Home Work 6

Saiprakash Nalubolu

B01037579

Q1. A) using Linear function in the output layer: θ(s) = s

```
Choose an activation function for the output layer:
1: Linear
2: Hyperbolic Tangent (tanh)
3: Sign
4: Sigmoid
Enter the number of your choice: 1
Gradients from backpropagation:
w1: [[-0.16365331 -0.08182666]
 [-0.16365331 -0.08182666]]
b1: [-0.16365331 -0.16365331]
w2: [[-0.52431873 -0.52431873]]
b2: [-0.94540028]
```

B) Calculating Gradient numerically through perturbing each weight one at a time by 0.0001

```
Numerical gradient check:
w1 numerical gradient: [[-0.16365331 -0.08182666]
 [-0.16365331 -0.08182666]]
b1 numerical gradient: [-0.16365331 -0.16365331]
w2 numerical gradient: [[-0.52431873 -0.52431873]]
b2 numerical gradient: [-0.94540028]
```

Repeating the same for tanh:

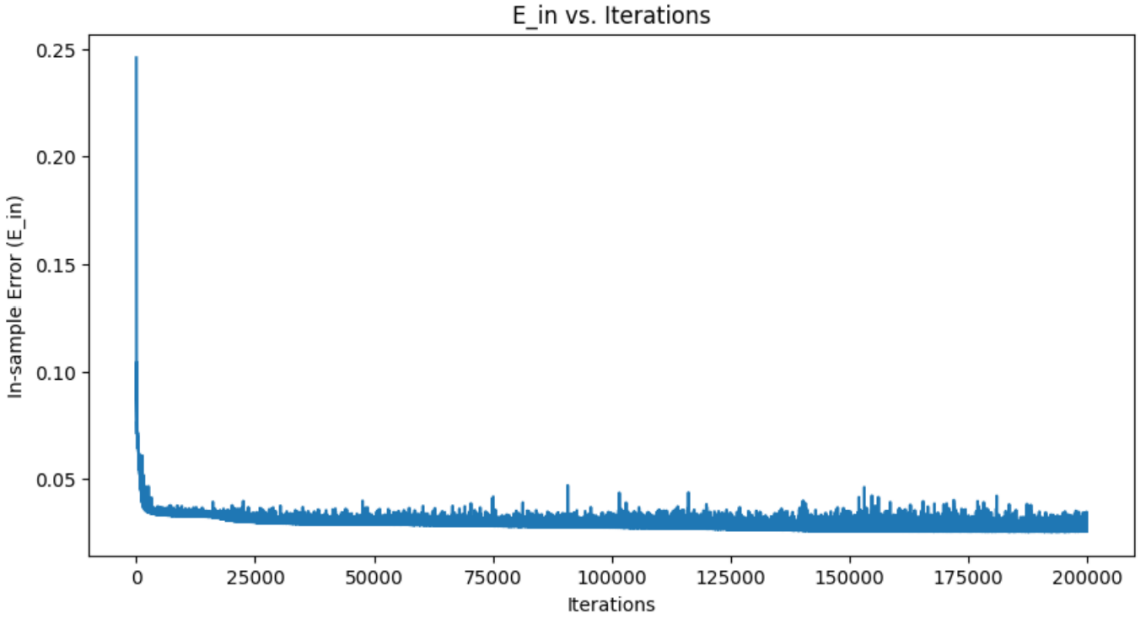
a)

