Project 2: Handwriting Comparison

Nishi Mehta
Graduate student, Computer Science
#Person 50291671
nishimeh@buffalo.edu

Abstract

This project aims to solve a handwriting comparison problem. Multiple images of different writers are taken and they are compared to check whether they are coming from the same writer or a pair of different writers. To solve this problem, Linear Regression, Logistic Regression and Neural Networks approach is used. Through this project, we compare the results of all these approaches and arrive on an optimal result.

1 Introduction

This project aims to perform handwriting comparison on a given dataset of images. The task at hand is to find the similarity between the samples. Later, we make a prediction whether the sample is from the same writer or different writer based on the similarity. In this project, we use a supervised learning model. We train our model through the data and then predict values for other data.

We use three approaches: Linear regression using stochastic gradient descent, logistic regression and neural networks to build the model for this problem. We then tune the hyperparameters and observe the results obtained by each approach.

1.1 Pairwise approach

In our dataset we have over 1000 images. To form the dataset, we make pairs of each image with the rest of the images. We then find the similarity between each of these pairs.

2 Dataset

Our dataset uses "AND" images samples extracted from CEDAR Letter dataset. Image snippets of the word "AND" were extracted from each of the manuscript using transcript-mapping function of CEDAR-FOX.
Example of Dataset Image

The above images example shows the images from the same writer.
As we can see the dataset images are not normalized. Some are tilted, the sizes of the images are not same and basically, they are not normalized. Hence, we need to generate some features

out of these images. If we use the images directly, we won’t be able to get good results because of the irregularity between images of the same writer.

2.1 Types of Feature Datasets

From the images, certain features are extracted using two methods. We obtain two different datasets.

2.1.1 Human Observed Dataset

The Human Observed dataset shows only the cursive samples in the data set, where for each image the features are entered by the human document examiner. There are total of 18 features for a pair of handwritten “AND” sample (9 features for each sample).
Sample of two image pairs of the Human Observed Dataset:

img_id_A	img_id_B	f_{A1}	f_{A2}	f_{A3}	f_{A4}	f_{A5}	f_{A6}	f_{A7}	f_{A8}	f_{A9}	f_{B1}	f_{B2}	f_{B3}	f_{B4}	f_{B5}	f_{B6}	f_{B7}	f_{B8}	f_{B9}	t
1121a_num1	1121b_num2	2	1	1	3	2	2	0	1	2	2	1	1	0	2	2	0	3	2	1
1121a_num1	1386b_num1	2	1	1	3	2	2	0	1	2	3	1	1	0	2	2	0	1	2	0

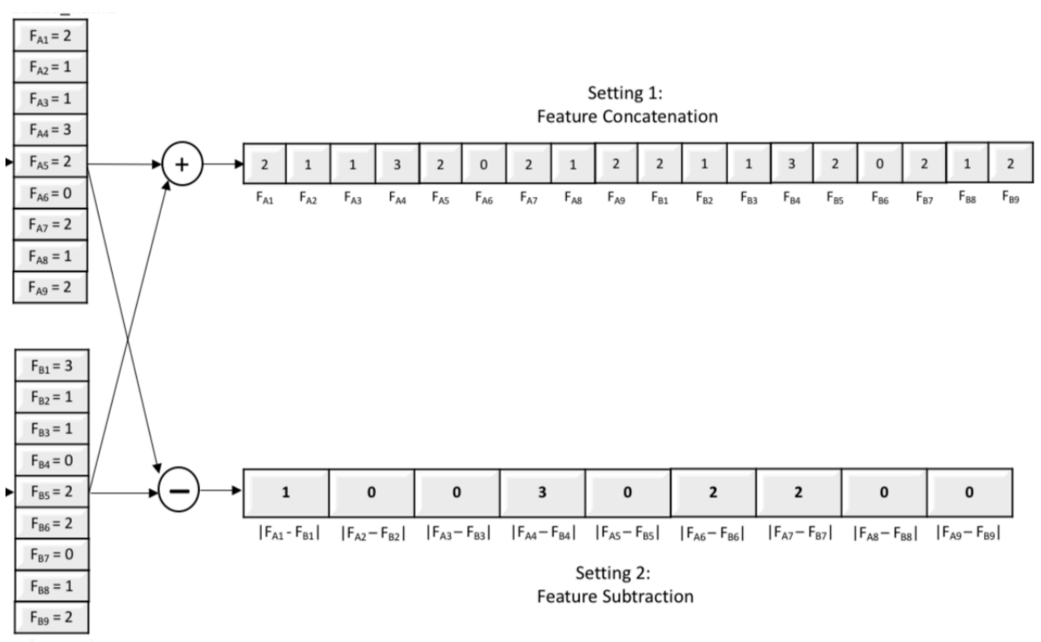
2.1.2 GSC dataset using feature engineering

Gradient Structural Concavity algorithm generates 512 sized feature vector for an input handwritten “AND”image. The entire dataset consists of 71,531 same writer pairs and 762,557 different writer pairs(rows).
Sample of two image pairs of the GSC Dataset:

img_id_A	img_id_B	f_{A1}	f_{A2}	f_{A3}	f_{A4}	f_{A5}	f_{A6}	. . .	f_{A512}	f_{B1}	f_{B2}	f_{B3}	f_{B4}	f_{B5}	f_{B6}	. . .	f_{B512}	t
1121a_num1	1121b_num2	0	1	1	0	1	0	. . .	0	0	1	1	0	0	1	. . .	1	1
1121a_num1	1386b_num1	0	1	1	0	1	0	. . .	0	1	1	1	0	1	0	. . .	0	0

2.2 Pairing Images

The task for pairing two image feature vectors can be done in two ways. One is by concatenation and the second way is by subtraction of the two features.
A figure depicting the same is shown below:



Hence, by using both settings and two datasets, we get 4 unique datasets.

1. Human Observed Dataset with concatenated features
2. Human Observed Dataset with subtracted features
3. GSC Dataset with concatenated features
4. GSC Dataset with subtracted features

We use all four datasets to evaluate our model.

2.3 Nature of the dataset

When we pair images, we have very few images with the same writer as we have on an average 10 images per writer for the Human Observed Dataset. However, the image pairs for different writers is very large. This makes the dataset skewed and close to the target value 0. Hence, we take equal pairs of same writer features and different writer features. For example, we have 791 same writer pairs in the Human-observed dataset, then we randomly take 791 different writer pairs from the different pairs to make 1592 pairs in total.

Another thing we can try is upscaling the data by generating synthetic data points. However, the model works the same way through the upscaled model also. There is not much difference in the accuracy observed.

Batching of the dataset is done for linear regression and logistic regression for faster performance.

3 Linear Regression (Stochastic Gradient Descent)

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

For performing Linear Regression, we use stochastic gradient descent approach.

As we have a huge number of features that is 512+512 = 1024 features for the GSC dataset concatenation, we use a radial basis function. We use 10 clusters and compute the design matrix and use it for gradient descent.

3.1 Design Matrix

To compute the design matrix, we take a radial basis function. The reason we use a design matrix and a basis function is because there is non-linearity in the data and without using a basis function it is not possible to obtain an optimal result.

We use the gaussian radial basis function which is defined as:

$$\emptyset_1(x_1) = e^{\frac{-(x_1-\mu_1)^2}{2\sigma^2}}$$

As our input data is in the form of vectors, we convert the above formula and use it as:

$$\emptyset_1(x_1) = e^{\frac{-(x_1-\mu_1)^T\Sigma^{-1}(x_1-\mu_1)}{2}}$$

$\emptyset$ is the design matrix and $\emptyset_1(x_1)$ is the value of the first basis function for x_1.
Here, x_1 is the feature vector for the 1st data-point.

We calculate the centroids of the data and all the centroids are represented by μ. Here, μ_1 is the centroid for the first cluster of the data.
Σ^{-1} is the covariance matrix which represents the variance of a particular feature.

Suppose, we have ten basis functions. Then we need to calculate 10 centroids for the data. Hence, we divide the data into 10 clusters. For generating these clusters, we use the simple k-means algorithm. We use k-means because it is a fast converging and simple algorithm.

The dimensions of the design matrix: the number of rows is equal to the number of data points and the number of columns is equal to the number of basis functions. In this case we use 10 basis functions.

The design matrix can be represented by:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix}$$

As we can see, we calculate various basis function outputs for each data point.
The covariance matrix of D dimensions can be represented as:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_D^2 \end{pmatrix}$$

Here, $\sigma_1{}^2$ signifies the variance of the 1st feature. We keep all the other values in the matrix as 0. This is because we don't need the variances between different feature vectors. Also, a bigger variance acts as a general classifier whereas a smaller variance acts as a local classifier. In order to make it a general classifier we do a dot product of the covariance matrix with a random large value.

3.2 Computation of weights

Stochastic gradient descent (often shortened to SGD), also known as incremental gradient descent, is an iterative method for optimizing an objective function.

In this method, we update the weights through multiple iterations to decrease the error of the model. We update the weights based on a parameter known as learning rate.

Hence, this can be represented through the below equation,

$$w_{i+1} = w_i + \eta \Delta \mathrm{w}$$

Suppose we take the case of one such weight w_1,

$$(w_1)^{i+1} = (w_1)^i + \eta \Delta \mathrm{w}_1$$

Where,

$$\Delta \mathrm{w}_1 = \frac{\partial}{\partial w_1} E$$

$$\Delta \mathrm{w}_1 = \frac{\partial}{\partial w_1} \tfrac{1}{2} \left(\mathrm{t}_n - \widehat{t_n}\right)^2$$

$$\Delta \mathrm{w}_1 = \frac{\partial}{\partial w_1} \tfrac{1}{2} \left(\mathrm{t}_n - w^T \emptyset(x)\right)^2$$

$$\Delta \mathrm{w}_1 = \tfrac{1}{2} * -2\left(\mathrm{t}_n - w^T \emptyset(x)\right) \frac{\partial}{\partial w_1} w^T \emptyset(x)$$

$$\Delta \mathrm{w}_1 = -\left(\mathrm{t}_n - w^T \emptyset(x)\right) \frac{\partial}{\partial w_1} \left(w_1 \emptyset_1(x) + w_2 \emptyset_2(x) + \cdots w_n \emptyset_n(x)\right)$$

Hence,

$$\Delta \mathrm{w}_1 = -\left(\mathrm{t}_n - w^T \emptyset(x)\right)\left(\emptyset_1(x)\right)$$

Therefore,

$$(w_1)^{i+1} = (w_1)^i - \eta\left(\mathrm{t}_n - w^T \emptyset(x)\right)\left(\emptyset_1(x)\right)$$

The above formula is the one we use for updating the weights.

3.3 Actual output values and $\mathrm{E}_{\mathrm{RMS}}$

After computing the design matrix and the weights, we compute the actual target values through this equation.

$$t = w^T \emptyset(x)$$

Next, we calculate the root mean square error to judge how well the model is performing.

$$E_{RMS} = \sqrt{2E(w^*)/N_v}$$

Where,

$$E(w^*) = ½\,(\mathrm{t}_n - \widehat{t_n})^2$$

And

$$\widehat{t_n} = w^T \emptyset(x)$$

Hence,

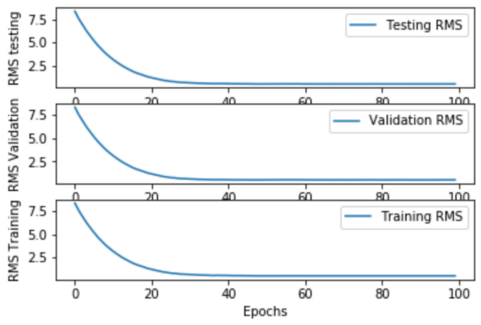
$$E_{RMS} = \sqrt{(\mathrm{t}_n - w^T \emptyset(x))^2/N_v}$$

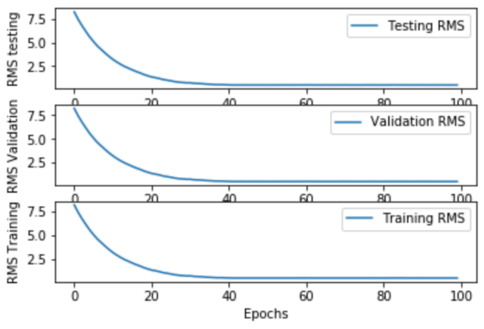
We calculate $\mathrm{E}_{\mathrm{RMS}}$ for the training, testing and the validation data.

3.4 Performance

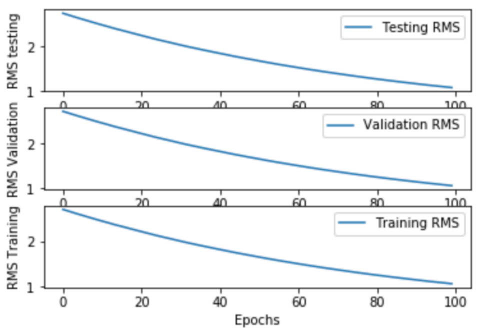
The performance of the data on the linear regression model is given below:

Dataset	Feature Type	ERMS Training	ERMS Validation	ERMS Testing	Accuracy
Human Observed	Concatenated	0.49951	0.49776	0.49849	59.23567
Human Observed	Subtracted	0.49964	0.49607	0.49891	61.78344
GSC	Concatenated	0.59679	0.5997	0.60067	51.15023
GSC	Subtracted	0.48195	0.48591	0.48826	61.57231

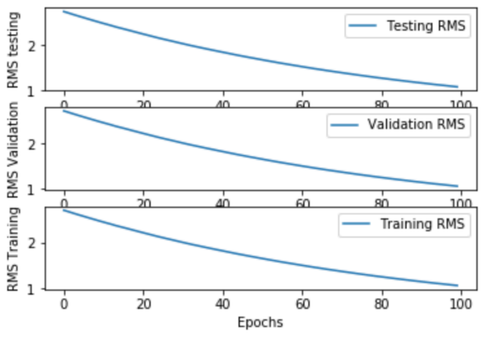




Human Observed Concatenated Human Observed Subtracted



GSC Concatenated



GSC Subtracted

Through these results we can observe that subtracting the data-points gives more accurate output than concatenation in the case of linear regression.

4 Logistic Regression

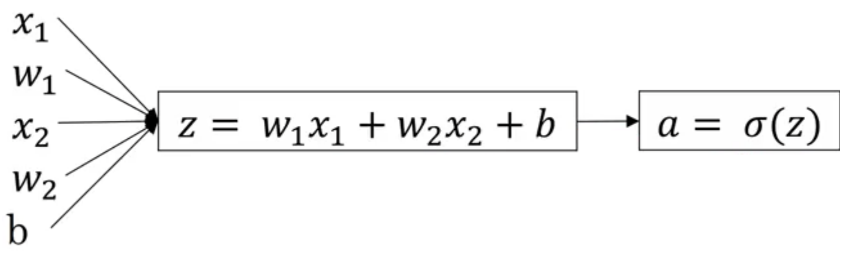
Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more independent variables.
In this case, we have at most 1024 features which we need to map to a binary output.

4.1 Genesis Equation and Computational Graph

$$\hat{y} = \sigma(w^T x)$$

Here 'σ' represents the sigmoid activation function.
'w' is the weight vector and 'x' is the feature vector.

Suppose we have two features X_1 and X_2, then we initialize two weights and a bias b.



Similarly, we can perform the same for multiple features.

4.2 Loss function

We use the cross-entropy loss function.
Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. A perfect model would have a log loss of 0.

$$L = -(yloga + (1-y)\log(1-a)$$

Where $a = \sigma(w^T x)$ and y = actual target value.

4.3 Computation of weights

In order to optimize the logistic regression model, we need to use gradient descent approach. We take gradient of the loss function with respect to the weight and change weights according to the gradient value to reach a minima.

In order to find the gradient $\frac{\partial L}{\partial w}$, we need to apply chain rule.

First, we find $\frac{\partial L}{\partial a}$

$$\frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$$

Where $a = \sigma(w^T x)$

$$\frac{\partial a}{\partial Z} = a(1-a)$$

Because derivative of $\sigma = \sigma(1-\sigma)$

$$\frac{\partial Z}{\partial W_i} = x_i$$

Therefore,

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial a} * \frac{\partial a}{\partial Z} * \frac{\partial Z}{\partial W_i}$$

$$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) * \ a(1-a) * x_i$$

$$= x_i(a-y)$$

$$\Delta \mathrm{w}_1 = \frac{\partial L}{\partial W_i} = x_i(\hat{y} - y)$$

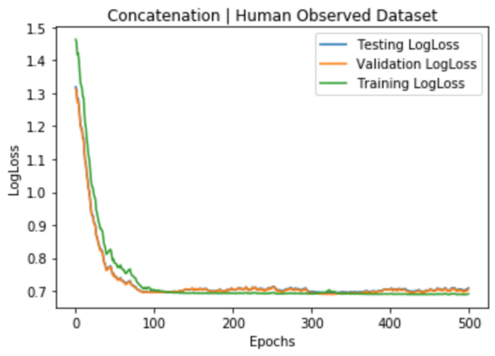
Therefore,

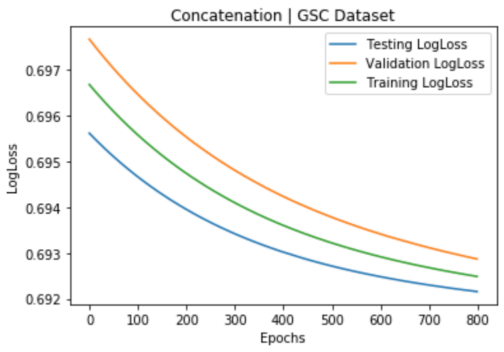
$$(w_i)^{t+1} = (w_i)^t - \eta(x_i(\hat{y} - y))$$

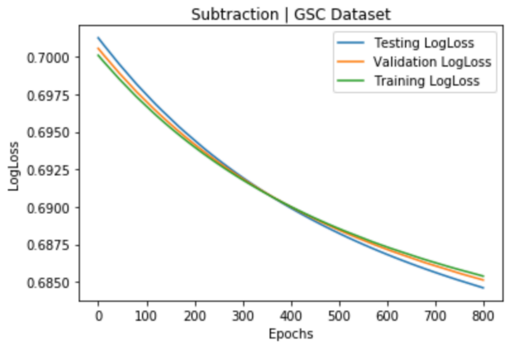
4.4 Performance

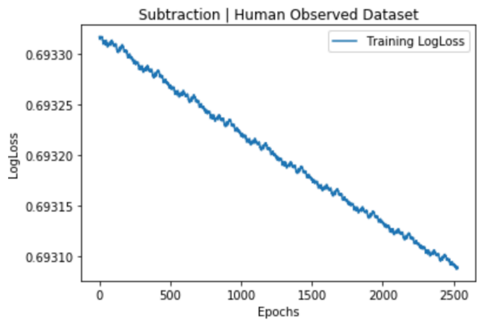
The performance of the data on the logistic regression model is given below:

Dataset	Feature Type	Log Loss Training	Log Loss Validation	Log Loss Testing	Accuracy
Human Observed	Concatenated	0.68474	0.68434	0.69288	57.324
Human Observed	Subtracted	0.69309	0.69265	0.69331	53.797
GSC	Concatenated	0.69249	0.69287	0.69216	50.890
GSC	Subtracted	0.68539	0.68513	0.68461	61.752









Through this we can observe that we get the best accuracy for the GSC dataset using feature subtraction setting.

5 Neural Network

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example.

5.1 Hyperparameters

We discuss some hyperparameters for the neural network and how they affect the performance of the model.

5.1.1 Learning Rate

The learning rate determines how gradually or quickly the weight of the neural network are updated. A high learning rate might not be able to finetune the network, however a low learning rate might become very slow for the parameters to adapt.

5.1.2 Dropout Rate

The dropout rate is defined as the fraction rate of input that is made 0 after each training iteration. This is done to prevent overfitting of data.

5.1.3 Hidden Layers and number of nodes

When a neural network has too few hidden neurons (<64), the network does not have the capacity to learn enough. On increasing the number of nodes to 128 the accuracy increases greatly. However, afterwards even if we increase the nodes by a value of 512, the accuracy increases by only 2%. Hence, we can say that after a point the hidden layer of nodes do not make much difference to the accuracy.

5.1.4 Epochs

One epoch is when an entire dataset is passed forward and backward through the neural network only once. As the number of epochs increases, more the number of times the weights are changed in the neural network and the curve goes from underfitting to optimal to overfitting curve.

5.1.5 Activation Function

An activation function is the output of that node, given an input or set of inputs.The output is usually between [-1,1], depending on the activation function. It decides if the information at the node is relevant or should be ignored.
Relu function: $A(x) = \max(0,x)$
As we can observe, the relu function works best on this data. The reason is that all the activation functions: tanh, sigmoid and others still keep the data sparse. However, relu does not activate around half of the nodes because of its nature.

5.1.6 Loss Function

The loss function is used to measure the inconsistency between actual and predicted output. It is a non-negative value, where the robustness of model increases along with the decrease of the value of loss function.

5.1.7 Hyperparameters used

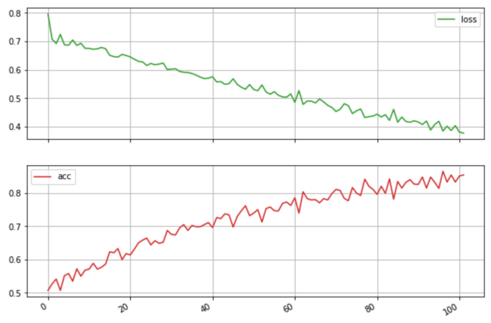
To train the model, we used the following hyperparameters and found that this combination gives the best output.

Hyperparameter	**Value**
Learning Rate	0.001
Dropout Rate	0.2
Epochs	10000
Number of nodes	1024
Activation function	Relu
Loss function	Categorical_Crossentropy

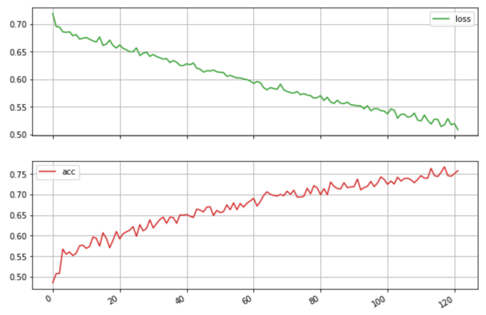
5.2 Performance

The performance of the neural network on the dataset is given below.

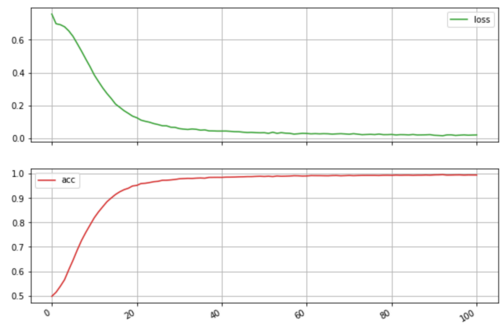
Dataset	Feature Type	Accuracy
Human Observed	Concatenated	82.743
Human Observed	Subtracted	73.514
GSC	Concatenated	80.044
GSC	Subtracted	86.904



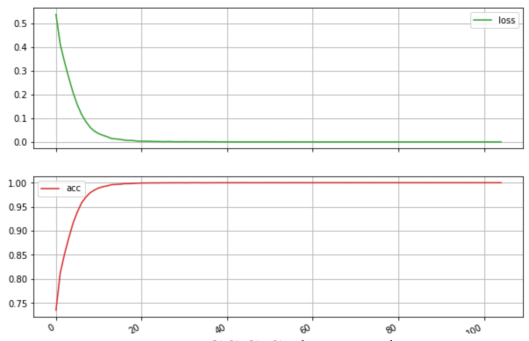
Human Observed Concatenated



Human Observed Subtracted



GSC Concatenated



GSC Subtracted

6 Conclusion

Through this project, we tried to solve the handwriting comparison problem using the CEDAR dataset. We compared the results using Linear regression, Logistic regression and Neural Networks. We found that generally the dataset with the subtracted features did better than the concatenated features. This can be because the dimensions of the feature vectors increase highly when we concatenate the features and it becomes difficult for the model to fit the high dimension data. We achieved better results in neural networks than the regression models. The GSC dataset performed better generally than human observed dataset.

References

[1] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html
[2] https://medium.com/@pushkarmandot/build-your-first-deep-learning-neural-network-model-using-keras-in-python-a90b5864116d
[3] https://docs.scipy.org/doc/numpy-1.15.1/index.html
[4] https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html