

# CSE 574 – INTRODUCTION TO MACHINE LEARNING – PROJECT 2

Submitted By: Keerthana Baskaran

UB# 50288944

## Contents:

1. *Introduction*
  - *Given Dataset*
2. *Implementation*
  - *Preprocessing the data*
3. *Experimental results*
  - *Linear regression model*
  - *Logistic regression model*
  - *Neural network model*
4. *Inference*
5. *Conclusion*

## 1. Introduction

The goal of the project requires you to apply machine learning to solve the handwriting comparison task in forensics. We formulate this as a problem of linear regression where we map a set of input features  $x$  to a real-valued scalar target  $y(x,w)$ . The task is to find similarity between the handwritten samples of the known and the questioned writer by using linear regression. Each instance in the CEDAR “AND” training data consists of set of input features for each handwritten “AND” sample. The features are 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}.

## Given 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.

## 2. Implementation

### 2.1 Preprocessing the dataset

The given dataset is human observed data and the GSC data. Human observed has same pair and different pair files and similarly GSC.

#### 2.1.1 Preprocessing the human observed dataset

Initially we must take the same pair file. For each image ID we must take the corresponding feature from the human observed feature file. Here there are two solutions.

1. We must concatenate the two features and train the model with that. We have 9 features for each image ID, when we concatenate we get 18 features.
2. The next one is subtracting the two features and use it to train the model. We will get 9 features here after subtraction

The number of records for same pair is 791. We use the same number of records from different pair to train the model accurately, So the total number of records will be  $791+791=1582$  records and features will be  $9+9=18$  for concatenation. For subtraction the number of records will be 1582 but the features will be 9.

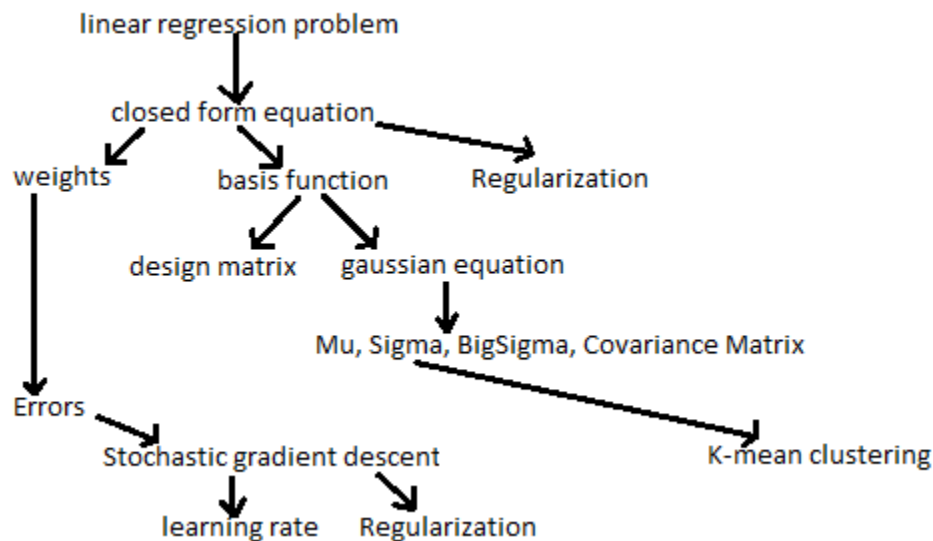
### 2.1.2 Preprocessing the GSC dataset

Like the human observed data, we must preprocess the GSC dataset. In GSC dataset each image has 512 features. After concatenation we get 1024 features. And after subtraction we get 512 features. Since the dataset is extremely large here also we randomly pick 70000 records from the entire dataset.

1. For concatenation the dimension of the dataset is  $1024 \times 70k$ . Here we need to check if there are any **zero columns (features with zero value)** in the dataset before proceeding. If few columns are zero, we have to delete them before giving input to the phi matrix
2. For subtraction the dimension of the dataset is  $512 \times 70k$ . The check for zero column in the dataset needs to be done and if anything is there then dropping those columns.

After preparing the dataset we split the dataset for training, testing and validation.

## 2.2 Linear Regression model



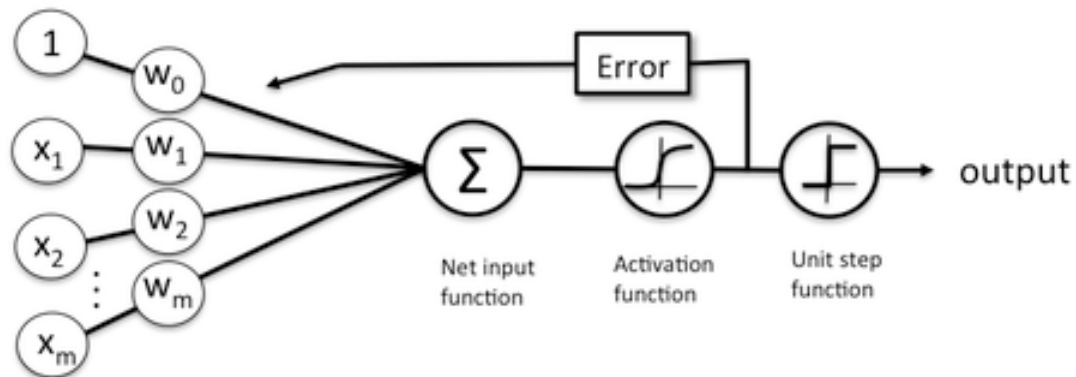
### 2.2.1 Closed form solution

1. We use two approaches, first we do the closed form solution and the first step in closed form solution is to define the number of basis function used in solving the problem. The data usually are not linear and cannot be modeled using linear function. So, we use **Radial**

**Basis function** to introduce a non-linearity component in the data. We create multiple Gaussian basis functions and then fit a linear model for each of these functions.

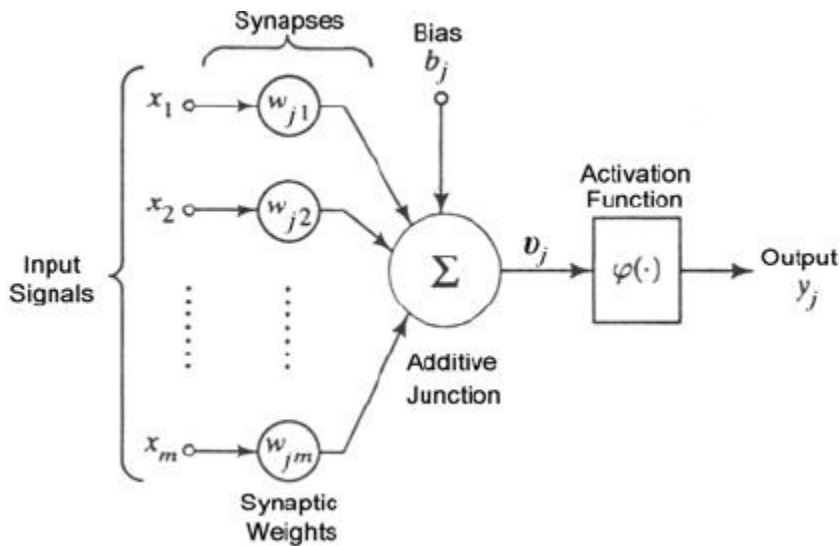
2. The next step is to formulate the **design matrix** using gaussian normal distribution. We have to choose the center for basis function. Here we use M basis functions instead of one polynomial function by dividing the feature space into M regions and applying M basis functions on input vectors. We use Gaussian equation to find center by dividing the space into M clusters. These clusters are found using the **k-means algorithm**.
3. We then find the **weights** by using the closed form likelihood solution and compute the model complexity and lambda is additional regularization term used. The error (**Erms**) is found by calculating the difference between the predicted value and the actual value. The goal is to minimize this error. We use **stochastic gradient descent** solution to minimize this error by using learning rate.

### 2.3 Logistics Regression Model



1. Logistic regression is a **classification algorithm** used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic **sigmoid function** to return a probability value which can then be mapped to two or more discrete classes.
2. We use binary logistic regression on the dataset and apply sigmoid function to get from a range of **0 to 1**.
3. The **genesis equation** is given by target = sigmoid x (weights and the input). Initially we randomly **initialize** the weights and send these weights along with input to the model. We compute the sigmoid value and the learning rate.
4. According to the computational graphs, we **backtrack** from the learning rate to find the loss function by taking derivative with respect to the weights.
5. For each epoch we find the loss or the errors and then keep **updating** the weights using the gradient descent solution.

### 2.4 Neural Network



1. We are processing the data using neural network. A neural network has **input layer, hidden and output layer**. Layers are made of nodes connected by neurons. Node is the place where computation happens.
2. We use **activation function** to eliminate linearity in the network and improve accuracy. **Loss function** is the sum of squared errors. We use adam,rmsprop etc.
3. To stop **overfitting** we use earlystopping and dropout rate

### 3. Experimental results

#### 3.1 Linear Regression

##### Hyperparameter setup for Linear Regression

Lambda is 0.03 for closed form and 0.01 for gradient descent, training percent =80, validation percent =10, M=10, Test percent =10

##### 3.1.2 Human observed dataset (concatenation of features)

Training the linear regression model for human observed data with same and different pair features concatenated.

Solution method	E_rms Training	E_rms Validation	E_rms testing
Closed form solution	0.48	0.62	0.62
Gradient descent solution	0.51	0.63	0.64

```
IPython console
Console 1/A

-----Human observed data (concatenation)-----
UBITname      = kbaskara
Person Number = 50288944
-----
-----LINEAR REGRESSION-----
-----Closed Form with Radial Basis Function-----
E_rms Training   = 0.4824859983988128
E_rms Validation = 0.6248537798367917
E_rms Testing    = 0.6243935954045643
-----
-----Please Wait for 2 mins!-----
-----
-----Gradient Descent Solution-----
E_rms Training   = 0.51133
E_rms Validation = 0.66494
E_rms Testing    = 0.66798
```

### Observation

The machine gives a **high error rate** for human observed dataset of concatenated features. We have taken **only 791** records of the entire different pair dataset which is less for the model to learn accurately. This is one of the reason that the model throws high error rate. By reducing the number of records to 500 the model gives a **greater error rate** of around 0.7. This shows that there is **no enough data** in human observed for the model to predict accurately.

#### 3.1.1 Human observed dataset (subtraction of features)

Training the model for human observed data with same and different pair features subtracted.

Solution method	E_rms Training	E_rms Validation	E_rms testing
Closed form solution	0.48	0.62	0.62
Gradient descent solution	0.51	0.73	0.70

```
IPython console
Console 1/A x
-----Human observed data (subtraction)-----
UBITname      = kbaskara
Person Number = 50288944
-----
-----LINEAR REGRESSION-----
-----Closed Form with Radial Basis Function-----
E_rms Training   = 0.482890842781911
E_rms Validation = 0.6256264251032059
E_rms Testing    = 0.6241634429558806
-----
-----Please Wait for 2 mins!-----
-----
-----Gradient Descent Solution-----
E_rms Training   = 0.5076
E_rms Validation = 0.73469
E_rms Testing    = 0.70907
```

## Observation

The error rate is high in human observed dataset with subtracted features. This is same as the human observed with concatenated features. Because **the less data** the model does not learn accurately.

### 3.1.2 GSC dataset (concatenation of features)

Training the model for GSC observed data with same and different pair features concatenated.

Solution method	E_rms Training	E_rms Validation	E_rms testing
Closed form solution	0.38	0.37	0.45
Gradient descent solution	0.38	0.39	0.44

```
IPython console
Console 1/A x
E_rms Training = 0.70507
-----GSC data (concatenation)-----
UBITname      = kbaskara
Person Number = 50288944
-----LINEAR REGRESSION-----
-----Closed Form with Radial Basis Function-----
E_rms Training  = 0.3806073157669986
E_rms Validation = 0.3797734628366294
E_rms Testing   = 0.45014012125294794
-----Please Wait for 2 mins!-----
-----Gradient Descent Solution-----
E_rms Training  = 0.39269
E_rms Validation = 0.39166
E_rms Testing   = 0.44859
```

## Observation

For the GSC dataset, in concatenated features, the error rate is much **lesser than** the human observed concatenated features. Here we have taken more than **70,000 data** to train the model. The given dataset is extremely large so have picked up an **optimal value** with which the model is able to predict properly. By reducing the number of data to be used for training the model gives higher error rate.

### 3.1.3 GSC dataset (Subtraction of features)

Training the linear regression model for GSC observed data with same and different pair features subtracted

Solution method	E_rms Training	E_rms Validation	E_rms testing
Closed form solution	0.33	0.33	0.40
Gradient descent solution	0.54	0.52	0.56

```
IPython console
Console 1/A x
E_rms Training = 0.44055
-----GSC data (subtraction)-----
UBITname      = kbaskara
Person Number = 50288944
-----LINEAR REGRESSION-----
-----Closed Form with Radial Basis Function-----
E_rms Training  = 0.3341290585122744
E_rms Validation = 0.3341060531690352
E_rms Testing   = 0.40519170627153694
-----Please Wait for 2 mins!-----
-----Gradient Descent Solution-----
E_rms Training  = 0.54224
E_rms Validation = 0.52679
E_rms Testing   = 0.5652
```

## Observation

The error rate for GSC dataset with features subtracted is **greater than** that of GSC concatenated. Both uses same set of data around 70,000 but the only difference is that the features are subtracted here. The model **shows greater error** in this case than the concatenated (both in human observed and GSC), which shows that the model **learns well in concatenated** features dataset.

## 3.2 Logistic Regression

### Hyperparameter setting for logistic regression

Lambda is 0.01 for gradient descent, training percent =80, validation percent =10, M=10, Test percent =10

#### 3.2.1 Human observed features (concatenation of features)

Training the logistic regression model with human observed dataset of concatenated features of same and different pair.

Error in training dataset	0.48
Error in Validation dataset	0.01
Error in testing dataset	0.01



```
IPython console
Console 1/A x
-----Human observed concatenation-----
UBITname      = KBASKARA
Person Number = 50288944
-----Logistical Regression-----
-----Gradient Descent Solution-----
Error Training  = 0.48379
Error Validation = 0.01875
Error Testing   = 0.01877
```

## Observation

Here in human observed data with concatenated same and different pair features the input data size is 791. Which means we have taken the entire dataset and trained the model. Because of this the model is able to predict properly.

### 3.2.2 Human observed features (subtraction of features)

Training the logistic regression model with human observed dataset of subtracted features of same and different pair.

Error in training dataset	0.48
Error in Validation dataset	0.02
Error in testing dataset	0.02

```
IPython console
Console 1/A x
-----Human observed subtraction-----
UBITname      = KBASKARA
Person Number = 50288944
-----Logistical Regression-----
-----Gradient Descent Solution-----
Error Training  = 0.48522
Error Validation = 0.02092
Error Testing   = 0.02094
```

## Observation

The model is able to predict accurately in human observed subtracted dataset with less error rate.

### 3.2.3 GSC dataset (concatenation of features)

Training the logistic regression model with GSC dataset of concatenated features of same and different pair.

Error in training dataset	0.38
Error in Validation dataset	0.37

Error in testing dataset	0.43
--------------------------	------

```

IPython console
Console 1/A
-----GSC dataset concatenation-----
UBITname      = KBASKARA
Person Number = 50288944
-----Logistical Regression-----
-----Gradient Descent Solution-----
Error Training  = 0.38109
Error Validation = 0.37955
Error Testing   = 0.43745

```

### Observation

The model gives less error rate than linear regression for GSC dataset with concatenated features. Which shows that logistic regression is better than linear regression for this dataset.

### 3.2.4 GSC dataset (subtraction of features)

Training the logistic regression model with GSC dataset of subtracted features of same and different pair.

Error in training dataset	0.34
Error in Validation dataset	0.34
Error in testing dataset	0.38

```

IPython console
Console 1/A
-----GSC Dataset subtraction-----
UBITname      = KBASKARA
Person Number = 50288944
-----Logistical Regression-----
-----Gradient Descent Solution-----
Error Training  = 0.34243
Error Validation = 0.34152
Error Testing   = 0.38752

```

### Observation

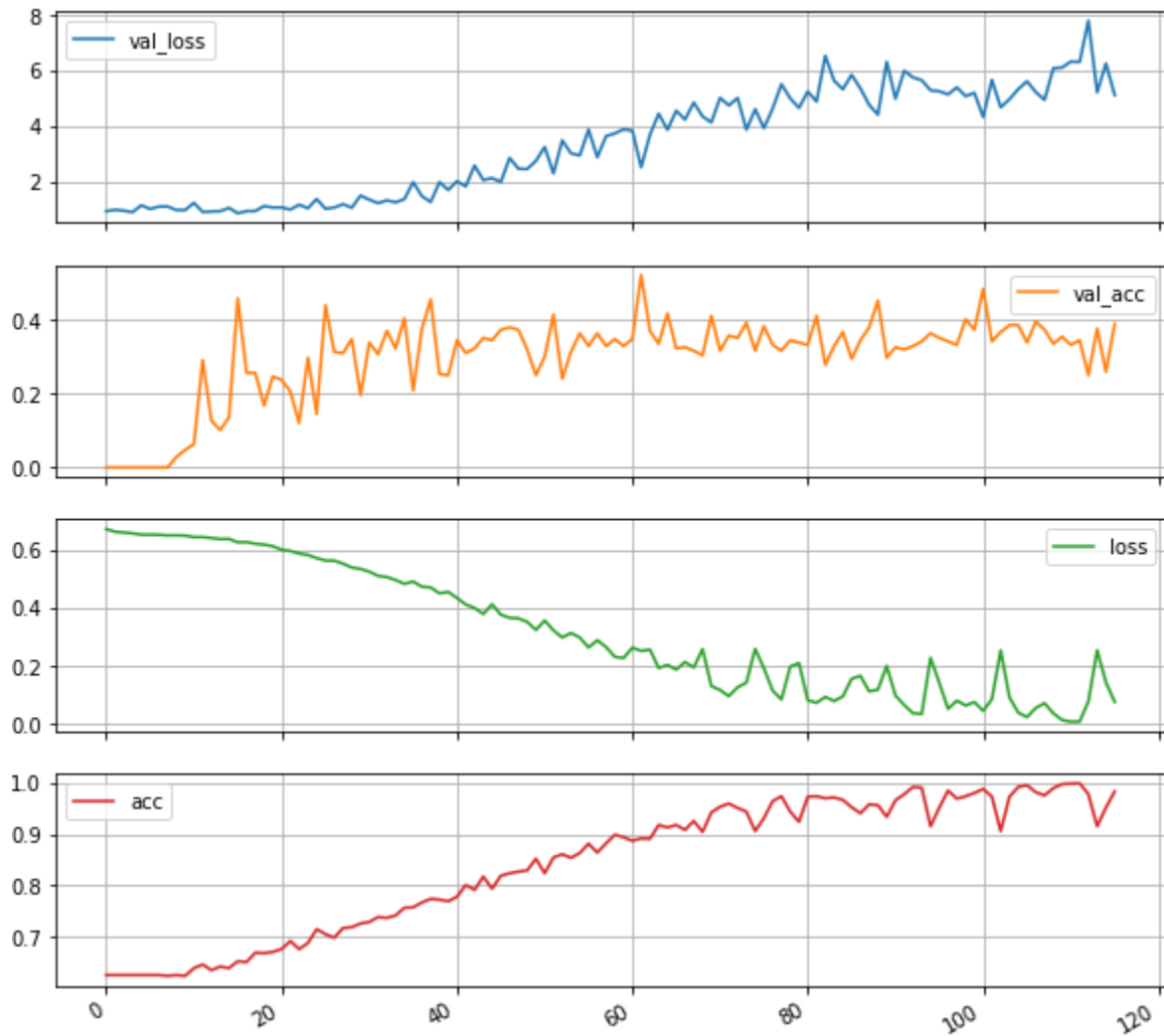
Error rate is very less than given in linear regression for same GSC dataset of subtracted features.

## 3.3 Neural network

### Hyperparameter setting for neural network

Validation split=0.2, batch size=10, epoch =1000, early patience =100, number of layers=3, activation function=relu and softmax, optimizer is adam, loss function is binary crossentropy

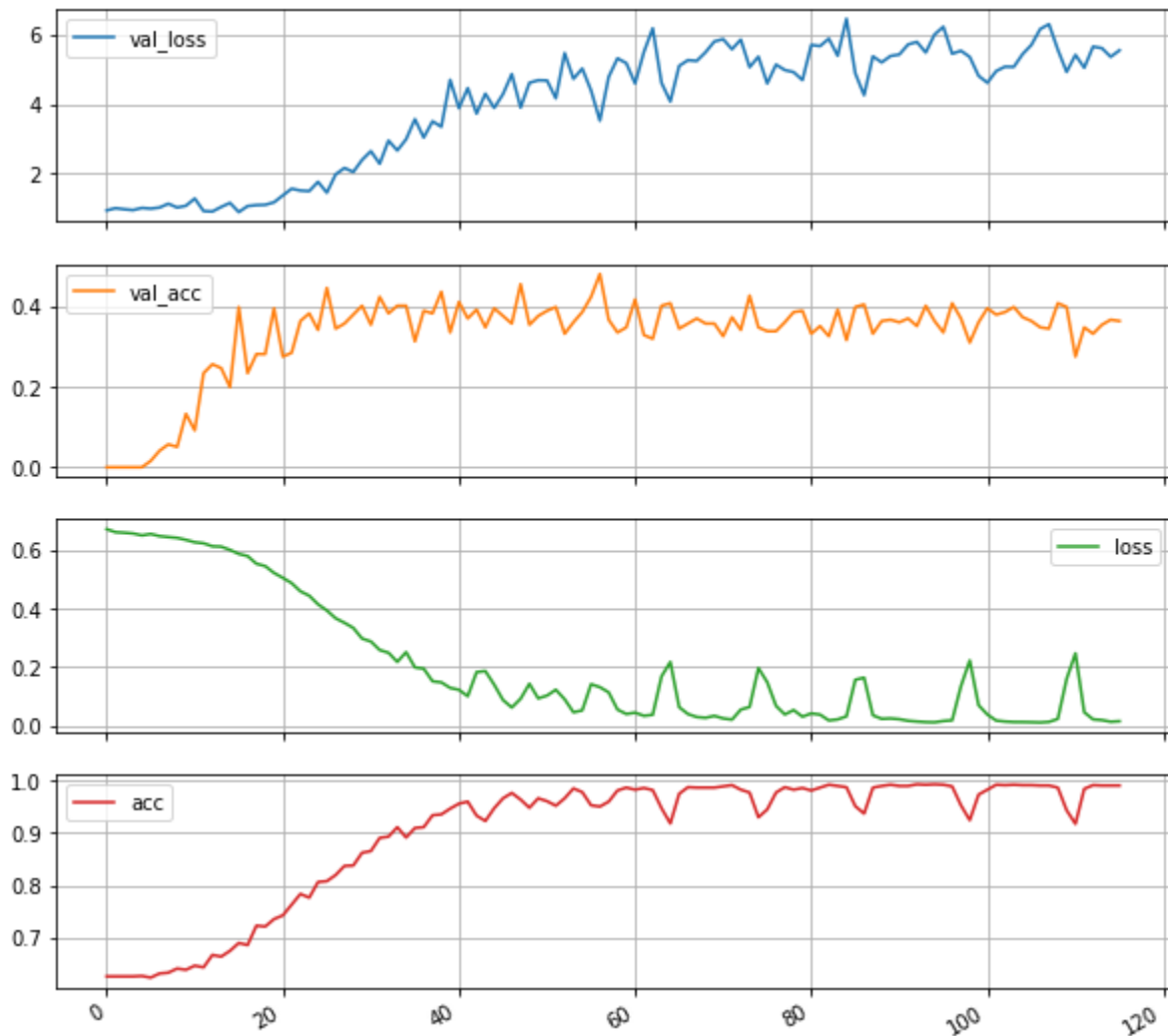
### 3.3.1 Human observed features (concatenation of features)



#### Observation

For the above networking setting for human observed data with concatenated features of same and different pair, we get a very low accuracy of only 30-40% the validation error is around 0.2 to 0.4 whereas the training accuracy comes high.

### 3.3.2 Human observed features (subtraction of features)



### Observation

For the above networking setting for human observed data with subtracted features, we get a very low accuracy of only 30-40% the validation error is around 0.2 to 0.4 whereas the training accuracy comes high.

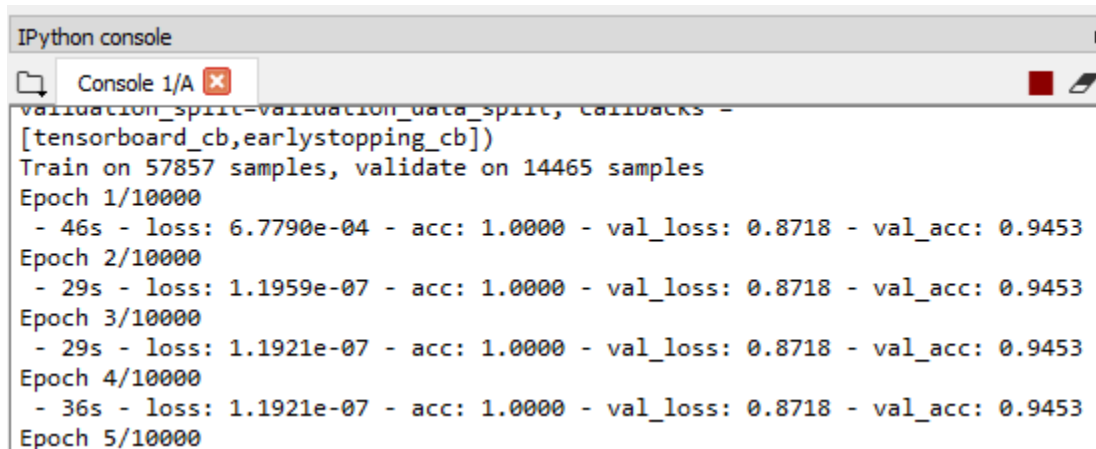
### 3.3.3 GSC dataset (concatenation of features)

```
IPython console
Console 1/A
validation_split=validation_data_split, callbacks =
[tensorboard_cb,earlystopping_cb])
Train on 57857 samples, validate on 14465 samples
Epoch 1/10000
- 46s - loss: 6.7790e-04 - acc: 1.0000 - val_loss: 0.8718 - val_acc: 0.9453
Epoch 2/10000
- 29s - loss: 1.1959e-07 - acc: 1.0000 - val_loss: 0.8718 - val_acc: 0.9453
Epoch 3/10000
```

### Observation

For the above networking setting for GSC dataset with concatenated features, we get a higher accuracy of 94% in less epochs.

### 3.3.4 GSC dataset (subtraction of features)



```
IPython console
Console 1/A
validation_split=validation_data_split, callbacks =
[tensorboard_cb,earlystopping_cb])
Train on 57857 samples, validate on 14465 samples
Epoch 1/10000
- 46s - loss: 6.7790e-04 - acc: 1.0000 - val_loss: 0.8718 - val_acc: 0.9453
Epoch 2/10000
- 29s - loss: 1.1959e-07 - acc: 1.0000 - val_loss: 0.8718 - val_acc: 0.9453
Epoch 3/10000
- 29s - loss: 1.1921e-07 - acc: 1.0000 - val_loss: 0.8718 - val_acc: 0.9453
Epoch 4/10000
- 36s - loss: 1.1921e-07 - acc: 1.0000 - val_loss: 0.8718 - val_acc: 0.9453
Epoch 5/10000
```

#### Observation

For the above networking setting for GSC dataset with subtracted features, we get a higher accuracy of 94%. Thus the model is comparatively good for GSC dataset.

## 4. Inference

1. The linear regression model is able to predict **comparatively better** in GSC model dataset because of the larger amount of data available
2. Dataset with concatenated features in both human observed and GSC dataset gives a **higher accuracy and less error rate** than the subtracted features in both GSC and human observed.
3. When the target values are extreme like 0 or 1 like in our dataset then it is **better to use logistics regression**. If the values are moderate then both linear and logistics works well but linear is preferred in that case.
4. The human observed dataset is **not giving high accuracy** when used neural network model. Whereas the GSC dataset gives **higher accuracy** than any other model in neural network

## 5. Conclusion

Thus by doing different model implementation for finding similarity between handwritten samples of known and questioned writer I found out that **neural network** gives a greater accuracy for the GSC dataset and **logistic regression** gives a greater accuracy for human observed dataset. **Concatenation** of features and training the model gives **greater** accuracy in all four dataset than subtracting the features. **GSC dataset** gives **greater** accuracy in all model because of more number of features and larger dataset.

Models	Human observed (concat)	Human Observed (subtract)	GSC dataset (concat)	GSC dataset (subtract)
Linear regression (validation error)	0.63	0.73	0.39	0.52
Logistic regression (validation error)	0.01	0.02	0.37	0.34
Neural network (Validation accuracy)	35%	32%	94%	94%

## References

1. <https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/>
2. <https://towardsdatascience.com/how-to-build-a-neural-network-with-keras-e8faa33d0ae4>
3. [https://www.google.com/search?safe=strict&rlz=1C1CHBF\\_enUS794US794&biw=1366&bih=577&tbm=isch&sa=1&ei=cMLbW6ftHsfN\\_Aa16K4AQ&q=logistic+regression+in+machine+learning&oq=logistic+reg&gs\\_l=img.1.0.35i39j0i0i67j0i0i67j0i3j0i67j0.2737.4467..5894...0.0..0.75.795.12.....1....1..gws-wiz-img.ZpBWBSBSbF8](https://www.google.com/search?safe=strict&rlz=1C1CHBF_enUS794US794&biw=1366&bih=577&tbm=isch&sa=1&ei=cMLbW6ftHsfN_Aa16K4AQ&q=logistic+regression+in+machine+learning&oq=logistic+reg&gs_l=img.1.0.35i39j0i0i67j0i0i67j0i3j0i67j0.2737.4467..5894...0.0..0.75.795.12.....1....1..gws-wiz-img.ZpBWBSBSbF8)
4. [https://www.google.com/search?safe=strict&rlz=1C1CHBF\\_enUS794US794&biw=1366&bih=577&tbm=isch&sa=1&ei=d8LbW9C\\_KKKb\\_Qbm64rIAg&q=neural+network+in+machine+learning&oq=neural&gs\\_l=img.1.0.35i39j0i67j0i2j0i67j0i5.25586.26264..27672...0.0..0.81.415.6.....1....1..gws-wiz-img.TJqxjurjJH8](https://www.google.com/search?safe=strict&rlz=1C1CHBF_enUS794US794&biw=1366&bih=577&tbm=isch&sa=1&ei=d8LbW9C_KKKb_Qbm64rIAg&q=neural+network+in+machine+learning&oq=neural&gs_l=img.1.0.35i39j0i67j0i2j0i67j0i5.25586.26264..27672...0.0..0.81.415.6.....1....1..gws-wiz-img.TJqxjurjJH8)