ML 574 Project 2

Daniel Amirtharaj

damirtha@buffalo.edu
UB Person Number 50291137

1 Projrect overview

This project aims to develop a predictive model for hand-writing recognition using machine learning. Features extracted for a pair of 'AND' samples in the CEDAR dataset are used to predict whether they are written by the same writer or not. Three approaches have been used to compare their respective advantages/disadvantages, i.e linear regression, logistic regression and neural networks.

Objective

The objective here is to predict if a given pair of 'AND' samples are written by the same writer or not, given feature vectors extracted by both human observations and the GSC algorithm. The following are the tasks involved.

1. Pre-Process the features extracted from human observations as well the GSC algorithm on the 'AND' samples to form feature vectors that represent a pair of these samples.
2. For each same-writer pair assign a target vector with value 1, and for each different writers pair assign a value 0 to the target vector.
3. Split these datasets generated into 3 sets, the training, validation and test sets.
4. Use linear regression and logistic regression with gradient descent to find weights to fit the model.
5. Use Neural networks to solve this as a classification problem.
6. Evaluate ERMS and accuracy of the models using the predicted labels and the given labels for each data sample.
7. Change parameters such as learning rate, number of basis functions and regularization coefficient and other neural network parameters and observe changes in performance, and finally pick model setting with best performance.
8. Using the learned parameters, generate an output file giving the correct classification for any range of inputs.

2 Analysis

Some of the key concepts used in the project have been described briefly here.

2.1 Handwriting analysis and ML

Handwriting analysis is a field vital to forensics and document examinations. Through advancements in handwriting recognition and document analysis with the development of ML, many problems have been solved. Discrepancies and challenges in traditional analysis has been greatly mitigated with the use of predictive models to ensure better accuracy as well as avoiding human error and judgment, in the recent years.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2.2 Linear Regression

Linear regression is one of the most basic of all ML models, but which when used right can help create powerful predictive models. They model predictions by mapping continuous input to continuous output. The output prediction is modeled as a linear function of the model's parameters as well as the input feature vector. An MSE loss function is minimized so the model's parameters predict the target labels as close as possible.

2.2.1 Radial Basis functions

Radial basis functions, add non-linearity to the dataset by transforming the input features into another set of transformed features that are linearly related to the target vector. This is extremely useful while modeling non-linear functions using linear regression. They are computed, by taking the gaussian radial basis function of each data sample centered around a point (centroid of cluster) in the f dimensional feature space of the input dataset.

2.2.2 k-means clustering

k-means is an advanced clustering algorithm that employs unsupervised learning. Here points in the f dimensional feature space are randomly chosen and are grouped into a fixed number of clusters M, each with a centroid such that a data sample in a cluster is closest to the centroid of that cluster. This is useful when similar data has to be grouped together based on certain parameters.

2.3 Logistic Regression

Logistic regression models predictions which are discrete and not continuous. The input data is still assumed to be continuous. It is implemented as an extension to linear regression by applying a sigmoid function to the prediction (linear function of parameters and input features). This compresses the range of the output to values between 0 and 1, which can interpreted as the probability that the model has predicted a certain class in a binary classification problem. This is ensured by using a loss function that maximizes the log-likelihood of label given the model's parameters.

2.4 Neural Networks

The problem given is a supervised classification problem. Supervised, since we already know the output labels, and classification because the output labels are discrete. To solve this problem any classification algorithm can be used. A Neural network has been used here to classify the data. Neural networks is a highly efficient and robust algorithm used widely today. It emulates the human brain in terms of how we learn, and process data. Neural networks can represent complex non-linear functions and have applications in complex learning problems such as computer vision. They are flexible and can also be configured to give better results depending on the problem requirements.

Configuration of Neural Networks The following are parameters of the Neural network classifier that can be tuned to enhance performance,

Neural network model parameters and properties

1. Number of hidden layers
2. Number of neurons in each layer
3. Activation function of each layer
4. Optimization algorithm
5. Error function

Training parameters,

1. Batch size
2. Learning rate
3. Number of epochs

Pre-processing parameters Input data can be processed or transformed to another form in order to improve the effectiveness of the algorithm

2.5 Regularization

Regularization is a tool used to avoid overfitting in machine learning algorithms. Since the training data can be tuned and adjusted to give the maximum performance on a dataset, its performance on the dataset is not reliable, and does not represent how the algorithm will behave with unseen data. In order to prevent the model from giving great training performances but poor ability to generalize, regularization is used. It ensures that the weights learned do not assume disproportionate or abnormally large values, preventing overfitting.

