 : no of black pieces threatened by red. (blacks beaten by red)
x_6 : no of red pieces threatened by blacks. (beaten by black)

$$\hat{v}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_2 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

(rep of $c(b)$)

$(w_0 - w_6)$ = numerical co-efficients (OR) wt of each feature.

where,
numerical co-effs : determines the weitage of features.
$w_0 \rightarrow$ additive constant.
$w_1, w_2, w_3 \ldots$: weights.
$x_1, x_2, x_3, x_4, x_5, x_6$: features.

STEP: (4). Choosing a Learning Algo for approximating the Target function.
To learn a target function (f) we need set of training examples.

Describe a particular board state (b) & its training value.
$V_{train}(b)$

ordered pair = $(b, V_{train}(b))$
(training example representation.

Ex:- lets assume black won the game.
(i.e $x_2 = 0$, which means no red).

$V_{train}(b) = +100$.

$b = (x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0)$

$< (x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0)$
$+100 >$.

We need to do 2 steps.

(i) Estimating training Values.
In every step, we consider successor.
(depending on the next step of opponent)

$V_{train}(b) \leftarrow \hat{V}(\text{successor}(b))$.

↓

represents the next board state.

(estimating that this move will help destroy opponent).

$\hat{V} \rightarrow$ represents approximation.

(ii) Adjusting the weights.

There are some algos to find wts of linear functions.
Here, we are using LMS (Least mean sq)

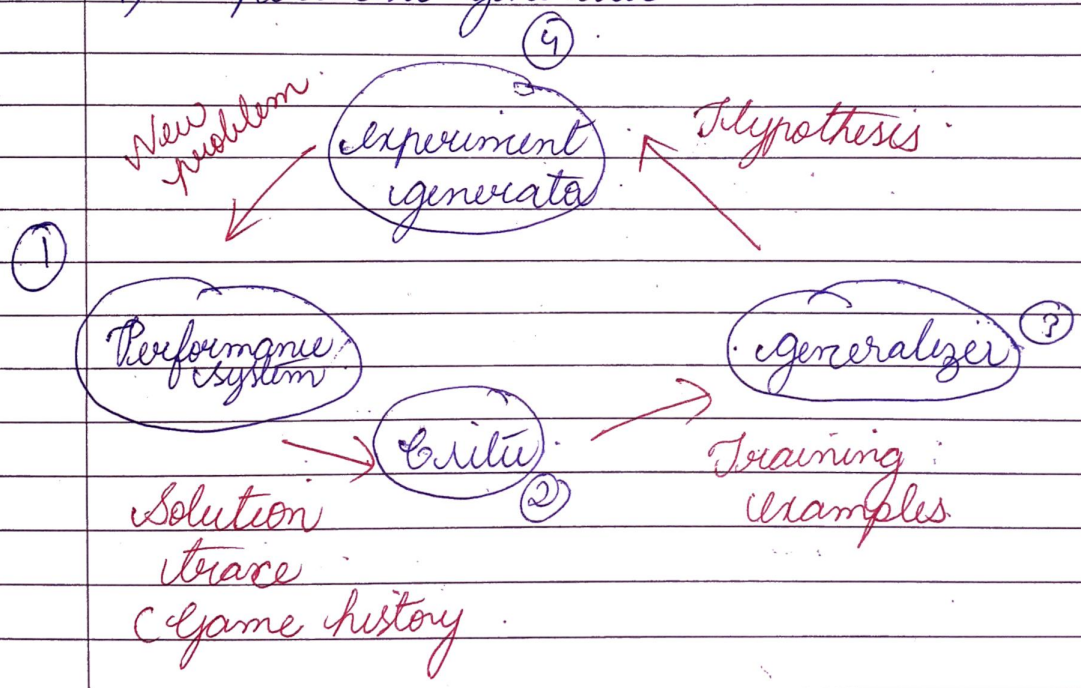
Error: $(V(train(b)) - \hat{V}(b))^2$

If error = 0, no need to change wts.
If error = +ve, each wt is increased in proportion.
If error = -ve, each wt is decreased in proportion.

* FINAL DESIGN :-

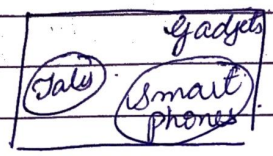
- has 4 modules.

1) performance system.
2) critic
3) generalizer.
4) experiment generator



_/_/_

* CONCEPT LEARNING.



unwise

* Features (Binary valued attributes)

	Tab	Smartphone	
1. Size :	large,	Small	$\rightarrow x_1$.
2. color :	Black,	blue	$\rightarrow x_2$.
3. Screentype :	flat,	Folded	$\rightarrow x_3$.
4. Shape :	Square,	Rectangle	$\rightarrow x_4$.

(i) concept $= \langle x_1, x_2, x_3, x_4 \rangle$ (4)

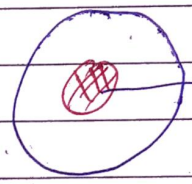
(ii) Tablet = < large, black, flat, Square >.

(iii) Smartphone = < Small, blue, folded, Rect >

$\therefore$ No of possible instances $= \boxed{2^d} = 2^4 = 16$ ✓

d = no of features.

$\therefore$ Total no of concepts $= 2^{4} = \boxed{2^{2^d}} = 2^{2^4} = 2^{16}$



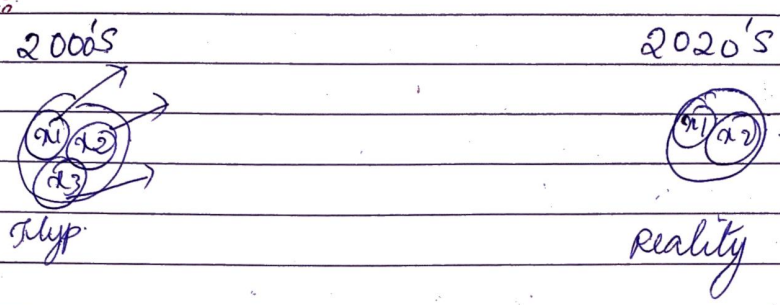
Target concept / hypothesis space

* If anything is represented with $\boxed{\langle \phi, \phi, \phi, \phi \rangle}$ then Reject all (most specific hypothesis

* $\boxed{\langle ?, ?, ?, ? \rangle}$ → accept all (most general hyp.

Main goal of CL :-?
finds all concepts / hypothesis that are consistant

Ex:-



always consistent.

* CONCEPT LEARNING AS A SEARCH.
Main goal of this search is to find the hypothesis that best fits the training examples.

Ex:- Enjoysport Learning Task.
* 6 attributes :- Sky (1), Air temp (2), humidity (3), wind (4), water (5), forcast (6)
3 values (sky) :- Rainy, cloudy, sunny.
Remaining attributes - only 2 values

* different instances possible = $3 \times 2 \times 2 \times 2 \times 2 \times 2$
$= \boxed{96}$

* Syntactically distinct hypothesis =
(Additionally 2 more values - ? & ϕ)
Eg:- < ϕ, Rainy, cloudy, sunny, ? >

$\therefore 5 \times 4 \times 4 \times 4 \times 4 \times 4 = \boxed{5120}$

* Semantically distinct hypothesis =
(Null taken as common
∴ only 1 more value).

$1 + (4 \times 3 \times 3 \times 3 \times 3 \times 3)$
$= 973$

After finding all the syntactically and semantically distinct hypothesis, we search the best match from all these.

* FIND (S) ALGO
(finding a maximally specific hypothesis)
– This algo considers only +ve examples
* Representation :-
most specific hypothesis ⇒ ϕ
most general hypothesis ⇒ ?

Algo :-
Step 1:- Initialize with most specific hypothesis (ϕ)
$h_0 = \langle \phi, \phi, \phi, \phi, \phi \rangle$ ⇒ 5 attributes.
Step 2:- For each +ve sample, for each attribute
if (value = hypothesis value) ⇒ Ignore
else
Replace with most general hypothesis (?)

Eg :-

Orig	Mnf.	clr	year	Type	Class
JP	Ho	B	1980	eco.	+(yes)
JP	To.	G	1970	spo	– (no)
JP.	To	B	1990	eco.	+
USA	AU	R	1980	eco	–
JP	Ho	W	1980	eco	+
JP	To	G	1980	eco	+
JP	Ho	R	1980	eco.	–

$\therefore$ $h_0 = \langle \phi, \phi, \phi, \phi, \phi \rangle$.
$h_1 = \langle$ 'JP', 'Mo', 'B', '1980', eco $\rangle$.
$h_2 = h_1$ [$\because$ h_2 is -ve so ignore]
$h_3 = \langle$ 'JP', ?, 'B', 1990?, eco $\rangle$.
$h_4 = h_3$.
$h_5 = \langle$ 'JP', ?, ?, ? eco $\rangle$
$h_6 = \langle$ 'JP', ?, ?, ? eco $\rangle$.

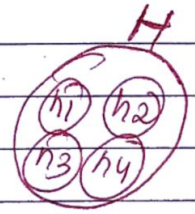
most general hyp.

Disadvantages :-
* considers only +ve values.
* h_6 may not be sole hyp that fits the complete data

* VERSION SPACE.
Subset of hypothesis H consistent with training examples (D).

$$VS_{H,D} = \{ h \in H / \text{consistent}(h, D) \}$$

M = hypothesis
D = training example



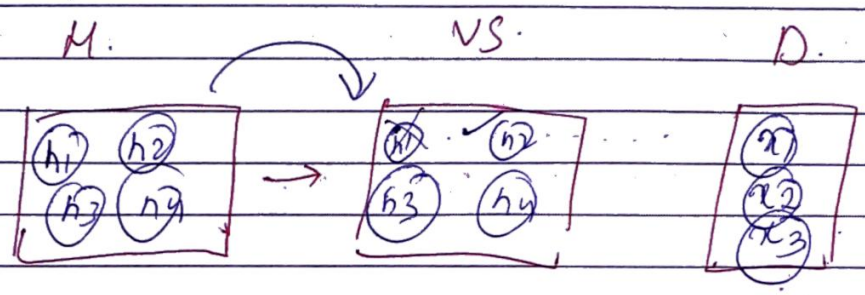
consistent
$h(x) = c(x)$

* Algorithm to eliminate version space

* LIST THEN ELIMINATE ALGO.
1. version space ← list containing every hypothesis in H.
2. from this step, we keep on removing inconsistent hyp from version space

for each training example, $\langle x, c(x) \rangle$.
remove any hypothesis that is ~~c~~
$h(x) \neq c(x)$

3. ~~Op~~ output the list of hypothesis into version space after checking for all training examples



for consistency $h(x) = c(x)$

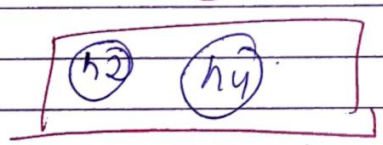
1) $\therefore h_1(x_1) = c(x_1)$
$h_1(x_2) \neq c(x_2)$

2) $h_2(x_1) = c(x_1)$.
$h_2(x_2) = c(x_2)$
$h_2(x_3) = c(x_3)$

3) $h_3(x_1) \neq c(x_1)$

4). $h_4(x_1) = c(x_1)$
$h_4(x_2) = c(x_2)$
$h_4(x_3) = c(x_3)$
~~$h_4 c(x_4)$~~ =

$\therefore$ VS



CANDIDATE ELIMINATION ALGO.

