

A Report on Hand Writing Comparison using Linear and Logistic Regression

Ekta Katiyar

Person Number: 

November 1, 2018

Abstract

In this experiment, we have studied how to compare to apply machine learning to solve the handwriting comparison task in forensics by minimizing the root mean square error and cost function for the datasets on which we are training our model. This experiment also helped in understanding various aspects of hyper parameters and model training methodologies in machine learning. These are the parameters whose value is set before the learning process begins and tuning hyperparameters impacts the Learning process of the model. If hyperparameters are poorly chosen, then the network will learn slowly or might not learn at all.

Introduction

Our problem statement was to solve the handwriting comparison task in forensics using Linear Regression and Logistic Regression model of machine learning.

Problem Definitions:

- First, we have to train our linear regression model with Human Observed Dataset of hand writing features using Stochastic Gradient descent in such a way that we obtain minimum root mean square error for Concatenated Features and Subtracted Features.
- Second, we have to train our linear regression model with GSC feature Dataset using Stochastic Gradient descent in such a way that we obtain minimum root mean square error for Concatenated Features and Subtracted Features.
- Third, we have to train our logistic regression model with Human Observed Dataset of hand writing features in order to obtain minimum cost function and maximum accuracy for Concatenated Features and Subtracted Features.
- Fourth, we have to train our logistic regression model with GSC feature Dataset in order to obtain minimum cost function and maximum accuracy for Concatenated Features and

Subtracted Features.

Experiments

Below is the list of network tuning done in the hyperparameters while training our model using Stochastic Gradient Descent(SGD):

1. Change in the regularizer.
2. Change in Learning rate
3. Change in weight during initialization
4. Change in the number of epochs

Below is the list of the network tuning done in the hyperparameters while training our model using Logistic Regression

1. Change in weight during initialization.
2. Change in the learning rate.
3. Change in the number of epochs.

Explanations

Human Observed Dataset: In order to implement Linear and Logistic regression model we will be using “AND” images samples extracted from CEDAR Letter dataset. This dataset contains the csv files that mentions the 18 features for each pair of handwriting sample along with the target values for different pairs and same pairs of handwriting in “diffn_pairs.csv” and “same_pairs.csv” respectively.

GSC Dataset: In order to implement Linear and Logistic regression model we will be Gradient Structural Concavity algorithm that generates 512 sized feature vectors for an input handwritten “AND” image. The entire dataset consists of 71,531 same writer pairs and 762,557 different rows along with the actual values for different pairs and same pairs of handwriting in “diffn_pairs.csv” and “same_pairs.csv” respectively.

After obtaining the dataset we will convert the target dataset into vector and will generate the data matrix for data in both Human Observed Dataset and GSC Dataset. Inside dataMatrix[] all the rows represent the number of queries and the columns represent the number of features.

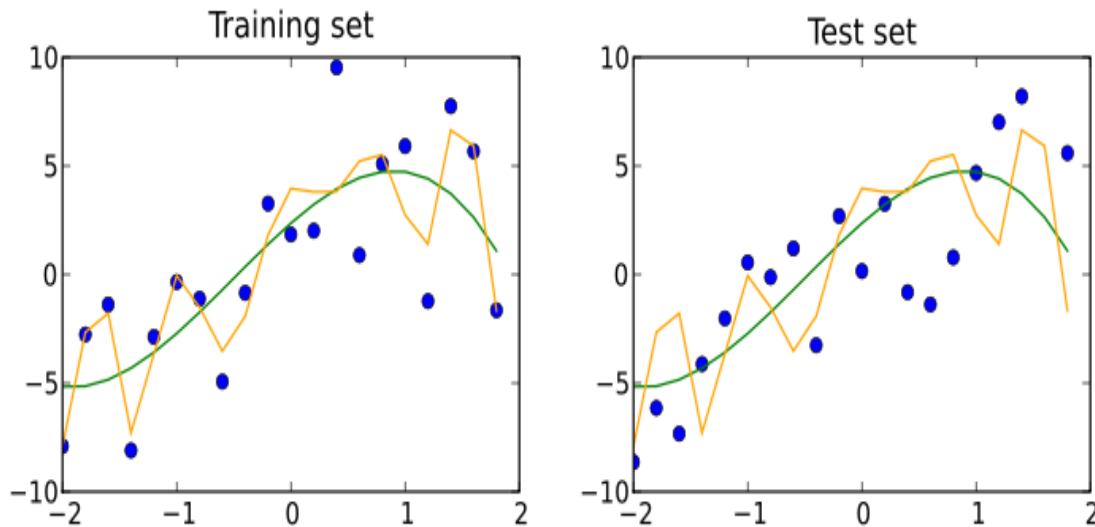
LINEAR REGRESSION:

