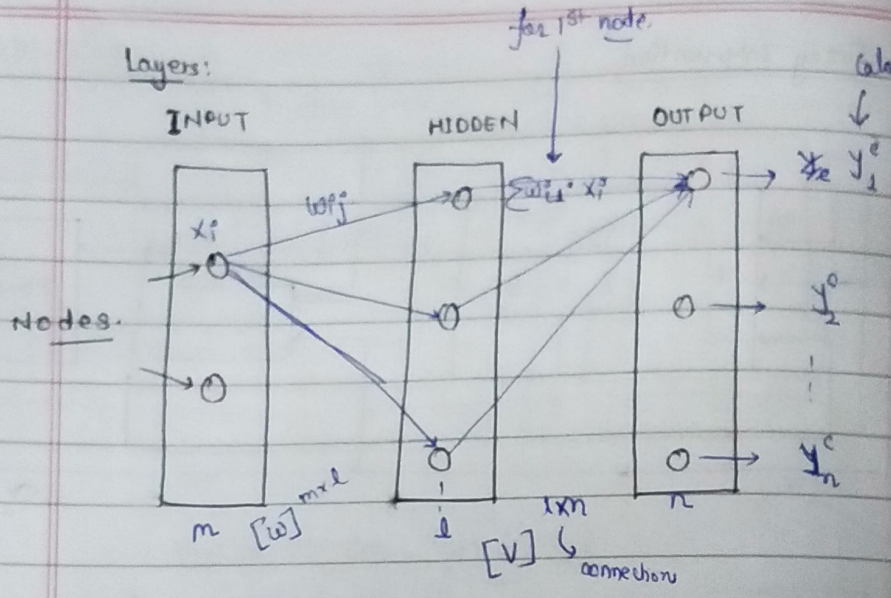
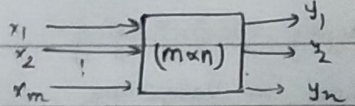
Neural Networks :-



$x_1, x_2 \ldots x_m$; $(y_1, y_2 \ldots y_n)$ → Actual Output



m - input n - output.

dim [w] = m x l

w_{ij} : ith node of input is connected with jth node of hidden.

↓

'weight'

dim [v] = l x n

V : weight for connection b/w Hidden and Output.

$$\text{Error} = \frac{\sum (y^c - y)^2}{n}$$

E (w, v). w, v - Arbitrarily chosen

Error should minimize.

Δw, Δv

$w_{i+1} = w_i + \Delta w_i$

Gradient Descent method :- $\frac{\partial E}{\partial w} = 0$ $\frac{\partial E}{\partial v} = 0$

↳ Optimum Value of w & v.

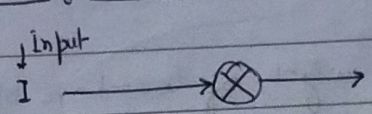
↳ Randomly Sampled (no bias) } 1000 data → 2/3 | 1/3 part divided randomly

↳ training the ~~Err~~ network : Error is very less.

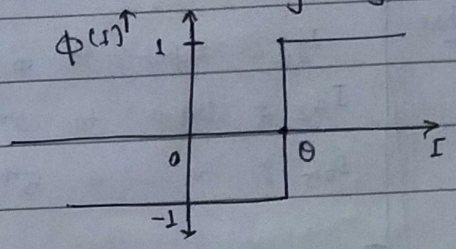
↳ testing

Artificial Neural Network

Thresholding

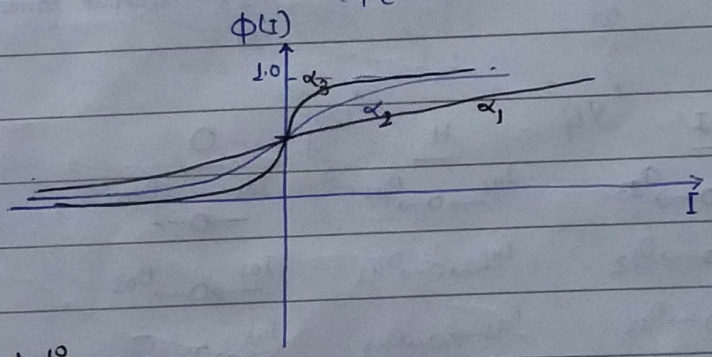


(I) Signum fn.



