1	2	3	4	5	6	7	8
5	5	5	5	5	5	1	5

$\frac{36}{40}$

CHENNAI MATHEMATICAL INSTITUTE
https://www.cmi.ac.in

End Semester Examination, Jan–May 2024

Name: Aritra	Roll Number: MCS202304
Date: 3rd May	Subject: DMML
Course & Year: MCS2023	Total No. of Pages: -

1. Sequence of coin tosses.

H heads, T Tails $\quad H+T=100$

$\left\{ \begin{array}{l} p_1 = \text{Probability of head given first coin} \\ q_1 = \text{" " of first coin} \end{array} \right\}$ particular

Now, $\overline{p_i}$ be the probability that the ~~head~~ particular head came from C_i
$\overline{q_i}$ " " " " " " tail " " C_i

Iteration procedure:

1) Initialize $p_1^0 = .4,\ p_2^0 = .6,\ p_3^0 = .6$ ← k^{th} round estimate of P_1, P_2, P_3

2) Done with k^{th} iteration. so we have p_1^K, p_2^K, p_3^K
at $(K+1)^{th}$ step:

$$\overline{p_i^K} = \frac{p_i^K}{\sum_i p_i^K}$$

$$q_i^K = \frac{q_i^K}{\sum_i q_i^K}$$

So, Expected Total no of heads by i^{th} coin $H_i = H \cdot \overline{p_i^K}$

" " " tails " i^{th} coin $T_i = T \cdot \overline{q_i^K}$

now, estimate $p_i^{K+1} = \frac{H_i}{H_i + T_i}$

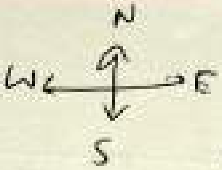
if $\max_i \left(\{ |p_i^{K+1} - p_i^K| \} \right) \leq \varepsilon$
return $(P_i = p_i^{K+1})$

else go to step 2.

The algorithm will stop if estimate of $\{P_1, P_2, P_3\}$ does not change much over an iteration, yielding almost identical values of P_1, P_2, P_3

i.e. $\hat{P}_i \cong P_i$.

5



Policy Evaluation

8. Given a uniformly random policy π.
(a)

▨ terminal states

So, $\pi(a|S) = \cdot 25$

$a \in \{S, N, E, W\} \rightarrow 4$ direction A.

South North East West

0	▨	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	▨ 15

also, at 0^{th} step, $V(S) = 0 \;\forall s \in \{0,..15\}$

$\therefore v_\pi^0(s) = 0 \;\forall s \in \{0, \dots 15\}$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Now the iteration algorithm says,

$$V_\pi^{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\,[r + \gamma\, v_\pi^k(s')]$$

as this process can go infinitely long,

let $\underline{\gamma = \cdot 5}$.

Now $v_\pi^{k+1=1}(0) = 0$

terminal state

$\Rightarrow P(a|S) = 0 \;\forall a \in A.$

$$V_\pi^1(1) = \sum_a (\cdot 25) \sum_{s',r} p(s',r|s,a)\,[r + \gamma\, v_\pi^0(s')]$$

next state and rewards

now, as the, given state and actions are deterministic, if we fix $a = E$ ($s=1$ is fixed)

$$P(s', r | 1, E) = 1 \text{ if} \begin{pmatrix} s' = 0 \\ r = -2 \end{pmatrix}$$

$\therefore P(s', r | 1, N) = 1$ if $\begin{pmatrix} s' = 1 \\ r = -1 \end{pmatrix}$ $= 0$ otherwise

$$\therefore = \cdot 25 \begin{pmatrix} [-2 + \cdot 5(0)] \\ + [-2 + \cdot 5(0)] \\ + [-2 + \cdot 5(0)] \\ + [-1 + \cdot 5(0)] \end{pmatrix} \begin{cases} \text{for Left} \\ \text{" right} \\ \text{" down} \\ \text{" up} \end{cases}$$

$= -1.75$

Similarly :

terminal 0	$-1 \uparrow .25$, $.25 \leftarrow -1$, $\rightarrow .25$ -2, $-2 \downarrow .25$ $= -1.75$	$-2 \leftarrow$ $\uparrow -1$ $\rightarrow -2$ $\downarrow -2$ $= -1.75$	$-1 \uparrow$, $-2 \leftarrow$, $\rightarrow -1$, $\downarrow -2$ $= -1.5$
$-2 \uparrow$, $-1 \leftarrow$, $\rightarrow -2$, $-2 \downarrow$ $= -1.75$	$-2 \leftarrow$ $\uparrow -2$ $\rightarrow -2$ $\downarrow -2$ $= -2$	$= -2$	$= -1.75$
-2.75	-2	-2	-1.75
-1.5	-1.75	-1.75	terminal 0

$V^1_\pi(s)$ (not all calculations are shown)

one round iteration will yield V_π as v^1_π.