2.6 Gradient descent algorithm

The gradient descent algorithm is used as an alternative solution to solve machine learning problems, and has several advantages over the closed form solution. Since the closed form solution involves taking the inverse of a matrix, it may not always be feasible to use it when solving for large datasets. Gradient descent also works well when modeling non-linear datasets whose cost functions may not be convex and have a single point of convergence.

2.7 Training/Validation/Test sets

The dataset is split into 3 sets, the training, validation and the test sets, usually in the ratio 8:1:1. The training dataset is used solely to train the model, while the validation dataset is used to tune hyper-parameters on the model, and finally the test set is used to measure the model's performance, and this gives a good idea on how well the model has learned, and its ability to generalize over unseen data samples.

3 Methodology

3.1 Feature extraction

Features were extracted from 'AND' samples in the CEDAR dataset using both human observations and GSC algorithm and were readily available before the start of this project.

Since the problem here was to classify whether the same writer has written a pair of 'AND's or if it were different writers, the features extracted in the previous step need to be paired up, so it can be fed to a model for prediction. The following steps were followed here to form the raw data for the model.

1. For each set of features extracted from both the human observed and GSC algorithms, data samples were paired to form new samples, by subtracting features and by concatenating features.
2. Based on the pair formed, that is if it were a same writer pair the target vector's value was set to 1 and if it were a different writers pair the target vector's value was set to 0.
3. The data obtained from the previous steps were consolidated and sampled (800 same writer and different writer pair samples each for human observed features and 10,000 same writer and different writer pair samples each for GSC features) and stored in 4 different .csv files as 4 different datasets.

3.2 Data Preprocessing

Since each dataset generated in the feature extraction step was used to train a model, each dataset was pre-processed separately. The dataset was split into 3 sets, the training, validation and test sets in the ratio of 8:1:1.

3.3 Linear Regression with Gradient descent algorithm

The equations that establish gradient descent were laid out in python with the help of the 'numpy' library. Gradient descent is implemented using fixed steps that move the weights learned along the gradient of the cost function, and this is laid out as a program in python. The basis functions generated as described below were used as the input feature vector for this model.

3.3.1 Generating Basis functions

Using the input features from the training set, M basis functions were generated in the following steps,

1. M cluster centroids were generated either by picking M random samples from the training dataset, or by applying k-means clustering on the f-dimensional feature space of the training set.
2. The variances corresponding to each feature dimension for sample data within a cluster were computed, this could be taken to be unique to each cluster or the uniform spread of the entire input data.
3. The basis for each data sample was computed for each cluster centroid and variance using the gaussian radial function, resulting in a design matrix 'phi' which was then used as the input to the linear regression algorithm.

3.4 Logistic Regression with Gradient descent algorithm

The equations that establish gradient descent for logistic regression were laid out in python with the help of the 'numpy' library. The only difference when compared to gradient descent here was the addition of the sigmoid function in the equation. Gradient descent is implemented using fixed steps that move the weights learned along the gradient of the cost function, and this is laid out as a program in python.

3.5 Neural networks

A neural network with 2 layers and with 100 neurons in each layer was modeled. The equations and links that establish the neural network model were laid out in python with the help of the 'tensorflow' library. Different hyper-parameters were attempted to see which performed best.

3.6 Validation of the model

The model was then validated using different values of M (number of basis functions), lambda (regularization parameter) and learning rate, neural network parameters etc., to see which hyper-parameter gave the best solution and was able to obtain a better model with better performance.

3.7 Testing

Now that the model has been trained with the input dataset, its performance on a dataset it has not encountered yet will be able to give a good idea of how good the model is at generalizing and predicting outputs of new unseen samples.

4 Results

The model was trained using the three models, linear regression, logistic regression and neural networks. Different configurations were chosen to be tested and their results were evaluated. The following sections below highlight the findings.

4.1 Linear regression

When the sampled dataset was run over a logistic regression model, to predict same or different writer pairs, the following were observed for the 4 different set of features used.

4.1.1 Human observed dataset with feature subtraction

The logistic regression algorithm with gradient descent performed best in this dataset for the following hyper-parameters. The dataset used here had all the same writer pairs in the dataset i.e 791 and 800 different writer pairs in the dataset.

Number of basis functions = 100, learning rate = 0.001, lambda = 0.005 and SGD batch size =1.

ERMS Training = 0.50129
ERMS Validation = 0.51158
ERMS Testing = 0.50015

The plot of ERMS with respect to the gradient descent iterations performed is plotted in the below figure.

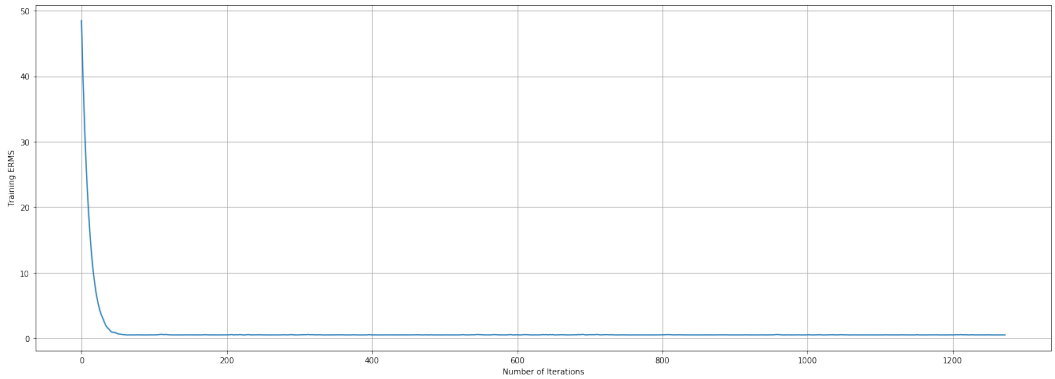

Figure 1: Training ERMS of the linear regression model for Human observed data with feature subtraction

The same model (and hyper-parameters) without radial basis functions performed relatively poorly with the following performance measures,

ERMS Training = 0.80168
ERMS Validation = 0.75469
ERMS Testing = 0.79123

4.1.2 Human observed dataset with feature concatenation

The linear regression algorithm with gradient descent performed best in this dataset for the following hyper-parameters. The dataset used here had all the same writer pairs in the dataset i.e 791 and 800 different writer pairs in the dataset.

Number of basis functions = 100, learning rate = 0.001, lambda = 0.005 and SGD batch size =1.

ERMS Training = 0.501
ERMS Validation = 0.50638
ERMS Testing = 0.50068

The plot of ERMS with respect to the gradient descent iterations performed is plotted in figure 2.

4.1.3 GSC dataset with feature subtraction

The linear regression algorithm with gradient descent performed best in this dataset for the following hyper-parameters. The dataset used here had 10,000 same writer pairs and 10,000 different writer

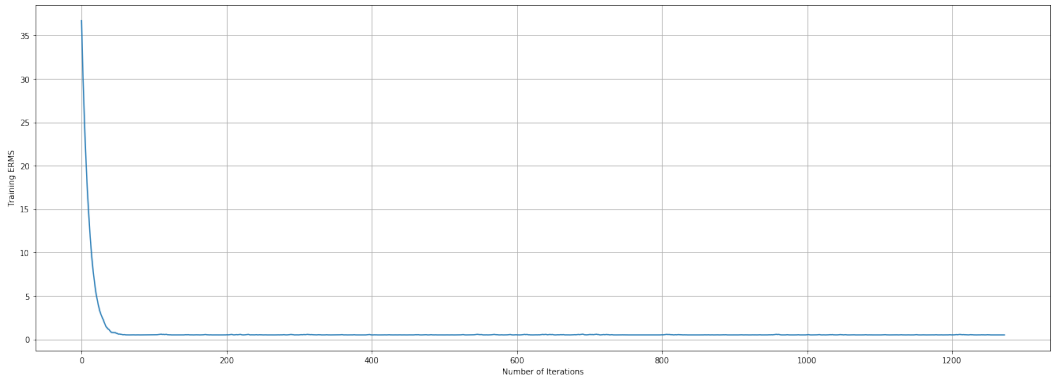

Figure 2: Training ERMS of the linear regression model for Human observed data with feature concatenation

pairs.

Number of basis functions = 100, learning rate = 0.001, lambda = 0.005 and SGD batch size = 20.

ERMS Training = 0.48806
ERMS Validation = 0.48672
ERMS Testing = 0.48261

The plot of ERMS with respect to the gradient descent iterations performed is plotted in figure 3.

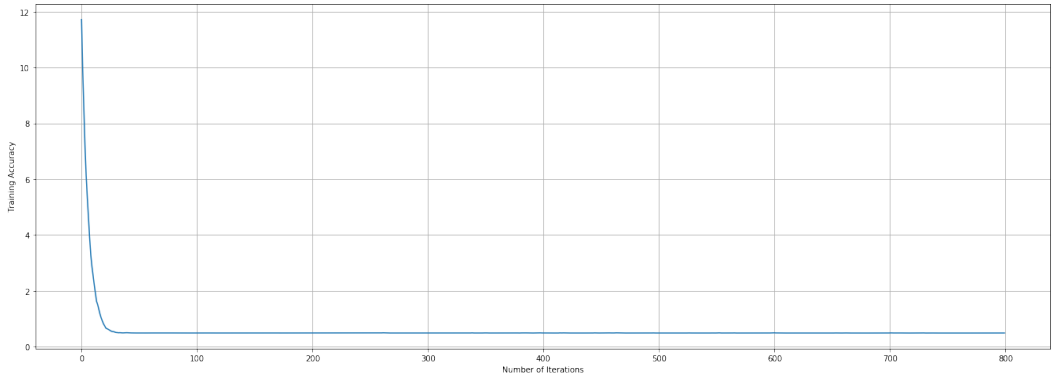

Figure 3: Training ERMS of the linear regression model for GSC data with feature subtraction

4.1.4 GSC dataset with feature concatenation

The linear regression algorithm with gradient descent performed best in this dataset for the following hyper-parameters. The dataset used here had 10,000 same writer pairs and 10,000 different writer pairs.

Number of basis functions = 100, learning rate = 0.001, lambda = 0.005 and SGD batch size = 20.

ERMS Training = 0.5305
ERMS Validation = 0.52842
ERMS Testing = 0.52263

The plot of ERMS with respect to the gradient descent iterations performed is plotted in figure 4.

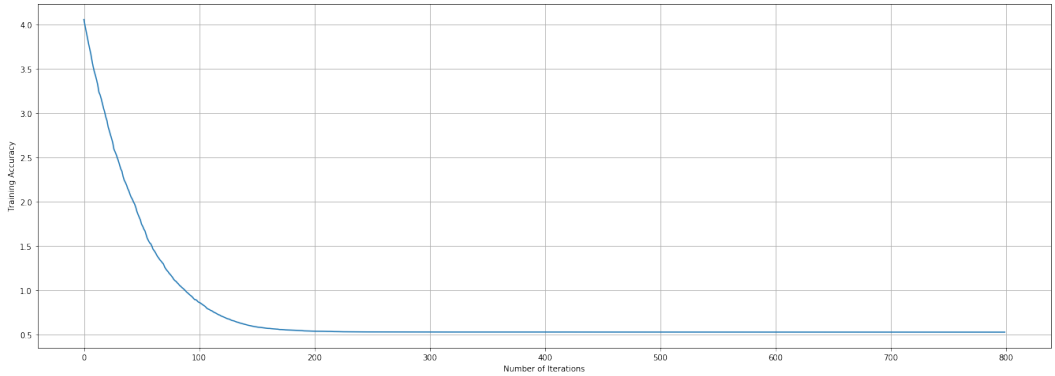

Figure 4: Training ERMS of the linear regression model for GSC data with feature concatenation

4.1.5 Observations

Although all the datasets performed similarly on the linear regression model, it can be observed that the GSC dataset with feature subtraction has slightly lower ERMS compared to the others. This can be attributed to the amount of data samples used for training as well as the quality of features, i.e the amount of information present in the subtracted features outweighed the other mode of feature concatenation. It can also be noted that despite having low ERMS, the model does not actually predict labels effectively, this can be attributed to the fact that the model has tried to fit a linear continuous function on the input dataset to reach the output labels, instead of classifying which label the input should be mapped to, resulting in a continuous output that is not representative of the actual target labels.

4.2 Logistic regression

When the sampled dataset was run over a logistic regression model, to predict same or different writer pairs, the following were observed for the 4 different set of features used.

4.2.1 Human observed dataset with feature subtraction

The logistic regression algorithm with gradient descent performed best in this dataset for the following hyper-parameters. The dataset used here had all the same writer pairs in the dataset i.e 791 and 800 different writer pairs in the dataset.

Learning rate = 0.05, lambda = 0.01 and SGD batch size = 200.

Accuracy Training = 52.71013
Accuracy Validation = 50.31447
Accuracy Testing = 50.0

4.2.2 Human observed dataset with feature concatenation

The logistic regression algorithm with gradient descent performed best in this dataset for the following hyper-parameters. The dataset used here had all the same writer pairs in the dataset i.e 791 and 800 different writer pairs in the dataset.

Learning rate = 0.05, lambda = 0.01 and SGD batch size = 200.

Accuracy Training = 52.23881
Accuracy Validation = 52.20126
Accuracy Testing = 49.36709

4.2.3 GSC dataset with feature subtraction

The logistic regression algorithm with gradient descent performed best in this dataset for the following hyper-parameters. he dataset used here had 10,000 same writer pairs and 10,000 different writer pairs in the dataset.

Learning rate = 0.05, lambda = 0.01 and SGD batch size = 142.

Accuracy Training = 53.90338
Accuracy Validation = 57.86164
Accuracy Testing = 52.89873

4.2.4 GSC dataset with feature concatenation

The logistic regression algorithm with gradient descent performed best in this dataset for the following hyper-parameters. The dataset used here had 10,000 same writer pairs and 10,000 different writer pairs in the dataset.

Learning rate = 0.05, lambda = 0.01 and SGD batch size = 20.

Accuracy Training = 52.55302
Accuracy Validation = 51.57233
Accuracy Testing = 51.28652

4.2.5 Observations

Although all the datasets performed similarly on the logistic regression model, it can be observed that the GSC dataset with feature subtraction has slightly higher accuracy compared to the others. This can be attributed to the amount of data samples used for training as well as the quality of features, i.e the amount of information present in the subtracted features outweighed the other mode of feature concatenation.

Also, logistic regression performed poorly and was hardly able to model the given feature set. Since logistic regression is only a linear classifier, its decision boundaries/surfaces are linear in the respective feature spaces. The information contained in the datasets on the other hand contain highly non-linear mappings which only a much more powerful algorithm can learn from.
The following plot, figure 5 re-iterates the previous statement and shows how the logistic regression model is unable to learn due to its linearity.

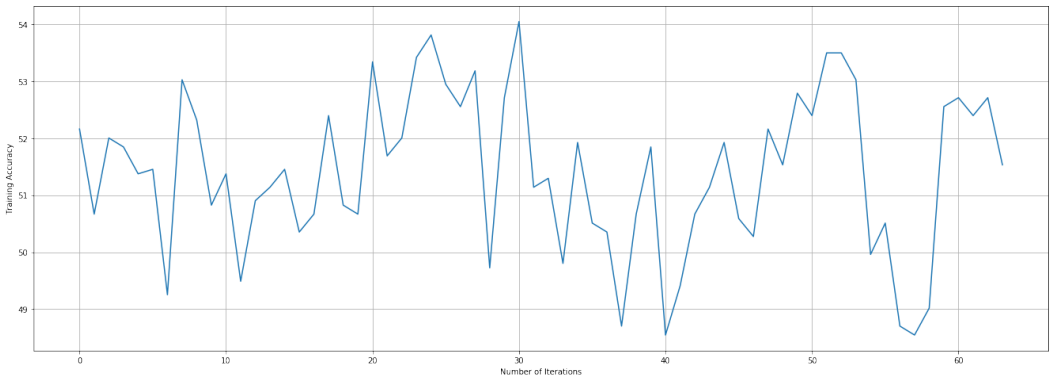

Figure 5: Training Accuracy of the logistic regression model for human observed data with feature subtractions

4.3 Neural networks

When the sampled dataset was run over a Neural networks model, to predict same or different writer pairs, the following were observed for the 4 different set of features used.

4.3.1 Human observed dataset with feature subtraction

The Neural networks algorithm performed best in this dataset for the following hyper-parameters. The dataset used here had all the same writer pairs in the dataset i.e 791 and 800 different writer pairs in the dataset.

Number of hidden layers = 2, number of neurons in each hidden layer = 100, dropout between layer 1 and 2 = 0.2, number of epochs = 1000, batch size = 150 and learning rate = 0.07

Training Accuracy: 98.66457187745483
Testing Accuracy: 52.53164556962025
Errors: 75 Correct :83

The plot of Accuracy with respect to the number of epochs is plotted in figure 6.

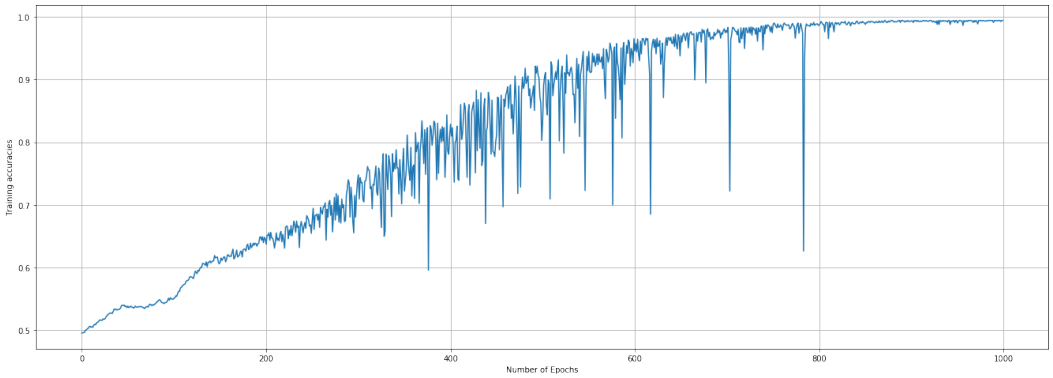

Figure 6: Training accuracy of the neural networks model for Human observed data with feature subtraction

4.3.2 Human observed dataset with feature concatenation

The Neural networks algorithm performed best in this dataset for the following hyper-parameters. The dataset used here had all the same writer pairs in the dataset i.e 791 and 800 different writer pairs in the dataset.

Number of hidden layers = 2, number of neurons in each hidden layer = 100, dropout between layer 1 and 2 = 0.2, number of epochs = 1000, batch size = 150 and learning rate = 0.07

Training Accuracy: 100.0
Testing Accuracy: 52.53164556962025
Errors: 75 Correct :83

The plot of Accuracy with respect to the number of epochs is plotted in figure 7.

4.3.3 GSC dataset with feature subtraction

The Neural networks algorithm performed best in this dataset for the following hyper-parameters. The dataset used here had 10,000 same writer pairs and 10,000 different writer pairs.

Number of hidden layers = 2, number of neurons in each hidden layer = 100, dropout between layer 1 and 2 = 0.2, number of epochs = 1000, batch size = 150 and learning rate = 0.07

Training Accuracy: 100.0

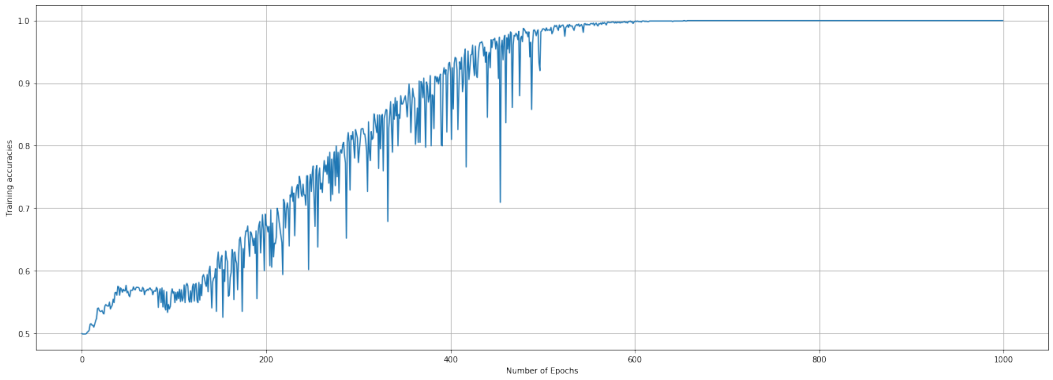

Figure 7: Training accuracy of the neural networks model for Human observed data with feature concatenation

Testing Accuracy: 91.19559779889946
Errors: 176 Correct :1823

The plot of Accuracy with respect to the number of epochs is plotted in figure 8.

