CS5691: PRML Assignment #2

Instructor : Prof. B. Ravindran
Topics: ANN,Ensemble Method,Kernel and SVM. **Deadline:** 12th November 2021

Teammate 1: (Pranab Kumar Rout) **Roll number:** CS21M045
Teammate 2: (Manish Kumar) **Roll number:** CS21M033

- This assignment has to be completed in teams of 2. Collaborations outside the team are strictly prohibited.
- Be precise with your explanations. Unnecessary verbosity will be penalized.
- Check the Moodle discussion forums regularly for updates regarding the assignment.
- Type your solutions in the provided LaTeX template file.
- We highly recommend using `Python 3.6+` and standard libraries like `numpy, Matplotlib, pandas`. You can choose to use your favourite programming language however the TAs will only be able to assist you with doubts related to Python.
- You are supposed to write your own algorithms, any library functions which implement these directly are strictly off the table. Using them will result in a straight zero on coding questions, `import` wisely!
- **Please start early and clear all doubts ASAP.**
- Please note that the TAs will **only** clarify doubts regarding problem statements. The TAs won't discuss any prospective solution or verify your solution or give hints.
- Post your doubt only on Moodle so everyone is on the same page.
- Posting doubts on Moodle that reveals the answer or gives hints may lead to penalty
- **Only one team member will submit the solution**
- For coding questions paste the link to your Colab Notebook of your code in the LaTeX solutions file as well as embed the result figures in your LaTeX solutions. Make sure no one other than TAs have access to the notebook. And do not delete the outputs from notebook before final submission . Any update made to notebook after deadline will result in standard late submission penalty.
- Late submission per day will attract a penalty of 10 percent of the total marks.

1. **[ANN]** In this Question, you will code a single layer ANN with Sigmoid Activation function and appropriate loss function from scratch. Train the ANN for the Dataset1 and Dataset2

   NOTE: Test Data should not be used for training.

   NOTE 2: You need to code from scratch.

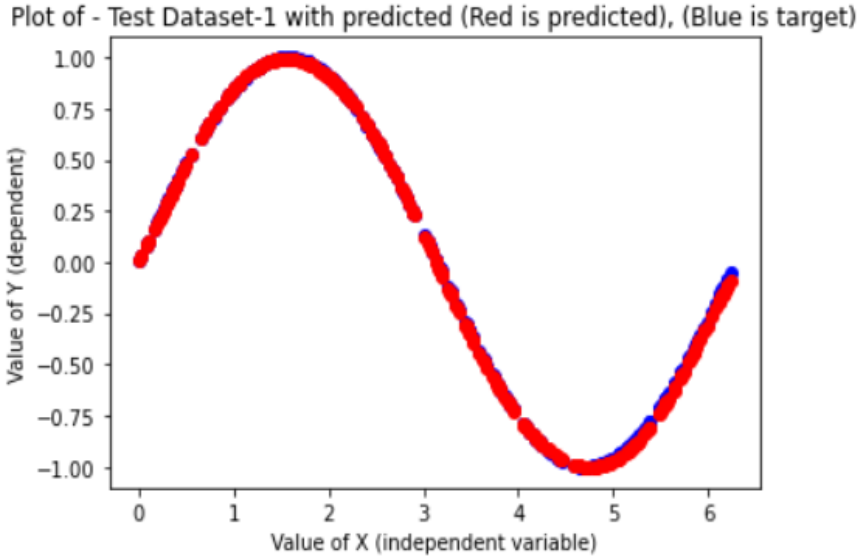
(a) (1 mark) Plot the training Data for Dataset1 and Dataset2.

Solution:

(b) (1 mark) **For data set 1** : (1) Write the number of nodes in the hidden layer and learning rate used (2) Plot Test Data and prediction on Test Data in the same graph with different colors and appropriate legend.

Solution:

No. of nodes used in hidden layer = 3
Learning Rate used = 0.2
No. of epochs to train = 5000

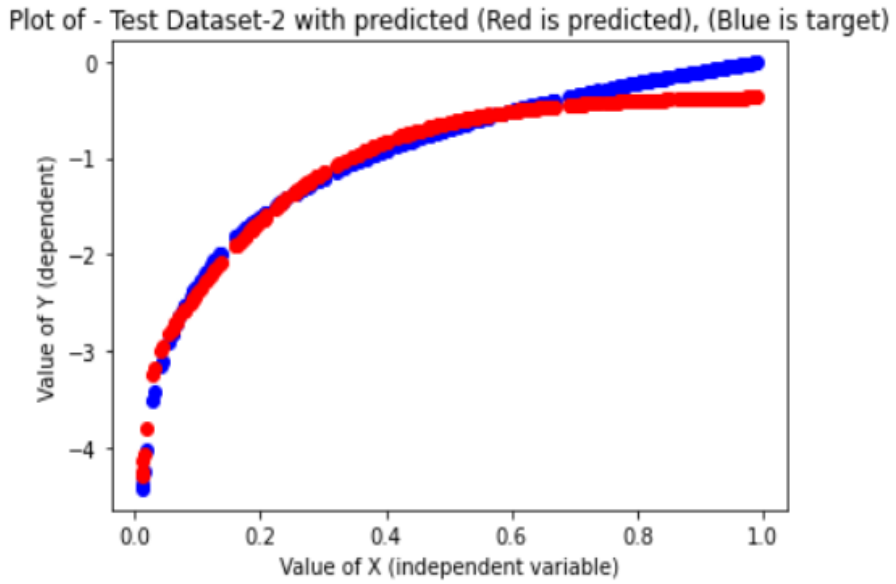


Red point - prediction on Test Data
Blue point - Actual plot on Test Data

(c) (1 mark) **For data set 2** : (1) Write the number of nodes in the hidden layer and learning rate used (2) Plot Test Data and prediction on Test Data in the same graph with different colors and appropriate legend.

Solution:

No. of nodes used in hidden layer = 2
Learning Rate used = 0.2
No. of epochs to train = 10,000



Red point - prediction on Test Data
Blue point - Actual plot on Test Data

(d) (1 mark) For each Dataset write average training Loss and average Test Loss.

Solution: For Dataset-1:
Average Training Loss = 0.00015092058683898826
Average Test Loss = 0.00013898049690079934

For Dataset-2:
Average Training Loss = 0.022944001093532954
Average Test Loss = 0.022058246129643033

(e) (1 mark) What Loss function did you use and why?

Solution:
The loss function that has been used is Mean Squared Error. For Regression problems, this is a very efficient loss function. MSE is calculated as the mean squared difference between the expected output and predicted output. The square function helps in eliminating the negative differences easily and on which our gradient descent algorithms can work to converge and give optimized result. Also when we square, for large mistakes we get larger loss.

(f) (3 marks) Paste the link to your Colab Notebook of your code. Make sure that your notebook is private and give access to all the TAs. Your Notebook must contain all the codes that you used to generate the above results. **Note :** Do not delete the outputs.

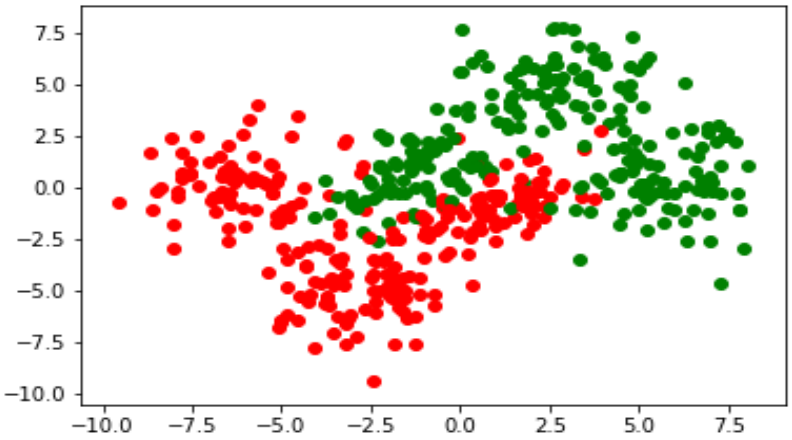
Solution: LINK
https://colab.research.google.com/drive/1JdhODdcVEWZXhgIxjutN5Cuj4JGRvFa4

2. **[AdaBoost]** In this question, you will code the AdaBoost algorithm. Follow the instructions in this Jupyter Notebook for this question. Find the dataset for the question here.

   NOTE: Test data should not be used for training.

   NOTE 2: You need to code from scratch. You can use the starter notebook though :) . Make a copy of it in your drive and start.

   (a) (1 mark) Plot the training data.

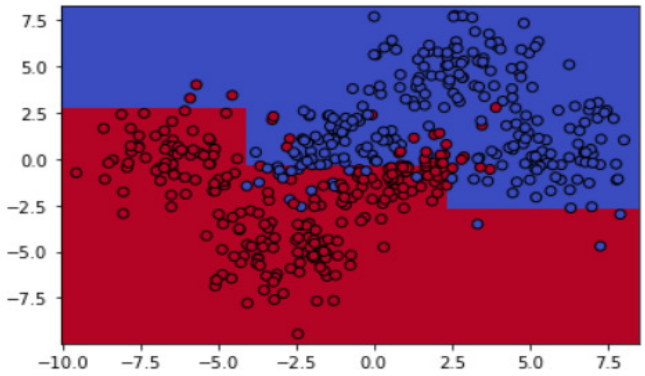


Solution:

   (b) (1 mark) **For training with k=5** : (1) Plot the learnt decision surface. (2) Write down the test accuracy.

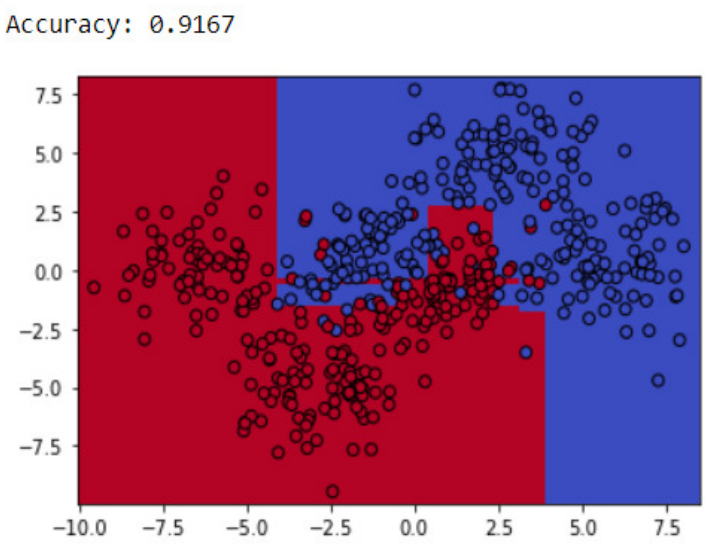
Solution: For training with k=5 $\implies$ Test Accuracy=0.90

```
Accuracy : 0.9
```



(c) (1 mark) **For training with k=100** : (1) Plot the learnt decision surface. (2) Write down the test accuracy.