(b) Policy improvement :

We can do the previous step until v^k_π converges.

i.e. $\max_s (v^{k+1}_\pi(s) - v^\pi_k(s)) < \theta$ ← given

and $v_\pi = v^k_\pi$

Once, we got the v_π we can do policy improvement.

by making policy deterministic. $\pi^*(s) = a \in A$.

(Question says, use v^1_π as $\hat{v}_\pi$)

(improving to)

Now, we are having a deterministic policy, so,

we will determine $\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} \sum_{s'} \sum_r P(s', r | s, a) [r + \gamma V_\pi(s')]$

N

W — E

S

To restate this, we will choose that direction only to which

Σ (reward I will get + $V_{\pi}(S)$) will be maximum

all 4 directions but Expected reward maximizes

we can go to if N is chosen (0, E)

now,

Terminal

So we will choose Nothing

$\pi^*(S)$

$V_{\pi}(S) = 0$ $0+(-1)$ ↑ $0+(-1)$ ← $(-1.75) + -2$ $(-1.75 + -2)$ as terminal = N / {}	-1.75 $-1.75-1$ $0-1$ $-1.75-1$ $-2-1$ So, W	-1.75 -2.75 -3.75 -3.5 N	-1.5 N&E
-1.75 N	-2 N	-2 N	-1.75 E
-1.75 W	-2 S	-2 S	-1.75 S
-1.5 S&W	-1.75 S	-1.75 E	0 S / {}

South or Nothing as terminal

So, $\pi(S)$ is non-deterministic, and the policy is better as $V_{\pi^*}(S) > V_{\pi}(S)$ $\forall S$. by construction of π^*.

5

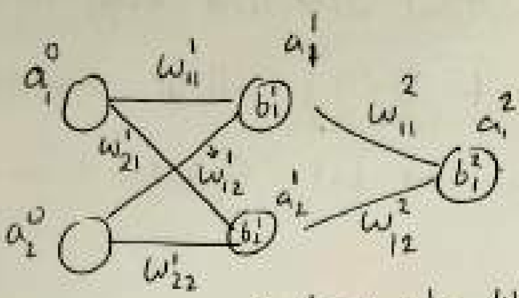
This way, we can improve policy by

policy ⟶ evaluate ⟶ improve ⟶ evaluate ⟶ improve

until improvement converges (i.e. $\pi^*_k(S) = \pi^*_{k+1}(S)$)

6. lets look at such a neural network.

Neural Network A (NNA)



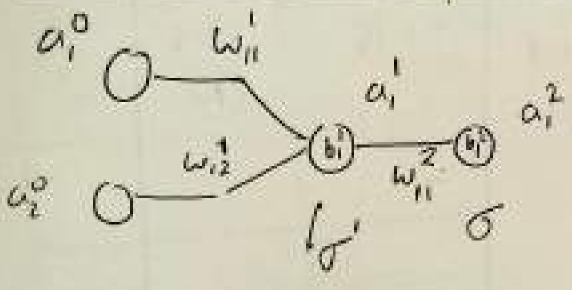
with activation function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$w_{11}^1 = w_{21}^1 = w_1^1$ $\quad w_{12}^2 = w_{11}^2 = w_1^2$ given

$w_{21}^1 = w_{22}^1 = w_2^1$ $\quad b_1^1 = b_1^2$

Neural Network B (NNB)



$$\sigma'(z) = \frac{2}{1+e^{-z}}$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

We will show that,

(NNA and NNB will work mostly same though the NNA has two nodes in hidden layer whereas NNB has one.)

