Name- Sairamvinay Vijayaraghavan
Student ID - 913603345

ECS 171: HW 2 Report

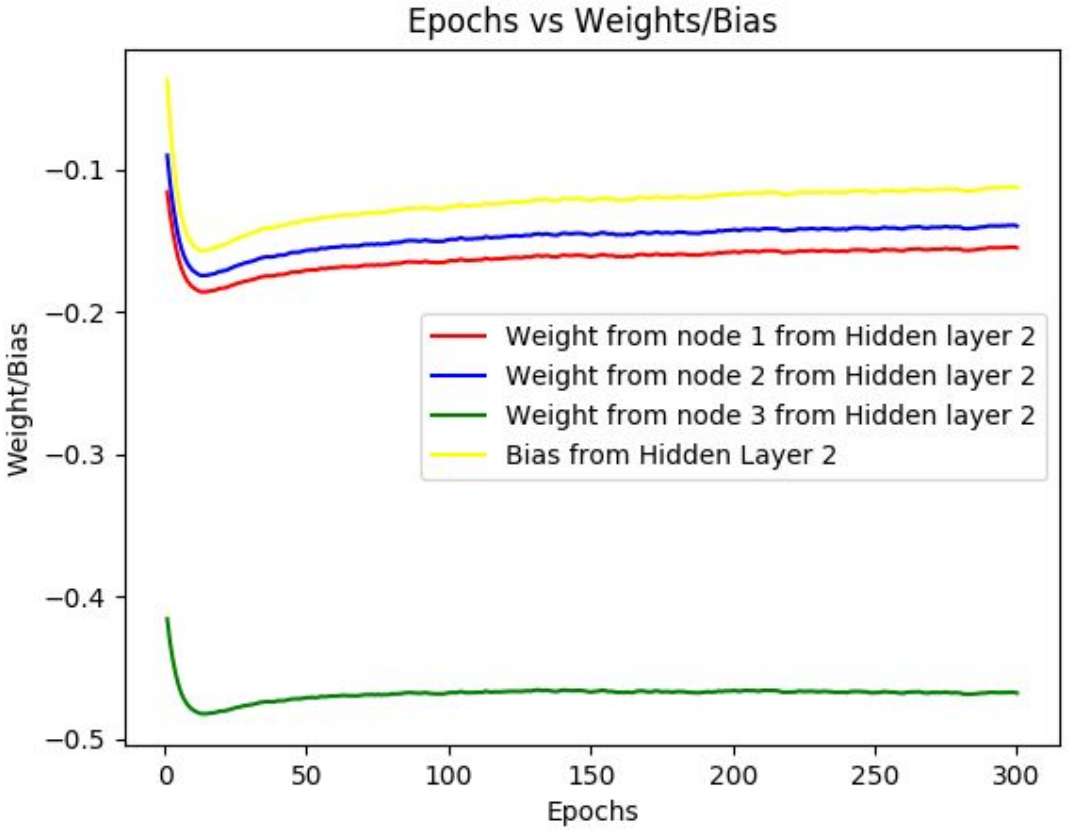
1)