Solution: For training with k=100
Test Accuracy=0.9167



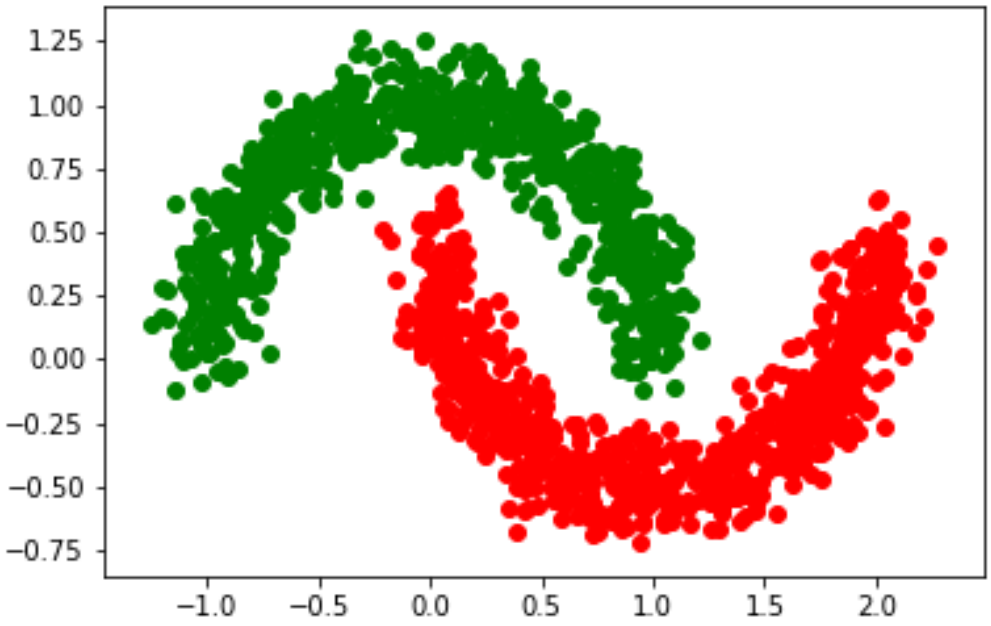
(d) (3 marks) Paste the link to your Colab Notebook of your code. Make sure that your notebook is private and give access to all the TAs. Your notebook must contain all the codes that you used to generate the above results. **Note :** Do not delete the outputs.

Solution:
LINK ADABOOST
https://colab.research.google.com/drive/1UyL7rNxrrzN5UNZcLA6MaEAC7SgcSCAF

3. **[Kernel]** Consider *Dataset_Kernel_Train.npy* and *Dataset_Kernel_Test.npy* for this question. Each row in the above matrices corresponds to a labelled data point where the first two entries correspond to its x and y co-ordinate, and the third entry $\in \{-1, 1\}$ indicates the class to which it belongs. Find the dataset for the question here. **NOTE: Test data should not be used for training.**

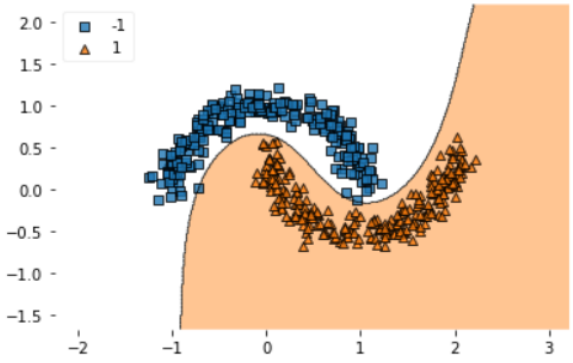
(a) (1 mark) Plot the training data points and indicate by different colours the points belonging to the different classes. Is the data linearly separable?

Solution:
Data is Not linearly separable.



(b) (1 mark) Using *sklearn.svm* (read the documentation here), bulid a classifier that classifies the data points in the testing data set using the Radial Basis Function (RBF) kernel.How do you tune the involved hyperparameters?

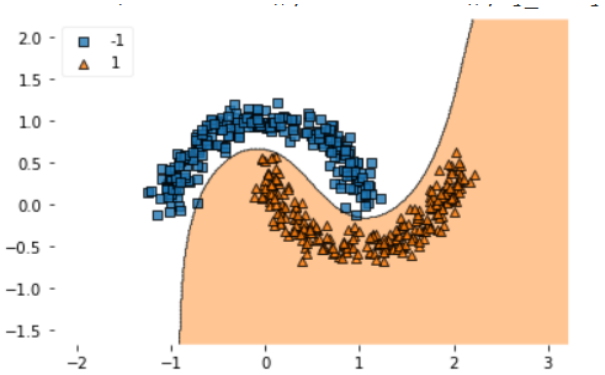
Solution:
For tuning the hyperparameters like (gamma and C), we tried with different combinations of values randomly.
gamma ranging from (0.1 to 0.5) and C ranging from (1,10,50,100)
We got an accuracy of 1 for (gamma=0.3 and C=10) right at the begining for both train and test data. So we didn't move further.

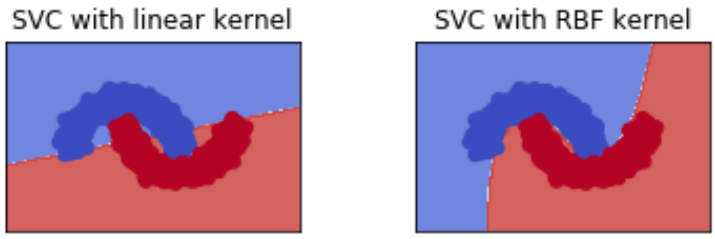


(c) (1 mark) Plot the separating curve and report the accuracy of prediction corresponding

to the tuned hyperparameters.

Solution:
Using Radial Basis Function (RBF) kernel,gamma=0.3 and C=10 Accuracy=1.0





(d) (3 marks) Paste the link to your Colab Notebook of your code. Make sure that your notebook is private and give access to all the TAs. Your notebook must contain all the codes that you used to generate the above results. **Note :** Do not delete the outputs.

Solution:
LINK SVM_RBF
https://colab.research.google.com/drive/1MHGpPj-nUIcPzePoSZAGzaRdd4mLLxbo

4. (2 marks) **[Ensemble of randomised algorithms]** Imagine we have an algorithm for solving some decision problem (*e.g.*, is a given number p a prime?). Suppose that the algorithm makes a decision at random and returns the correct answer with probability $\frac{1}{2} + \delta$, for some $\delta > 0$, which is just a bit better than a random guess. To improve the performance, we run the

algorithm N times and take the majority vote. Show that for any $\epsilon \in (0,1)$, the answer is correct with probability $1-\epsilon$, as long as $N > (1/2)\delta^{-2}\ln(\epsilon^{-1})$.