→ uses the concept of version space.
→ considers both +ve & -ve values (yes & NO).
→ both specific & general hypotheses.

→ for +ve samples move from specific to general.
→ for -ve samples move from general to specific

$S = \{\phi\ \phi\ \phi\ \phi\ \phi\}$ ↓
$G = \{?\ ?\ ?\ ?\ ?\ ?\}$ ↑

Algorithm

1. Initialize both general & specific hypothesis (S & G).
   $S = \{\phi, \phi, \phi, \phi, \ldots \phi\}$ depends on
   $G_1 = \{?\ ?\ ?\ ?\ \ldots ?\}$. no of attributes.

2. for each example,
   if example is positive.
   make specific to general.
   else if -ve.
   make general to specific

Example: Enjoy sport

$S_0 = \{\phi\ \phi\ \phi\ \phi\ \phi\ \phi\}$ $G_0 = \{?\ ?\ ?\ ?\ ?\ ?\}$.

i) +ve.
$S_1 = \{$'Sunny', 'warm', 'normal', 'strong', 'warm' 'same'$\}$
$G_1 = \{?\ ?\ ?\ ?\ ?\ ?\ ?\}$.

+ve, (specific to general -

__/__/__

Dataset (Enjoy Sport)

Sky	Temp	Humidity	Wind	Water	Forecast	Enjoy
Sunny	Warm	Normal	Strong	Warm	Same	yes (+ve)
Sunny	Warm	High	Strong	Warm	Same	yes (+ve)
Rainy	Cold	High	Strong	Warm	Change	no (-ve)
Sunny	Warm	High	Strong	Cold	Change	yes (+ve)

Example based on the above dataset.

1) +ve.

$S_1 = \{$'Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same'$\}$

$G_1 = \{??????\}$

2) +ve.

(Specific to general).

$S_2 = \{$'Sunny', 'Warm', ?, 'Strong', 'Warm', 'Same'$\}$

$G_2 = \{??????\}$

3) -ve.

(general to specific).

$S_3 = \{$'Sunny', 'Warm', ? 'Strong' 'Warm' 'Same'$\}$

$G_3 = \{<$'Sunny' ?????$>$, $<$? 'Warm' ????$>$, $<$????? 'Same'$>\}$

4) +ve.

$S_4 = \{$'Sunny', 'Warm', ? 'Strong' ? ?$>$

$G_4 = \{<$'Sunny' ?????$>$, $<$? 'Warm' ?????$>\}$

$\therefore S_4$ & G_4 = final hypothesis

$\wedge$ = and.

__/__/__

INDUCTIVE BIAS.

Remarks on CE & VS Algo
1) Will the CE algo gives us correct hypothes?
2) What training example should the learner request next?

Inductive Learning
from examples we derive rules.

Deductive Learning
Already existing rules are applied to our examples

* Biased Hypothesis Space.
does not consider all types of training examples:
Solution → includes all hypothesis.

ex: sunny $\wedge$ warm $\wedge$ normal $\wedge$ strong $\wedge$ cool $\wedge$ change = yes.

* Unbiased hypothesis Space:-
Providing a hypothesis capable of representing set of all examples.

Possible Instances :- $3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96$
Target concept :- 2^{96} (huge).
(practically not possible)

* Idea of Inductive Bias

learner generalises beyond the observed training examples to infer new examples

'>' → inductively inferred from

$x > y \Rightarrow y$ is inductively inferred from x

Example:-

Learning Algorithm → L

Training data $D_c = \{x, c(x)\}$

New instance $= x_i$

Represented as $L(x_i, D_c)$

$\therefore (D_c \wedge x_i) > L(x_i, D_c)$

DECISION TREE ALGO.

→ used in : tree structured classification & regression

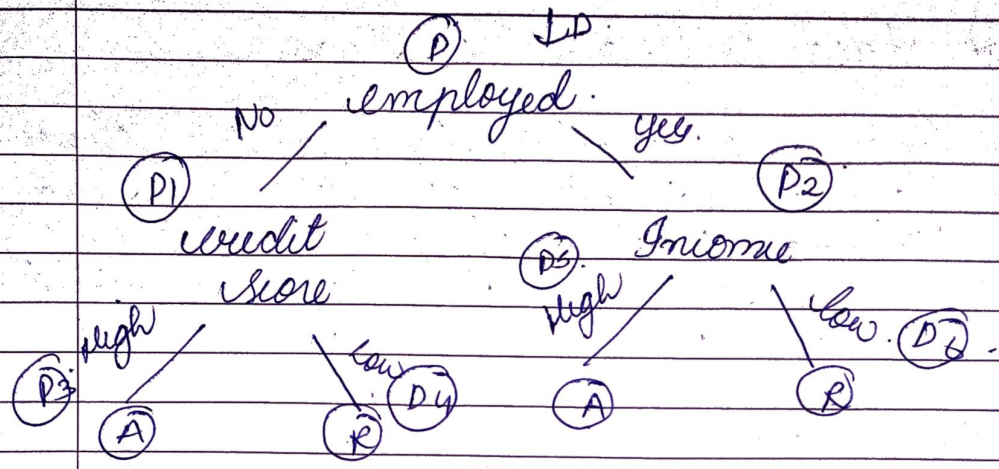
Dataset → Algorithm → classifies the data. (decision tree algo)

* 2 types of nodes

1. Decision node
2. leaf node.

Ex:- Loan system

Decides if the loan should be approved (rejected)



Algorithm

1. In the given dataset, choose a target feature attribute

2. Calculate information gain of target attribute

$$\boxed{IG = \frac{-P}{P+N} \log_2 \left(\frac{P}{P+N}\right) - \frac{N}{P+N} \log_2 \left(\frac{N}{P+N}\right)}$$

3. For remaining attributes, find entropy

$$\boxed{\text{Entropy} = IG \times \text{Probability}}$$

$$E(A) = \Sigma \frac{P_i + N_i}{P+N} I(P_i N_i)$$

4. Calculate gain $= \boxed{IG - E(A)}$

(Infogain − Entropy)

Practical Implementation of Decision Tree Algo.

Dataset :-

Age	Competition	Type	Profit
old	yes	S/W	Down
old	no	S/W	Down
old	no	H/W	Down
mid	yes	S/W	Down
mid	yes	H/W	Down
mid	no	H/W	Up
mid	no	S/W	Up
new	yes	S/W	Up
new	no	H/W	Up
new	no	S/W	Up

Step 1:- Target attribute = Profit.

Step 2:- Information gain.

$$I_G = \frac{-P}{P+N} \log_2\left(\frac{P}{P+N}\right) - \frac{N}{P+N} \log_2\left(\frac{N}{P+N}\right)$$

$\Rightarrow$ P = count (down) = 5.
N = count (up) = 5.

$$= \frac{-5}{10} \log_2\left(\frac{5}{10}\right) - \frac{5}{10} \log_2\left(\frac{5}{10}\right)$$

$$= -\left(\frac{1}{2} \log_2(2^{-1}) + \frac{1}{2} \log_2(2^{-1})\right)$$

$\log_n a^m = m \log a$

__/__/__

$$-\left(\frac{1}{2} \times -1 \log_2 2 + \frac{1}{2} \times -1 \log_2 2\right)$$

$$= -\left(\frac{1}{2} \times -1 + \frac{1}{2} \times -1\right)$$

$$-(-1) = 1$$

$IG = 1$

Step 3. Calculate entropy for remaining attributes

$$E(A) = \frac{\Sigma P_i N_i}{P+N} I(P_i N_i)$$

[IG × Probability]

(i) prepare a table for each attribute

rows: values of undertaken attribute (old, mid, new).

columns: values of target attribute (down up).

↓	down	up
old	3	0
mid	2	2
new.	0	3.

Entropy = IG × Probability

P = down count.

N = up count

① → $IG(old) = -\left(\frac{3}{3}\log\left(\frac{3}{3}\right) + \frac{0}{3}\log\left(\frac{0}{3}\right)\right) = 0.$

→ probability $= 3/10.$

→ entropy (old) $= 0 \times \frac{3}{10} = 0$

② → $IG(mid) = -\left(\frac{2}{4}\log\left(\frac{2}{4}\right) + \frac{2}{4}\log\left(\frac{2}{4}\right)\right]$

$= -\left(\frac{1}{2}\log\left(\frac{1}{2}\right) + \frac{1}{2}\log\left(\frac{1}{2}\right)\right]$

$= -\left(\frac{-1}{2} + \frac{-1}{2}\right) = -(-1) = 1.$

→ probability $= 4/10.$

→ entropy (mid) $= \frac{1 \times 4}{10} = 0.4.$

③ → $IG(new) = -\left(\frac{0}{3}\log\left(\frac{0}{3}\right) + \frac{3}{3}\log\left(\frac{3}{3}\right)\right] = 0.$

→ probability $= \frac{3}{10}.$

→ entropy (new) $= 0 \times \frac{3}{10} = 0.$

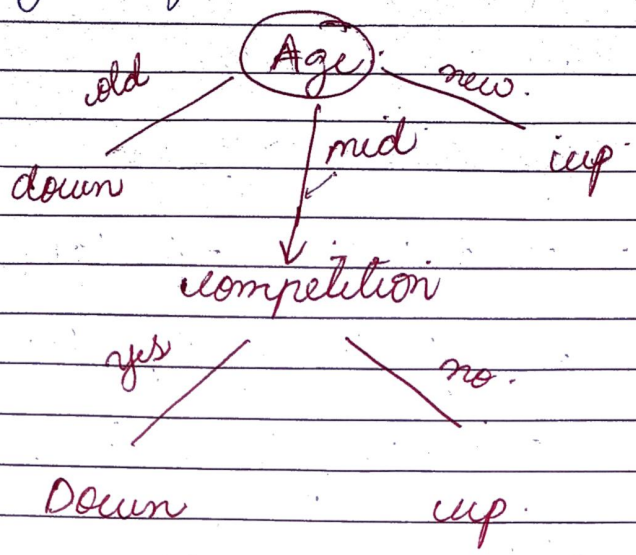
Entropy (Age) $= E(O) + E(M) + E(N) = 0 \cdot 0.4 + 0 = 0.4$

Step ④. Gain $= IG - E(A) = 1 - 0.4 = 0.6.$

For the same way, calculate gain of other attributes.

