Ceng463 Homework2 Report

Samet Umut Yiğitoğlu 260201056

May 16, 2022

1 Introduction

In the homework, I have used 2 hidden layers with a size of 200 and 150 neurons respectively, and with a learning rate of 0.05. First, I have created a set that contains all the words in the benefits review, side effects review, and comments review. I create a dictionary for every row of data with the keys of this set. These dictionary values are my inputs. As can be understood the size of the input layer is determined by the length of the set. The output layer size is 10 since there are 10 ratings possible. And the output is one hot encoding. I have tested the code with Python 3.8. So I recommend running it with it.

2 Neural Network

2.1 What was the reason that your network learned?

The reason behind why my neural network learned is training. After the forward pass network calculates the loss and then back propagates. With back propagation, the network calculates deltas and then adjusts the weights and biases according to it.

2.2 How would you improve the accuracy more?

We can improve the accuracy by:

- Adding new features
- Increasing number of hidden layers
- Increasing number of neurons
- Adjusting learning rate
- Adding more training data
- Weight and bias initialization
- Increasing number of epochs

2.3 What were the trade-offs you face with?

Increasing the accuracy almost always comes with a cost of efficiency. At least that is what I faced in my homework. I have added more neurons to increase the accuracy. But it is very slow right now. It takes like 13 minutes on my machine. Before designing this network I had other designs in my mind but they failed because of a lack of memory. So I believe memory is another trade-off when it comes to designing accurate neural networks.

Accuracy	Loss
0.037371134020618556	13.022218112063975
0.24484536082474226	0.8791135516035978
0.24742268041237114	0.8658088741164016
0.23711340206185566	0.8646205704560774
0.24355670103092783	0.8600471064555926
0.23067010309278352	0.8668116300946304
0.24742268041237114	0.8620922851727851
0.24613402061855671	0.8606688455051412
0.24613402061855671	0.8569577811795441
0.24484536082474226	0.8601880839544623
0.24355670103092783	0.8568874591392602
0.2345360824742268	0.8640893532157123
0.2422680412371134	0.8602876268519213
0.24742268041237114	0.8594933249044993
0.2422680412371134	0.8551213432396966
0.23067010309278352	0.8582299584784022
0.21520618556701032	0.8590106821974578
0.21262886597938144	0.876769545066457
0.23711340206185566	0.857476814560712
0.24097938144329897	0.8591261709699406
0.24484536082474226	0.8567044225190522
0.23582474226804123	0.8566087410266879
0.20876288659793815	0.8731963981886586
0.22293814432989692	0.8909756208300281
0.24355670103092783	0.8543961865548975
0.24613402061855671	0.8570916237465394
0.25	0.8623856767082179
0.24355670103092783	0.8546916599797617
0.22422680412371135	0.8885255465804741
0.22551546391752578	0.8983293252136207

Table 1: Accuracy and loss results

2.4 The reasoning of why did you design your network in that way?

When I first run the network the accuracy was very low. It had like 2 hidden layers with the size of 10 neurons and the learning rate was around 0.5. The first thing I did was to increase the size of the hidden layers to 200 and 150. It did help but it was still too low. So I thought maybe our learning rate is too high so that gradient descent is overshooting. Then I reduced the learning rate to 0.05 and the accuracy improved significantly. Loss and activation functions were given in the assignment pdf. Therefore I used those functions.

2.5 Results

As we can see in Figure 1 the accuracy at the begging of the epoch 0 was very low but accuracy went up as the model continued to train. After a while the accuracy peaked at around 0.25 and the plot is almost flattened. I believe I could have increased the accuracy of the model if I implemented a more advanced neural network. In the second epoch, we can see that accuracy is more stabilized (Figure 2). And in the last epoch, we can see that the accuracy is improved considerably since the epoch is higher. We can also apply the same thing to the loss. It dropped after the first few training and then it stabilized around 0.875.

The number of epochs did help more accurate results but adding more epochs after a point doesn't really change anything. I also run the code with 50 epochs but honestly, the improvement of the accuracy was so low that I left it at 5.

The result that I got is 0.26833976833976836 for accuracy and 8485132833611978 for loss.

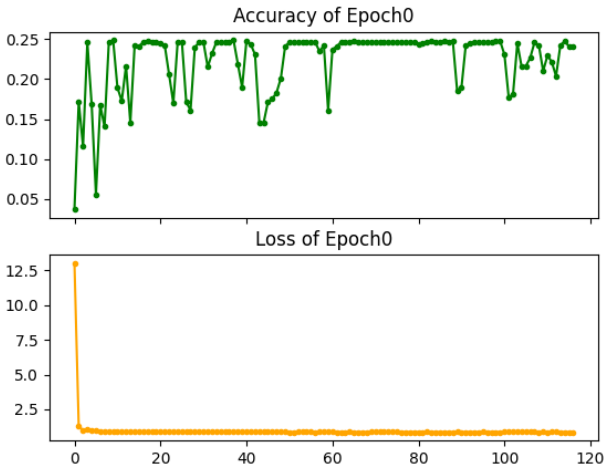

Figure 1: Accuracy and loss of epoch0.