Hint 1: Try to calculate the probability with which the answer is not correct i.e. when the majority votes are not correct.

Hint 2: What value of N will you require so that the above probability is less than ϵ. Rearrange Inequalities :-)

Solution: Given P(algo is correct)=$\frac{1}{2}+\delta$
Boosting perform By running algo N times and Taking Majority Vote.
Bernauli Random Variable
$X_i = \begin{cases} 1 \text{ if algo is wrong in ith attempt} \\ 0 \text{ otherwise} \end{cases}$
$\sum_{i=1}^{N} > \frac{N}{2}$ (Boosted algo is wrong)
Want this bad event to have small probability, so upper bound its probability :
$P(\sum X_i > \frac{N}{2}) = P(\sum x_i - (\frac{1}{2}-\delta)N > N\delta)$
$<= \exp(\frac{-2N^2\delta^2}{N})$ (using chernoff inequality)
$<= \exp(-2N\delta^2)$
this is less then equal to ϵ if
$N > \frac{log(\frac{1}{\epsilon})}{2\delta^2}$
X_i being the indicators of the wrong answers *and* this is less then equal to ϵ
so the answer is correct with probability atleast $1-\epsilon$

5. (2 marks) **[Boosting]** Consider an additive ensemble model of the form $f_m(\mathbf{x}) = \frac{1}{2}\sum_{l=1}^{m}\alpha_l y_l(\mathbf{x})$, where y_l's are individual models and α_l's are weights. Show that the sequential minimization of the sum-of-squares error function for the above model trained in the style of boosting (i.e. y_m is trained after accounting the weaknesses of f_{m-1}) simply involves fitting each new base classifier y_m to the residual errors $t_n - f_{m-1}(\mathbf{x}_n)$ from previous model.

Solution: Given additive ensemble model
$\mathrm{f}_m(x) = \frac{1}{2}\sum_{i=1}^{m}\alpha_i y_i(x) - - - - - - - - - - equation(1)$
$\alpha_i -> $ weight
$y_i -> $ individual model
we have to minimize the sum of square error for above model when it is trained in the style of boosting
The sum of square error is :
Error= $\frac{1}{2}\sum_{i=1}^{m}(t_i - f_m(x_i))^2 - - - - - - - - - - equation(2)$

Effectively our goal is to minimize Error w.r.t both weight α_i and parameter q ,the base classifiers $y_i(x)$, but since additive boosting ,at each iteration 'm' we solve optimal functions and its corresponding coefficients to add to the current expansion i.e $f_{m-1}(x)$
Thus $f_m(x)$

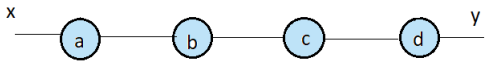
$f_m(x) = f_{m-1}(x) + \frac{1}{2}\alpha_m y_m(x)$

Put the value of $f_m(x)$ in equation(2)

Error=$\frac{1}{2}\sum_{i=1}^{m}(t_i - f_{m-1}(x) - \frac{1}{2}\alpha_m y_m(x))^2$

if we see$(t_i - f_{m-1}(x_i)$is actually he residual form of the previous step,previous(m-1) model.simply involves fitting each new base classifier ym to the residual errors from previous model.

6. (2 marks) **[Backpropagation]** We are trying to train the following chain like neural network with back-propagation. Assume that the transfer functions are sigmoid activation functions i.e. $g(x) = \frac{1}{1+e^{-x}}$. Let the input $x = 0.5$, the target output $y = 1$, all the weights are initially set to 1 and the bias for each node is -0.5.



(a) Give an expression that compares the magnitudes of the gradient updates for weights (δ) across the consecutive nodes.
(b) How does the magnitude of the gradient update vary across the network/chain as we move away from the output unit?

Solution:

Given Sigmoid activation function g(x)=$\frac{1}{1+e^{-x}}$
input x= 0.5
target output y= 1
weights are initially set to 1 and the bias for each node is -0.5

Net input at a

$a_{net} = (w_a)x + bias = 0.5 - .5 = 0$

$a_{out} = \frac{1}{1+e^{-a_{net}}} = \frac{1}{1+e^{-0}} = 0.5$

Net input at b

$b_{net} = w_b * a_{out} + bias = 1 * 0.5 - 0.5 = 0$

$$b_{out} = \frac{1}{1+e^{-b_{net}}} = \frac{1}{1+e^{-0}} = 0.5$$

Net input at c

$c_{net} = w_c * b_{out} + bias = 1 * 0.5 - 0.5 = 0$

$$c_{out} = \frac{1}{1+e^{-c_{net}}} = \frac{1}{1+e^{-0}} = 0.5$$

Net input at d

$d_{net} = w_d * c_{out} + bias = 1 * 0.5 - 0.5 = 0$

$$d_{out} = \frac{1}{1+e^{-d_{net}}} = \frac{1}{1+e^{-0}} = 0.5$$

$d_{out} \neq$ target y=1

$$\text{E}_{Total} = \text{Error} = \frac{1}{2}(target - d_{out})^2 = \frac{1}{2}(1-0.5)^2 = 0.125$$

Backpropagation

$$\frac{\partial E}{\partial w_d} = \frac{\partial E}{\partial d_{out}} * \frac{\partial d_{out}}{\partial d_{net}} * \frac{\partial d_{net}}{\partial w_d}$$

$$\frac{\partial E}{\partial d_{out}} = -(target - d_{out}) = -(1-0.5) = -0.5$$

$$\frac{\partial d_{out}}{\partial d_{net}} = d_{out}(1-d_{out}) = 0.5(1-0.5) = 0.25$$

$$\delta_d = \frac{\partial E}{\partial d_{net}} = -0.125 \qquad (1)$$

$$\frac{\partial d_{net}}{\partial w_d} = c_{out} = 0.5$$

$$\frac{\partial E}{\partial w_d} = -0.5 * 0.25 * 0.5 = -0.0625$$

For Weight W_c

$$\frac{\partial E}{\partial w_c} = \frac{\partial E}{\partial d_{net}} * \frac{\partial d_{net}}{\partial c_{out}} * \frac{\partial c_{out}}{\partial c_{net}} * \frac{\partial c_{net}}{\partial w_c}$$

$$\delta_c = \frac{\partial E}{\partial c_{net}} = \delta_d * w_d * c_{out}(1-c_{out}) \qquad (2)$$
$$= (-0.125)(0.5)(0.5) = -0.03125$$

