CSE-574 Introduction to Machine Learning

Project 2

Apoorva Biseria
abiseria@buffalo.edu
UB person number -50291145

1. Introduction

This project deals with using Machine Learning algorithm that is Linear Regression, Logistic Regression and Neural Network to identify whether the handwriting belongs to the same writer or different writer. We are provided with two types of datasets Human observed dataset and GSC dataset. For the human observed dataset we have 9 features for each and every writing and for GSC dataset we have 512 features for every writing. For comparing two handwritings we are using two methods here subtraction as well as concatenation. Subtraction implies taking the absolute value of the difference of respective values and concatenation implies placing the feature side by side of the respective pairs. If we use concatenation the dataset will contain 18 features concatenating the features of both the handwritings, and for GSC it will have 1024 features. If we use subtraction the number of features will be 9 and 512 for human observed and GSC respectively. The model is trained for similar as well as different writers. But, the number of combinations of different writers' sample is much more than that of same. So we take only that much number of samples from different as present in same which brings us to a uniform distribution of data. For each and every dataset we have performed linear regression, logistic regression as well performed neural network using Keras.

2. Processing of dataset

We were provided with two types of dataset Human Observed as well as GSC dataset. Each dataset had two types of files which were same_pairs and different_pairs. For same pairs the target value was 1 and for different the target value was 0.

a) **For Human Observed Dataset:** The same pair has around 780 rows and the different pair had much larger rows, so I randomly took around 780 rows from different pairs and merged it with same pairs, and then shuffled it again, This ensured a uniform distribution of data.
b) **For GSC Dataset:** The same pair had around 71K rows and different pairs had around 700K rows, thus I randomly took 71K rows from different and merged it with same pairs and shuffled it for uniform distribution of data. For performing linear regression on the GSC dataset we need to perform the inverse of the dataset, but in the GSC dataset, some columns were completely zero, so that columns were needed to be removed in order to find the inverse of the matrix.

3. Performance Metric

For linear regression, E-rms $E_{RMS} = \sqrt{2E(w)/N_V}$ has been calculated whereas for Logistic as well as Keras implementation accuracy has been calculated.

4. Hyperparameter setting and result for different models

4.1 Linear Regression on Human Observed Data with concatenation

Training Percent = 70
No.of basis function = 18
Learning Rate = 0.01
E-rms Testing obtained = 0.49968

4.2 Logistic Regression on Human Observed Dataset with concatenation

Training Percent = 70
Learning Rate = 0.01
Accuracy Testing obtained = 48.7288

4.3 Linear Regression on Human Observed Dataset with subtraction

Training Percent = 70
No.of basis function = 9
Learning Rate = 0.01
E-rms Testing obtained = 0.49992

4.4 Logistic Regression on Human Observed Dataset with subtraction

Training Percent = 70
Learning Rate = 0.01
Accuracy Testing obtained = 50.7288

4.5 Keras implementation on Human Observed Dataset with concatenation

dropout = 0.2
input = 18
first dense layer nodes = 512
second dense layer nodes = 512
third layer = 1
Activation on first layer = relu
Activation on second layer = relu
Activation on first layer = sigmoid
Optimizer = rmsprop
Loss= binary_crossentropy
Validation data split = 0.2
Epochs = 1100
Batch size = 128
Accuracy Testing obtained = 48.7288

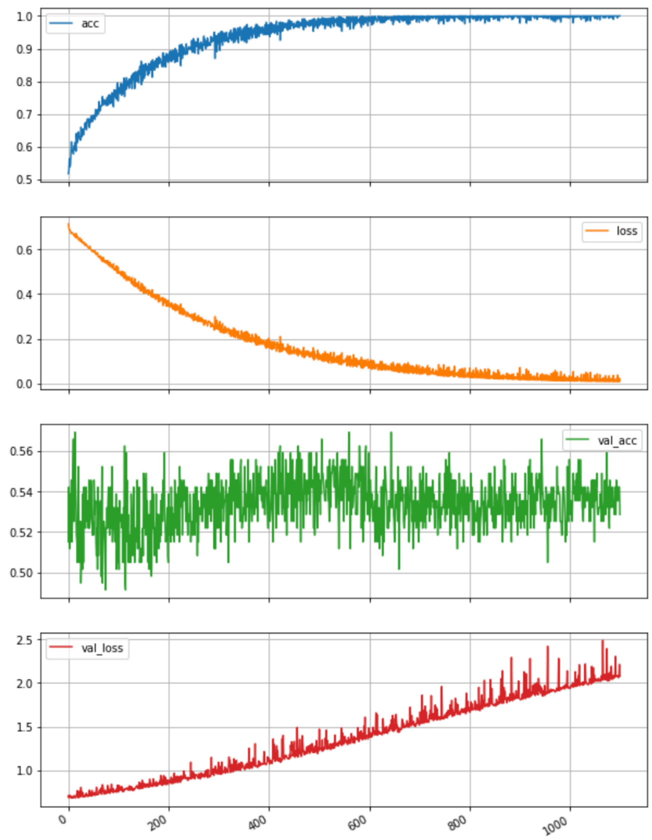

Figure - 4.1

4.6 Keras implementation on Human Observed Dataset subtraction

dropout = 0.2
input = 9
first dense layer nodes = 512
second dense layer nodes = 512
third layer = 1
Activation on first layer = relu
Activation on second layer = relu
Activation on first layer = sigmoid
Optimizer = rmsprop
Loss= binary_crossentropy
Validation data split = 0.2
Epochs = 1100
Batch size = 128
Accuracy Testing obtained = 53.7288

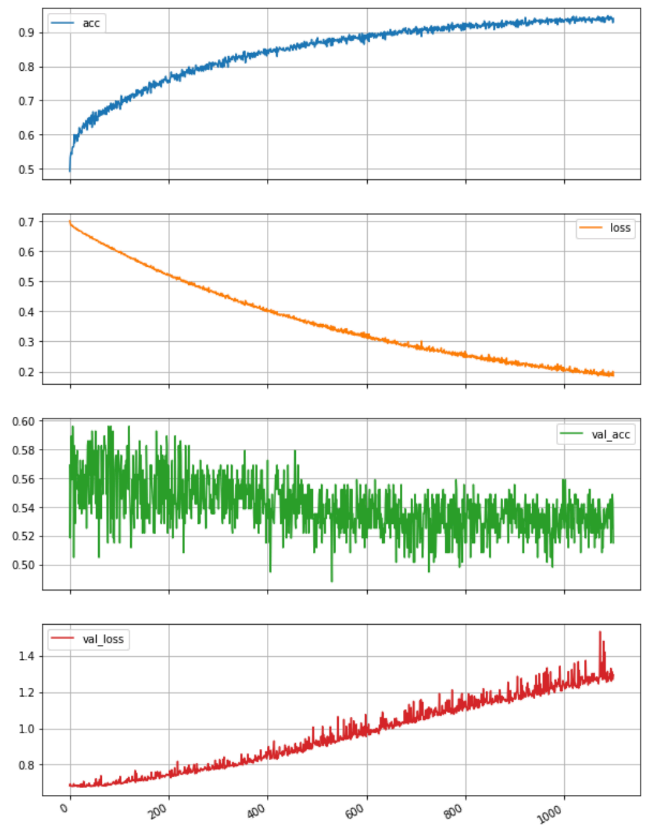

Figure - 4.2

4.7 Linear Regression GSC Data with concatenation

Training Percent = 80
No.of basis function = 20
Learning Rate = 0.01
E-rms Testing obtained = 0.67101

4.8 Logistic Regression on GSCDataset with concatenation

Training Percent = 70
Learning Rate = 0.01
Accuracy Testing obtained = 55.084

4.9 Linear Regression on GSC Dataset with subtraction

Training Percent = 80
No.of basis function = 10
Learning Rate = 0.01
E-rms Testing obtained = 0.57614

4.10 Logistic Regression on GSC with subtraction

Training Percent = 70
Learning Rate = 0.01
Accuracy Testing obtained = 49.3672

4.11 Keras implementation on GSC with concatenation

dropout = 0.2
input = 1024
first dense layer nodes = 512
second dense layer nodes = 512
third layer = 1
Activation on first layer = relu
Activation on second layer = relu
Activation on first layer = sigmoid
Optimizer = rmsprop
Loss= binary_crossentropy
Validation data split = 0.2
Epochs = 50
Batch size = 128
Accuracy Testing obtained = 93.67

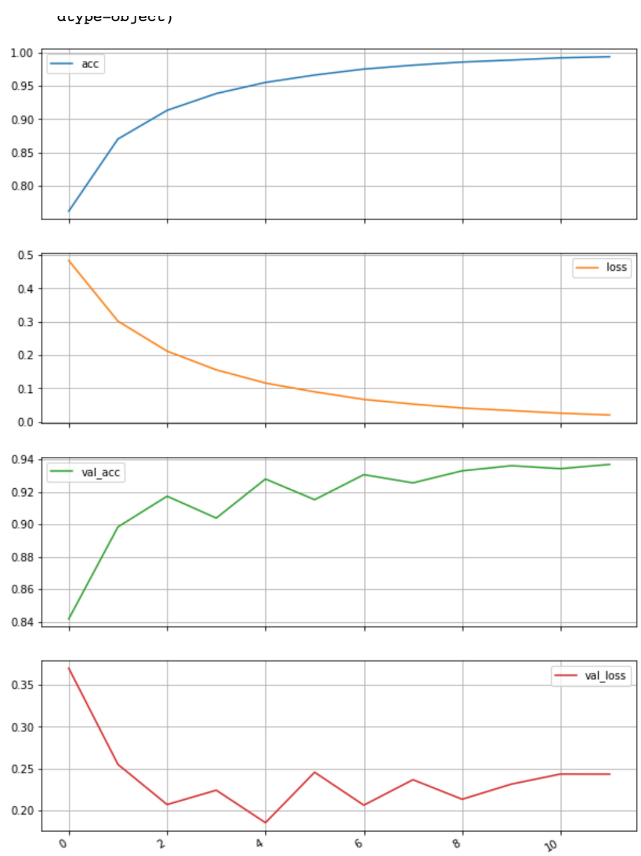

Figure - 4.3

4.11 Keras implementation on GSC with concatenation

dropout = 0.2
input = 1024
first dense layer nodes = 512
second dense layer nodes = 512
third layer = 1

