Project 2: Handwriting Comparison Task in Forensics

Sahil Suhas Pathak
Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY, 14214
sahilsuh@buffalo.edu
UBITName: sahilsuh, Person Number: 50289739

Abstract

Our task is to find the similarity between handwritten samples of the known and the questioned writer. We formulate this as a problem of linear regression and logistic regression, where we train a linear and logistic regression model on Human Observed features dataset, where the features are entered by human document examiners manually and on the GSC features dataset, where the features are extracted using Gradient Structural Concavity (GSC).

1 Introduction

For the implementation, we are using the CEDAR “AND” training dataset which consists of set of input features for each handwritten “AND” sample. Here we are working on two datasets, namely, Human Observed Dataset and GSC Features Dataset. Also, we are adapting two settings wherein we perform feature concatenation and feature subtraction between two respective “AND” samples.

Following are the steps illustrating how the implementation is carried out:

1. Processing the dataset i.e. Performing Feature Concatenation and Feature Subtraction

2. Importing dataset,

- i.e. for Human Observed Dataset, Feature Concatenation & Feature Subtraction
- & for GSC Features Dataset, Feature Concatenation & Feature Subtraction

3. Partition the dataset. We partition the dataset into 3 categories, Training, Validation and testing. The default case is when the Training data has 80% share and 10% share for both validation and testing data.

4. For a particular set of hyper parameters, we find the updated weights, where we train a regression model on the Training dataset using Stochastic Gradient Descent Solution.

5. Tune the hyper parameters to have a better performance on the Validation set.

6. Apply the model on the test set and evaluate the accuracy and the error.

2 Overview of the Datasets

As mentioned, we have two datasets, the Human Observed one and the one with GSC features. In Human Observed Dataset, we have 791 same writer pairs and 293,032 different writer pairs with 9 features for every “AND” sample. Whereas, for the GSC Features dataset, we have 71,531 same writer pairs and 762,557 different writer pairs with 512 features for every “AND” sample.

2.1 Processing Human Observed Features Dataset

The default (initial) setting is where we have three files,
HumanObserved-Features-Data.csv, which contains image id's and features. Every image is recognized by a specific name where the first four alphanumeric digits refer to the writer number, followed by a,b & c, which denotes the page number.

Steps for Processing the Human Observed Features Dataset:

1. Accept the HumanObserved-Features-Data.csv as a pandas dataframe.
2. Make a dictionary (HOD_mydict) where the 'key' would be the image id and 'value' would be the feature vector for that particular image id.
3. Accept the diffn_pairs.csv and same_pairs.csv files.
4. Since the number of different pairs (293,032) is way more than what we have in same pairs (791), we consider a sample of images where the number of images chosen are equal to the number of same pairs i.e. 791.
5. We then append the same pairs and the sampled different pairs together and store it in a new dataframe with their respective target values.
6. We then retrieve the particular feature vector from the dictionary ('value') by addressing it through the image id A ('key') and store it in a variable field called 'f1'. As the dictionary stores a (key, value) pair, we are able to extract those feature vectors for particular image id's. Similarly, we do the same for image id B and store it in a variable field called 'f2'.
7. Now for Concatenation, we perform f3 = f1 + f2, since both are lists, they get concatenated and we get 18 features.
8. We then store these 18 features for respective pair of image id A and image id B, we repeat the above process till all pairs have concatenated feature vectors.
9. After doing the above process, we randomly shuffle the pairs.
10. For Subtraction, we perform f3 = f1 – f2, but we do take absolute values into consideration.
11. Even for Subtraction, we randomly shuffle the pairs.
12. Thus, for Concatenation setting, we generate two files,
    i. HOD_concat_dataset_f.csv: Stores all the concatenated features of the image pairs.
    ii. HOD_concat_dataset_t.csv: Stores all the target values of the image pairs.

Similarly, for Subtraction, we have two files,
   i. HOD_sub_dataset_f.csv: Stores all the subtracted features of the image pairs.
   ii. HOD_sub_dataset_t.csv: Stores all the target values of the image pairs.

2.2 Processing GSC Features Dataset

The default (initial) setting is where we have three files,
GSC-Features.csv, which contains image id's and features. Every image is recognized by a specific name where the first four alphanumeric digits refer to the writer number, followed by a,b & c, which denotes the page number.

Steps for Processing the GSC Features Dataset:

1. Accept the GSC-Features.csv as a pandas dataframe.
2. Accept the diffn_pairs.csv and same_pairs.csv files.
3. We consider a sample of images from diffn_pairs.csv, where the number of images chosen are equal to the number of same pairs i.e. 71,531.
4. We then append the same pairs and the sampled different pairs together and store it in a new dataframe with their respective target values.
5. Now for Concatenation, we perform pd.merge operation twice, where for first instance we bring in all those features corresponding to image id A and for the second instance we bring in all those features corresponding to image id B.

6. This is how we store these 1024 (512 + 512) features for respective pair of image id A and image id B, we repeat the above process till all pairs have concatenated feature vectors.
7. For Subtraction, we consider working on a duplicate copy of Concatenated Dataset. Here we store our first feature of image id A i.e. f1_x and the first feature of image id B f1_y and subtract both the feature columns and append those columns to a new dataframe called "final_GSC_sub_dataset" and we do take absolute values into consideration.
8. We then randomly shuffle the pairs in Concatenated Dataset.
9. Even for Subtraction, we randomly shuffle the pairs.
10. Thus, for Concatenation setting, we generate two files,

i. GSC_concat_dataset_f.csv: Stores all the concatenated features of the image pairs.
ii. GSC_concat_dataset_t.csv: Stores all the target values of the image pairs.

Similarly, for Subtraction, we have two files,

i. GSC_sub_dataset_f.csv: Stores all the subtracted features of the image pairs.
ii. GSC_sub_dataset_t.csv: Stores all the target values of the image pairs.

3 Partitioning the Dataset

Here, we are performing Shuffled Writer Partitioning. It denotes that the writer can be in training set or can be in testing set but not necessarily that it has to be in either of the sets. The default configuration is where the training set accounts for 80% of data samples and validation and testing set both account for 10% share respectively.

4 Linear Regression on Human Observed Dataset

To perform Linear Regression, we are using Stochastic Gradient Descent, the algorithm, first takes a random initial weight. Then it updates the weight value as:

$$w^{\tau+1} = w^{\tau} + \Delta w^{\tau}$$

Where, $\Delta w^{\tau} = -\eta^{\tau}\nabla E$, Δw^{τ} is the weight update parameter. The minus sign indicates that the computations go along the opposite direction of the gradient of the error.

η^{τ} is the learning rate and $\nabla E = \nabla E_D + \lambda \nabla E_w$

Also, $\nabla E_D = -(t_n - w^T \emptyset(x_n))\emptyset(x_n)$, is nothing but the derivative of the

$E_D = \frac{1}{2}\sum_1^n (t_n - w^T \emptyset(x_n))^2$ and $\nabla E_w = w$

4.1 Steps for Calculation

1. We compute the Design matrix $\emptyset$.
2. We calculate the following terms in the order, ∇E_D, $\lambda \nabla E_w$, ∇E, Δw^{τ}, $w^{\tau+1}$.
4. Using these values, we train our model with initial set of hyper-parameters.
5. The ERMS i.e. the Root mean squared error value for the training, validation and test is determined.
6. We tune the hyper-parameters to obtain a better performance and try to reduce ERMS value.
7. We do the above procedures for both Concatenated Features as well as for Subtracted Features setting.

4.2 Observations for Human Observed Data, Concatenated Features

1. Learning Rate = 0.001, lambda = 0.05, M = 5
E_rms Training = 0.49963
E_rms Validation = 0.49876
E_rms Testing = 0.49877

2. Learning Rate = 0.001 lambda = 0.5, M = 5
E_rms Training = 0.50015
E_rms Validation = 0.49876
E_rms Testing = 0.49973

3. Learning Rate = 0.001 lambda = 1, M = 5
E_rms Training = 0.50339
E_rms Validation = 0.49997
E_rms Testing = 0.50361

4. Learning Rate = 0.001 lambda = 10, M = 5
E_rms Training = 0.58839
E_rms Validation = 0.57402
E_rms Testing = 0.59222

Inference: From the above observations, we can say that for increasing values of lambda for a certain learning rate and number of basis function, we see an increasing trend in the ERMS value for all the three i.e. training, validation and testing data samples.

5. Learning Rate = 0.1, lambda = 2, M = 5
E_rms Training = 0.49963
E_rms Validation = 0.49876
E_rms Testing = 0.49877

6. Learning Rate = 0.1, lambda = 2, M = 5
E_rms Training = 0.49962
E_rms Validation = 0.49874
E_rms Testing = 0.49881

7. Learning Rate = 0.001, lambda = 0.5, M = 15
E_rms Training = 0.49962
E_rms Validation = 0.49874
E_rms Testing = 0.49881

8. Learning Rate = 0.001, M=15, lambda = 0.5, Training on only 500 data points
E_rms Training = 0.49962
E_rms Validation = 0.49875
E_rms Testing = 0.4988

9. Learning Rate = 0.0001 lambda = 10, M = 5
E_rms Training = 0.62162
E_rms Validation = 0.60544
E_rms Testing = 0.62608

10. Learning Rate = 0.01, lambda = 0.2, M = 9
E_rms Training = 0.49963
E_rms Validation = 0.49876
E_rms Testing = 0.49849

11. Learning Rate = 0.001, lambda = 0.8, M = 10
E_rms Training = 0.49921
E_rms Validation = 0.49748
E_rms Testing = 0.49724

Confusion Matrix for Obs. 11.
Training Data - [[631, 0],
[635, 0]]

Validation Data - [[73, 0],
[85, 0]]

Testing Data - [[86, 0],
[71, 0]]

Plots for Obs. No. 11. ERMS values for number of iterations

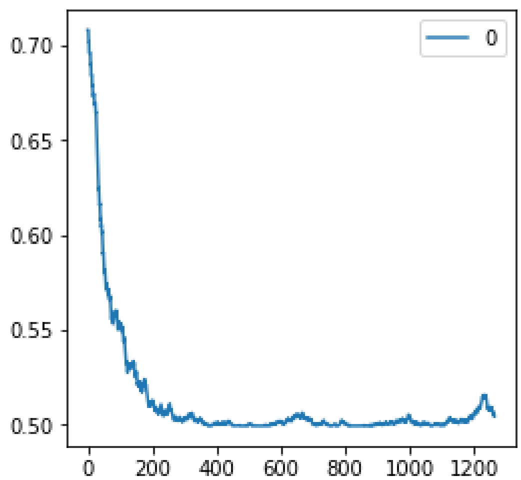

Figure 1. Training EMRS

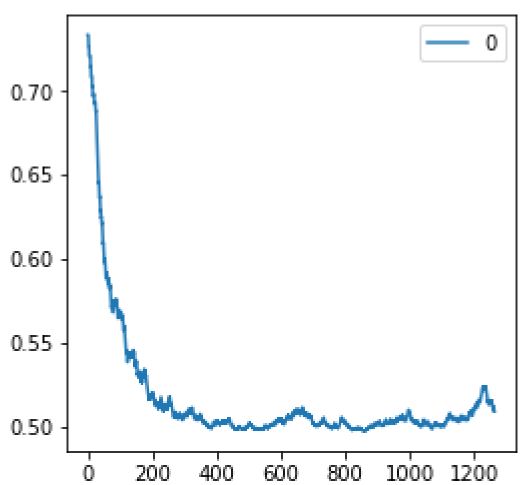

Figure 2. Validation ERMS

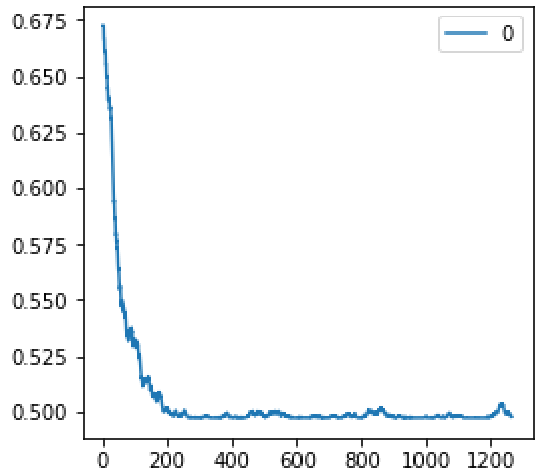

Figure 3. Testing ERMS

Note: The above observation has a 0 score for precision as well as for recall. This is not what is expected. Having zero for both precision and recall is the worst case scenario. This problem is duly solved up to a certain extent by Logistic Regression.

Inference: From the above observations, even when we tweak the hyper-parameters, the ERMS value remains fairly constant in the range of 0.45 to 0.55. Only if we substantially increase lambda values, we see an upward trend in the ERMS value. This could be because, Human Observed Data has very less number of features, and also the training set has considerably very less number of data points to work with. Also, it might be a case where the dataset itself is inconsistent and thus we are unable to achieve the accuracy when we train on this dataset. Also, from reading number 8, we trained on even less number of data points i.e. 500, there we got an ERMS value of 0.49 approximately. Moreover, if we consider Observation 11, the confusion matrix clearly indicates that our model is poorly performing.

4.3 Observations for Human Observed Data, Subtracted Features

1. Learning Rate = 0.001, m = 15, lambda = 0.5
E_rms Training = 0.49984
E_rms Validation = 0.49483
E_rms Testing = 0.49761

2. Learning Rate = 0.001, M = 5, lambda = 0.05
E_rms Training = 0.49979
E_rms Validation = 0.50018
E_rms Testing = 0.50045

3. Learning Rate = 0.001, M = 10, lambda = 0.05
E_rms Training = 0.49986
E_rms Validation = 0.49517
E_rms Testing = 0.4976

4. Learning Rate = 0.001, M =15, lambda = 0.05
E_rms Training = 0.50618
E_rms Validation = 0.52048
E_rms Testing = 0.49897

5. Learning Rate = 0.0001, M = 15, lambda = 1
E_rms Training = 0.50913
E_rms Validation = 0.52616
E_rms Testing = 0.5005

6. Learning Rate = 0.001, M = 10, lambda = 0.5
E_rms Training = 0.49986
E_rms Validation = 0.49775
E_rms Testing = 0.4976

Confusion Matrix for Obs. No. 6.
Training Data - [[197, 442],
[181, 446]]

Validation Data - [[26, 40],
[30, 62]]

Testing Data - [[24, 62],
[20, 51]]

Inference: From the above observations, we can see that the ERMS values are fairly constant even if we have changed certain parameters. Again, as mentioned for the Concatenated Setting, the dataset might be inconsistent or there aren't enough features for the model to work with. By looking at the confusion matrix for Observation No. 6. We can see that the model is predicting some False Positives and False Negatives apart from the True Positives and True Negatives.

5 Linear Regression on GSC Features Dataset

Since the dataset is large, we initially train on 1000 data samples and later on evaluate the need so as to estimate how many more data samples are required. Here, the ERMS value is taken into consideration when estimating the number of additional data samples to be included.

5.1 Observations GSC Features Dataset, Concatenated Features

1. Learning Rate = 0.1, lambda = 0.5, Training on 1000 data points
E_rms Training = 0.6697
E_rms Validation = 0.66744
E_rms Testing = 0.67099

2. Learning Rate = 0.01, lambda = 0.1, Training on 2000 data points
E_rms Training = 0.58773
E_rms Validation = 0.58754
E_rms Testing = 0.58539

3. Learning Rate - 0.001, M = 10, lambda = 0.1, Training on 2000 data points
E_rms Training = 0.52437
E_rms Validation = 0.52332
E_rms Testing = 0.52567

4. Learning Rate = 0.001, lambda = 0.1, M = 10, Training on 3000 data points
E_rms Training = 0.52063
E_rms Validation = 0.51962
E_rms Testing = 0.52204

Confusion Matrix for Obs. No. 4.
Training set - [[39210 18002]
[37516 19722]]

Validation set - [[4947 2223]
[4651 2485]]

Training set - [[4843 2306]
[4765 2391]]

Inference: From the above observations, we started with 1000 data points, then we considered 2000 data points, as we considered more and more points, the ERMS value decreased and reached a value of ~0.52. I believe that if we consider more data points the accuracy will increase up to a certain value but the computation will indeed take a significant amount of time.

5.2 Observations GSC Features Dataset, Subtracted Features

1. Learning Rate = 0.001, lambda = 0.1, M = 9
E_rms Training = 0.52151
E_rms Validation = 0.52049
E_rms Testing = 0.52288

Confusion Matrix for Obs. No 1
Training set - [[41829 15383]
[40376 16862]]

Validation set - [[5245 1925]
[5016 2120]]

Testing set - [[5174 1975]
[5076 2080]]

Plots for Obs. No. 1. ERMS values for number for iterations.

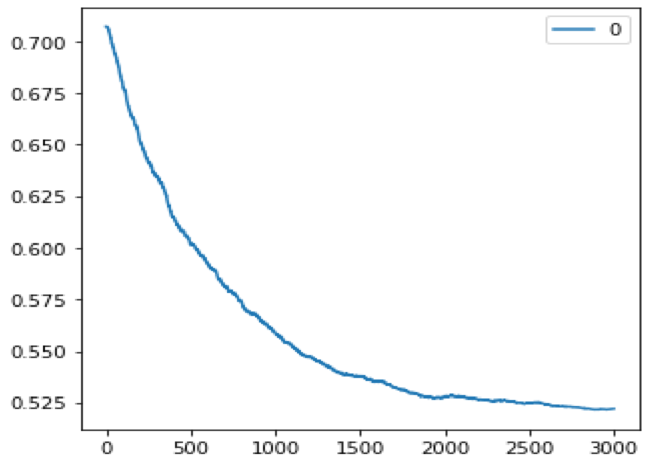

Figure 4. Training ERMS

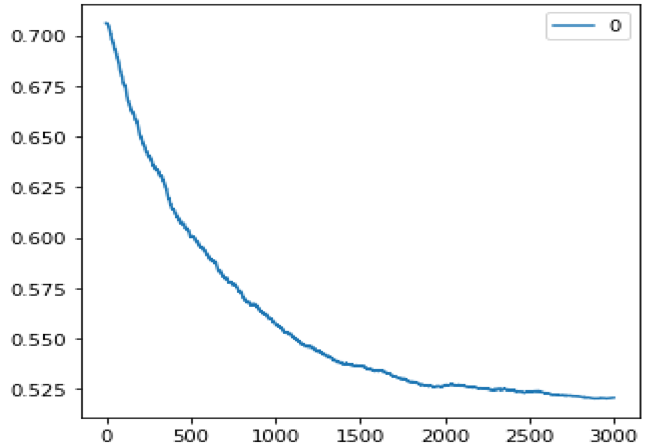

Figure 5. Validation ERMS

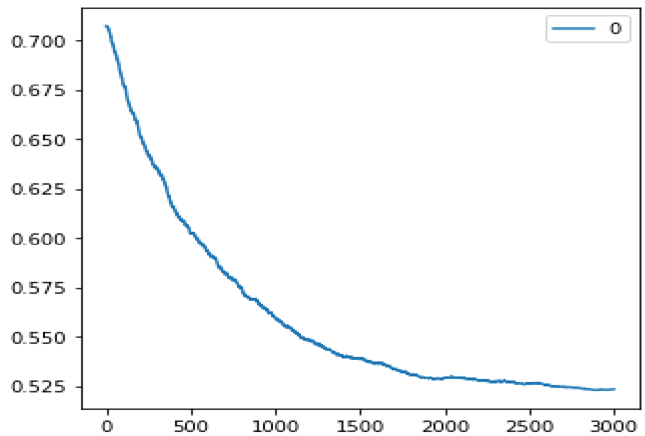

Figure 6. Testing ERMS

6 Logistic Regression on Human Observed Dataset

Here, we consider the dataset with features and target. The dataset, as mentioned, is generated during the data processing stage in Linear Regression. For Concatenated setting, there are in total 19 columns where 18 are the feature columns and the last column is for the target. Similarly, for Subtracted Setting there are in total 10 columns where 9 columns are the feature columns and last column is for the target.

6.1 Observations for Human Observed Data, Concatenated Features

1. Learning Rate = 0.001, Epochs = 10000

Final Training Cost: 0.67849096634423848
Training Accuracy: 57.74092
Validation Cost: 0.6819576974048419
Validation Accuracy: 56.32911392405063
Testing Cost: 0.7041326505277529
Testing Accuracy: 50.955414012738856

Confusion Matrix for Obs. No. 1:
Training set - [[348, 283],
[258, 377]]

Validation set - [[39, 34],
[35, 50]]

Test set - [[44, 42],
[35, 36]]

2. Learning Rate = 0.0001, Epochs = 10000

Final Training Cost: 0.686908281589743
Training Accuracy: 57.50395
Validation Cost: 0.687674292940304
Validation Accuracy: 56.962025316455694
Testing Cost: 0.6943079681389915
Testing Accuracy: 50.318471337579616

3. Learning Rate = 0.1, Epochs = 10000

Final Training Cost: 0.6770307694588661
Training Accuracy: 57.66193
Validation Cost: 0.6863988669400515
Validation Accuracy: 57.59493670886076
Testing Cost: 0.7121401724502271
Testing Accuracy: 49.044585987261144

Inference: The observations are very similar when compared to what we had in Linear Regression for the same setting, but in fact these are much better than what we observed in Linear Regression in terms of predicting the correct label for a particular sample. If we recall Observation No. 11 from Human Observed Data, Concatenated Features, the confusion matrix is biased to just one of the labels and predicting the same label for a different one. But, in Logistic Regression, our model is performing in a fairly satisfactory way. By referring to the Observation No. 1. The model is trying to predict True Positives and True Negatives, though it is not efficient enough to correctly predict each and every sample.

6.2 Observations for Human Observed Data, Subtracted Features

1. Learning Rate = 0.001, epochs = 5000

Final Training Cost: 0.6870462855941741
Training Accuracy: 55.60821
Validation Cost: 0.6979625327016026
Validation Accuracy: 50.63291139240506
Testing Cost: 0.7016182988652503
Testing Accuracy: 45.22292993630573

2. Learning Rate = 0.001, epochs = 10000

Final Training Cost: 0.6862674878995099
Training Accuracy: 55.60821
Validation Cost: 0.7008188352429199
Validation Accuracy: 49.36708860759494
Testing Cost: 0.70658689018561
Testing Accuracy: 45.22292993630573

Confusion Matrix for Obs. No. 2:

Training set: [[368, 270],
[299, 329]]

Validation set: [[31, 40],
[40, 47]]
Testing set: [[39, 42],
[44, 32]]

3. Learning Rate = 0.00001, Epochs = 10000

Final Training Cost: 0.6928406872056896
Training Accuracy: 55.60821
Validation Cost: 0.6932907024340789
Validation Accuracy: 51.265822784810126
Testing Cost: 0.693271847295292
Testing Accuracy: 47.13375796178344

Confusion Matrix for Obs. No. 3:
Training set - [[439, 199],
[367, 261]]

Validation set - [[44, 27],
[50, 37]]

Testing set - [[49, 32],
[51, 25]]

Inference: Logistic Regression on Subtracted Features dataset is giving an accuracy of around ~55% on the training set and around ~47% on the testing set. But, by looking at the confusion matrix, the model is trying to predict the True positives and True Negatives up to a certain extent, still the accuracy is relatively low. This indeed conveys that accuracy is not the only parameter to judge but we can also seek for precision score and recall score to get an overall estimate of our classifier performance.

7 Logistic Regression on GSC Features Dataset

Here, we consider the dataset with features and target. The dataset, as mentioned, is generated during the data processing stage in Linear Regression. For Concatenated setting, there are in total 1025 columns where 1024 are the feature columns and the last column is for the target. Similarly, for Subtracted Setting there are in total 513 columns where 512 columns are the feature columns and last column is for the target.

7.1 Observations GSC Features Dataset, Concatenated Features

1. Learning Rate = 0.001, **Epochs 100**

Final Training Cost: 0.6916859684926552
Training Accuracy: 51.48362
Validation Cost: 0.6919364462523283
Validation Accuracy: 50.98560044736474
Testing Cost: 0.6915924466900626
Testing Accuracy: 51.869975533030406

Confusion Matrix for Obs. No. 1.
Training set - [[8657, 48546],
[6981, 50266]]

Validation set - [[1045, 6156],
[856, 6249]]

Testing set - [[1049, 6077],
[808, 6371]]

2. Learning Rate = 0.01, **Epochs 100**

Final Training Cost: 0.684569432562124
Training Accuracy: 56.24552
Validation Cost: 0.6858390404435903
Validation Accuracy: 55.40332727526912
Testing Cost: 0.6842839350173158
Testing Accuracy: 56.37189793778399

3. Learning Rate = 0.01, **Epochs = 200**

Final Training Cost: 0.6808206387031006
Training Accuracy: 57.02578
Validation Cost: 0.682775489471735
Validation Accuracy: 56.088354536558086
Testing Cost: 0.6805153088966246
Testing Accuracy: 57.189793778399164

Training Precision Score - 0.5712392429891678
Training Recall Score - 0.5646933463762294

Validation Precision Score - 0.5589795040848502
Validation Recall Score - 0.5489092188599578

Testing Precision Score - 0.5739106066974919
Testing Recall Score - 0.5705530018108371

4. Learning Rate = 0.01, **Epochs = 400**

Final Training Cost: 0.6763075558230568
Training Accuracy: 58.09174
Validation Cost: 0.6790491210045215
Validation Accuracy: 56.99706416888019
Testing Cost: 0.6761056735501167
Testing Accuracy: 58.03565186997553

Confusion Matrix for Obs. No. 4
Training set - [[33675, 23528],
[24436, 32811]]
Validation set - [[4216, 2985],
[3167, 3938]]
Testing set - [[4161, 2965],
[3038, 4141]]

Training Precision Score - 0.5823852038552335
Training Recall Score - 0.5731479378832078

Validation Precision Score - 0.5688285425393615
Validation Recall Score - 0.5542575650950036

Testing Precision Score - 0.5827469743878413
Testing Recall Score 0.5768212843014348

Note: For the same learning rate of 0.01, the Accuracy on all the three data partitions increased as and when we moved from Epochs 100, 200 to 400. I believe that the accuracy can further be increased if we consider more epochs. The environment required for such computations is restricted by the usage of personal device.