for NNB

$$a_1^1 = \sigma'(z_1^1)$$

$$z_1^1 = (w_{11}^1 a_1^0 + w_{12}^1 a_2^0) + b_1$$

$$\therefore a_1^1 = \sigma'(z_1^1)$$

and $z_1^2 = w_{11}^2 a_1^1$

$$a_1^2 = \sigma(z_2^1)$$

for NNA

$$a_1^1 = \sigma(z_1^1)$$

$$z_1^1 = (w_{i1}^1 a_1^0 + w_{i2} a_2^0) + b_i^1$$

so, $z_1^1 = z_2^1$

$a_1^1 = a_2^1$

$$\therefore a_i^1 = \sigma(z_i^1)$$

$$z_1^2 = w_{11} a_1^1 + w_{12} a_2^1 + b_1^2$$

$$a_1^2 = \sigma(z_1^2)$$

$\therefore$ both of the neural networks

have same cost function same learning rate and same changes in
as the backpropagation happens, so that after one iteration they output the same value on same training set

So, by this;

Corollary: final weight and bias of two nodes of NNA have exact same value and that will be proportional to the weight and bias of NNB.

Corollary: If we have such nodes we can always remove one and modify the activation of the other one to get the exact same learning and output.

As having more nodes generally means more computation, it's generally better to keep it short and avoid multiple identical nodes in same layer (as this will speedup the learning process).

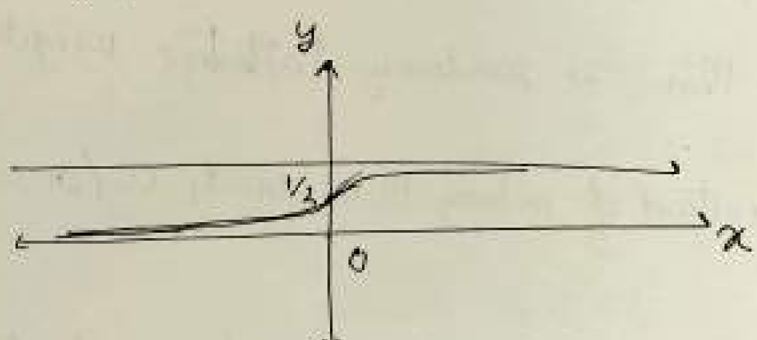
So, to avoid this, we randomly initialize weights and biases of such networks instead of making them exactly 0. (or some identical values).

This will guarantee that the training of Neural network is faster.

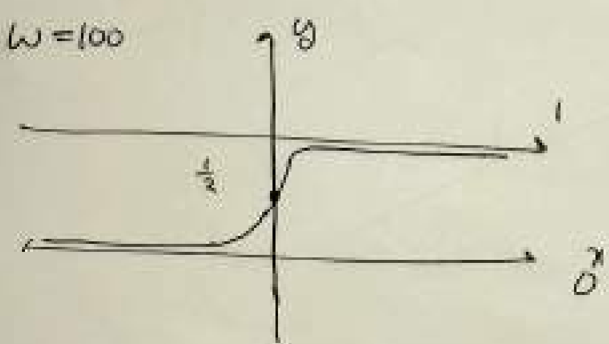
(5) (a) $z = wx+b$ max value of $z = 1$ as $e^{-z} > 0 \; \forall z$

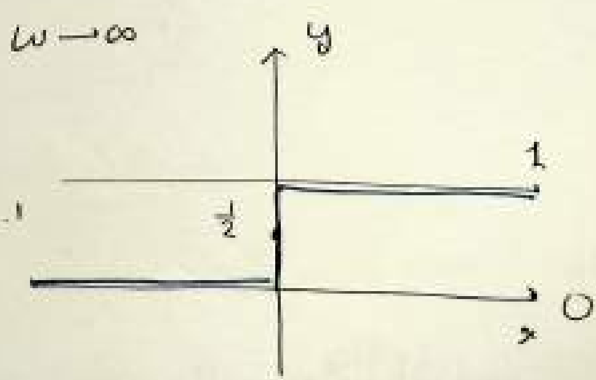
$\sigma(z) = \frac{1}{1+e^{-z}}$ min value of $z = 0$ as $e^{-z} \to \infty$ $z \to -\infty$

~~Varying~~ General graph for $w=1$ and $b=0$ $\sigma(z)=\sigma(x) = \frac{1}{2}$ if $z=0$



If we vary w keeping $b=0$.





