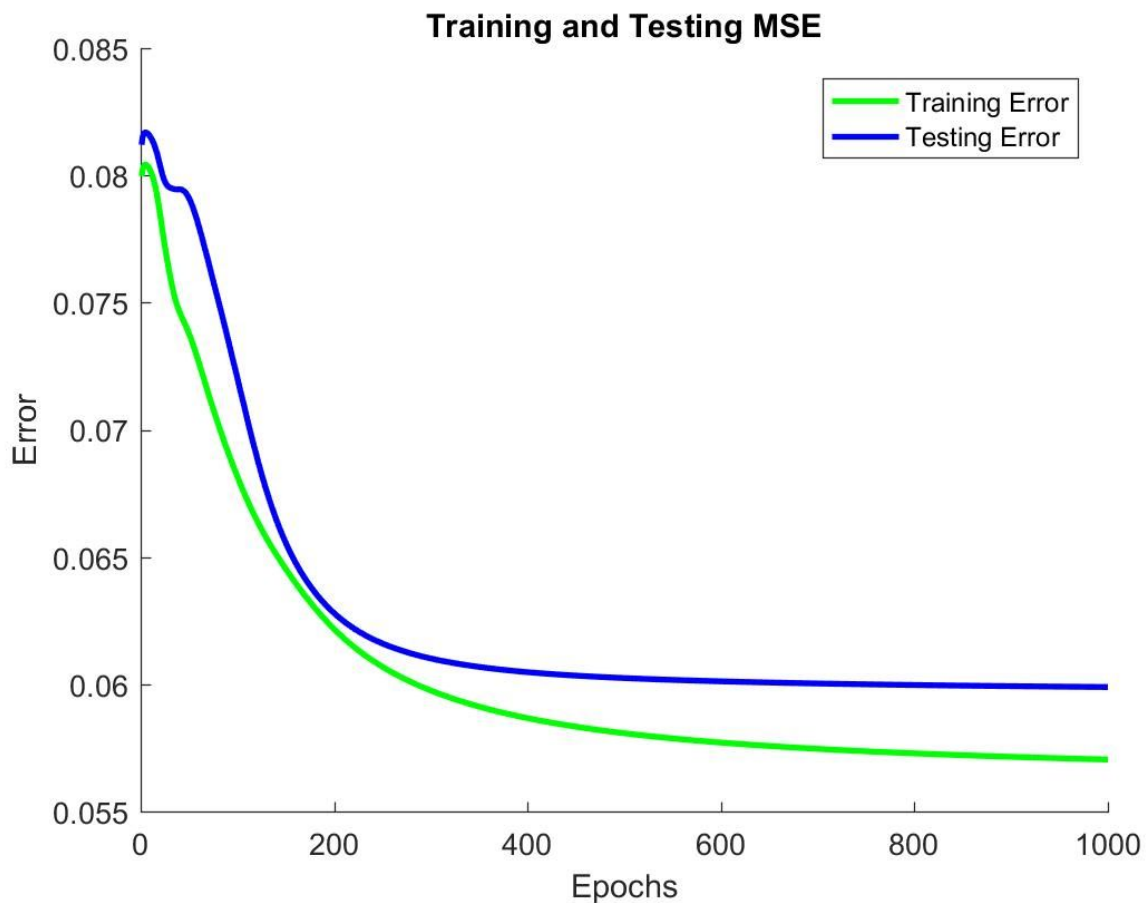
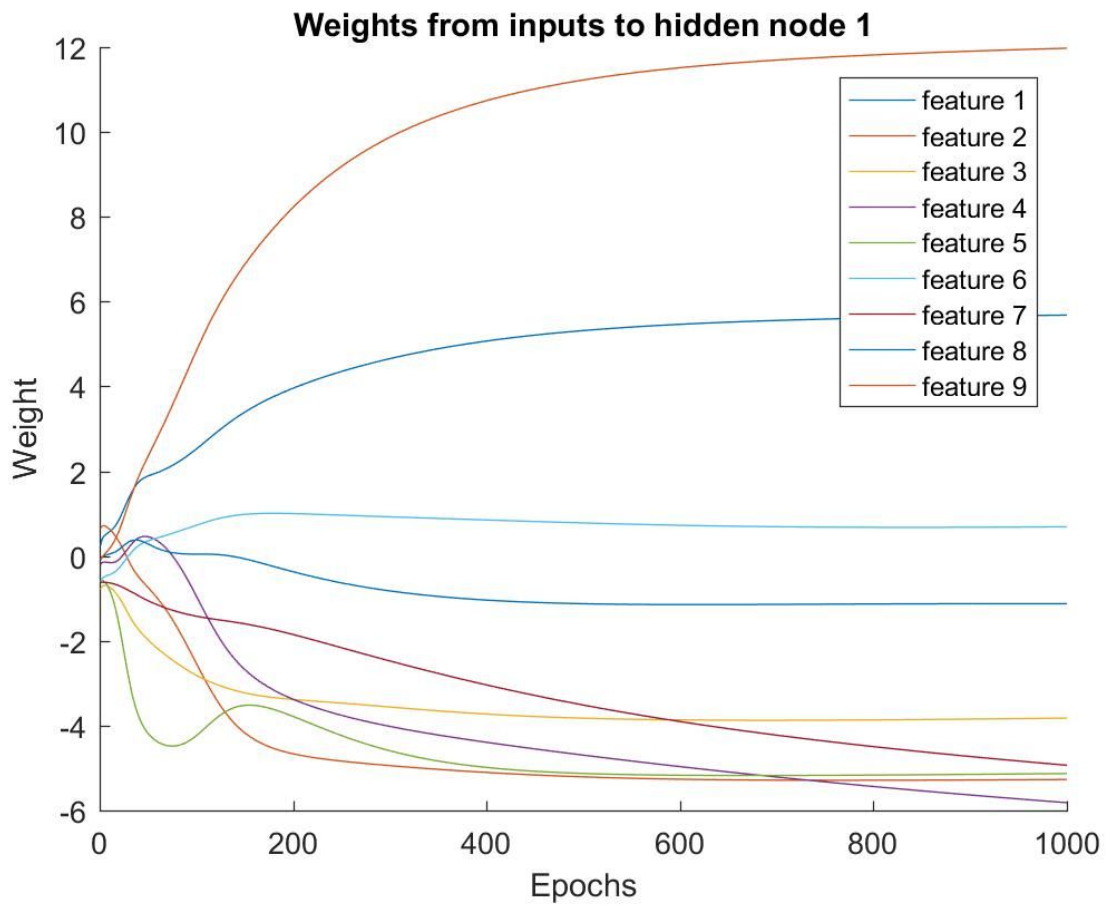


