Intro To Machine Learning

Q1a:

The hypermeters randomly initialized were Weights and Bias

Weights = [[0.11671336, 0.05025692],

[-0.45470146, -0.17954758]]

Bias= [0.44845312, 0.65353984]

{Note: These are not the final values if you run the code you might get different values, because I have not used seed value}

The hyperparameters which were used for tuning were learning-rate, Regularizer and no.of epochs and after some trial and error the values finalized for these hyperparameters are as followed: -

Learning-Rate (α) = 0.01

Regularizer (λ) = 0.8

Epoch = 500

For the target values were converted to one-hot-encoding as followed: -

Y = [[1. 0.]

[0. 1.]

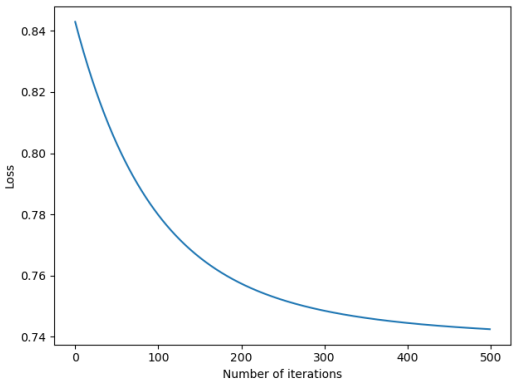
[0. 1.]

[1. 0.]]

The predicted values were compared using the argmax value to retrieve the index of the list with a max value as and if the index of largest value in the target label(1) and predicted label were same then the prediction was said to be correct for that input of dataset this was possible because of the one-hot-encoding.

Since this is a Softmax Regression model the one-hot-encoding allows us to get multiclass probabilities for the a single input of data

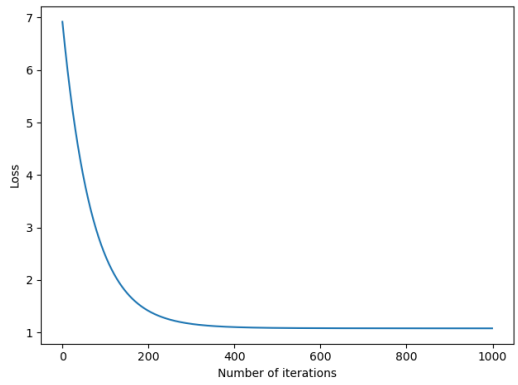
Below is the plot of loss per epoch:



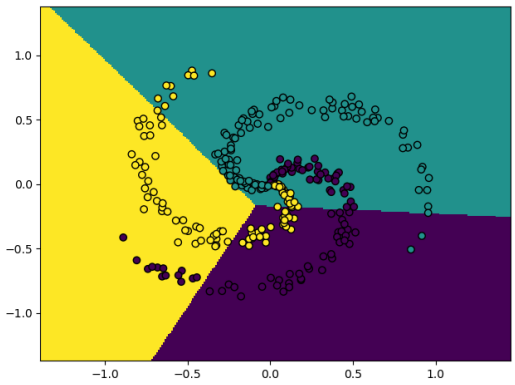
Model's Accuracy = 50%

Q1b:

Error Rate per epoch:



Decision Boundary for the data:



Model Accuracy: - 58.666666666666664

Since the dataset which was fit to the model there were some tuning required to get an optimal solution and the aforementioned accuracy is the maximum accuracy I could get

The tunings were done as followed:

Epochs were changed to 1000

Regularizer was changed to 0.7

And the one-hot-encoding was changed to 3 bits as there were three target labels to be predicted for every individual input of dataset.

Q1c.

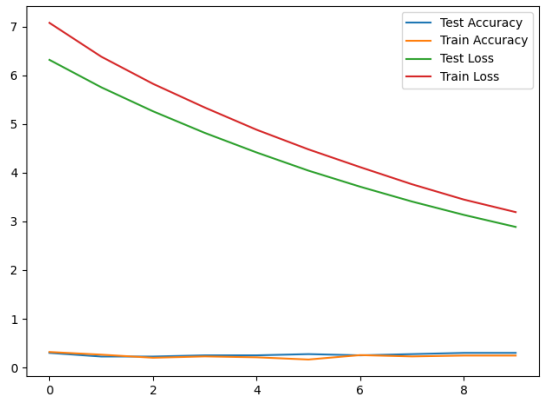
For 10 Epochs:

Test Accuracy after all the Epochs 0.6

Train Accuracy after all the Epochs 0.6272727272727273

Test Loss after all the Epochs 1.9532220441731365

Train Loss after all the Epochs 1.985143059922207



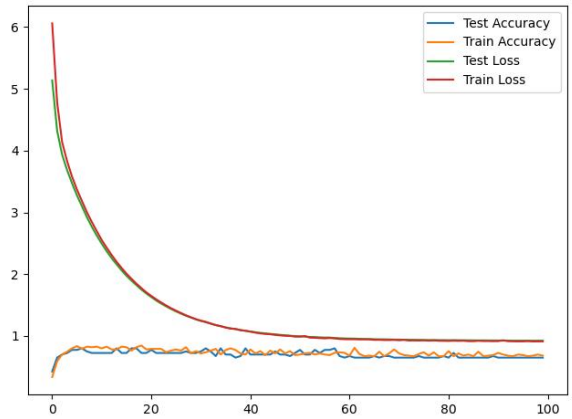
For 100 Epochs:

Test Accuracy after all the Epochs 0.65

Train Accuracy after all the Epochs 0.7818181818181819

Test Loss after all the Epochs 0.8351759759887585

Train Loss after all the Epochs 0.8197522035153589



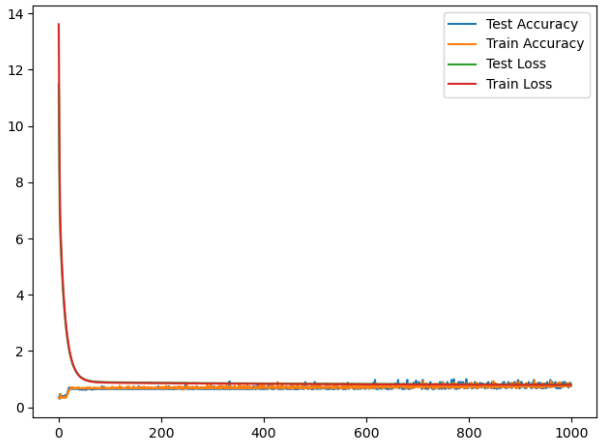
For 1000 Epochs:

Test Accuracy after all the Epochs 0.75

Train Accuracy after all the Epochs 0.8454545454545455

Test Loss after all the Epochs 0.809952854644232

Train Loss after all the Epochs 0.8050364269244771



As you can see that the model with higher number of epochs is giving more training accuracy than testing accuracy, this phenomenon is called as bias. Bias means that the model is more focused towards the input from the training set than the testing set hence for the training set the model will perform very well but for the testing set it won't perform as good. You can see this in the data recorded above that the training set accuracy is increasing much more steeply than the testing set accuracy.

This happens because of the model isn't subjected to newer data in more epochs it is trained and tuned as per the training data with more epochs the model is getting subjected to same limited amount of data over and over again.

To address this problem, we can either get more data for the model or tune the model by increasing the value of the reglarizer, since this model isn't ideal these methods will seldom work on such models. But in an ideal situation these solutions are recommended for the aforementioned problem.

Q2

A:

The maximum likelihood for each parameters are as followed:

in html True False

Spam True [0.7558139534883721, 0.5869565217391305]

Spam False [0.2441860465116279, 0.41304347826086957]

##

has emoji True False

Spam True [0.19767441860465115, 0.1473429951690821]

Spam False [0.8023255813953488, 0.8526570048309179]

##

sent to list True False

Spam True [0.06976744186046512, 0.3115942028985507]

Spam False [0.9302325581395349, 0.6884057971014492]

##

from .com True False

Spam True [0.7441860465116279, 0.2753623188405797]

Spam False [0.2558139534883721, 0.7246376811594203]

##

has my name True False

Spam True [0.3488372093023256, 0.6014492753623188]

Spam False [0.6511627906976745, 0.39855072463768115]

##

has sig True False

Spam True [0.6627906976744186, 0.32367149758454106]

Spam False [0.3372093023255814, 0.6763285024154589]

##

sentences True False

Spam True [Mean: 3.9767441860465116, Standard Deviation: 3.720389399675499]

Spam False [Mean: 6.190821256038648, Standard Deviation: 6.400785315876665]

##

words True False

Spam True [Mean: 68.83720930232558, Standard Deviation: 79.34559221200647]

Spam False [Mean: 70.77053140096618, Standard Deviation: 912.7661847417676]

##

B:

On a Threshold of P(Y) = 0.5 we get the following results: -

Correct Classifications: 67.5

Misclassifications rate: 32.49999999999999

C:

Maximum Accuracy Obtained after randomly ignoring few features 80.0

This features ignored here were 0(in html), 1(has emoji), 3(from.com), 7(# sentences these were randomly chosen: