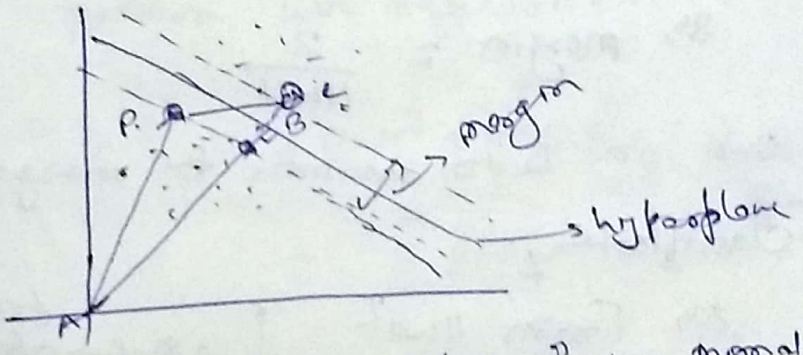
SOC - C3 10:30 AM

MIT2019090, VISHAL KUMAR

02.) (a) Ans Vladimir vapnik is credited for developing support vector machine (SVM). His main contribution was to use statistical approach for classifying rather than probabilistic approach, he has been awarded by Benjamin Franklin medal.



When we project vector $\vec{w}$ i.e normal to hyperplane, we get the line on which side of the hyperplane

$$\vec{w}.\vec{x} \geq c \quad \text{or} \quad c = -b$$

Decision Rule
$$\begin{cases} \vec{w}.\vec{x} + b \geq 0 & \text{then +ve sample} \\ \vec{w}.\vec{x} + b \leq 0 & \text{then -ve "} \end{cases}$$

or $\vec{w}.\vec{x} + b \geq 1 \quad -(1) \quad y = +1$

$\vec{w}.\vec{x} + b \leq -1 \quad (1) \quad y = -1$

combining these two,

$$y^{(i)}(\vec{w}.\vec{x} + \vec{b}) - 1 \geq 0 \quad -(3)$$

AB when projection on unit vector

$$(\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|w\|}$$

From (3), we get

$$\vec{x}_+ \cdot w + b - 1 = 0 \Rightarrow \vec{x}_+ = \frac{1-b}{w}$$

for negative vector

$$\vec{x}_- = \frac{b-1}{w}$$

So, margin $= \frac{2}{\|w\|}$

Our aim is to maximize the margin for better classification.

So $\left[\max \frac{\|w\|^2}{2} \right]$ → for mathematically convenient

↓

subject to $y^{(i)} [w x^{(i)} + b) - 1 \geq 0$

Lagrangian

$$L = \frac{1}{2}\|w\|^2 - \sum \lambda_i \left[y^{(i)} (w x^{(i)} + b) - 1 \right]$$

↓

Lagrange's multiplier

$$\frac{\partial L}{\partial w} = 0 \quad \& \quad \frac{\partial L}{\partial b} = 0$$

So, $\frac{\partial L}{\partial w} = 0 \Rightarrow \vec{w} - \sum \lambda_i y_i x_i = 0$

$$\Rightarrow \vec{w} = \sum \lambda_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow -\sum \alpha_i y_i = 0$$

$$\Rightarrow \boxed{\sum \alpha_i y_i = 0}$$

For dual problem we will put these 2 values in our Lagrange.

$$\boxed{L(w, b, \alpha) = \sum \alpha_i - \frac{1}{2} \sum_m \sum_n \alpha_i \alpha_j x_i x_j y_i y_j}$$

Dual problem with regularization term.

$$L(w, b, \alpha, \xi, \beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i(w_i x_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{m} \beta_i \xi_i$$

$$\text{S.t.} \quad \sum_{i=1}^{m} y_i \alpha_i = 0$$

$$\alpha_i \geq 0, \quad i = 1, 2, 3 \ldots n$$

Q2) (b) Ans →

Dual problem was important because in primal we took α i.e. lagrangian multiplier on support vectors so,

i) complexity increased with increase in no. of support vectors

ii) support vectors are to be mentioned before hand.