Where did the Baker go?

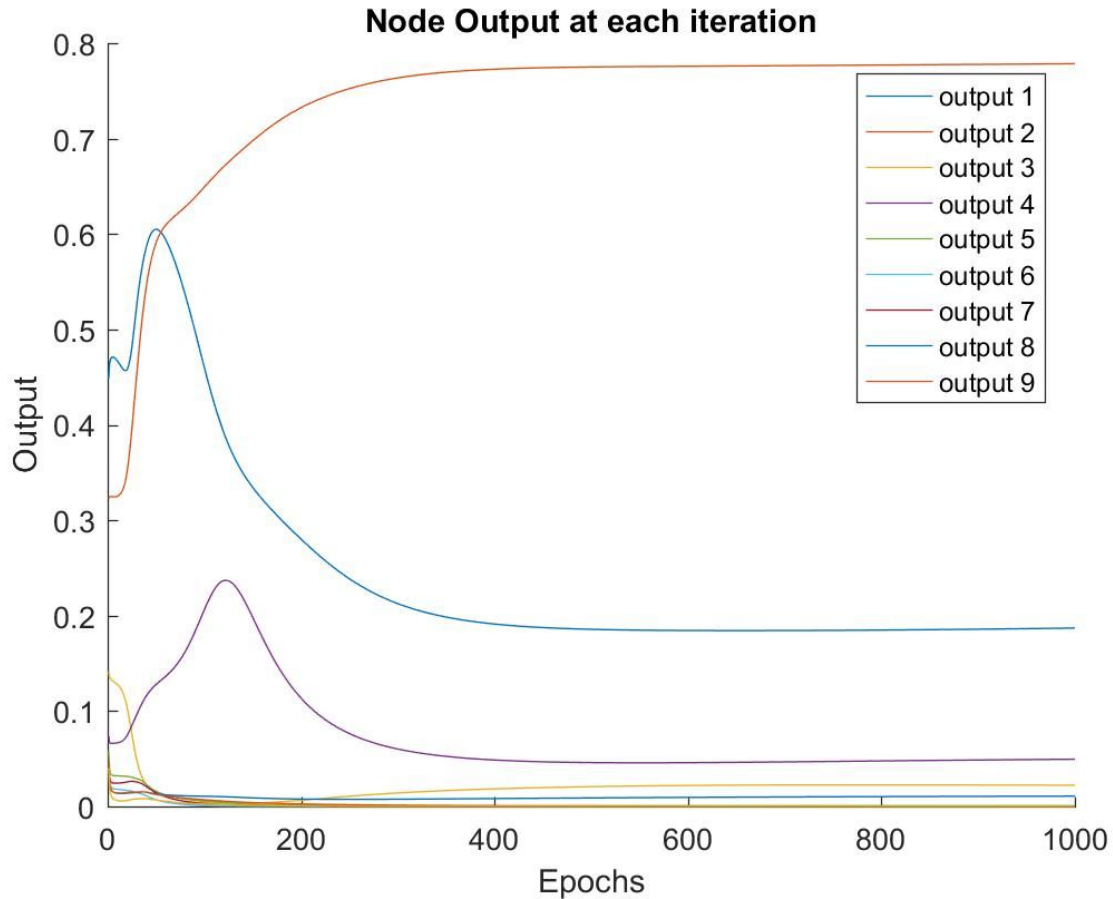
1) For this assignment I created my own neural network function that can take in nodes and layers as parameters so the same function has been used for all the assignments. The initial weights were set as random numbers between $[-1,1]$ which yielded better results, since weights get trimmed to positive and negative values. A concern with this artificial neural network is overfitting because the training error decreases over time and the testing error decreases at a slower rate. Because of this 250-300 epochs may be ideal to avoid overfitting. This is also where the training and testing accuracy level out, however the plots for this problem show 1000 epochs to highlight convergence. An alpha of 0.01 was optimal, with higher alpha the accuracy would fall to about 30-40%. The final test error is 0.0608 and final train error is 0.0567.



