APL 405 – HOMEWORK(2)

Machine Learning for Mechanics

Author
S.B.AMLAN DAS – 2021AM10780
Department Of Applied Mechanics
March – 2024

1 Question - 1

1.1 Part - a

Starting with the defination and expanding the expreesion of MSE ,

$$\begin{aligned}
MSE(\hat{w}) &= E_T[\|\hat{w} - w^*\|_2^2] \\
&= E_T[(\hat{w} - w^*)^T(\hat{w} - w^*)] \\
&= E_T[(\hat{w} - \mu_{\hat{w}} + \mu_{\hat{w}} - w^*)^T(\hat{w} - \mu_{\hat{w}} + \mu_{\hat{w}} - w^*)] \\
&= E_T[(\hat{w} - \mu_{\hat{w}})^T(\hat{w} - \mu_{\hat{w}}) + 2(\hat{w} - \mu_{\hat{w}})^T(\mu_{\hat{w}} - w^*) + (\mu_{\hat{w}} - w^*)^T(\mu_{\hat{w}} - w^*)] \\
&= E_T[\|\hat{w} - \mu_{\hat{w}}\|_2^2] + 2E_T[(\hat{w} - \mu_{\hat{w}})^T(\mu_{\hat{w}} - w^*)] + \|\mu_{\hat{w}} - w^*\|_2^2 \\
&= E_T[\|\hat{w} - \mu_{\hat{w}}\|_2^2] + 2(\mu_{\hat{w}} - w^*)^T E_T[\hat{w} - \mu_{\hat{w}}] + \|\mu_{\hat{w}} - w^*\|_2^2 \\
&= E_T[\|\hat{w} - \mu_{\hat{w}}\|_2^2] + 2(\mu_{\hat{w}} - w^*)^T(\mu_{\hat{w}} - \mu_{\hat{w}}) + \|\mu_{\hat{w}} - w^*\|_2^2 \\
&= E_T[\|\hat{w} - \mu_{\hat{w}}\|_2^2] + 0 + \|\mu_{\hat{w}} - w^*\|_2^2 \\
&= E_T[\|\hat{w} - \mu_{\hat{w}}\|_2^2] + \|\mu_{\hat{w}} - w^*\|_2^2 \\
&= Var(\hat{w}) + \|\mu_{\hat{w}} - w^*\|_2^2 \\
&= tr(Cov(\hat{w})) + \|\mu_{\hat{w}} - w^*\|_2^2
\end{aligned}$$

Hence the result has been proved .

1.2 Part - b

1. Ordinary Least Squares (OLS) Estimator:

Given that $\hat{w}_{OLS} = \arg\min_w \|y - Xw\|_2^2$, we know that the OLS estimator can be derived as:

$$\hat{w}_{OLS} = (X^T X)^{-1} X^T y$$

Now, let's compute its expectation:

$$E_T[\hat{w}_{OLS}] = E_T[(X^T X)^{-1} X^T y]$$

Since X and y are random variables, their expectation is involved. Given the model, $y = Xw^* + \epsilon$, where ϵ follows a normal distribution, we have:

$$E_T[y] = Xw^*$$

Hence:

$$E_T[\hat{w}_{OLS}] = (X^T X)^{-1} X^T E_T[y] = (X^T X)^{-1} X^T X w^* = w^*$$

So, $E_T[\hat{w}_{OLS}] = w^*$, showing that the OLS estimator is an unbiased estimator of w^*.

2. Ridge Regression Estimator:

Given that $\hat{w}_{ridge} = \arg\min_w \|y - Xw\|_2^2 + \lambda\|w\|_2^2$, we know that the Ridge regression estimator can be derived as:

$$\hat{w}_{ridge} = (X^T X + \lambda I_p)^{-1} X^T y$$

Now, let's compute its expectation:

$$E_T[\hat{w}_{ridge}] = E_T[(X^T X + \lambda I_p)^{-1} X^T y]$$

Using the same reasoning as before, we can compute:

$$E_T[\hat{w}_{ridge}] = (X^T X + \lambda I_p)^{-1} X^T E_T[y]$$

$$= (X^T X + \lambda I_p)^{-1} X^T X w^* = (X^T X + \lambda I_p)^{-1} X^T X w^*$$

Thus, $E_T[\hat{w}_{ridge}] = (X^T X + \lambda I_p)^{-1} X^T w^*$.

This completes part (b), showing that $\hat{w}_{OLS}$ is an unbiased estimator of w^*, whereas $\hat{w}_{ridge}$ is a biased estimator of w^*.

1.3 Part - c

Ridge Regression Variance:

First, let's compute the trace of the covariance matrix for the Ridge regression estimator, $\text{tr}[\text{Cov}(\hat{w}_{ridge})]$.