(II) Sigmoidal, $\phi(I) = \frac{1}{1+e^{-\alpha I}} = \frac{1}{1+e^{-\alpha(I-\theta)}}$

↑ thresholding ~~value~~ value.

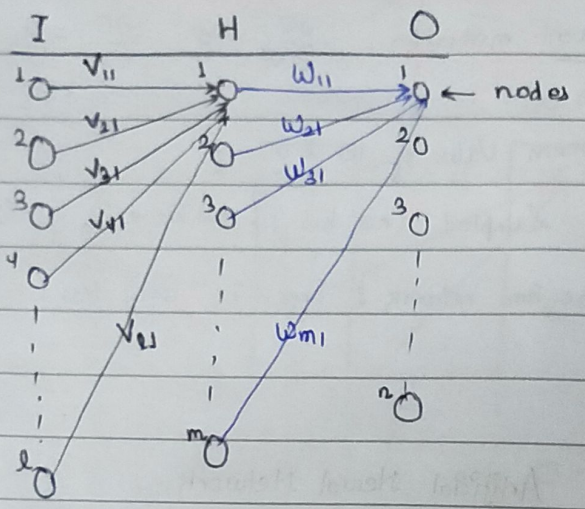


$\alpha_1 < \alpha_2 < \alpha_3$

(III) Hyperbolic, $\phi(I) = \tanh(I)$

hidden layer

$l \times m \times n$

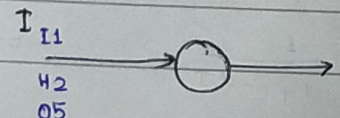


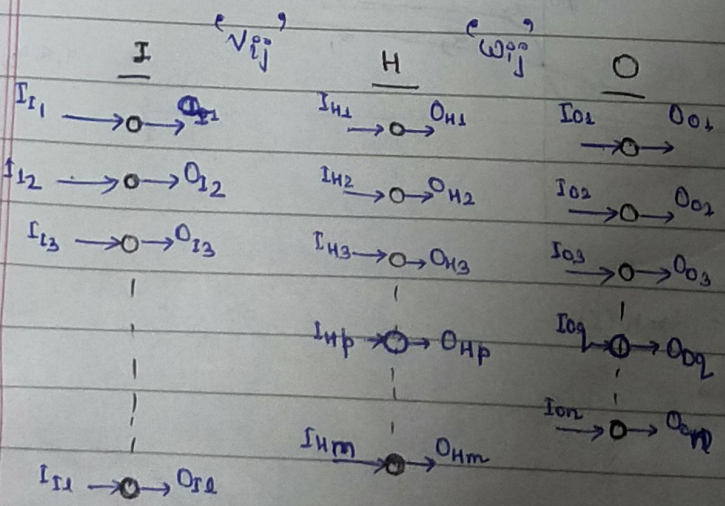
V_{ij} = weight of connection b/w i^{th} Input node with j^{th} hidden node.

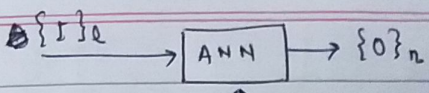
W_{ij} = weight of connection b/w i^{th} hidden node with j^{th} output node

nomenclature,

- I_{I1} : Input to the 1st node of Input layer
- I_{H2} : Input to 2nd node of hidden layer
- I_{O5} : " " 5th node of output layer







↑
Black box (All process 'inside').