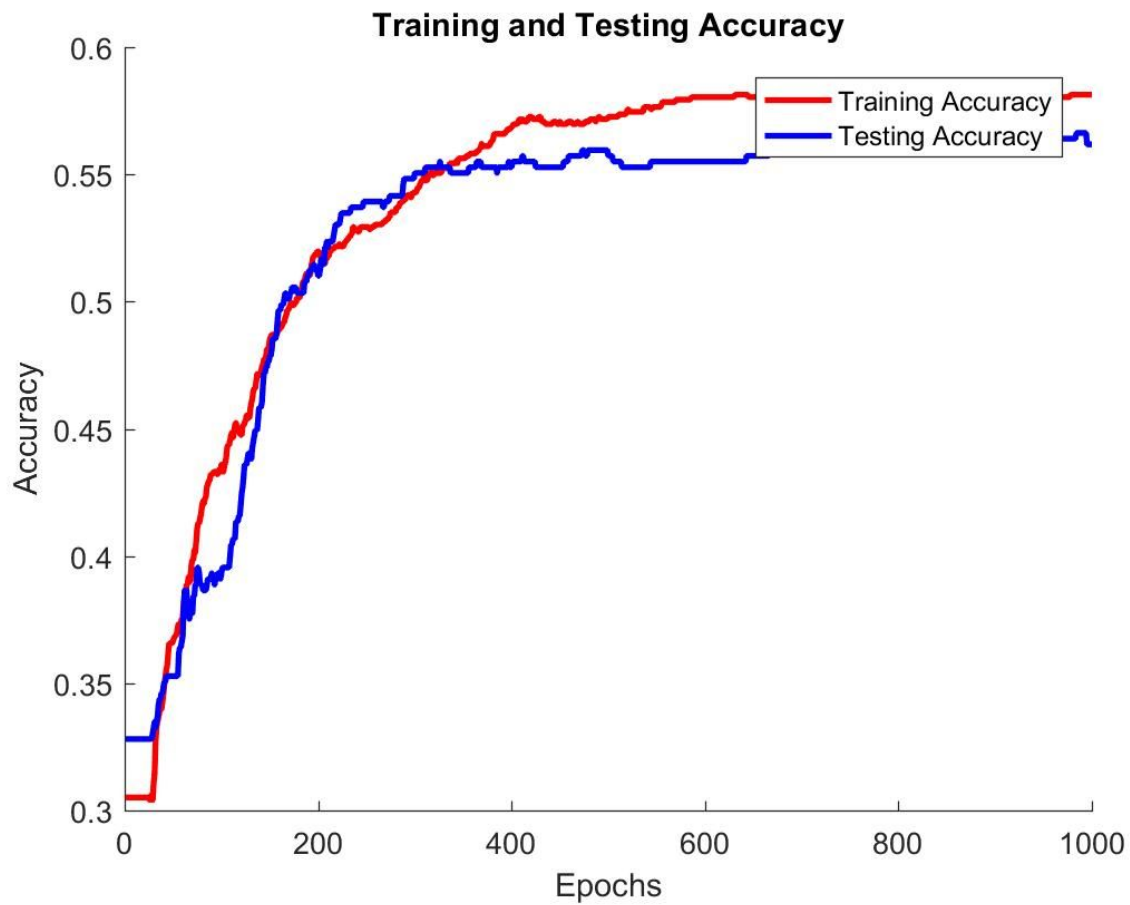


The weights from inputs to hidden node 1 were plotted and the other results were similar to the graph above. It did not seem necessary to plot all three. They all converge at around 250 epochs again showing the goldilocks zone.

