A Report On Application of Machine Learning Algorithms To Handwriting Comparison Problem

Miki Padhiary
UBID: mikipadh Person Number: 50286289

November 2, 2018

Abstract

In this experiment, We studied how to apply machine learning to solve the handwriting comparison task in forensics. Our objective was to find similarity between the handwritten samples of the known and the questioned writer by using linear regression, logistic regression and neural network solutions. We devised this as a problem of linear regression where we map an input vector x to a real-valued scalar target $y(x, w)$. Experiments on various hyper-parameters such as learning rate, number of basis function, regularization term, dropout, number of hidden layers, were performed for their effect on the accuracy of the program.The best accuracy was achieved using neural networks.

1 Introduction

Following approaches were taken:
1. Trained a model using a closed-form solution.
2. Trained a model using stochastic gradient descent (SGD).
3. Trained a model using logistic regression
4. Trained a model using neural networks.

For the experiment, features were obtained from two different sources:
1. Human Observed features: Features entered by human document examiners manually.
2. GSC features: Features extracted using Gradient Structural Concavity (GSC) algorithm.

The target values are scalars that can take two values 1:same writer, 0:different writers. We had used linear regression to obtain real values which is more useful for finding similarity. Data were preprocessed and splitted into training(80%), validation(10%) and testing(10%) purposes. Training data is the data which is used to learn a model. Validation data is used while testing to check if the training is correctly performed or not. Testing data is the data on which the testing is done.
The following machine learning models were implemented:
1. **Linear Regression Model**

2 Linear Regression Model

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 our case, the linear regression function is defined as:

$$y(x, w) = w^T \phi(x) \tag{1}$$

where $\mathbf{w} = (\mathbf{w}_0,\mathbf{w}_1,...,\mathbf{w}_{m-1})$ is a weight vector to be learned from training samples and $\phi = (\phi_0,\phi_1,...,\phi_{m-1})^\top$ is a vector of $\mathbf{M}$ basis functions.

Gaussian radial basis function is defined as:

$$\phi_j(x) = exp(-\frac{1}{2}(x - \mu_j)^T \Sigma_j^{-1}(x - \mu_j) \tag{2}$$

where μ_j is the center of the basis function and $\mathbf{\Sigma}_j$ decides how broadly the basis function spreads.

Our objective is to minimize the sum-of-squares error

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2 \tag{3}$$

where ϕ is the design matrix

2.1 Closed Form Solution

Closed-form solution with least-squared regularization is defined by

$$\boldsymbol{W}* = (\lambda \boldsymbol{I} + \boldsymbol{\phi}^T\boldsymbol{\phi})^{-1}\boldsymbol{\phi}^T\boldsymbol{t} \tag{4}$$

The RMS Error is calculated using the following formulae:

$$\boldsymbol{E}_{RMS} = \sqrt{\frac{\boldsymbol{E}(w*)}{N_v}} \tag{5}$$

2.2 Stochastic Gradient Descent

Gradient descent is an algorithm that minimizes functions. Given a function defined by a set of parameters, gradient descent starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimize the function. Weight is calculated using the following:

$$\boldsymbol{W}^{(\tau+1)} = \boldsymbol{W}^{(\tau)} + \Delta \boldsymbol{W}^{(\tau)} \tag{6}$$

where $\boldsymbol{\Delta W}^{(\tau)} = -\eta^{\tau} \nabla E$. Here, η^{τ} is the learning rate.

3 Logistic Regression

Logistic regression models the probability of each input belonging to a particular category. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. Logistic regression is a predictive model. A standard logistic function is called sigmoid function and is given as:

$$\boldsymbol{y(x, w)} = \sigma(w^T X) \tag{7}$$

where $\mathbf{w} = (\mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_{m-1})$ is a weight vector to be learned from training samples and $\mathbf{X}$ is the data matrix. Loss Function for Logistic Regression is:

$$\boldsymbol{E_D(w)} = \frac{1}{m}(-t^T log(y(x, w) - (1-t)^T log(1 - y(x, w)] \tag{8}$$

where $\mathbf{t} = (\mathbf{t}_0, \mathbf{t}_1, ..., \mathbf{t}_n)$ is the vector of outputs. We calculate the weight using

$$\boldsymbol{W}^{(\tau+1)} = \boldsymbol{W}^{(\tau)} + \Delta \boldsymbol{W}^{(\tau)} \tag{9}$$

where $\boldsymbol{\Delta W}^{(\tau)} = -\eta^{\tau} \nabla E_D(w)$. Here, η^{τ} is the learning rate.

4 Neural Network

To solve our problem we had created a Densely Connected neural network using the sequential model offered by keras. Neural network uses the examples to automatically infer rules for recognizing handwritten digits. Furthermore, by increasing the number of training examples, the network can learn more about handwriting, and so improve its accuracy.

5 Major Steps Performed

(a) Extract features and target from the dataset.

(b) Delete the features which have 0 variances.

(c) Split the data into training, validation and testing data-set. The training set takes around 80% of the total. The validation set takes about 10% . The testing set takes the rest. The three sets should NOT overlap.

(d) Patch the dataset into closed form solution.

(e) Patch the dataset into gradient descent solution.

(f) Patch the dataset into logistic regression model.

(g) Patch the dataset into neural network model.

6 Experiments

The following hyper-parameters are analyzed and their effect on the error and accuracy of the **Linear Regression model**:

1. Regularization Term λ
2. Choosing Learning Rate η
3. Number of Basis Functions M

Following hyper-parameter are analyzed for their effect on accuracy of the **Logistic Regression model**:

1. Choosing Learning Rate η

Following hyper-parameters are analyzed for their effect on accuracy of **Neural Network**:

1. Change in Number of nodes in hidden layer
2. Change in Drop out
3. Change in activation function
4. Change in batch size
5. Change in optimizers

7 Hyper-parameters at a glance

A hyper-parameter is a parameter whose value is set before the learning process begins. Different model training algorithms require different hyper-parameters. The training algorithm learns the parameters from the data. For our situation, the following hyper-parameters were tuned in order to increase the accuracy and decrease the error.

Regularization Term λ: Regularization is a technique used to solve over-fitting problems. It is penalizing the loss function by adding a multiple of L norms of the weight vector v. Regularization tuning can be performed by **cross-validation**. Cross Validation is dividing the training data, train the model for a fixed value of λ and test it on remaining subsets. Repeat the process while varying λ. Select the best λ that minimizes the error function[*Refer Figure 1*]

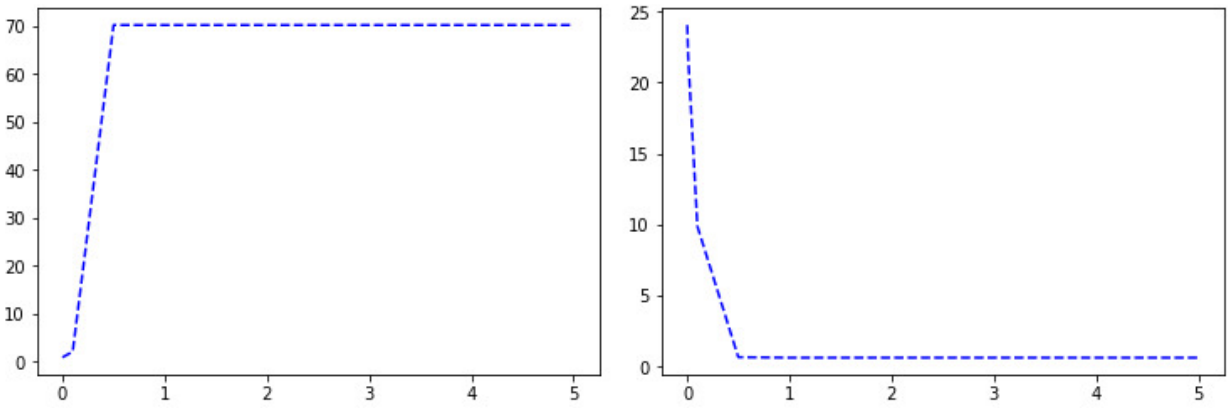

Figure 1: Accuracy Vs Error for Regularization Function

Number of Basis Functions M: The space T is divided into **K** clusters $\{C_1, C_2, ...C_k\}$ that corresponds to the number of basis functions.The centroid of these clusters form the μ_j vector. As data-set is not linear we use basis functions, which introduces non linearity. [*Refer Figure 2*]

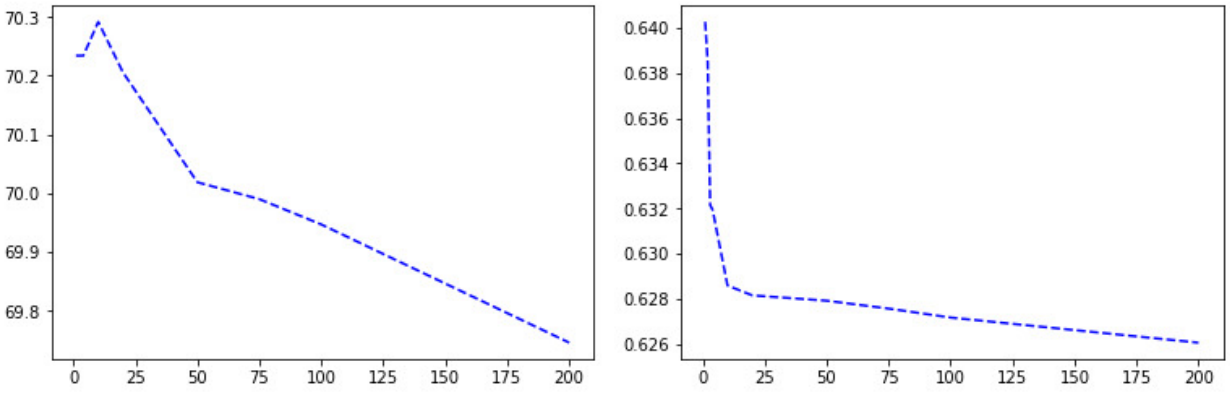

Figure 2: Accuracy Vs Error for basis Function

Learning Rate η: Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect to the loss gradient. The lower the value, the slower we travel along the downward slope[*Refer figure 3*]

Number of nodes in hidden layer: Hidden nodes have two important characteristics. First, they only receive input from the other nodes, such as input or preceding hidden nodes. Second, they only output to other nodes, either as output or other, following hidden nodes. Hidden nodes are not directly connected to the incoming data or to the eventual output. They are often grouped into fully connected hidden layers.

Drop Out: Dropout is a regularization technique for reducing overfitting in neural net-

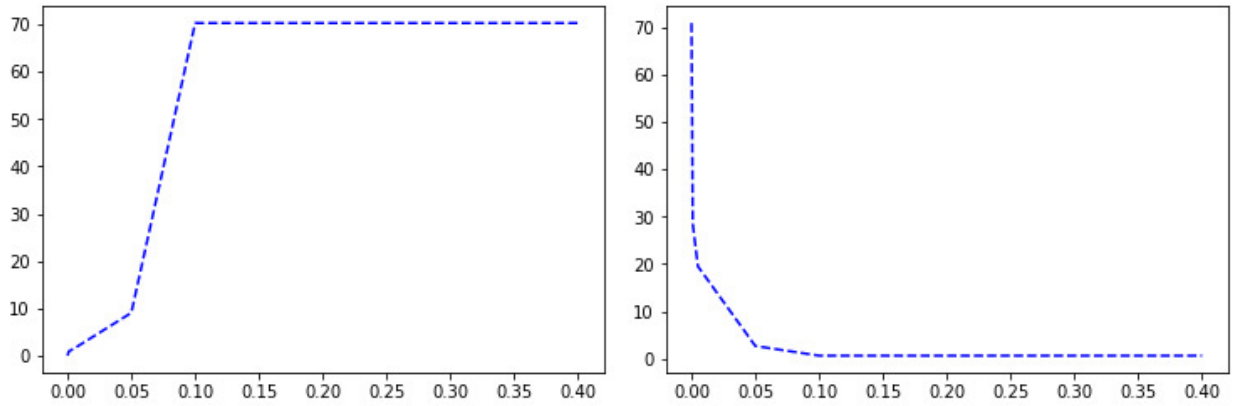

Figure 3: Accuracy Vs Error for Learning rate

works by preventing complex co-adaptations on training data. The drop out rate is set to be 0.2.

Activation Function: The activation function of a node defines the output of that node, or "neuron," given an input or set of inputs. This output is then used as input for the next node and so on until a desired solution to the original problem is found.

Batch Size: Batch Size refers to the number of training examples utilised in one iteration. The default batch size is 32.

Optimizers: Optimizers shape and mold the model into its most accurate possible form by dealing continuously with the weights. They help in minimizing the Error functions.

8 Investigations

The best accuracy and error retrieved after changing various parameters are described below.