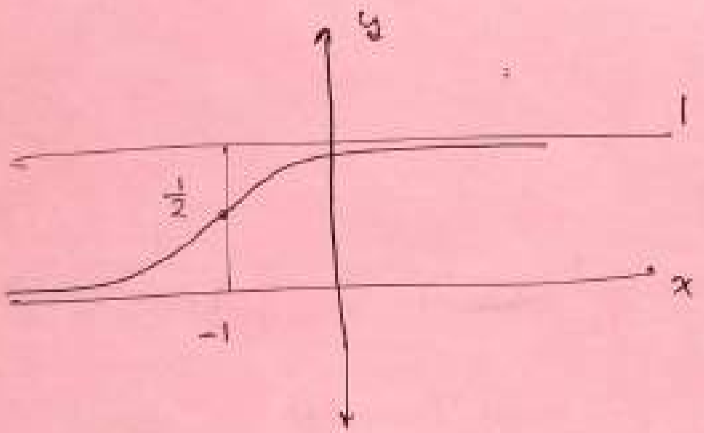
If we vary b keeping w constant

$$\sigma(z) = \frac{1}{1+e^{-(wx+b)}}$$

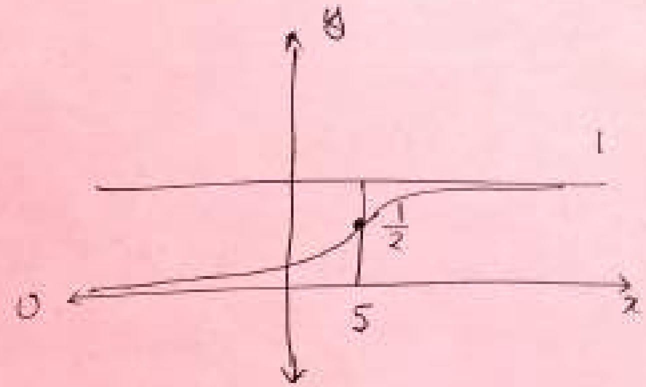
$$= \frac{1}{1+e^{-w(x+\frac{b}{w})}}$$

So keeping $w=1$

$b = w = 1$

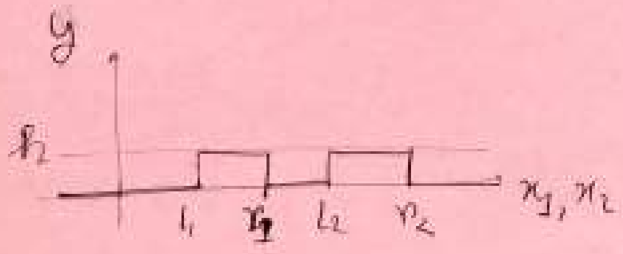


$b = -5w = -5$



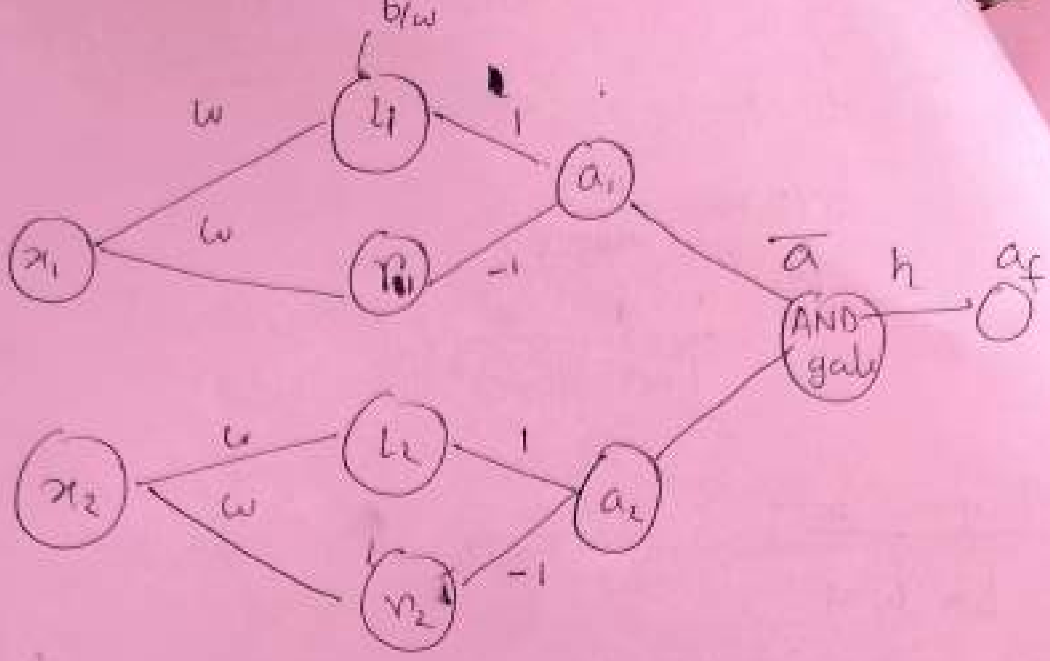
So, w determines the sharpness } of sigmoid.
Where b determines the center }