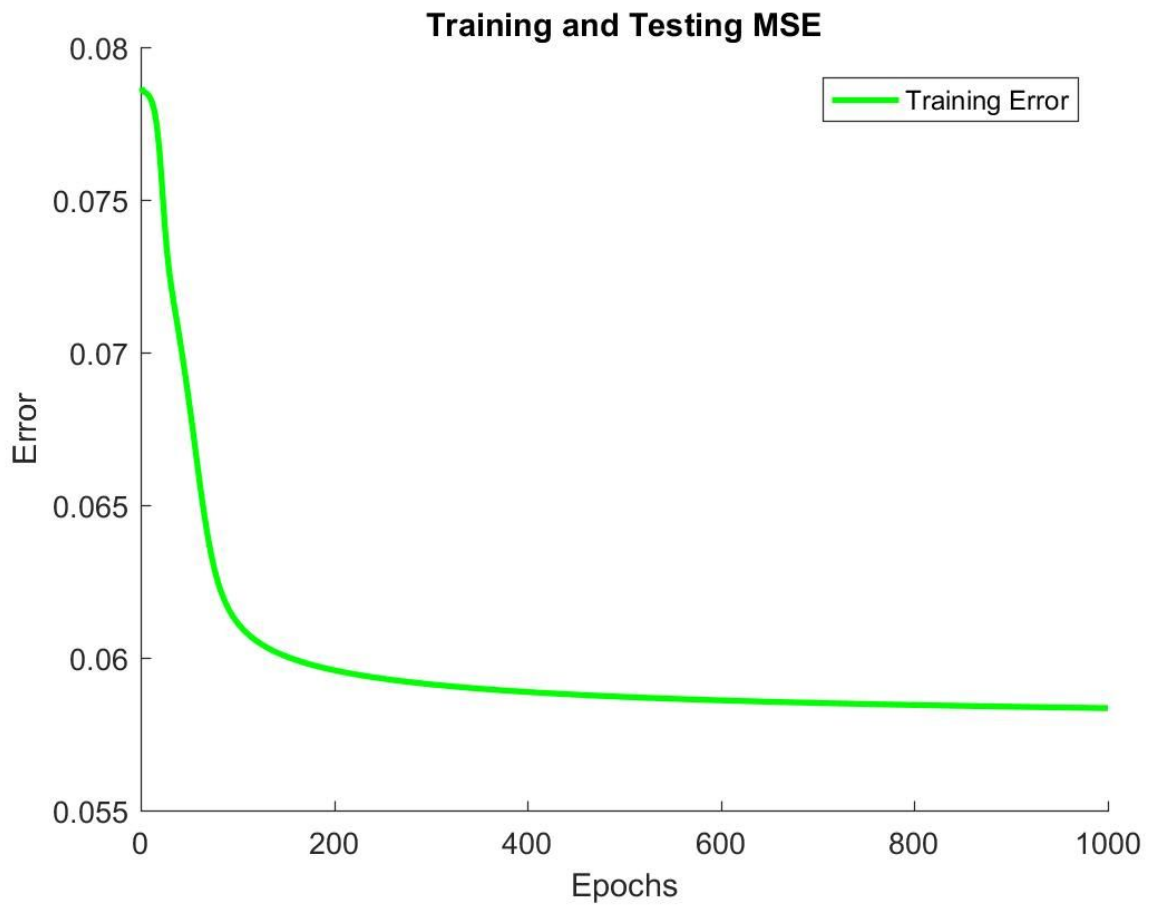
Thi



This shows the last sample after each epoch. Other samples yield similar results, however it is clear why the accuracy is not high because the graph will occasionally not show the correct output. To compare accuracy, the max of these outputs was taken, in this case $y_{\text{Test}} = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$ meaning the actual output is 'CYT'. It overpredicts 'NUC' but lowering the epochs to 250 yields better results for this specific sample but the overall accuracy goes down.



2) Using the whole dataset as the training set, the training error goes down to 0.0584. This is with 1000 epochs and this result highlights that the improvement in learning with the larger dataset is not necessarily a large improvement. The training error for part one was 0.0617 so there is a 0.01 improvement, showing it can learn better with larger datasets.



3) In quotes, the equation solved for by hand is shown. Below the quoted equations will be the code I used to perform that task as well preceded by "//". The workspace is stored as "part3.mat".