8.1 Linear Regression

8.1.1 Regularization Term λ:

The program is executed for λ values ranging from 0.001 to 5 and the graphs obtained is plotted[*Refer figure 1*].
It is seen that for lower values of , the accuracy is low in the case of Closed Form. However as the value is increased, the accuracy increases but after a certain point it does not increase more for both Closed Form and Stochastic Gradient Descent solutions.
Reason When λ is low, the overfitting is high and as a result error is high and accuracy goes for a toll. As we increase λ, the accuracy increases and λ is stabilized.

8.1.2 Number of Basis Functions M:

The program is executed for M values ranging from 1 to 200 and the graphs obtained is plotted above[*Refer figure 2*]. It has been observed that decreasing M to 1 does not have any impact on accuracy nor on the error. However, increasing it to 200, decreases both the accuracy and error values.
Reason This could be attributed to the fact that an increase in number of clusters reduces the variance, Σ for both Closed Form and Stochastic Gradient Descent solutions.

8.1.3 Learning Rate η:

This is only applicable to Gradient Descent Solution. The program is executed for η values ranging from 0.00001 to 0.4 and the graphs obtained is plotted[*Refer figure 3*]. In the perfect case, if learning rate is increased then the error should decrease and the accuracy should increase. In our current scenario, as we go on increasing learning rate, the accuracy increases for a while and then it becomes constant. The error value goes on decreasing with increase in learning rate.

8.2 Neural Network:

8.2.1 Number of nodes in hidden layer:

The program is executed for different values of Hidden layer nodes between 50 to 800 and the accuracy of the program on the testing data is plotted [*Refer Figure 4(left)*]. There is no relation between hidden layer and accuracy as per the graph observed , but for a value of 256 it gave the higher accuracy.

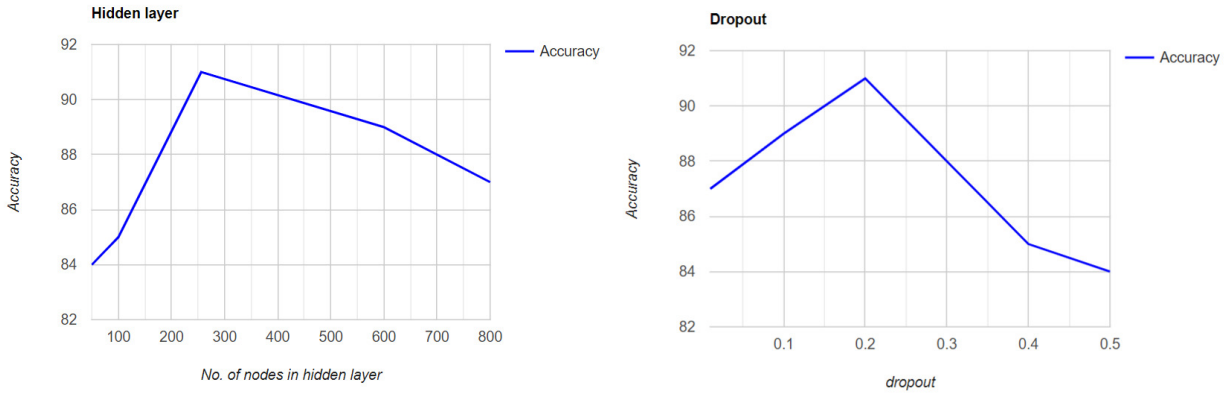

Figure 4: Hidden Layer and Dropout

8.2.2 Drop Out:

The program is executed for different values of Hidden layer nodes between 0.01 to 0.5 and the accuracy of the program on the testing data is plotted [*Refer Figure 4(right)*]. Accuracy

decreases in increasing the dropout. A value of 0.2 in the dropout provides the maximum accuracy.

8.2.3 Activation Function:

Different optimizers such as ReLU , Sigmoid and softmax were tested and the accuracy was reported to be 81%, 91%, 87% respectively[*Refer figure 5(left)*].

8.2.4 Batch Size:

The accuracy remained constant for all the different values of batch sizes from 16 to 128.

8.2.5 Optimizers:

Different Optimizer functions such as AdaDelta, SGD, RMSprop, Adam and Adagrad were tested. The corresponding accuracies were 91%, 60%, 86%, 89%, 83%[*Refer figure 5(right*].

