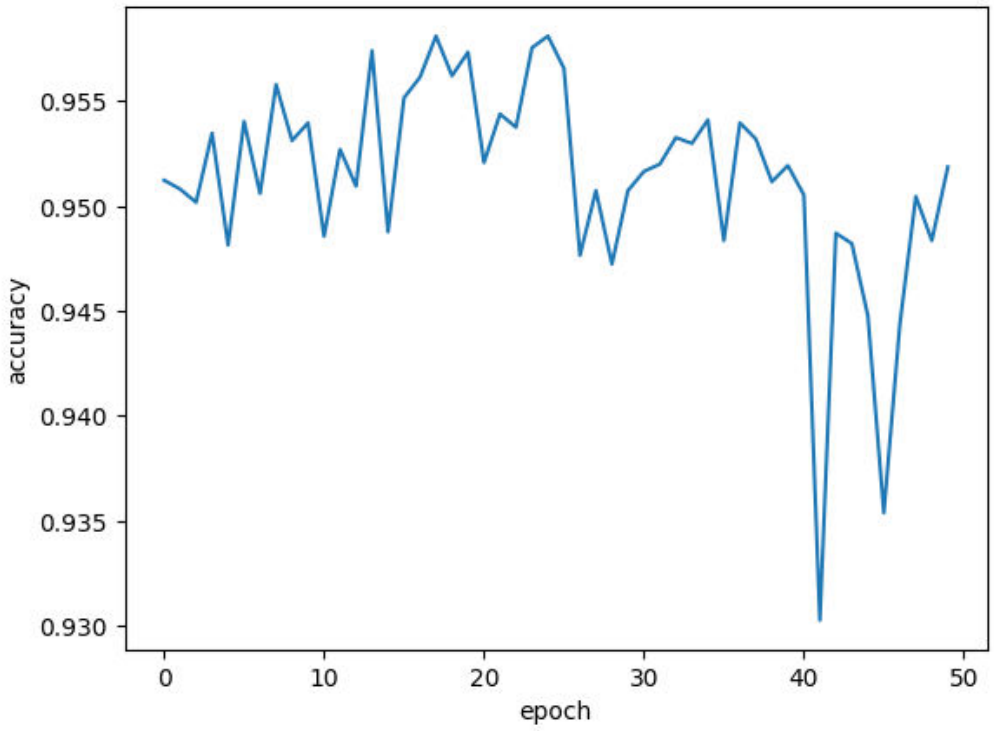
Neural Networks:

1(a)
Applied 5-fold cross-validation

Accuracy obtained with each epoch is as follows:

Epoch	Accuracy
0	0.951232673054
10	0.94856374187
20	0.9520728584
30	0.951652076526
40	0.950530229467
49	0.951863316657

Graph of Epoch vs Accuracy is as follows:



Choice of Hyperparameters: max_iter=50, batch_size=20 and learning_rate=0.001

Method of choosing Hyperparameters = grid search

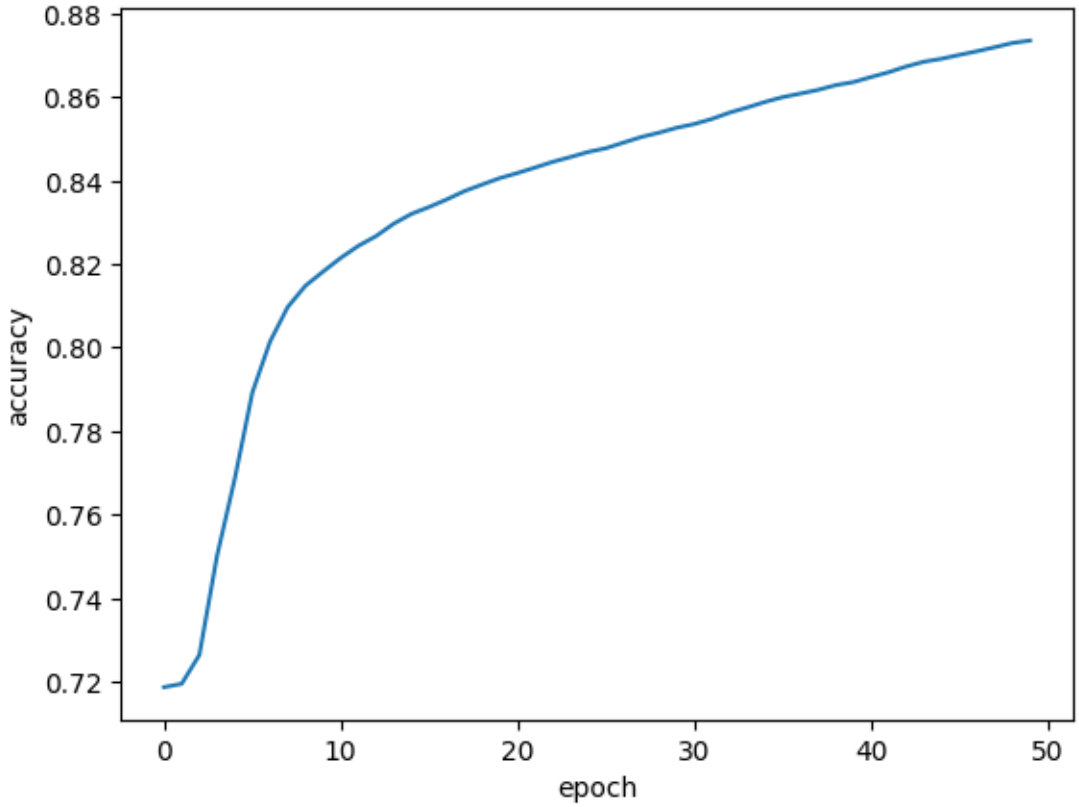
Justification for the choice: If we take large values of learning rate then we may overshoot the minimum,causing the algorithm to climb out of the valley.This was the case when learning rate was taken as 0.1. With eta=0.01 the overshooting problem occured when it came close to minimum.For max_iter, since the dataset was small, so it got trained with only 50 number of iterations.

