Name - Sagar Suman

Roll No. 2019197

ML Assignment 1 Report

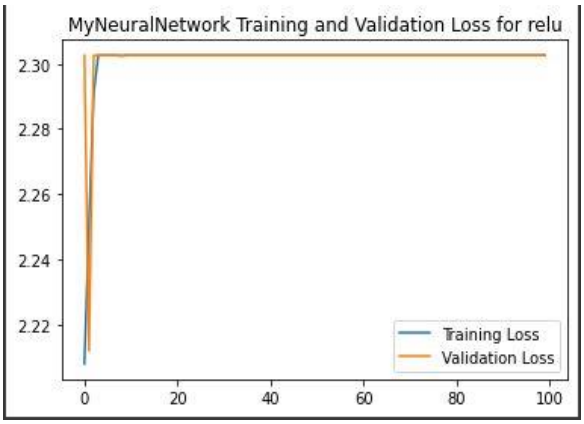
Programming Question 1 – Backpropagation

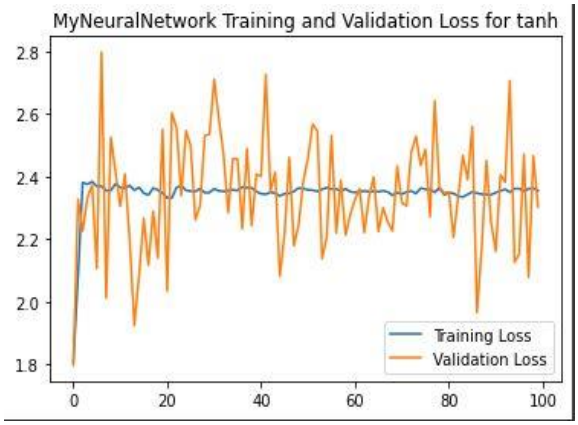
For part 1 to 4,

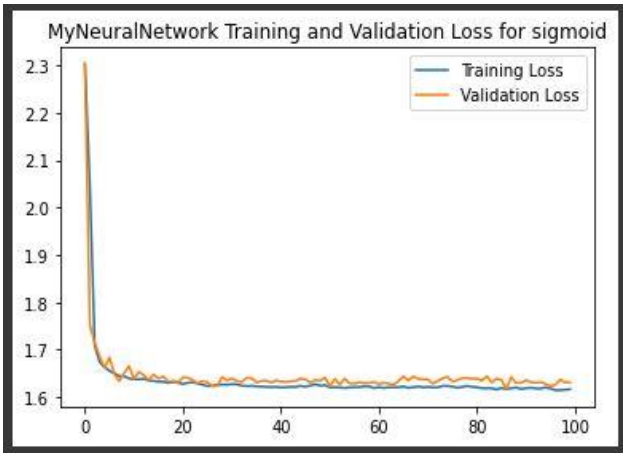
Refer to the code attached for function implementations.

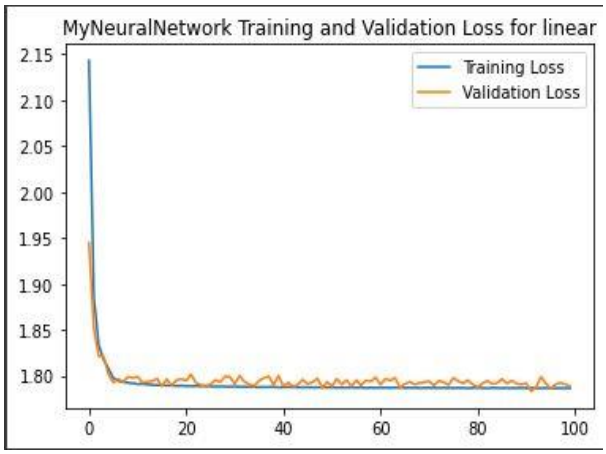
For part 5 –

Training loss vs epoch curve and validation loss. Note that we have trained the Neural Network with linear activation with 0.001 learning rate (As it was giving nan values on 0.1 and 0.01 learning rate) and rest are trained with 0.1 learning rate and batch size = 1.