Let

INPUT LAYER 1. # for input layer, transfer function is $I_{Ii} = O_{Ii}$ (i^{th} node).

$\therefore \quad I_{I_1} = O_{I_1} \; ; \quad I_{I_2} = O_{I_2} \; ; \; \ldots \; ; \; I_{Il} = O_{Il}$

$$\underset{l\times 1}{\{O\}_I} = \underset{l\times 1}{\{I\}_I} \leftarrow \text{Input layer.}$$

HIDDEN LAYER 2. Input, $I_{H1} = V_{11} O_{I1} + V_{21} O_{I2} + V_{31} O_{I3} + \ldots + V_{l1} O_{Il}$

for pth node, in hidden layer.

$$I_{Hp} = V_{1p} O_{I1} + V_{2p} O_{I2} + V_{3p} O_{I3} + \ldots + V_{lp} O_{Il}$$

$$I_{Hp} = \sum_{j=1}^{l} V_{jp} O_{Ij}$$

$p = 1, 2, 3, \ldots, m$ (for all nodes of hidden layer)

Matrix form $\Rightarrow$ $\underset{m\times 1}{\{I\}_H} = \underset{m\times l}{[V]^T} \cdot \underset{l\times 1}{\{O\}_I}$

HIDDEN Output, for pth Node of hidden layer,

$$O_{Hp} = \frac{1}{1 + \exp\left(-a\left(I_{Hp} - \theta_{Hp}\right)\right)} \quad \text{(Sigmoidal)}$$

$$\underset{m\times 1}{\{O\}_H} = \left\{ \frac{1}{1 + \exp\left(-a\left(I_{Hp} - \theta_{Hp}\right)\right)} \right\}_{m\times 1}$$

$p = 1, 2, \ldots m$.

θ_{Hp} = initially all same.

OUTPUT LAYER

3. Input,

$$I_{o1} = w_{11} O_{H1} + w_{21} O_{H2} + \dots + w_{m1} O_{Hm}$$

matrix form :- Input to Output layer,

$$\underset{n\times 1}{\{I\}_o} = \underset{n\times m}{[w]^T} \underset{m\times 1}{\{O\}_H}$$

$$I_{oq} = w_{1q} O_{H1} + w_{2q} O_{H2} + \dots + w_{mq} O_{Hm}$$

Output,

$$O_{oq} = \frac{1}{1+\exp(-\alpha(I_{oq} - \theta_{oq}))}$$

$$\underset{n\times 1}{\{O\}_o} = \left\{ \frac{1}{1+\exp(-\alpha(I_{oq} - \theta_{oq}))} \right\}$$

$q = 1, 2, \dots, n$

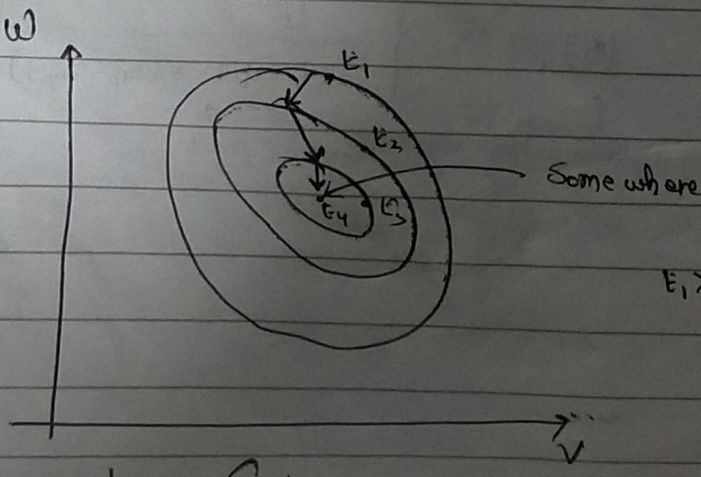
↑ Forward Calculation of training Data Set.