1(b)
Applied 5-fold cross validation on training data
Accuracy obtained with each epoch is as follows:

Epoch	Validation Accuracy
0	0.7187
10	0.821483333333
20	0.8418
30	0.853533333333
40	0.864783333333
49	0.873516666667

Accuracy on test data : 0.90

Graph of Epoch vs Accuracy is as follows:



Choice of Hyperparameters: max_iter=50, batch_size=20 and learning_rate=0.00001

Method of choosing Hyperparameters = grid search

Justification for the choice: If we take large values of learning rate then we may overshoot the minimum,causing the algorithm to climb out of the valley.This was the case when learning rate was taken as 0.001. With eta=0.0001 the overshooting problem occured when it came close to minimum.For max_iter, as eta was taken as less, so it got trained with only 50 number of iterations.

1(c).

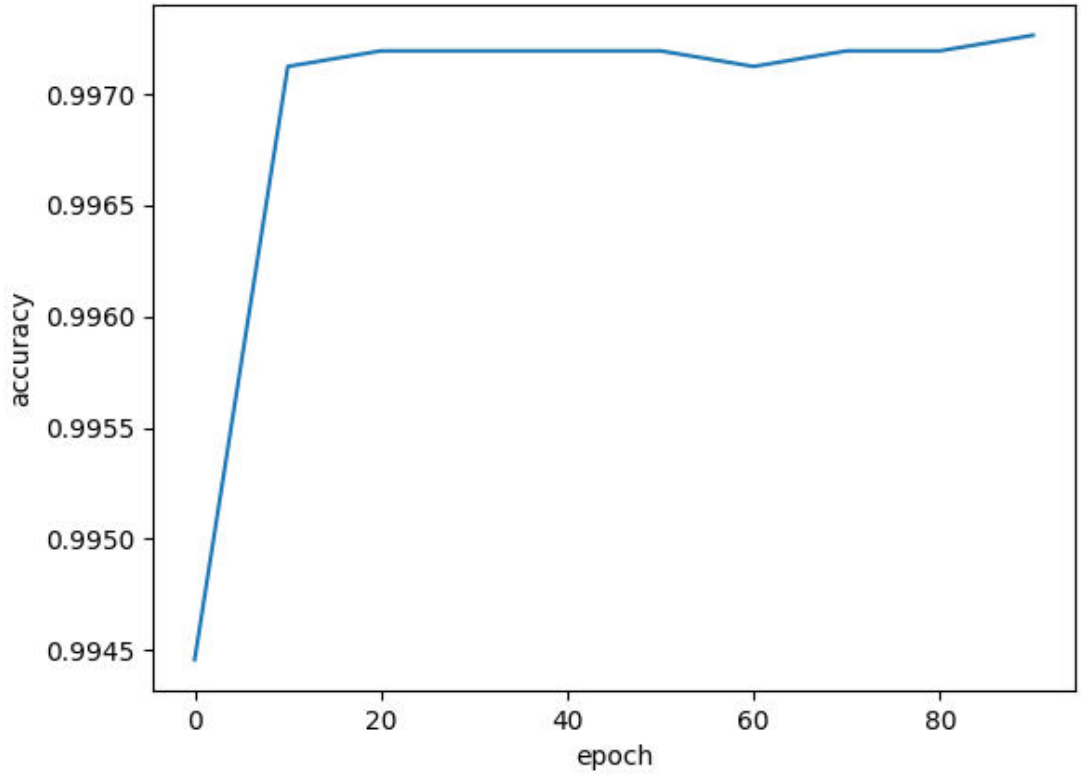
Part A

ReLU

Accuracy obtained with each epoch is as follows:

Epoch	Accuracy
0	0.994457814125
10	0.997123717748
20	0.997193868572
30	0.997193868572
40	0.997193868572
50	0.997193868572
60	0.997123717748
70	0.997193868572
80	0.997193868572
90	0.997264019396

Graph of Epoch vs Accuracy is as follows:



Choice of Hyperparameters: max_iter=100, batch_size=20 and learning_rate=0.0001

Method of choosing Hyperparameters = grid search

Justification for the choice: If we take large values of learning rate then we may overshoot the minimum,causing the algorithm to climb out of the valley.This was the case when learning rate was taken as 0.01. With eta=0.001 the overshooting problem occured when it came close to minimum.For max_iter, since ReLU is not as good as sigmoid for classification problem,max_iter is more as compared to 1A to obtain good accuracy.

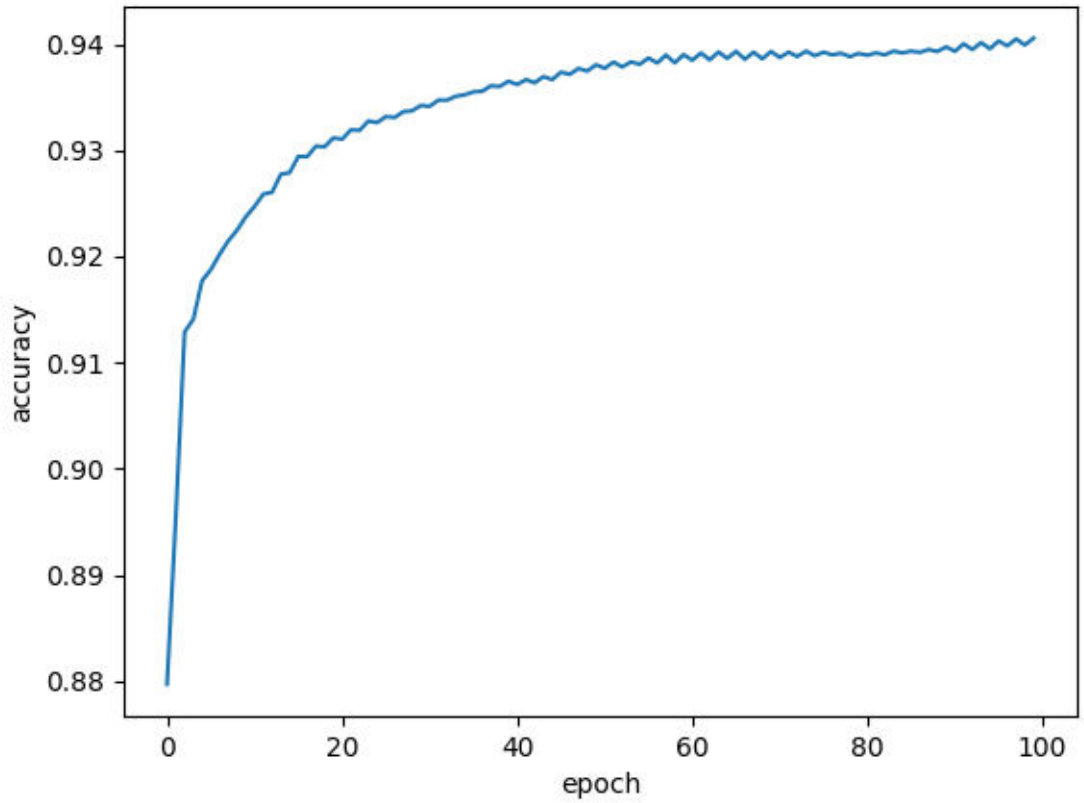
Part B

ReLU

Accuracy obtained with each epoch is as follows:

Epoch	Accuracy
0	0.8797
10	0.924716666667
20	0.93105
30	0.93415
40	0.9362
50	0.937733333333
60	0.9385
70	0.93875
80	0.938966666667
90	0.939333333333
99	0.940583333333

Graph of Epoch vs Accuracy is as follows:



Choice of Hyperparameters: max_iter=100, batch_size=len(td) and learning_rate=0.000001

Method of choosing Hyperparameters = grid search

Justification for the choice: Since ReLU is not working good for high learning rate,it got reduced to above mentioned number.For max_iter, since ReLU is not as good as sigmoid for classification problem,max_iter is more as compared to 1B to obtain good accuracy.

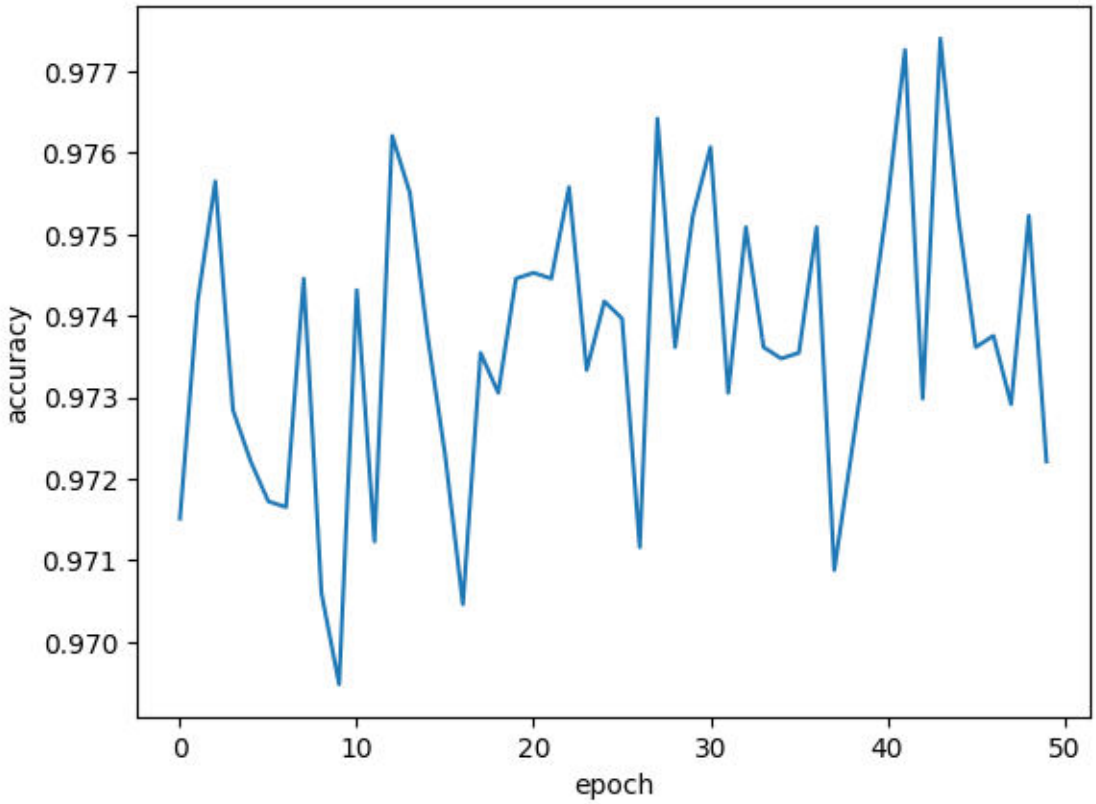
Q2.

(a)
Accuracy obtained with each epoch is as follows:

Epoch	Accuracy(1A)	Accuracy(2A)
0	0.951232673054	0.971511282591
10	0.94856374187	0.974317635548
20	0.9520728584	0.974528506464
30	0.951652076526	0.976072464571
40	0.950530229467	0.975371005557

49	0.951863316657	0.972212150861

Graph of Epoch vs Accuracy(2A) is as follows:



Choice of Hyperparameters: max_iter=50, batch_size=20 and learning_rate_init=0.001 and learning_rate="adaptive"

Method of choosing Hyperparameters = grid search

Justification for deviation in accuracy:
Sklearn model performs better than self made model due to difference in setting of hyperparameters .For example, in sklearn, learning_rate was taken as adaptive, so autoatically updates the learning rate when cost stops decreasing.It also uses regularization which is not done in self build model.

