CS5691: PRML Assignment #2

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

Teammate 1: Karthikeyan S **Roll number:** CS21M028
Teammate 2: N Kausik **Roll number:** CS21M037

- 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.

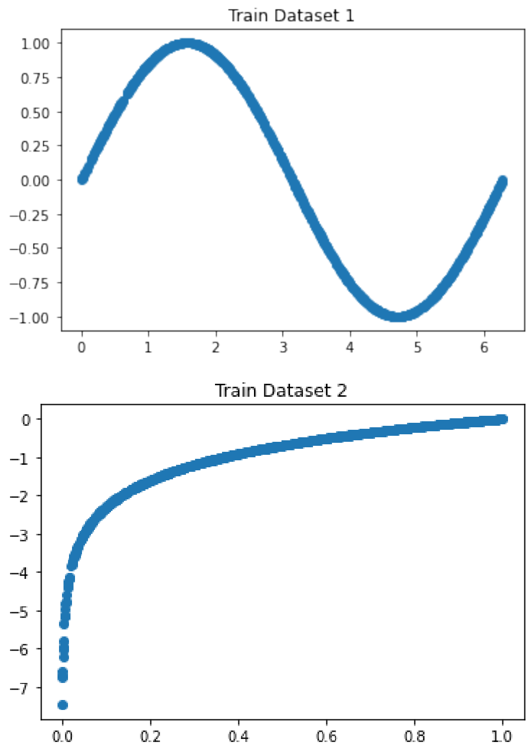
1. **[ANN]** In this Question, you will code a single layer ANN with Sigmoid Activation function and appropriate loss function from scratch. Train the ANN for the Dataset1 and Dataset2

   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.

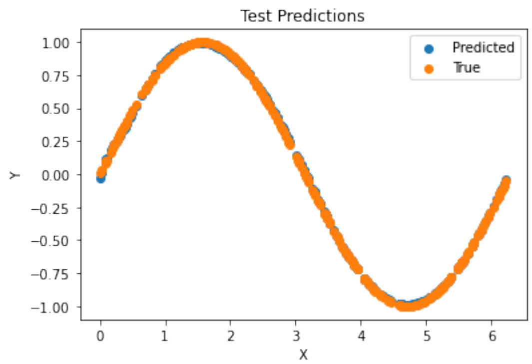
Solution:

(1)

Number of nodes in hidden layer = 100

Learning Rate = 0.05

(2)



(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.

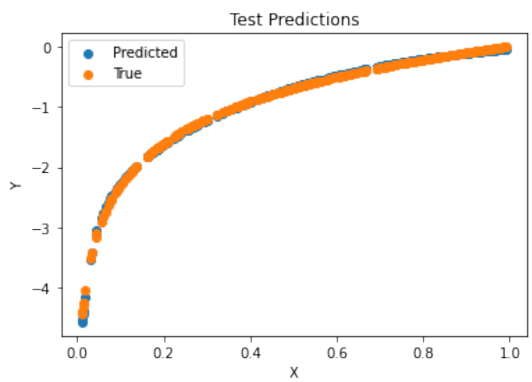
Solution:

(1)

Number of nodes in hidden layer = 100

Learning Rate = 0.1

(2)



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

Solution:

Dataset 1

Average Training Loss = 9.888204278306072147e-05

Average Test Loss = 9.844131432442626353 4e-05

Dataset 2

Average Training Loss = 0.007744668299998842338

Average Test Loss = 0.000717767987738861843 8

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

Solution:

We used the MSE (Mean Square Error) loss function. Here, we are trying to fit a function to the given dataset distributions and the output is a continuous value. This makes it a regression problem and MSE generally works well for regression problems.

(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:

Colab Notebook Link

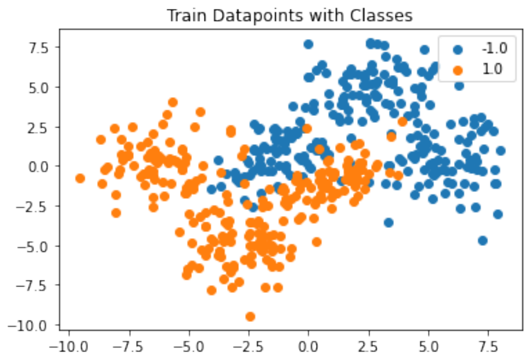
2. [**AdaBoost**] In this question, you will code the AdaBoost algorithm. Follow the instructions in this Jupyter Notebook for this question. Find the dataset for the question here.

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.

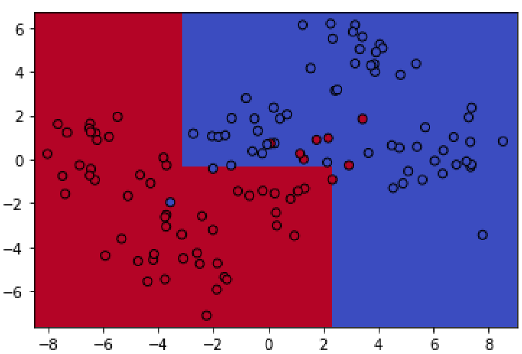
Solution:



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

Solution:

(1)

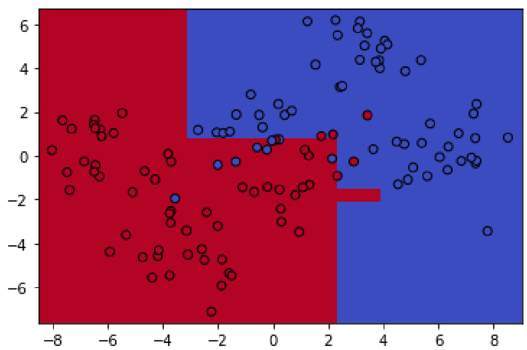


(2)
Test Accuracy = 92.5%

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

Solution:

(1)



(2)

Test Accuracy = 90.0%

(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:

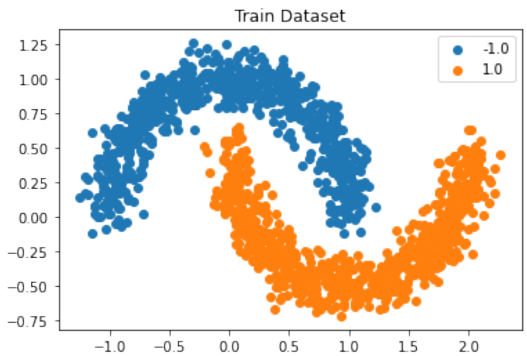
Colab Notebook Link

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:

From the plot, the 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:

Since there are only 1500 training data points, the SVM fitting takes very less time. Hence for tuning hyperparams we ran a Exhaustive Search by running for a equally spaced set of 10 values for C ranging from 0.1 to 2.0 and gamma ranging from 0.01 to 1.0. Then the hyperparams with least missclassifications on the test data was considered.