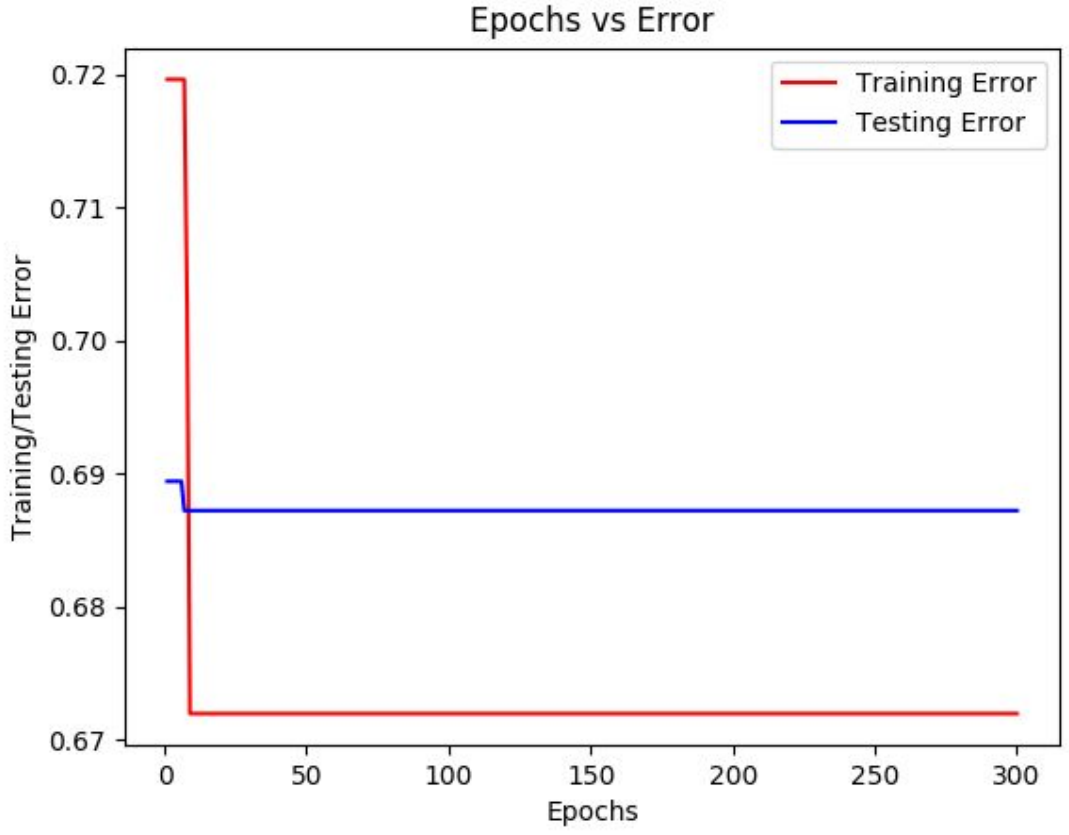
a) I had used two methods to perform outlier detection: Isolation Forest and One-Class SVM. Yes, using both these methods confirms that there are outliers in the data set being used. The outlier percentage of the number of samples in the data set for Isolation Forest is **10.040431266846362%** and for One Class SVM is **44.94609164420485**%

b) Definitely, both of these outlier detection algorithms don't seem to be in agreement with each other. This is because both of these algorithms use different approaches to detect outliers. Isolation Forest calculates scores using distances between features of each sample while OneClass SVM fits a contour plot of all sample features and detects outliers if the sample is found outside the contour.

c) This is because Isolation Forest performs outlier detection using anomaly scores amongst all samples in the dataset. Hence, this algorithm performs generally well on the data set since it compares amongst the data set itself and it doesn't try anything regarding novelty detection. However, One Class SVM performs novelty detection which makes it appear sensitive to outliers and hence it ends up over classifying outliers in the data set. Also, the parameters used for this algorithm such as the gamma, nu parameters determine the outlier detection and it classifies a large number of samples as outliers.

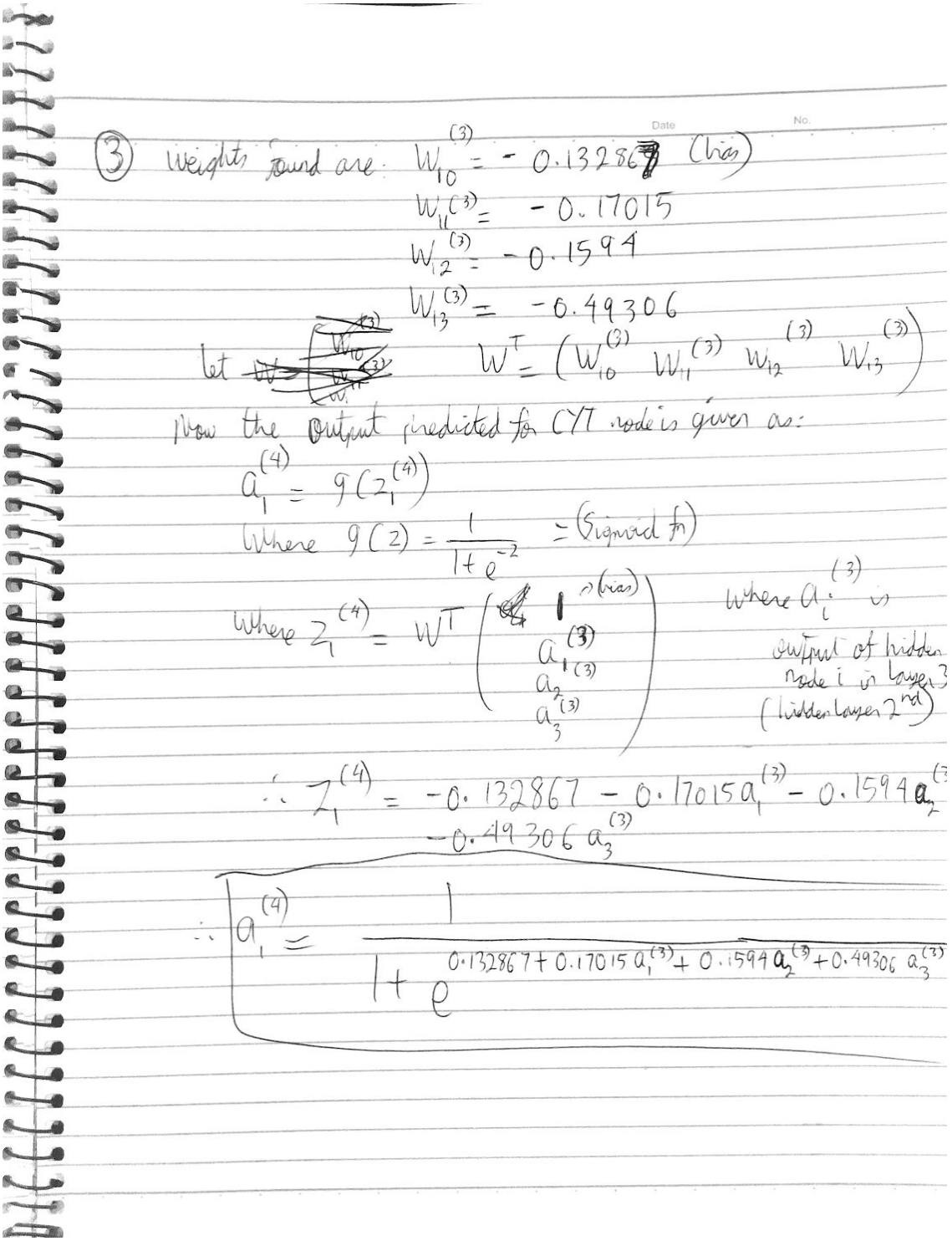
2) The outlier detection was performed and the graphs of weights and training/testing error (misclassification ratio = 1 - accuracy) are presented as follows:


Epochs vs Weights/Bias
Weight/Bias
−0.1
−0.2
−0.3
−0.4
−0.5
0
50
100
150
200
250
300
Epochs
Weight from node 1 from Hidden layer 2
Weight from node 2 from Hidden layer 2
Weight from node 3 from Hidden layer 2
Bias from Hidden Layer 2


Epochs vs Error
Training/Testing Error
0.72
0.71
0.70
0.69
0.68
0.67
0
50
100
150
200
250
300
Epochs
Training Error
Testing Error

3) I had not applied the outlier removal for this phase and the testing misclassification error ratio is **0.6880053908355794**.

