Kshitij Chhajed

10th October 2021

1. **Problem statement:**
   Train using Logistic Regression: Use Gradient Descent for logistic regression to train the model using a group of hyper-parameters. Use the model to classify whether a patient has diabetes(class 1) or not (class 0).

2. **Dataset:**
   In order to implement logistic regression, dataset with data of 768 diabetic and non-diabetic patients was used. I have split the dataset manually and used 60% data for training, 20% for validation and 20% for testing. The dataset contains medical data of female patients above the age of 21 consisting of below 8 features.
   a. **Glucose (Blood Glucose level)**
   b. **Pregnancies (The number of pregnancies the patient has had)**
   c. **Blood Pressure(mm Hg)**
   d. **Skin Thickness(Triceps skin fold thickness (mm))**
   e. **Insulin level**
   f. **BMI (Body Mass Index : weight in kg/(height in m)2)**
   g. **Diabetes Pedigree Function**
   h. **Age (In years)**

3. **Languages/tools used:**
   I have used Python as a language and VScode as the editor.

Part 1: Logistic regression

4. **Methodology and Results:**
   a. **Extracting and pre-processing data:**
      I used Pandas (python library) dataframe and read_csv() method to capture data from CSV file. The data in the file was randomly sampled using sample() method. The original outputs (class1 = patient has diabetes, class0 = patient doesn't have diabetes) are stored in separate array before performing scaling. The sampled data was then scaled using scaling() function so that the values from the data are scaled on an appropriate level which can be helpful for handling the data. The scaling function was defined on the below formula:

      X[i] = (X[i] – X(mean))/Range(X)
      Where Range(X) = X(max) – X(min), X(mean) = mean/average value of all X[i] observations

      The data is then split into 3 parts of 60%, 20% and 20% for training, validation and testing purposes respectively. The splitting is done manually based on the no. of observations available in dataset.

b. **Logistic regression:**

Hyper-parameters (Learning rate (alpha), weights(w), epochs(iterations)) were initialized for preparing the model. Since there were 8 features, a weight vector (9*1) was initialized to zero values. I included the bias (c) also as weight by concatenating the data later with a column vector of ones(1) so that the bias is calculated automatically and is not scaled/distorted. Logistic regression uses the sigmoid function as shown in the below image.

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

Here, x = b0+b1.y; where b0 is constant and y is the independent variable

The cost function is defined as below for logistic regression:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))]$$

This function provides us with the error (deviation from actual) for the model. It needs to be minimum for an ideal model. Hence, this function was minimized using gradient descent algorithm so that the weights could be adjusted such that the model converges near to the real value within lesser number of iterations. The below expression denotes gradient descent implemented in the project:

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

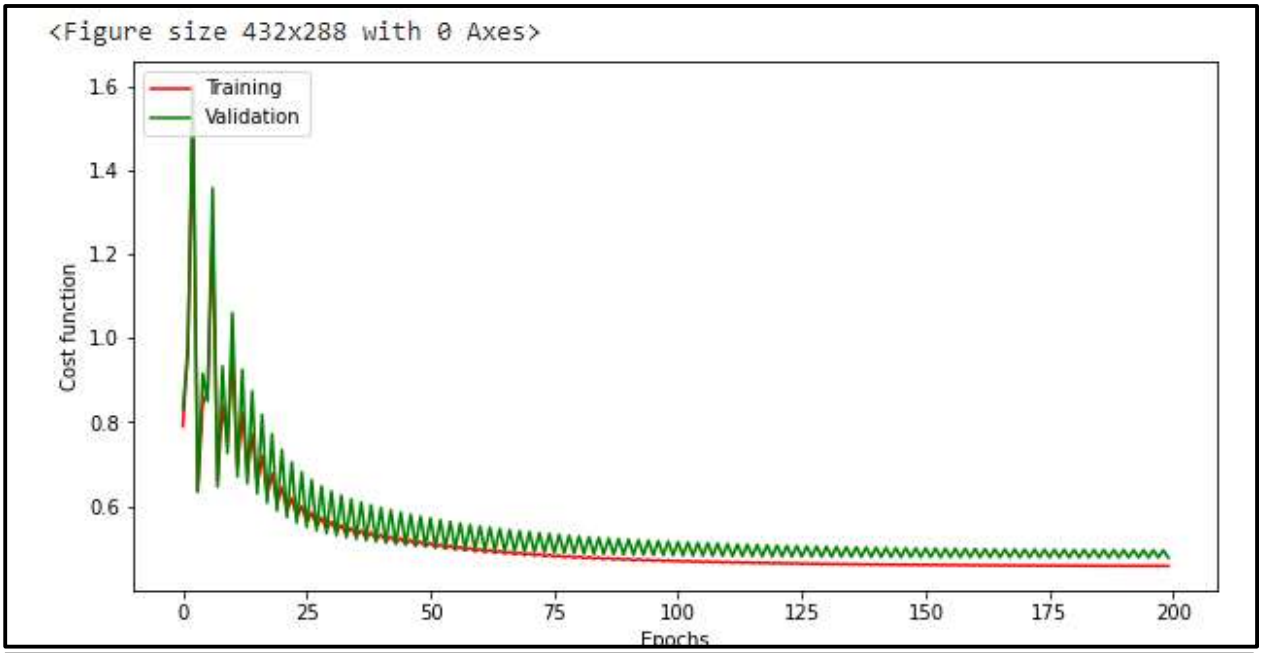
Different functions were prepared for calculation of sigmoid function, cost function. The gradient descent algorithm was looped over 200 number of epochs (iterations). The learning rate (alpha) was kept at 0.03. Every next epoch in the loop was fed with newer values of weights and sigmoid functions so that the model comes nearer to ideal/actual values.