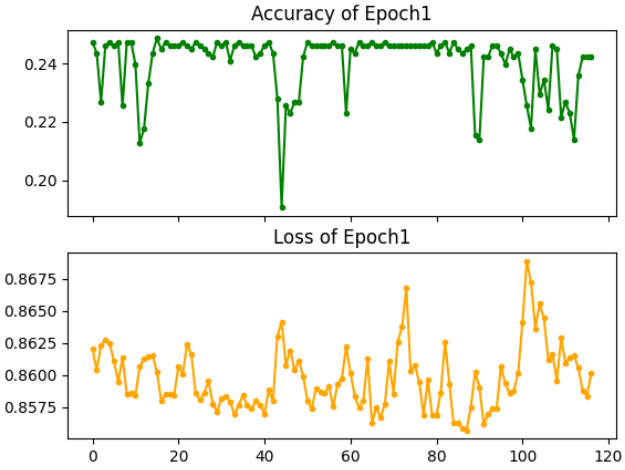

Figure 2: Accuracy and loss of epoch1.

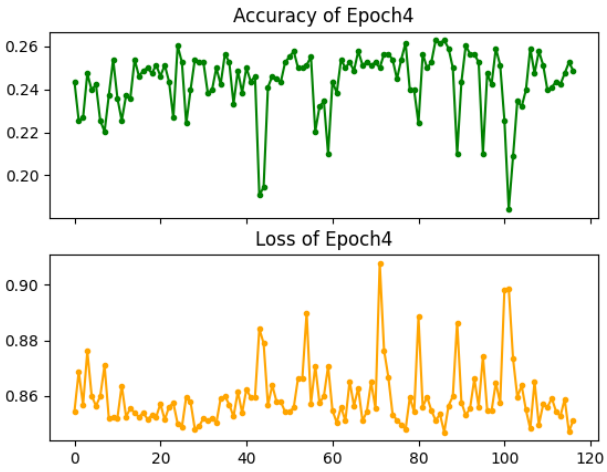

Figure 3: Accuracy and loss of epoch4.

```
Epoch= 0, Coverage= %68.67, Accuracy= 0.24355670103092783, Loss= 0.8600471064555926
Epoch= 0, Coverage= %85.84, Accuracy= 0.23067010309278352, Loss= 0.8668116300946304
Epoch= 1, Coverage= %0.00, Accuracy= 0.24742268041237114, Loss= 0.8620922851727851
Epoch= 1, Coverage= %17.17, Accuracy= 0.24613402061855671, Loss= 0.8606688455051412
Epoch= 1, Coverage= %34.33, Accuracy= 0.24613402061855671, Loss= 0.8569577811795441
Epoch= 1, Coverage= %51.50, Accuracy= 0.24484536082474226, Loss= 0.8601880839544623
Epoch= 1, Coverage= %68.67, Accuracy= 0.24355670103092783, Loss= 0.8568874591392602
Epoch= 1, Coverage= %85.84, Accuracy= 0.2345360824742268, Loss= 0.8640893532157123
Epoch= 2, Coverage= %0.00, Accuracy= 0.2422680412371134, Loss= 0.8602876268519213
Epoch= 2, Coverage= %17.17, Accuracy= 0.24742268041237114, Loss= 0.8594933249044993
Epoch= 2, Coverage= %34.33, Accuracy= 0.2422680412371134, Loss= 0.8551213432396966
Epoch= 2, Coverage= %51.50, Accuracy= 0.23067010309278352, Loss= 0.8582299584784022
Epoch= 2, Coverage= %68.67, Accuracy= 0.21520618556701032, Loss= 0.8590106821974578
Epoch= 2, Coverage= %85.84, Accuracy= 0.21262886597938144, Loss= 0.876769545066457
Epoch= 3, Coverage= %0.00, Accuracy= 0.23711340206185566, Loss= 0.857476814560712
Epoch= 3, Coverage= %17.17, Accuracy= 0.24097938144329897, Loss= 0.8591261709699406
Epoch= 3, Coverage= %34.33, Accuracy= 0.24484536082474226, Loss= 0.8567044225190522
Epoch= 3, Coverage= %51.50, Accuracy= 0.23582474226804123, Loss= 0.8566087410266879
Epoch= 3, Coverage= %68.67, Accuracy= 0.20876288659793815, Loss= 0.8731963981886586
Epoch= 3, Coverage= %85.84, Accuracy= 0.22293814432989692, Loss= 0.8909756208300281
Epoch= 4, Coverage= %0.00, Accuracy= 0.24355670103092783, Loss= 0.8543961865548975
Epoch= 4, Coverage= %17.17, Accuracy= 0.24613402061855671, Loss= 0.8570916237465394
Epoch= 4, Coverage= %34.33, Accuracy= 0.25, Loss= 0.8623856767082179
Epoch= 4, Coverage= %51.50, Accuracy= 0.24355670103092783, Loss= 0.8546916599797617
Epoch= 4, Coverage= %68.67, Accuracy= 0.22422680412371135, Loss= 0.8885255465804741
Epoch= 4, Coverage= %85.84, Accuracy= 0.22551546391752578, Loss= 0.8983293252136207
Epoch= 4, Coverage= %99.96Test Scores:
(0.26833976833976836, 0.8485132833611978)
```

Figure 4: Results