CSE 574

INTRODUCTION TO MACHINE LEARNING

PROJECT - 2

Sai Saket Regulapati

UBID: saisaket

UB Person Number: 50286747

Table of Contents

CHAPTER I - LINEAR REGRESSION USING GRADIENT DESCENT METHOD

LINEAR REGRESSION is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables:

- One variable, denoted *x*, is regarded as the **predictor**, **explanatory**, or **independent** variable.
- The other variable, denoted *y*, is regarded as the **response**, **outcome**, or **dependent** variable.

When the Linear Regression is dependent on one predictor variable it is called Simple Linear Regression, whereas if it is dependent on multiple predictor variables it is called **Multivariate Linear Regression.**

In Simple Linear Regression we represent dependent variable as the linear function of the predictor variable. In Multivariate Linear Regression we transform the multiple predictor variables into a single tensor and make the outcome a linear function of that tensor.

Through Linear Regression modelling, we will be able to predict new variables based on the data present. In the Current problem for the Linear Regression problem we use the **Gradient Descent** solution.

DATA PREPROCESSING:

Reading Data: The data is read from the CSV file and the feature variables are separated from the target variables and are stored in two lists for our convenience.

Remove Collinearity: Regression model will try to overfit when you have highly correlated input variables.

Removing Zero Columns: We try to remove the columns from the data features because that might result in the formation of singular matrix. Then formation of Design matrix will be impossible. How the processing is done is explained in the Gradient Descent solution.

GRADIENT DESCENT SOLUTION:

The gradient descent function makes use of the Basis functions, Co-Variance Matrix and the Design Matrix. These are explained below:

Basis function:

"Every continuous function in the function space can be represented as a linear combination of basis functions, just as every vector in a vector space can be represented as a linear combination of basis vectors."

Basis function is an element of basis for function space. It is mainly used to linearize the method.

Co-Variance matrix:

The co-variance matrix consists of relation/variation of a feature with respect to itself. The covariance matrix only consists of diagonal elements on our case. Because variance w.r.t. is other features is not needed for us. The matrix will be a diagonal matrix with diagonal elements representing the variances of the features.

- Co-Variance Matrix can be derived from the GetBigSigma function of the program
- In the problem for the GSC dataset, I have added a little noise to rows so that I get no singular matrix.

Design matrix:

Design matrix consists of features and the corresponding values of the features throughout the dataset.

- Design matrix can be derived from GetPhiMatrix function of program

The Gradient Descent Solution:

In the gradient descent solution, we calculate the weights for the prediction, count the mean error, multiplied with learning rate and change the weights accordingly. This process is done until the weights cannot be further decreased.

The formula for Gradient Descent solution is mentioned below:

$$W_{(r+1)} = W_r - \eta * \nabla E(W_{(T)})$$

The derivative ∇E is explained below.

Here we shall consider the problem of evaluating ∇En(w) for one such term in the error function. This may be used directly for sequential optimization, or the results can be accumulated over the training set in the case of batch methods.

$$E(w)=\Sigma E_n(w)$$

Consider first a simple linear model in which the outputs yk are linear combinations of the input variables xi so that

$$Y_k=\Sigma W_{ki} X_i$$

Together with an error function that, for a particular input pattern n, takes the form

$$En = ½(\Sigma Y_{nk}-T_{nk})^2$$

Where T is the target variable,

The gradient of this error function with respect to a weight Wji is given by

$$\partial E_n/\partial W_{ji} = (Y_{nj} – T_{nj})X_{ni}$$

OUTPUT With GSC dataset:

Output 1 (Concatenation):

```
----------Gradient Descent Solution--------------------
M =  10
Lambda  =  0.03
eta=  20
E_rms Training   = 0.29755
E_rms Validation = 0.30541
E_rms Testing    = 0.2922
Accuracy TR      = 97.56554
Accuracy Val     = 96.0
Accuracy Test    = 97.48744
```

Output 1 (Subtraction):

```
----------Gradient Descent Solution--------------------
M = 10
Lambda  = 0.03
eta=0.01
E_rms Training   = 0.36229
E_rms Validation = 0.37957
E_rms Testing    = 0.37295
Accuracy TR      = 83.83271
Accuracy Val     = 78.5
Accuracy Test    = 79.8995
```