$\underset{n\times 1}{\{O\}_o}$ will be little different from Actual values of Output $\{O\}$

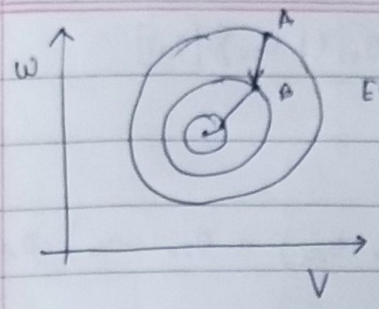
So, to reduce Error, $\frac{\partial E}{\partial w} \simeq 0$ $\frac{\partial E}{\partial v} \simeq 0$ for particular (v, w)

↑ $E(v, w)$ – So for minm E, we need particular, w & v.

Contour plot:-



by Gradient Descent we come to minm Error point.



$E(V, \omega)$

$V_{i+1} = V_i + \Delta V$

$$\bar{a} = \frac{\partial E}{\partial V}\hat{i} + \frac{\partial E}{\partial \omega}\hat{j}$$

$$\bar{AB} = -\eta \cdot \bar{G}$$

$$= -\eta \left[\left(\frac{\partial E}{\partial V}\right)\hat{i} + \left(\frac{\partial E}{\partial \omega}\right)\hat{j} \right]$$

$$E_K = \frac{1}{2}(T_K - O_{OK})^2$$

$$\bar{AB} = (V_{i+1} - V_i)\hat{i} + (\omega_{i+1} - \omega_i)\hat{j}$$

$$= \Delta V \hat{i} + \Delta\omega \hat{j}$$

$$\left[\Delta V = -\eta \left(\frac{\partial E}{\partial V}\right) \right]$$

$$\left[\Delta\omega = -\eta \left(\frac{\partial E}{\partial \omega}\right) \right]$$

$$\omega^{n+1} = \omega^n + (\Delta\omega)^n$$

$$\frac{\partial E_K}{\partial \omega_{iK}} = \left(\frac{\partial E_K}{\partial O_{OK}}\right) \times \left(\frac{\partial O_{OK}}{\partial I_{OK}}\right) \times \left(\frac{\partial I_{OK}}{\partial \omega_{iK}}\right)$$

$$\frac{\partial E_K}{\partial O_{OK}} = -(T_K - O_{OK})$$

Eqn → ② $O_{OK} = \left\{ \frac{1}{1 + e^{-\alpha(I_{OK} - \theta_{OK})}} \right\}$

$$\frac{\partial O_{OK}}{\partial I_{OK}} = \frac{-e^{-\alpha(I_{OK} - \theta_{OK})}}{\{1 + e^{-\alpha(I_{OK} - \theta_{OK})}\}^2} \quad (\because \alpha) = \frac{\alpha\, e^{-\alpha(I_{OK} - \theta_{OK})}}{\{1 + e^{-\alpha(I_{OK} - \theta_{OK})}\}^2}$$

$$= \alpha\, O_{OK} \times (1 - O_{OK})$$

Eq² → $I_{OK} = \omega_{1K} O_{H1} + \omega_{2K} O_{H2} + \cdots + \omega_{nK} O_{Hm}$

$$I_{OK} = \sum \omega_{iK} O_{Hi}$$

$$\frac{\partial I_{OK}}{\partial \omega_{iK}} = O_{Hi} \qquad 1 \le i \le m$$

m hidden layers.