(b) So our rectangular box graph will be,



2D input

if both$(l_i \leq x_i \leq r_i)$ then h
input
else 0



$\underline{w = \text{very high}}$

now if $x_i < l_i, r_i$

then $a_i = \frac{1}{1+e^{-w(x_i - l_i)}} - \frac{1}{1+e^{-w(x_i - r_i)}}$

$= 0$ as ($w = \text{high} \to \infty$, $1 + e^{-w(x_i - l_i)}$)

$\frac{1}{1+e^{-w(x_i - l_i)}} = 0.$

if $a_i = 1$ if $l_i \le x_i \le r_i$

$a_i = 0$ if $x_i \geq r_i, l_i$

So, $a_i = 1$ iff $l_i \le x_i \le r_i$

now $\overline{a} = a_1 \text{ (and) } a_2$

if both of them are 1

then only,

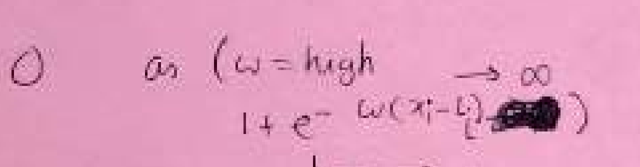
$a_f = \overline{a} \cdot h$

$= h$

else, 0.

5

and gate design using neural network



we can do $\begin{pmatrix} w_1 = 1, w_2 = 1 \\ b_1 = -1 \end{pmatrix}$

iff both $a_i = 1$

$\left\{ \begin{array}{l} \text{then only } \overline{a} = 1+1-1 = 1 \\ \text{else } \overline{a} = \max(0, \overline{a}) = 0 \end{array} \right\}$

(4) <u>how a model with 0-1 loss function learn?</u>

for a training data, currently partially learnt classifier run that training data and if misclassify cost function is 1.

and otherwise " " " 0.

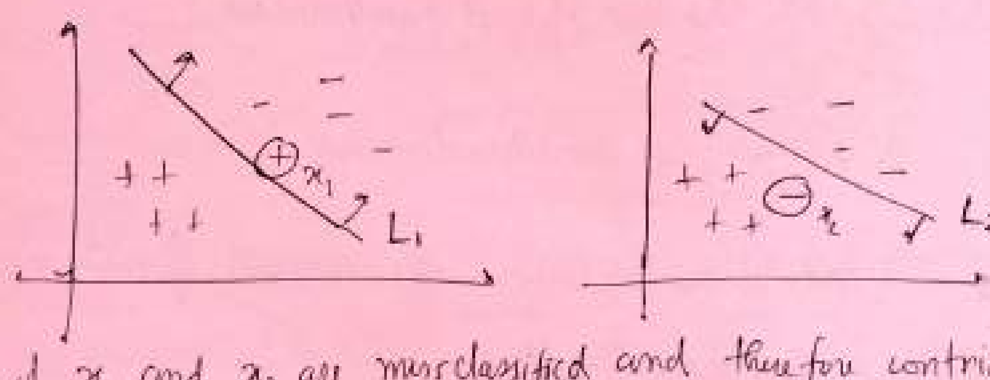
In that case, we can run gradient descent iteratively to tweak the parameters [~~... or coefficients or slope and bias~~]

In case of perception algorithm,

if we have data in n dimensional space, we need to find a (n-1) dimensional hyperplane to effectively separate the data using labels

In 2D

<u>Now,</u>



both of x_1 and x_2 are misclassified and therefore contribute +1 towards the cost function.

Also, from the picture, it's clear that L_1 should be dragged towards the center (origin) whereas L_2 should be pushed away

So, both the datapoints have same contribution towards loss function but the action that should be taken are completely opposite.

That's why 0-1 cost function is not a good model.

This is the reason, we use count ($(w_i x_i + b_i) y_i < 0$) as the missclassification in each iteration

> 0 if correctly classified

< 0 if misclassified

also we know value of the $y_i \in \{-1, 1\}$.