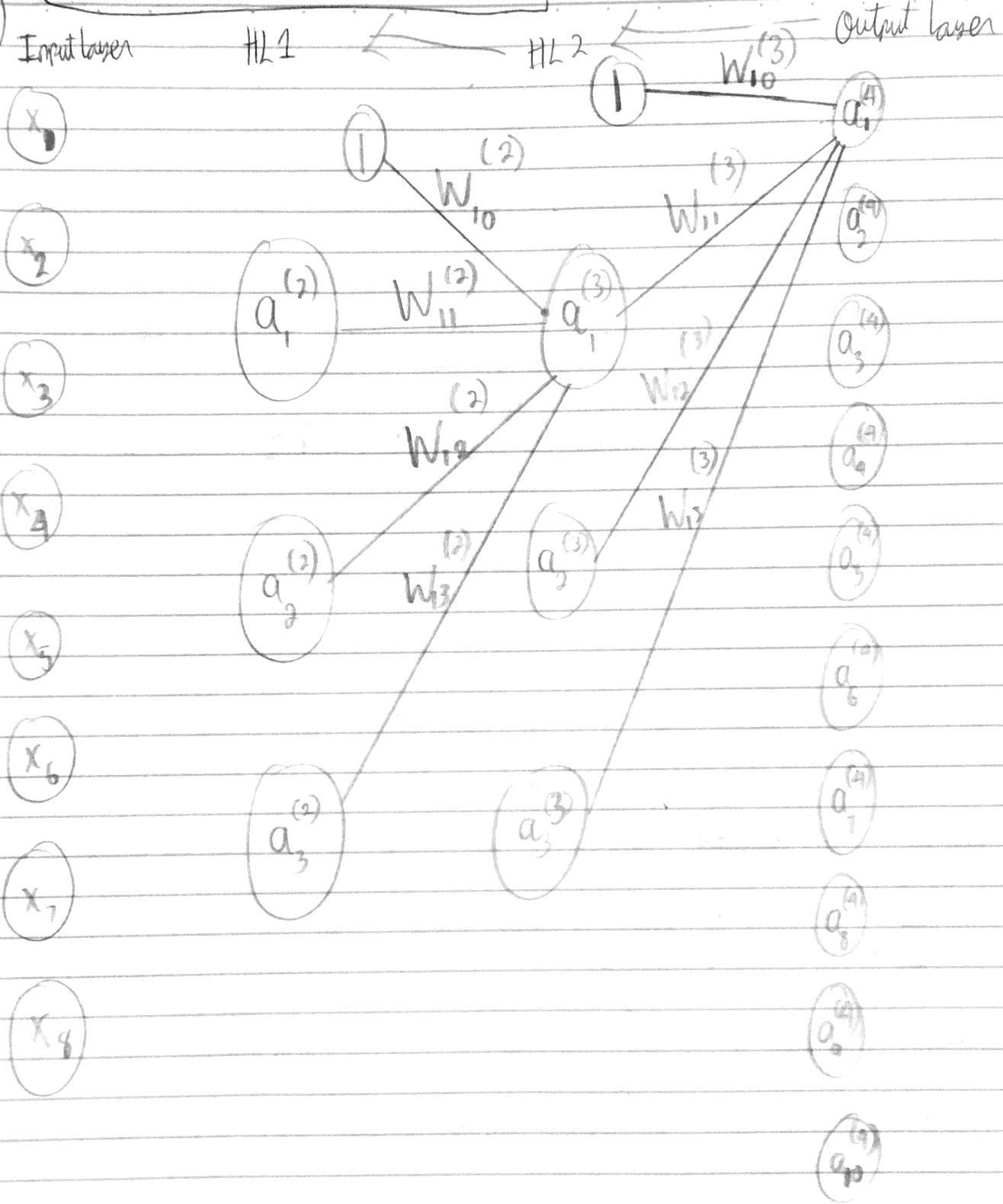
The activation formula for the final layer is as follows as shown in the picture



4) The following pictures show my calculation by hand:
OUTPUT BY HAND: Please find attached pictures on the next coming pages:

4

The Neural Network Architecture



Output layer $(L) = 4$; Using notations like $W_{kj}^{(L-1)}$ means [from node j in layer $L-1$] to node k in layer L.

Using Back propogation formula.

$$W_{kj}^{(L-1)} := W_{kj}^{(L-1)} - \eta \frac{\partial RSS}{\partial W_{kj}^{(L-1)}}$$ for output k [and only for Output layer] for $j = 0, 1, 2, 3$ (~~hidden layer~~ ~~$= 3$~~; nodes)

Let's pick 1st output node $\therefore k = 1$

Let $\eta = 0.01$; initializing $W_{1j} = 1$ for $j = 0$ (bias), 1, 2, 3 and all other weights as 0

So we need $\frac{\partial RSS}{\partial W_{kj}^{(L-1)}}$ which is given as:

$$\frac{\partial RSS}{\partial W_{kj}^{(L-1)}} = -\left(y_k - a_k^{(L)}\right) \cdot \left(1 - a_k^{(L)}\right) a_k^{(L)} \cdot a_j^{(L-1)}$$

where y_k is ~~actual~~ output for output k and $a_k^{(L)}$ is predicted output for output k.

$a_j^{(L-1)}$ is value of node j in layer $L-1$

FEED FORWARD

~~Given~~ We need to perform forward pass 1st before back propogation

Given: all weights initialized at 0 except the weights we Calculate for this problem.

I chose to update for Output layer node 1 and [second] hidden layer node 1

$\therefore j = 1$; $k = 1$ in this problem.

$\therefore$ We have $W_{10}^{(3)} = W_{11}^{(3)} = W_{12}^{(3)} = W_{13}^{(3)} = 1$ (output layer); $W_{10}^{(2)} = W_{11}^{(2)} = W_{12}^{(2)} = W_{13}^{(2)} = 1$ (second hidden layer)

Date No

Rest all weights are initialized 0. ; $sigmoid(z) = \frac{1}{1+e^{-z}}$

Input for 1st sample = [0.58, 0.61, 0.47, 0.13, 0.5, 0, 0.48, 0.22]

Actual Output for 1st sample = MNIT = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

let $X = (0.58 \;\; 0.61 \;\; 0.47 \;\; 0.13 \;\; 0.5 \;\; 0 \;\; 0.48 \;\; 0.22)^T$

Since all other weights but the ones we are using to upde all the 0 initially;

we get: $a_1^{(2)} = sigm\left(\sum_{i=1}^{8} w_{1i}^{(1)} x_i + w_{10}^{(1)}\right) = sigm(0^T x + 0)$ (bias)

$= sigm(0+0) = sigm(0) = \frac{1}{1+e^{-0}} = \frac{1}{2} = 0.5$

Similarly $a_2^{(2)} = sigm\left(\sum_{i=1}^{8} w_{2i}^{(1)} x_i + w_{20}^{(1)}\right) = sigm(0^T x + 0) = 0.5$ (bias)

$a_3^{(2)} = sigm(0) = 0.5$

Now we have $a_1^{(2)} = a_2^{(2)} = a_3^{(2)} = 0.5$

Now we obtain $a_1^{(3)}$; $a_2^{(3)}$; $a_3^{(3)}$

$a_1^{(3)} = sigm\left(\sum_{i=1}^{3} a_i^{(2)} w_{1i}^{(2)} + w_{10}^{(2)}\right)$

$= sigm((1 \times 0.5) + (1 \times 0.5) + (1 \times 0.5) + 1) = sigm(1.5+1) = sigm(2.5)$

$= 0.924$

$a_2^{(3)} = sigm\left(\sum_{i=1}^{3} a_i^{(2)} w_{2i}^{(2)} + w_{20}^{(2)}\right) = sigm(0) = 0.5$

$a_3^{(3)} = sigm\left(\sum_{i=1}^{3} a_i^{(2)} w_{3i}^{(2)} + w_{30}^{(2)}\right) = sigm(0) = 0.5$

Now finding only $a_1^{(4)}$ will suffice as output node we choose is only 1st first output node.