So, primal equation failed. Thats why we need dual problem.

$$\max L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle \quad -(1)$$

Subject to $\alpha_i \geq 0, i = 1, 2, \ldots n$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$\min . f(w)$

S.t $h_i(w) = 0 \; ; i = 1, 2, 3 \ldots$

$g_i(w) \leq 0 \; : i = 1, 2, 3 \ldots$

$$L(w, \alpha, \beta) = f(w) - \left(\sum_{i=1}^{n} \alpha_i g(w) + \sum \beta_i h_i(w) \right) \quad -(2)$$

$$L_p(w) = \max_{\alpha, \beta \geq 0} L(w, \alpha, \beta) \quad -(3)$$

$$P = \min_{w} \max_{\alpha, \beta \geq 0} L(w, \alpha, \beta)$$

$$= \min_{w} L_p(w) \quad -(4)$$

$L_p(w)$ if $g(w) > 0$ i.e violating the constraints

If violated,

$L_p(w) \to \infty$

If $h_i(w) = 0$ (violating constraint)

$L_p(w) \to \infty$

So $L_p(w) = \begin{cases} f(w) & \text{, if constraint satisfied} \\ \infty & \text{otherwise} \end{cases}$

$$D^* = \max_{\alpha,\beta \geq 0} \min_{w} L(w, \alpha, \beta)$$

$$= \max_{\alpha,\beta \geq 0} \theta_D(\alpha, \beta)$$

$D^* = P^*$ for max margin,

Solution of dual problem,

$$D^* = P^* = L(w^*, \alpha^*, \beta^*)$$

to max one, w^*, α^*, β^*, we must satisfy,

$$\frac{\partial}{\partial w} L(w, \alpha^*, \beta) = 0$$

$$\frac{\partial}{\partial \beta_i} L(w^*, \alpha^*, \beta^*) = 0$$

$$\alpha^* g(w^*) = 0$$

$$g_i(w) \leq 0, \alpha \geq 0$$

Basic coordinate ascent algorithm,

$\max W(\alpha_1, \alpha_2, \alpha_3, \ldots \alpha_m)$ (without any constraint)

Repeat { for $i = 1$ to m

$$\alpha_i = W(\alpha_1, \alpha_2, \ldots \alpha_m)$$

Sequential minimal Optimization in SMO, α_i are taken at a time,

$$\alpha_1 y - \sum_{i=1}^{m} \alpha_i y_i = 0$$

Select

→ Select α_i, α_j heuristically

→ Hold all α_i fixed except α_1, α_i

→ optimize W(α) wrt α_1, α_j (s.t. KKT)

Let's put $i=1$ & $j=2$

randomly

update α_1 & α_2 s.t. $\sum_{i=2}^{m} \alpha_i y_i = 0$

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = -\sum_{i=1}^{m} \alpha_i y^{(i)} - \xi_i$$

$$0 \le \alpha_i \le C$$

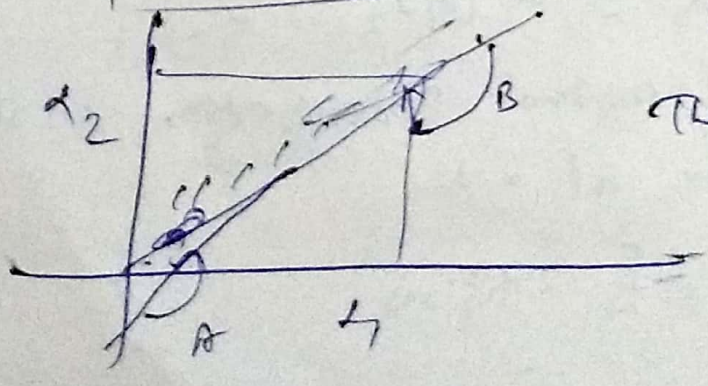
Repeat until convergence

1) a select some pair of α_i α_j to update next respect to α.

2) Re optimize W(α) wrt α_i, α_j holding all α_k $(k \neq i, j)$

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = -\sum \alpha_i y_i = \xi_i$$

$$\boxed{\alpha_i = \frac{\xi_j - \alpha_2 y^{(i)}}{y}}$$



$W(\alpha_2) = A\alpha_2^2 + B\alpha_2 + C$

$L \le \alpha_2 \le H$

Then we can maximize quadratic eqn.

$$\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2 > H \\ \alpha_2 & \text{if } L \le \alpha_2 \le H \\ L & \text{if } \alpha_2 < L \end{cases}$$

05) (a.) Ans →

A generative model learns the joint probability distribution $P(x, y)$. It predicts the conditional probability with the help of Bayes theorem.

