Project 2 Report **Koustubh Vijay Kulkarni**
CSE 574
Prof. Sargur N. Srihari UBID: Kkulkarn
TA: Mihir Chauhan, Tiehang Duan, Alina Vereshchaka and others Person Number: 50288207

Objective

The objective of this project is to find similarity between the handwritten samples of the known and the questioned writer by using linear regression, logistic regression and Neural Network.

Task

Task is to find similarity between the handwritten samples of the known and the questioned writer. We are given two datasets containing various features for the image data that we have. There are two datasets:-

1. **Human Observed features**: Features entered by human document examiners manually. This dataset has 9 features for each image sample.
2. **GSC features**: Features extracted using Gradient Structural Concavity (GSC) algorithm.This dataset has 512 features for each image sample.

We have to apply two settings under which we need to perform linear regression, logistic regression and ANN on each of the above mentioned datasets. They are :-

1. **Feature Concatenation**:- Concatenate each feature of each image sample for a pair and form a dataset of 18 features for human dataset and dataset of 1024 features for GSC.
2. **Feature subtraction**:- Subtract each feature of first sample from second sample in a pair and form a dataset of 9 features for human dataset and dataset of 512 features for GSC.

So in total we have to work with 4 datasets and we need to apply these 3 algorithms (Linear, Logistic and ANN) on each of them.

Dataset Pre-processing

We need to process the existing dataset so that we can train our model efficiently. The datasets that are given to us have 2 CSV files containing data of 2 image ids in 2 columns and target output in 3rd one. One CSV file contains similar writer image samples while second CSV contains different writer image samples.
There is another CSV file which contains the total features for each of the image we have.
What we have to do?
We need to take data from first CSV file viz. 'same_pairs' and data from second CSV file viz. 'diffn_pairs' and we need to take features from third file and we need to either subtract each feature of first image with respective feature of another image or need to concatinate them.
Why we need to do sampling?
If we look into the dataset of Human Observed features, we have 791 records in 'same_pairs' file and around 293032 records in 'diffn_pairs' file. If we take all records from 'diffn_pairs' file then our dataset will be **Skewed**, meaning it will have more samples of different pair writers

and hence our model will not train correctly with such data.
Hence, we take 791 pairs randomly from 'diffn_pairs' file and add them with 791 pairs of similar writer records.
In case of GSC datset, we have around 70 thousand samples in each file. Here, I have taken 4 thousand random samples from each file and merged them together.

Linear Regression Approach

In our first task, we are using Linear regression with Stochastic Gradient descent. Here, we are using Gausian radial basis function to provide non-linearity to the model. In simple words, Linear regression is a method for modeling the relationship between two scalar values: the input variable x and the output variable y.

$$y(x, w) = w^T \phi(x)$$

In order to predict the value of target, we need to learn the weight of each feature which can give us the best result given a new sample. First we use **K-means** clustering algorithm to form 'M' clusters which represents degree of our model. Then we fit each cluster with a Gaussian radial basis function. After that we use these basis functions for linear regression.There will be total 'M' Gaussian radial basis functions for our model. Assuming $\phi_0(x) = 1$ for whatever input, w_0 becomes the bias term. Each basis function $\phi_j(x)$ converts the input vector x into a scalar value.
Once our design matrix, BigSigma matrix are generated we can find optimal weights by stochastic gradient descent method.

Logistic Regression Approach

Logistic regression is a classification algorithm which is used to map observations to a discrete set of classes. This is exact opposite of linear regression which outputs continuous number values. Logistic regression transforms its output using the **sigmoid** function to return a probability value which can then be mapped to two or more discrete classes.
Why we use Sigmoid?
In order to map predicted values to probabilities, we use the Sigmoid function. It maps any real value into another value between 0 and 1.
Loss Function:- Instead of Mean Squared Error, we use a cost function called Cross-Entropy for logistic regression.

Neural Network Approach

We are using Keras - which is a high-level neural networks API, written in Python which runs on top of TensorFlow and few other backend platforms. Layers in our Neural network will have following characteristics-

- **Input Layer** - Input layer will have nodes equal to number of features in the specific dataset. So for example, Human Observed dataset with concatenation will have 18 nodes in Input Layer.
- **Hidden Layer** - We will be having one hidden layer. We have to tune in number of nodes in this layer to achieve descent accuracy.

- **Output Layer** - We will be having an output layer which will have 1 node.

In this model, we are using 'Reactified Linear Unit'(Relu) as an activation function for our first layer. Relu simply activates nodes which has positive value.This means that at a time only a few neurons are activated making the network sparse which makes it efficient and easy for computation.
In the second layer, we are using 'Sigmoid' as an activation function. Sigmoid converts hidden layer outputs to any value between 0 and 1 which represents the probabilities of each bucket from output layer.

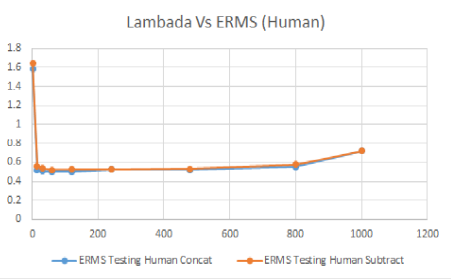
Analysis & Testing

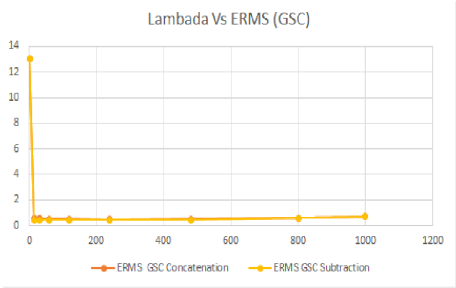
I tried to change few hyper-parameters and observed accuracy for each of them for several combinations of them. Here I am listing few of my observations-

0.1 Output for Linear Regression

1. **Change in Regularization Parameter(Lambada)** I changed Lambada for various values and observed the following results. I have added graphs for comparison between Human Concatenation and Subtraction datasets and GSC Concatenation and Subtraction datasets.

Lambda(λ)	ERMS Human Concatenation	ERMS Human Subtraction	ERMS GSC Concatenation	ERMS GSC Subtraction
2	1.58	1.65	13.017	13.059
15	0.52	0.56	0.528	0.433
30	0.503	0.534	0.525	0.411
60	0.499	0.521	0.497	0.415
120	0.499	0.523	0.512	0.424
240	0.524	0.525	0.449	0.416
480	0.521	0.531	0.511	0.425
800	0.552	0.577	0.587	0.559
1000	0.72	0.719	0.692	0.699

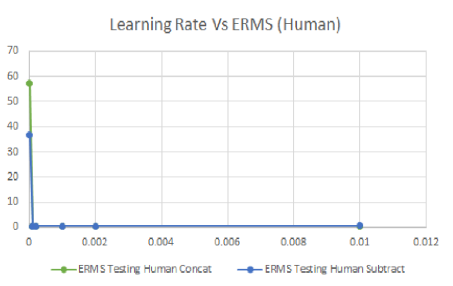


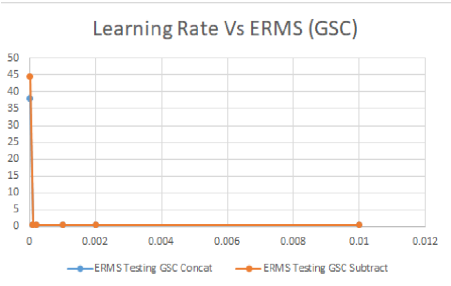


I observed that when Lambada was very low, it was over-fitting with the training data and was not very generalized enough. Hence, ERMS was very high, as I increased the lambada, ERMS value started decreasing.
Another observation was for both GSC and Human datasets, there was no significant difference in Concatenation and Subtraction dataset.

2. **Change in Learning Rate**
I changed Learning Rate for various values and observed the following results. I have added graphs for comparison between Human Concatenation and Subtraction datasets and GSC Concatenation and Subtraction datasets.

Learning Rate	ERMS Testing Human Concat	ERMS Testing Human Subtract	ERMS Testing GSC Concat	ERMS Testing GSC Subtract
0.00001	57.231	37.122	38.132	44.492
0.0001	0.482	0.526	0.497	0.405
0.0002	0.498	0.513	0.498	0.403
0.001	0.496	0.516	0.492	0.403
0.002	0.5	0.523	0.497	0.41
0.01	0.535	0.624	0.493	0.442

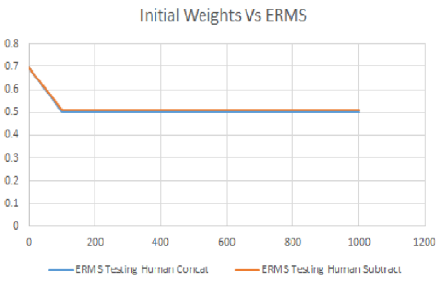


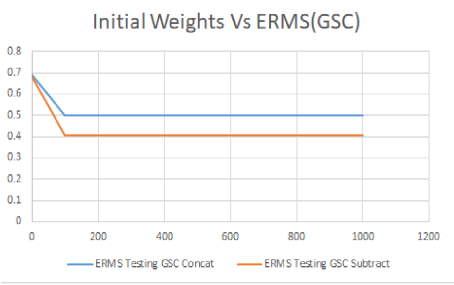


My observation was when learning rate was very very small, it did not converge faster and hence the weights that we got were not optimal. Hence, ERMS in such cases are higher.

3. **Initialization of Weights**
   I randomly initialized weights at first and multiplied it with some factor. I have plotted the graph and table for the results that I observed below. I have added graphs for comparison between Human Concatenation and Subtraction datasets and GSC Concatenation and Subtraction datasets.

Initial Weights	ERMS Testing Human Concat	ERMS Testing Human Subtract	ERMS Testing GSC Concat	ERMS Testing GSC Subtract
0	0.698	0.697	0.688	0.683
100	0.498	0.521	0.496	0.403
400	0.499	0.513	0.493	0.415
800	0.501	0.514	0.488	0.403
1000	0.501	0.513	0.495	0.404





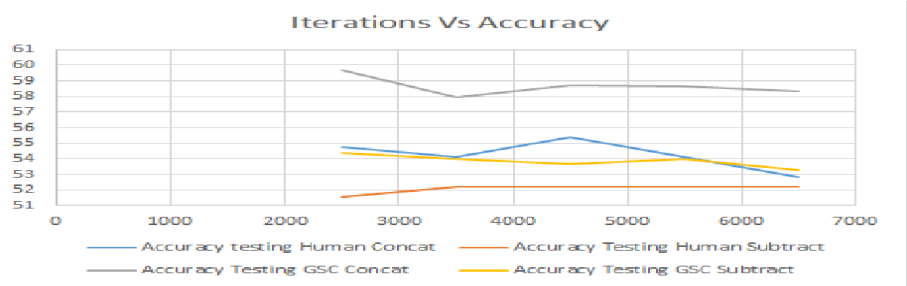
4. **Strange Case I observed For GSC**
   I observed a strange scenario while working with GSC dataset where I was getting 100 % accuracy but my ERMS for validation and testing was very very low. The reason behind that was while splitting the data into training, validation and testing; coincidentally all samples in validation and testing set had target values 0. This is the reason behind low ERMS and High accuracy. Solution to this problem was **shuffling** the entire data randomly so that this doesn't happen.

```
----------Gradient Descent Solution For GSC Concatenated Dataset--------------------
M =14
Lambda  = 2
eta=0.1
E_rms Training   = 0.72641
E_rms Validation = 0.02137
E_rms Testing    = 0.02122
Accuracy Training   = 37.5
Accuracy Validation = 100.0
Accuracy Testing    = 100.0
```

0.2 Output for Logistic Regression

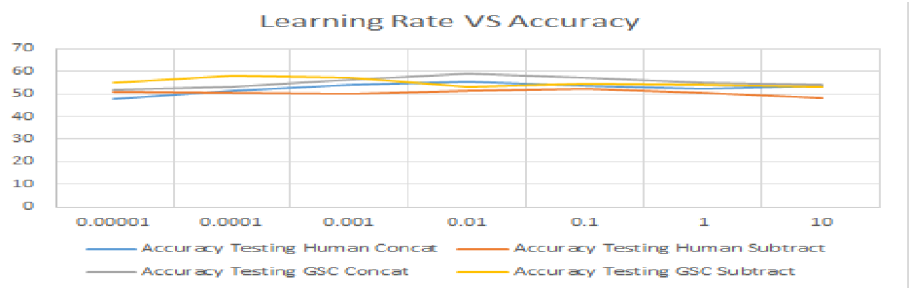
1. **Changing No. Of Iteration** For logistic regression, I have changed the number of iterations for which we update the weights. Here are the results for the comparison of all 4 datasets.

Number of Iteration	Accuracy testing Human Concat	Accuracy Testing Human Subtract	Accuracy Testing GSC Concat	Accuracy Testing GSC Subtract
2500	54.779	51.592	59.668	54.344
3500	54.152	52.229	57.939	54
4500	55.414	52.229	58.745	53.655
5500	54.14	52.229	58.667	54.008
6500	52.866	52.229	58.343	53.322



2. **Changing Learning Rate** For logistic regression, I have changed the learning rate. Here are the results for the comparison of all 4 datasets.

Learning Rate	Accuracy Testing Human Concat	Accuracy Testing Human Subtract	Accuracy Testing GSC Concat	Accuracy Testing GSC Subtract
0.00001	47.8	50.98	52.01	55.001
0.0001	51.65	50.47	53.22	58.03
0.001	54.32	50.05	56.29	57.33
0.01	55.45	51.4	59.001	53.23
0.1	53.88	52.33	57.3	54.72
1	52.22	50.62	54.98	53.99
10	53.62	48.31	54.01	53.23



0.3 Output for Neural Network

1. **Comparison of All four Datasets for Neural Network** As our Human dataset has around 1500 records, I decided to take same number of samples for GSC dataset for comparison and here is my result.
   Accuracy For GSC Concatenation Dataset:- 66 %
   Accuracy For GSC Subtraction Dataset:- 53.33 %
   Accuracy For Human Concatenation Dataset:-48.1 %
   Accuracy For Human Subtraction Dataset:-51.26 %
   So clearly it is showing that GSC dataset has better result than Human dataset.

2. **Taking bigger sample set for GSC dataset** I have taken different sample size as input for GSC dataset and found that for larger data it is showing more accuracy. Also, concatenation dataset is performing better than the Subtraction dataset.

No. Of Samples	GSC Concatenated	GSC Sub
1000	68	53
2000	65	54
3000	73.33	55.33
4000	74.5	65.5
6000	80.33	65.16
8000	82.87	69.75

Final Evaluation

1. **Algorithm comparison**:-
   After testing our datasets on all 3 algorithms, I observed that Neural Network approach works better for all the datasets. For human observed datasets, the difference in accuracy is not that big for Neural, Logistic and Linear but for GSC it was quite significant. I observed that in particular case, I got accuracy for GSC concatenation dataset in neural network as **82.87%** and in linear and logistic it was around 55% and 60%.
   If we compare between linear and logistic, later one performs slightly better than the former.

2. **Setting comparison**:-
   I observed that out of the two settings viz. 'Feature Concatenation' and 'Feature subtraction', I got better results in Feature concatenation datasets. If we take example of GSC dataset for neural network (table I attached in Neural Network section), it is quite clear that Concatenation dataset has more accuracy than the subtraction dataset.

3. **Dataset Comparison**:-
   My observation with both the dataset is that GSC gives more accuracy than the Human observed dataset. I think the reason behind that is, as GSC has more feature and more data, our model can learn better in that case.

References

[1] Class and Recitation notes.

[2] Linear Regression, Basis Functions, Least Squares.Available from World Wide Web: - http://cs.brown.edu/people/pfelzens/engn2520/CS1420_Lecture_2.pdf

[3] Regularization in Machine Learning. Available from World Wide Web:- https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

[4] https://medium.com/@martinpella/logistic-regression-from-scratch-in-python-124c5636b8ac

[5] https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html