On testing on test set we get –

```
MyNeuralNetwork Accuracy after 50 epoch for  relu : 0.09683333333333333
MyNeuralNetwork Accuracy after 100 epoch for relu : 0.09683333333333333
```

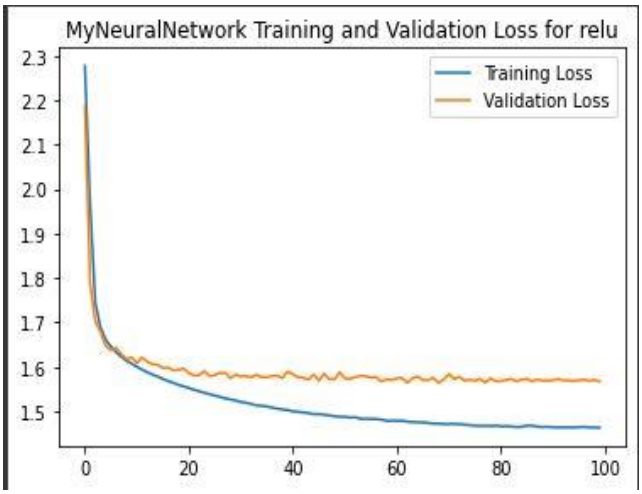
```
MyNeuralNetwork Accuracy after 50 epoch for  tanh : 0.07166666666666667
MyNeuralNetwork Accuracy after 100 epoch for tanh : 0.26033333333333336
```

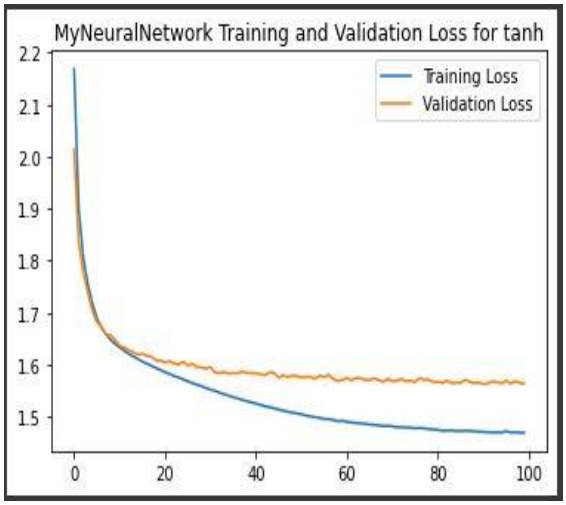
```
MyNeuralNetwork Accuracy after 50 epoch for  sigmoid : 0.8686666666666667
MyNeuralNetwork Accuracy after 100 epoch for sigmoid : 0.879
```

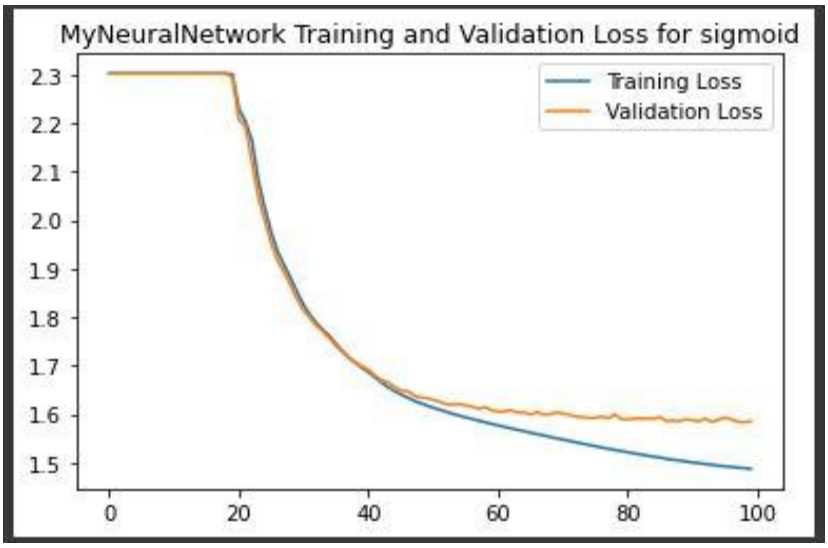
```
MyNeuralNetwork Accuracy after 50 epoch for  linear : 0.8196666666666667
MyNeuralNetwork Accuracy after 100 epoch for linear : 0.8183333333333334
```

We can see tanh and relu are performing very poorly on test set, around 26% and10% accuracy. And moreover if we look at their training vs validation loss curve, we can see that they are sort of straight line. Linear and sigmoid are performing good. One reason that I think, could be possible is due to large learning rate. Because of large learning rate relu and tanh are performing poorly.