$c_{net} = w_c * b_{out} - 0.5$

$$\frac{\partial c_{net}}{\partial w_c} = b_{out} = 0.5$$

$$\frac{\partial E}{\partial w_c} = -0.125 * 0.25 * 0.5 = -0.015625$$

For Weight W_b

$$\frac{\partial E}{\partial w_b} = \delta_b * \frac{\partial b_{net}}{\partial w_b}$$
$$\delta_b = \frac{\partial E}{\partial c_{net}} * \frac{\partial c_{net}}{\partial b_{out}} * \frac{\partial b_{out}}{\partial b_{net}}$$
$$= \delta_c * w_c * b_{out}(1 - b_{out})$$
$$= (-0.03125) * (0.5)(0.5) = -0.0078125 \quad (3)$$

$$\frac{\partial b_{net}}{\partial w_b} = a_{out} = 0.5$$
$$\frac{\partial E}{\partial w_b} = -0.03125 * 0.25 * 0.5 = -0.00390625$$

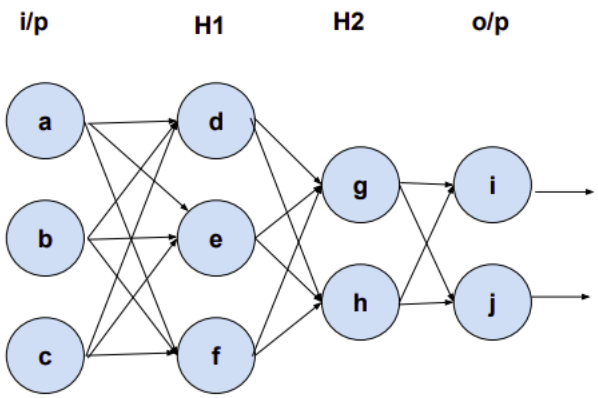
For Weight W_a
$$\frac{\partial E}{\partial w_a} = \delta_a * \frac{\partial a_{net}}{\partial w_a}$$
$$\delta_a = \frac{\partial E}{\partial b_{net}} * \frac{\partial b_{net}}{\partial a_{out}} * \frac{\partial a_{out}}{\partial a_{net}}$$
$$= \delta_b * w_b * a_{out}(1 - a_{out})$$
$$= (-0.0078125) * (0.5)(0.5) = -0.001953125 \quad (4)$$

$$\frac{\partial a_{net}}{\partial w_a} = x = 0.5$$
$$\frac{\partial E}{\partial w_a} = -0.0078125 * 0.25 * 0.5 = -0.0009765625$$

1. $\frac{\delta_c}{\delta_d} = \frac{\delta_d * 0.25}{\delta_d} = 0.25$
2. $\frac{\delta_b}{\delta_c} = \frac{\delta_c * 0.25}{\delta_c} = 0.25$
3. $\frac{\delta_a}{\delta_b} = \frac{\delta_b * 0.25}{\delta_b} = 0.25$

δ for every node is scaling down by 0.25 as compared to δ of adjacent node

7. (2 marks) **[NN & Activation Functions]** The following diagram represents a feed-forward network with two hidden layers.



A weight on connection between nodes x and y is denoted by w_{xy}, such as w_{ad} is the weight on the connection between nodes a and d. The following table lists all the weights in the network:

$w_{ad} = 0.5$	$w_{be} = -1.4$	$w_{cf} = -1.25$	$w_{eh} = -2$	$w_{gj} = -1.5$
$w_{ae} = 0.9$	$w_{bf} = 0.75$	$w_{dg} = 1$	$w_{fg} = 3$	$w_{hi} = 0.5$
$w_{af} = -2$	$w_{cd} = 0$	$w_{dh} = 3$	$w_{fh} = 1.25$	$w_{hj} = -0.25$
$w_{bd} = 1.3$	$w_{ce} = 0.3$	$w_{eg} = 2.5$	$w_{gi} = 2.5$	

Find the output of the network for the following input vectors:
$V_1 = [0.2, 1, 3]$, $V2 = [2.5, 3, 7]$, $V3 = [0.75, -2, 3]$

(a) If *sigmoid* activation function is used in both H1 & H2

(b) If *tanh* activation function is used in both H1 & H2

(c) If *sigmoid* activation is used in H1 and *tanh* in H2

(d) If *tanh* activation is used in H1 & ReLU activation in H2

Please provide all steps and explain the same.

Solution:

(a) V_1=[0.2,1,3] $\implies$ $a = 0.2,\ b = 1,\ c = 3$

$$\text{Pre at H1} = \begin{bmatrix} 0.2*0.5 + 1*1.3 + 3*0 \\ 0.2*0.9 - 1*1.4 + 3*0.3 \\ -0.2*2 + 1*0.75 - 3*1.25 \end{bmatrix} = \begin{bmatrix} 1.4 \\ -0.32 \\ -3.4 \end{bmatrix}$$

$$\text{out at H1} = \begin{bmatrix} sigmoid(1.4) \\ sigmoid(-0.32) \\ sigmoid(-3.4) \end{bmatrix} = \begin{bmatrix} 0.80218389 \\ 0.42067575 \\ 0.03229546 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.80218389 \\ 0.42067575 \\ 0.03229546 \end{bmatrix} = \begin{bmatrix} 1.95075965 \\ 1.6055695 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} sigmoid(1.95075965) \\ sigmoid(1.6055695) \end{bmatrix} = \begin{bmatrix} 0.87552945 \\ 0.83279536 \end{bmatrix}$$

$$\text{Net o/p at } \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0.87552945 \\ 0.83279536 \end{bmatrix} = \begin{bmatrix} 2.60522131 \\ -1.52149302 \end{bmatrix}$$