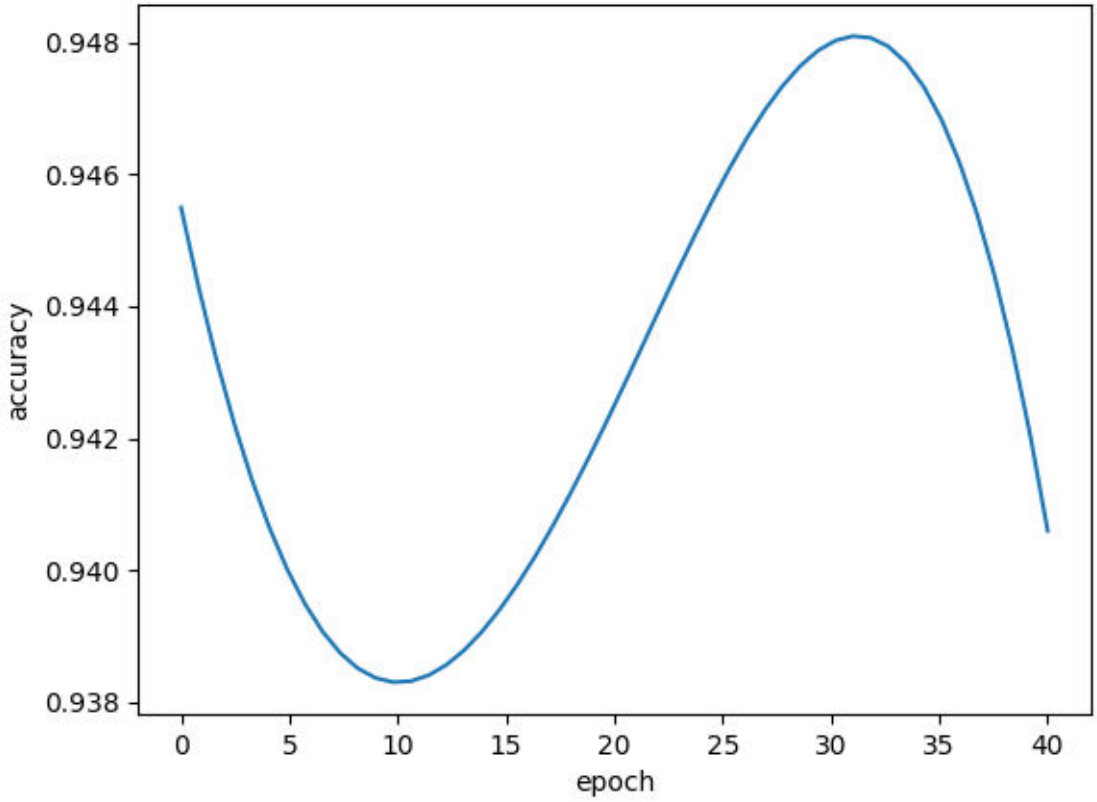
Q2.

(b)
Accuracy obtained with each epoch is as follows:

Epoch	Accuracy(1B)	Accuracy(2B)
0	0.7187	0.9455
10	0.821483333333	0.9383
20	0.8418	0.9425
30	0.853533333333	0.948
40	0.864783333333	0.9406

49	0.873516666667	

Graph of Epoch vs Accuracy(2B) is as follows:



Choice of Hyperparameters: max_iter=50, batch_size=20 and learning_rate_init=0.001 and learning_rate="adaptive"

Method of choosing Hyperparameters = grid search

Justification for deviation in accuracy:
Sklearn model performs better than self made model due to difference in setting of hyperparameters .For example, in sklearn, learning_rate was taken as adaptive, so autoatically updates the learning rate when cost stops decreasing.It also uses regularization which is not done in self build model.

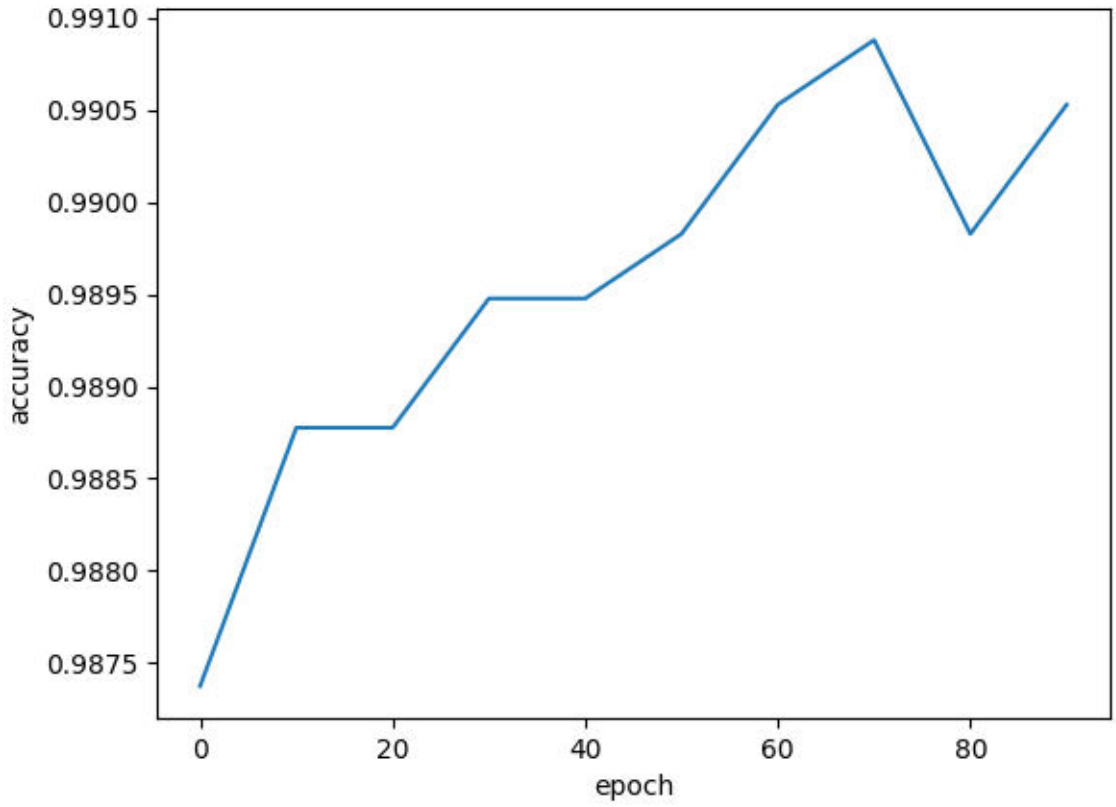
Q3(a)
Dataset: MNIST Subset
Network Structure 1:

- num of hidden layers = 3
- sizes = 100,50,30

Accuracy obtained with each epoch is as follows:

Epoch	Accuracy
0	0.987372851631
10	0.988775868116
20	0.988775868116
30	0.989477376359
40	0.989477376359
50	0.989828130481
60	0.990529638723
70	0.93875
80	0.989828130481
90	0.990529638723

Graph of Epoch vs Accuracy is as follows:



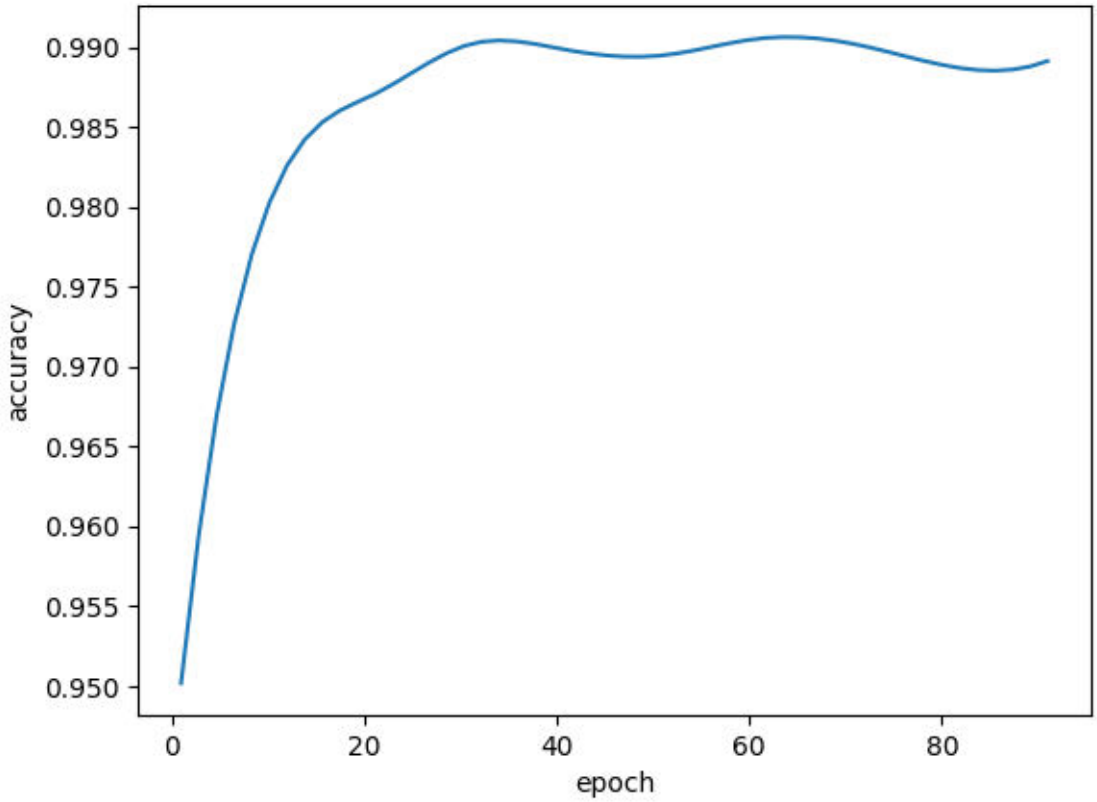
Network Structure 2:

- num of hidden layers = 1
- sizes = 100,

Accuracy obtained with each epoch is as follows:

Epoch	Accuracy
1	0.950192914767
11	0.981410031568
21	0.98702209751
31	0.990178884602
41	0.989828130481
51	0.989477376359
61	0.990529638723
71	0.990178884602
81	0.988775868116
91	0.989126622238

Graph of Epoch vs Accuracy is as follows:



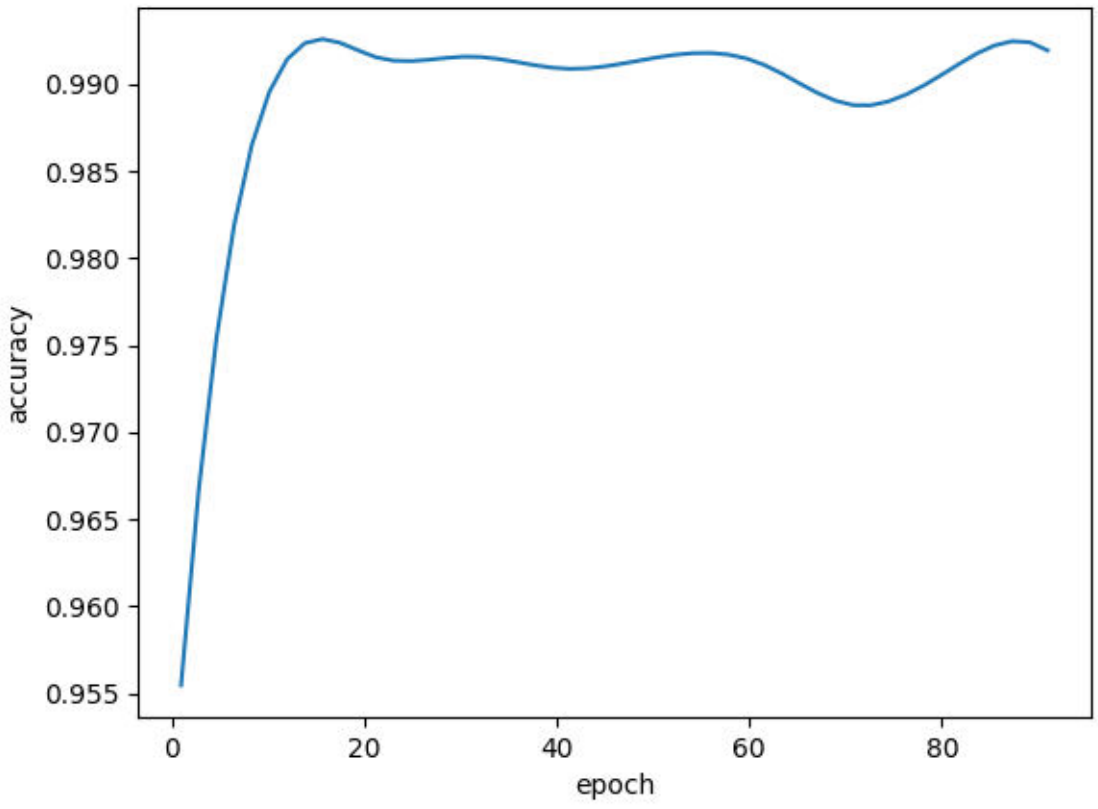
Network Structure 3:

- num of hidden layers = 2
- sizes = 200,100
- activation =logistic

Accuracy obtained with each epoch is as follows:

Epoch	Accuracy
1	0.955454226587
11	0.990529638723
21	0.991581901087
31	0.991581901087
41	0.990880392845
51	0.991581901087
61	0.991231146966
71	0.988775868116
81	0.990880392845
91	0.991932655209