Given that the Ridge regression estimator is $\hat{w}_{ridge} = (X^T X + \lambda I_p)^{-1} X^T y$, the covariance matrix can be derived as:

$$\text{Cov}(\hat{w}_{ridge}) = \sigma^2 (X^T X + \lambda I_p)^{-1}$$

The trace of a matrix is the sum of its eigenvalues.

Let $\gamma_1 \geq \gamma_2 \geq ... \geq \gamma_p$ denote the eigenvalues of the matrix $X^T X$ arranged in non-decreasing order. Then, the eigenvalues of $X^T X + \lambda I_p$ are $\gamma_i + \lambda$ for $i = 1, 2, ..., p$.

So, the trace of $\text{Cov}(\hat{w}_{ridge})$ can be computed as:

$$\text{tr}[\text{Cov}(\hat{w}_{ridge})] = \sigma^2 \sum_{i=1}^{p} \frac{1}{\gamma_i + \lambda}$$

OLS Variance:

Now, let's compute the trace of the covariance matrix for the OLS estimator, $\text{tr}[\text{Cov}(\hat{w}_{OLS})]$.

Given that the OLS estimator is $\hat{w}_{OLS} = (X^T X)^{-1} X^T y$, the covariance matrix can be derived as:

$$\text{Cov}(\hat{w}_{OLS}) = \sigma^2 (X^T X)^{-1}$$

The trace of this covariance matrix is the sum of reciprocals of the eigenvalues of $X^T X$, which is:

$$\text{tr}[\text{Cov}(\hat{w}_{OLS})] = \sigma^2 \sum_{i=1}^{p} \frac{1}{\gamma_i}$$

Comparison:
We have:

$$\text{tr}[\text{Cov}(\hat{w}_{ridge})] = \sigma^2 \sum_{i=1}^{p} \frac{1}{\gamma_i + \lambda}$$

$$\text{tr}[\text{Cov}(\hat{w}_{OLS})] = \sigma^2 \sum_{i=1}^{p} \frac{1}{\gamma_i}$$

Since γ_i are eigenvalues of $X^T X$, and $\gamma_i + \lambda > \gamma_i$ for all i, it follows that:

$$\text{tr}[\text{Cov}(\hat{w}_{ridge})] < \text{tr}[\text{Cov}(\hat{w}_{OLS})]$$

This completes part (c), showing that the trace of the covariance matrix for Ridge regression is less than the trace of the covariance matrix for OLS regression.

2 Question - 2

2.1 part - a

Implementing Batch Gradient Descent

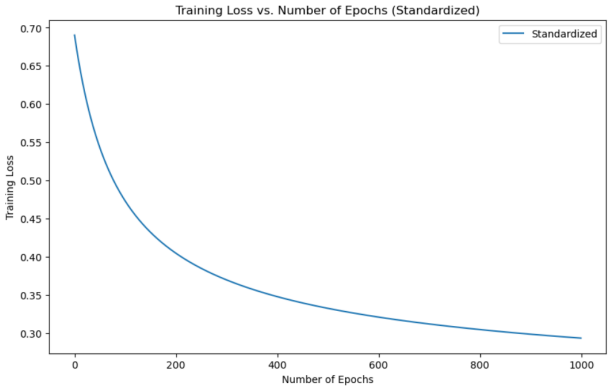

Figure 1: Batch gradient Descent (Standardized)

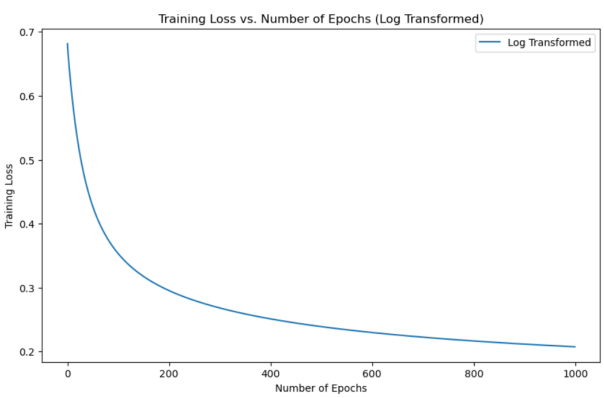

Figure 2: Batch gradient Descent (Log transformed)

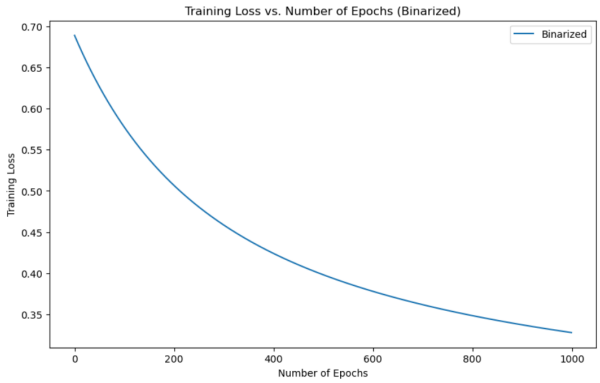

Figure 3: Batch gradient Descent (Binarized)

2.2 part - b

Implementing Stochastic Gradient Descent

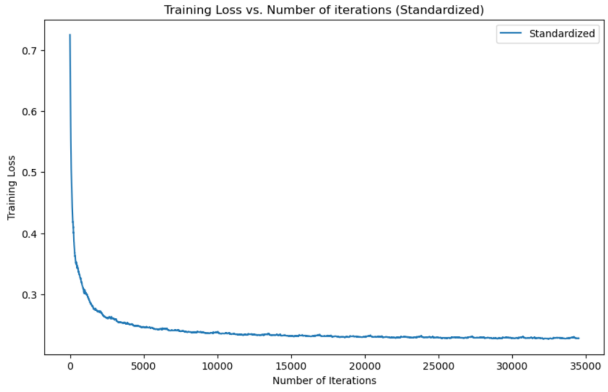

Figure 4: Stochastic Gradient Descent (Standardized)

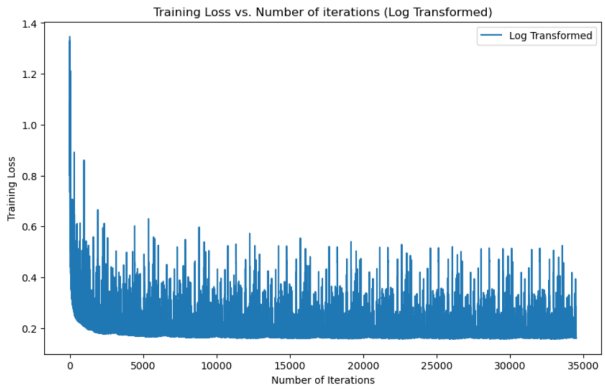

Figure 5: Stochastic Gradient Descent (Log transformed)

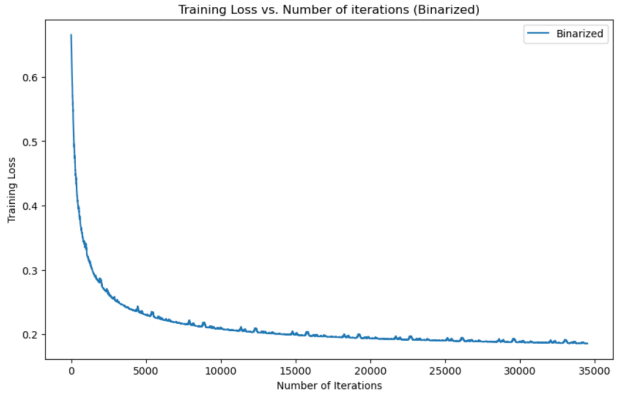

Figure 6: Stochastic Gradient Descent (Binarized)

2.3 part - c

Implementing Stochastic Gradient Descent with variable Learning Rate

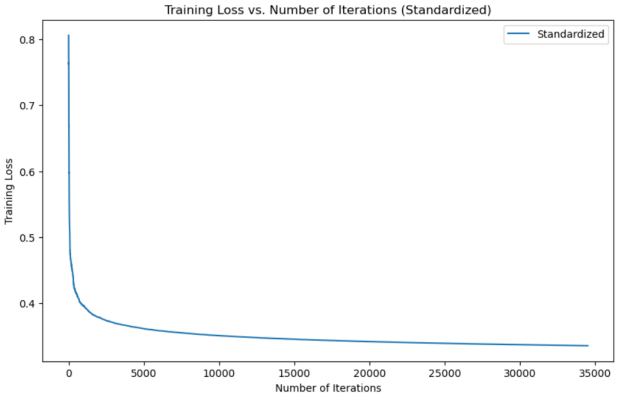

Figure 7: SGD with variable learning Rate (Standardized)

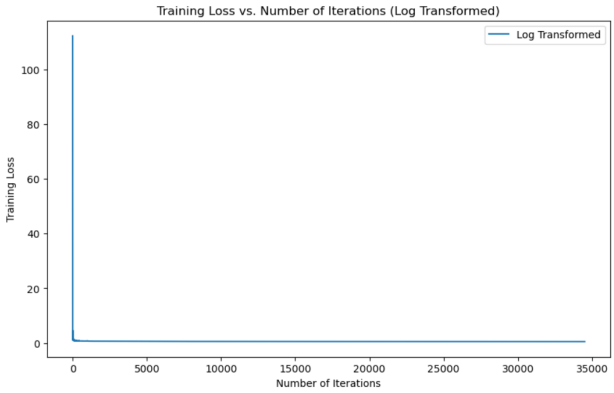

Figure 8: SGD with variable learning Rate (Log transformed)