A Discriminative model learns the conditional probability distribution $P(y/x)$. Both of these models are generally used in supervised learning problem.

Gaussian Discriminant Analysis (GDA) algorithm is a generative learning algorithm. As, we know, generative learning algorithm make strong assumption and can't predict the conditional probability directly. GDA assumes that $P(X/y)$ is distributed according to a multivariate normal distribution

A multivariate gaussian distribution is fit for each class. This allows us to find $P(y)$ and $P(X/y)$. Using this two we can finally find $P(y/x)$, which is required for prediction.

GDA	Naive Bayes
i.) Strong modeling Assumptions	i.) Even stronger modeling assumptions.
ii) $d(d+1)/2 + 2d + 1$ Parameters	ii) $3d + 1$ Parameters
iii) A simple closed form solution	iii) Super simple closed form solution.

GDA makes an assumption about the probability distribution of the $P(X/y=k)$ where k is one of the classes. And it can be easily proved that for GDA, if the initial assumptions about distribution of $P(X/y=k)$ (Gaussian) and $P(y)$ (Bernoulli) are true, the $P(y/x)$ can be expressed as sigmoid but vice-versa is not true.

04) Ans → Genetic Algorithm intended by Prof John Holland.

Problem → $f(x) = x - x^2$, $x \in [0,15]$; $x \in Z$

We will take x in bits ie $(0000)_2$ to $(1111)_2$

Initial chromosome population =6

~~Iteration 1~~:

Initial Population	x	f(x)			
1010	10	-90	0.36	1.45	1
1100	12	-132	0.53	2.13	2
0011	3	-6	0.02	0.09	0
0101					

Iteration 1:

Initial Population	x	f(x)	$\frac{f_i}{\Sigma f_i}$	$\frac{f_i}{\bar{f}}$	Roulette wheel
1010	10	-90	0.36	2.18	2
1100	12	-132	0.53	3.20	3
0011	3	-6	0.02	0.14	0
0101	5	-20	0.08	0.48	0
0000	0	0	0	0	0
0001	1	0	0	0	0

$\Sigma = -248$

$\bar{f} = -41.3$

mating :-

1100 } 1110
1010 } 1000

11 00 } 1111
00 11 } 0000

1100 } 1101
0101 } 0100

Iteration 2:

Population	x	f(x)	$\frac{f_i}{\Sigma f_i}$	$\frac{f_i}{\bar{f}}$	Roulette wheel
1110	14	182	0.3	1.77	2
1000	8	56	0.09	0.54	1
1101	13	156	0.25	1.52	2
0100	4	12	0.019	0.12	0
1111	15	210	0.34	2.04	2
0000	0	0	0	0	0

$\Sigma = 616$

$\bar{f} = 102.7$

Schema theory :

H = | 1 | * | * | 1 | * |

Instances of H are :→ 10010, 10011, 10110, 10111, 11010, 11011, 11110, 11111

Order of H = 2 (No. of fixed bits)