Problem 3

" $\delta^L = a^L - y^k$ "

// $d\{L\} = a\{numLayers+1\} - yTrain(i,:);$

$[0.4124, 0.0712, -0.967, 0.026, 0.049, 0.0003, 0.014, 0.018, 0.035, 0.0123]$

$= [0.4124, 0.071, 0.032, 0.0261, 0.0498, 2.62 \cdot 10^{-4}, 0.0144, 0.0182, 0.0358, 0.0123] - [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]$

" $\frac{\partial L(w)}{\partial w_{ij}^{(l-1)}} = -a_j^{(l-1)} \delta_i^{(l)}$ "

// $g\{L\} = [1, a\{numLayers\}]' * d\{L\}$

$\begin{bmatrix} 1 \\ 0.3397 \\ 0.1153 \\ 0.376 \end{bmatrix} \cdot \begin{bmatrix} 0.4124, 0.071, 0.032, 0.0261, 0.0498, 2.62 \cdot 10^{-4}, 0.0144, 0.0182, 0.0358, 0.0123 \end{bmatrix}$

$= \begin{bmatrix} 0.412 & 0.0712 & -0.9675 & 0.026 & 0.0498 & 2.62 \cdot 10^{-4} & 0.0144 & 0.0182 & 0.0358 & 0.0123 \\ 0.14 & 0.0242 & -0.3286 & 0.008 & 0.0169 & 8.92 \cdot 10^{-5} & 0.0049 & 0.0057 & 0.0017 & 0.0057 \\ 0.047 & 0.0082 & -0.1115 & 0.003 & 0.0057 & 3.02 \cdot 10^{-5} & 0.0017 & 0.0014 & 0.0049 & 0.0014 \\ 0.164 & 0.0283 & -0.3846 & 0.01 & 0.0198 & 1.0449 \cdot 10^{-4} & 0.0057 & 0.0042 & 0.0014 & 0.0049 \end{bmatrix}$

$\begin{bmatrix} 0.0182 & 0.0358 & 0.0123 \\ 0.0049 & 0.0121 & 0.0042 \\ 0.0017 & 0.004 & 0.0014 \\ 0.0057 & 0.014 & 0.0049 \end{bmatrix}$

$$W_{ij}^{(l-1)} = W_{ij}^{(l-1)} - \alpha (a_i^{(l-1)} s_j^{(l)})$$

$$W_{\{numlayers+1\}} = W_{\{numlayers+1\}} - \alpha g \{1\}$$

~~$$-2.677 \quad -2.952254$$~~

$$\begin{bmatrix} -1.94 & -2.04 & -6.57 & 3.058 & 0.002 & -1.406 & -2.34 & -1.889 & \dots \\ 3.92 & 3.92 & -0.31 & -9.2 & -3.49 & -5.63 & -1.74 & -1.541 & \dots \\ -0.72 & -3.18 & 5.96 & -6.906 & 1.065 & 2.91 & 3.012 & -1.017 & \dots \\ -1.76 & -0.51 & 2.9 & 2.765 & -4.052 & -4.18 & -4.406 & -1.802 & \dots \end{bmatrix}$$

$$\begin{bmatrix} \dots & -2.95 & -1.594 \\ \dots & -2.29 & -3.096 \\ \dots & 0.36 & -1.23 \\ \dots & 0.007 & -2.6744 \end{bmatrix}$$

$$-(0.01) \begin{bmatrix} 0.412 & 0.07 & -0.9675 & 0.026 & 0.049 & 2.62 \cdot 10^{-4} \\ 0.14 & 0.02 & -0.3286 & 0.0089 & 0.0169 & 8.927 \cdot 10^{-5} & \dots \\ 0.047 & 0.008 & -1.1115 & 0.003 & 0.005 & 3.029 \cdot 10^{-5} & \dots \\ 0.164 & 0.028 & -0.3846 & 0.0104 & 0.0198 & 1.0449 \cdot 10^{-4} & \dots \end{bmatrix}$$

$$\begin{bmatrix} \dots & 2.628 \cdot 10^{-4} & 0.0144 & 0.0182 & 0.0358 & 0.0123 \\ \dots & 8.927 \cdot 10^{-5} & 0.0049 & 0.006 & 0.0121 & 0.0042 \\ \dots & 3.029 \cdot 10^{-5} & 0.0017 & 0.002 & 0.004 & 0.0014 \\ \dots & 1.044 \cdot 10^{-4} & 0.0057 & 0.007 & 0.014 & 0.0049 \end{bmatrix}$$

$$= \begin{bmatrix} -1.95 & -2.05 & -6.56 & 3.06 & 0.001 & -1.406 & -2.348 & \dots \\ 3.92 & 3.92 & -0.31 & -9.2 & -3.49 & -5.634 & -1.44 & \dots \\ -0.72 & -3.19 & 5.97 & -6.91 & 1.065 & 2.915 & 3.01 & \dots \\ -1.77 & -0.52 & 2.9 & 2.77 & -4.053 & -4.188 & -4.406 & \dots \end{bmatrix}$$

$$\begin{bmatrix} \dots & -1.89 & -2.9525 & -1.5944 \\ \dots & -1.54 & -2.299 & -3.0968 \\ \dots & -1.01 & 0.36 & -1.23 \\ \dots & -1.5 & 0.0076 & -2.67 \end{bmatrix}$$

$$\text{//sum}\{l-1\} = d\{l-1\} * W\{\text{numLayers} - l + 3\}';$$

$$[0.41, 0.07, -0.967, 0.026, 0.0448, 2.62 \cdot 10^{-4}, 0.014, 0.0182, 0.0358, 0.0123]$$