- **Training data:** The data we used to feed to our model for it to learn the flow of information is known as Training Data. 80% of the total data will be taken as Training data.

- **Validation data:** A validation dataset is a sample of data held back from training your model that is used to give an estimate of how skilled our model is after tuning the model hyperparameters.

We will be considering the 10% of our raw dataset as Validation Data for our model.

- **Test Data:** It gives an unbiased estimate of how well is our model is tuned when comparing or selecting between the final models. It follows the same probability distribution as the training dataset.



We will be considering the 10% of the whole dataset as the testing data.

- **Radial Basis Function:** It is a real valued function ϕ whose value depends upon the distance from origin in such a way that

$$\phi(x) = \phi(\|x\|)$$

or alternatively on the distance from a point p , such that

$$\phi(x, p) = \phi(\|x - p\|)$$

Radial Basis function can be represented in the below form

$$y(x) = \sum_{i=1}^N w_i \phi(\|x - x_i\|)$$

where the approximation function $y(x)$ is represented as the sum of N radial basis functions

x_i is the center for each radial basis function.

w_i is the coefficient of weights associated.

- **ERMS:** It refers to the root mean square error which is a measure to represent the difference between the predicted value from the model and the value estimated by the model.

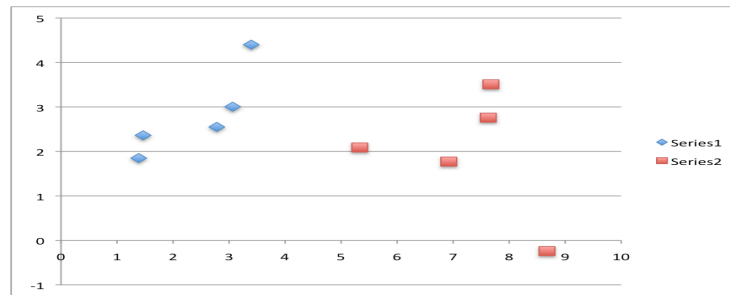
$$ERMS = \frac{1}{2}(t_n - \hat{t}_n)^2$$

Where t_n is the expected target value

And \hat{t}_n is the observed target value from the model

LOGISTIC REGRESSION:

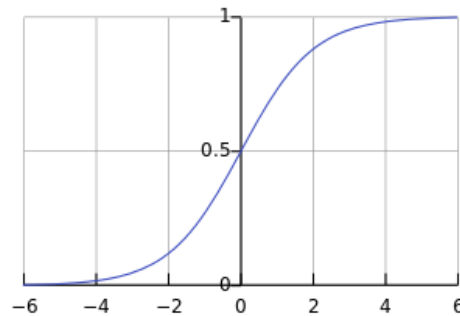
- Logistic regression is used when we want to develop a model that will provide you the probability which can then be mapped to two or more discrete classes. Logistic regression predictions are discrete (only specific values and categories are allowed). We can also view probability scores underlying the model's classification. This is how the graph of logistic regression looks like



- **Sigmoid Activation:** It results the conditional probabilities of the prediction. Sigmoid function maps a real value into another value between 0 and 1. These values can be interpreted as uncertainties for the prediction of 0 and 1. The sigmoid function is given by

$$S(x) = \frac{1}{1+e^{-x}}$$

And the graphical representation of sigmoid is given by



- **Cost Function:** Since we have used sigmoid, the prediction function will be non-linear and gradient descent will not lead us to the optimum minimum. Therefore, instead of Mean Squared Error, we use cost function called Cross-Entropy, also termed as log loss. Cost function can be given by

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [h_{\theta}(x^{(i)}) - y^{(i)}] x_j^{(i)}$$

- **Accuracy:** It measures how correct our predictions were by comparing the predicted labels with the actual labels and dividing it by the total number of labels

$$\text{accuracy} = (\text{count}/\text{len}(\text{predicted_labels})) * 100$$

where count is the occurrence where predicted label matches with the actual label.