$$a_1^{(4)} = \text{sigm}\left(\sum_{i=1}^{3} W_{1i}^{(3)} a_i^{(3)} + W_{10}^{(3)}\right)$$

$$= \text{sigm}\left(W_{10}^{(3)} + W_{11}^{(3)} \cdot a_1^{(3)} + W_{12}^{(3)} \cdot a_2^{(3)} + W_{13}^{(3)} \cdot a_3^{(3)}\right)$$

$$= \text{sigm}\left(1 + (1 \times 0.924) + (1 \times 0.5) + (1 \times 0.5)\right)$$

$$= \text{sigm}(1 + 0.924 + 0.5 + 0.5) = \text{sigm}(2.924) = \boxed{0.9490}$$

Now we have $a_1^{(4)} = 0.949$; $y_1 = 0$ (as MIT means only 1st O/P node as 1)

BACK PROP

So using Back propagation: we find $\delta_1^{(4)} = (y_1 - a_1^{(4)}) a_1^{(4)} (1 - a_1^{(4)}) = -0.0459$

for $j = 0, 1, 2, 3$ (bias) for output layer

$$W_{1j}^{(3)} := W_{1j}^{(3)} + \eta (y_1 - a_1^{(4)}) (1 - a_1^{(4)}) \cdot a_j^{(3)}$$

$j = 0$:. $W_{10}^{(3)} := W_{10}^{(3)} + (0.01)(0 - 0.9490)(1 - 0.9490) \cdot a_0^{(3)}$ ($a_0 = 1$)

$$:= 1 + (0.01) \times (-0.949) \times (0.051) \times 1$$

$$:= 1 - (0.01 \times 0.949^2 \times 0.051) = 0.999541$$

$j = 1$; $W_{11}^{(3)} := W_{11}^{(3)} + (0.01)(0 - 0.9490)(1 - 0.9490)(a_1^{(3)} \times 0.949)$

$$:= 1 - [(0.01 \times 0.949^2 \times 0.051) \times 0.924]$$

$$:= 1 - 0.000424 = 0.999576$$

Date No.

$j=2$; $W_{12}^{(3)} := W_{12}^{(3)} + (0.01)(0-0.949)(0.051) \times 0.5 \times 0.949$

$:= 1 - (0.01 \times 0.949^2 \times 0.051 \times 0.5) = 1 - 0.00023$

$:= 0.99977$

$j=3$; $W_{13}^{(3)} := W_{13}^{(3)} - (0.01)(0.949)^2(0.051) \times 0.5 = 0.99977$

$:=$ (Same as $W_{12}^{(3)}$ as $a_3^{(3)} = a_2^{(3)} = 0.5$)

$\therefore$

$W_{10}^{(3)} = 0.999541$; $W_{11}^{(3)} = 0.999576$

$W_{12}^{(3)} = 0.99977$; $W_{13}^{(3)} = 0.99977$

2nd

For hidden layer $(L-1) = (3)$

We need to update as: (have $j=1$ fixed)

for $i = 1, 2, 3$

$$W_{1i}^{(2)} := W_{1i}^{(2)} + \eta \, \delta_1^{(3)} \, a_i^{(2)}$$

where $\delta_1^{(3)} = \sum_{k=1}^{10} \left(\delta_k^{(4)} W_{k1}^{(3)} (1 - a_1^{(3)}) a_1^{(3)} \right)$

where we use old weight values (as it is all simultaneous updating)

$$\delta_1^{(3)} = \left[\delta_1^{(4)} W_{11}^{(3)} a_1^{(3)} (1 - a_1^{(3)}) + \sum_{k=2}^{10} \delta_k^{(4)} W_{k1}^{(3)} a_1^{(3)} (1 - a_1^{(3)}) \right]$$

Now since we use old initial values & we had not used $W_{k1}^{(3)}$ for $k = 2, \ldots, 10$ $\therefore W_{k1}^{(3)} = 0$ $k = 2, 3, 4, \ldots, 10$

Date No.