Gain (completion) $= 0.124.$
Gain (Type) $= 0.$
Gain (Age) $= 0.6.$

Highest gain → root node.



old → all down.
mid → some ↓ some ↑.
new → all up.

Why competition → next highest gain.
Why not other attribute (Type)

Type - gain = 0 ie, even in DT also, no requirement
∴ It can be ignored.

Appropriate problem for decision tree making.
DT algo is best suited to problems with the foll characters.

(Age → old, mid, new).

1. Instances are represented by attribute value pair.
2. The target function has discrete of values

__/__/__

ID_3 = DT.

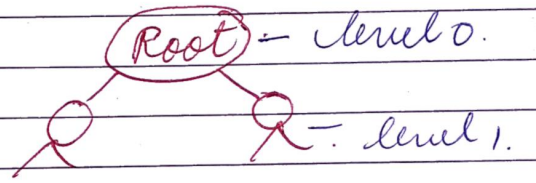
3. Training data can have errors
4. may contain missing attribute also

* HYPOTHESIS SPACE SEARCH IN DECISION TREE * LEARNING

ID_3 can be characterized as searching a space of hypothesis for one that fits training examples.

ID_3 → will search set of possible decision trees from available hypothesis.

ID_3 performs simple to complex searching.



First start with empty tree & keep on adding.

- every discrete valued (finite) function can be described by some decision tree.
- avoids major risk of searching incomplete hypothesis
- has only single current hypothesis
- can't determine alternative DT
- backtracking is not possible → (A*)
- can be extended easily to noisy data also.

* INDUCTIVE BIAS IN DECISION TREE ALGO.

- Set of assumptions.
Inductive bias of ID3 consists of describing the basis by which ID3 chooses one consistent decision tree over all possible DT's.

→ ID3 Search Strategy.
1. Selects in favour of shorter trees over longer ones.
2. Selects elements with highest IG as root attribute over lowest IG ones

→ Types of Inductive bias.
1. Restrictive bias - biased on conditions.
2. Preference bias - biased on priorities.

ID3 ⇒ Preference.
Vs CE ⇒ Restrictive.

→ Why short hypothesis?
According to OCCUMS RAZOR.
Prefer simplest hypothesis that fits the data.

* ISSUES IN DECISION TREE LEARNING.
1. Overfitting the data.
2. Incorporating continuous valued attributes.
3. Determining depth of tree for correct classification.

4. Handling attributes with different costs.
5. Alternative measures for selecting attributes.

ISSUES OF ML.

* What algo's exist for learning gtf from ste.
* What algos should be used
* Which algo is best for which type of problem.
* How much training data is sufficient.
* When & how does prior knowledge held by the learner, guide the process of generalising from examples?
* What is the best strategy for choosing next training example.
* How does the choice of this strategy alter the complexity of learning problem.
* What kinds of methods should be used
* What methods should be used to reduce learning overhead.
* For which type of data, which methods is used.
* What specific functions should the system attempt to learn? Can this process be automated
* How can the learner automatically alter its rep to improve its ability to rep & learn target func.

Version Space Representation Algo.

* Let X be an arbitary set of instances.
* and 'H' be a set of boolean valued hypothesis defined over X.
* Let $c: X \rightarrow \{0,1\}$ be an arbitary target concept defined over X.
* Let 'D' be an arbitary set of training examples $\{(x, c(x))\}$.
* For all X, H, c and D such that S and G are well defined.
* $VS_{H,D} = \{h \in H \mid (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$

The Basic Decision Tree Algo. (ID3)

* The core ID3 (Iterative Dichotomizer 3) algorithm is used for learning decision trees.
* It employs a top-down, greedy search through space of possible decision trees. (S.O.P)
* The ID3 algo learns decision trees by constructing them top-down beginning with the question "Which attribute should be tested at the root of the tree?
* ID3 uses Information gain measure to select among the candidate attributes at each step while growing the tree.

Entropy – Homogenity of Examples

* Entropy is a measure that characterizes the impurity of collection of examples. (IOCOE)
* In a collection of S containing +ve & -ve examples of some target concept, the entropy of S relative to the boolean classification is

$$\text{Entropy}(S) = P_{\oplus} \log_2 P_{\oplus} - P_{\ominus} \log_2 P_{\ominus}.$$

* where,

→ p^+ is the proportion of +ve examples in S

→ p^- is the proportion of -ve examples in S.

* In all calculations involving entropy, we define $0 \log 0$ to be 0.

Perceptrons

* One type of ANN System is based on a unit called as perceptrons.
* A perceptron takes the ~~values~~ vectors of a real valued inputs.
* Then, it calculates a linear combo of these inputs.
* Then it outputs 1 if the result is greater than some threshold & -1 otherwise.
* More precisely, if given inputs x_1 through x_n, the output of $(x_1 \ldots x_n)$ computed by the perceptron is

$$O(x_1 \ldots x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 \ldots w_n x_n \\ -1 \text{ otherwise} \end{cases}$$

Information Gain :- Measures the expected reduction in entropy.

* A measure called, Information gain simply measures the amount of reduction caused by partitioning the examples according to the given attribute.

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Here,

→ S is the collection of examples. COE

→ A is the given attribute values,

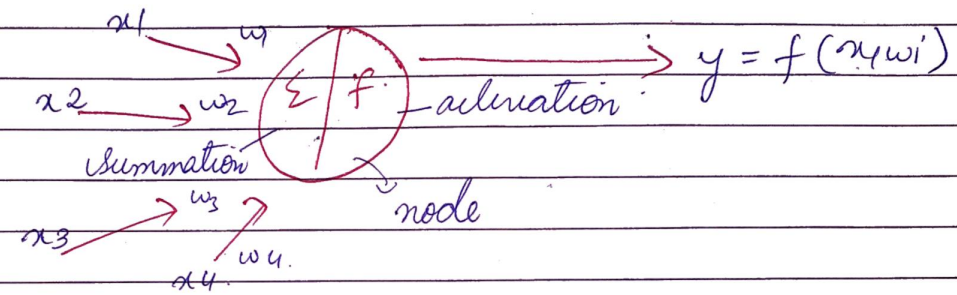
→ Values (A) is the set of all possible values for attribute A

→ S_v is the subset of S for which attribute 'A' has value v.

Unit 2:-

Artifical Neural Networks:-

Human brain - millions of neurons.
Biological → Neurons.
Formal → Nodes.
Natural functions are done artifically
Ex:- When we touch a hot vessel



where,
x_1, x_2, x_3, x_4, x_n are the input symbols.
w_1, w_2, w_2 are the weights of input symbols

Parts of a Node

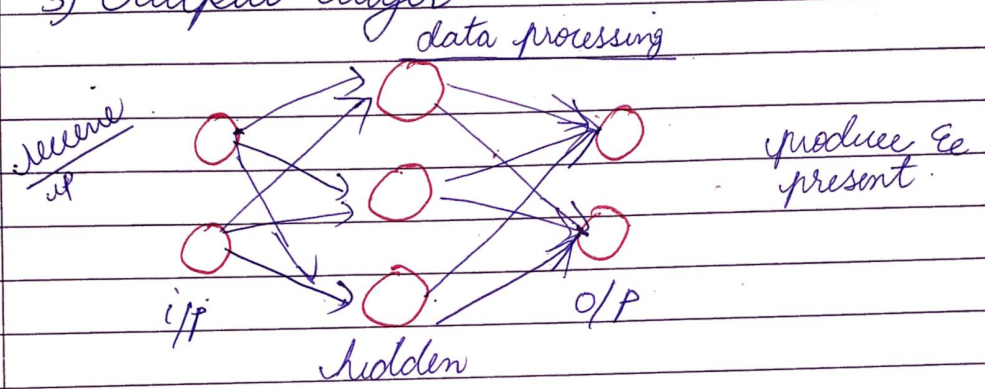
A node has 2 parts.

1). **Summation:-**
Calculates the weighted sum of all the inputs.
$= x_1w_1 + x_2w_2 + x_3w_3 + \dots x_nw_n$ $\quad \Sigma x_iw_i$
Once the weighted sum is calculated it is sent to activation function

2). **Activation function:**
It generates the o/p based on i/p given.

* Representation of Artificial Neural Networks.

ANN is divided into 3 parts.
1) Input Layer
2). Hidden layer.
3) Output layer



* Appropriate problems for Neural n/w learning.

(In which situations you can apply the

1). Instances have many attribute value pairs.
2) Target function has discrete values, continuous values or combination of both.
3) Training examples with errors or missing values.
4) Long Training times are acceptable.
5) Fast Evaluation of the target func learnt.
6) Ability for humans to understand the target function learnt by machine is not important.

_/_/_

* PERCEPTRON :-

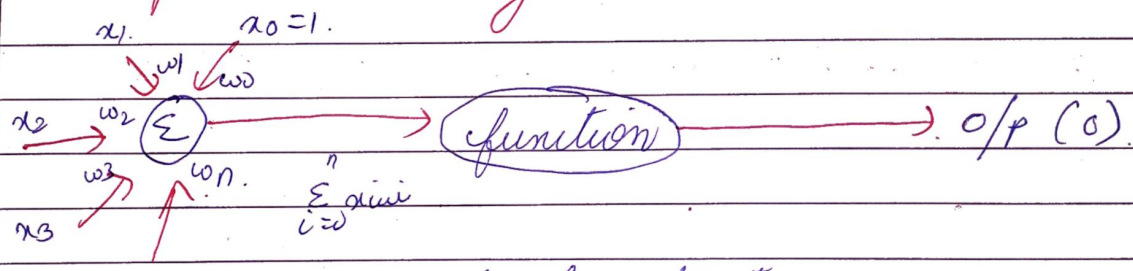
- basic unit used to build ANN.
- takes real valued i/p calculates linear combination of these i/p & generates output.

Output = 1 if result > threshold
-1 otherwise

Eg :- $O(x_1, x_2, x_3 \dots x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \dots w_n x_n > 0 \\ -1 & \text{otherwise} \end{cases}$

where $w_1, w_2, \dots w_n \rightarrow$ weights of inputs

Perceptron training Rule.



o = actual output
t = target output

if actual = target ⇒ weights are fixed.
otherwise ⇒ changed

How to change weights?