(c) (1 mark) Plot the separating curve and report the accuracy of prediction corresponding to the tuned hyperparameters.

Solution:

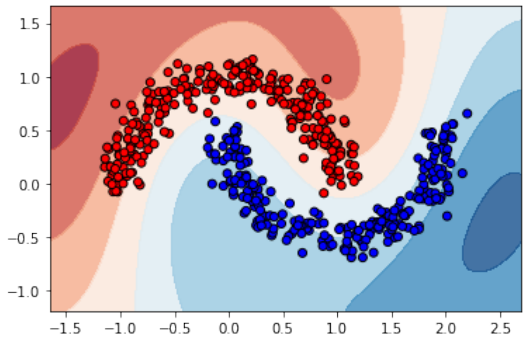
Final Hyperparams were,

C = Regularisation Parameter = 2.0

gamma = Kernel coefficient for RGF kernel = 0.45

Missclassifications on test data = 0 / 500

Accuracy = 100%



(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:

Colab Notebook Link

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:

We can make use of the Hoeffding's inequality as follows.
Let us assume we have independent random variables, $X_i \in [N]$
Let us assume that $\forall$ i $X_i \in [a_i, b_i]$ then $\forall$ t ≥ 0,
$P(S_n - E[S_n] \geq k) \leq e^{\frac{-2k^2}{\sum_{i=1}^{N}(b_i-a_i)^2}}$ where $S_n = \sum_{i=1}^{N}(X_i)$

Now let us prove it.
As per the hint let us solve this by considering the answers that is not correct, so that later it can be extended to the answers which is correct.
Let $X_1, X_2, X_3, \ldots, X_n$ be n independent random variables denoting the outcomes in multiple Bernoulli trials.
Let us represent the success(wrong guess) and failure(correct guess) by 0 and 1 respectively (as per hint 1).
We know that, Probability that algorithm returns correct answer $= \frac{1}{2} + \delta$, $\exists \delta > 0$
$\implies$ Probability that algorithm returns wrong answer $= \frac{1}{2} - \delta$, $\exists \delta > 0$

Let the probability of success in a Bernoulli trial is denoted by p, then the Expectation of Bernoulli trial $= \mathbf{E}(x) = p$ and we , as per the hint 1, consider the success as getting the wrong answer.
$\implies E(X_i) = \frac{1}{2} - \delta$

Let us consider a case where the number of correct and wrong guesses are same.
$\implies$ The number of wrong guesses in N trials$=\sum_{i=1}^{N}(X_i) = \dfrac{N}{2}$

$\therefore \sum_{i=1}^{N}(X_i - E(X_i)) = \sum_{i=1}^{N}(X_i - E(X_i)) = \sum_{i=1}^{N}(X_i) - \sum_{i=1}^{N}(E(X_i)) = \delta N$

Let us consider a case where the number of wrong guesses is atleast the number correct guesses.
$\implies$ The number of wrong guesses in N trials$=\sum_{i=1}^{N}(X_i) \geq \dfrac{N}{2}$

$\therefore \sum_{i=1}^{N}(X_i - E(X_i)) = \sum_{i=1}^{N}(X_i - E(X_i)) \geq \delta N$

Let us substitute $a_i = 0$ and $b_i = 1$ on the Hoeffding's inequality, we get

$$\implies P(\sum_{i=1}^{N}(X_i - E(X_i)) \geq \delta N) \leq e^{\dfrac{-2(\delta N)^2}{N}}$$

$$\implies P(\sum_{i=1}^{N}(X_i - E(X_i)) \geq \delta N) \leq e^{-2\delta^2 N}$$

As per hint 2, we need to find the N for which the probability that answer is wrong is less than equal to ϵ. $\implies$ N is to be found for which the probability that answer is correct is aleast $1 - \epsilon$.
So by Hoeffding's inequality, $P(\sum_{i=1}^{N}(X_i - E(X_i)) \geq \delta N) \leq e^{-2\delta^2 N} \leq \epsilon \implies e^{-2\delta^2 N} \leq \epsilon$

Let us take log on both sides, $\implies -2\delta^2 N \leq \ln(\epsilon)$

Multiplying by the above relation by -1, $\implies 2\delta^2 N \geq -\ln(\epsilon)$

$\implies N \geq \dfrac{1}{2\delta^2}\ln(\dfrac{1}{\epsilon}) \implies N \geq \dfrac{1}{2\delta^2}\ln(\epsilon^{-1})$

$\therefore$ It is shown that for any $\epsilon \in (0, 1)$, the answer is correct with probability $1 - \epsilon$, as long as $N \geq (1/2)\delta^{-2}\ln(\epsilon^{-1})$.

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:

Let the dataset be represented as $\{(x_1,t_1),(x_2,t_2),...,(x_n,t_n)\}$, where x_i and t_i denote the features and target label of the i^{th} data point respectively

Let us also assume that y_i denote the i^{th} classifier with it's corresponding weight α_i.

It is given that the **additive ensemble model** of the form $f_m(\mathbf{x}) = \dfrac{1}{2}\sum_{l=1}^{m}\alpha_l y_l(\mathbf{x})$ is used.