ARTIFICIAL NEURAL NETWORKS:

- **Error Function/Loss function:** Use to determine a node in the graph where there lies the minimum value of the error using “Gradient Descent” methodology. It basically tells us that how well our model will predict the expected output.
- **Activation Function:** Activation function is basically used for controlling the output traffic i.e it basically decides whether the output need to be activated or not on the basis of the weighted sum calculated for that particular node. The deciding factor for the activation of a node is the threshold value. If the weighted sum is beyond the threshold value, that node will get activated.

STOCHASTIC GRADIENT DESCENT SOLUTION USING LINEAR REGRESSION

Step 1: First step is the initialization of the random weights for the nodes. In SGD, we update the weights after each training sample.

Step 2: We will then update the weights iteratively using below equation

$$\nabla E = \nabla E_D + \lambda \nabla E_W$$

$$\text{where } \nabla E_D = -\{t_n - w^T \phi(\lambda_n) \phi(x_n)\}$$

$$\text{and } \nabla E_W = w^T$$

$$\text{therefore, } \lambda \nabla E_W = \lambda \cdot w^T$$

Step 3: Here we will determine the target value \mathbf{t} using $\mathbf{t} = \mathbf{w}^T \phi(\mathbf{x})$, where $\phi(\mathbf{x})$ will vary for training, validation and testing dataset.

Step 4: We need to calculate the ERMS now by using

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

where w^* is the solution and N_V is the size of the test dataset.

$$\text{and } E(w^*) = \frac{1}{2} (t_n - \hat{t}_n)^2$$

Step 5: Similarly, we will calculate the E_{RMS} for validation and testing data set.

LOGISTIC REGRESSION TO FIND MAXIMUM ACCURACY

Step 1: After segregating the dataset we will train our model passing initial weights, learning rate and epoch size in order to obtain the updated weights and cost history.

Step 2: We will then calculate the cost function using updated weights, features and labels to obtain the minimal cost. Given the equation $J(\theta) = \frac{1}{m} \sum_{i=1}^m [h_{\theta}(x^{(i)}) - y^{(i)}] x_j^{(i)}$

$$\begin{aligned} \text{Cost}(h_{\theta}(x), y) &= -\log(h_{\theta}(x)) && \text{if } y=1 \\ \text{Cost}(h_{\theta}(x), y) &= -\log(1 - h_{\theta}(x)) && \text{if } y=0 \end{aligned}$$

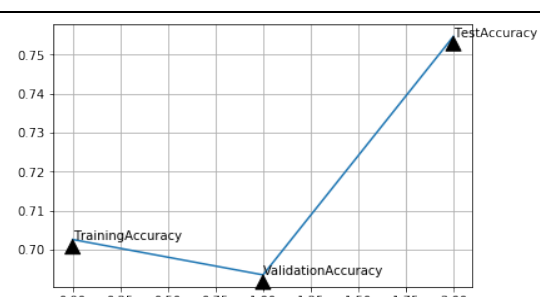
Step 3: Third step is to calculate the predicted weights by passing features and weights using the equation of Sigmoidal function.

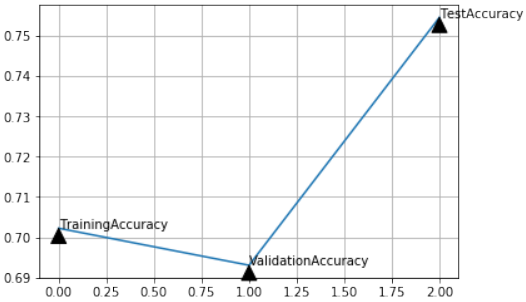
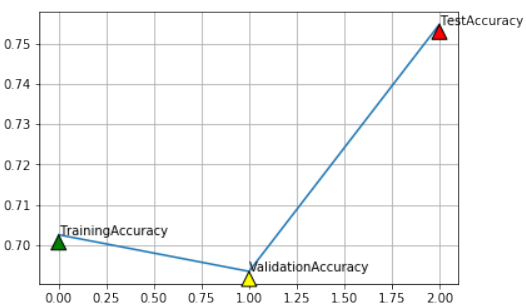
Step 4: By using the predicted labels and actual labels, we will compute the accuracy of our model. Since cost function penalizes wrong predictions more than it rewards the right predictions due to which increasing prediction accuracy (closer to 0 or 1) has diminishing returns on reducing cost due to logistic nature of cost function.

Observations

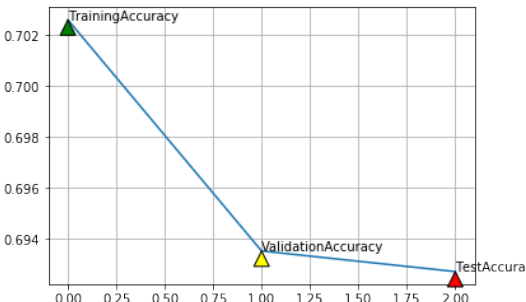
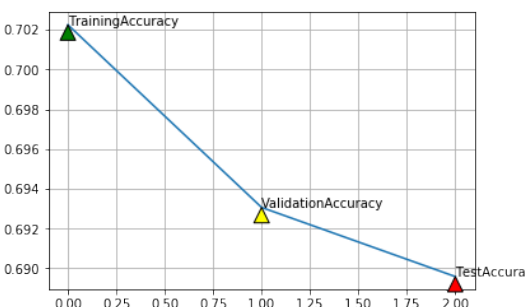
- Tuning hyperparameters in stochastic gradient descent using **Linear Regression** for Human Observed Dataset for minimal mean squared error.

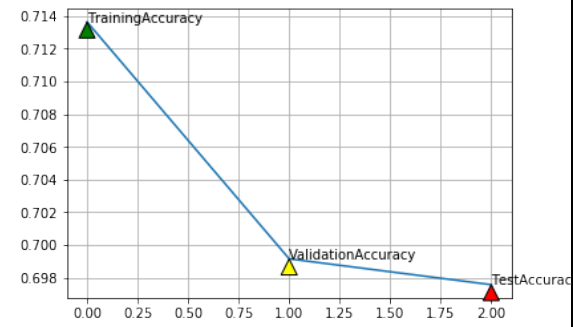
CONCATENATE

Hyperparameters Tuned	E_rms Training	E_rms Validation	E_rms Testing	Graph Plots
Regularizer(La)=1 learningRate=0.01 Size of epochs=200	0.70332	0.69353	0.75472	

Regularizer(La)=3 learningRate=0.5 Size of epochs=500	0.70225	0.69306	0.75458	
Regularizer(La)=9 learningRate=0.01 Size of epochs=50	0.70262	0.69355	0.75473	

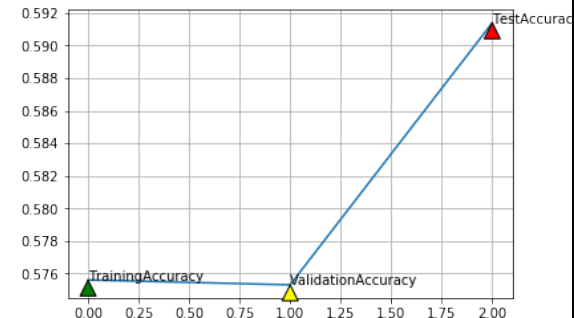
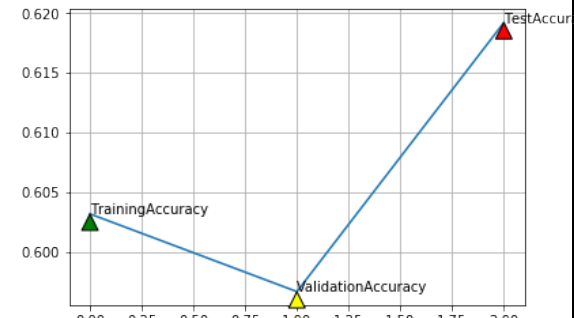
SUBTRACT