Output 2 (Concatenation):

```
----------Gradient Descent Solution--------------------
M =  10
Lambda  =  0.02
eta=  2
E_rms Training   = 0.29314
E_rms Validation = 0.28968
E_rms Testing    = 0.25926
Accuracy TR      = 98.43945
Accuracy Val     = 99.0
Accuracy Test    = 99.49749
```

Output 2 (Subtraction):

```
----------Gradient Descent Solution--------------------
M =  10
Lambda  =  0.02
eta= 2
E_rms Training   = 0.29178
E_rms Validation = 0.27841
E_rms Testing    = 0.297
Accuracy TR      = 92.82147
Accuracy Val     = 95.0
Accuracy Test    = 89.94975
```

Output with Human Feature Data:

Output 3 (Concatenation):

```
----------Gradient Descent Solution for Human Data--------------------
M =  10
Lambda  =  0.03
eta= 20
E_rms Training   = 0.50836
E_rms Validation = 0.50275
E_rms Testing    = 0.52675
Accuracy TR      = 66.14049
Accuracy Val     = 64.55696
Accuracy Test    = 63.05732
```

Output 3 (Subtraction):

```
----------Gradient Descent Solution for Human Data--------------------
M =  10
Lambda  =  0.03
eta= 20
E_rms Training   = 0.51955
E_rms Validation = 0.4946
E_rms Testing    = 0.50793
Accuracy TR      = 67.87687
Accuracy Val     = 69.62025
Accuracy Test    = 68.78981
```

Output 4 (Concatenation):

```
----------Gradient Descent Solution for Human Data--------------------
M =  10
Lambda  =  0.02
eta= 10
E_rms Training   = 0.38852
E_rms Validation = 0.41914
E_rms Testing    = 0.42589
Accuracy TR      = 77.66377
Accuracy Val     = 75.94937
Accuracy Test    = 73.24841
```

Output 4 (Subtract):

```
----------Gradient Descent Solution for Human Data--------------------
M =  10
Lambda  =  0.02
eta= 10
E_rms Training   = 0.45469
E_rms Validation = 0.47274
E_rms Testing    = 0.43335
Accuracy TR      = 71.90213
Accuracy Val     = 72.1519
Accuracy Test    = 72.61146
```

CHAPTER II – NEURAL NETWORK

Neural Network: An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It itself is not an algorithm but uses many machine learning paradigms.

In the Neural Network we transfer data from one layer to another. We process the data while doing it. There are mainly three layers Input layer, Neural layer and the Output layer.

INITIALIZING WEIGHTS:

When we are about to transmit the data from one layer to another we use weights. These weights are generated in a randomized fashion by the use of normal distribution made from number of nodes in both sides. These weights help us in predicting the unknown values that we have to guess.

Consider input_weights. These are assigned when transmission between input and hidden layer is done. So the weights are generated here by the number of input layer nodes, hidden layer nodes. A normal distribution is made and random values are derived from it. The same with output_weights.

COMPUTING VALUES:

Values are computed in the hidden layer by the use of matmul function. This multiplies the input tensors and the hidden weights assigned to them and passes these values to the hidden layer.

Now these computed values are passed into an ACTIVATION FUNCTION. An activation function of nodes decides the output of the node based on the given inputs. There are various activation functions that can be used like Rectified Linear Unit (ReLU), Sinusoid, Sinc and Gaussian. In our problem we have used the ReLU activation function. Now these values after activation function are sent to the output layer after matmul function with output weights.

ERROR FUNCTION:

The error function uses soft_max squashes the range of output to be in between 0 and 1. All the outputs are fitted into this range. Then the reduce_mean function is used to predict the margin of error present in between the expected values and real values.

TRAINING:

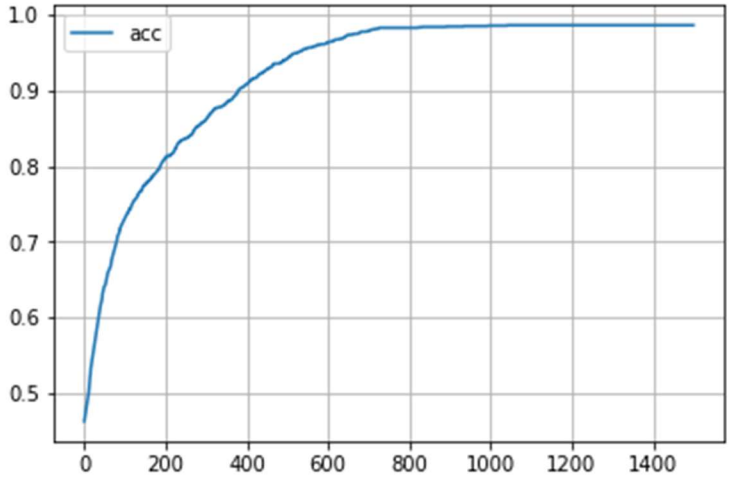
In the data training process the method makes use of error range generated from error function and tries to minimize the error by changing the weight of the transmitted values. These weights are changed by every epoch. An epoch is one complete iteration of complete training dataset. This also uses the GradientDescentOptimizer

PREDICTION:

Prediction makes use of the argmax to makes predictions by the use of the outputs from output_layer.

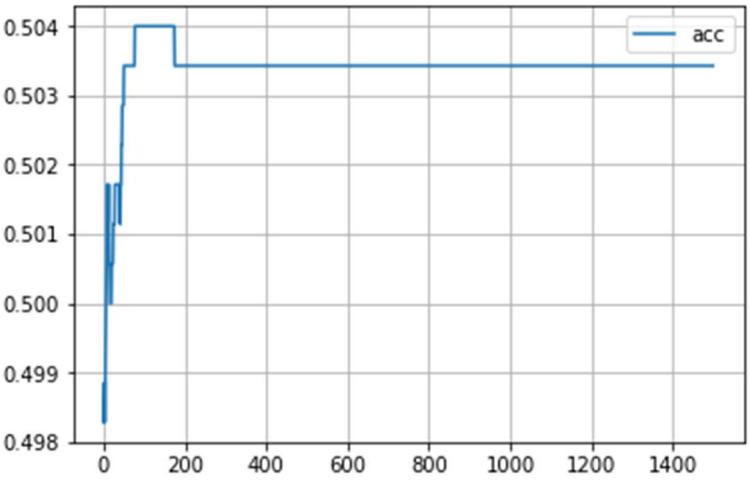
OUTPUT with GSC DataSet:
The epochs for all the categories has been said to 1500 and the batch size to 120.

Output 1 (Concatenation):
Learning Rate = 0.000075
Neural Nodes = 1250



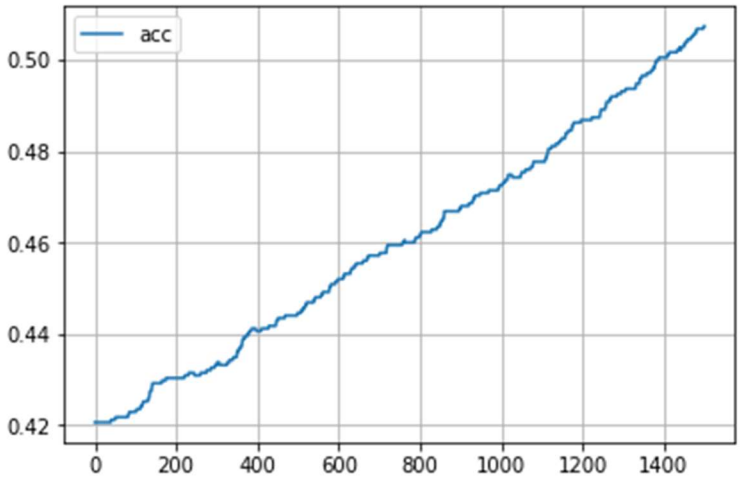

```
Errors: 4  Correct :248
Testing Accuracy: 98.4126984126984
```

Output 1 (Subtraction):
Parameters same as above.



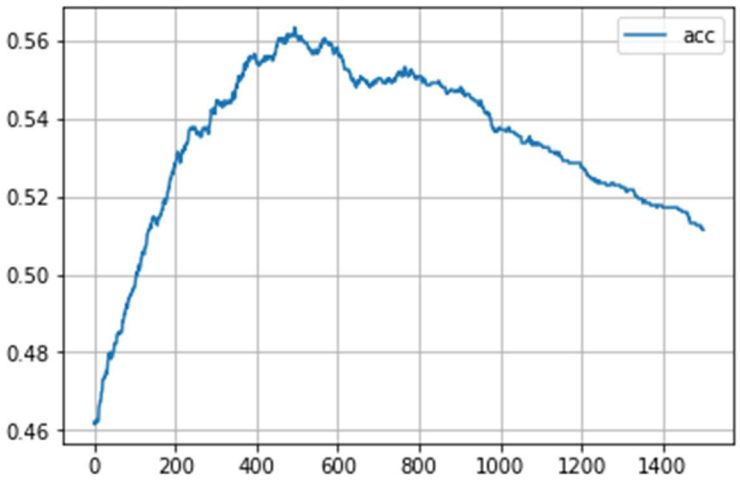

```
Errors: 132  Correct :120
Testing Accuracy: 47.61904761904761
```

Output 2 (Concatenation):
Neural Nodes=100
Learning Rate = 0.000025




```
Errors: 116  Correct :136
Testing Accuracy: 53.96825396825397
```

Output 2 (Subtraction):
Parameters same as above.