$\therefore$ The **sum of squares error** is given by, $\text{Error}(f_k(\mathbf{x})) = \sum_{i=1}^{n}(t_i - f_k(\mathbf{x}_i))^2$

Then for an intermediate step **k**, the additive ensemble model is given by,
$f_k(\mathbf{x}) = \dfrac{1}{2}\sum_{l=1}^{k}\alpha_l y_l(\mathbf{x})$

The next step **k+1** of the additive ensemble model can be represented as follows:
$f_{k+1}(\mathbf{x}) = \dfrac{1}{2}\left(\sum_{l=1}^{k}\alpha_l y_l(\mathbf{x})\right) + \dfrac{1}{2}\alpha_{k+1}y_{k+1}(\mathbf{x})$

Then the error for the **k+1**th step, $Error(f_{k+1}(\mathbf{x})) = \sum_{i=1}^{n}(t_i - f_{k+1}(x_i))^2$

As additive ensemble is considered, based on this error, the parameters α_{k+1} and y_{k+1} gets updated on the k+1th step, so as to improve the classifier at the next step.

$\min(\text{Error}(f_{k+1})) = \min\left(\sum_{i=1}^{n}(t_i - f_{k+1}(x_i)\right)^2$

$\implies \min\left(\sum_{i=1}^{n}\left(t_i - \left(\frac{1}{2}\sum_{l=1}^{k+1}\alpha_l y_l(\mathbf{x})\right)\right)^2\right)$

$\implies \text{argmin}_{(\alpha_{k+1},y_{k+1})}\sum_{i=1}^{n}\left(t_i - \frac{1}{2}\left(\sum_{l=1}^{k}\alpha_l y_l(\mathbf{x})\right) - \frac{1}{2}\alpha_{k+1}y_{k+1}(\mathbf{x})\right)^2$

we know that $f(k) = \frac{1}{2}\left(\sum_{l=1}^{k}\alpha_l y_l(\mathbf{x})\right)$

$\implies \min(\text{Error}(f_{k+1})) = \text{argmin}_{(\alpha_{k+1},y_{k+1})}\sum_{i=1}^{n}\left(t_i - f(k) - \frac{1}{2}\alpha_{k+1}y_{k+1}(\mathbf{x})\right)^2$

$t_i - f(k)$ in the above relation represents the residual error with respect to the previous model.

$\implies \min(\text{Error}(f_{k+1})) = \text{argmin}_{(\alpha_{k+1},y_{k+1})}\sum_{i=1}^{n}\left(\text{Residual}_{i,k} - \dfrac{1}{2}\alpha_{k+1}y_{k+1}(\mathbf{x})\right)^2$

As we can observe that the y_{k+1} is found by fitting the residual errors of the previous model (y_k) and this is done by minimizing the sum of squares error.
$\therefore$ We can observe that minimizing the sum of squares error is just fitting y_{k+1} over residual errors of y_k.

Now let us consider the above relation for the mth step which is given by,

$\implies \min(\text{Error}(f_m)) = \text{argmin}_{(\alpha_m, y_m)} \sum_{i=1}^{n} \left(\text{Residual}_{i,m-1} - \frac{1}{2}\alpha_m y_m(\mathbf{x})\right)^2$

$\therefore$We have shown that **Minimizing the sum of squares error of y_m is just fitting y_m to the residual errors of y_{m-1}.**

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:

Let the Error be calculated through the Loss function, E=$\frac{1}{2}(y-\hat{y})^2$, here y is the true output and $\hat{y}$ is the predicted output.
Let us interpret the neural network given as follows
x represents the input layer where inputs are given and a, b, c, and d are hidden layers and y is the output layer where we get the output.
Consider the weights between nodes (x,a), (a,b),(b,c),(c,d), (d,y) be $w_{xa}, w_{ab}, w_{bc}, w_{cd}, w_{dy}$ respectively and it is given that all these weights are initialized to 1 initially.
Also consider the bias at nodes a, b, c, and d be b_a, b_b, b_c, b_d respectively and it is given that all bias values are -0.5 initially
The activation function to be used at each hidden layer is given by sigmoid(x), $g(x) = \frac{1}{1+e^{-x}}$ and the derivative of sigmoid, $g(x)' = g(x)(1-g(x))$
Let $\delta_1, \delta_2, \delta_3, \delta_4, \delta_5$ be the magnitudes of gradient updates for the weights across nodes a, b, c, d, and output layer(y) respectively.
To discuss about the back propagation, let us first compute the error through a forward pass.

Forward Pass
Let the subscript in and out represent input and output at a specific node.

Node a: $a_{in} = x * w_{xa} + b_a = 0$ and $a_{\text{out}} = g(a_{in})$=g(0)=0.5
Node b: $b_{in} = a_{\text{out}} * w_{ab} + b_b = 0$ and $b_{\text{out}} = g(b_{in})$=g(0)=0.5
Node c: $c_{in} = b_{\text{out}} * w_{bc} + b_c = 0$ and $c_{\text{out}} = g(c_{in})$=g(0)=0.5
Node d: $d_{in} = c_{\text{out}} * w_{cd} + b_d = 0$ and $d_{\text{out}} = g(c_{in})$=g(0)=0.5
Predicted output= $\hat{y} = d_{\text{out}} * w_{dy}$=0.5

Error=$\frac{1}{2}(y-\hat{y})^2 = 0.125$

Back Propagation
In contrary to the forward pass, we start the updates from the output layer.

Output layer(y):

we know that Error=$\frac{1}{2}(y-\hat{y})^2$
We need to find how much does the Error change with respect to output(predicted).
$\delta_5 = \frac{\delta E}{\delta \hat{y}} = 2 * \frac{1}{2}(y-\hat{y})^{2-1} = -(y-\hat{y})$
$\therefore \delta_5 = \hat{y} - y = 0.5 - 1 \implies \delta_5 = -0.5$
Now let us find how much does the Error change with respect to weight $w_d y$ so that we can update the weight.
$\frac{\delta E}{\delta w_{dy}} = \delta_5 * \frac{\delta \hat{y}}{\delta w_{dy}} = \delta_5 * d_{out} = -0.5 * 0.5 = -(0.5)^2$ or -0.25

Node d:

$$\delta_4 = \frac{\delta E}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta d_{out}} \frac{\delta d_{out}}{\delta d_{in}} \implies \frac{\delta E}{\delta w_{cd}} = \delta_4 \frac{\delta d_{in}}{\delta w_{cd}} = \delta_4 * c_{out}$$

$\therefore \delta_4 = (\hat{y} - y)(w_{dy})(d_{out})(1 - d_{out})$
$\implies \delta_4 = (-0.5)(1)(0.5)(0.5)$
$\implies \delta_4 = -(0.5)^3$
$\implies \dfrac{\delta E}{\delta w_{cd}} = \delta_4 * c_{out} = -(0.5)^3 * (0.5) = -(0.5)^4$

Node c:

$$\delta_3 = \frac{\delta E}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta d_{out}} \frac{\delta d_{out}}{\delta d_{in}} \frac{\delta d_{in}}{\delta c_{out}} \frac{\delta c_{out}}{\delta c_{in}} \implies \frac{\delta E}{\delta w_{bc}} = \delta_3 * \frac{\delta c_{in}}{\delta w_{bc}} = \delta_3 * b_{out}$$

$\therefore \delta_3 = (\hat{y} - y)(w_{dy})(d_{out})(1 - d_{out})(w_{cd})(c_{out})(1 - c_{out})$
$\implies \delta_3 = (-0.5)(1)(0.5)(0.5)(1)(0.5)(0.5)$
$\implies \delta_3 = -(0.5)^5$
$\implies \dfrac{\delta E}{\delta w_{bc}} = \delta_3 * b_{out} = -(0.5)^5 * (0.5) = -(0.5)^6$

Node b:

$$\delta_2 = \frac{\delta E}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta d_{out}} \frac{\delta d_{out}}{\delta d_{in}} \frac{\delta d_{in}}{\delta c_{out}} \frac{\delta c_{out}}{\delta c_{in}} \frac{\delta c_{in}}{\delta b_{out}} \frac{\delta b_{out}}{\delta b_{in}} \implies \frac{\delta E}{\delta w_{ab}} = \delta_2 * \frac{\delta b_{in}}{\delta w_{ab}} = \delta_2 * a_{out}$$

$\therefore \delta_2 = (\hat{y} - y)(w_{dy})(d_{out})(1 - d_{out})(w_{cd})(c_{out})(1 - c_{out})(w_{bc})(b_{out})(1 - b_{out})$
$\implies \delta_2 = (-0.5)(1)(0.5)(0.5)(1)(0.5)(0.5)(1)(0.5)(1 - 0.5)$
$\implies \delta_2 = -(0.5)^7$
$\implies \dfrac{\delta E}{\delta w_{ab}} = \delta_2 * a_{out} = -(0.5)^7 * (0.5) = -(0.5)^8$

Node a:

$$\delta_1 = \frac{\delta E}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta d_{out}} \frac{\delta d_{out}}{\delta d_{in}} \frac{\delta d_{in}}{\delta c_{out}} \frac{\delta c_{out}}{\delta c_{in}} \frac{\delta c_{in}}{\delta b_{out}} \frac{\delta b_{out}}{\delta b_{in}} \frac{\delta b_{in}}{\delta a_{out}} \frac{\delta a_{out}}{\delta a_{in}} \implies \frac{\delta E}{\delta w_{xa}} = \delta_1 * \frac{\delta a_{in}}{\delta w_{xa}} = \delta_1 * x$$

$\therefore \delta_1 = (\hat{y} - y)(w_{dy})(d_{out})(1 - d_{out})(w_{cd})(c_{out})(1 - c_{out})(w_{bc})(b_{out})(1 - b_{out})(w_{ab})(a_{out})(1 - a_{out})$
$\implies \delta_1 = (-0.5)(1)(0.5)(0.5)(1)(0.5)(0.5)(1)(0.5)(1 - 0.5)(1)(0.5)(0.5)$
$\implies \delta_1 = -(0.5)^9$
$\implies \dfrac{\delta E}{\delta w_{xa}} = \delta_1 x = -(0.5)^9 * (0.5) = -(0.5)^{10}$

Solutions:

(a) Give an expression that compares the magnitudes of the gradient updates for weights(δ)across the consecutive nodes:

From the above calculation, δ at each node are identified.

The magnitudes of the gradient updates for weights(δ)across the consecutive nodes can be compared by finding the ratio of the δ's as follows:

$\delta_1 = -(0.5)^9; \delta_2 = -(0.5)^7; \delta_3 = -(0.5)^5; \delta_4 = -(0.5)^3; \delta_5 = -0.5$

$$\implies \delta_1 : \delta_2 = \frac{-(0.5)^9}{-(0.5)^7} = \frac{(\frac{1}{2})^2}{1} = \frac{\frac{1}{4}}{1} = \frac{1}{4} = 1 : 4$$

$$\implies \delta_2 : \delta_3 = \frac{-(0.5)^7}{-(0.5)^5} = \frac{(\frac{1}{2})^2}{1} = \frac{\frac{1}{4}}{1} = \frac{1}{4} = 1 : 4 \equiv 4 : 16$$

$$\implies \delta_3 : \delta_4 = \frac{-(0.5)^5}{-(0.5)^3} = \frac{(\frac{1}{2})^2}{1} = \frac{\frac{1}{4}}{1} = \frac{1}{4} = 1 : 4 \equiv 16 : 64$$

$$\implies \delta_4 : \delta_5 = \frac{-(0.5)^3}{-(0.5)} = \frac{(\frac{1}{2})^2}{1} = \frac{\frac{1}{4}}{1} = \frac{1}{4} = 1 : 4 \equiv 64 : 256$$

$$\implies \delta_1 : \delta_2 : \delta_3 : \delta_4 : \delta_5 = 1 : 4 : 16 : 64 : 256$$

The magnitudes of the gradient updates for weights(δ)across the consecutive nodes are in the ratio of $1 : 4 : 16 : 64 : 256$ respectively.