Hyperparameter s Tuned	E_rms Training	E_rms Validation	E_rms Testing	Graph Plots
Regularizer(La)=1 learningRate=0.01 Size of epochs=200	0.70261	0.69353	0.69272	
Regularizer(La)=3 learningRate=0.5 Size of epochs=500	0.70225	0.69306	0.68958	

Regularizer(La)= 5 learningRate= 0.005 Size of epochs= 400	0.71363	0.69916	0.69758	 <table><tr><th>Epoch</th><th>Training Accuracy</th><th>Validation Accuracy</th><th>Test Accuracy</th></tr><tr><td>0.00</td><td>0.71363</td><td>-</td><td>-</td></tr><tr><td>1.00</td><td>-</td><td>0.69916</td><td>-</td></tr><tr><td>2.00</td><td>-</td><td>-</td><td>0.69758</td></tr></table>	Epoch	Training Accuracy	Validation Accuracy	Test Accuracy	0.00	0.71363	-	-	1.00	-	0.69916	-	2.00	-	-	0.69758
Epoch	Training Accuracy	Validation Accuracy	Test Accuracy																	
0.00	0.71363	-	-																	
1.00	-	0.69916	-																	
2.00	-	-	0.69758																	

- Tuning hyperparameters in gradient descent using **Linear Regression** for GSC Dataset for minimal mean squared error.

CONCATENATE

Hyperparameters Tuned	E_rms Training	E_rms Validation	E_rms Testing	Graph Plots																
Regularizer(La)=1 learningRate=0.01 Size of epochs=100	0.5756	0.5753	0.5914	 <table><thead><tr><th>Epoch</th><th>Training Accuracy</th><th>Validation Accuracy</th><th>Test Accuracy</th></tr></thead><tbody><tr><td>0.00</td><td>0.5756</td><td>-</td><td>-</td></tr><tr><td>1.00</td><td>-</td><td>0.5753</td><td>-</td></tr><tr><td>2.00</td><td>-</td><td>-</td><td>0.5914</td></tr></tbody></table>	Epoch	Training Accuracy	Validation Accuracy	Test Accuracy	0.00	0.5756	-	-	1.00	-	0.5753	-	2.00	-	-	0.5914
Epoch	Training Accuracy	Validation Accuracy	Test Accuracy																	
0.00	0.5756	-	-																	
1.00	-	0.5753	-																	
2.00	-	-	0.5914																	
Regularizer(La)=3 learningRate=0.5 Size of epochs=500	0.6032	0.5967	0.6192	 <table><thead><tr><th>Epoch</th><th>Training Accuracy</th><th>Validation Accuracy</th><th>Test Accuracy</th></tr></thead><tbody><tr><td>0.00</td><td>0.6032</td><td>-</td><td>-</td></tr><tr><td>1.00</td><td>-</td><td>0.5967</td><td>-</td></tr><tr><td>2.00</td><td>-</td><td>-</td><td>0.6192</td></tr></tbody></table>	Epoch	Training Accuracy	Validation Accuracy	Test Accuracy	0.00	0.6032	-	-	1.00	-	0.5967	-	2.00	-	-	0.6192
Epoch	Training Accuracy	Validation Accuracy	Test Accuracy																	
0.00	0.6032	-	-																	
1.00	-	0.5967	-																	
2.00	-	-	0.6192																	

SUBTRACT

Hyperparameter s Tuned	E_rms Training	E_rms Validation	E_rms Testing	Graph Plots
Regularizer(La)=1 learningRate=0.01 Size of epochs=200	0.5714	0.5564	0.5189	
Regularizer(La)=3 learningRate=0.5 Size of epochs=500	0.6225	0.6190	0.6058	

- Tuning hyperparameters in **Logistic Regression** for Human Observed Dataset for maximum accuracy.

CONCATENATE

Hyperparameters	Accuracy	Graph
Learning Rate=0.5 Epochs=400	Accuracy: 76.675094	

Learning Rate=0.002 Epochs=100	Accuracy: 75. 853350	
-----------------------------------	-------------------------	--

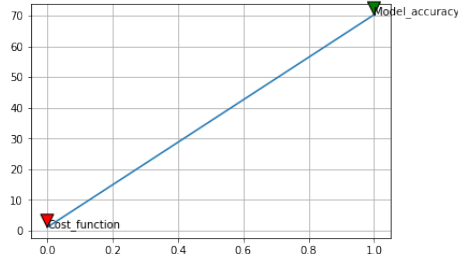
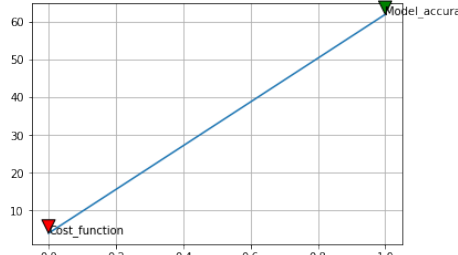
SUBTRACT

Hyperparameters	Accuracy	Graph
Learning Rate=0.001 Epochs=300	Accuracy: 68. 2383246	
Learning Rate=0.8 Epochs=600	Accuracy: 71. 8320646	

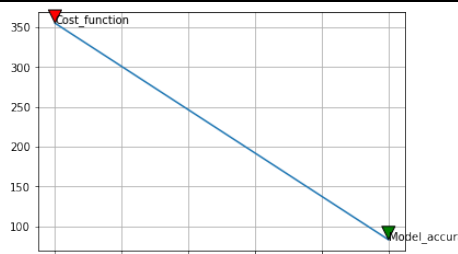
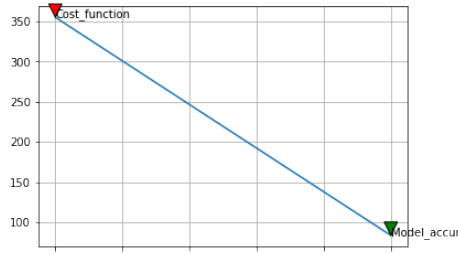
- Tuning hyperparameters in **Logistic Regression** for GSC Dataset for maximum accuracy.

CONCATENATE

Hyperparameters	Accuracy	Graph

Learning Rate=0.2 Epochs=400	Accuracy: 70.2 33884	
Learning Rate=0.5 Epochs=200	Accuracy: 61.9 71732535544035	

SUBTRACT

Hyperparameters	Accuracy	Graph
Learning Rate=0.1 Epochs=200	Accuracy: 83.38 0632	
Learning Rate=0.005 Epochs= 1000	Accuracy: 83.71 475304413471	

ARTIFICIAL NUERAL NETWORKS OBSERVATIONS:

For Concatenated feature in Human Observed Dataset.

Hyperparameters	Accuracy
Number of neuron hidden layer=2000 Learning Rate=0.02, No. of epochs=7500, Batch size=50	Accuracy: 54.43037974683544

For Subtracted features in Human Observed Dataset.

Hyperparameters	Accuracy
Number of neuron hidden layer=1000 Learning Rate=0.01, No. of epochs=900, Batch size=100	Accuracy: 62.65544366878769

For Concatenated feature in GSC Dataset.

Hyperparameters	Accuracy
Number of neuron hidden layer=1000, Learning Rate=0.03, No. of epochs=2500, Batch size=200	Accuracy: 81.78772974988764

For Subtracted features in GSC Dataset.

Hyperparameters	Accuracy
Number of neuron hidden layer=1000 Learning Rate=0.08, No. of epochs=2000, Batch size=800	Accuracy: 76.76655078774553

Conclusion

We have achieved an optimal model with the minimal fluctuations on tuning the hyperparameters for Linear Regression, Logistic regression and Neural Networks.

References

- [1] <https://www.statisticssolutions.com/what-is-logistic-regression/>
- [2] Pattern Recognition and Machine Learning Christopher M Bishop
- [3] <http://www.numpy.org/>
- [4] https://ml-readthedocs.io/en/latest/logistic_regression.html#sigmoid-activation/