Graph of Epoch vs Accuracy is as follows:



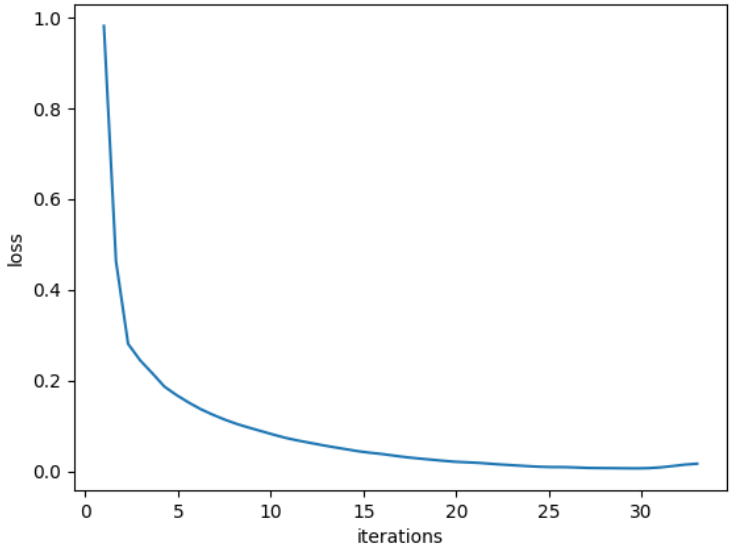
Dataset: MNIST Dataset

Network Structure 1:

- num of hidden layers = 2
- sizes = 800,400
- activation =logistic

```
garvita@garvita-HP-Pavilion-15-Notebook-PC:~/Desktop/HW-3-NN$ python Q3b.py
Iteration 1, loss = 0.98179175
Iteration 2, loss = 0.33760748
Iteration 3, loss = 0.24319401
Iteration 4, loss = 0.19703776
Iteration 5, loss = 0.16589730
Iteration 6, loss = 0.14155628
Iteration 7, loss = 0.12246964
Iteration 8, loss = 0.10658163
Iteration 9, loss = 0.09410578
Iteration 10, loss = 0.08242464
Iteration 11, loss = 0.07166652
Iteration 12, loss = 0.06359692
Iteration 13, loss = 0.05586975
Iteration 14, loss = 0.04901569
Iteration 15, loss = 0.04249762
Iteration 16, loss = 0.03801438
Iteration 17, loss = 0.03245222
Iteration 18, loss = 0.02806958
Iteration 19, loss = 0.02425226
Iteration 20, loss = 0.02077965
Iteration 21, loss = 0.01899502
Iteration 22, loss = 0.01606421
Iteration 23, loss = 0.01340566
Iteration 24, loss = 0.01092802
Iteration 25, loss = 0.00930133
Iteration 26, loss = 0.00903488
Iteration 27, loss = 0.00745955
Iteration 28, loss = 0.00698693
Iteration 29, loss = 0.00659994
Iteration 30, loss = 0.00646925
Iteration 31, loss = 0.00845120
Iteration 32, loss = 0.01303723
Iteration 33, loss = 0.01667849
Training loss did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
score  0.9762
```

Graph of Iterations vs Loss is as follows:

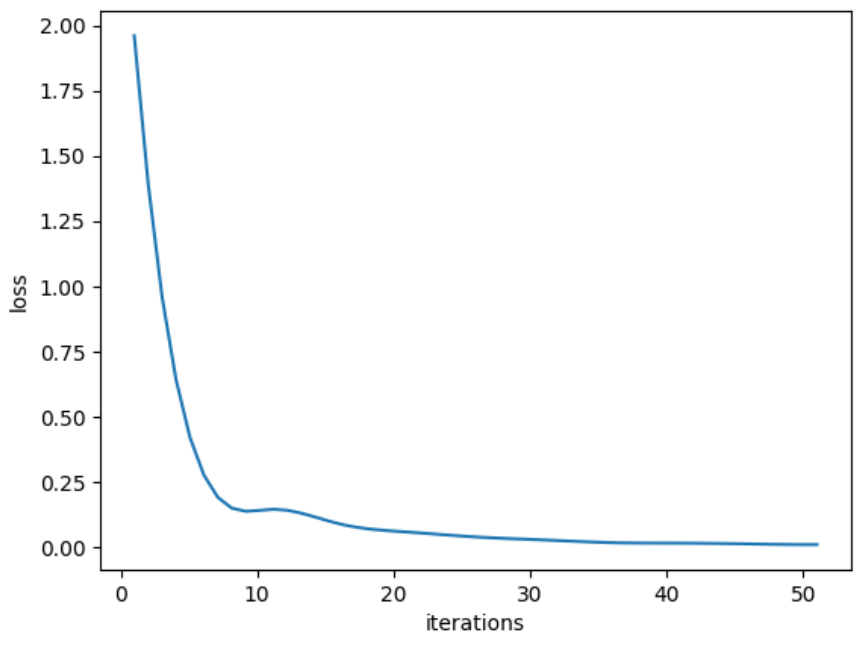


Network Structure 2:

- num of hidden layers = 3
- sizes = 400,200,100
- activation =logistic

```
Iteration 13, loss = 0.11972951
Iteration 14, loss = 0.10938313
Iteration 15, loss = 0.09918914
Iteration 16, loss = 0.09100515
Iteration 17, loss = 0.08185494
Iteration 18, loss = 0.07625494
Iteration 19, loss = 0.07044967
Iteration 20, loss = 0.06449928
Iteration 21, loss = 0.05951442
Iteration 22, loss = 0.05450331
Iteration 23, loss = 0.05070598
Iteration 24, loss = 0.04725323
Iteration 25, loss = 0.04358431
Iteration 26, loss = 0.04058519
Iteration 27, loss = 0.03744438
Iteration 28, loss = 0.03465684
Iteration 29, loss = 0.03295911
Iteration 30, loss = 0.03137921
Iteration 31, loss = 0.02913889
Iteration 32, loss = 0.02712169
Iteration 33, loss = 0.02574386
Iteration 34, loss = 0.02424204
Iteration 35, loss = 0.02245606
Iteration 36, loss = 0.01878326
Iteration 37, loss = 0.01813082
Iteration 38, loss = 0.01738118
Iteration 39, loss = 0.01654995
Iteration 40, loss = 0.01603695
Iteration 41, loss = 0.01686007
Iteration 42, loss = 0.01752622
Iteration 43, loss = 0.01517109
Iteration 44, loss = 0.01465881
Iteration 45, loss = 0.01276235
Iteration 46, loss = 0.01393551
Iteration 47, loss = 0.01179733
Iteration 48, loss = 0.01222481
Iteration 49, loss = 0.01098456
Iteration 50, loss = 0.01114716
Iteration 51, loss = 0.01150644
Iteration 52, loss = 0.01223240
Training loss did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
score  0.97
```