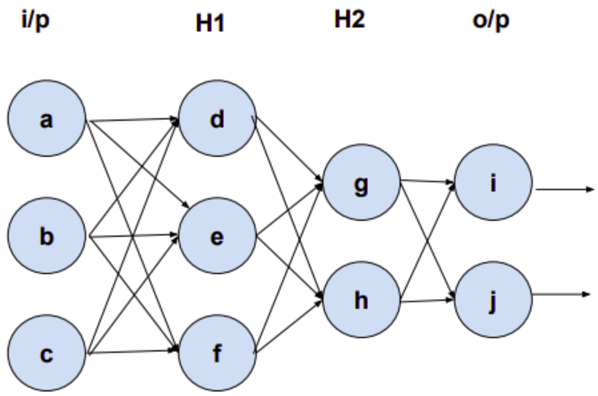
(b) How does the magnitude of the gradient update vary across the network/chain as we move away from the output unit?

Let us consider the ratio we obtained in the previous step,
$\implies$ Ratio when observed from Input to output $= 1 : 4 : 16 : 64 : 256$
We are asked to observe the change in a manner of away from the output unit
$\implies$ Ratio when observed from Output to Input $= 256 : 64 : 16 : 4 : 1$

If we carefully observe the numbers in the ratio, we can interpret that each number is divided by 4 to obtain the next number.
$\implies$ The numbers form a **Geometric series** with a=256 and common ratio, r $=\frac{1}{4}$

$\therefore$ At each step moving away from the output unit, the magnitude of the gradient update gets divided by 4 at each further step.

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) If sigmoid is used in both H1 and H2:

For input, V_1=[0.2,1,3]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 0.2*0.5+1*1.3+3*0=1.4
Ouput at Node e= sigmoid(1.4)=$\dfrac{1}{1+e^{1.4}}$ = 0.8022
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 0.2*1.3+1*(-1.4)+3*0.3=-0.32
Ouput at Node e= sigmoid(-0.32)=$\dfrac{1}{1+e^{-0.32}}$ = 0.4207
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 0.2*(-2)+1*0.75+3*(-1.75)=-3.4
Ouput at Node f= sigmoid(-3.4)=$\dfrac{1}{1+e^{-3.4}}$ = 0.0323

H2
Input to node g= $\mathrm{d_{output}}$*w_{dg} +$\mathrm{e_{output}}$*w_{eg}+$\mathrm{f_{output}}$*w_{fg}= 0.8022*1+0.4207*2.5+0.0323*3
$\Longrightarrow$ g_{input}=1.9507
Ouput at Node g= sigmoid(1.9507)=$\dfrac{1}{1+e^{1.9507}}$ = 0.8755
Input to node h=$\mathrm{d_{output}}$*w_{dh} +$\mathrm{e_{output}}$*w_{eh}+$\mathrm{f_{output}}$*w_{fh}= 0.8022*3+0.4207*(-2)+0.0323*1.25
$\Longrightarrow$ h_{input}=1.6056
Ouput at Node h= sigmoid(1.6056)=$\dfrac{1}{1+e^{1.6056}}$ = 0.8328

Output

Ouput at node i= $\mathrm{g_{output}}$*w_{gi} +$\mathrm{h_{output}}$*w_{hi}= 0.8755*2.5+0.8328*0.5=2.6052
Output at node j= $\mathrm{g_{output}}$*w_{gj} +$\mathrm{h_{output}}$*w_{hj}= 0.8755*(-1.5)+0.8328*(-0.25)=-1.5215
Output of the network for the input V1= [2.6052,-1.5215]

For input, V_2=[2.5,3,7]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 2.5*0.5+3*1.3+7*0=5.15
Ouput at Node d= sigmoid(5.15)=$\dfrac{1}{1+e^{5.15}}$ = 0.9942
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 2.5*1.3+3*(-1.4)+7*0.3=0.15
Ouput at Node e= sigmoid(0.15)=$\dfrac{1}{1+e^{0.15}}$ = 0.5374
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 2.5*(-2)+3*0.75+7*(-1.75)=-11.5
Ouput at Node f= sigmoid(-11.5)=$\dfrac{1}{1+e^{-11.5}}$ = 0.00001

H2
Input to node g= $\mathrm{d_{output}}$*w_{dg} +$\mathrm{e_{output}}$*w_{eg}+$\mathrm{f_{output}}$*w_{fg}= 0.9942*1+0.5374*2.5+0.00001*3
$\Longrightarrow$ g_{input}=2.3378

Ouput at Node g= sigmoid(2.3378)=$\dfrac{1}{1+e^{2.3378}}$ = 0.9119
Input to node h=$\mathrm{d_{output}}$*w_{dh} +$\mathrm{e_{output}}$*w_{eh}+$\mathrm{f_{output}}$*w_{fh}= 0.9942*3+0.5374*(-2)+0.00001*1.25
$\Longrightarrow$ h_{input}=1.9078
Ouput at Node h= sigmoid(1.9078)=$\dfrac{1}{1+e^{1.9078}}$ = 0.8707

Output

Ouput at node i= $\mathrm{g_{output}}$*w_{gi} +$\mathrm{h_{output}}$*w_{hi}= 0.9119*2.5+0.8707*0.5=2.7152
Output at node j= $\mathrm{g_{output}}$*w_{gj} +$\mathrm{h_{output}}$*w_{hj}= 0.9119*(-1.5)+0.8707*(-0.25)=-1.5856
Output of the network for the input V2= [2.7152,-1.5856]

For input, V_3=[0.75,-2,3]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 0.75*0.5+(-2)*1.3+3*0=-0.925
Ouput at Node d= sigmoid(-0.925)=$\dfrac{1}{1+e^{-0.925}}$ = 0.2839
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 0.75*1.3+(-2)*(-1.4)+3*0.3=2.975
Ouput at Node e= sigmoid(2.975)=$\dfrac{1}{1+e^{2.975}}$ = 0.9514
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 0.75*(-2)+(-2)*0.75+3*(-1.75)=-6
Ouput at Node f= sigmoid(-6)=$\dfrac{1}{1+e^{-6}}$ = 0.00247