Depending on value of y_i we can tweak the line in required direction

(our perceptron uses)

The advantages of this type of cost functions are

1) Not only it considers direction

2) but also, determines the amount of nudge needs to be made on the separating line

OK (5)

(3) Clustering can be used for image segmentation depending on the number of objects we need to identify from the image.

In general, the most important (visible) object (may) can be found if we make the number of clusters 2.

to get K most important (visible) objects, we may use (K+1) clusters

Explanation :-

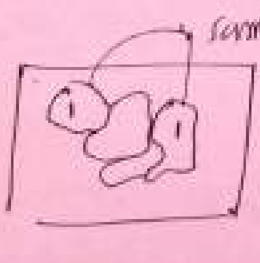
1> First we have an RGB image on which we want to apply clustering

2> If we make the number of clusters as 10-12, in general the image lose minor information. (As most of the colors can be slightly modified to obey a cluster representation point.

this small clusters will be merged with one of the adjacent clusters depending on the similarity of colours

3> Moreover, here the distance in terms of pixels are the color difference (not the euclidean distance). So clusters can be disjoint when viewed from euclidean distance perspective.

same cluster → same ~~color~~ color ~~image~~ content.



④ The most important (visible) object is assumed to have same color distribution throughout.

(we want to extract the information about what percentage of globe has water) ⟶ we will assume that globe has water color as blue throughout

⑤ If we decrease the number cluster some objects will lose its color but still can be identified as structure

⑥ finally if we make the number of cluster as 2.

there will be only two colors.

1) color of the most important object (from which the object can be identified)

2) other major color, aggregation of all the objects

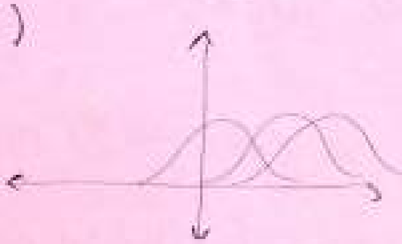
Thus way we can find the most important (visible) object from the image using clustering

5

Clustering points using a mixture of gaussian can be done following the same method as expectation maximization.

(Random Variable $X_1 \cdots X_K$)

for simplicity, we are assuming that, all the gaussians have same variance σ^2 and different mean. $(\mu_1 \ldots \mu_K)$



now, given a point x_i,

the probability of $P(X_1 = x_i)$

$$\propto e^{-\frac{(x_i - \mu_1)}{2\sigma^2}}$$ (I forgot the distribution)

$$\therefore P(X_j = x_i)$$

$$\propto = e^{-\frac{(x_i - \mu_j)}{2\sigma^2}}$$

If we get the probability,

we can calculate the likelihood of that point x_i belongs to gaussian X_j can be computed as follows

$$L(x_i \in X_j) = \frac{P(X_j = x_i)}{\sum_{j'} P(X_{j'} = x_i)}$$

(we can also use softmax) if we want to smooth it a bit $\frac{e^P}{\sum e^P}$

Now, instead of spreading the point over gaussian (as we did in case of expectation maximization) we say that, $x_i \in \underset{j}{\operatorname{argmax}}\, L(x_i \in X_j)$

that is x_i belongs to j^{th} gaussian

Following way, we can do clustering for n points by assigning each point to the most probable gaussian

PTO

This algorithm can also be done to classify outliers.

The intuitive idea will be that, if a point lies far away from all the gaussian, it will have almost same probability for appearing in the every gaussian

So $L(x_i \in X_j)$ will be small and almost same for all the X_j gaussian

Previously, if a point has very high likelihood, appearing in a particular gaussian say X_j

all the other $L(x_i \in X_k)$ will be low.

So, based on the $L(x_i \in X_j)$ distribution (if all the values are small and almost same) we classify a point as an outlier.