$1 \le i \le m \qquad 1 \le k \le n$

$$\therefore \frac{\partial E_k}{\partial w_{ik}} = [-(T_k - O_{ok})] \cdot [\alpha \cdot O_{ok}(1 - O_{ok})] \cdot [O_{Hi}]$$

$$-\eta \left(\frac{\partial E_k}{\partial w_{ik}}\right) = -\eta \left[-(T_k - O_{ok}) \cdot \alpha \cdot O_{ok}(1 - O_{ok}) \cdot O_{Hi} \right]$$

$$\underset{m \times n}{[\Delta w]} = -\eta \left(\frac{\partial E}{\partial w}\right)$$

In vector form / matrix form,

$$\underset{m \times n}{[\Delta w]} = \eta \underset{m \times 1}{\{O\}_H} \cdot \underset{1 \times n}{\langle d \rangle}$$

(row vector)

$$\langle d \rangle = \langle \alpha ((T_k - O_{ok}) O_{ok} \cdot (1 - O_{ok})) \rangle$$

$k = 1, \ldots, n$

$$\underset{1 \times m}{[\Delta v]} = \eta \underset{1 \times 1}{\{I\}} \underset{1 \times m}{\langle d^* \rangle}$$

$$d_i^* = e_i (O_{Hi})(1 - O_{Hi})$$

where, $e_i = w_{ik} \cdot d_k$

* $\eta \rightarrow$ learning rate.

higher value of η : may diverge.

Very lower value of η : take time to converge.

Updating values,

$$w^{n+1} = w^n + \beta [\Delta w]^n$$

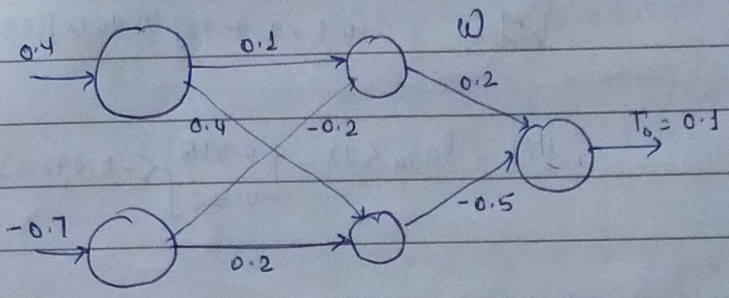
$$v^{n+1} = v^n + \alpha [\Delta v]^n$$

β : momentum factor.

$$[\Delta w]^n = \eta \{O\}_H \langle d \rangle + [\Delta w]^{n-1} \cdot \beta$$

	Inputs		Output
	I_1	I_2	O
1.	0.4	-0.7	0.1
2.	0.3	-0.5	0.05
3.	0.6	0.1	0.3
4.	0.2	0.4	0.25
5.	0.1	-0.2	0.12

Since values are within ~~range~~ [-1, +1] so no need to normalize.



Step 1: $\{O\}_I = \{I\}_I = \begin{Bmatrix} 0.4 \\ -0.7 \end{Bmatrix}$

Step 2: $[V]^0 = \begin{bmatrix} 0.1 & 0.4 \\ -0.2 & 0.2 \end{bmatrix}_{2\times 2}$; $[W]^0 = \begin{Bmatrix} 0.2 \\ -0.5 \end{Bmatrix}_{2\times 1}$

Step 3: $\{I\}_H = [V]^T \{O\}_I$ as

$$= \begin{bmatrix} 0.1 & -0.2 \\ 0.4 & 0.2 \end{bmatrix} \begin{Bmatrix} 0.4 \\ -0.7 \end{Bmatrix} = \begin{Bmatrix} 0.18 \\ 0.02 \end{Bmatrix}$$

Step 4: $\{O\}_H = \begin{Bmatrix} \frac{1}{1+e^{-0.18}} \\ \frac{1}{1+e^{-0.02}} \end{Bmatrix} = \begin{bmatrix} 0.5448 \\ 0.505 \end{bmatrix}$

Step 5: $\{I\}_O = [W]^T \{O\}_H = [0.2 \quad -0.5] \begin{bmatrix} 0.5448 \\ 0.505 \end{bmatrix}$