H2
Input to node g= $\mathrm{d_{output}}$*w_{dg} +$\mathrm{e_{output}}$*w_{eg}+$\mathrm{f_{output}}$*w_{fg}= 0.2839*1+0.9514*2.5+ 0.00247*3
$\Longrightarrow$ g_{input}=2.6699
Ouput at Node g= sigmoid(2.6699)=$\dfrac{1}{1+e^{2.6699}}$ = 0.9352
Input to node h=$\mathrm{d_{output}}$*w_{dh} +$\mathrm{e_{output}}$*w_{eh}+$\mathrm{f_{output}}$*w_{fh}= 0.2839*3+0.9514*(-2)+ 0.00247*1.25
$\Longrightarrow$ h_{input}=-1.0479
Ouput at Node h= sigmoid(-1.0479)=$\dfrac{1}{1+e^{-1.0479}}$ = 0.2596

Output

Ouput at node i= $\mathrm{g_{output}}$*w_{gi} +$\mathrm{h_{output}}$*w_{hi}= 0.9352*2.5+0.2596*0.5=2.4678
Output at node j= $\mathrm{g_{output}}$*w_{gj} +$\mathrm{h_{output}}$*w_{hj}= 0.9352*(-1.5)+0.2596*(-0.25)=-1.4677
Output of the network for the input V3= [2.4678,-1.4677]

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

For input, V_1=[0.2,1,3]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 0.2*0.5+1*1.3+3*0=1.4
Ouput at Node e= tanh(1.4)=0.8853
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 0.2*1.3+1*(-1.4)+3*0.3=-0.32
Ouput at Node e= tanh(-0.32)=-0.3095
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 0.2*(-2)+1*0.75+3*(-1.75)=-3.4
Ouput at Node f= tanh(-3.4)=-0.9977

H2
Input to node g= $\mathrm{d_{output}}$*w_{dg} +$\mathrm{e_{output}}$*w_{eg}+$\mathrm{f_{output}}$*w_{fg}= 0.8853*1+(-0.3095)*2.5+(-0.9977)*3
$\implies$ g_{input}=-2.8817
Ouput at Node g= tanh(-2.8817)=-0.9937
Input to node h=$\mathrm{d_{output}}$*w_{dh} +$\mathrm{e_{output}}$*w_{eh}+$\mathrm{f_{output}}$*w_{fh}= 0.8853*3+(-0.3095)*(-2)+(-0.9977)*1.25
$\implies$ h_{input}=2.0278
Ouput at Node h= tanh(2.0278)=0.9659

Output

Ouput at node i= $\mathrm{g_{output}}$*w_{gi} +$\mathrm{h_{output}}$*w_{hi}= -0.9937*2.5+0.9659*0.5=-2.0013
Output at node j= $\mathrm{g_{output}}$*w_{gj} +$\mathrm{h_{output}}$*w_{hj}= -0.9937*(-1.5)+0.9659*(-0.25)=1.2491
Output of the network for the input V1= [-2.0013,1.2491]

For input, V_2=[2.5,3,7]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 2.5*0.5+3*1.3+7*0=5.15
Ouput at Node d= tanh(5.15)=0.9999
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 2.5*1.3+3*(-1.4)+7*0.3=0.15
Ouput at Node e= tanh(0.15)=0.1488
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 2.5*(-2)+3*0.75+7*(-1.75)=-11.5
Ouput at Node f= tanh(-11.5)=-1

H2
Input to node g= $\mathrm{d_{output}}$*w_{dg} +$\mathrm{e_{output}}$*w_{eg}+$\mathrm{f_{output}}$*w_{fg}= 0.9999*1+0.1488*2.5+-1*3
$\implies$ g_{input}=-1.6278
Ouput at Node g= tanh(-1.6278)=-0.9257
Input to node h=$\mathrm{d_{output}}$*w_{dh} +$\mathrm{e_{output}}$*w_{eh}+$\mathrm{f_{output}}$*w_{fh}= 0.9999*3+0.1488*(-2)+(-1)*1.25
$\implies$ h_{input}=1.4520
Ouput at Node h= tanh(1.4520)=0.8960

Output

Ouput at node i= $\mathrm{g_{output}}$*w_{gi} +$\mathrm{h_{output}}$*w_{hi}= -0.9257*2.5+0.8960*0.5=-1.8663
Output at node j= $\mathrm{g_{output}}$*w_{gj} +$\mathrm{h_{output}}$*w_{hj}= -0.9257*(-1.5)+0.8960*(-0.25)=1.1646

Output of the network for the input V2= [-1.8663,1.1646]

For input, V_3=[0.75,-2,3]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 0.75*0.5+(-2)*1.3+3*0=-0.925
Ouput at Node d= tanh(-0.925)=-0.7282
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 0.75*1.3+(-2)*(-1.4)+3*0.3=2.975
Ouput at Node e= tanh(2.975)=0.9948
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 0.75*(-2)+(-2)*0.75+3*(-1.75)=-6
Ouput at Node f= tanh(-6)=-0.9999

H2
Input to node g= d_{output}*w_{dg} +e_{output}*w_{eg}+f_{output}*w_{fg}= -0.7282*1+0.9948*2.5+ -0.9999*3
$\implies g_{input}$=-1.2412
Ouput at Node g= tanh(-1.2412)=-0.8458
Input to node h=d_{output}*w_{dh} +e_{output}*w_{eh}+f_{output}*w_{fh}= -0.7282*3+0.9948*(-2)+ -0.9999*1.25
$\implies h_{input}$=-5.4243
Ouput at Node h= sigmoid(-5.4243)=-0.9999

Output

Ouput at node i= g_{output}*w_{gi} +h_{output}*w_{hi}= -0.8458*2.5+(-0.9999)*0.5=-2.6144
Output at node j= g_{output}*w_{gj} +h_{output}*w_{hj}= -0.8458*(-1.5)+(-0.9999)*(-0.25)=1.5186
Output of the network for the input V3= [-2.6144,1.5186]