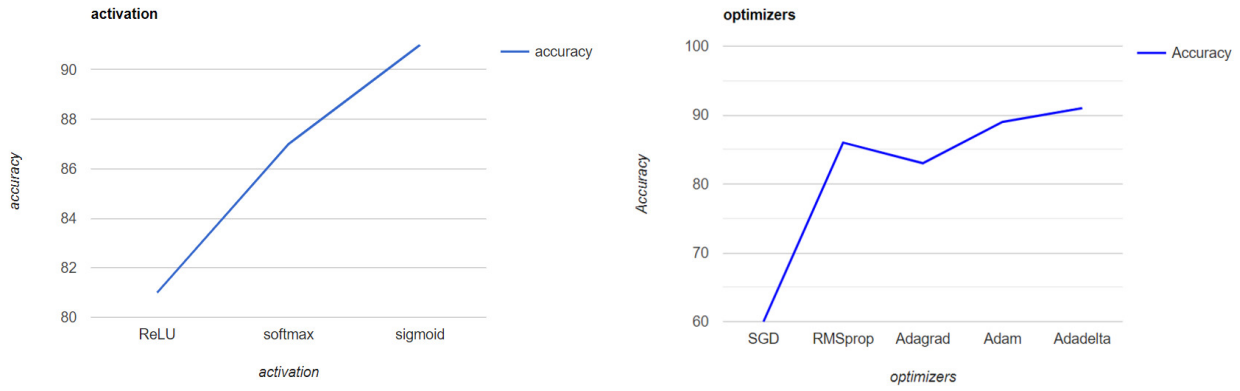

Figure 5: Activation and Optimizer Functions

9 Results

9.1 Comparison between Linear Regression, Logistic Regression and Neural Networks

This section mentions the comparison between the various machine learning models tested. The models were tested on both Human Observed Dataset and GSC Dataset for feature concatenation and feature subtraction.

9.1.1 Linear Regression:

Accuracy obtained using Linear Regression on both Human Observed Feature Data set and GSC Feature Data were not upto the mark for both the features i.e concatenation and

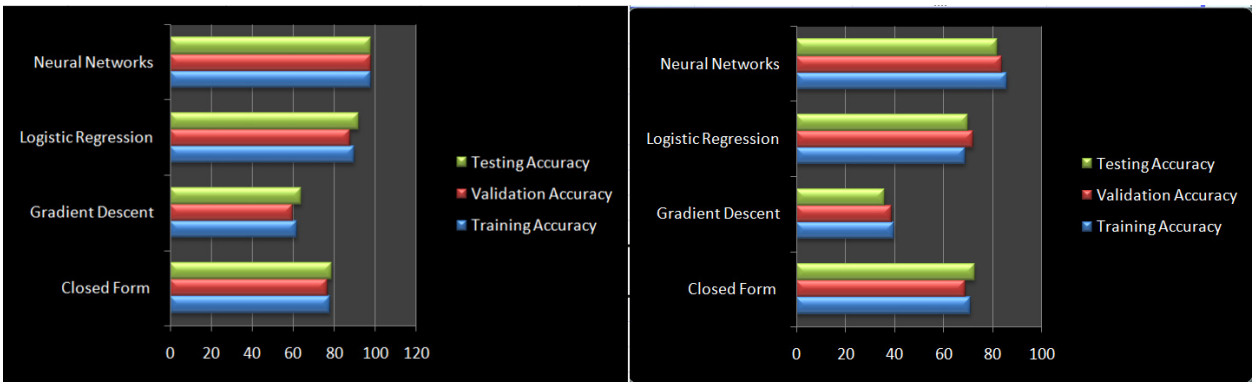

Figure 6: Accuracy of different models on Human Observed Dataset for Feature Concatenation(left) and Feature Subtraction(right)