$\therefore \delta_1^{(3)}$ reduces to $= \delta_1^{(4)} W_{11}^{(3)} a_1^{(3)} (1 - a_1^{(3)})$

~~$= (-0.000484) \times 1 \times 0.924 \times (1 - 0.924)$~~ (old $W_{11}^{(3)}$)

~~$= -3.399 \times 10^{-5}$~~

New: $W_{10}^{(2)} := W_{10}^{(2)} + 0.01 \times (-3.399 \times 10^{-5}) \times a_0^{(2)}$

$W_{11}^{(2)} := W_{11}^{(2)} + 0.01 \times (-3.399 \times 10^{-5}) \times a_1^{(2)}$

$W_{12}^{(2)} := W_{12}^{(2)} +$

$W_{13}^{(2)} := W_{13}^{(3)} +$

$\delta_1^{(3)} = -0.0459 \times 1$ (old weight) $\times 0.924 \times (1 - 0.924)$

$= -0.0032$

$i = 0, 1, 2, 3$

$\therefore i=0;$ $W_{10}^{(2)} := W_{10}^{(2)} + (0.01) \times (-0.0032) \times a_0^{(2)}$

$:= 1 - (0.01 \times 0.0032) \times 1 = 0.999968$

$i=1;$ $W_{11}^{(2)} := W_{11}^{(2)} + (0.01)(-0.0032) \times 0.5$

$:= 1 - (1.6 \times 10^{-5}) = 0.999984$

Similarly $W_{12}^{(2)} = W_{13}^{(2)} = 0.999984$

Since $a_1^{(2)} = a_2^{(2)} = a_3^{(2)} = 0.5$

$\therefore$ Same calculation

OUTPUT BY CODE: The following pictures show the output of the code for just the first sample and one single epoch.

	Bias	Weight from first node of previous hidden layer	Weight from second node of previous hidden layer	Weight from third node of previous hidden layer
Output Layer first node	0.9999082	0.9999151	0.9999541	0.9999541
Second Hidden Layer first node	0.99999356	0.9999968	0.9999968	0.9999968

As we can see, the results do agree but not coincide with each other. However, they are off by a magnitude of 10^(-4) for Output layer weights from second hidden layer and of magnitude 10^(-5) for 2nd Hidden Layer weights from the first hidden layer, which explains that they are relatively very close but not exactly equal to each other.

This is probably because of the slightly different update of weights within the code while it was much more naive in my calculation which focussed only on these nodes in particular.

5) Batch Size as 10 and number of epochs as 300 and learning rate = 0.001

Testing Error Ratio	#nodes = 3	#nodes = 6	#nodes = 9	#nodes = 12
#Layers = 1	0.6872246695	0.693832599	0.6872246695	0.6872246695
#Layers = 2	0.6894273126	0.682819383	0.6872246695	0.6872246695
#Layers = 3	0.6894273126	0.6894273126	0.6872246695	0.6872246695

From the table, we get the optimal configuration as **2 hidden layers and 6 hidden nodes**. The error does not seem to improve for increasing hidden layers but it seems to decrease slightly for decreasing hidden nodes in my case.

6) The class predicted by the ANN is "**CYT**"

7) Changing the loss to binary_crossentropy and the inner activations to relu and outer activations to softmax does indeed prove to be a better choice. This can be witnessed by the small value of misclassifications. Definitely, it is a better choice

This change means there is a definite improvement in prediction since we have better loss function which predicts better using binary_crossentropy than MSE which just calculates the error between prediction and target however cross entropy looks to maximize the likelihood probability.

The grid search error matrix is added as follows:

300 epochs, 1 batch size and learning rate = 0.01

Testing Error Ratio	#nodes = 3	#nodes = 6	#nodes = 9	#nodes = 12
#Layers = 1	0.0861234	0.098458	0.078634	0.079075
#Layers = 2	0.10000002	0.0955947	0.08303965	0.081938
#Layers = 3	0.0951542	0.0911895	0.08303965	0.078414108