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

For input, V_1=[0.2,1,3]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 0.2*0.5+1*1.3+3*0=1.4
Ouput at Node e= sigmoid(1.4)=$\dfrac{1}{1+e^{1.4}} = 0.8022$
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 0.2*1.3+1*(-1.4)+3*0.3=-0.32
Ouput at Node e= sigmoid(-0.32)=$\dfrac{1}{1+e^{-0.32}} = 0.4207$
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 0.2*(-2)+1*0.75+3*(-1.75)=-3.4
Ouput at Node f= sigmoid(-3.4)=$\dfrac{1}{1+e^{-3.4}} = 0.0323$

H2
Input to node g= d_{output}*w_{dg} +e_{output}*w_{eg}+f_{output}*w_{fg}= 0.8022*1+0.4207*2.5+0.0323*3
$\implies g_{input}$=1.9507

Ouput at Node g= tanh(1.9507)=0.9603
Input to node h=$d_{\text{output}}*w_{dh}$ +$e_{\text{output}}*w_{eh}$+$f_{\text{output}}*w_{fh}$= 0.8022*3+0.4207*(-2)+0.0323*1.25
$\implies h_{input}$=1.6056
Ouput at Node h= tanh(1.6056)=0.9225

Output

Ouput at node i= $g_{\text{output}}*w_{gi}$ +$h_{\text{output}}*w_{hi}$= 0.9603*2.5+0.9225*0.5= 2.8621
Output at node j= $g_{\text{output}}*w_{gj}$ +$h_{\text{output}}*w_{hj}$= 0.9603*(-1.5)+0.9225*(-0.25)=-1.6711
Output of the network for the input V1= [2.8621,-1.6711]

For input, V_2=[2.5,3,7]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 2.5*0.5+3*1.3+7*0=5.15
Ouput at Node d= sigmoid(5.15)=$\dfrac{1}{1+e^{5.15}} = 0.9942$
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 2.5*1.3+3*(-1.4)+7*0.3=0.15
Ouput at Node e= sigmoid(0.15)=$\dfrac{1}{1+e^{0.15}} = 0.5374$
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 2.5*(-2)+3*0.75+7*(-1.75)=-11.5
Ouput at Node f= sigmoid(-11.5)=$\dfrac{1}{1+e^{-11.5}} = 0.00001$

H2
Input to node g= $d_{\text{output}}*w_{dg}$ +$e_{\text{output}}*w_{eg}$+$f_{\text{output}}*w_{fg}$= 0.9942*1+0.5374*2.5+0.00001*3
$\implies g_{input}$=2.3378
Ouput at Node g= tanh(2.3378)=0.9815
Input to node h=$d_{\text{output}}*w_{dh}$ +$e_{\text{output}}*w_{eh}$+$f_{\text{output}}*w_{fh}$= 0.9942*3+0.5374*(-2)+0.00001*1.25
$\implies h_{input}$=1.9078
Ouput at Node h= tanh(1.9078)=$\dfrac{1}{1+e^{1.9078}} = 0.8707$

Output

Ouput at node i= $g_{\text{output}}*w_{gi}$ +$h_{\text{output}}*w_{hi}$= 0.9815*2.5+0.8707*0.5=2.7152
Output at node j= $g_{\text{output}}*w_{gj}$ +$h_{\text{output}}*w_{hj}$= 0.9815*(-1.5)+0.8707*(-0.25)=-1.5856
Output of the network for the input V2= [2.7152,-1.5856]

For input, V_3=[0.75,-2,3]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 0.75*0.5+(-2)*1.3+3*0=-0.925
Ouput at Node d= sigmoid(-0.925)=$\dfrac{1}{1+e^{-0.925}} = 0.2839$
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 0.75*1.3+(-2)*(-1.4)+3*0.3=2.975

Ouput at Node e= sigmoid(2.975)=$\dfrac{1}{1+e^{2.975}}$ = 0.9514
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 0.75*(-2)+(-2)*0.75+3*(-1.75)=-6
Ouput at Node f= sigmoid(-6)=$\dfrac{1}{1+e^{-6}}$ = 0.00247

H2
Input to node g= $\mathrm{d_{output}}$*w_{dg} +$\mathrm{e_{output}}$*w_{eg}+$\mathrm{f_{output}}$*w_{fg}= 0.2839*1+0.9514*2.5+ 0.00247*3
$\implies$ g_{input}=2.6699
Ouput at Node g= tanh(2.6699)=0.9904
Input to node h=$\mathrm{d_{output}}$*w_{dh} +$\mathrm{e_{output}}$*w_{eh}+$\mathrm{f_{output}}$*w_{fh}= 0.2839*3+0.9514*(-2)+ 0.00247*1.25
$\implies$ h_{input}=-1.0479
Ouput at Node h= tanh(-1.0479)=-0.7810

Output

Ouput at node i= $\mathrm{g_{output}}$*w_{gi} +$\mathrm{h_{output}}$*w_{hi}= 0.9904*2.5+-0.7810*0.5=2.0856
Output at node j= $\mathrm{g_{output}}$*w_{gj} +$\mathrm{h_{output}}$*w_{hj}= 0.9904*(-1.5)+-0.7810*(-0.25)=-1.2904
Output of the network for the input V3= [2.0856,-1.2904]

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

For input, V_1=[0.2,1,3]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 0.2*0.5+1*1.3+3*0=1.4
Ouput at Node e= tanh(1.4)=0.8853
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 0.2*1.3+1*(-1.4)+3*0.3=-0.32
Ouput at Node e= tanh(-0.32)=-0.3095
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 0.2*(-2)+1*0.75+3*(-1.75)=-3.4
Ouput at Node f= tanh(-3.4)=-0.9977