So I decided to train the relu, tanh and sigmoid with **0.001 learning** and rate. And found the following train vs validation curves –







And following accuracy results –

```
MyNeuralNetwork Accuracy after 50 epoch for  relu : 0.952
MyNeuralNetwork Accuracy after 100 epoch for relu : 0.978
```

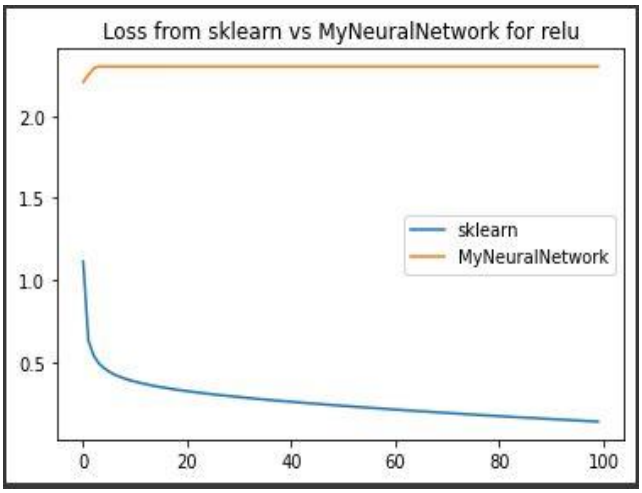
```
MyNeuralNetwork Accuracy after 50 epoch for  sigmoid : 0.8945
MyNeuralNetwork Accuracy after 100 epoch for sigmoid : 0.9698333333333333
```

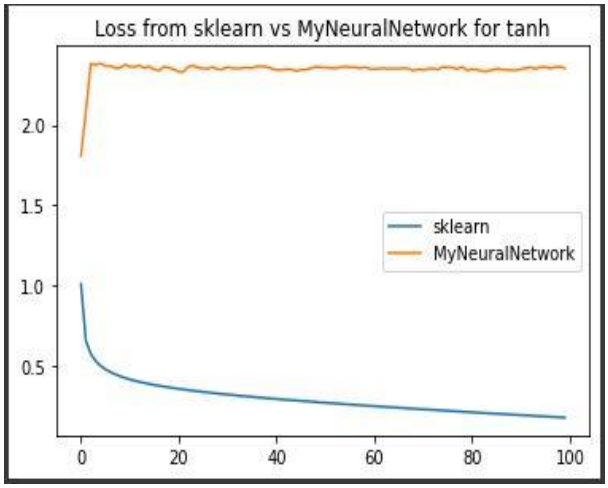
```
MyNeuralNetwork Accuracy after 50 epoch for  tanh : 0.9575
MyNeuralNetwork Accuracy after 100 epoch for tanh : 0.9733333333333334
```

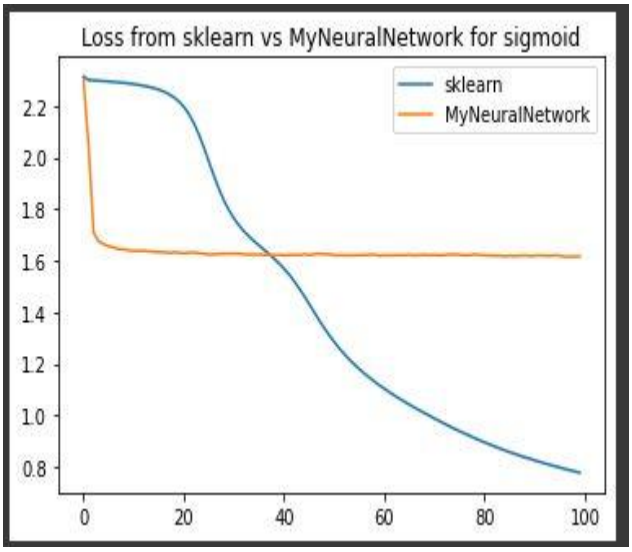
And we can see that accuracy has improved drastically for all models.