$\{I\}_0 = -0.14354$

Step 6: $\{O\}_0 = \left(\frac{1}{1+e^{0.14314}}\right) = 0.4642.$

Step 7: $\text{Error} = \sum (T_0 - O_0)^2 = (0.1 - 0.4642)^2 = 0.13264$

Step 8:

$$d = (T_0 - O_{01})(O_{01})(1 - O_{01})$$

$$[Y] = = (0.1 - 0.4642)(0.4642)(0.5358) = -0.09058.$$

$$[Y] = \{O\}_H \langle d \rangle = \begin{Bmatrix} 0.5448 \\ 0.505 \end{Bmatrix} \langle -0.09058 \rangle = \begin{Bmatrix} -0.0493 \\ -0.0457 \end{Bmatrix}$$

Step 9: $[\Delta w]^1 = \alpha [\Delta w]^0 + \eta [Y]$

α → momentum factor [initially $[\Delta w]^0 = 0$]

$$[\Delta w]^1 = \eta [Y] \qquad \eta = 0.6 \qquad \alpha = 1$$

$$= \begin{Bmatrix} -0.02958 \\ -0.02742 \end{Bmatrix}$$

Step 10:

$$\{e\} = [w][d] = \begin{Bmatrix} 0.2 \\ -0.5 \end{Bmatrix} (-0.09058) = \begin{Bmatrix} -0.018116 \\ 0.045291 \end{Bmatrix}$$

Step 11:

$$\{d^*\} = \begin{Bmatrix} (-0.018116)(0.5448)(1-0.5448) \\ (0.045291)(0.505)(1-0.505) \end{Bmatrix} = \begin{Bmatrix} -4.492 \times 10^{-3} \\ 0.011231 \end{Bmatrix}$$

Step 12:

$$[x] = \{o\}_I \langle d^* \rangle = \begin{Bmatrix} 0.4 \\ -0.7 \end{Bmatrix} \langle -0.00449 \quad 0.01132 \rangle$$

$$= \begin{bmatrix} -0.001796 & 0.004528 \\ 0.003143 & -0.007924 \end{bmatrix}$$

Step 13:

$$[\Delta V]^1 = \alpha [\Delta V]^0 + \eta [x] = \begin{bmatrix} -0.001077 & 0.002716 \\ 0.001885 & -0.004754 \end{bmatrix}$$

α → $\{0\}$

Step 14:

$$[V]^1 = \begin{bmatrix} 0.1 & 0.4 \\ -0.2 & 0.2 \end{bmatrix} + \begin{bmatrix} -0.001077 & 0.002716 \\ 0.001885 & -0.004754 \end{bmatrix}$$

$$= \begin{bmatrix} 0.0989 & 0.4027 \\ -0.1981 & 0.1952 \end{bmatrix}.$$

$$[w]^1 = \begin{Bmatrix} 0.2 \\ -0.5 \end{Bmatrix} + \begin{Bmatrix} -0.02958 \\ -0.02742 \end{Bmatrix} = \begin{Bmatrix} 0.17042 \\ -0.52742 \end{Bmatrix}$$

Step 15: With updated weights $[v]$ and $[w]$, error is calculated again and next training set is taken & the error will be adjusted.

Genetic Algorithm (GA)

