CSE474/574 Introduction to Machine Learning Programming Assignment 2

Non Linear Models for Supervised Learning

Group 48

Dhruv Sureshbhai Patel (#50321707)

Saumya Jay Dholakia (#50320175)

TABLE OF CONTENTS:

Report 1:

1) The ‘reviews’, ‘labels’ and the ‘vocabulary’ files were initially read and the data was embedded in the notebook. The labels here are also referred to as sentiments. They basically denote the connotation of the words used in the movie reviews.
2) The train and test data was then split with the first 24000 entries of the data belonging to the training part and the remaining 1000 entries belonging to the testing part respectively. Hence the magnitude of the testing and train data used to test the hand crafted classifier was different. This can also have a marked effect on the final accuracy of the model. If the volume of training data used was lower, it could have reduced the accuracy of the model to some extent.
3) Every entry of the review dataset is actually a paragraph. Hence, to extract a definite count of every word occuring in the entire dataset, the data present in every individual entry was split into words which were then counted and compared with the entries in the label dataset. Hence a sentiment label of ‘POSITIVE’ or ‘NEGATIVE’ was assigned to every word of the entire dataset.
4) The ‘zip’ function was used to map the label data(words) to the review data(words). The ‘split’ function was used to split the paragraphs into words. The positive and negative words were updated in the counter using the ‘update’ function. The ‘most common’ function was used to narrow down the search in the counter function to the most frequently used words.
5) Words occurring greater than 100 times were only considered as part of the word count. Hence the positive to negative ratios for all such words were calculated. To avoid NaN values in the solution, a dummy value of 1 was assigned to the denominator and only the positive word counts were considered in those cases. Words occurring more in the positive section have a positive to negative ratio greater than 1 whereas the words occurring in the negative section have a ratio of less than 1. Words with a ratio of 1 are considered Neutral.
6) The positive to negative ratios are then converted to the log scale wherein a ratio greater than 0.5 indicates positive words, a ratio between -0.5 and 0.5 indicates a neutral word and a ratio below 0.5 indicates a negative word respectively. Accordingly the positive and negative counters are updated.
7) Finally if the total positive words in the counter are greater than the set of negative words then a positive label is returned else a negative label is returned. This ratio denotes the feature of different words in the dataset.
8) Now that a given set of weight labels were generated and the classifier fitted, it was tested on the test data split initially. The ‘reviews_test’ data was compared with the corresponding labels or features of the words generated in the ‘pos_neg_ratios’ using the ‘nonml_classifier’ function. The pos_neg_ratios’ contains the data labels obtained using the training data. The function spits out ‘POSITIVE’ or ‘NEGATIVE’ data labels using the ‘reviews_test’ data and stores it in the ‘prediction_test’ object.

9) Now to determine the accuracy of the given classifier, the ‘sentiments_test’ data was compared with the corresponding data labels present in the ‘prediction_test’ object. The number of correct matches found between the ‘sentiments_test’ and the ‘prediction_test’ divided by the total number of entries in the ‘sentiments_test’ gives the accuracy of the classifier.
10) The corresponding accuracy was found out to be 78.2%. Please refer the screenshot below:

```
predictions_test = []
for r in reviews_test:
    l = nonml_classifier(r,pos_neg_ratios)
    predictions_test.append(l)

# calculate accuracy
correct = 0
for l,p in zip(sentiments_test,predictions_test):
    if l == p:
        correct = correct + 1
print('Accuracy of the model = {}'.format(correct/len(sentiments_test)))
```

```
Accuracy of the model = 0.782
```

Report 2:

1) A simple multi layer perceptron is implemented for a given set of input parameters such as width, number of hidden layers, type of activation and objective functions etc.
2) Accordingly a training accuracy of 93.82% and a testing accuracy of 86% was found. The time consumed was 63.16 seconds for this run. Please refer the snap attached below:

```
Train acc: 0.938208, Test_acc: 0.860000
Time elapsed - 63.61324906349182 seconds.
```

   Its accuracy is better than rule-based algorithm(Approach 1)
3) This can be considered as the baseline case and serves as a reference for future calculations and variable cases explored in report 3.
4) The implementation of the plain vanilla neural network indicates a drastic change in the training and testing accuracies of the data when compared to the simple hand crafter non linear classifier. Hence the implementation of the neural network to classify complex data posits a definite advantage in terms of the computation time and accuracies over the rule based non linear classifier.