$$* \begin{bmatrix} -2.68 & -3.7832 & 2.06 \\ -1.39 & 3.444 & -6.38 \\ -2.74 & 3.0829 & -2.017 \\ 13.05 & 4.194 & 1.57 \\ -0.719 & 6.1763 & 2.59 \\ -0.498 & -4.0644 & 0.386 \\ -2.594 & 0.788 & 0.54 \\ -1.4 & 0.236 & 1.92 \\ 1.89 & -6.89 & 1.17 \end{bmatrix}$$

$$= [5.2939, 1.6156, -6.4037, -3.8248]$$

$$\text{//} d\{l\} = \text{sum}\{l-1\}(2:\text{end}) * (1 - a\{\text{numLayers} - l + 2\}) * a\{\text{numLayers} - l + 2\}$$

$$\text{// } \delta^{(l-1)} = (W^{(l-1)})^T \delta^{(l)} \cdot a^{(l-1)}$$

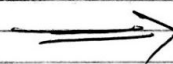
$$[1.6156, -6.4037, -3.8248] * (1 - [0.3397, 0.1153, 0.3976]) * [0.3397, 0.1153, 0.3976]$$

$$= [0.3624, -0.6531, -0.9161]$$

$$\text{// } g\{l\} = [1, x_{\text{train}}(i, :)]' * d\{l\}$$

$$\begin{bmatrix} 1 \\ 0.58 \\ 0.61 \\ 0.47 \\ 0.13 \\ 0.5 \\ 0 \\ 0.48 \\ 0.22 \end{bmatrix}$$

$$* [0.3624, -0.6531, -0.9161]$$



$$= \begin{bmatrix} 0.3624 & -0.653 & -0.9161 \\ 0.21 & -0.378 & -0.5313 \\ 0.22 & -0.398 & -0.5588 \\ 0.17 & -0.3 & -0.4305 \\ 0.04 & -0.08 & -0.1191 \\ 0.18 & -0.32 & -0.458 \\ 0 & 0 & 0 \\ 0.1739 & -0.31 & -0.43 \\ 0.0797 & -0.14 & -0.2 \end{bmatrix} // g_{EL3}$$

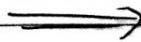
$$W^{(1)} = W^{(1)} - \alpha g_{EL3}$$

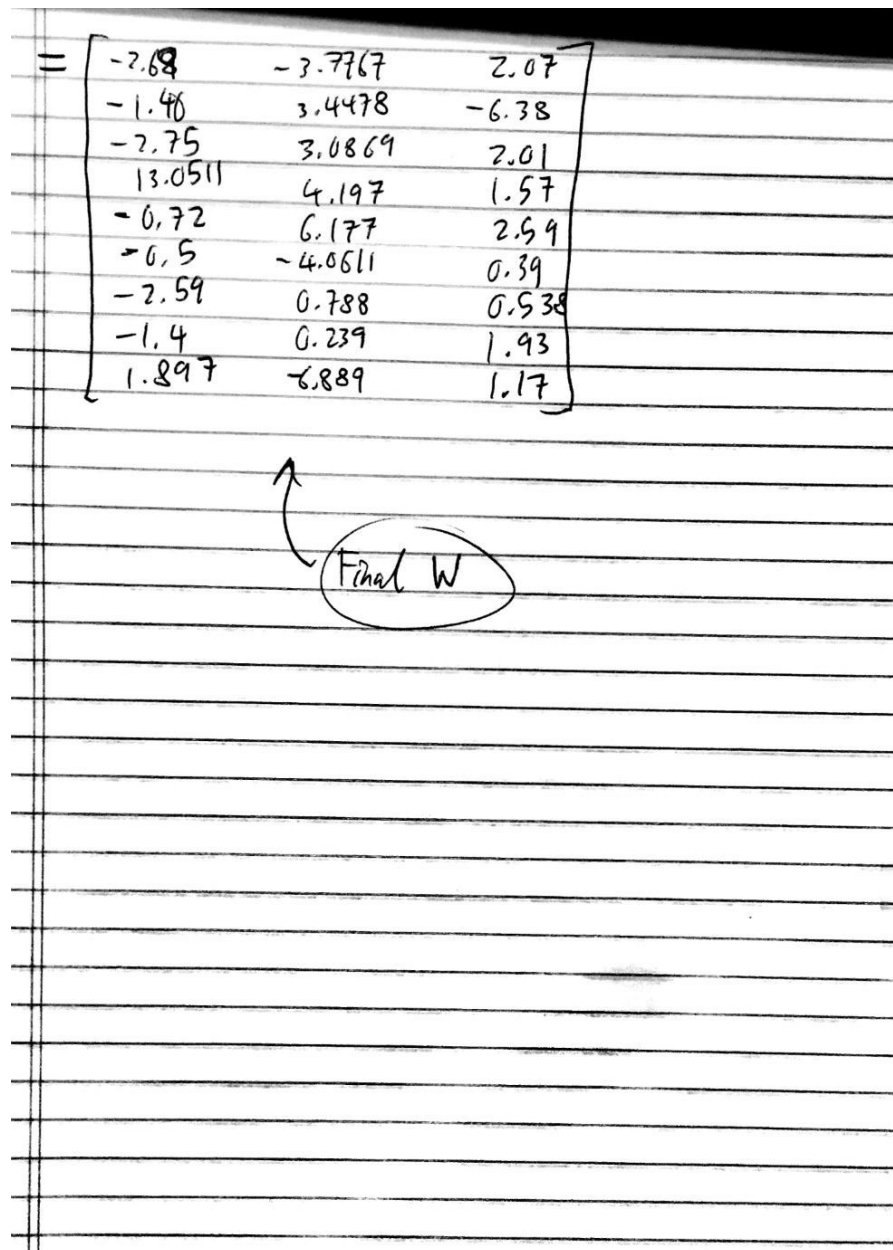
$$\begin{bmatrix} 13.05 & 4.194 & 1.57 \\ -0.719 & 6.17 & 2.59 \\ -0.498 & 4.06 & 0.38 \\ -2.594 & 0.78 & 0.53 \\ -1.402 & 0.2367 & 1.928 \\ 1.89 & -6.89 & 1.17 \end{bmatrix} - (0.01) \begin{bmatrix} 0.3624 & -0.653 & -0.9161 \\ 0.21 & -0.378 & -0.5313 \\ 0.22 & -0.398 & -0.5588 \\ 0.17 & -0.3 & -0.43 \\ 0.04 & -0.08 & -0.1191 \\ 0.18 & -0.32 & -0.458 \\ 0 & 0 & 0 \\ 0.1739 & -0.31 & -0.43 \\ 0.0797 & -0.14 & -0.2 \end{bmatrix}$$