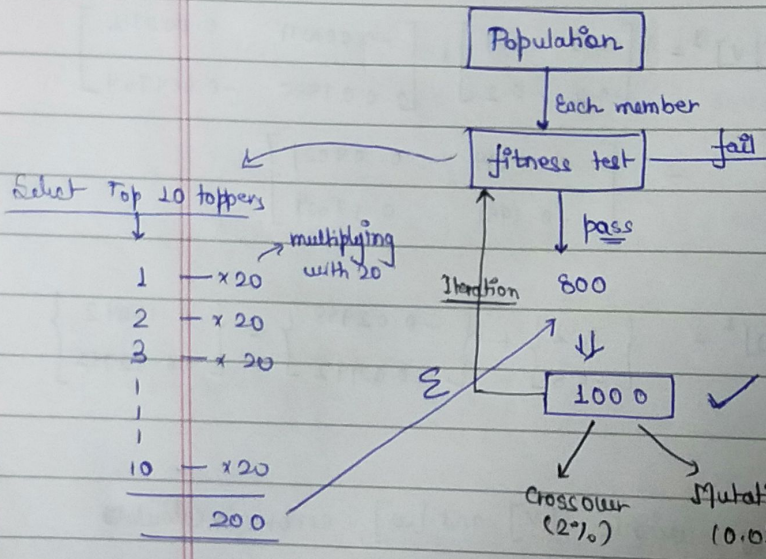
classmate
Date
Page

Darwin's theory :-

"Survival of the fittest"

New breeds / class of living things come to into existence through the processes of "reproduction", "cross over", "mutation" among existing organism.

- Gradient free
- Multiple initial Guess



Cross Over :-

Randomly Select '2' Element from population

Crossover is done when all members 2^6 are not present in population like in chose new members in population

random chose any location

make apart

6-bit

A1 A2

1) 0 1 0 1 0 1

B1 B2

2) 1 0 1 0 1 0

1 2 3 4 5

two parts

2^6 members.

A1 - 2 bit A2 - 4 bit

B1 - 2 bit B2 - 4 bit

do Crossover

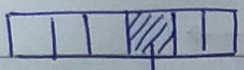
B1	A2

A1	B2

"Mutation"

1) Randomly select one string
2) Randomly select any bit position
3) flip the value at that particular 'bit position' (0 → 1 or 1 → 0)

Why important??



let, 4th bit position for all member = 0.

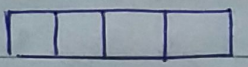
but for the optimum solution that particular bit position has to be '1',

then we will never reach optimum solution.

How to decide String length??

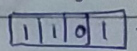
Suppose we need to decide for

(X_{min}) 0 ↔ 5 (X_{max})



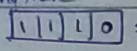
→ $2^L - 1 = 15$

let's say

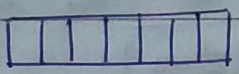


= 2

and then

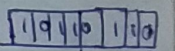


= 2.33



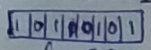
→ $2^L - 1 = 255$

If



= 2

then



= 2.02

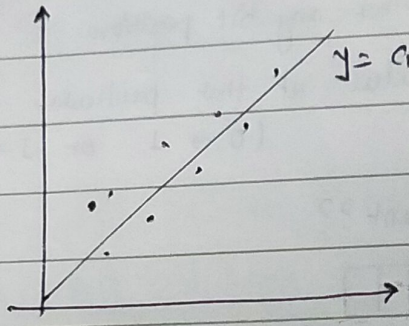
Least Count = $\frac{X_{max} - X_{min}}{2^L - 1}$

L = 4 → 0.33

L = 8 → 0.02

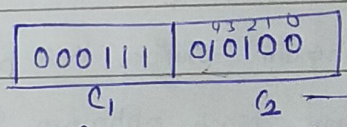
Data given :- (x_1, y_1) (x_2, y_2) (x_3, y_3) (x_4, y_4)

→ Straight line fit.



Lets take 12 bit,

(1)



$b_2 = 2^4 + 2^2 = 20$

$b_1 = 7$ (decimal equivalent)

Range :- (-2) to 5 → (1)

accuracy :- $\frac{\text{Range}}{2^b - 1} = \frac{7}{2^6 - 1}$

$$c = c_{min} + \frac{b}{(2^b - 1)} (c_{max} - c_{min})$$

b = 7, $(c_{max} - c_{min})$ = 7, c_{min} = -2, b = 6

$$= -1.22$$

$c_1 = -1.22$

$$c_2 = -2 + \frac{20 \times (7)}{2^6 - 1} = 0.22$$

(1) lets,

01001 10	00111 00

b_1	b_2
18	12

$$c_1 = -2 + \frac{18 \times 7}{2^6 - 1} = 0$$

$$c_2 = -2 + \frac{12 \times 7}{63} = -2/3 = -0.67$$

No. of strings.