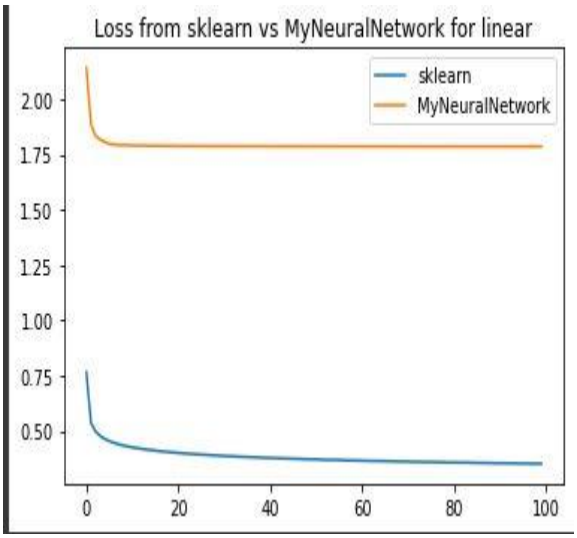
For Part 6 –

Note that these results are for batch size = 1 and 0.1 learning rate for sigmoid,relu and tanh and 0.001 for linear activation for MyNeuralNetwork. Sklearn has 0.001 learning rate and batchsize = 32 for all models.









On testing on test set we get –

```
MyNeuralNetwork Accuracy after 50 epoch for  relu : 0.09683333333333333
MyNeuralNetwork Accuracy after 100 epoch for relu : 0.09683333333333333
Sklearn Accuracy after 50 epoch for Relu: 0.9121666666666667
Sklearn Accuracy after 100 epoch for Relu: 0.9448333333333333
```

```
MyNeuralNetwork Accuracy after 50 epoch for  tanh : 0.07166666666666667
MyNeuralNetwork Accuracy after 100 epoch for tanh : 0.26033333333333336
Sklearn Accuracy after 50 epoch for tanh: 0.9038333333333334
Sklearn Accuracy after 100 epoch for tanh: 0.9328333333333333
```

```
MyNeuralNetwork Accuracy after 50 epoch for  sigmoid : 0.8686666666666667
MyNeuralNetwork Accuracy after 100 epoch for sigmoid : 0.879
Sklearn Accuracy after 50 epoch for Sigmoid: 0.553
Sklearn Accuracy after 100 epoch for Sigmoid: 0.7265
```

```
MyNeuralNetwork Accuracy after 50 epoch for  linear : 0.8196666666666667
MyNeuralNetwork Accuracy after 100 epoch for linear : 0.8183333333333334
Sklearn Accuracy after 50 epoch for linear: 0.8666666666666667
Sklearn Accuracy after 100 epoch for linear: 0.8711666666666666
```

Overall wise, **relu** is performing best (accuracy 94%) with **sklearrn i**mplementation (leaning rate = 0.001).

With my implementation with 0.1 learning rate **sigmoid** is performing best with 87.9% accuracy. However, when my model is trained with 0.001 my model is giving 97.8% accuracy with **relu.**

Because my model(learning rate = 0.001) is trained with batch size =1 it is giving better accuracy as compared to sklearn (learning rate = 0.001) with batch size = 32.

ML → Assignment - 2
Name → Sagar Suman
Roll → 2019197

THEORY QUESTION

(1.) KL divergence is $KL(P||Q) = \sum_{z} P(z) \log \frac{P(z)}{Q(z)}$

Assumption → If we assume
$P(y_i|x_i)$ is n-dimensional vector for each input, such that it is one-hot encoded.

Then, cross entropy loss for multiclass problem is same as KL divergence. → Proof ↓

As we know, cross-entropy,

$H(P, Q) = H(P) + D_{KL}(P||Q)$

as, $P(y_i|x_i)$ → is one-hot encoded → $[0, 0, \ldots, 1, \ldots, 0]$

$H(P(y|x)) = -\sum_{x \in X} x \log P(y_i|x_i) = -\sum_{x_i : P_i(x) \neq 1} 0 \log 0 + 1 \log 1$

so, $H(P(y|x)) = 0$

and Hence,

$H(P, Q) = D_{KL}(P||Q) = -\mathbb{E}_{x \sim P} \log Q(x)$