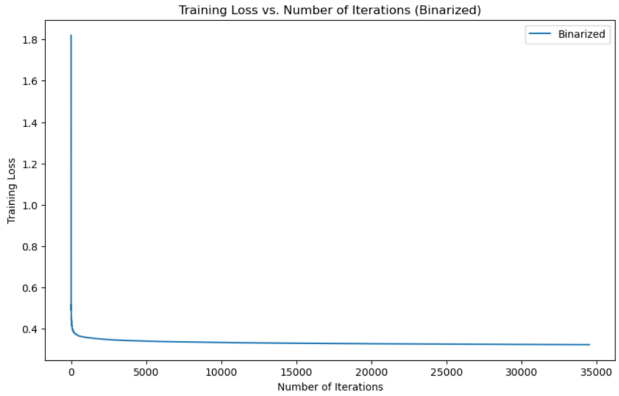

Figure 9: SGD with variable learning Rate (Binarized)

2.4 part - d

Here I am taking the Regularized parameter (lambda) = 0.1.

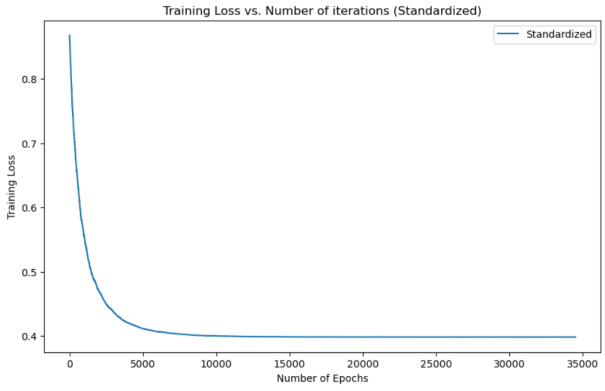

Figure 10: Regularised SGD with Regularised parameter = 0.1

3 Question - 3

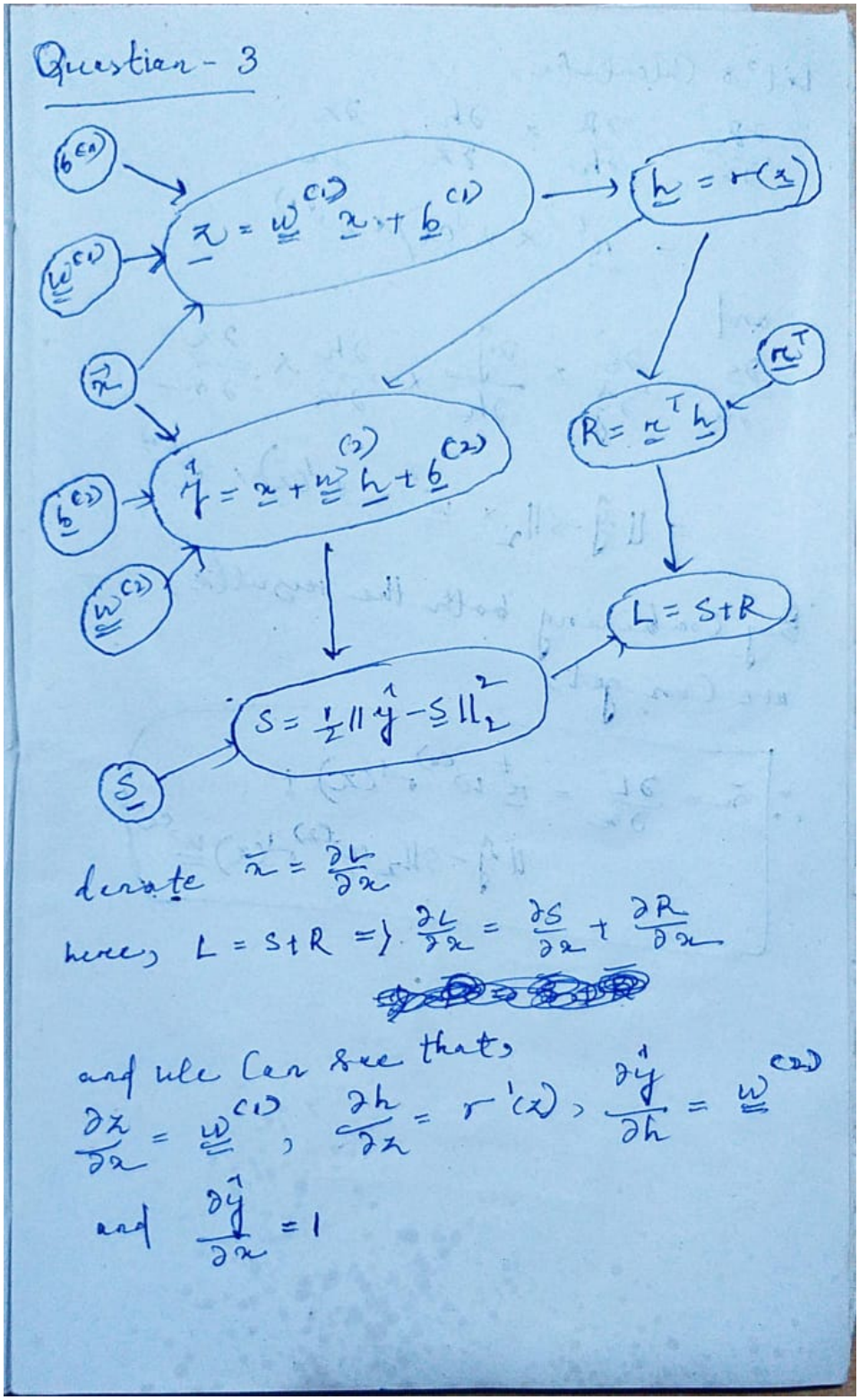

Figure 11: Question 3 (page - 1)

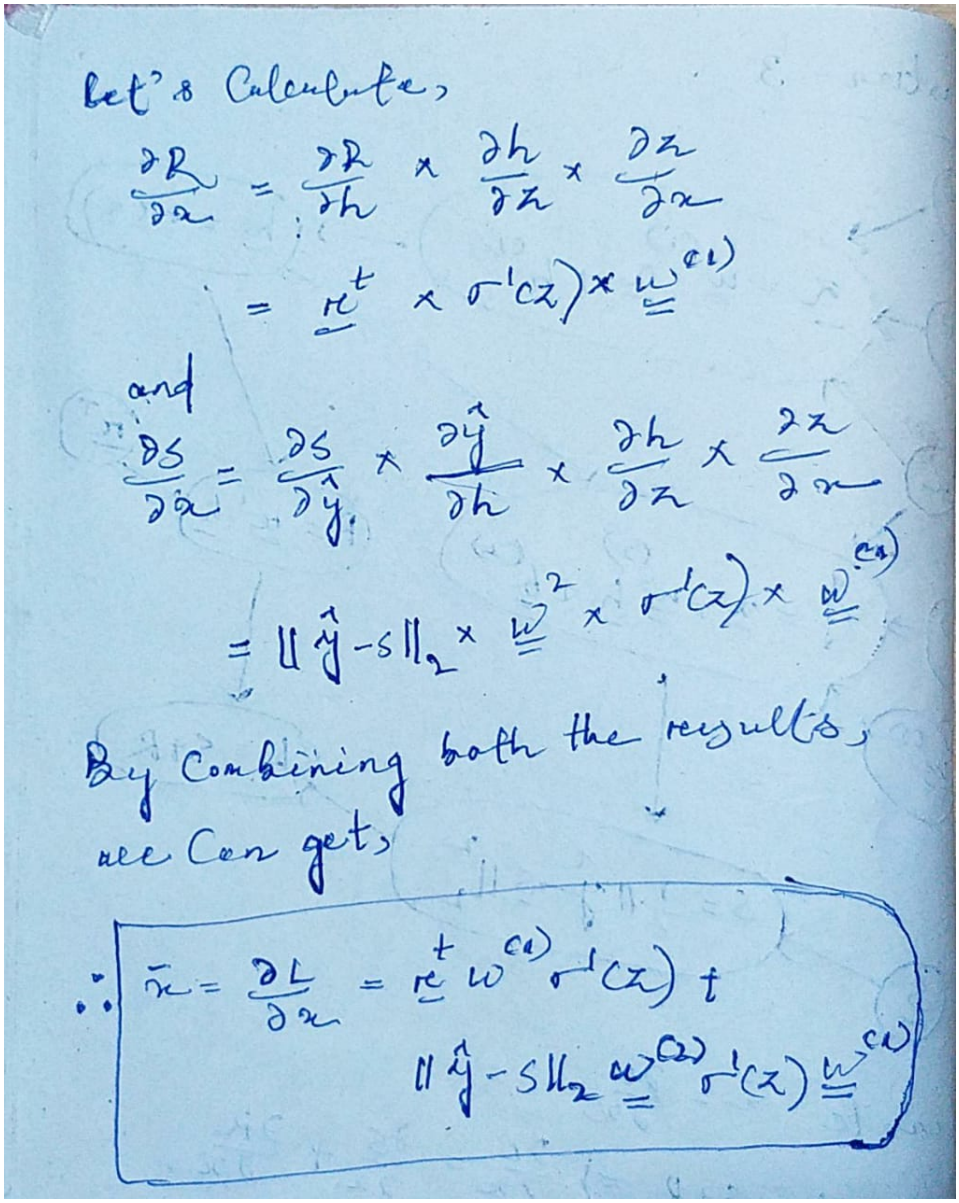

Figure 12: Question 3 (page - 2)

4 Question - 4

4.1 Part - 1

Training accuracy is noisy because we are using mini batch gradient descent. With mini-batch gradient descent, we're making updates based on only a portion of the data in each iteration. This means that our path towards optimizing parameters isn't always straightforward because each batch is making its own greedy decision. We might encounter bumps along the road, but over time, with enough iterations, we tend to move closer to the global minimum.