Final output at $(i, j) = [\mathbf{2.60522131} \quad \mathbf{-1.52149302}]$

V_2=[2.5, 3, 7] $\implies a = 2.5,\ b = 3,\ c = 7$

$$\text{Pre at H1} = \begin{bmatrix} 2.5 * 0.5 + 3 * 1.30 + 7 * 0 \\ 2.5 * 0.9 - 3 * 1.40 + 7 * 0.3 \\ -2.5 * 2.0 + 3 * 0.75 - 7 * 1.25 \end{bmatrix} = \begin{bmatrix} 5.15 \\ 0.15 \\ -11.5 \end{bmatrix}$$

$$\text{out at H1} = \begin{bmatrix} sigmoid(5.15) \\ sigmoid(0.15) \\ sigmoid(-11.5) \end{bmatrix} = \begin{bmatrix} 0.99423403 \\ 0.53742984 \\ 0.00001013 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.99423403 \\ 0.53742984 \\ 0.00001013 \end{bmatrix} = \begin{bmatrix} 2.33783904 \\ 1.90785508 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} sigmoid(2.33783904) \\ sigmoid(1.90785508) \end{bmatrix} = \begin{bmatrix} 0.91196274 \\ 0.87077798 \end{bmatrix}$$

$$\text{Net o/p at } \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0.91196274 \\ 0.87077798 \end{bmatrix} = \begin{bmatrix} 2.71529585 \\ -1.58563861 \end{bmatrix}$$

Final output at $(i, j) = [\mathbf{2.71529585} \quad \mathbf{-1.58563861}]$

V_3=[0.75, -2, 3] $\implies a = 0.75,\ b = -2,\ c = 3$

$$\text{Pre at H1} = \begin{bmatrix} 0.75 * 0.5 - 2 * 1.30 + 3 * 0 \\ 0.75 * 0.9 + 2 * 1.40 + 3 * 0.3 \\ -0.75 * 2.0 - 2 * 0.75 - 3 * 1.25 \end{bmatrix} = \begin{bmatrix} -2.225 \\ 4.375 \\ -6.75 \end{bmatrix}$$

$$\text{out at H1} = \begin{bmatrix} sigmoid(-2.225) \\ sigmoid(4.375) \\ sigmoid(-6.75) \end{bmatrix} = \begin{bmatrix} 0.09752784 \\ 0.98756835 \\ 0.00116951 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.09752784 \\ 0.98756835 \\ 0.00116951 \end{bmatrix} = \begin{bmatrix} 2.56995724 \\ -1.6810913 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} sigmoid(2.56995724) \\ sigmoid(-1.6810913) \end{bmatrix} = \begin{bmatrix} 0.92890287 \\ 0.15695102 \end{bmatrix}$$

$$\text{Net o/p at } \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0.92890287 \\ 0.15695102 \end{bmatrix} = \begin{bmatrix} 2.40073269 \\ -1.43259206 \end{bmatrix}$$

Final output at $(i, j) = [\mathbf{2.40073269} \quad \mathbf{-1.43259206}]$

(b) V_1=[0.2,1,3] $\implies a = 0.2,\ b = 1,\ c = 3$

$$\text{Pre at H1} = \begin{bmatrix} 0.2 * 0.5 + 1 * 1.3 + 3 * 0 \\ 0.2 * 0.9 - 1 * 1.4 + 3 * 0.3 \\ -0.2 * 2 + 1 * 0.75 - 3 * 1.25 \end{bmatrix} = \begin{bmatrix} 1.4 \\ -0.32 \\ -3.4 \end{bmatrix}$$

$$\text{out at H1} = \begin{bmatrix} tanh(1.4) \\ tanh(-0.32) \\ tanh(-3.4) \end{bmatrix} = \begin{bmatrix} 0.88535165 \\ -0.30950692 \\ -0.99777493 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.88535165 \\ -0.30950692 \\ -0.99777493 \end{bmatrix} = \begin{bmatrix} -2.88174044 \\ 2.02785013 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} tanh(-2.88174044) \\ tanh(2.02785013) \end{bmatrix} = \begin{bmatrix} -0.99373934 \\ 0.96594329 \end{bmatrix}$$

$$\text{Net o/p at } \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} -0.99373934 \\ 0.96594329 \end{bmatrix} = \begin{bmatrix} -2.0013767 \\ 1.24912318 \end{bmatrix}$$

Final output at $(i, j) = [\mathbf{-2.0013767 \;\; 1.24912318}]$

V_2=[2.5, 3, 7] $\implies a = 2.5,\ b = 3,\ c = 7$

$$\text{Pre at H1} = \begin{bmatrix} 2.5 * 0.5 + 3 * 1.30 + 7 * 0 \\ 2.5 * 0.9 - 3 * 1.40 + 7 * 0.3 \\ -2.5 * 2.0 + 3 * 3.75 - 7 * 1.25 \end{bmatrix} = \begin{bmatrix} 5.15 \\ 0.15 \\ -11.5 \end{bmatrix}$$

$$\text{out at H1} = \begin{bmatrix} tanh(5.15) \\ tanh(0.1) \\ tanh(-11.5) \end{bmatrix} = \begin{bmatrix} 0.99993274 \\ 0.14888503 \\ -0.9999999 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.99993274 \\ 0.14888503 \\ -0.9999999 \end{bmatrix} = \begin{bmatrix} -1.62785468 \\ 1.45202814 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} tanh(-1.62785468) \\ tanh(1.45202814) \end{bmatrix} = \begin{bmatrix} -0.92575546 \\ 0.89609318 \end{bmatrix}$$

$$\text{Net o/p at } \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} -0.92575546 \\ 0.89609318 \end{bmatrix} = \begin{bmatrix} -1.86634206 \\ 1.16460989 \end{bmatrix}$$

