Name: Vaibhav Agrawal
Roll Number: 15CE10057
Assignment 4: Neural Networks
Machine Learning (CS60050)

In this assignment you will build a multilayer neural network and classify whether a given message is SPAM or HAM (non-spam). The dataset has 5574 messages, each annotated as SPAM or HAM. The dataset can be downloaded from: https://drive.google.com/file/d/1o-ek6ZLVUnpdT4on74DTNVMKa4e_ctMU/view?usp=sh aring

Preprocessing:

1. Break each message into tokens (any sequence of characters separated by blanks, tabs, returns, dots, commas, colons and dashes can be considered as tokens)
2. Remove a standard set of English stopwords, e.g., the set available at https://gist.github.com/sebleier/554280
3. Apply Porter stemming.

Consider 80% of the dataset (randomly selected) as training data, and the rest 20% as test data. Compute the set of distinct tokens in the dataset (denoted as V). Represent each messageasa(|V| x1)vector,whereeachentry *j*i s1or0dependingonwhethertoken *j* is present in the message. This is your input representation and should be fed into the input layer. This representation is usually referred to as 'one hot encoding'. *If you cannot manage a network with all distinct tokens, you can consider the most frequent 500 tokens only.*

PART 1

Build a neural network with 1 hidden layer and perform the text classification task.
Neural network specifics:

1. No of hidden layers : 1
2. No. of neurons in hidden layer: 100
3. Non-linearity in the layer : Relu
4. Use 1 neuron in the output layer. Use a suitable threshold value, and classify a message as SPAM if the score is above threshold or HAM if it is below threshold.
5. Optimisation algorithm : Stochastic Gradient Descent (SGD)
6. Loss function : categorical cross entropy loss
7.

The function Relu is defined as : f(x) = max(0,x) It's derivative : f'(x) = 0, if x<=0
= 1, otherwise
You should define the relu function and its derivative. Do not use any inbuilt library for this.
Do a random initialisation of the weights. Use learning rate 0.1.

Implementation :
Have the following modules/functions in your code :

1. Preprocess: Use this module to preprocess the data and divide into train and test.
2. Data loader : Use this module to load all datasets and create mini batches
3. Weight initialiser : This module should initialize all weights
4. Forward pass: Define the forward() function where you do a forward pass of the neural network.
5. Backpropagation : Define a backward() function where you compute the loss and do a backward pass (backpropagation) of the neural network and update all weights.
6. Training : Implement a simple minibatch SGD loop and train your neural network, using forward and backward passes. Continue the experiment till training error becomes very low. Finally it should print the accuracy after training.
7. Test: To test the learned model weights on the test set.

PART 2
Build a neural network as follows:

- No. of hidden layers = 2
- No. of neurons in the two hidden layers should be taken as command-line arguments .
- No. of neurons in the output layer = 2

Use sigmoid function for non-linearity in this part. Do not use any inbuilt python library.
Use softmax function in the output layer. Softmax will convert the outputs of the neural network in each neuron to the probability of a message being SPAM or HAM. Based on the higher probability you will classify a message to its class. You can see this video for implementing the softmax function
Define the forward and backward passes accordingly.
The implementation should have the same functions as defined in Part 1.

