Assignment 4
Name: Seshasai

Question 1:

Id	Y	D0	hx1(x1>6)	Hx2(x2>6.5)	D1	hx1(x1>8.5)	hx2(x2>9)
1	-	1/10	-	-	1/4e	-	-
2	-	1/10	-	+	1/4e	-	+
3	-	1/10	-	-	1/4e	-	-
4	-	1/10	+	+	e/4	-	-
5	-	1/10	-	-	1/4e	-	-
6	-	1/10	-	-	1/4e	-	-
7	+	1/10	+	+	1/4e	-	-
8	+	1/10	-	+	e/4	-	+
9	+	1/10	+	+	1/4e	+	+
10	+	1/10	+	+	1/4e	-	-

For round 1:
error= 2/10=0.2
alpha= ½ln(1- error / error)
alpha= ½ln(1-0.2/0.2)
alpha= 1
Z1 = sqrt(error(1-error))

Z1=4/10

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

D(1,1) =1/10*e^(-1*-1*-1)/(4/10)=1/4e

We can do similarly for D(1,i) where 1<= i <=10

For round 2:

error= 3/10=0.3
alpha= ½ln(1- error / error)
alpha= ½ln(1-0.3/0.3)
alpha= .6115
Final Hypothesis is based on

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Id	Y
1	-
2	-
3	-
4	+
5	-
6	-
7	+
8	-
9	+
10	+

Assignment 4
Name: Seshasai

Question 2:

Stocastic Gradient Descent for weight updates.

In our problem we ~~…~~ our loss function is cross entropy.

$$E = y\log\hat{y} - (1-y)\log\hat{y}$$

finding gradient of loss function w.r.t weights connecting hidden layer to output.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial s\hat{y}_i}\frac{\partial s\hat{y}_i}{\partial w_{ji}} \quad \text{— (a)}$$

(1) $$\frac{\partial E}{\partial \hat{y}_i} = \frac{-y_i}{\hat{y}_i} + \frac{1-y_i}{1-\hat{y}_i} = \frac{\hat{y}_i - y_i}{\hat{y}_i(1-\hat{y}_i)} \qquad \frac{\partial \hat{y}_i}{\partial s\hat{y}_i} = \hat{y}_i(1-\hat{y}_i) \quad \text{— (2)}$$

$$\frac{\partial s\hat{y}_i}{\partial w_{ji}} = z_j \quad \text{— (3)}$$

Substituting (1) (2) (3) equations in (a) we get gradient of loss w.r.t to weights connecting hidden to output.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\hat{y}_i - y_i}{\hat{y}(1-\hat{y}_i)}\,\hat{y}_i(1-\hat{y}_i)z_j = (\hat{y}_i - y_i)z_j$$

$$\frac{\partial E}{\partial w_{ji}} = (\hat{y}_i - y_i)z_j$$

$i \to$ index of hidden layer node
$j \to$ index of output layer node.

Assignment 4
Name: Seshasai

Computing gradient of loss function w.r.t to weights from input layer to hidden layer.

$$\frac{\partial E}{\partial \omega^h_{kj}} = \frac{\partial E}{\partial S_{z_j}} \frac{\partial S_{z_j}}{\partial \omega^h_{kj}}$$

$$= \sum_{i=1}^{no} \frac{\partial E}{\partial S_{\hat{y}_i}} \frac{\partial S_{\hat{y}_i}}{\partial z_j} \frac{\partial z_j}{\partial S_{z_j}} \frac{\partial S_{z_j}}{\partial \omega^h_{kj}}$$

looking at each component seperatley

$$\frac{\partial E}{\partial S_{\hat{y}_i}} = \hat{y}_i - y_i \; ; \; \frac{\partial S_{\hat{y}_i}}{\partial z_j} = \omega^o_{ji} \quad \frac{\partial z_j}{\partial S_{z_j}} = (1 - z_j^2) \quad \frac{\partial S_{z_j}}{\partial \omega^h_{kj}} = x$$

putting back everything together

$$\frac{\partial E}{\partial \omega^h_{jk}} = \sum_{i=1}^{no} (\hat{y}_i - y_i)\,\omega^o_{ji}\,(1 - z_j^2)\,x_k.$$

j → node in hidden layer.
K → node in input layer.
no → ~~outpu~~ number of output nodes.