Inference: For Obs. No. 3 & 4, we can see the precision and recall values for all the data partitions. Precision gives us; Out of all the examples the classifier labelled as positive, what fraction were correct? On the other hand, recall answers; Out of all the positive examples there were, what fraction did the classifier pick up? Having a score of 1 in precision and having a score of recall as 1 is the best case scenario. But, if we try to achieve a perfect recall (simply making the classifier label all the examples as positive), this will in turn make the classifier suffer from horrible precision. Thus we need to balance out both the scores.

7.2 Observations GSC Features Dataset, Subtracted Features

1. Learning Rate = 0.01, **Epochs 300**

Final Training Cost: 0.6854800577304223
Training Accuracy: 57.83748
Validation Cost: 0.6855910243163944
Validation Accuracy: 57.77995246749616
Testing Cost: 0.686214396019872
Testing Accuracy: 57.01502970989165

Confusion Matrix for Obs. No. 1
Training set - [[33503, 23709],
[24546, 32692]]

Validation set - [[4154, 3016],
[3024, 4112]]

Testing set - [[4094, 3055],
[3094, 4062]]

Training Precision Score - 0.579635112852609
Training Recall Score - 0.5711590202313148

Validation Precision Score - 0.5768799102132436
Validation Recall Score - 0.5762331838565022

Testing Precision Score - 0.5707461008852044
Testing Recall Score - 0.5676355505869201

2. Learning Rate = 0.01, **Epochs 300**

Training Accuracy: 58.23853
Validation Cost: 0.6822456193818929
Validation Accuracy: 58.12945617223543
Testing Cost: 0.683083936460204
Testing Accuracy: 57.154840964697655

Confusion Matrix for Obs. No. 2
Training set - [[33503, 23709],
[24546, 32692]]

Validation set - [[4154, 3016],
[3024, 4112]]

Testing set - [[4094, 3055],
[3094, 4062]]

8 Neural Network Implementation

Here, we first accept the dataset. Then we partition it according to the features and the target.

8.1 Observations Human Observed Dataset, Concatenated Features

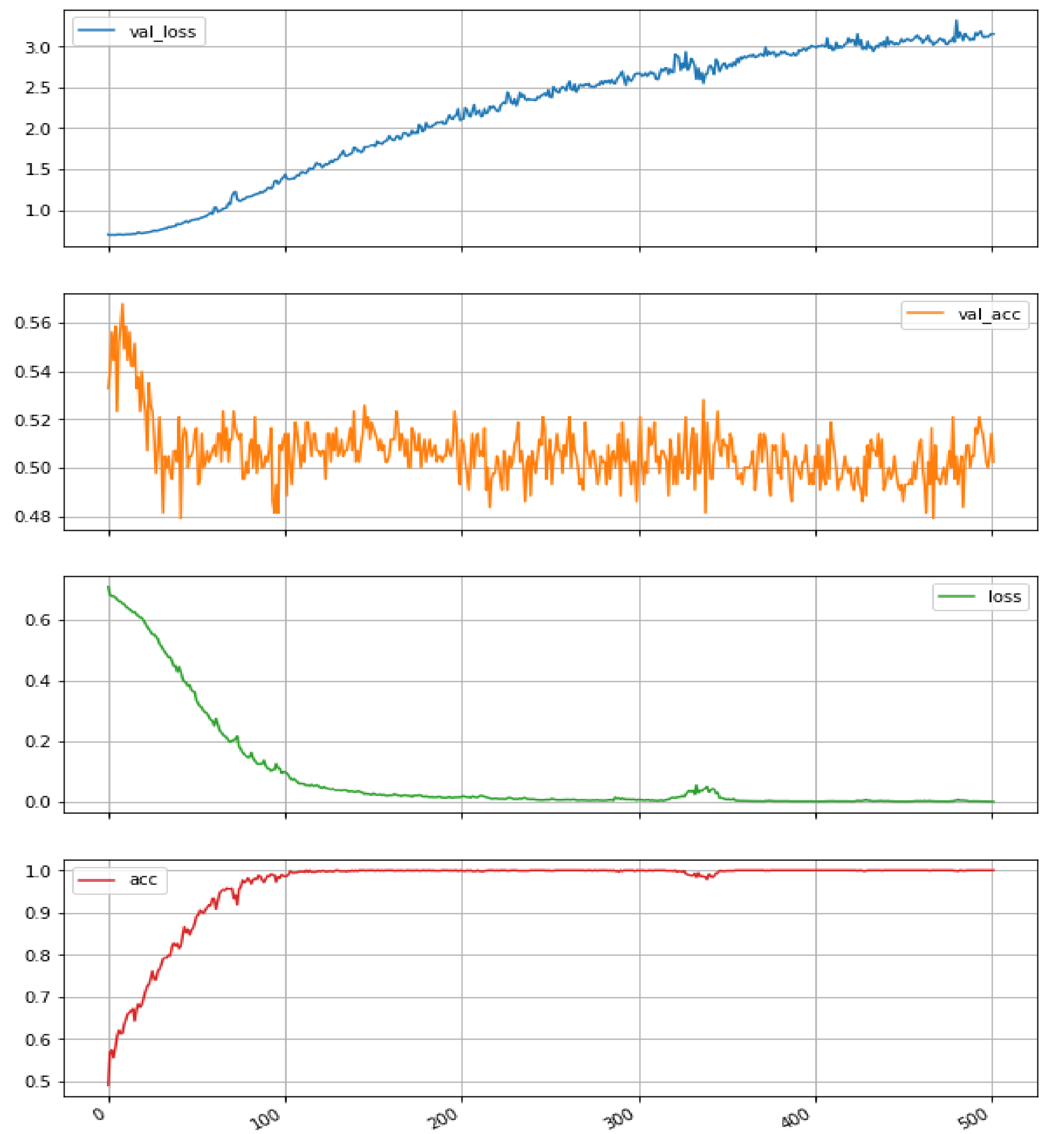

Figure 7. A. Validation Loss, B. Validation Accuracy, C. Training Loss, D. Training Accuracy

Here, we have used SGD with Learning Rate as 0.1, loss is ‘binary_crossentropy', optimizer is ‘adam’ and metrics is 'accuracy'.

Optimum value – Training Loss - 0.0046, Training Accuracy: 0.9990, Validation_loss: 3.0459 - Validation_Accuracy: 0.4836
Epoch 00501: early stopping

Inference: As mentioned before, the Human Observed Dataset has very less number of features to work with. Thus the output of the Validation i.e. the Accuracy parameter is fairly distorted.

8.2 Observations Human Observed Dataset, Subtracted Features

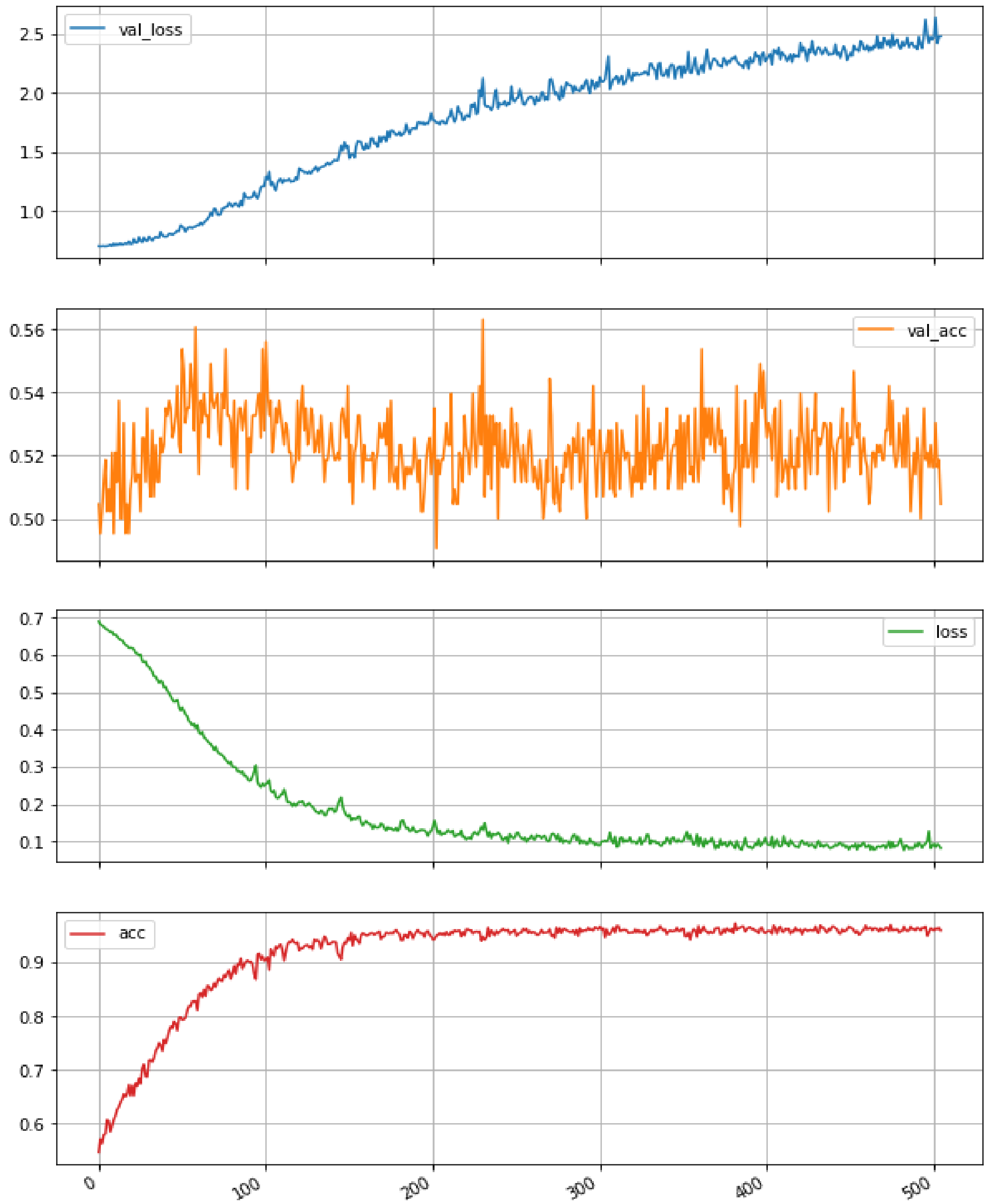

Figure 8. A. Validation Loss, B. Validation Accuracy, C. Training Loss, D. Training Accuracy

Note: Here we have used SGD with learning rate as 0.01. loss is 'binary_crossentropy', optimizer is 'adam' and metrics is 'accuracy'.
Also, validation_data_split = 0.3, num_epochs = 5000, model_batch_size = 150, early_patience = 500

Optimum Value: Training loss: 0.0819, Training Accuracy: 0.9598 - Validation_loss: 2.4775, Validation_Accuracy: 0.5047
Epoch 00505: early stopping

8.3 Observations of GSC Features Dataset, Concatenated Features

For the above mentioned configuration except for learning rate = 0.1, here is the loss and the accuracy per epoch,

Train on 90129 samples, validate on 38627 samples
Epoch 1/10
90129/90129 - 43s 477us/step - loss: 0.5018 - acc: 0.7394 - val_loss: 0.3959 - val_acc: 0.8177
Epoch 2/10
90129/90129 - 43s 478us/step - loss: 0.3319 - acc: 0.8542 - val_loss: 0.2973 - val_acc: 0.8692
Epoch 3/10
90129/90129 - 44s 484us/step - loss: 0.2342 - acc: 0.9035 - val_loss: 0.2779 - val_acc: 0.8832
Epoch 4/10
90129/90129 - 45s 496us/step - loss: 0.1657 - acc: 0.9343 - val_loss: 0.2581 - val_acc: 0.8967
Epoch 5/10
90129/90129 - 42s 463us/step - loss: 0.1215 - acc: 0.9529 - val_loss: 0.2182 - val_acc: 0.9138
Epoch 6/10
90129/90129 - 36s 404us/step - loss: 0.0895 - acc: 0.9664 - val_loss: 0.2229 - val_acc: 0.9186
Epoch 7/10
90129/90129 - 40s 442us/step - loss: 0.0673 - acc: 0.9749 - val_loss: 0.2396 - val_acc: 0.9183
Epoch 8/10
90129/90129 - 42s 466us/step - loss: 0.0567 - acc: 0.9791 - val_loss: 0.2449 - val_acc: 0.9180
Epoch 9/10
90129/90129 - 39s 435us/step - loss: 0.0446 - acc: 0.9838 - val_loss: 0.2459 - val_acc: 0.9229
Epoch 10/10
90129/90129 - 41s 457us/step - loss: 0.0387 - acc: 0.9857 - val_loss: 0.2661 - val_acc: 0.9210

Inference: In just 10 epochs, we got a Validation Accuracy of 92%. This accuracy will further increase if we increase the number of epochs and can reach up to 100%.

8.4 Observations of GSC Features Dataset, Subtracted Features

Train on 90129 samples, validate on 38627 samples
Epoch 1/10
90129/90129 - 20s 217us/step - loss: 0.0215 - acc: 0.9928 - val_loss: 0.7589 - val_acc: 0.8230
Epoch 2/10
90129/90129 - 20s 227us/step - loss: 0.0151 - acc: 0.9950 - val_loss: 0.8192 - val_acc: 0.8197
Epoch 3/10
90129/90129 - 22s 240us/step - loss: 0.0268 - acc: 0.9902 - val_loss: 0.8094 - val_acc: 0.8108
Epoch 4/10
90129/90129 - 20s 216us/step - loss: 0.0319 - acc: 0.9883 - val_loss: 0.8677 - val_acc: 0.8146
Epoch 5/10
90129/90129 - 20s 222us/step - loss: 0.0261 - acc: 0.9905 - val_loss: 0.8306 - val_acc: 0.8202
Epoch 6/10
90129/90129 - 22s 247us/step - loss: 0.0270 - acc: 0.9903 - val_loss: 0.8274 - val_acc: 0.8124
Epoch 7/10
90129/90129 - 22s 246us/step - loss: 0.0249 - acc: 0.9911 - val_loss: 0.8752 - val_acc: 0.8140

Epoch 8/10
90129/90129 - 20s 224us/step - loss: 0.0231 - acc: 0.9920 - val_loss: 0.8768 - val_acc: 0.8197
Epoch 9/10
90129/90129 - 23s 250us/step - loss: 0.0220 - acc: 0.9920 - val_loss: 0.9007 - val_acc: 0.8169
Epoch 10/10
90129/90129 - 26s 293us/step - loss: 0.0243 - acc: 0.9912 - val_loss: 0.8320 - val_acc: 0.8185

Inference: In just 10 epochs, we got a Validation Accuracy of 82%. This accuracy will further increase if we increase the number of epochs and can reach up to 95 - 100%.

9 Execution

Run Linear Regression Notebook for Human Observed Data and for GSC Features Data first. It will then generate the necessary datasets which would be used for Logistic Regression as well as for Neural Network Implementation.

10 References

Teaching Assistants – CSE 574 – Introduction to Machine Learning

https://tryolabs.com/blog/2013/03/25/why-accuracy-alone-bad-measure-classification-tasks-and-what-we-can-do-about-it/

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html