Final output at $(i, j) = [\mathbf{-1.86634206 \;\; 1.16460989}]$

V_3=[0.75, -2, 3] $\implies a = 0.75,\ b = -2,\ c = 3$

$$\text{Pre at H1} = \begin{bmatrix} 0.75 * 0.5 - 2 * 1.30 + 3 * 0 \\ 0.75 * 0.9 + 2 * 1.40 + 3 * 0.3 \\ -0.75 * 2.0 - 2 * 0.75 - 3 * 1.25 \end{bmatrix} = \begin{bmatrix} -2.225 \\ 4.375 \\ -6.75 \end{bmatrix}$$

$$\text{out at H1} = \begin{bmatrix} tanh(-2.225) \\ tanh(4.375) \\ tanh(-6.75) \end{bmatrix} = \begin{bmatrix} -0.9769125 \\ 0.99968313 \\ -0.99999726 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} -0.9769125 \\ 0.99968313 \\ -0.99999726 \end{bmatrix} = \begin{bmatrix} -1.47769645 \\ -6.18010031 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} tanh(-1.47769645) \\ tanh(-6.18010031) \end{bmatrix} = \begin{bmatrix} 0.90103551 \\ -0.99999143 \end{bmatrix}$$

Net o/p at $\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0.90103551 \\ -0.99999143 \end{bmatrix} = \begin{bmatrix} -2.75258448 \\ 1.60155112 \end{bmatrix}$

Final output at $(i, j) = [\mathbf{-2.75258448 \;\; 1.60155112}]$

(c) V_1=[0.2,1,3] $\implies a = 0.2,\ b = 1,\ c = 3$

Pre at H1 $= \begin{bmatrix} 0.2 * 0.5 + 1 * 1.3 + 3 * 0 \\ 0.2 * 0.9 - 1 * 1.4 + 3 * 0.3 \\ -0.2 * 2 + 1 * 0.75 - 3 * 1.25 \end{bmatrix} = \begin{bmatrix} 1.4 \\ -0.32 \\ -3.4 \end{bmatrix}$

out at H1 $= \begin{bmatrix} sigmoid(1.4) \\ sigmoid(-0.32) \\ sigmoid(-3.4) \end{bmatrix} = \begin{bmatrix} 0.80218389 \\ 0.42067575 \\ 0.03229546 \end{bmatrix}$

Pre at H2 $= \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.80218389 \\ 0.42067575 \\ 0.03229546 \end{bmatrix} = \begin{bmatrix} 1.95075965 \\ 1.6055695 \end{bmatrix}$

out at H2 $= \begin{bmatrix} tanh(1.95075965) \\ tanh(1.6055695) \end{bmatrix} = \begin{bmatrix} 0.96037844 \\ 0.92250262 \end{bmatrix}$

Net o/p at $\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0.96037844 \\ 0.92250262 \end{bmatrix} = \begin{bmatrix} 2.8621974 \\ -1.67119331 \end{bmatrix}$

Final output at $(i, j) = [\mathbf{2.8621974 \quad -1.67119331}]$

V_2=[2.5, 3, 7] $\implies a = 2.5,\ b = 3,\ c = 7$

Pre at H1 $= \begin{bmatrix} 2.5 * 0.5 + 3 * 1.30 + 7 * 0 \\ 2.5 * 0.9 - 3 * 1.40 + 7 * 0.3 \\ -2.5 * 2.0 + 3 * 0.75 - 7 * 1.25 \end{bmatrix} = \begin{bmatrix} 5.15 \\ 0.15 \\ -11.5 \end{bmatrix}$

out at H1 $= \begin{bmatrix} sigmoid(5.15) \\ sigmoid(0.15) \\ sigmoid(-11.5) \end{bmatrix} = \begin{bmatrix} 0.99423403 \\ 0.53742984 \\ 0.00001013 \end{bmatrix}$

Pre at H2 $= \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.99423403 \\ 0.53742984 \\ 0.00001013 \end{bmatrix} = \begin{bmatrix} 2.33783904 \\ 1.90785508 \end{bmatrix}$

out at H2 $= \begin{bmatrix} tanh(2.33783904) \\ tanh(1.90785508) \end{bmatrix} = \begin{bmatrix} 0.98153368 \\ 0.9569049 \end{bmatrix}$

Net o/p at $\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0.98153368 \\ 0.9569049 \end{bmatrix} = \begin{bmatrix} 2.93228666 \\ -1.71152675 \end{bmatrix}$

Final output at $(i, j) = [\mathbf{2.93228666 \quad -1.71152675}]$

V_3=[0.75, -2, 3] $\implies a = 0.75,\ b = -2,\ c = 3$

Pre at H1 $= \begin{bmatrix} 0.75 * 0.5 - 2 * 1.30 + 3 * 0 \\ 0.75 * 0.9 + 2 * 1.40 + 3 * 0.3 \\ -0.75 * 2.0 - 2 * 0.75 - 3 * 1.25 \end{bmatrix} = \begin{bmatrix} -2.225 \\ 4.375 \\ -6.75 \end{bmatrix}$

$$\text{out at H1} = \begin{bmatrix} sigmoid(-2.225) \\ sigmoid(4.375) \\ sigmoid(-6.75) \end{bmatrix} = \begin{bmatrix} 0.09752784 \\ 0.98756835 \\ 0.00116951 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.09752784 \\ 0.98756835 \\ 0.00116951 \end{bmatrix} = \begin{bmatrix} 2.56995724 \\ -1.6810913 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} tanh(2.56995724) \\ tanh(-1.6810913) \end{bmatrix} = \begin{bmatrix} 0.98835186 \\ -0.93300303 \end{bmatrix}$$

$$\text{Net o/p at } \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0.98835186 \\ -0.93300303 \end{bmatrix} = \begin{bmatrix} 2.00437813 \\ -1.24927703 \end{bmatrix}$$

