Hand Writing Analysis

Sarang Agarwal
Graduate Student, Department of Computer Science
University at Buffalo
sarangag@buffalo.edu

Abstract

Hand Writing analysis is a common problem used by many crime investigation agencies like FBI. Here we are given images of different 'AND' images along with feature set related to it. We are given 2 different dataset namely Human dataset and GSC dataset. So our aim lies here to train our model on different feature set and determine whether the 'AND' image was written by the same person or not.

It falls into the category of classification where our Target Value will be 0 or 1 where 0 means not matching and 1 means of the same person.

1 Introduction to Different Data Set Used

We have 2 data sets in our problem.

Human Observed Dataset: We have 2 human observers A and B who have observed the handwriting sample and gave value for 9 feature each totaling to 18 feature values.

GSC Dataset: Developed using Gradient Structural Concavity, it will give 512 features each and structure wise its almost similar to Human Observed Data set.

2 Code Structure

I want to introduce to the code structure I am following since it has 3 different types of algorithm implementations and might be confusing.

Main.py

It is the entry point of my program. It also contains linear regression approach.

- **It has output.txt which has all the logs. Nothing will come on Console.**
- I have implemented all 4 combinations namely Human-Concat , Human-Subtract, GSC-Concat, GSC-Subtract.
- After obtaining the required data set in its desired form, I call SGD linear regression method.
- After that I make call to **logisticregression.py.** It contains Logistic regression code.

Logisticregression.py

It has all the required functions for logistic regression. It is simply called from within main.py and it fetches desired output.

Neuralnetwork.py

This file contains code for just the Neural network.

So , I will call my main function in main.py file 4 times for each data set combination and it will in turn do linear regression, call and do logistic regression and call and implement Neural Network on them.

3 Linear Regression

We are already familiar from the last project about the concept of Basis function and Guassian's distribution.

Here we use Basis function ranging from 5-15 to fit features length 9,18,512 and 1024 into the respective Basis function matrix size.

As the name, Linear regression used Linear equation to transform different curves into Linear problem and find its solution.

We know the final output expected per data set is t which is values 0,1.

Human Dataset Observations

Subtraction

Please see on next page for Graph. As expected, since we just have 9 features, when we increase **Basis functions from 5-9, ERMS reduces**.

But after increasing it further from 9 upwards, ERMs becomes constant.

This is due to fact that we have just 9 features, and obviously more clusters than this numbers doesn't really makes any difference in the model prediction and hence constant above this.

I tried changing **Co-variance** factor also but I didn't see much difference.

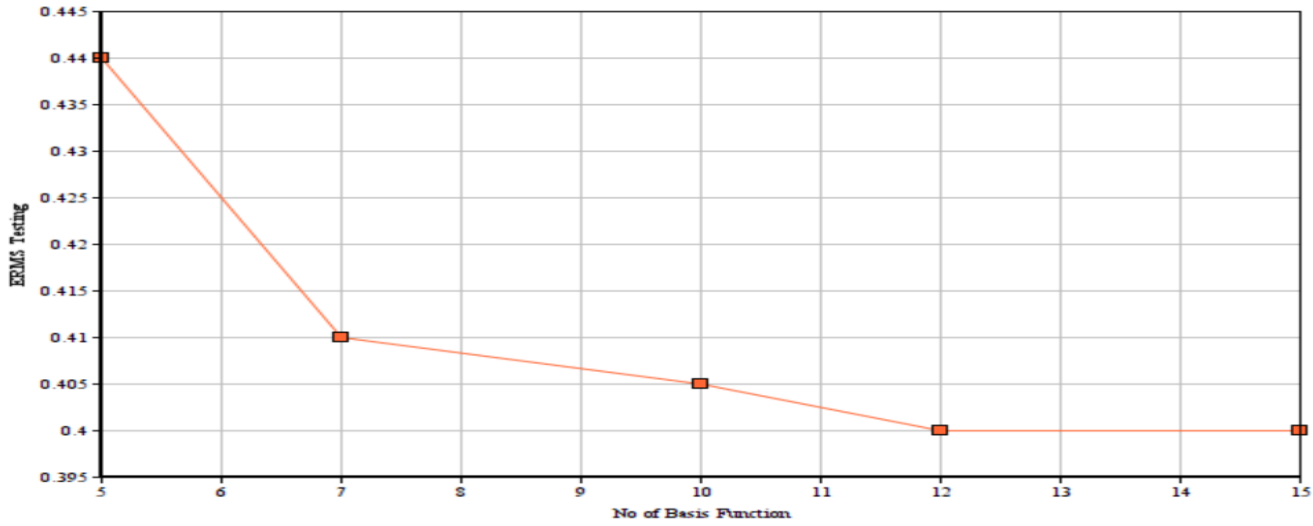
Have plotted a graph with what values I could manage.

Co-Variance factor is determined through **BigSigma = np.dot(200,BigSigma)**

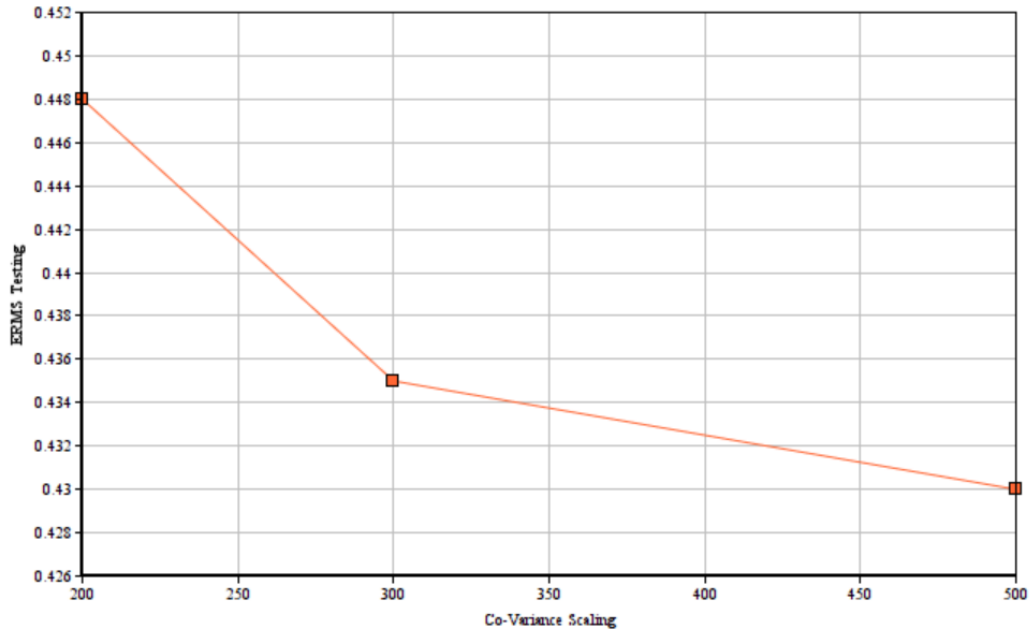
Here 200 is the scaling factor.

I also changed no of iterations over and over again but couldn't find any significant difference based on numbers of iterations.

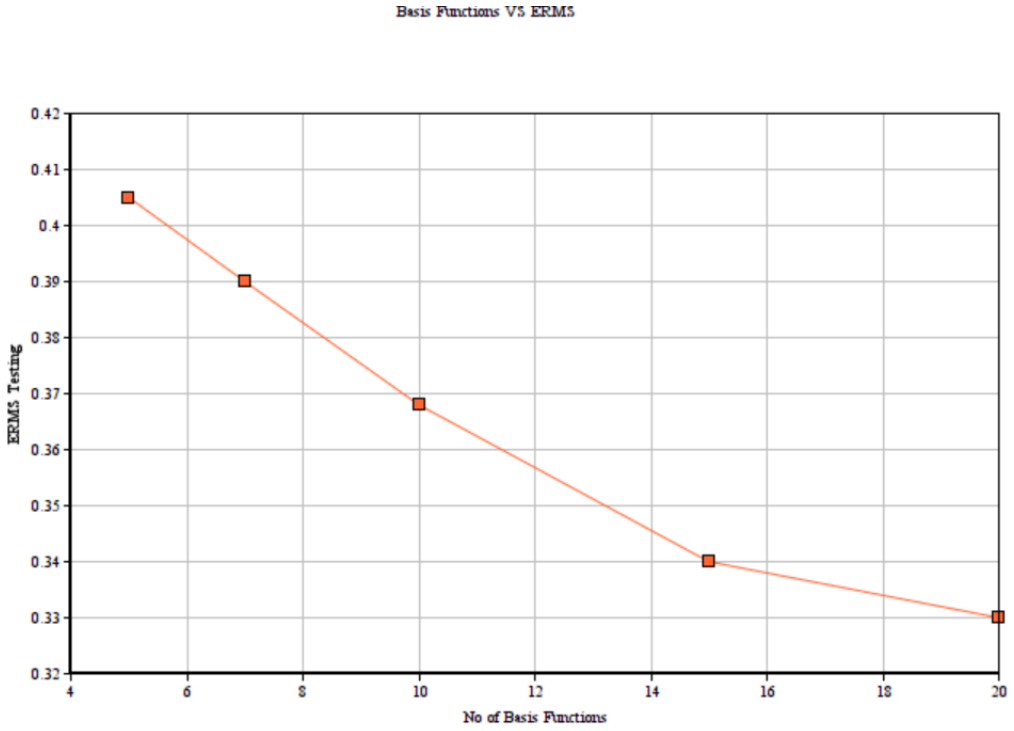
Basis VS ERMS

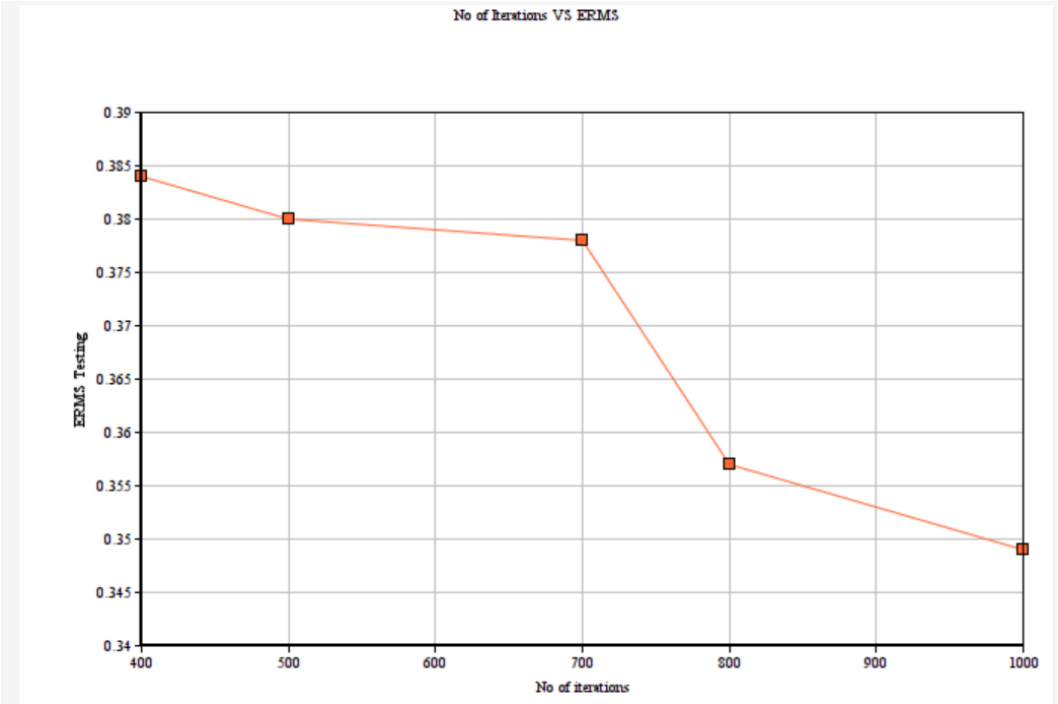


Co-Variance Scaling VS ERMS



Concatenation





I increased number of Basis functions here from 5-20.
Since the feature numbers here are 18, therefore we can see that ERMS decreasing till 15 after

which when I increased no of basis functions to 20, there is not much of fluctuation in ERMs value.

For no of iterations, we do see some decrease in ERMS but it is not too significant.

Conclusion

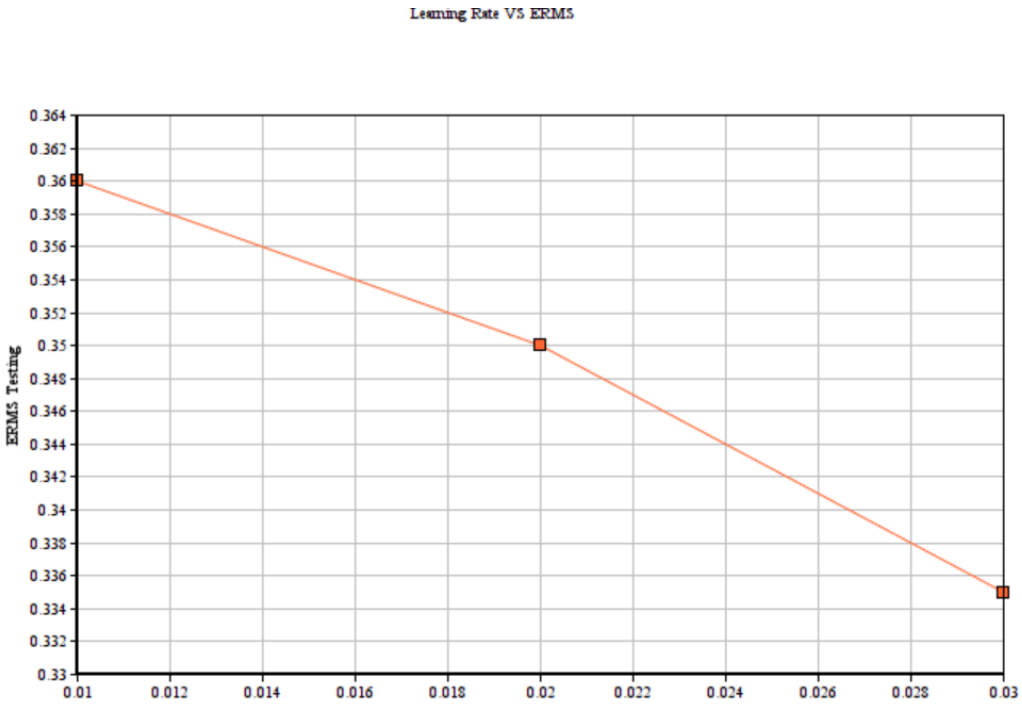
We can clearly see that ERMs in case of Concatenation is lesser. This is due to that fact that we have more number of iterations to train upon.

GSC Data set

Subtraction

In GSC, we have 512 features for Subtraction. Hence obviously we can expect better performance in the starting itself.

Rest is self-explanatory to quite an extent. I will just plot couple of graphs below .



Another interesting thing that I found out is that when I took Random Data Sample, at that time I am getting more ERMS.
Whereas when I am taking first few samples, like fixed number of rows from beginning, then I get less ERMS.

Concatenation

As expected, concatenation is a huge dataset.
Therefore it is having improved accuracy and hence low ERMS.

Iterations	E rms training	E rms validation	E rms testing
100	0.39	0.392	0.391
200	0.375	0.372	0.374
300	0.362	0.358	0.36
500	0.34	0.342	0.3401
800	0.339	0.341	0.34

As seen from above table, after increasing iterations to a certain value, ERMS values becomes constant at a certain value and doesn't decreases futher.

Most optimized Value of ERMS

0.34

Please Note: I tried running loop for 71000 rows, but it is hanging in my system. Hence I have taken 2500 row set only in case of GSC data set.

4 Logistic Regression

Unlike Linear Regression, Logistic Regression establishes the relationship between Input and Output variable as a function of Natural Logarithm and find the most optimal weights or coefficients to model the data.

We use Sigmoidal function as classifier . It classifies in our example any value above 0.5 to 1 and lower values are left as-is.