$$\begin{bmatrix} -2.68 & -3.78 & 2.06 \\ -1.39 & 3.44 & -6.38 \\ -2.74 & 3.08 & -2.01 \\ 13.05 & 4.194 & 1.57 \\ -0.71 & 6.176 & 2.59 \\ -0.49 & 4.06 & 0.38 \\ -2.59 & 0.7881 & 0.53 \\ -1.4 & 0.2367 & 1.928 \\ 1.8982 & -6.89 & 1.17 \end{bmatrix}$$

=

Next page





A handwritten 10x3 matrix of weights is shown on lined paper. The matrix is enclosed in large square brackets. Below the matrix, the text "Final W" is circled, and an arrow points from the circle to the matrix.

-2.68	-3.7767	2.07
-1.46	3.4478	-6.38
-2.75	3.0869	2.01
13.0511	4.197	1.57
-0.72	6.177	2.59
-0.5	-4.0611	0.39
-2.59	0.788	0.538
-1.4	0.239	1.93
1.897	-6.889	1.17

alpha = 0.01

```
xTrain(i,:) = [0.58,0.61,0.47,0.13,0.5,0,0.48,0.22];
```

```
yTrain(i,:) = [0,0,1,0,0,0,0,0,0];
```

Were the inputs. The neural network function returns firstW and firstA which are the first samples of the last epoch, these were saved and are below:

w{1, 2}

1	2	3	4	5	6	7	8	9	10
-1.94...	-2.04...	-6.57...	3.0588	0.0022	-1.40...	-2.34...	-1.88...	-2.95...	-1.59...
3.9228	3.9252	-0.31...	-9.20...	-3.49...	-5.63...	-1.44...	-1.54...	-2.29...	-3.09...
-0.72...	-3.18...	5.9691	-6.90...	1.0654	2.9155	3.0121	-1.01...	0.3615	-1.23...
-1.76...	-0.51...	2.9006	2.7652	-4.05...	-4.18...	-4.40...	-1.50...	0.0078	-2.67...

W{1,1}

-2.68...	-3.78...	2.0610
-1.39...	3.4440	-6.38...
-2.74...	3.0829	-2.01...
13.05...	4.1948	1.5705
-0.71...	6.1763	2.5924
-0.49...	-4.06...	0.3867
-2.59...	0.7881	0.5387
-1.40...	0.2367	1.9283
1.8982	-6.89...	1.1715

Since the final w was not returned but the workspace was stored, I verified this in the command line of matlab.

The final equation $w\{1\}-0.01*g\{2\}$ is shown below:

```
Trial>> w{1}-0.01*g{2}
```

```
ans =
```

```

-2.6849    -3.7767     2.0701
-1.3956     3.4478    -6.3832
-2.7494     3.0869    -2.0117
13.0511     4.1978     1.5748
-0.7203     6.1772     2.5936
-0.5001    -4.0611     0.3913
-2.5948     0.7881     0.5387
-1.4043     0.2399     1.9327
 1.8974    -6.8889     1.1735

```

Since alpha is only 0.01 the final weight does not change significantly.