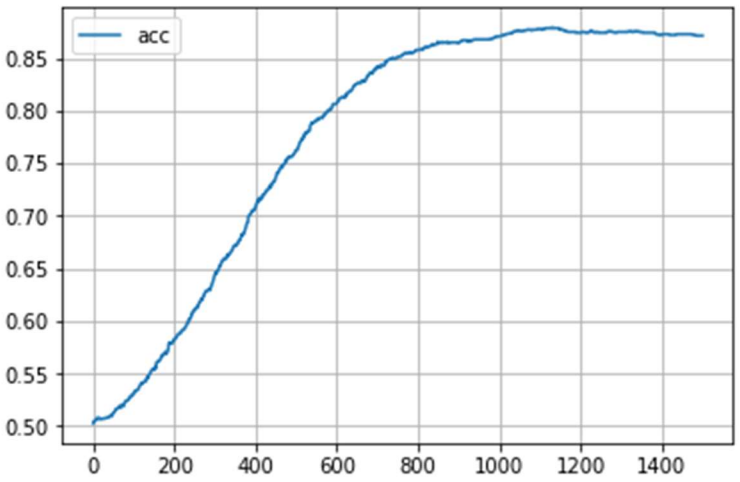



```
Errors: 111  Correct :141
Testing Accuracy: 55.952380952380956
```

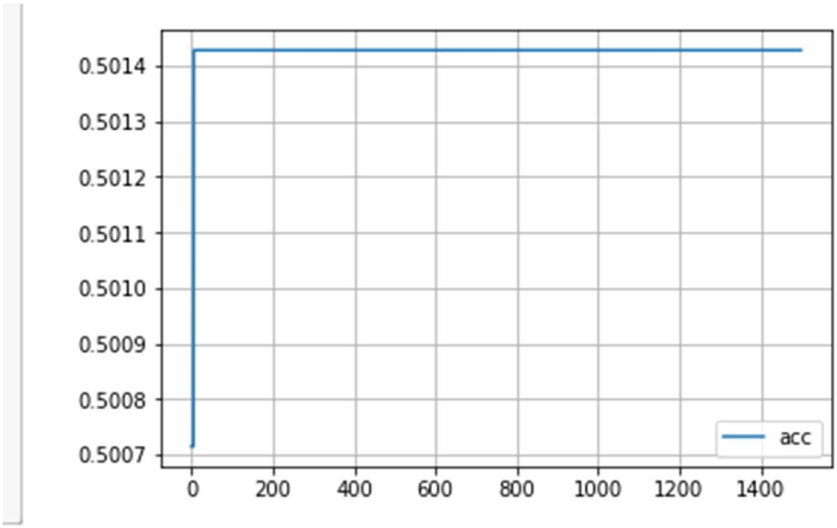
Output for the Human Dataset:

Output 3 (Concatenation):
Learning Rate = 0.000075
Hidden Nodes=250

```
Errors: 26  Correct :157
Testing Accuracy: 85.79234972677595
```



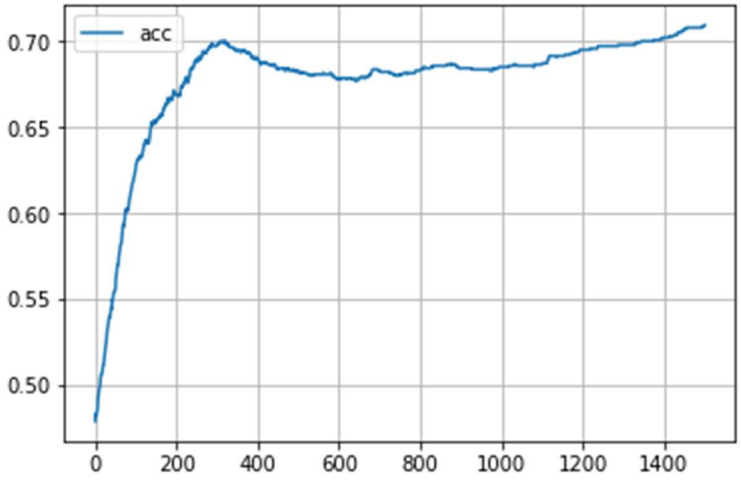
Output 3 (Subtraction):
Parameters same as above.