Report 3:

1) The complexity of the given multi layer perceptron is increased by using the following ways.
    a) Increasing the width (number of units) within an individual layer.
        1) The number of hidden layers and the number of epochs (50) used here are kept constant.(Only 1 hidden layer is used). The width of the hidden layers used originally was 10. For this case the width was increased to 50.

```
Train acc: 0.914292, Test_acc: 0.848750
Time elapsed - 47.048447370529175 seconds.
```

        It is observed that the training accuracy was reduced from 93% for a width of 10 units to 91% for a width of 50 units. The testing accuracy also dropped from 86% to 84%. However the time consumed was reduced.
        2) In this case the width was further increased to 100.

```
Train acc: 0.881667, Test_acc: 0.837500
Time elapsed - 65.15324091911316 seconds.
```

        It is observed that the training accuracy drops from 91% for a width of 50 units to 88% for a width of 100 units. The testing accuracy also decreases from 84% to 83%. The time required for the computation however increases.
        3) There are some rules of thumb which may be used to increase the width of the hidden layers.
            a) The number of hidden neurons should be between the size of the input layer and the size of the output layer.
            b) The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
            c) The number of hidden neurons should be less than twice the size of the input layer.
        4) The resultant decrease in accuracy is mainly due to overfitting when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers.
    b) Adding more hidden layers to the architecture.
        1) In this case the hidden layers are increased from 1 to 2 keeping all other hyperparameters constant.

```
Train acc: 0.927000, Test_acc: 0.863750
Time elapsed - 30.07545018196106 seconds.
```

        The training accuracy drops by 1% as compared to the previous case but the testing accuracy remains constant. The time required decreases by almost 50%.

2) In this case hidden layers are increased from 2 to 3 keeping all other hyperparameters constant.

```
Train acc: 0.939167, Test_acc: 0.848750
Time elapsed - 32.069719552993774 seconds.
```

The training accuracy surprisingly remains unchanged as compared to the previous case and the testing accuracy remains constant. The time required increases.

3) In this case hidden layers are increased from 3 to 5 keeping all other hyperparameters constant.

```
Train acc: 0.939000, Test_acc: 0.833750
Time elapsed - 33.41009330749512 seconds.
```

The training accuracy surprisingly remains unchanged as compared to the previous case and the testing accuracy drops by 1%. The time required increases.

4) The entire formulation of deep learning typically revolves around increasing the number of hidden layers. This often tends to create overfitting for simple perceptron networks, however for multi layer perceptrons especially in this case when dealing with non linear classifiers, making the neural network is preferred since they have better capabilities to represent convex functions and convert them into a higher dimensional space.

5) In this case, the accuracies do not increase or decrease radically but with a drastic increase in the number of hidden layers, it could result in a loss of accuracy values.

c) Increase the number of training epochs.

1) In this case, the number of epochs were changed from 50 to 100 keeping all other hyperparameters constant.

```
Train acc: 0.951250, Test_acc: 0.850000
Time elapsed - 58.28065776824951 seconds.
```

The training accuracy increases from 93% to 95% but the testing accuracy tends to drop by 1%. The time required decreases slightly as compared to the baseline case.

2) In this case, the number of epochs were changed from 100 to 150 keeping all other hyperparameters constant.

```
Train acc: 0.953458, Test_acc: 0.835000
Time elapsed - 87.16807579994202 seconds.
```

The training accuracy remains constant when compared with the previous case. However there is a loss in testing accuracy of 2%. The time required also increases significantly as compared to the previous case.

3) In this case, the number of epochs were changed from 150 to 200 keeping all other hyperparameters constant.

```
Train acc: 0.954958, Test_acc: 0.853750
Time elapsed - 113.9172055721283 seconds.
```

The training accuracy remains constant when compared with the previous case. The testing accuracy again increases by 2%. The time required also increases significantly as compared to the previous case as expected.

4) The number of epochs does not have a significant impact on the performance of the neural network. This value should be set at a high value initially. After a certain stage as observed from the above results the accuracy values tend to saturate to a constant level. After this stage they tend to drop due to overfitting of the data. Hence this serves as a marker to stop the iterations involved.
5) The number of epochs required also depends on the batch size. A smaller batch size can increase the internal iterations required to complete the feed forward and the back propagation of the data which in turn depends on the type of optimization algorithm used.

2) Lastly, the ignore words function deletes the neutral words from the train and test data available. This considerably reduces the amount of time required for the linear scanning of all the entries and also reduces the overall computation time.

3) Overall, different hyperparameters have different kinds of effects on the performance of the non linear classifier. Increasing the number of hidden layers can lead to overfitting of the data as already mentioned. Similarly, increasing the width of the hidden layers also tends to reduce the overall accuracy. Increasing the number of epochs does not have any effect in terms of accuracies but can serve as a threshold or an indicator to observe the apparent decline in the accuracies after a certain time period. The ignore word() function helps to filter out data which lies in log range of -0.5 to +0.5. This helps in reducing effective time taken to form a decision boundary or a non linear surface in this case.

Report 4:

The configuration of the PC used was the intel i5 quad core 5th generation processor with a 1 GB Nvidia graphics card. Some of the runs were also executed on the Google Colab server.

1) Run the evaluation of the 1 hidden layer neural network:(28x28)
   Training accuracy = 79.45%
   Test Accuracy = 67.85%
   Run Time = 4 sec per epoch (Approximately 30 Min)

```
Epoch 500/500
100000/100000 [==============================] - 11s 111us/step - loss: 0.8410 - accuracy: 0.7945
Testing accuracy 0.67856
```

Epoch	Time	Loss	Accuracy
1	4Sec	320%	61%
30	4Sec	115%	76%
160	4Sec	83%	80%
300	4Sec	80%	80%
500	**4Sec**	**84%**	**79%**

The neural network reports the maximum accuracy for the baseline case of 1 hidden layer and an image resolution of 28X28. The computation time required is also higher for the baseline case owing to the larger volume of the input data required to be processed.