4) Train error matrix (layersXnodes):

0.0621	0.0596	0.0605	0.0618
0.0761	0.0802	0.0978	0.0928
0.0811	0.0921	0.0978	0.1014

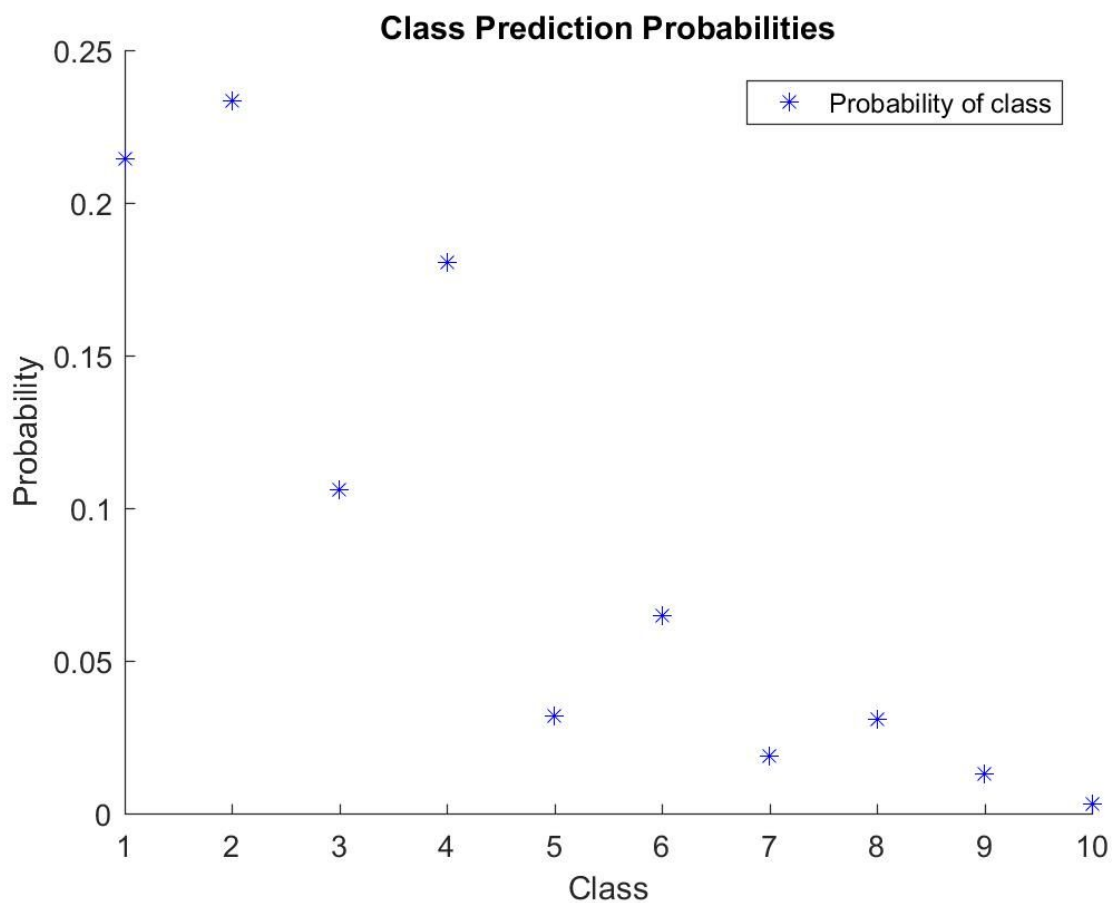
The test error
(layersXnodes) :

matrix is below

0.0619	0.0590	0.0609	0.0612
0.0814	0.0829	0.0961	0.0948
0.0821	0.0911	0.0962	0.0994

From these results it appears that 1 layer and 6 nodes produces the most accurate results. 1 layer 9 nodes is also quite appealing. With too many layers and nodes it appears that overfitting is occurring. The graph below makes this apparent.

5)



The neural network predicts 'NUC' to be the most probable location along with 'CYT' and 'MIT' being likely as well.

6) Error is the best measure of uncertainty we have. Suppose we predicted 'CYT' with probability 1 and all others 0 and this was the correct answer, then we know with 100% certainty this is correct. Taking the difference between the predicted array and this ideal condition (yTest), we see that a larger deviation from this is larger uncertainty. Therefore the uncertainty for part 5 would be.

0.1232

0.3240

0.0001

0.0018

0.0001

0.0000

0.0000

0.0006

0.0000

0.0000

Is the difference. The sum of this array is 0.33225. So the uncertainty is 32%!