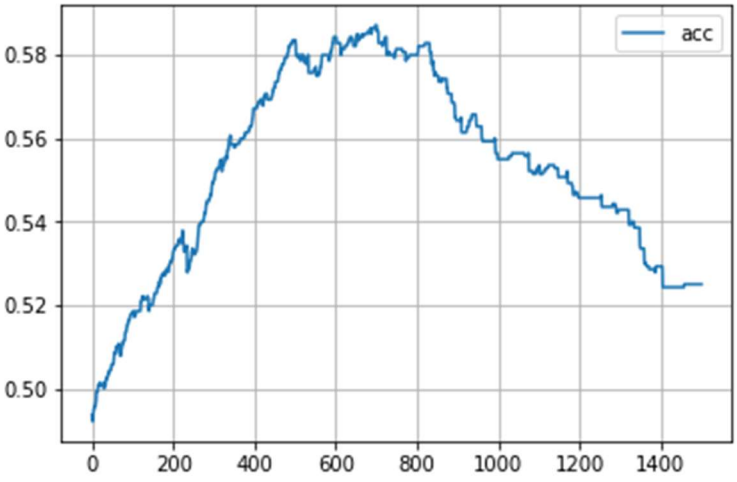

```
Errors: 94  Correct :89
Testing Accuracy: 48.63387978142077
```

Output 4(Concatenation):
Hidden Nodes=100
Learning Rate=0.000075

```
Errors: 60  Correct :123
Testing Accuracy: 67.21311475409836
```



Output 4 (Subtraction):
Hidden Nodes:10
Learning Rate:0.000025




```
Errors: 90  Correct :93
Testing Accuracy: 50.81967213114754
```

CHAPTER III – LOGISTIC REGRESSION

Logistic Regression:

In regression analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a logistic model; it is a form of binomial regression. Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, win/lose, alive/dead or healthy/sick.

These are represented by an indicator variable, where the two values are labeled "0" and "1". In our case the same pairs have been represented by 1 and different pairs by 0.

Loss function:

Functions have parameters/weights (represented by theta in our notation) and we want to find the best values for them.

To start we pick random values and we need a way to measure how well the algorithm performs using those random weights. That measure is computed using the loss function, defined as:

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

Even in the Logistic Regression we reduce the loss function using the Gradient Descent method which has already been used and explained in the Linear Regression part. In this the partial derivative is formulated below:

$$\frac{\delta J(\theta)}{\delta \theta_j} = \frac{1}{m} X^T (g(X\theta) - y)$$

Predictions:

By calling the sigmoid function we get the probability that some input x belongs to class 1. Let's take all probabilities ≥ 0.5 = class 1 and all probabilities < 0.5 = class 0. This threshold should be defined depending on the business problem we were working.

Outputs using GSC Dataset:

Output 1 (Concatenation):

Learning Rate: 0.1
Iterations: 1000

Accuracy:0.988

Output 1 (Subtraction):

Parameters same as above.

Accuracy:0.99

Output 2 (Concatenation):

Learning Rate: 0.05
Iterations: 1000
Accuracy: 0.99

Output 2 (Subtraction):

Parameters same as above.

Accuracy: 0.9722

Outputs using Human Dataset:

Output 3 (Concatenation):

Learning Rate: 0.1
Iterations: 300
Accuracy: 85.8

Output 3 (Subtraction):

Parameters same as above.
Accuracy: 61.7

Output 4 (Concatenation):

Learning Rate: 0.1
Iterations: 3000
Accuracy: 90.7

Output 4 (Subtraction):

Parameters same as above.
Accuracy: 83.1

REFERENCES:

[1]: https://en.wikipedia.org/wiki/Logistic_regression

[2]: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

[3]: https://onlinecourses.science.psu.edu/stat501/node/251/

[4]: https://machinelearningmastery.com/gradient-descent-for-machine-learning/

[5]: https://medium.com/@karpathy/software-2-0-a64152b37c35

[6]: Pattern Learning and Machine Learning by CHRISTOPHER M. BISHOP