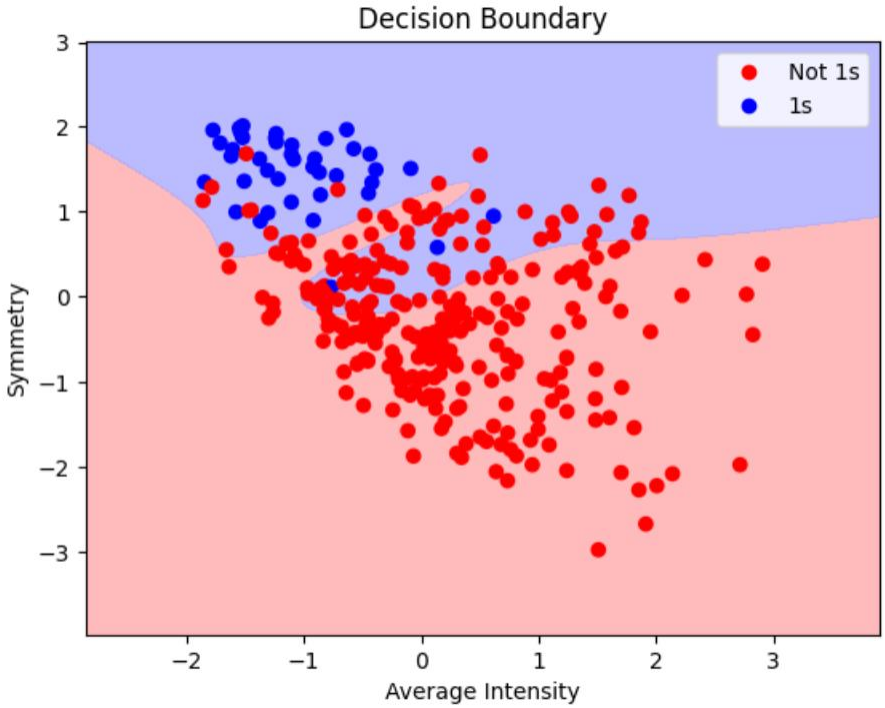
```
Choose an activation function for the output layer:
1: Linear
2: Hyperbolic Tangent (tanh)
3: Sign
4: Sigmoid
Enter the number of your choice: 2
Gradients from backpropagation:
w1: [[-0.13709607 -0.06854804]
 [-0.13709607 -0.06854804]]
b1: [-0.13709607 -0.13709607]
w2: [[-0.43923364 -0.43923364]]
b2: [-0.79198315]
```

b)

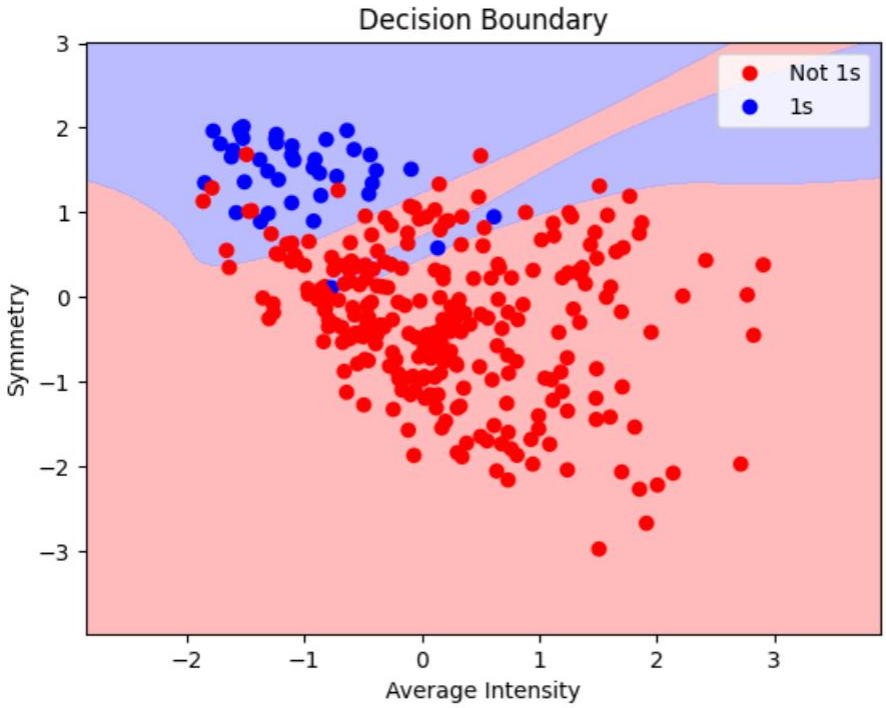
```
Numerical gradient check:
w1 numerical gradient: [[-0.13709608 -0.06854804]
 [-0.13709608 -0.06854804]]
b1 numerical gradient: [-0.13709608 -0.13709608]
w2 numerical gradient: [[-0.43923364 -0.43923364]]
b2 numerical gradient: [-0.79198316]
```

Q2. A)