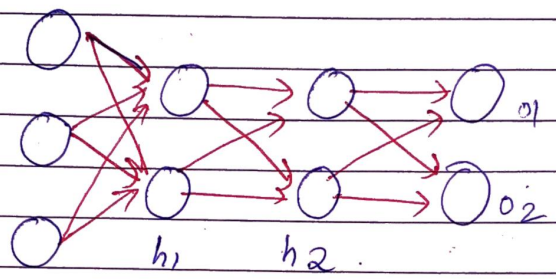
Initially, random weights.
Later, keep on applying iterations and check if (o = t)

Formula to change weight

$$w_i \leftarrow w_i + \Delta w_i$$
$$\Delta w_i = n(t - o)\, x_i$$

η = learning rate
t = target o/p
o = actual o/p
x_i = input associated with w_i.

* Multilayer Neural Networks.

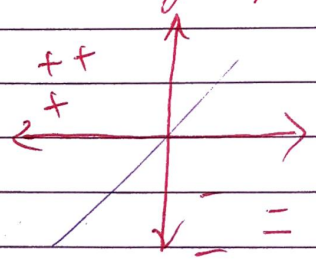


- has 2 hidden layers
It can also go back-back prop

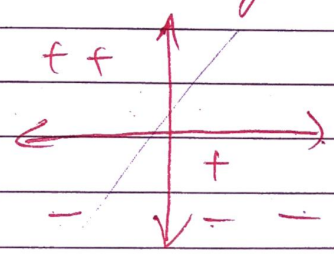
* Gradient descent and the Delta Rule.

Delta Rule → used for non linearly seperable data.
Perceptron training Rule → works good only for linearly seperable

Linearly seperable

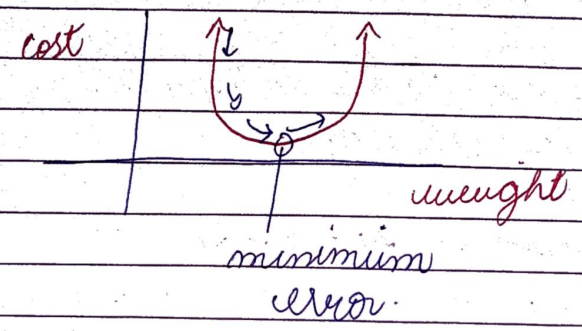


Non linearly seperable



Main Ideas of Delta Rule.

* Uses gradient descent rule to find out the best weight.



Gradient Decent Rule

How to modify weights

$$\boxed{\begin{aligned} w_i &\leftarrow w_i + \Delta w_i \\ \Delta w_i &= -\eta \nabla E(w) \end{aligned}}$$

$\nabla E(w) \rightarrow$ derivative of error w.r.t wts.
$\rightarrow$ also called gradient

$$\nabla E(w) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_n} \right].$$

$$\frac{\partial E}{\partial w_0} = \frac{\partial}{\partial w_0} \left(\frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \right)^2$$ $\Rightarrow$ Formula to find error

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

$$\frac{\partial E}{\partial w_i} = \frac{1}{2} \frac{\partial}{\partial w_i} \sum (t_d - o_d)^2.$$

$$\frac{d}{dx}(a^n) = n \cdot a^{n-1}$$

$$= \sum \frac{1}{2} \times 2 \times (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d).$$

$$= \sum (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - w_d x_d)$$

$$\frac{\partial E}{\partial w_i} = \sum (t_d - o_d)(-x_d)$$

$$\therefore \Delta w_i = -\eta \nabla E(w) = -\eta \times \sum (t_d - o_d)(-x_d)$$

$$\eta \in (td - od) \times d.$$

$$\therefore \Delta w_i = \eta \in (td - od) \times d$$
$$w_i \leftarrow w_i + \Delta w_i$$

* **BACK PROPAGATION ALGORITM**

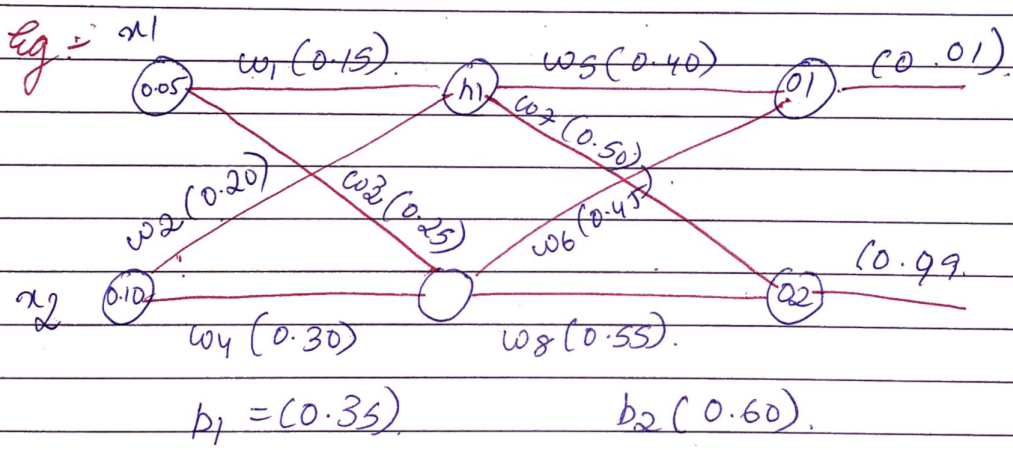
(backward propagation of error)
when error occurs, we go in backward direction
i.e output → hidden → input.

Part 1: Calculate forward propagation error

1) Calculate h_1 (in and out)

$$\boxed{h_1(in)} = w_1 x_1 + w_2 x_2 + b_1$$
$$= 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35$$
$$= 0.377$$

Eg :-



$b_1 = (0.35)$ $\qquad$ $b_2 (0.60)$.

$$\boxed{h_1(out) = \frac{1}{1 + e^{-h_1(in)}}} = \frac{1}{1 + e^{-(0.3777)}} = 0.5932$$

2) Calculate h_2 (in & out)

(i) $h_2(in) = x_1 w_3 + x_2 w_4 + b_1$

$= 0.05 \times 0.25 + 0.10 \times 0.3 + 0.35$

$= 0.3925$

(ii) $h_2(out) = \frac{1}{1+e^{-(0.3925)}} = 0.5968$

3). Calculate O_1 (in and out).

(i) $O_1(in) = h_1(out) \times w_5 + h_2(out) \times w_6 + b_2$

$= 0.593 \times 0.4 + 0.596 \times 0.45 + 0.6$

$= 1.105$

(ii) $O_1(out) = \frac{1}{(1+e^{-O_1(in)})} = 0.7513$

4). Calculate O_2 (in and out)

(i) $O_2(in) = h_1(out) \times w_7 + h_2(out) \times w_8 + b_2$

$= 0.5932 \times 0.50 + 0.5968 \times 0.55 + 0.6$

$= 1.22484$

(ii) $O_2(out) = \frac{1}{(1+e^{-O_2(in)})} = 0.7729$

5). Calculate ϵ total.

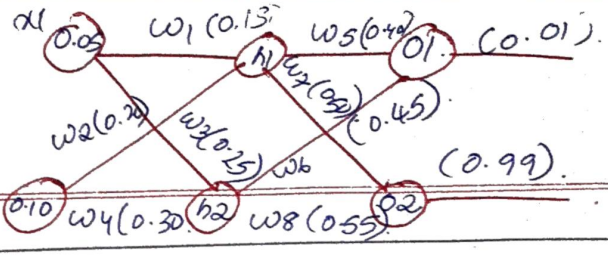
$\epsilon\ total = \Sigma \frac{1}{2}(target - o(p))^2$

$= \epsilon_{01} + \epsilon_{02}$

$= \frac{1}{2}(0.01 - 0.7513)^2 + \frac{1}{2}(0.99 - 0.7729)^2$

$= \frac{1}{2}(-0.413)^2 + \frac{1}{2}(0.2171)^2$

$= 0.274 + 0.0235 = \boxed{0.29837}$ (approx)



Part 2:

Calculating backward propagation error
(output layer → hidden layer).

w_5, w_6, w_7 & w_8

first let us adjust weight w_5.

η = learning rate 0.6

$$* \quad \boxed{w_5^* = w_5 - \eta \frac{\partial E_{total}}{\partial w_5}}$$

$$* \quad \boxed{\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}}$$

(i). $$\boxed{\frac{\partial total}{\partial out_{o1}} = out_{o1} - target_{o1}}$$

$$= 0.751365 - 0.01$$
$$= 0.7413565$$

(ii) $$\boxed{\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1})}$$

$$= 0.751365(1 - 0.751356)$$
$$= 0.186815602$$

(iii) $$\boxed{\frac{\partial net_{o1}}{\partial w_5} = out_{h1}}$$ (h1 out)

$$= 0.59326992$$

$$* \frac{\partial E_{total}}{\partial w_5} = 0.74135 65 \times 0.186815602 \times 0.59326992.$$

$$= 0.08216704.$$

$$w_5^* = w_5 - \eta \frac{\partial E_{total}}{\partial w_5}$$

$$= 0.4 - 0.6 \times 0.08216704 = 0.350699776.$$

do the same for w_6, w_7, w_8

Part 3: Calculating backward propagation of error.

(hidden $\rightarrow$ input layer.

(w_1, w_2, w_3, w_4)

First let's adjust w_1.

$$w_1^* = w_1 - \eta \frac{\partial E_{total}}{\partial w_1}.$$

$$\boxed{\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out\, h_1} \times \frac{\partial out(h_1)}{\partial net\, h_1} \times \frac{\partial net\, h_1}{\partial w_1}.}$$

$$\boxed{\frac{\partial E_{total}}{\partial out(h_1)} = \frac{\partial E_{o1}}{\partial out\, h_1} + \frac{\partial E_{o2}}{\partial out\, h_2}.}$$

$$\frac{\partial E_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial out_{h1}} \qquad \frac{\partial E_{o2}}{\partial net_{o2}} \times \frac{\partial net_{o2}}{\partial out_{h1}}$$

$$\downarrow \qquad\qquad \downarrow$$

$$\frac{\partial E_{o1}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \qquad \frac{\partial E_{o2}}{\partial out_{o2}} \times \frac{\partial out_{o2}}{\partial out_{o2}}$$

$$\frac{\partial E_{o2}}{\partial out_{o2}} = (out_{o2} - target\ o_2)$$

$$= 0.772928465 - 0.99$$

$$= -0.217071535$$

REMARKS ON BACK PROPAGATION ALGO:-

1). Convergence & local minima
2) Representational power of feed forward networks
3) Hypothesis Space Search & inductive bias
4) Hidden layer Representation.
5) Generalisation, overfitting & stopping criteria.

functions :- Boolean
Continuous.
Arbitary.

* AN ILLUSTRATIVE EXAMPLE: FACE RECOGNITION:-

→ Face recognition is a category of biometric software that maps individuals facial features & stores the data as face print
→ Machine learning & image processing are the backbone for FR

Image processing:-
* It involves the process of Computer Vision.
*(i) Image reading :- Reads any image

between 0 - 255.
For any image, there are 3 colors, RGB.

(ii) OpenCV :- Python library to solve computer vision

Machine Learning

* It takes dataset as input & learn from data identifies the pattern. (height, weight, color etc)
* ML does 3 major functions in FR.

(i) Deriving the feature vector.
(ii) Matching algos
(iii) Face Recognition operations.
- a) face detection
- b) face analysis.
- c) Image to data conversion
- d) Match finding.

Face Recognition Softwares

a) DeepVision AI.
b) Sense Time.
c) Amazon Recognition.
d) Face first.
e) True face
f) Cognitec.

* Applications

Hospital Security
Airline Industry etc.
Attendence System.
Phone unlock.

ADVANCED TOPICS IN NEURAL NETWORKS.

3 advanced topics.

1) Alternative error function :-

* Adding an extra penalty term to $E(w)$ can reduce the problem of overfitting.
* Each weight is multiplied with a constant in each iteration.

∴ We can redefine E as

$$E(\bar{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in op} (t_{kd} - o_{kd})^2 + \gamma \sum w_{ji}^2$$

2) Alternative Error minimization procedures

* Linear Search.
* Conjugate gradient descent method

3) Recurrent Neural Networks (RNN)

ANN that can be applied to time series data. It uses the o/p of n/w units at a time $\boxed{t}$ as input to other unit $\boxed{t+1}$

$(t) \xrightarrow{in} (t+1) \rightarrow (t+2)$

$\frac{1}{5}(1+1+1+1+1)$

$\frac{1}{5} \times 5 = 1$

n = no of samples

$f(x)$ = least target fun.

~~$h(x)$ = predicted fun by hypo~~

SAMPLE ERROR AND TRUE ERROR.

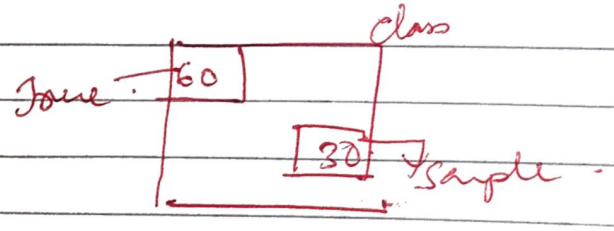
* **SAMPLE ERROR:-**
The error rate of hypothesis over a sample of data.

$$\text{Sample error} = \text{error}_S(h) = \frac{1}{n} \sum_{s \in S} \delta(f(x), h(x))$$

$$(f(x), h(x)) = 1 \text{ if } f(x) \neq h(x)$$
$$= 0 \text{ otherwise}$$

* **TRUE ERROR.**
The error rate of hypothesis over the entire distribution D

$$\text{True error} = \text{error}_D(h) = P[f(x) \neq h(x)]$$



EVALUATING THE HYPOTHESIS-

* evaluating the accuracy of hypothesis is the basis of ML.
* for evaluating the accuracy of hypothesis, we mainly focus on 3 Questions

1). Hypothesis is accurate over limited sample of data, then what about additional data.

2) We have 2 hypothesis - one is better than other when used on sample of data. Then will better hypothesis work good for general data also?

3) When we have limited data, what is the best way to use this data for both learning & estimation hypothesis?

* Motivation:-

→ Sometimes very precise hypothesis is required. In that cases estimating the accuracy is very important.

→ Example: Medical Treatment.

* Estimating Hypothesis Accuracy:-

Let us make some assumptions,

- There is some space of possible instances X over which various target functions may be defined.
- Different Instances will have different frequencies.
- Based on frequency, each instance in X will have some unknown probability.
- Trainer will teach the machine about the training examples of

target functions.

- Trainer will send a particular instance x_i along with target value $f(x)$
- Target function: $f: X \rightarrow \{0,1\}$
- Target function classifies each instance into different categories based on requirement.
- Now, after classification, what is the probable error in this estimation we made.

- Error - 2 types
  a) Sample error.
  b) True error.

BASICS OF SAMPLING THEORY

1). Random Variable :-
Name of an experiment with probabilistic outcomes.
Ex: $P(X=H) = 0.5$ $P(X=T) = 0.5$.

2) Probability Distribution:-
Probability distribution for a random variable Y $P(Y=y_i)$ that Y will take on value y_i for each possible y_i

$1/4 = 0.25$

{HH, TT, HT, TH}

Ex:- When 2 coins are tossed, let Y represents no of heads. $P(Y \leq 1)$
$= P(Y=0) + P(Y=1)$.
$= 0.25 + 0.5 = 0.75$

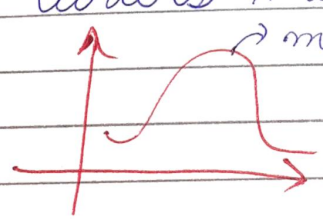
3) Expected Value or Mean:-
Sum of obs / Tot no of obs.
$E(Y) = \sum \underbrace{y_i}_{RV} \underbrace{P(Y=y_i)}_{PD}$ $\therefore$ RV × PD.

4) Variance:- Shows dispersion of data from its mean.
$Var(Y) = E[(\underbrace{Y}_{RV} - \underbrace{E(Y)}_{mean})^2]$

5) Standard Deviation:- $\sqrt{\text{Variance}}$
$= \sqrt{var(Y)}$

6) Binomial Distribution:-
Gives probability of observing r heads in a series of n indep coin tosses.
Ex:- finding prob of 2 heads in 3 tosses

7). Normal Distribution
Bell Shaped probability distribution that covers many natural values.



8) Central Limit Theorem:-
If you are having population with mean μ and SD as σ & you take large random samples from that population with replacement, then the distribution will approximately become normal.
Example 80, 40, 60.

9) Estimator:-
It is a random variable I used to estimate some parameter of an underlying population.
Ex:- $P(y = 50)$

10) Estimation Bias:-
Sample Mean $= \mu_y$.
Population Mean $= \mu_p$.
$EB = \mu_y - \mu_p$. $\therefore \mu_y - \mu_p = 0 =$ unbiased

11) Confidence Interval:-
Probability (P) at N%
(Area under ND curve in which data will fall 95% of the times)

DIFFERENCE IN ERROR OF TWO HYPOTHESIS

* Hypothesis Testing
We evaluate 2 mutually exclusive statements on population using sample data and decide upon population belongs to which statement

* Steps in hypothesis testing
1) make initial assumption (h0) - null hyp
2) collect data
3) gather evidence to accept or reject null hypothesis
(we should also assume an alternate hypothesis h1).
→ 2 errors are possible
(Type 1 and Type 2 error)

- Type 1 error
We implicitly know that h0 is correct but we do not have enough evidence to prove that
then automatically h1 is taken as correct

→ Type 2 error
When we do not have enough evidence to prove the truth of h0, without any choice we approve h1.
We dont know about truth of h0 ie h1

	h_0	h_1
do not reject	OK	Type 2 error
Reject	Type 1 error	OK

* COMPARING LEARNING ALGO'S.

We have so many algorithms in ML. There is always a confusion about which one to choose.

∴ We consider some factors.

1) Time Complexity – training & testing
2) Space Complexity
3) Sample Complexity – no of examples
4) Unbiased data – 60 – 30K, 70
5) Online & Offline algo's. Online data updated atio.
6) Parallelizability – multiple operation – share work load.
7) Parametricity – fixed parameter (with size). Non parametric – changes with size

BAYES THEOREM:

Bayes Theorem gives probability of an event based on prior knowledge of conditions

$$P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}$$

$P(B/A)$ → likelihood

$P(A)$ → hypothesis

$P(B)$ → marginal

$P(A)$ → Prior

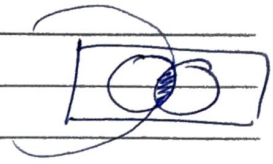
Proof:- let us take 2 events A & B.

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

P(A/B) - conditional prob

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

$$P(A/B) \cdot P(B) = P(A \cap B) \quad \text{--- (1)}$$
$$P(B/A) \cdot P(A) = P(B \cap A) \quad \text{--- (2)}$$



$$P(A/B) \cdot P(B) = P(B/A) \cdot P(A)$$
$$P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)} \longrightarrow \text{Bayes}$$

Terms Explanation:-

A = hypothesis
B = given data.

P (A/B) = finding prob of hypothesis when prob of training examples are given.

P(B/A):- finding prob of given data provided with prob of hypothesis that is true.

P(A) = Probability of hypothesis before observing given data.

P(B). = Probability of given data.

Face - J, K, Q.
4x3=12.

Example:- P(King / face).

$$= \frac{P(\text{Face}/\text{King}) \cdot P(\text{King})}{P(\text{Face})}$$

$$= \frac{\frac{1 \times 4}{52}}{\frac{12}{52}} = \frac{\frac{1 \times 1}{13}}{\frac{3}{13}} = \frac{1}{3}.$$

$P(K/F) = 1/3$.

BAYES THEOREM AND CONCEPT LEARNING.

What is the relation between Bayes Theorem & concept learning?

→ Bayes theorem calculates the probability of each possible hypothesis & outputs the most probable one.

* Brute force Bayes Concept Learning.

1) For each hypothesis in H, calculate posterior probability.

$$P(h/D) = \frac{P(D/h)\,P(h)}{P(D)} \quad - (1)$$

2). Output the hypothesis h_{MAP} (h_{ML}) - maximum likelihood with highest priority

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D)$$

To calculate we need values of $P(h)$. & $P(D|h)$.

Some assumptions:-

1) Training data D is noise free.
2) Target concept c is present in hypothesis space H.
3) We have no prior reason to believe that any hypothesis is more probable than any other.

$$P(h) = \frac{1}{|H|} \text{ for all } h \text{ in } H.$$

$$P(D|h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases}$$

~~From ①~~

$$P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)} \quad \text{— ①}$$

From ①.

case 1: Hypothesis h is inconsistent i.e. $P(D|h) = 0$.

$$P(h|D) = \frac{0 \times P(h)}{P(D)} = 0.$$

case 2: Hypothesis is consistent i.e. $P(D|h) = 1$.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} = \frac{1}{\frac{|VSH,D|}{|H|}} = \frac{1}{|VSH,D|}.$$

$$= \frac{1 \times P(h)}{P(D)}$$

$$\therefore P(h|D) = \begin{cases} \frac{1}{|VSH,D|} & \text{if } h \text{ is consistent with } D. \\ 0 & \text{otherwise.} \end{cases}$$

(If its inconsistent it is '0')

MAXIMUM LIKELIHOOD AND LEAST SQUARED ERROR HYPOTHESIS -

To find the maximum likelihood hypothesis in Bayesian learning,

$$h_{ML} = (h_{MAP}) = \underset{h \in H}{\operatorname{argmax}} P(h|D)$$

let us take training instances $(x_1, x_2 .. x_n)$ and consider target values $D = (d_1, d_2, .. d_m)$

$\therefore$ we write $P(D/h)$ as product of $P(d_i/h)$.

$$h_{MAP} = \text{argmax} \prod_{i=1}^{m} P(d_i/h).$$

By assuming normal distribution

$$\boxed{f(x/\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$$

$$h_{MAP} = \underset{h \in h}{\text{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(d_i-\mu)^2}.$$

$$= \underset{h \in H}{\text{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(d_i-h(x_i))^2}$$

$$= \underset{h \in H}{\text{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(d_i-h(x_i))^2}$$

→ Use ln function:

$$= \underset{h \in H}{\text{argmax}} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

$$= \underset{h \in H}{\text{argmax}} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

$$= \underset{h \in H}{\text{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2.$$

$\therefore h_{MAP}$ or $h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2$

least squared error.

MINIMUM DESCRIPTION LENGTH PRINCIPLE:-

Assumption:-

→ Representing a concept in minimum possible way.

→ then it is said to be good one.

Mathematically,

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(D/h) P(h)$$

Applying logarithm,

$$\underset{h \in H}{\operatorname{argmax}} \log P(D/h) + \log P(h)$$

$$\underset{h \in H}{\operatorname{argmin}} - \log P(D/h) - \log P(h)$$

(minimum length / short hypothesis is prefered)

Example:-

1) Let us consider a probability of designing a code to transmit messages drawn at random from a set D.
Where probability of drawing an ith msg = P_i.

2). While transmitting, we want a code that minimizes the expected no of bits.

To do this we should assign shorter codes to the most probable

We represent the length of message i w.r to c as $L_c(i)$

$$\therefore h_{MAP} = \arg\min L_{C_H}(h) + L_{C_{D/h}}(D/h)$$

$C_H \rightarrow$ optimal encoding for H.
$C_{D/H} \rightarrow$ " " " D given H.

$$\boxed{\therefore h_{MDL} = h_{MAP}}$$

BAYES OPTIMAL CLASSIFIER:-

Bayes Optimal classifier is a probabilistic model that makes the most probable prediction for a new example.

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

for a dataset,

yes/No.

$$x = \{x_1, x_2, x_3 \cdots x_n\}\{y_n\}$$

$$P(xy/x_1, x_2 \ldots x_n) = \frac{[P(x_1/y) \cdot P(x_2/y) \cdots P(x_n/y)] \times P(y)}{P(x_1) \cdot P(x_2) \cdots P(x_n)}$$

$$= \frac{P(y) \prod_{i=1}^{n} P(x_i/y)}{P(x_1)\,P(x_2) \cdots P(x_n)}$$

ML cont.

NAIVE BAYES CLASSIFIER

classification technique based on Bayes theorem with an assumption of Independence among features

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

Fruit = { Yellow, Sweet, long }

Fruit	Yellow	Sweet	long	Total
Orange	350	450	0	650
Banana	400	300	35	400
Others	50	100	50	150
Total	800	850	400	1200

$$P(Y/O) = \frac{P(O/Y)\,P(Y)}{P(O)} = \frac{\frac{350}{800} \times \frac{800}{1200}}{\frac{650}{1200}} = 0.53$$

$$P(S/O) = \frac{P(O/S)\,.\,P(S)}{P(O)} = \frac{\frac{450}{850} \times \frac{850}{1200}}{\frac{650}{1200}} = 0.69$$

$$\frac{0.529 \times 0.70}{0.541}$$

$$P(L/O) = 0$$

$\therefore$ P(fruit / orange) =

$P(Y/O) \times P(S/O) \times P(L/O)$

$= 0.53 \times 0.63 \times 0 = 0.$

$$P(Y/B) = \frac{P(B/Y)\,P(Y)}{P(B)} = \frac{\frac{400}{800} \times \frac{800}{1200}}{\frac{400}{1200}} \quad \frac{0.5 \times 0.66}{0.33}$$

$$P(S/B) = \frac{P(B/S)\,P(S)}{P(B)} = \frac{\frac{300}{850} \times \frac{850}{1200}}{\frac{400}{1200}} \quad \frac{0.35 \times 0.70}{0.33}$$

$$P(L/B) = \frac{P(B/L)\,P(L)}{P(B)} = \frac{\frac{35}{400} \times \frac{150}{1200}}{\frac{400}{1200}} \quad \frac{0.09}{0.33}$$

P (Fruit / Banana) $= P(Y/B) \times P(S/B) \times P(L/B)$

$= 1 \times 0.75 \times 0.89 = 0.65$

P (Fruit / others) $= P(Y/O) \times P(S/O) \times P(L/O)$

$0.072 =$

5* GIBBS ALGO

* Although Bayes Optimal classifier obtains the best performance, it is costly as we have computing the posterior probability for every hypothesis in M & then combine the predictions of each hypothesis to classify each new instant.
* Gibbs Algo is an alternative to Bayes Optimal classifier but less optimal method

Gibbs Algo

1) * Choose a hypothesis h from M at random according to posterior probability distribution over M.
* Use h to predict the classification of next instance

$$2[\text{Error Gibbs}] \leq 2E[\text{error Bayes OBTained}]$$

* Limitation :- The expected value of error of the gibbs algo is at most twice the expectation value of error of Bayes Optimal classifier

LOCALLY WEIGHTED REGRESSION.

* The phrase locally weighted regression is called local because the function is approximated based only on data near the query point
* Weighted because the contribution of each training example is weighted by its distance from the query point & regression for approximating real valued function
* LWR is generalization of KNN.
* It constructs an explicit approximation to f over the local region corresponds to query point

Regression :-

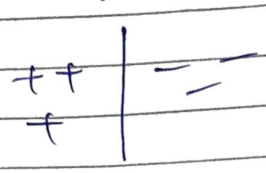
Statistical tool used to understand & quantify the relation b/w 2/ more variables.

Linear Regression :-

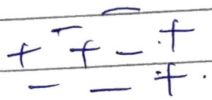
$$y = \beta_0 + \beta_1 x + \varepsilon$$

→ Best suited for linearly separable data only

y = dependent variable.
x = Independent variable.
β_0 = constant / Intercept.
β_1 = x-slope / co-eff.
ε = error.



But we want separate



So we use Locally weighted Regression

* To overcome the problem of non linearly Separable data.
* LWR algo assigns weight to data to overcome the problem
* Computationally more expensive.
* The distance from the predicting value increases the weight assigned to that value will decrease.
* Dist ↑ wt ↓.

finding weights → Kernel Smoothing.

$$D = ae^{\frac{-\|x - x_0\|}{2c^2}}$$

x = each training i/p.
x_0 = value predict

* we construct a weight matrix (W).
for each training i/p (x) & for the
value we are trying to predict (x_0)
* Weighted matrix - diagonal matrix

$$\beta = (x' W x)^{-1} x^{-1} W y.$$

β = model parameter.

then prediction can be defined as

$$y = \beta . x_0, \quad y \Rightarrow \text{prediction}$$

Drawbacks

* Need to evaluate the whole dataset everytime.
* Computation cost is more.
* Memory requirement is more

6* KNN - Algo.

* ~~the~~ Sample for training data.

Ex.
Given data Query $\Rightarrow x = (\text{maths} = 6, CS = 8)$
and $k = 3$.
Classification - Pass / Fail

	Maths	CS	Result
1)	4	3	F
2)	6	7	P
3)	7	8	P
4)	5	6	F
5)	8	8	P

∴ 3P / 0F

Euclidean distance (d).

$$d = \sqrt{|x_0 - x_{A_1}|^2 + |x_{02} - x_{A_2}|^2}$$

O = observed value
A = actual value

1) Calculate $d_1 = \sqrt{(6-4)^2 + (8-3)^2}$

$= \sqrt{2^2 + 5^2} = \sqrt{29} = 5.38$.

2) $d_2 = \sqrt{(6-6)^2 + (8-7)^2} = \sqrt{0+1} = 1$

3) $d_3 = \sqrt{(6-7)^2 + (8-8)^2} = \sqrt{1+0} = 1$.

4) $d_4 = \sqrt{(6-5)^2 + (8-5)^2} = = 3.16$.

5) $d_5 = \sqrt{(6-8)^2 + (8-8)^2} = 2$.

* Now choose 3 nearest neighbor which must be as mini as possible
* Now check table, as its pass so given query is evaluated to pass

GA

REMARKS ON LAZY AND EAGER LEARNING.

* Lazy Learning.

→ Simply stores the training data & waits until it gets a test tuple.
→ less training time more prediction time
→ All instance based learning algo.