Activation on first layer = relu
Activation on second layer = relu
Activation on first layer = sigmoid
Optimizer = rmsprop
Loss= binary_crossentropy
Validation data split = 0.2
Epochs = 50
Batch size = 128
Accuracy Testing obtained = 93.67

4.12 Keras implementation on GSC with subtraction
dropout = 0.2
input = 512
first dense layer nodes = 512
second dense layer nodes = 512
third layer = 1
Activation on first layer = relu
Activation on second layer = relu
Activation on first layer = sigmoid
Optimizer = rmsprop
Loss= binary_crossentropy
Validation data split = 0.2
Epochs = 50
Batch size = 128
Accuracy Testing obtained = 85.53

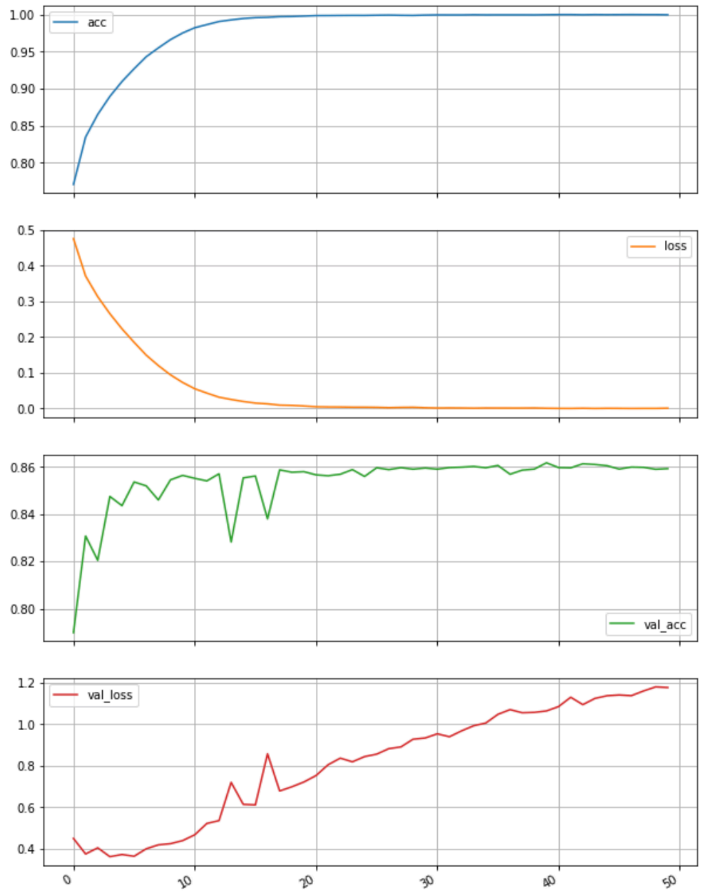

Figure - 4.4

5. Which dataset performed better?

Since GSC had more feature vector(512) and more training data(about 140k), all the three models performed much better than the Human observed dataset which had less training data and less feature vector. The GSC data was much more elaborate and thus allows the different models to perform better.

6. Which model performed better?

According to the given accuracies as well as E-rms observed, neural network the best among the three, and it performed much better on the GSC dataset. Keras is a much more advanced method and performs well on the settings provided. Linear Regression and Logistic Regression performed with 0.50 E-rms and 50% accuracy approximately whereas Keras gives upto 93% accuracy.

7. Conclusion

	Linear Regression (E-rms Testing)	Logistic Regression (Accuracy Testing)	Neural Network (Accuracy Testing)
Human Observed (concatenation)	**0.49968**	**48.7288**	**48.7288**
Human Observed (subtraction)	**0.49992**	**50.7288**	**53.78**
GSC (concatenation)	**0.67101**	**57.6592**	**93.25**
GSC (subtraction)	**0.58309**	**49.3672**	**85.69**

1. Neural Network model performed better on the GSC dataset than the other models. On the human observed dataset, the outputs were pretty comparable and cannot be assigned which model performed better.
2. For Human Observed data feature subtraction gave better results than feature concatenation.
3. For GSC dataset, feature concatenation performed better in case of logistic regression and neural network

5. References

[1]"Matplotlib Bar chart – Python Tutorial", Pythonspot.com, 2018. [Online]. Available: https://pythonspot.com/matplotlib-bar-chart/. [Accessed: 17- Oct- 2018].

[2] "Understanding Learning Rates and How It Improves Performance in Deep Learning", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10. [Accessed: 31- Oct- 2018].