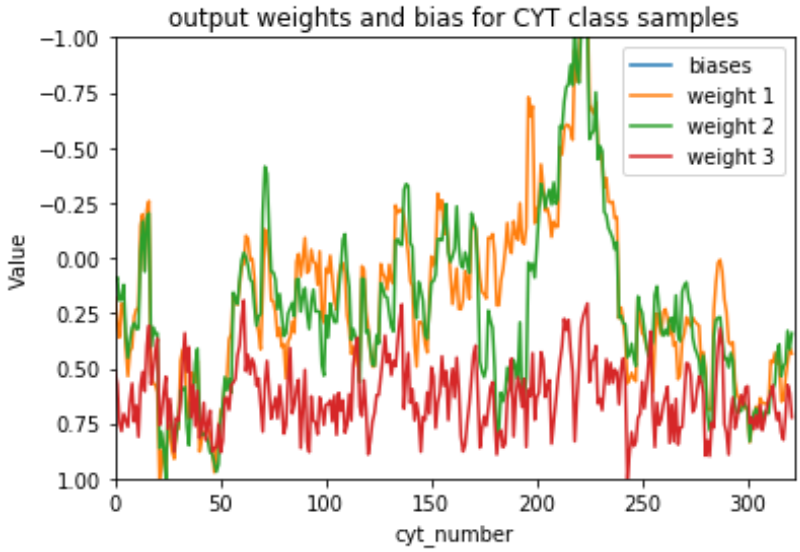
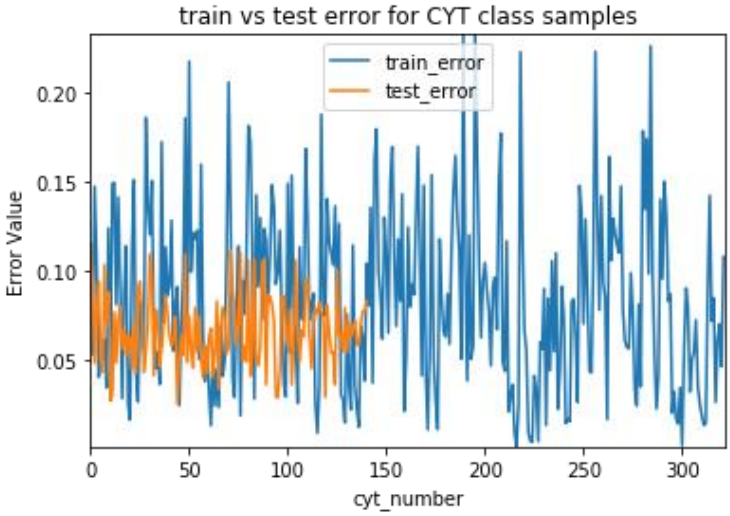
Melody Chang
912110826