Defining length of following schemata :-

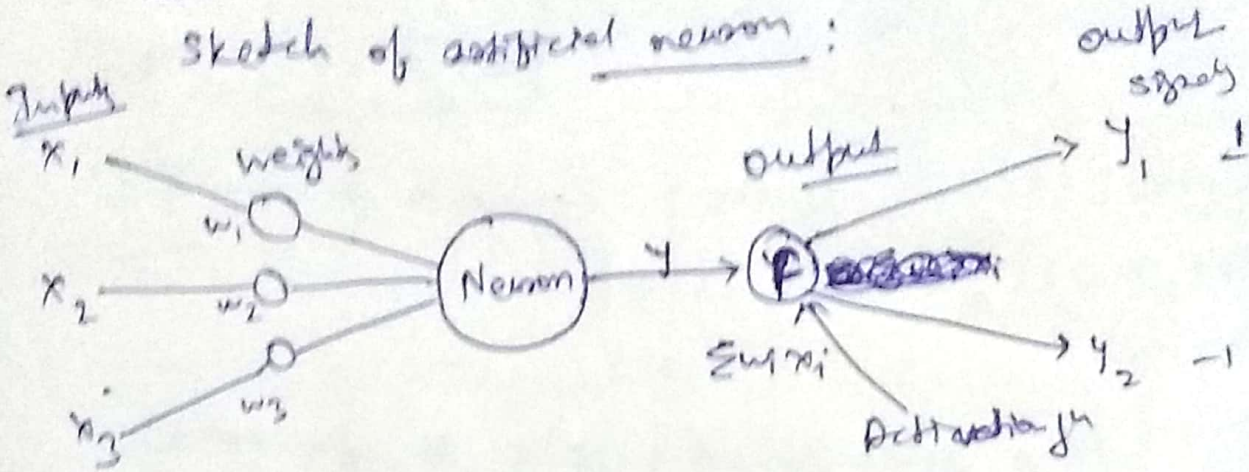
H_1 : * * 1 0 0 ⇒ 5-3 = 2

H_2 : * 0 * 1 * 0 * ⇒ 6-2 = 4

H_3 : 1 * * * * * * 0 ⇒ 8-1 = 7

03) Ans →

Sketch of artificial neuron :



$$y = \begin{cases} +1 & , \text{if } \Sigma x_i w_i \geq \theta \\ -1 & , \text{if } \Sigma x_i w_i < \theta \end{cases}$$

Activation function used for generating signals can be any sign function.

The middle layer hides the desired output that's why it is also called hidden layer. The neurons present in the hidden layer can be observed through the change in the behaviour of the network's input / output. The hidden layer present in between the input and output layer and from outside it is present as abstract unit.

The gradient of the error function with respect to the layer present is called as error gradient.

(a) Error gradient terms:

i) First term is the difference between output and target value.

ii) Second term is the derivative of output layer activation function

iii) Third term is the activation output of node j in the hidden layer.

For any weight w'_{11}, it can be written as:

$$= \frac{\partial E}{\partial w'_{11}}$$

$$\delta_k(p) = y_k(p)\,[1 - y_k(p)]\,e_k(p)$$

where, $e_k(p) = y_{d,k}(p) - y_k(p)$

The error gradient is required to calculate in order to find the effect of node of change of error on the weights present in the neural network.

Any particular weight can be calculated as

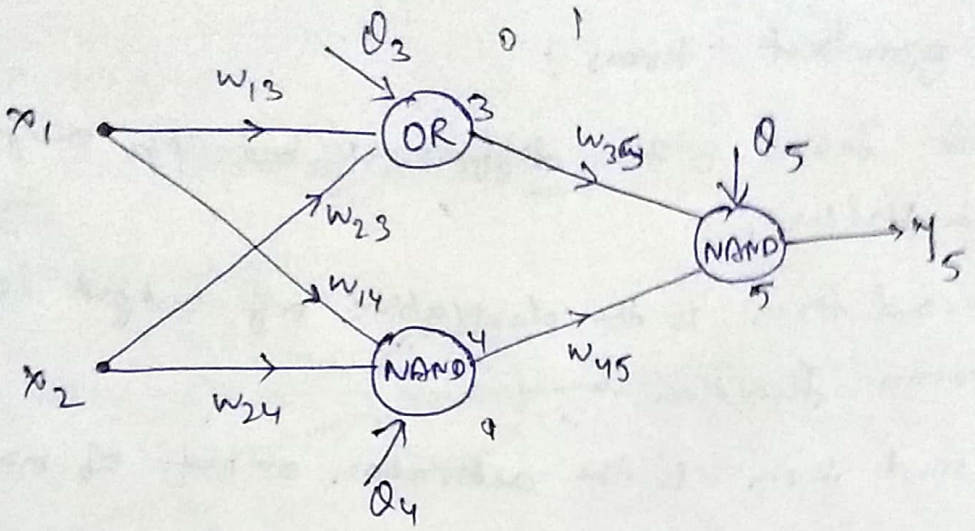
$$(w'_{11})_+ = w'_{11} - \eta \cdot \frac{\partial E}{\partial w'_{11}}$$

Working of XOR:

Neural network is a three layer network, which consists of three neurons as,

1) 2 NAND neurons

2) 1 OR neuron



x_1	x_2	y_5
0	0	1
0	1	0
1	0	0
1	1	1

Let's bias terms θ has fixed initial value = 1

Works

i) Initialization of the weights:

$w_{13} = 0.5$, $w_{14} = 0.9$, $w_{23} = 0.4$, $w_{24} = 1.0$,

$w_{35} = -1.2$, $w_{45} = 1.1$, $\theta_3 = 0.8$, $\theta_4 = -0.1$, $\theta_5 = 0.3$

ii) Activation function:

Pass x_i forward through the network.