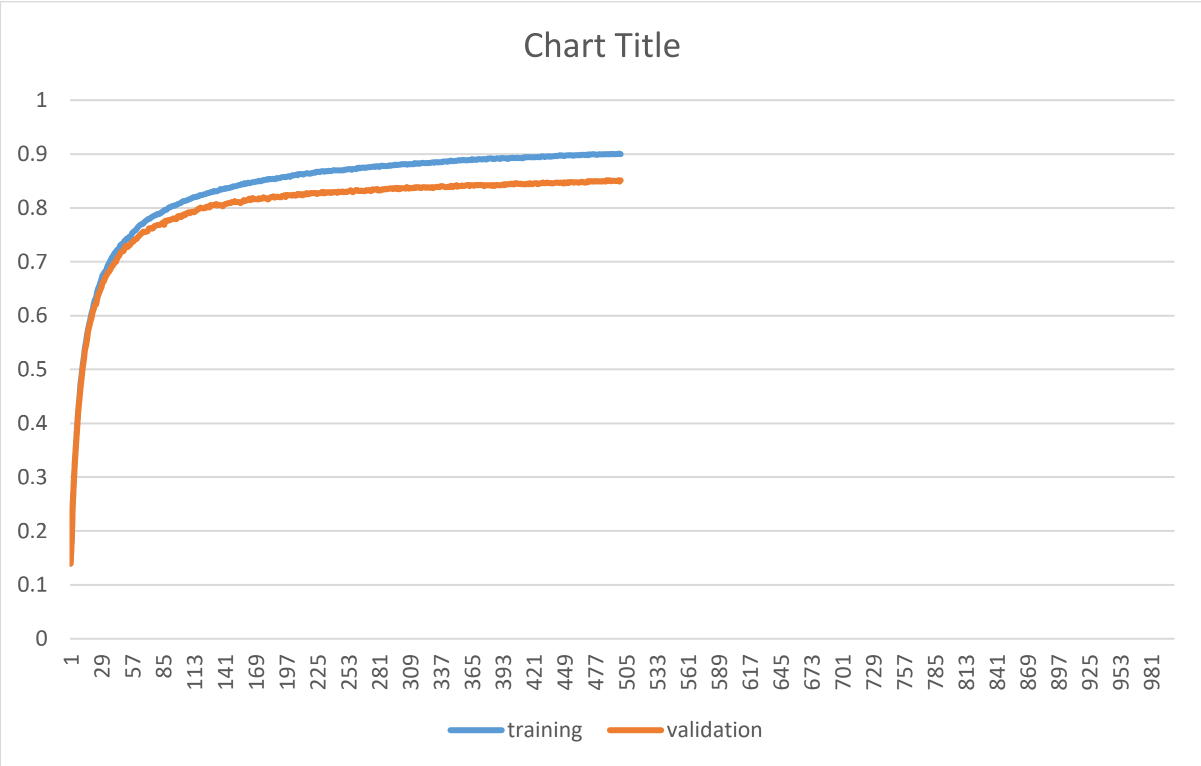
Assignment 4

Name: Seshasai

b.

Learning Rate	Weight initialization	Halting criteria	Validation Accuracy
0.0000063	Gaussian distribution	1000 epoch	88.56
0.000004	Gaussian distribution	1000epoch	86.85
0.00000006	Gaussian distribution	1000 epoch	83.78
0.0000063	Random initialization	1000epoch	10.53

The way I selected hyper parameters for my network is by selecting the combination of parameters that gave me the best validation accuracy. So from the above table I have decided to choose learning rate of 0.0000063, initialize weights from a Gaussian distribution and 1000 epoch of training



Plot of training vs validation

From the graph we can say that network is improving the performance as we increase the number of epochs. This plot is for 500 epochs.

The testing accuracy I get is 86.8.

Summary of Findings:

1. Identifying correct hyper parameters is crucial for neural network performance.
2. Network weights have to be initialized from a Gaussian distribution
3. If we initialize network weights randomly the network for some reason always guesses same label
4. Matrix operations have using batch learning the program runs very fast

Assignment 4
Name: Seshasai

5. Online learning for neural network coded on python is very very slow.
6. For some reason my network seems to converge at validation accuracy of 87%-89% and train accuracy of 91% and there is no improvement of accuracy beyond this point
7. I guess there is something wrong with how I am initializing my weights. I feel so because I think the gradient descent algorithm is finding a local minima and not global minima.