Graph of Iterations vs Loss is as follows:

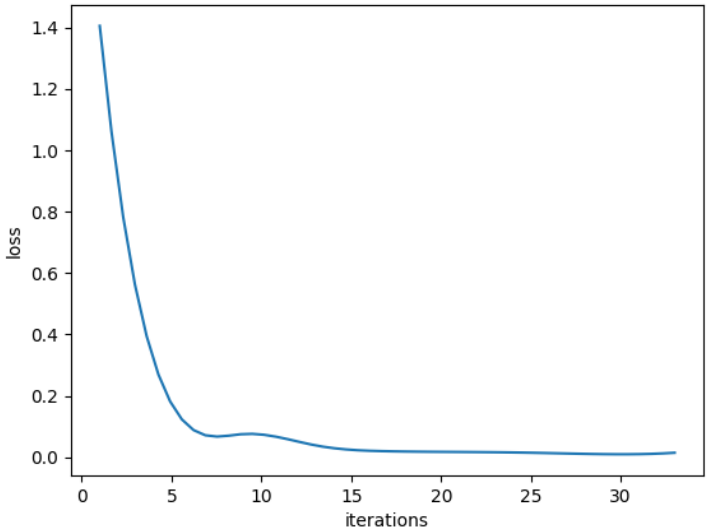


Network Structure 3:

- num of hidden layers = 3
- sizes = 1000,500,200
- activation =logistic

```
garvita@garvita-HP-Pavilion-15-Notebook-PC:~/Desktop/HW-3-NN$ python Q3b.py
Iteration 1, loss = 1.40602211
Iteration 2, loss = 0.46299275
Iteration 3, loss = 0.28361163
Iteration 4, loss = 0.21477658
Iteration 5, loss = 0.17305881
Iteration 6, loss = 0.14236757
Iteration 7, loss = 0.11975025
Iteration 8, loss = 0.10248769
Iteration 9, loss = 0.08858607
Iteration 10, loss = 0.07570269
Iteration 11, loss = 0.06562801
Iteration 12, loss = 0.05704857
Iteration 13, loss = 0.04989504
Iteration 14, loss = 0.04386241
Iteration 15, loss = 0.03880484
Iteration 16, loss = 0.03240038
Iteration 17, loss = 0.02925948
Iteration 18, loss = 0.02453400
Iteration 19, loss = 0.02127336
Iteration 20, loss = 0.01958353
Iteration 21, loss = 0.01648480
Iteration 22, loss = 0.01664497
Iteration 23, loss = 0.01635198
Iteration 24, loss = 0.01572526
Iteration 25, loss = 0.01724365
Iteration 26, loss = 0.01666746
Iteration 27, loss = 0.01560886
Iteration 28, loss = 0.01345528
Iteration 29, loss = 0.01411616
Iteration 30, loss = 0.01469102
Iteration 31, loss = 0.01233290
Iteration 32, loss = 0.00927808
Iteration 33, loss = 0.01011124
Iteration 34, loss = 0.00968257
Iteration 35, loss = 0.01025035
Training loss did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
score  0.9756
```

Graph of Iterations vs Loss is as follows:

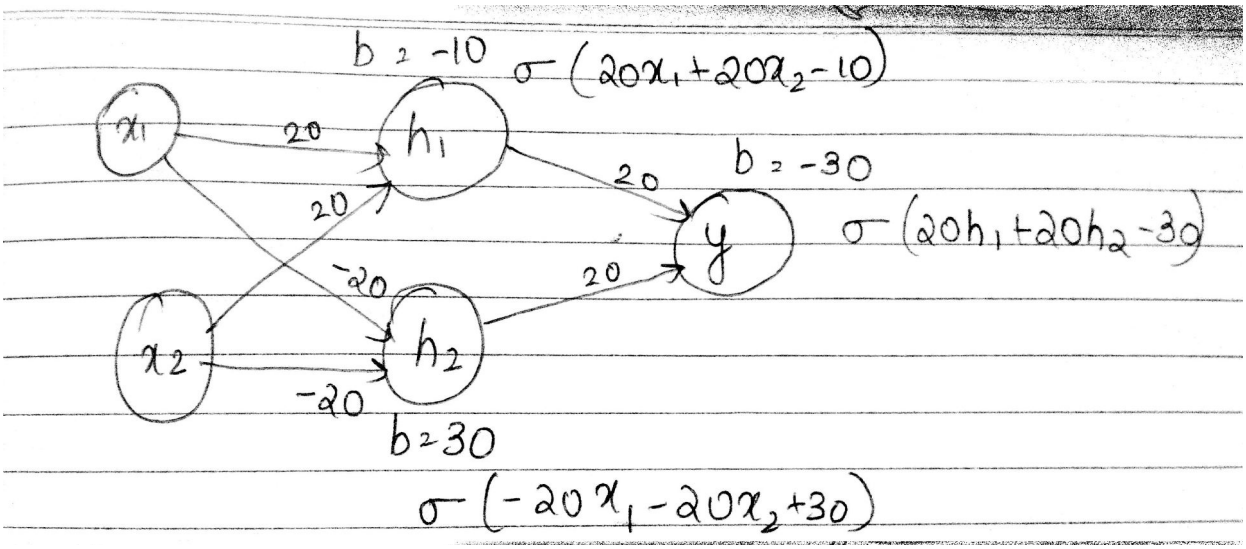


INSIGHTS AND RESULTS:

- For MNIST subset the network structure with 2 hidden layers(Network 3) performs better than 1 and 3 hidden layers.
- For complete Dataset,network structure 1 with two hidden layers performed best.
- The models tried are with 1 hidden layers, 2 hidden layers and 3 layers.
- 2 hidden layers is ideal for neural networks as 1 layer fails to recognize complex objects and more than 2 results in vanishing gradient problem.(Both for dataset A and dataset B)
- The number of neuron is ideally between neurons in input and output layers.The number of hidden neurons should be less than twice the size of the input layer.(Both for A and B best structure has neurons less than twice the size of input)
- Too few neurons leads to underfitting and too many leads to overfitting.