Activation f^n ⇒ sigmoid.

$$y_3 = \text{sigmoid}(x_1 w_{13} + x_2 w_{23} - \theta_3)$$

$$= \frac{1}{1 + e^{-(0.5 \times 1 + 1 \times 0.4 - 0.8)}} = 0.5250$$

Here x_1 and $x_2 = 1$

$y_4 = \text{sigmoid}(x_1 w_{14} + x_2 w_{24} - \theta_4)$

$= 0.8808$

Now output at y_5:

$= \text{sigmoid}(y_3 w_{35} + y_4 w_{45} - \theta_5)$

$= 0.5097$

Now, compute Error →

$e = y_{d,5} - y_5 = 1 - 0.5097 = 0.4903$

→ Compute all derivatives)

$\delta_5 = y_5(1-y_5)e = 0.5097(1-0.5097)(0.4903)$

$= 0.1225$

$\Delta w_{35} = \alpha \cdot y_3 \cdot \delta_5 \quad (\alpha = 0.1)$

$= 0.1 \times 0.5250 \times 0.1225 = 0.0064$

$\Delta w_{45} = \alpha\, y_4 \cdot \delta_5$

$= 0.1 \times 0.8808 \times 0.1225 = 0.010$

error gradient for neuron 3 & 4,

$\delta_3 = y_3(1-y_3)\delta_5 \cdot w_{25} = 0.8346$

$\delta_4 = y_4(1-y_4)\delta_5(w_{45}) = 0.0141$

The change in weights

$\Delta w_{13} = \alpha x_1 \delta_3 = 0.1 \times 1 \times 0.0344 = 0.00344$

$\Delta w_{23} = \alpha x_2 \delta_3 = 0.1 \times 1 \times 0.0344 = 0.00344$

$\Delta \theta_3 = \alpha (-1) \delta_3 = -0.00344$

$\Delta w_{14} = \alpha x_1 \delta_4 = 0.00141$

$\Delta w_{24} = \alpha x_2 \delta_4 = 0.00141$

$\Delta \theta_4 = \alpha (-1) \delta_4 = -0.00141$

d → weight updations:—

$w_{13} = w_{13} + \Delta w_{13} = 0.5034$

$w_{14} = w_{14} + \Delta w_{14} = 0.9014$

$w_{23} = w_{23} + \Delta w_{23} = 0.4014$

$w_{24} = w_{24} + \Delta w_{24} = 1.0014$

$w_{35} = w_{35} + \Delta w_{35} = 1.1926$

$w_{45} = w_{45} + \Delta w_{45} = 1.110$

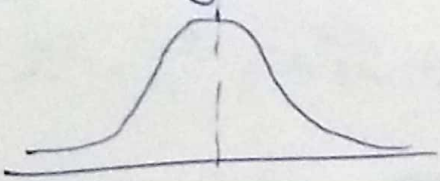
$\theta_3 = \theta_3 + \Delta \theta_3 = 0.3964$

$\theta_4 = \theta_4 + \Delta \theta_4 = -0.1041$

$\theta_5 = \theta_5 + \Delta \theta_5 = 0.2834$

Q5.) (b.) Ans

In the experiment we were told to classify the data in two classes using LDA with raw data itself and one by changing the raw data into normal distribution using Box Muller Transformation. As we can see that, after applying box-muller transformation the accuracy significantly increase. After applying the box-muller transformation, the data is distributed in gaussian bell shape as follows;



Data belongs to same class may be found near to mean or away from mean. This result in some form of clusters. The both of data point near to mean is high as compared to with points away from mean results in better probability model to classify the data.

05.) (c) Ans

Let's assume we have d-dim sample

$\{x^1, x^2, \ldots x^N\}$ N_1 which belongs to w_1 and N_2 belong to w_2.