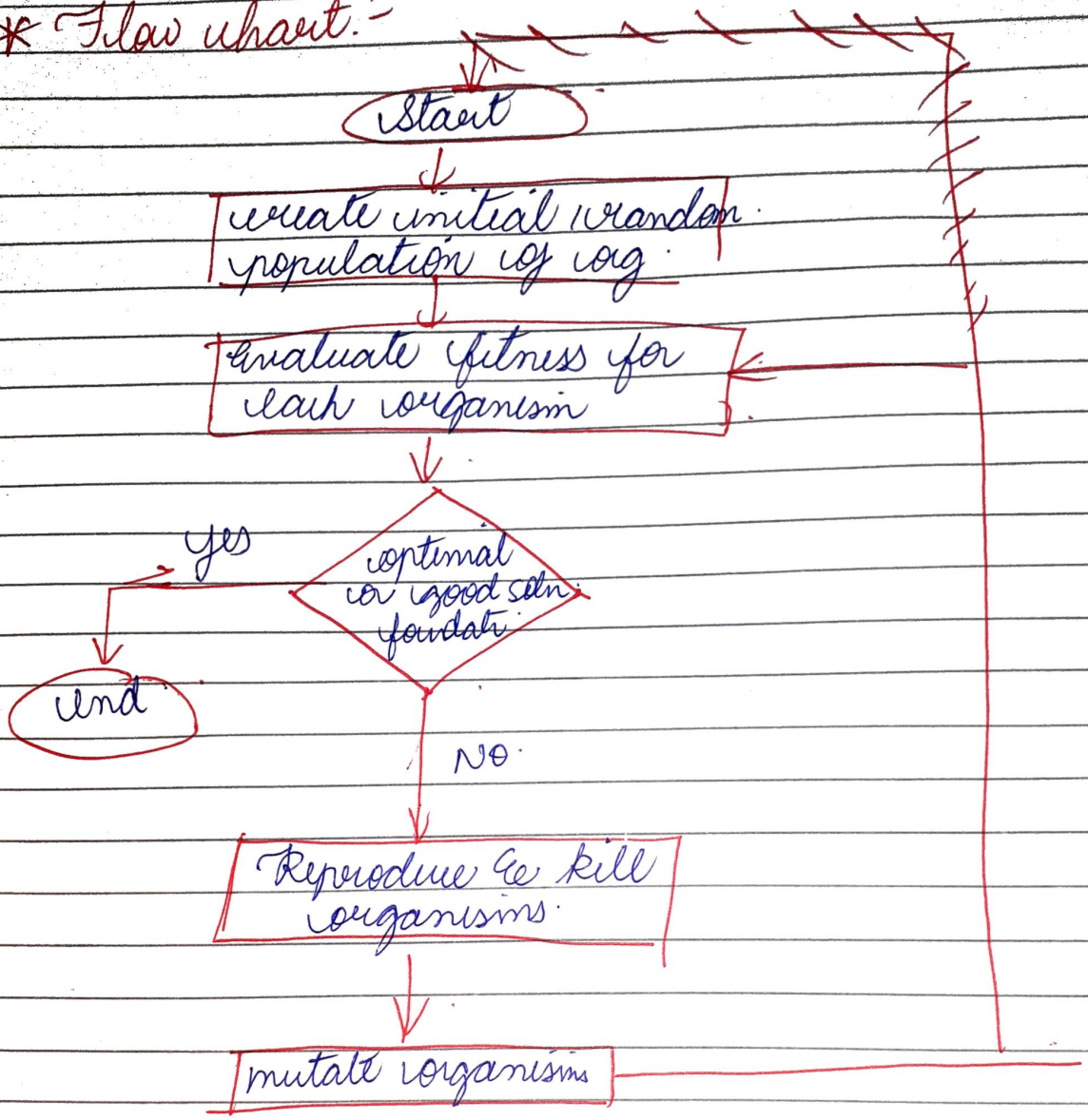
Eager learning:

→ When we give a training set, it constructs a model for classification before getting new example.
→ more training time less prediction time
→ Eg Decision tree, Naive Bayes, ANN

GENETIC ALGO

* They belong to evolutionary Algo.
- adaptive heuristic search Algo.
- based on genetics & natural selection.
- used to generate high quality solution for an optimization problem

* Flow chart:-



* Operations:
1) Selection
2) Crossover
3) Mutation
4) Encoding

* EXAMPE For Genetic Algo.

$f(x) = x^2$, maximize this function with x in interval [0, 31].

1) Generate initial population at random (called Genotypes)

01101 (13), 11000 (24), 01000 (8), 10011 (19)

$\boxed{N=4}$

2) Calculate fitness

$f(x) = x^2$

13 → 169, 24 → ~~286~~ 276, 8 → 64, 9 → 36.

3) Select any two parents based on fitness.

$$P_i = F_i / \sum_{j=1}^{n} F_j$$

No	Initial population	x value	$F(x)=x^2$	P_i	exp. to occ. $=N \times P$
				(169/1170)	(4×0.14)
1	01101	13	169	0.14	0.56
2	11000	24	576	0.49	1.97
3	01000	8	64	0.06	0.22
4	10011	19	36	0.31	1.23
			1170	1.00	4.00

1.97 has highest values but to do crossover 2 values r needed

also strike eg 0.22 & replace with 1.97

4). Crossover :-

→ can be either one point / 2 point / n point.

eg :- 100 | 111 | 01 = 10001011.
101 | 010 | 11. 10111101.

No).	Initial	crossover point	After crossover	x	$f(x)=x^2$.
1	01101.	4.	01100. (8 4 2 1)	12.	144.
2	11000.	4.	11001. (16 8 4 2 1)	25	625 max
3	11000	2	11011.	27.	729 max.
4.	10011.	2.	10000. (16 8 4 2 1)	16.	256
					1754

2) Mutation :-

applied to each child after crossover.

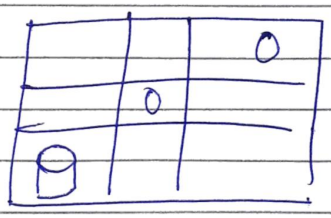
No)	Initial	After mutation	x.	$f(x)=x^2$.
1	01100	11100.	26.	676.
~~2~~	11001	11001	25.	625.
~~3~~	11011	11011.	27.	729.
4	10000	10100.	18.	324
				2354.

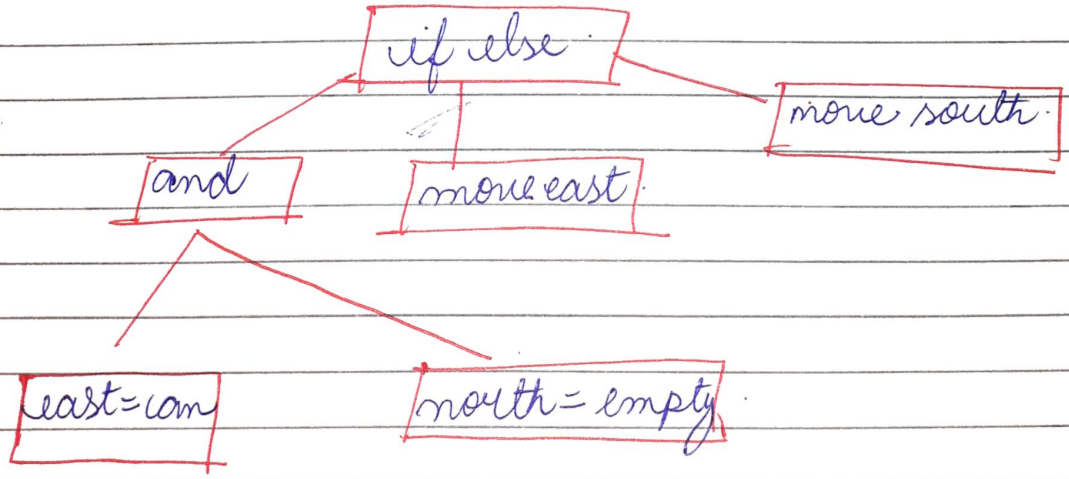
GENETIC PROGRAMMING.

* extension of genetic algorithms.
* main idea - represent a computer program as a tree.
* used when exact solution is not known in advance.
* All operations are same as genetic algo
* Eg:- Plastic can collecting Robot Navigation.

```
if (east = can & north = empty).
    then move east
else move south
```