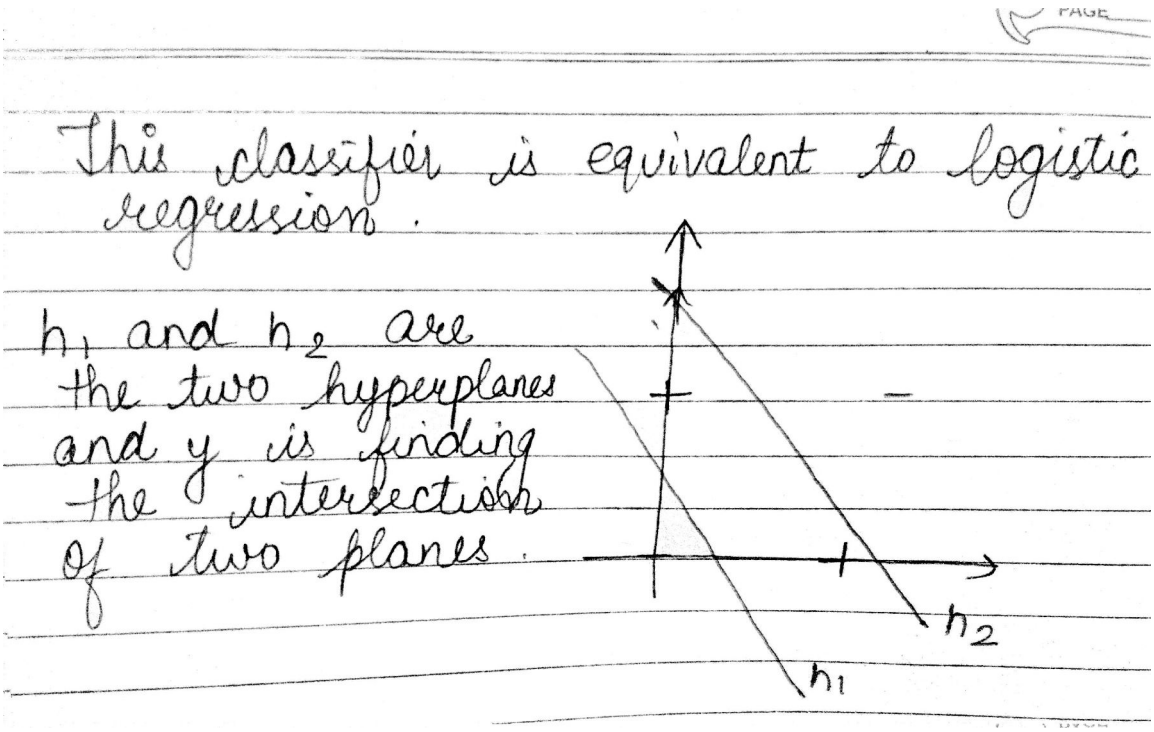
Theory Questions

Q1. yes, a neural net of arbitrary depth using just linear activation functions **can be used** to model the XOR truth table.



For example: when x1=1 and x2=1
h1=fn(30)=1
h2=fn(-10)=0
y=fn(-10)=0
assuming fn(x)=0 if x<0 and fn(x)=1 if x>0 (linear activations)

It is analogous to logistic regression.



Q2. When X uses sigmoid function,the reasons for bad training may be as follows:

- sigmoid suffers from flatspot problem that is when value of output neuron is near 1 or 0.The value of derivative of sigmoid which is given by $o_j(1-o_j)$ where o_j is the sigmoid output from unit j approaches 0 which is backpropogated.This is known as **vanishing gradient problem.** The network refuses to learn furthur or becomes very slow.

When X uses ReLU activation,**the problem get worsen** due to following reason:

- ReLU suffers from **dying ReLU problem** i.e. for negative X, the gradient can go towards 0.So, for activations in that regions the weights will not be adjusted during descent and the neurons in that region stops responding to changing input.This causes several neurons to die making a substantial part of network passive.

Preprocessing Tehnique while using sigmoid function is:

- Adding a small value such as 0.1 to the derivative function will remove the problem of vanishing gradient. As no change in activation function is made, this is done only at the time of training.

Preprocessing Tehnique while using ReLU function is:

- Use Leaky ReLU i.e. add a little slope to the horizontal line($x<0$).This makes the gradient non-zero and prevent from dying ReLU problem.

Q3.Learning slow-down problem in quadratic cost function is due to derivative of sigmoid function which gets very small when the neuron's output is close to 1.The equation of quadratic cost function is given by:

$$C = \frac{(y-a)^2}{2},$$

The derivative of this term w.r.t w and b is given by:

$$\frac{\partial C}{\partial w} = (a - y)\sigma'(z)x = a\sigma'(z)$$
$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z) = a\sigma'(z),$$

Thus,when derivative of sigmoid gets small(when value of output neuron gets close to 1),derivation of cost function w.r.t. w and b also gets small which originates slow learning.

This is not the case in cross-entropy cost function.Cross-entropy function is given by:

$$C = -\frac{1}{n}\sum_x [y \ln a + (1 - y)\ln(1 - a)]$$

Here n is the total number of training datapoints and y is desired output.
Differentiating it w.r.t w gives:

$$\begin{aligned}\frac{\partial C}{\partial w_j} &= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1 - y)}{1 - \sigma(z)}\right)\frac{\partial \sigma}{\partial w_j} \\ &= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1 - y)}{1 - \sigma(z)}\right)\sigma'(z)x_j.\end{aligned}$$

On furthur simpliying it, we get

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y).$$

This expression suggests that the rate at which weights are learned depends of difference between actual and desired output.The larger the error,the faster the neuron will learn.**This function is free of slow learning problem as it is free from derivative of sigmoid which is the actual reaon for slowdown.**
Similarly expression for derivative w.r.t. bias is:

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z) - y)$$