$y = w^T x$ where $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ and $w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$

So the mean of each class in x & y feature space is.

$$\mu_i = \frac{1}{N} \sum_{x \in w_i} x$$

$$\mu_i^n = \frac{1}{N} \sum_{y \in w_i} y = \frac{1}{N_i} \sum_{x \in w_i} w^T x.$$

$$= \frac{w^T}{N} \sum_{x \in w_i} x = w^T \mu_i$$

So we can choose distance between the projected means as our objective function or cost.

$$J(w) = |\mu_1^n - \mu_2^n| = |w^T \mu_1 - w^T \mu_2|$$

$$= |w^T (\mu_1 - \mu_2)|$$

This is Fisher's solution to maximize a func.

$$S_i^n = \sum_{y \in w_i} (y - \mu_i^n)^2$$

variance within M class

By: $J(w) = \frac{|\mu_1^n - \mu_2^n|^2}{s_1^{n2} + s_2^{n2}}$

→ we have two class

Now Projection of y as a d^n of scatter in feature space x:

$$\hat{s}_1^2 = \sum_{y \in w_1} (y - \hat{\mu}_1)^2 = \sum_{x \in w_1} (w^T x - w^T \mu_1)^2$$

$$= \sum_{x \in w_1} w^T (x - \mu)(x - \mu_1)^T w$$

$$= w^T s_1 w$$

Now,

$$\hat{s}_1^2 + \hat{s}_2^2 = w^T s_1 w + w^T s_2 w = w^T (s_1 + s_2) w$$

Now diffbet between pri mean.

$$(\hat{\mu}_1 - \hat{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2$$

$$= w^T (\mu_1 - \mu_2)(\mu_1 - \mu_2) w$$

$$= w^T s_B w = \hat{s}_B$$

Hence, finally,

$$J(w) = \frac{(\hat{\mu}_1 - \hat{\mu}_2)^2}{\hat{s}_1^2 + \hat{s}_2^2} = \frac{w^T s_B w}{w^T s_w w}$$

→ ~~A=diagonal dm~~

→ A dataset divided into train & test set, both PCA & LDA are applied in train data.

→ Produce PCA & LDA model

→ original data in train set is two transformed train set are produced, which contain PCs instead of original feature & other LDA.

→ Then these transformed datasets are merged so that train finally have both

Q1.) Answer →

If x is A1 (0.5) AND

y is B1 (0.2) → Then z is C1

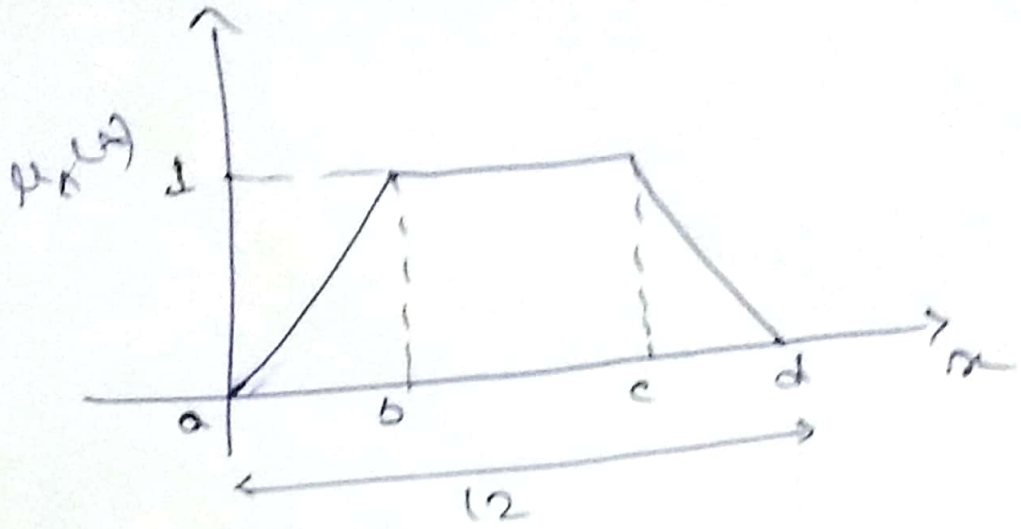
and,

If x is A1 (0.2) OR

y is B1 (0.5) then z is C1

where C1 is a membership function defined by an isosceles trapezium.

① is core $(\underline{0, 2})$, $[0, 1]$



$C_1(x: a,b,c,d) = 0$ if $x \leq a$

$= \frac{x-a}{b-a}$ if $a \leq x \leq b$

$= 1$ if $b \leq x \leq c$

$= \frac{d-x}{d-c}$ if $c \leq x \leq d$

$= 0$ if $d \leq x$

MI12d9090

1:25 PM