y_c^i → i^{th} training data.

	b_1	b_2	c_1	c_2	y_c^i (calculated y_c^i)	$\hat{y}$ by using c_1 c_2	y_c^2	y_c^3	$\sum (b_i - y_c^i)^2$
1.	7	20	-1.22	0.22	-1	-0.78	-0.34	0.1	f^1
2.	18	12	0	-0.67	-0.67	-1.34	-2.68	-4.02	f^2
3.									f^3
⋮									⋮

Sum

In the book (it is calculated as $y = c_1 x + c_2$) ??

given data:-

x	y_a
1	1
2	2
3	3
6	6

$$f(y) = 400 - \sum (y_a - y_c)^2$$ } maximizing this.

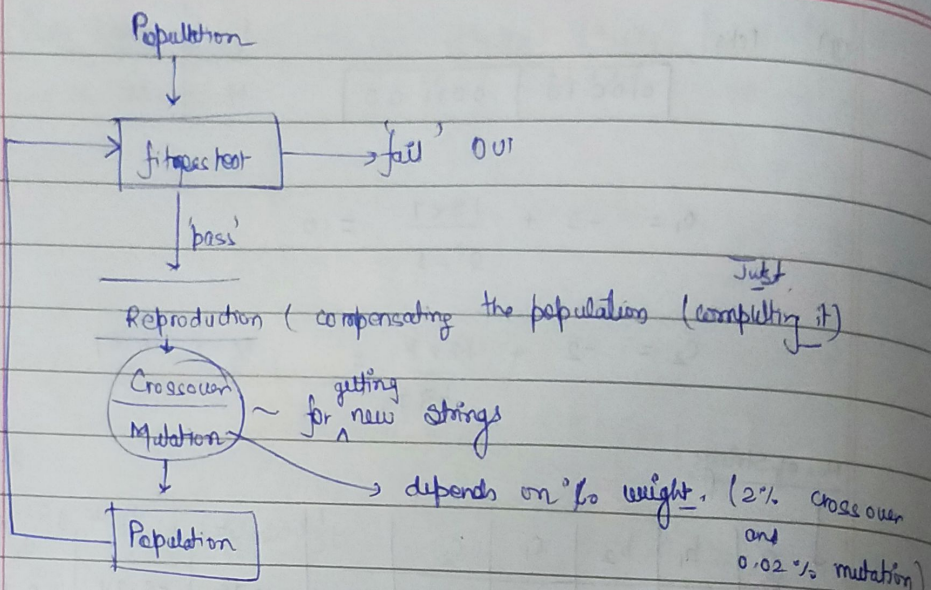
↑ Select any large number.

$f^i(y)$ – find Sum, Average, & maximum.

Expected Count = $\frac{f^i}{\text{Average}}$	Actual Count (After setting Cut-off for fitness test)

for next iteration:-

*. That string with "highest Actual count" is replicated with the lowest and then for new strings take as Crossover.



Constrained Optimization:-

Minimize $f(x)$ subject to $g_i(x) \leq 0$

$i = 1, 2, 3, \ldots, m$

Constraints, $C_i = g_i(x)$ if $g_i(x) > 0$

$= 0$ if $g_i(x) \leq 0$

$C_i \rightarrow$ violation coefficient.

$$C = \sum C_i$$

$$\phi(x) = f(x)\left[1 + KC\right]$$

$K =$ Penalty function $= 10$

① Minimize,

$$f(x) = (x_1 - 1.5)^2 + (x_2 - 4)^2$$

Subject to,

$$4.5\,x_1 + x_2^2 - 18 \leq 0$$

$$2x_1 - x_2 - 1 \geq 0$$

$0 \leq x_1$, $x_2 \geq 4$

Crossover, C.O = 20%.

Mutation, M = 3%.

Minimize, $\phi(x)$