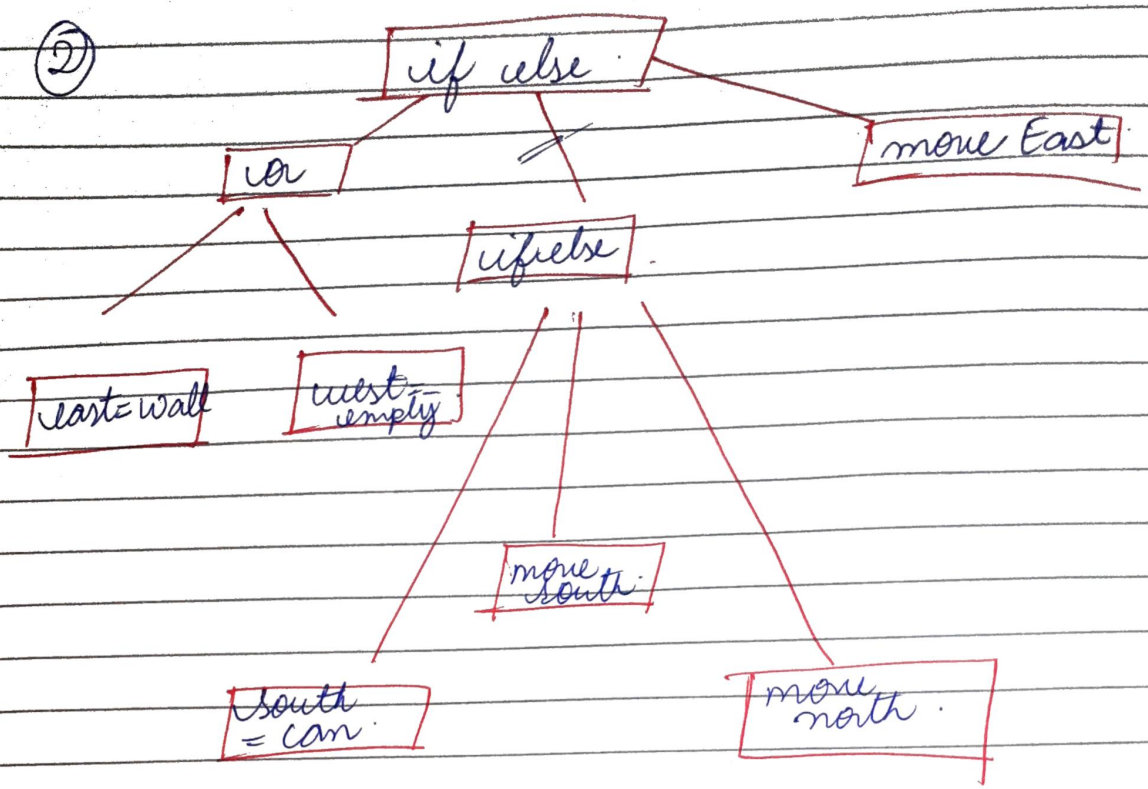
①



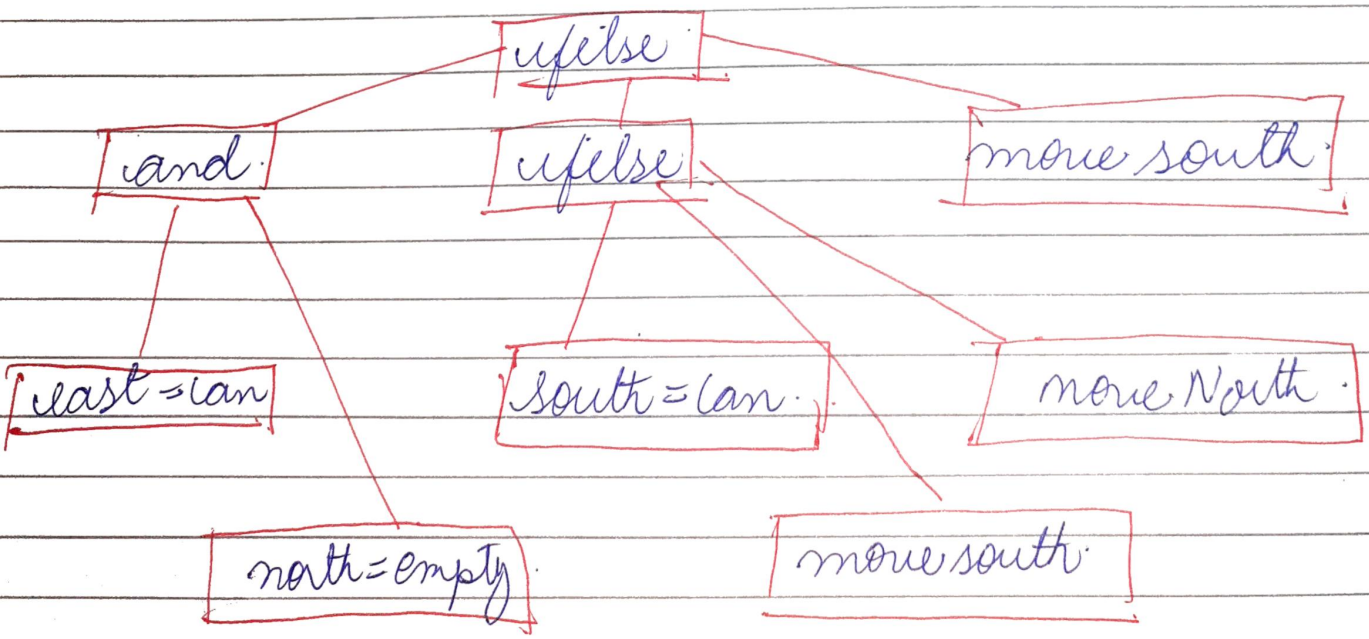


Neb tree.

(2)



Crossover resultant tree.



* MODELS OF EVALUATION & LEARNING.

2 evaluation models.

1) Lamarckian evolution.
2) Baldwin effect.

1)

* Believed that individual genetic makeup is changed by lifetime exp. i.e if an organism changes during its life to adapt to environment, then those changes are passed to its offsprings.

* Eg:- Giraffe.

2).

* Baldwin explained about the learning behavior

[2 things]

i) Genotype – genetic code (DNA).
– global search.

(ii) Phenotype – your characteristics (behavior)
– local search

* measures cost of learning not in terms of money but in terms of time & energy.

Ex:- fish

PARALLELISING GENETIC ALGO.

Parallel genetic algo uses multiple algorithms to solve a single task.

2 categories.

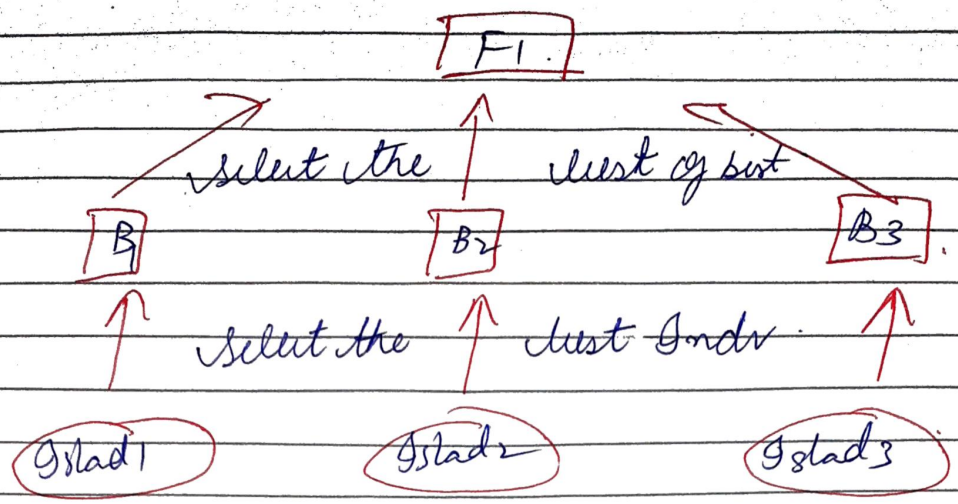
1) Fine-Grained.
detailed description which deals with much smaller components.

2) Coarse-grained.
* divides into fewer components.
(size of component is more than that of fine-grained).
* all the algo's solve the same task & once they obtain soln best one is selected.
* also called as island model.
* They do not depend on CPU
*

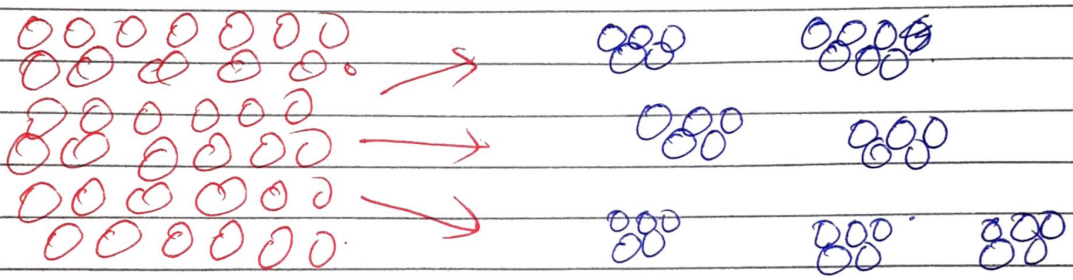
* Adv:- avoid prob of running

They can run parallely

eg:



eg2.



LEARNING SET OF RULES

2 ways.

① way:- first learn decision tree & binarize that tree into rules, one rule for each leaf node.

② way:- use a genetic algo that encodes each rule as a bit string.

* We have 2 types of algos that learn directly from set of rules

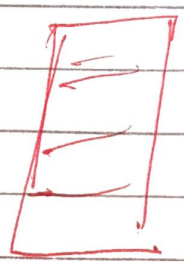
1) First Order rules
2) Sequential covering algo.

* Sequential Covering Algo:-

→ It Sequentially covers each & every rule.
→ extracts rules from dataset directly.
→ It is mainly based on one of evaluation measure.
  a) accuracy - confidence factor.
  b) coverage - cover most of Training examples

* Algorithm:-

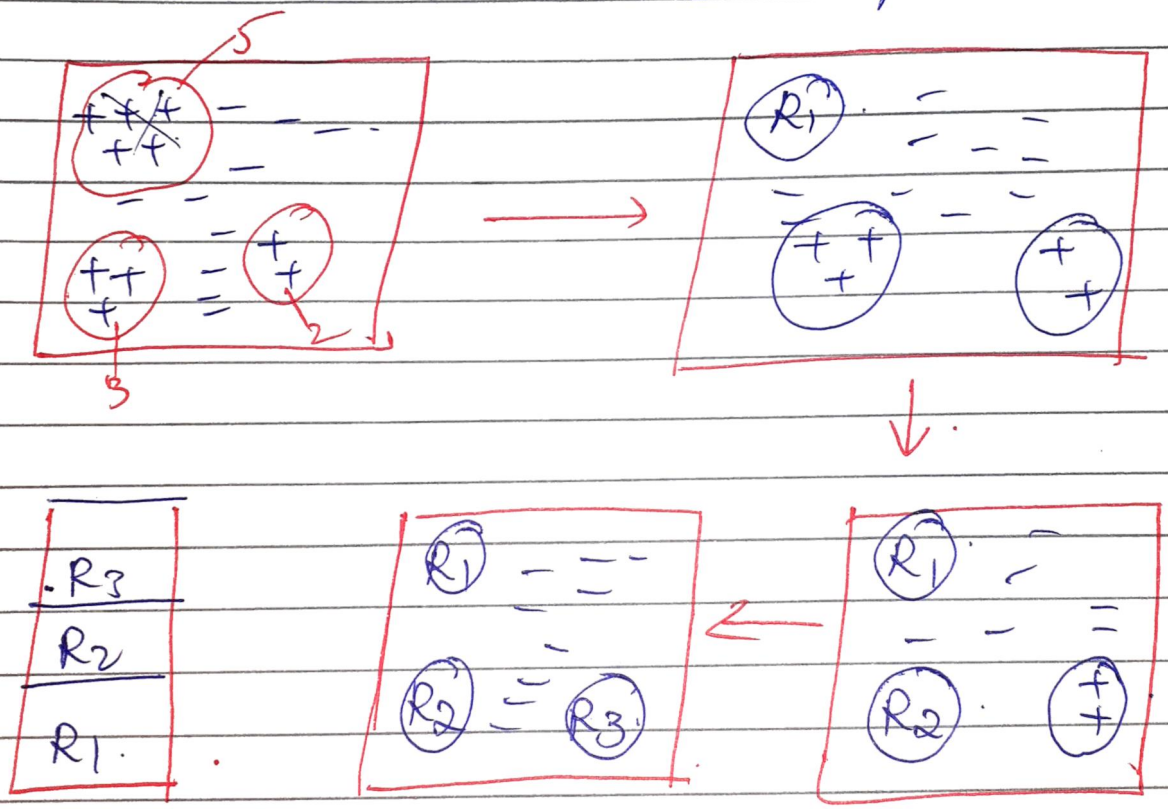
1) It creates an empty set of decision list (R)
2) A function called "learn One Rule" function is used, it extracts best rule for class "y"
   If all training records ∈ class y ⇒ +.
   If all training records ∉ class y ⇒ −



3) Get desirable values (only +ve).
4) Eliminate records (i.e the desirable ones).
5) New rule is added to the bottom of R

Example:-

Let us take a dataset with both +ve and -ve values. Samples



* First Order Learning Algo.

→ implemented with the help of PROLOG. (uses horn clauses) right with.

→ First order logic is much expensive than propositional logic

→ It allows a finer grain of specification & reasoning.

Example : daughter(x, y)

where (x, y) are {Name, mother, father, male, Female}.

Training examples are represented as.
(person 1, person 2, target attribute)

P1 ((name1 = Ann, mother1 = Sue, Father1 = Bob, male1 = F, Female1 = T)

P2 ((name2 = Lola, mother2 = gill, Father2 = Joe, male2 = T, Female2 = F).

TA = daughter1,2 = T

But, with this example only few rules would be defined

if (father$_1$ = Bob) $\wedge$ (Name2 = Bob) $\wedge$ (Female$_1$ = T)

then daughter$_{1,2}$ = T.

from first order rule,

If father (y, x) $\wedge$ Female (y) then daughter(x, y)

Terms in Expressions :-

Constants → Bob, 23.
Variables → x, y, z
predicate Symbols → Female, Father (only T/F)
Function Symbols → age. (only const)
Connectives ($\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$).
Quantifier → ($\forall$, $\exists$)