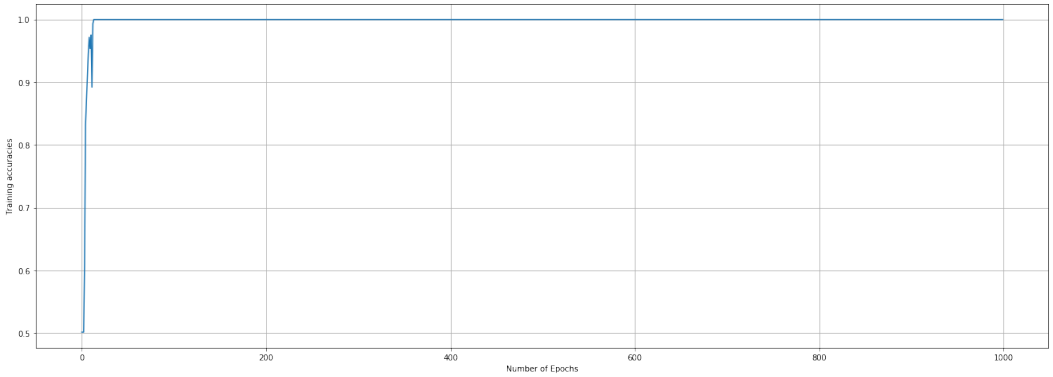

Figure 8: Training accuracy of the neural networks model for GSC data with feature subtraction

4.3.4 GSC dataset with feature concatenation

The Neural networks algorithm performed best in this dataset for the following hyper-parameters. The dataset used here had 10,000 same writer pairs and 10,000 different writer pairs.

Number of hidden layers = 2, number of neurons in each hidden layer = 100, dropout between layer 1 and 2 = 0.2, number of epochs = 1000, batch size = 150 and learning rate = 0.07

Training Accuracy: 100.0
Testing Accuracy: 91.19559779889946
Errors: 176 Correct :1823
The plot of Accuracy with respect to the number of epochs is plotted in figure 9.

4.3.5 Observations

It can be observed that the model is overfitting heavily on the human observed feature data sets, despite having a dropout in the first hidden layer. This can be attributed to the fact that there are very few samples for the network to learn a very non-linear function from. The network thus ends up

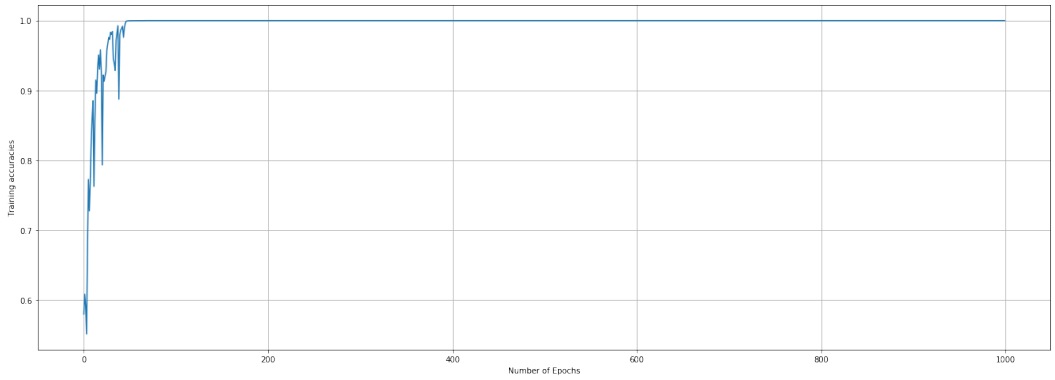

Figure 9: Training accuracy of the neural networks model for GSC data with feature concatenation

learning a non-linear function that is able to represent the training data but is not representative of the test dataset.

On the GSC datasets the model is able to deliver better performance, and high accuracies on both the training and test datasets. Despite having more number of features, the GSC datasets perform better since they have a large number of data samples that help the model learn better about the input data set and its non-linear mapping to the output labels, without over-fitting and estimating a mapping that represents the known dataset but not unseen data.

5 Conclusion

Training the 3 models using the 4 different feature sets has highlighted how models perform differently on different data distributions. Since the given dataset had highly non-linear mapping between input features and output data labels, both linear regression and logistic regression were unable to make reasonable predictions. Even though linear regression used radial basis functions, it simply was not powerful enough to model the actual highly non-linear mapping. Logistic regression on the other hand is a linear classifier, with a linear decision surface, which simply was not good enough for this dataset.

Neural networks outperformed both these models, and have shown how powerful they can be when modeling very non-linear datasets.

References

[1] Numpy documentation
`https://docs.scipy.org/doc/numpy/reference/`

[2] Code posted in UB learns, and used in previous projects.

[3] Gradient Descent Algorithms, by Sebastian Ruder.
`http://ruder.io/optimizing-gradient-descent/`