5

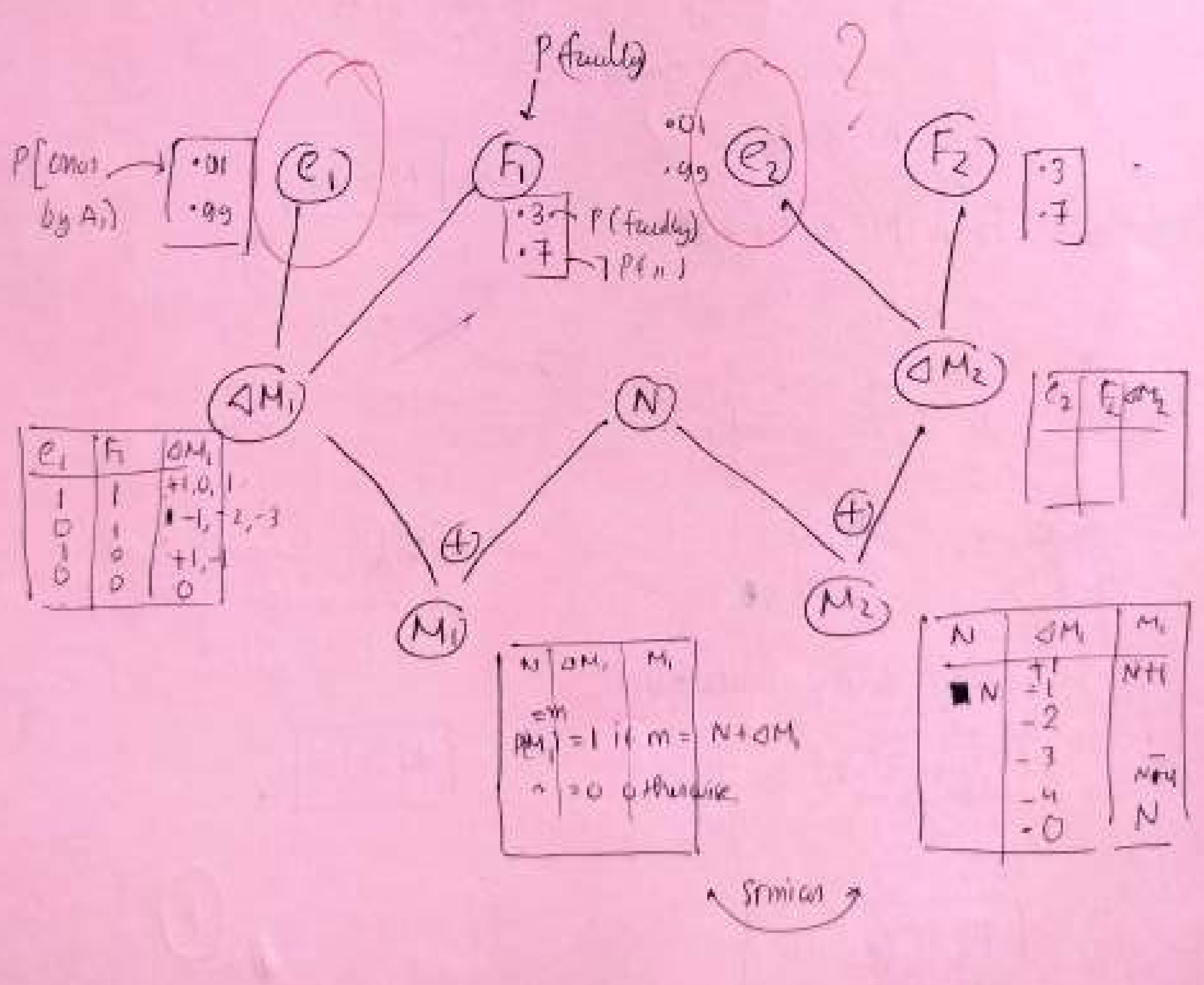
7. A_1 - astronomer 1
A_2 — " 2
$F_1 \rightarrow T_1$ being faulty
$F_2 \rightarrow T_2$ being faulty
$M_1 \rightarrow$ Number of stars by A_1
$M_2 \rightarrow$ " " " by A_2

N — Number of stars

addition of extra errors

e_1 — miscount by A_1
e_2 — miscount by A_2

ΔM_1 — numb of stars that can be miscounted by M_1
ΔM_2 — " " stars " " " by M_2



(b) $M_1 = 12$

$M_2 = 14$

possible values of N can be calculated as follows:

if F_1 is faulty and Astronomer miscounted

then $N \leq M_1 + \underset{\text{for faulty}}{3} + 1$ ← for error by A_1

$N \leq 16$ so, $\boxed{N \leq 16}$

Similarly for F_2, e_2; $N \leq 14 + 4$

≤ 18

Moreover, if the case was A_1 overcounted, then

$N \geq 12 - 1$

≥ 11

Similarly if A_2 overcounted,

$N \geq 14 - 1$ $\boxed{N \geq 13}$.

≥ 13

$\therefore$ $\boxed{13 \leq N \leq 16}$

Not clear. Need 4 estimates for 4 combination of F_1, F_2

1