H2
Input to node g= $\mathrm{d_{output}}$*w_{dg} +$\mathrm{e_{output}}$*w_{eg}+$\mathrm{f_{output}}$*w_{fg}= 0.8853*1+(-0.3095)*2.5+(-0.9977)*3
$\implies$ g_{input}=-2.8817
Ouput at Node g= ReLU(-2.8817)=0
Input to node h=$\mathrm{d_{output}}$*w_{dh} +$\mathrm{e_{output}}$*w_{eh}+$\mathrm{f_{output}}$*w_{fh}= 0.8853*3+(-0.3095)*(-2)+(-0.9977)*1.25
$\implies$ h_{input}=2.0278
Ouput at Node h= ReLU(2.0278)=0.9659

Output

Ouput at node i= $\mathrm{g_{output}}$*w_{gi} +$\mathrm{h_{output}}$*w_{hi}= 0*2.5+2.0278*0.5=1.0139

Output at node j= $g_{\text{output}}*w_{gj}$ +$h_{\text{output}}*w_{hj}$= 0*(-1.5)+2.0278*(-0.25)=-0.5069
Output of the network for the input V1= [1.0139,-0.5069]

For input, V_2=[2.5,3,7]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 2.5*0.5+3*1.3+7*0=5.15
Ouput at Node d= tanh(5.15)=0.9999
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 2.5*1.3+3*(-1.4)+7*0.3=0.15
Ouput at Node e= tanh(0.15)=0.1488
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 2.5*(-2)+3*0.75+7*(-1.75)=-11.5
Ouput at Node f= tanh(-11.5)=-1

H2
Input to node g= $d_{\text{output}}*w_{dg}$ +$e_{\text{output}}*w_{eg}$+$f_{\text{output}}*w_{fg}$= 0.9999*1+0.1488*2.5+-1*3
$\implies g_{input}$=-1.6278
Ouput at Node g= relu(-1.6278)=0
Input to node h=$d_{\text{output}}*w_{dh}$ +$e_{\text{output}}*w_{eh}$+$f_{\text{output}}*w_{fh}$= 0.9999*3+0.1488*(-2)+(-1)*1.25
$\implies h_{input}$=1.4520
Ouput at Node h= ReLU(1.4520)=1.4520

Output

Ouput at node i= $g_{\text{output}}*w_{gi}$ +$h_{\text{output}}*w_{hi}$= 0*2.5+1.4520*0.5=0.7260
Output at node j= $g_{\text{output}}*w_{gj}$ +$h_{\text{output}}*w_{hj}$= 0*(-1.5)+1.4520*(-0.25)=-0.3630
Output of the network for the input V2= [0.7260,-0.3630]

For input, V_3=[0.75,-2,3]

H1
Input to node d= a*w_{ad} +b*w_{bd}+c*w_{cd}= 0.75*0.5+(-2)*1.3+3*0=-0.925
Ouput at Node d= tanh(-0.925)=-0.7282
Input to node e= a*w_{ae} +b*w_{be}+c*w_{ce}= 0.75*1.3+(-2)*(-1.4)+3*0.3=2.975
Ouput at Node e= tanh(2.975)=0.9948
Input to node f= a*w_{af} +b*w_{bf}+c*w_{cf}= 0.75*(-2)+(-2)*0.75+3*(-1.75)=-6
Ouput at Node f= tanh(-6)=-0.9999

H2
Input to node g= $d_{\text{output}}*w_{dg}$ +$e_{\text{output}}*w_{eg}$+$f_{\text{output}}*w_{fg}$= -0.7282*1+0.9948*2.5+ -0.9999*3
$\implies g_{input}$=-1.2412
Ouput at Node g= ReLU(-1.2412)=0
Input to node h=$d_{\text{output}}*w_{dh}$ +$e_{\text{output}}*w_{eh}$+$f_{\text{output}}*w_{fh}$= -0.7282*3+0.9948*(-2)+ -0.9999*1.25
$\implies h_{input}$=-5.4243
Ouput at Node h= ReLU(-5.4243)=0

Output

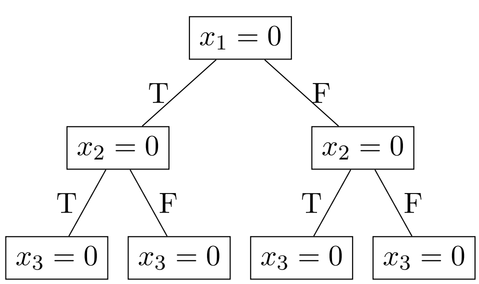
Ouput at node i= $\mathrm{g}_{\mathrm{output}} * w_{gi}$ $+\mathrm{h}_{\mathrm{output}} * w_{hi}$= 0*2.5+0*0.5=0
Output at node j= $\mathrm{g}_{\mathrm{output}} * w_{gj}$ $+\mathrm{h}_{\mathrm{output}} * w_{hj}$= 0*(-1.5)+0*(-0.25)=0
Output of the network for the input V3= [0,0]

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
$y = sign\Big(\sum_{i=1}^{m}\big(2x_i - 1\big)\Big)$
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: $y = sign(\sum_{i=1}^{m})(2x_i - 1) = \begin{cases} 1, & \text{if } \sum_{i=1}^{m}(x_i) > \frac{m}{2} \\ -1, & \text{otherwise} \end{cases}$

From the formula, we can infer that output is 1 if there are more 1s than 0s in the input m values. Also, output is -1 otherwise.

The decision tree is constructed as follows, at each node we are checking if the x_i value is 0 and if it is true, then the left path is taken from that node and if it is false (i.e $x_i = 1$) then the right path is taken from that node.



and it splits upto x_m.

Hence maximum number of leaves for a decision tree fitting this will be for a complete binary tree of depth m. Hence max number of leaves is 2^m.

We are given that the true classifier is a majority vote over the features and thus we need to have more number of 1's than 0's in the m-length feature vector x to get a true output. Hence the decision tree must split atleast $\frac{m}{2}$ times and hence minimum number of leaves is $2^{\frac{m}{2}}$.

Hence the decision tree has number of leaves ranging from $2^{\frac{m}{2}}$ to 2^m.