It is given by

$$y = \frac{1}{1 + e^{-x}}.$$

We differentiate the Loss Function J(Theta) with respect to weight.

Then at each iteration we update the weight by subtracting this value and reaching an optimal weight value at the end. Hence our algorithm converges and we get an optimal value.

I am not writing or going in detail for each data set through logistic regression.

I will just summarize the Accuracy I got for different Data set below.

I have implemented all the 4 data set on Logistic Regression as well.

Accuracy	**DataSet**	**Learning Rate**
0.72	Human Subtraction	0.1
0.91	Human Concatenation	0.1
0.82	GSC Subtraction	0.1
0.94	GSC Concatenation	0.1

Accuracy	**DataSet**	**Learning Rate**
0.75	Human Subtraction	0.3
0.92	Human Concatenation	0.3
0.83	GSC Subtraction	0.3
0.958	GSC Concatenation	0.3

Accuracy	**DataSet**	**Iterations**	**Learning Rate**
0.71	Human Subtraction	100	0.1
0.90	Human Concatenation	100	0.1
0.83	GSC Subtraction	100	0.1
0.92	GSC Concatenation	100	0.1

Accuracy	**DataSet**	**Iterations**	**Learning Rate**
0.74	Human Subtraction	400	0.1
0.91	Human Concatenation	400	0.1
0.88	GSC Subtraction	400	0.1
0.96	GSC Concatenation	400	0.1

My program hung more than 400 iterations.

5 Neural Network

I am not giving too much detail about Neural Network as we did the same thing in Project 1.2.

I am basically using:

- Sequential Model
- 3 Layered model
- Relu as activatin function in Input layers and Sigmoid function in Output layer to get binary values.
- Loss function is binary_crossentropy.

The code which I am submitting Trains the model on entire data with 250 Epochs of batch size 15 and then predicts the value on Same Training Set only.

However, I tries with breaking it down also in 90:10 configuration where I gave 90% data as Training and then used last 10% to make prediction. There also Accuracy I got was about

```
Epoch 141/150
 - 0s - loss: 0.0316 - acc: 0.9932
Epoch 142/150
 - 0s - loss: 0.0210 - acc: 0.9941
Epoch 143/150
 - 0s - loss: 0.0127 - acc: 0.9980
Epoch 144/150
 - 0s - loss: 0.0115 - acc: 0.9980
Epoch 145/150
 - 0s - loss: 0.0130 - acc: 0.9971
Epoch 146/150
 - 0s - loss: 0.0145 - acc: 0.9961
Epoch 147/150
 - 0s - loss: 0.0156 - acc: 0.9951
Epoch 148/150
 - 0s - loss: 0.0122 - acc: 0.9980
Epoch 149/150
 - 0s - loss: 0.0100 - acc: 0.9980
Epoch 150/150
 - 0s - loss: 0.0116 - acc: 0.9980
----------Accuracy of Neural Network is---------:99.8046875
```

Screen Shot from Sample Run

Accuracy	**DataSet**	**Epochs**	**Batch Size**
0.91	Human Subtraction	150	15
0.96	Human Concatenation	250	15

0.94	GSC Subtraction	250	15
0.99	GSC Concatenation	150	15

Using the Data split into 90:10

Accuracy	**DataSet**	**Epochs**	**Batch Size**
0.88	Human Subtraction	150	15
0.91	Human Concatenation	250	15
0.90	GSC Subtraction	250	15
0.92	GSC Concatenation	150	15

6 Conclusion

We can see how number of feature and amount of data rows effects the ERMS and Accuracy.

Even then our models are so powerful if tunes with correct parameters that they can have an increased accuracy.

References

1. https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
2. https://stackoverflow.com/questions/6740918/creating-a-dictionary-from-a-csv-file
3. http://mathworld.wolfram.com/SigmoidFunction.html
4. https://simple.wikipedia.org/wiki/Logistic_Regression
5. Previous Project 1.1 and 1.2 for code references.
6. https://www.onlinecharttool.com/graph?selected_graph=xy

All above links are mostly used for Code references.