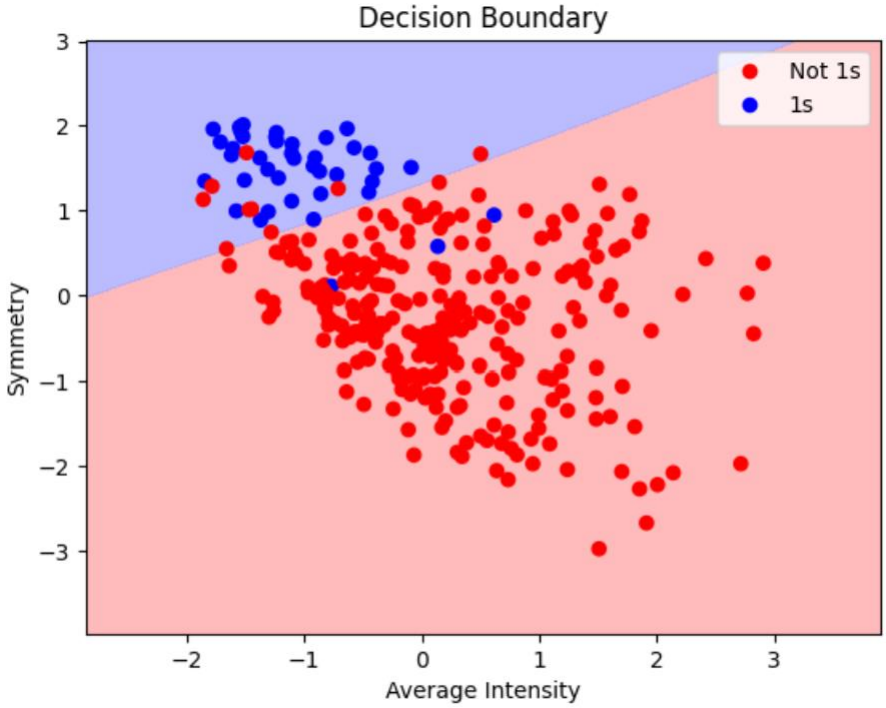




Decision boundary after obtaining classifier after 2*10^6 iterations.


Decision Boundary
Not 1s
1s
Symmetry
Average Intensity
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

b)


Decision Boundary
Not 1s
1s
Symmetry
Average Intensity
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

c)

Q3.

i, `E_test after polynomial transformation (Test Error): 0.03537735849056604`(uploaded from HomeWork-3)

ii, `Test error (E_test) for k-NN with k=31: 0.03132141416857104`(from Homework-5)

iii, `Final Test Error: 0.15603467437208268` (Based on early stop neural networks)

The First two test errors are very similar, but the Test error in the neural networks is very high when compared to the other two.
There might be many reasons for this,

Complexity of the Model: Neural networks, especially with multiple hidden layers or many hidden units, can be quite complex. With more parameters to learn, they are powerful but also more prone to overfitting unless regularized properly or provided with a lot of data. We have taken steps to regularize data, early stopping is one of the techniques to prevent overfitting.

Early Stopping: While early stopping is a form of regularization to prevent overfitting, it may also stop training before the network has sufficiently learned the patterns in the data. This is possible if the validation set is not representative of the overall distribution.

Data Preprocessing and Feature Engineering: Neural networks can be sensitive to the scale and distribution of input data. If the data isn't normalized or standardized appropriately, the network might not learn effectively. *In contrast*, methods like k-NN can be less sensitive to this because they rely on distances which can be scale-invariant depending on the distance metric used.

Stochasticity: The stochastic nature of neural network training, especially with stochastic gradient descent, with different runs we might get different results. The Error I got from Neural network at early stopping(3.iii) will be a different value if I would have chosen different initial values.

Comparison Fairness: Differences in data splits can cause significant variation in performance. Since we are selecting random 300 data points we are not sure on which data we training, so this might cause difference in the value of Etest for Neural networks with early stopping.

Inherent Suitability: Finally, some methods are inherently more suitable for certain types of problems due to their bias-variance characteristics. For the MNIST classification task, a simple model with well-chosen features might outperform a complex model if the extra complexity does not provide additional discriminative power.

The fact that the other methods have similar performance suggests they are either more suited to this particular problem or were better tuned during the cross-validation phase. To improve the neural network's performance, I would need to revisit the network architecture, regularization strategy, and hyperparameter settings, and ensure that the data preprocessing is appropriate for this method. Additionally, cross-validation can also be used to tune the neural network's hyperparameters.

https://colab.research.google.com/drive/1DFCN7UZbkmFivMvCQyDL5CKRkr_Q6Xmt#scrollTo=rKhae9GTIQsM