Final output at $(i, j) = [\mathbf{2.00437813} \quad -\mathbf{1.24927703}]$

(d) V_1=[0.2,1,3] $\implies a = 0.2,\ b = 1,\ c = 3$

$$\text{Pre at H1} = \begin{bmatrix} 0.2 * 0.5 + 1 * 1.3 + 3 * 0 \\ 0.2 * 0.9 - 1 * 1.4 + 3 * 0.3 \\ -0.2 * 2 + 1 * 0.75 - 3 * 1.25 \end{bmatrix} = \begin{bmatrix} 1.4 \\ -0.32 \\ -3.4 \end{bmatrix}$$

$$\text{out at H1} = \begin{bmatrix} tanh(1.4) \\ tanh(-0.32) \\ tanh(-3.4) \end{bmatrix} = \begin{bmatrix} 0.88535165 \\ -0.30950692 \\ -0.99777493 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.88535165 \\ -0.30950692 \\ -0.99777493 \end{bmatrix} = \begin{bmatrix} -2.88174044 \\ 2.02785013 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} reLU(-2.88174044) \\ reLU(2.02785013) \end{bmatrix} = \begin{bmatrix} 0 \\ 2.02785013 \end{bmatrix}$$

$$\text{Net o/p at } \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0 \\ 2.02785013 \end{bmatrix} = \begin{bmatrix} 1.01392506 \\ -0.50696253 \end{bmatrix}$$

Final output at $(i, j) = [\mathbf{1.01392506} \quad -\mathbf{0.50696253}]$

V_2=[2.5, 3, 7] $\implies a = 2.5,\ b = 3,\ c = 7$

$$\text{Pre at H1} = \begin{bmatrix} 2.5 * 0.5 + 3 * 1.30 + 7 * 0 \\ 2.5 * 0.9 - 3 * 1.40 + 7 * 0.3 \\ -2.5 * 2.0 + 3 * 3.75 - 7 * 1.25 \end{bmatrix} = \begin{bmatrix} 5.15 \\ 0.15 \\ -11.5 \end{bmatrix}$$

$$\text{out at H1} = \begin{bmatrix} tanh(5.15) \\ tanh(0.1) \\ tanh(-11.5) \end{bmatrix} = \begin{bmatrix} 0.99993274 \\ 0.14888503 \\ -0.9999999 \end{bmatrix}$$

$$\text{Pre at H2} = \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.99993274 \\ 0.14888503 \\ -0.9999999 \end{bmatrix} = \begin{bmatrix} -1.62785468 \\ 1.45202814 \end{bmatrix}$$

$$\text{out at H2} = \begin{bmatrix} reLU(-1.62785468) \\ reLU(1.45202814) \end{bmatrix} = \begin{bmatrix} 0 \\ 1.45202814 \end{bmatrix}$$

Net o/p at $\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0 \\ 1.45202814 \end{bmatrix} = \begin{bmatrix} 0.72601407 \\ -0.36300704 \end{bmatrix}$

Final output at $(i, j) = [\mathbf{0.72601407} \quad \mathbf{-0.36300704}]$

V_3=[0.75, -2, 3] $\implies a = 0.75,\ b = -2,\ c = 3$

Pre at H1 $= \begin{bmatrix} 0.75 * 0.5 - 2 * 1.30 + 3 * 0 \\ 0.75 * 0.9 + 2 * 1.40 + 3 * 0.3 \\ -0.75 * 2.0 - 2 * 0.75 - 3 * 1.25 \end{bmatrix} = \begin{bmatrix} -2.225 \\ 4.375 \\ -6.75 \end{bmatrix}$

out at H1 $= \begin{bmatrix} tanh(-2.225) \\ tanh(4.375) \\ tanh(-6.75) \end{bmatrix} = \begin{bmatrix} -0.9769125 \\ 0.99968313 \\ -0.99999726 \end{bmatrix}$

Pre at H2 $= \begin{bmatrix} 1 & 2.5 & 3.00 \\ 3 & -2.0 & 1.25 \end{bmatrix} \begin{bmatrix} -0.9769125 \\ 0.99968313 \\ -0.99999726 \end{bmatrix} = \begin{bmatrix} -1.47769645 \\ -6.18010031 \end{bmatrix}$

out at H2 $= \begin{bmatrix} reLU(-1.47769645) \\ reLU(-6.18010031) \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$

Net o/p at $\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 2.5 & 0.5 \\ -1.5 & -0.25 \end{bmatrix} \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$

Final output at $(i, j) = [\mathbf{0.0} \ \ \mathbf{0.0}]$

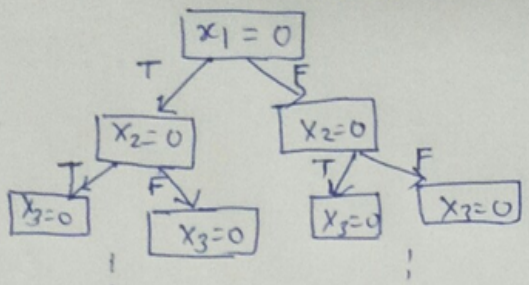
8. (2 marks) **[Decision Trees]** Consider a dataset with each data point $x \in \{0, 1\}^m$, i.e., x is a binary valued feature vector with m features, and the class label $y \in \{+1, -1\}$. Suppose the true classifier is a majority vote over the features, such that
$y = sign\Big(\sum_{i=1}^{m} (2x_i - 1)\Big)$

where x_i is the i^{th} component of the feature vector. Suppose you build a binary decision tree with minimum depth, that is consistent with the data described above. What is the range of of number of leaves which such a decision tree will have?

Solution:

m = height of the tree
Minimum Leaf Node = $2^{\frac{m}{2}}$
Maximum leaf node = 2^m

(8) The decision tree look like



Min. leaf node $= 2^{m/2}$

Max^m leaf node $= 2^m$

$m \rightarrow$ heigh of the tree.