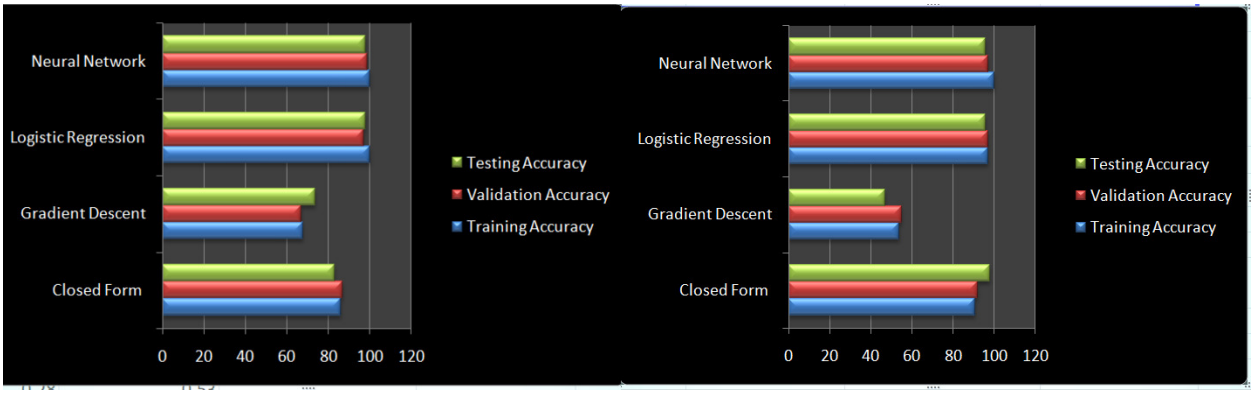

Figure 7: Accuracy of different models on GSC Dataset for Feature Concatenation(left) and Feature Subtraction(right)

subtraction. Accuracy obtained from **Closed Form** was still good but **stochastic gradient descent** performed poorly[*Refer figure 6 and 7*]

The $\mathbf{E_{RMS}}$ values obtained for linear regression models are depicted below:

DataSet Type	Human Observed Dataset	GSC Dataset
Closed Form: Training	0.41	0.32
Closed Form: Validation	0.42	0.33
Closed Form: Testing	0.41	0.35
Gradient Descent: Training	0.51	0.50
Gradient Descent: Validation	0.52	0.51
Gradient Descent: Testing	0.50	0.42

Table 1: Feature Concatenation

DataSet Type	Human Observed Dataset	GSC Dataset
Closed Form: Training	0.45	0.28
Closed Form: Validation	0.45	0.28
Closed Form: Testing	0.44	0.30
Gradient Descent: Training	0.55	0.53
Gradient Descent: Validation	0.55	0.52
Gradient Descent: Testing	0.56	0.55

Table 2: Feature Subtraction

9.1.2 Logistic Regression:

Logistic Regression performed well on both Human Observed Feature Data set and GSC Feature Data than Linear Regression. However, the accuracy was not 100%. For feature addition the accuracy was around 92% and for feature subtraction the accuracy was 70% for Human Observed Dataset. Similary for GSC Dataset, feature addition predicted an accuracy of 98% and subtraction gave an accuracy of 97%. It can be said that Logistic Regression performed exceptionally well on GSC Dataset [*Refer figure 6 and 7*]

9.1.3 Neural Network:

The best accuracy was achieved using neural network solution. For human observed dataset, the accuracy prediction for concatenation feature was 98% and for subtraction feature was 86% respectively. The accuracy on GSC dataset was also higher(98% and 96%) and better than the previous solutions discussed.[*Refer figure 6 and 7*]

10 Conclusions

In this report, three Machine Learning method namely Linear Regression, Logistic Regression, and Neural network were studied to solve Handwriting Comparison problem. Various changes to hyper-parameters were done to increase the accuracy and simultaneously minimize the sum-of-squares error. The best accuracy of 64%, 92% and 98% was achieved on training, validation and testing data set for Human Observed Pairs for feature concatenation. Similarly, an accuracy of 74%, 98% and 98% was achieved on training, validation and testing data set of GSC for feature concatenation. For feature subtraction, the training, validation and testing accuracy were reported to be of 36%, 70%, and 82% for human observed dataset. For GSC,feature subtraction the values are 47%, 96%, and 96% for corresponding training, validation and testing dataset. It is also observed that feature concatenation performs better than feature subtraction.

References

[1] Linear Regression
https://en.wikipedia.org/wiki/Linear_regression

[2] Regularization
https://codeburst.io/what-is-regularization-in-machine-learning-aed5a1c36590

[3] Neural Networks
http://neuralnetworksanddeeplearning.com/chap1.html/

[4] Project Description
https://ublearns.buffalo.edu/bbcswebdav/pid-4735901-dt-content-rid-20615288_1/courses/2189_24904_COMB/Project2-Description%5BUpdated%5D%281%29.pdf

[5] Gradient Descent
https://en.wikipedia.org/wiki/Gradient_descent

[6] Hyper-parameters
https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)

[7] Learning Rate and its impact
https://towardsdatascience.com/understanding-learning-rates-and\-how-it-improves-performance-in-deep-learning-d0d4059c1c10