ECS171 HW#2

1. *Refer to part1and2.py*

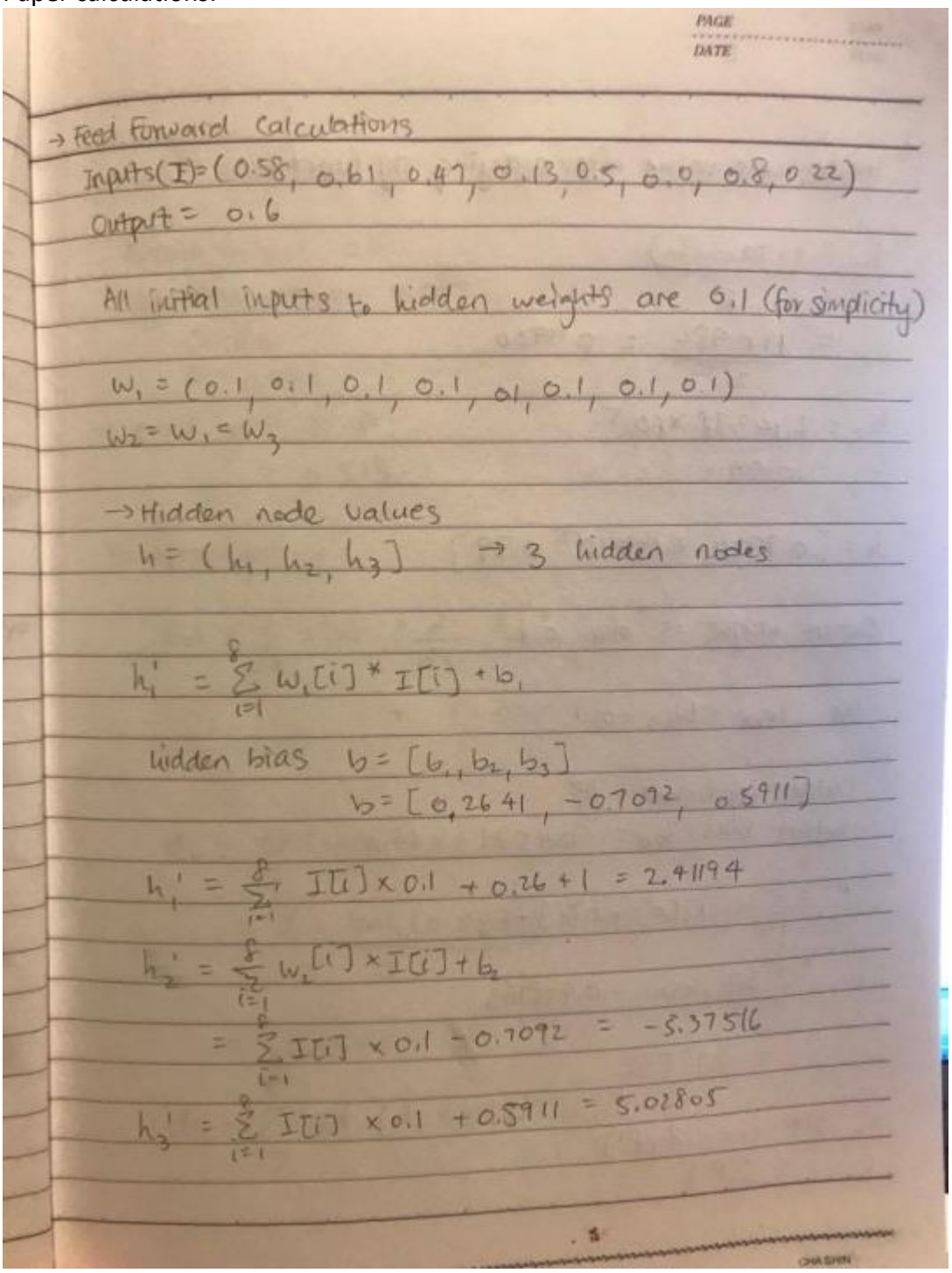




2. *Refer to part1and2.py*
   After retraining the error was 55.02.

3. *Refer to part3.py*

Paper calculations:



→ Feed Forward Calculations

Inputs(I) = (0.58, 0.61, 0.47, 0.13, 0.5, 0.0, 0.8, 0.22)

Output = 0.6

All initial inputs to hidden weights are 0.1 (for simplicity)

$W_1 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)$

$W_2 = W_1 = W_3$

→ Hidden node values

$h = (h_1, h_2, h_3]$ → 3 hidden nodes

$$h_1' = \sum_{i=1}^{8} W_1[i] * I[i] + b_1$$

hidden bias $b = [b_1, b_2, b_3]$

$b = [0.2641, -0.7092, 0.5911]$

$$h_1' = \sum_{i=1}^{8} I[i] \times 0.1 + 0.2641 = 2.41194$$

$$h_2' = \sum_{i=1}^{8} W_2[i] \times I[i] + b_2$$

$$= \sum_{i=1}^{8} I[i] \times 0.1 - 0.7092 = -5.37516$$

$$h_3' = \sum_{i=1}^{8} I[i] \times 0.1 + 0.5911 = 5.02805$$

DATE

Hidden node value after applying sig function:

$h_1 = \frac{1+\tanh(h_1)}{2}$

$= \frac{1+0.984}{2} = 0.9920$

$h_2 = 2.14378 \times 10^{-5}$

$h_3 = 0.99$

$h = [0.9920, 2.14 \times 10^{-5}, 0.99]$

Output weight is also 0.1

$W_{01} = W_{02} = W_{03} = 0.1$

Output - predicted = $\hat{P}$

output bias $b_0 = -0.4581$

$P' = \sum_{i=1}^{3} W_0[i][0] * h[i] + b_0$

$= \sum_{i=1}^{3} h[i] \times 0.1 - 0.458163$

$= -1.17528$

$P = \text{sigmoid}(P')$

$= S(P')$

PAGE
DATE

$\therefore P = 0.08701$

Actual output = 0.6

Predicted output = 0.087

$O = 0.6$

$P = 0.087$

→ error $e = O - P$

$= 0.513$

Backpropagation

$$Sd = \frac{\partial}{\partial x} s(x) = \frac{\partial}{\partial x} \frac{(1 + \tanh(x))}{2}$$

$$= \frac{(1 - \tanh^2(x))}{2}$$

$$d_o = sd(\text{output}) \times e$$

$$= \frac{(1 - \tanh(0.087)^2)}{2} \times 0.513$$

$$= 0.2545$$

$$d_h = \sum_{i=1}^{3} sd\,(d_o \times w_o[i][o]) \times e$$

$dh = [0.0054, 0.0127, 0.0053]$

learning rate is 0.2

$\therefore d = 0.2$

$W_{o1} = W_{o1} + l \times h_1 \times dh_1$

$dh_1 = dh[0]$

$= 0.0057$

$W_{o1} = 0.1$ (old weight)

update:

$W_{o1} = 0.1 + 0.2 \times 0.99 \times 0.0054$

$= 0.1505$

$W_{o2} = 0.100001$

$W_{o3} = 0.1509$

$W_o = [0.1505, 0.1, 0.1509]$

Input for hidden weights W_1, W_2, W_3

$W_1 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]$

$W_1[0] = 0.1 + l \times II[0] \times dh[0]$

$= 0.1 + 0.2 \times 0.58 \times 0.0054$

$= 0.10062$

PAGE
DATE

$W_1 = [0.10062, 0.1, 0.1, 0.10062, 0.10066, 0.1, 0.10062, 0.1006]$

$W_2 = [0.100508, 0.100627, 0.10066, 0.100508, 0.100627, 0.10066, 0.10050, 0.100627]$

$W_3 = [0.10066, 0.10050, 0.10062, 0.10066, 0.100508, 0.100627, 0.10066, 0.100508]$

Biases

$b_0 = -0.4072$

$b = [b_1, b_2, b_3)$

$= [0.265, -0.7067, 0.5922]$

Program output:

```
learning rate = 0.2
('inputs' = [0.58, 0.61, 0.47, 0.13, 0.5, 0.0, 0.48, 0.22])
('input to hidden weights' = [[0.1, 0.1, 0.1], [0.1, 0.1, 0.1], [0.1, 0.1, 0.1], [0.1, 0.1, 0.1], [0.1, 0.1, 0.1],
[0.1, 0.1, 0.1], [0.1, 0.1, 0.1], [0.1, 0.1, 0.1], [0.1, 0.1, 0.1]])
('hidden bias' = [0.26411771, -0.70927067, 0.59113199])
('hidden' = [0.9920285332461958, 2.143785747221738e-05, 0.9999570792326473])
('input * weight + bias' = -0.35896018667538043)
('input * weight + bias' = -0.4581608962142528)
('input * weight + bias' = -0.3581673320767353)
('sum of input * weight + bias' = -1.1752884149663685)
('hidden to output' = [[0.1], [0.1], [0.1]])
('output bias' = [-0.45816304])
('outputs' = [0.0870199338460374])
('error(actual - predicted)' = 0.5129800661539625)
('output delta = ((1 - tanh(predicted out)^2)/2) * error' = 0.2545575335223504)
('hidden deltas = ((1 - tanh(sum(output delta*hidden to output weights))^2)/2) * error' = 0.005410536145576321)
('hidden deltas = ((1 - tanh(sum(output delta*hidden to output weights))^2)/2) * error' = 0.01272787667026802)
('hidden deltas = ((1 - tanh(sum(output delta*hidden to output weights))^2)/2) * error' = 0.005345731096351682)
('hidden to output weights update = hidden to output weights + learning rate*output delta*hidden' = 0.1505056673213893)
('hidden to output weights update = hidden to output weights + learning rate*output delta*hidden' =
0.10000109143362443)
('hidden to output weights update = hidden to output weights + learning rate*output delta*hidden' =
0.15090932154353526)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.1)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.1)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10066008540976032)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.1)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10066008540976032)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10050859039768419)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10066008540976032)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10050859039768419)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10066008540976032)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10050859039768419)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10066008540976032)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10050859039768419)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10066008540976032)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10050859039768419)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10066008540976032)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10050859039768419)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10062762219288686)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10066008540976032)
('input to hidden weights update = input to hidden weights + learning rate*hidden deltas*input' = 0.10050859039768419)
('output biases update = output biases weights + learning rate*output deltas' = -0.40725153329552993)
('input biases update = input biases weights + learning rate*hidden deltas' = 0.2651998172291153)
('input biases update = input biases weights + learning rate*hidden deltas' = -0.7067250946659464)
('input biases update = input biases weights + learning rate*hidden deltas' = 0.5922011362192704)
('total mean square error' = 0.1315742741356619)
```

4. *Refer to part4.py*

```
[0, 3, 6, 9, 12]
[1, 0.04031782679520577, 0.03941068685842449, 0.26057338430583726, 0.26057384377810083]
[2, 0.04142462046408087, 0.038126932825398197, 0.2325451482872513, 0.2605743242174108]
[3, 0.043147008502794564, 0.03527483860189166, 0.23168759637099212, 0.2605743220674406]
Best structure:
                Number of hidden layers = 3
                Neurons per layer= 6
```

The optimal configuration would be 3 hidden layers and 6 nodes. Looking at my output, having less layers and less nodes leads to a smaller accuracy.

5. Using the model we built in the previous questions, we can predict the class for the unknown sample.

```
input = [0.49, 0.51, 0.52, 0.23, 0.55, 0.03, 0.52, 0.39]

output mapping:

['CYT ', 'ERL', 'EXC', 'ME1', 'ME2', 'ME3', 'MIT', 'NUC', 'POX', 'VAC']

ouput vector [26.788311, -14.24732494, -7.07037067, -2.73716855, -21.4946785, -17.22546196, -20.5649147, 1.76462388,
-15.90768528, -2.12868118]

sigmoid(output) = [00000000e+00, 6.49329479e-07, 8.49196222e-04, 6.08154275e-02, 4.62359456e-10, 3.30428058e-08,
1.17157750e-09, 8.53787839e-01, 1.23418431e-07, 1.06340259e-01]

sum of sigmoid(output) = 2.02179

output probabilities = [4.94610399e-01, 3.21165118e-07, 4.20021272e-04, 3.00799422e-02, 2.28687791e-10, 1.63433143e-08,
5.79474413e-10, 4.22292322e-01, 6.10440409e-08, 5.25969975e-02]

so output is [0] which is 'CYT ' with 0.49 probability
```

   The class is CYT with 0.49 probability. However, class NUC is very close as well, with 0.42 probability.

6. I think a good classification of uncertainty would be to see how close the probabilities are to 1 or 0, since 1 would mean that its 100% certain that it's the class, while 0 would mean that it's not close at all. 0.5 can therefore be the cutoff threshold since its in the middle. So for the unknown sample above, I classified this into CYT with probability 0.49. With 0.5 as the cutoff, the uncertainty can therefore be 0.5-0.49 = 0.1 uncertainty.