2) Compare the performance when hidden layers are increased:
   a) **<u>For 3 Hidden layers:</u>**

   With 3 hidden layers, there must be overfitting of the data. This causes a sharp decline in the accuracy. The time required also is higher in this case as the width of the layers is kept the same (256 neurons for all the 2 layers).
   Training accuracy = 21%
   Test data Accuracy = 19%
   Run time = Approximately 3 hours 30 Min

```
Epoch 500/500
100000/100000 [==============================] - 19s 194us/step - loss: 2.4514 - accuracy: 0.2113
```

Epoch	Time	Loss	Accuracy
1	20 Sec	145%	67%
50	20 Sec	93%	73%
100	20 Sec	101%	75%
150	20 Sec	147%	65%
250	20 Sec	379%	27%
500	**20 Sec**	**245.14%**	**21.13%**

b) <u>For 5 Hidden Layers:</u>

Similar things happen in the 5 Hidden layer too. After Certain number of epochs, accuracy starts to drop significantly. This is also due to overfitting of the data. The accuracy is lower than the previous cases.

Training accuracy = 9%

Test data Accuracy = 9%

Run Time = Approximately 4 Hours 30 min

```
Epoch 500/500
100000/100000 [==============================] - 20s 200us/step - loss: 2.3028 - accuracy: 0.0987
Testing accuracy 0.1
```

Epoch	Time	Loss	Accuracy
1	20Sec	112%	70%
9	20Sec	132%	69%
51	20Sec	232%	15%
150	20Sec	230%	10%
300	20Sec	238%	9%
500	**20Sec**	**278%**	**9%**

From the above result, It can be concluded that there should be an optimal number of hidden layers(and neurons) to get a more accurate result. This unique set of hyperparameters can be obtained through brute force search or grid searching algorithms.

3) Use the resize_image() function to reduce the resolution of the images
   1) For (20 x 20)
      Training Accuracy = 73%
      Testing Accuracy = 65%
      Run Time = Approximately 2 hours 30 minutes

```
Epoch 500/500
100000/100000 [==============================] - 8s 81us/step - loss: 1.0383 - accuracy: 0.7328
Testing accuracy 0.65568
```

Epoch	Time	Loss	Accuracy
1	8Sec	112%	63%
50	8Sec	127%	74%
150	8Sec	120%	73%
300	8Sec	102%	73%
500	**8Sec**	**103%**	**73%**

   2) For (15 x 15)
      Train Accuracy = 75%
      Test Accuracy = 65%
      Run Time = Approximately 50 min

```
Epoch 500/500
100000/100000 [==============================] - 6s 58us/step - loss: 1.0005 - accuracy: 0.7528
Testing accuracy 0.69328
```

Epoch	Time	Loss	Accuracy
1	6Sec	324%	68%
32	6Sec	127%	75%
150	6Sec	113%	75%
310	6Sec	106%	75%
500	**6Sec**	**100%**	**75%**

3) For (10 x 10)
Training Accuracy = 73%
Test Accuracy = 70%
Run Time = 35 Min (Approximately)

```
Epoch 500/500
100000/100000 [==============================] - 4s 40us/step - loss: 1.1702 - accuracy: 0.7348
Testing accuracy 0.70056
```

Epoch	Time	Loss	Accuracy
1	4Sec	299%	69%
30	4Sec	118%	75%
160	4Sec	126%	74%
300	4Sec	122%	73%
500	**4Sec**	**117%**	**73%**

4) For (5 x 5)
Training Accuracy = 72.32%
Testing Accuracy = 72%
Run Time = 20 Min (Approximately)

```
100000/100000 [==============================] - 3s 34us/step - loss: 1.0874 - accuracy: 0.7206
Epoch 500/500
100000/100000 [==============================] - 3s 34us/step - loss: 1.0866 - accuracy: 0.7222
Testing accuracy 0.72116
```

Epoch	Time	Loss	Accuracy
1	3Sec	173%	68
30	3Sec	98%	75%
179	3Sec	119%	72%
300	3Sec	111%	71%
500	**3Sec**	**108%**	**72%**

4) The following conclusions can be drawn from the above analysis:(PART 3/3)

1) As the image resolution decreases from 28 X 28 to 5 X 5, it results in a loss of train and test accuracies keeping all the other related hyperparameters such as the number of hidden layers, epochs and the width constant for every run.
2) Also it is seen from the image visualization in the code, that as the resolution is reduced the input data stream gets compressed and becomes blurry. This causes a loss in many of the features present in the original image. This also hinders the ability of the non linear classifier to recognize the right data which can be seen from the loss in the testing and training accuracies.
3) The testing accuracy on an average remains lower than the training accuracies in most of the cases but can be improved by exposing the neural network to a larger training dataset.
4) The values of the accuracies obtained for all the cases vary to a certain extent after restarting the code every time. This happens due to a random selection of the images from the data set. However with multiple runs, the neural network can learn the most important features quite well.
5) There is a significant reduction on the computation time involved as the image resolution is decreased.