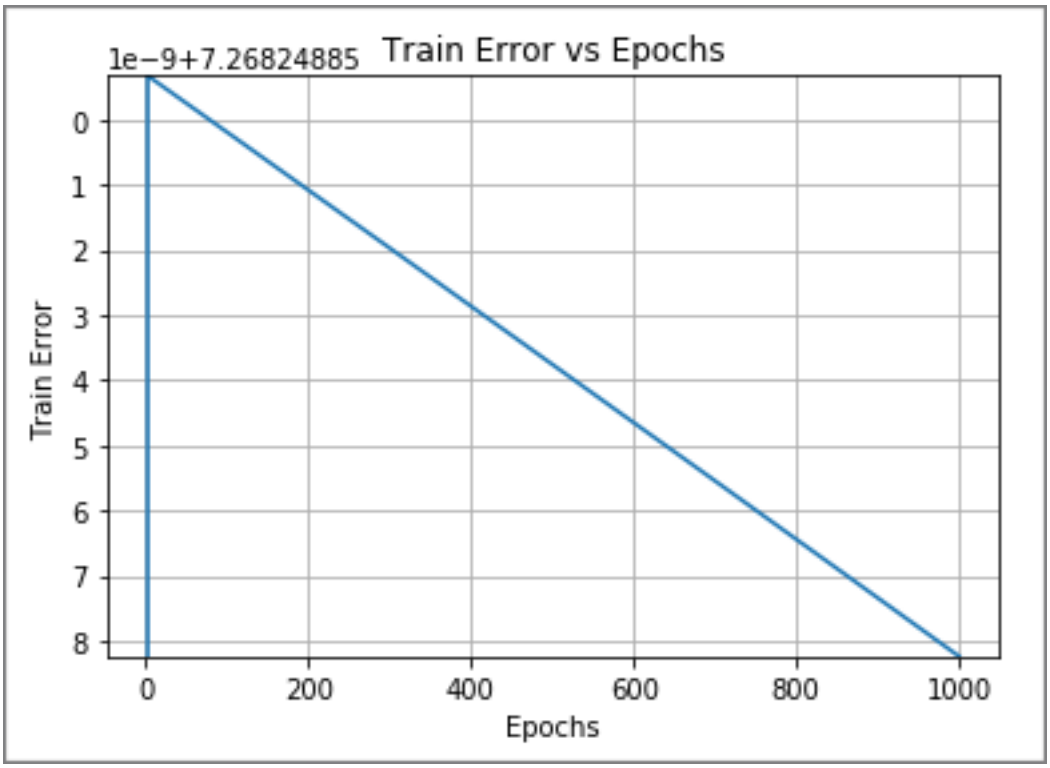
Report
For both parts, report :

1. Training set error over number of epochs
2. Test set errors over number of epochs
3. Final Test set accuracy

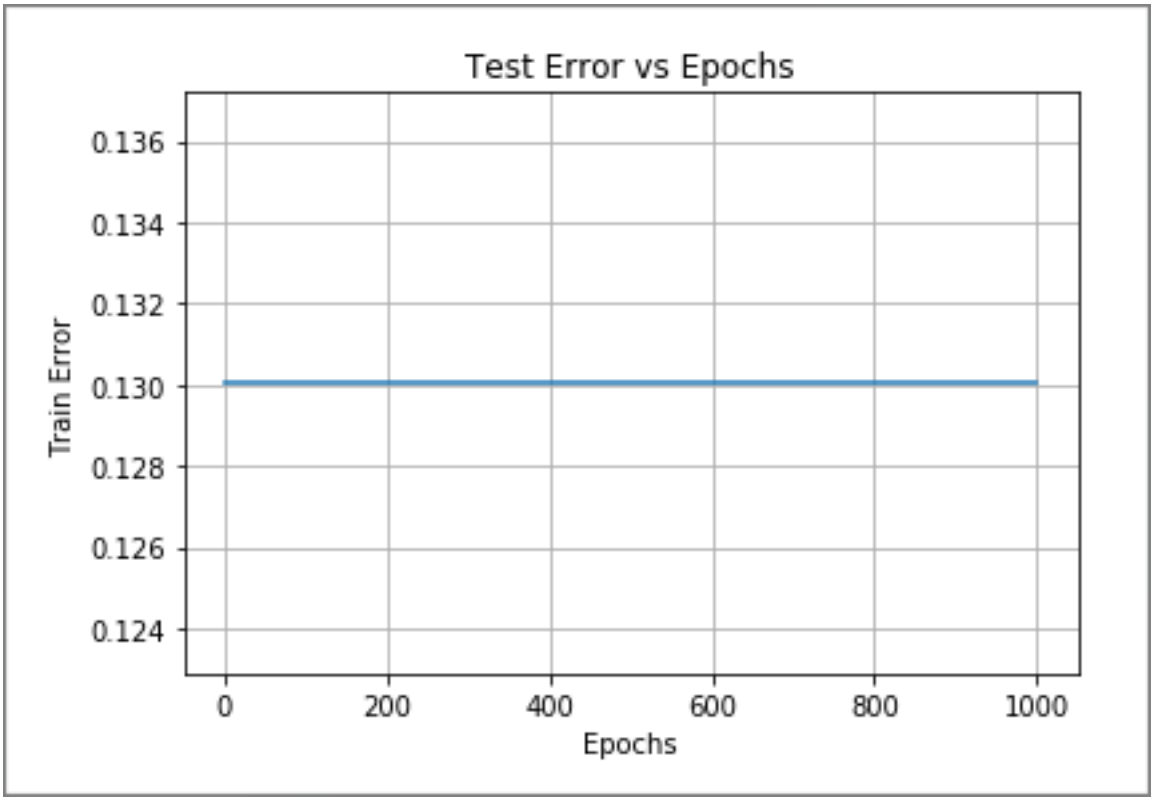
Report:

Part 1:

1. Training set error over number of epochs



2. Test set errors over number of epochs

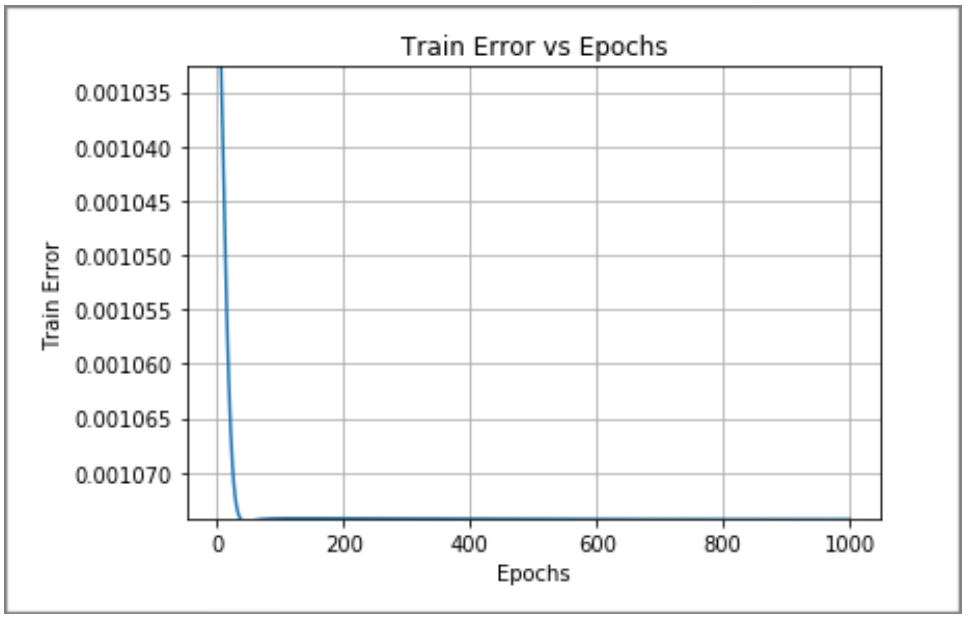


3. Final Test set accuracy = 86.99%

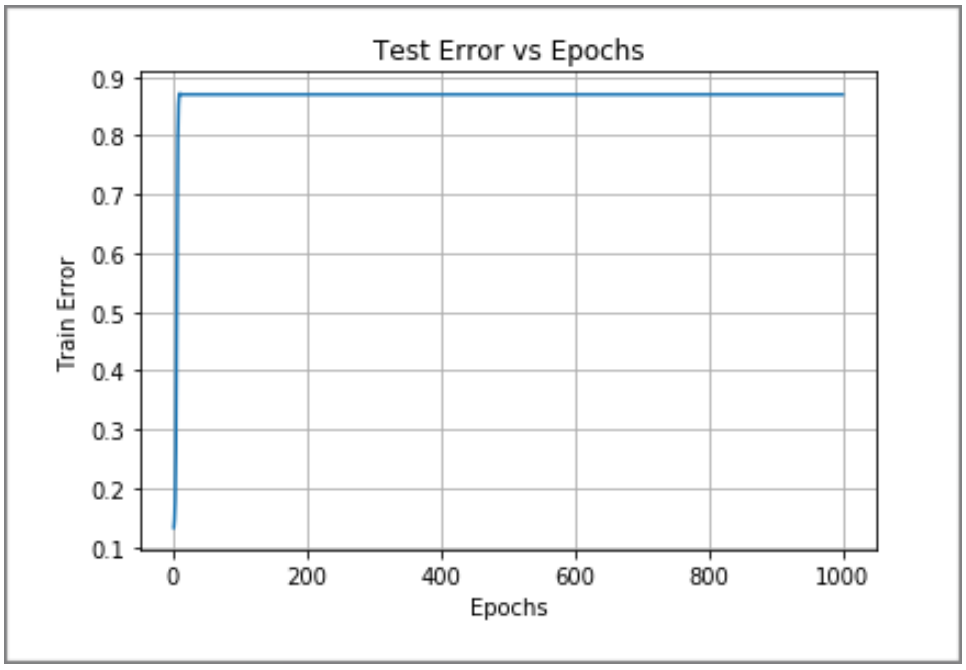
Part 2:

Considering:
The number of nodes in hidden layer 1: 10
The number of nodes in hidden layer 2: 5

1. Training set error over number of epochs



2. Test set errors over number of epochs



3. Final Test set accuracy = 13%