4.2 Part - 2

At last I am getting the test accuracy of 92.09 percent.

4.3 Part - 3

This single-layer neural network has (784*10) 7840 + 10 which is 7850 parameters.

4.4 Part - 4

Consider the training process as a journey through our dataset. Each pass over the training data is what we call an epoch. With our training dataset consisting of 60,000 samples and a batch size of 100, we perform a total of (60000/100) 600 updates (iterations) in each epoch.

In our specific case, our loop runs 2000 times, and within each loop iteration, we cover approximately 600 updates, corresponding to one epoch. So, by the time our loop finishes, we've completed roughly (2000/600) 3.33 epochs. Since we can't have a fraction of an epoch practically, this means we've completed 3 full epochs, and we've partially covered the training data in the remaining updates. It's like making several laps around a track – we've completed a few full laps and made progress beyond that before reaching the finish line.

4.5 Part - 5

Now with two layers, the classification accuracy is 94.60 percent, which is greater than the single layer network.

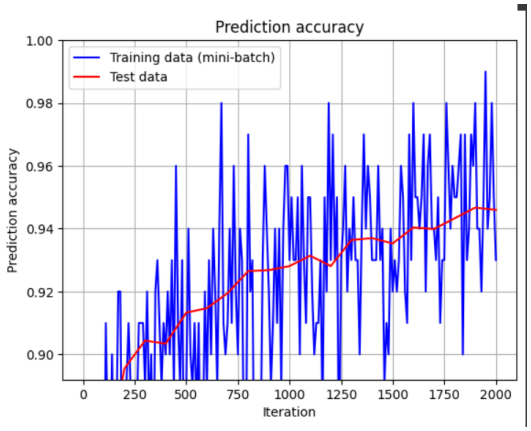

Figure 13: With 2 layers

4.6 Part - 6

With hidden units (U) = 10 , test accuracy – 92.26 percent

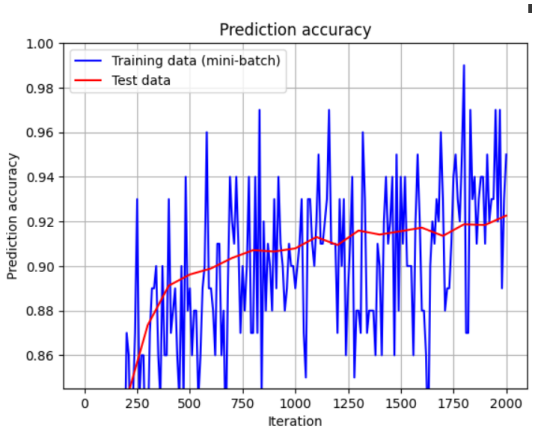

Figure 14: With Hidden layer having 10 neurons

With hidden units (U) = 471 , test accuracy – 94.70 percent

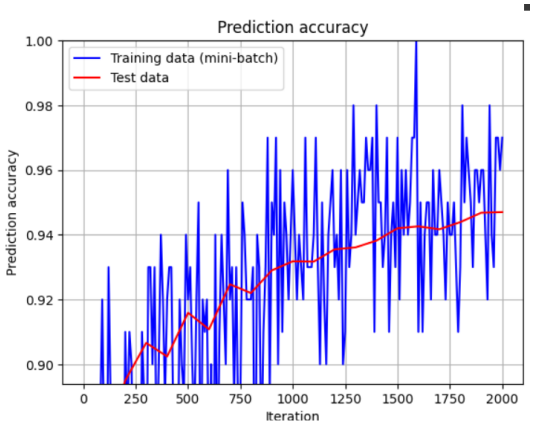

Figure 15: With Hidden layer having 471 neurons

With hidden units (U) = 743 , test accuracy – 95.12 percent

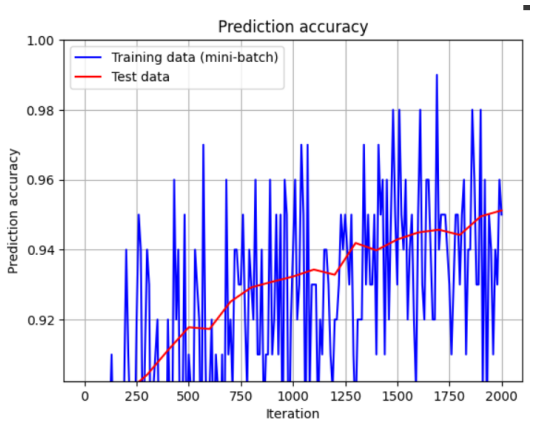

Figure 16: With Hidden layer having 743 neurons

The test accuracy is increasing with increase in no of hidden layer neurons.

4.7 Part - 7

If we initialise the weights as zeros then the test accuracy we are getting is 81 . 1 percent which is very less than what we have got previously.

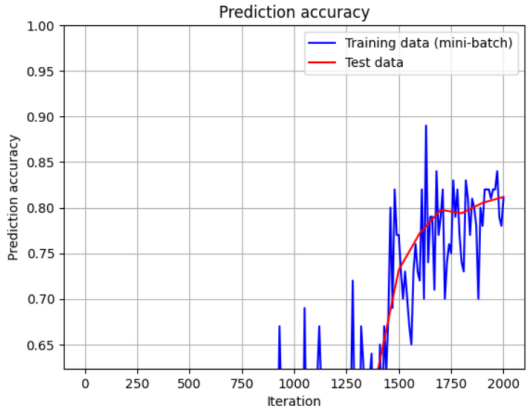

Figure 17: Initialisation with zeros

4.8 Part - 8

For five layers, the test accuracy is 86.61 percent. It is less than the two-neural net with random parameters.

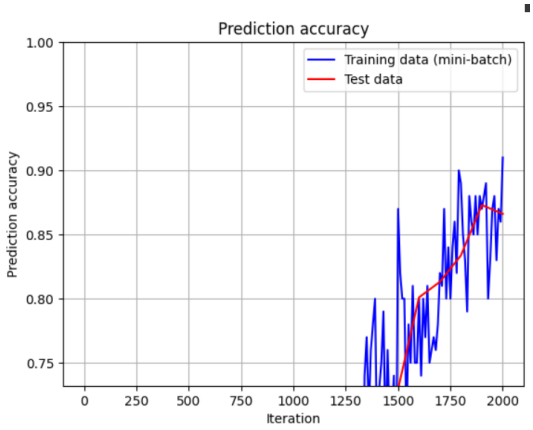

Figure 18: With Five layers

4.9 Part - 9

With ReLU activation instead of sigmoid on the hidden layers, the test accuracy is 97.30 percent, which is more than what was obtained with sigmoid.

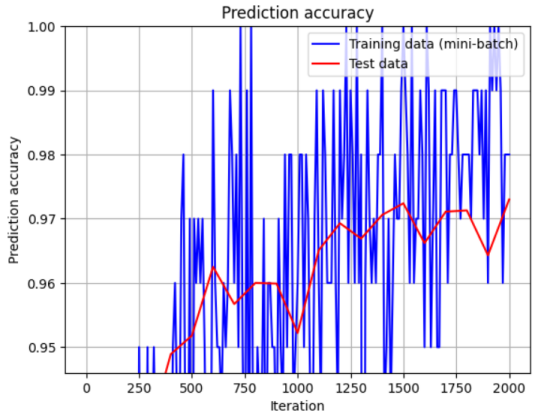

Figure 19: ReLU activation

4.10 Part - 10

When parameters are initialized to zeros, and sigmoid activation is used, the test accuracy is 11.35 percent.

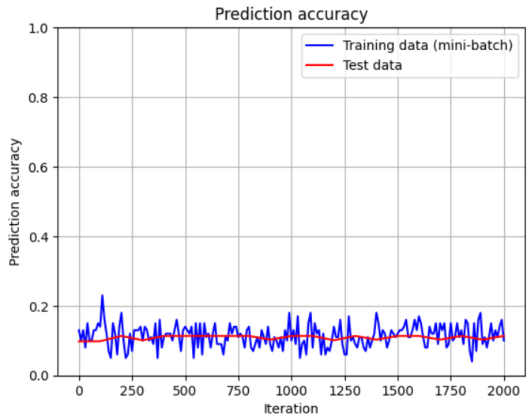

Figure 20: Zeros Initialisation and Sigmoid Activation

4.11 Part - 11

I am not finding any NAN values even after running the programme for 6000 iterations.

4.12 Part - 12

Input layer size: 784 + 1 (bias) = 785
Hidden layer 1 size: 200 + 1 (bias) = 201
Hidden layer 2 size: 100 + 1 (bias) = 101
Hidden layer 3 size: 60 + 1 (bias) = 61
Hidden layer 4 size: 30 + 1 (bias) = 31
Output layer = 10
Total parameters = 785*200 + 201*100 + 101*60 + 61*30 + 31*10 = 1,85,300

4.13 Part - 13

With the utilization of optim.Adam coupled with a learning rate of 0.003, ReLU activations, random weight initialization, and bias initialization set to 0.1, the test accuracy impressively reaches 97.45 percent. This outperforms the results obtained with optim.SGD and ReLU activations, even with random weights, and a learning rate of 0.5.Two critical modifications contribute to this significant enhancement. Firstly, the adoption of a lower learning rate, which, when combined with Adam, allows for more meticulous adjustments based on the historical gradient behavior. Consequently, despite the reduced learning rate, Adam optimally fine-tunes the updates, facilitating more efficient progress in the same direction. This refined adjustment mechanism effectively boosts the overall accuracy.

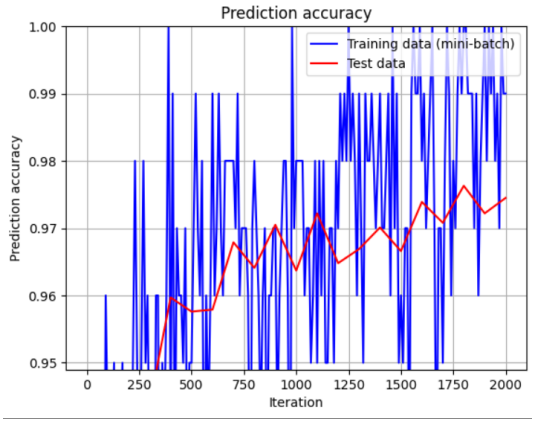

Figure 21: Adam optimizer with lower learning rate (2000 iterations)

4.14 Part - 14

With 10000 iterations I am getting the test accuracy at the 10000 step as 97.84 percent how ever I have touched an accuracy of greater than 98 percent at many iteration steps . So , by increasing the iteration number we are getting a slightly high accuracy (not significantly high).

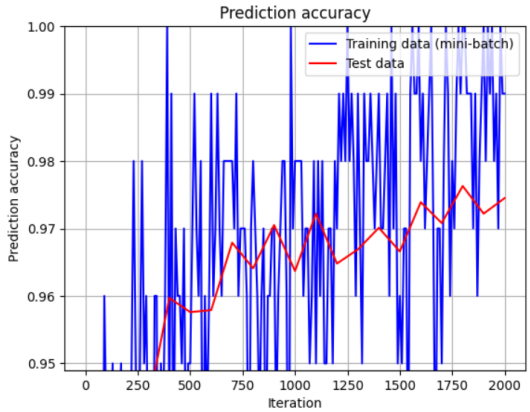

Figure 22: Adam optimizer with lower learning rate (10000 iterations)

5 Question - 5

5.1 Part - 1

Total number of hidden units in 3rd layer Q3 is (12*7*7) 588. U3flat in the given code will be the total number of hidden units = 588.

5.2 Part - 2

With five layers (three convolutional layers, two dense layers) and size of last 2 layers being 200 and 100, Test accuracy is 98.67 percent.

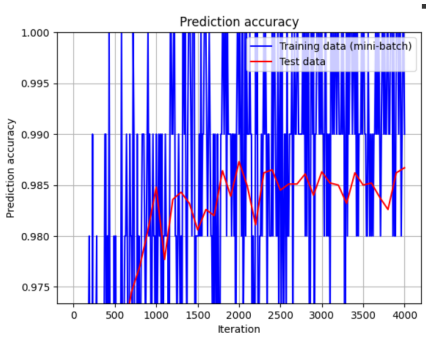

Figure 23: With five layers

5.3 Part - 3

When using the adjusted learning rate, the final test accuracy at the end of 6000 iterations is 98.97 percent.

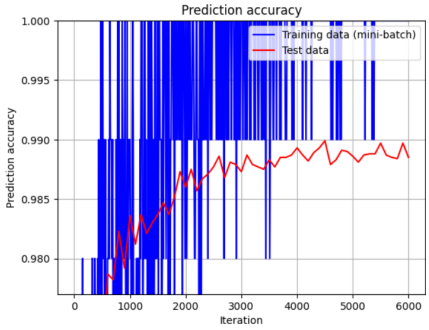

Figure 24: Using adjusted learning rate

5.4 Part - 4

It's evident that the training accuracy has hit the ceiling at 100 percent, and despite further iterations, it doesn't seem to budge. Additionally, the test accuracy has plateaued and isn't showing any signs of improvement. These observations suggest that our model is beginning to overfit the data. This phenomenon wasn't as pronounced in the previous neural network

without CNN. However, with the added complexity of convolution and dense layers, our neural network appears to be succumbing to overfitting.

5.5 Part - 5

Yes,I managed to get over 99 percent prediction accuracy on test data using the adjusted learning rate and with 200 and 100 layers.

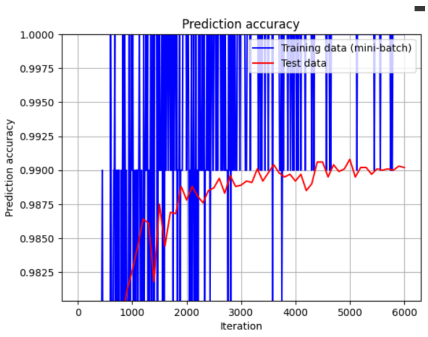

Figure 25: Getting 99 percent accuracy