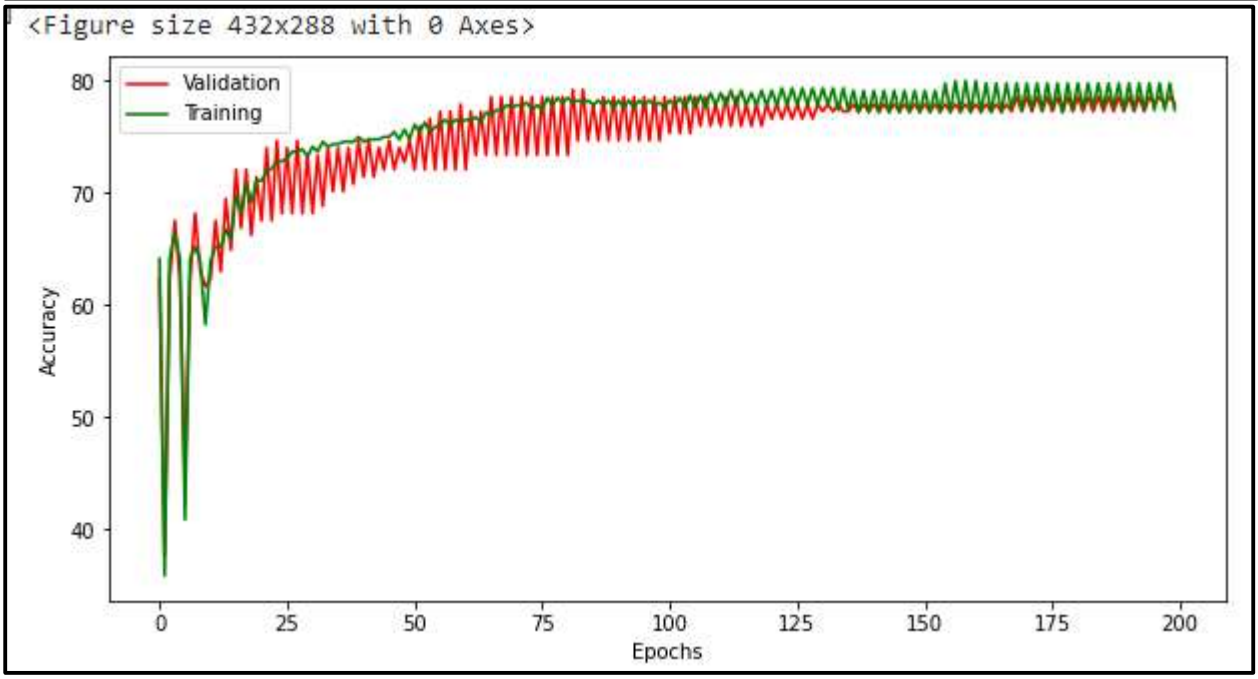
c. **Validation and Testing:**

The model was trained on initial 60% of data and the accuracy of the model was tested for both the validation and training parts of the dataset. Below were some of the results from a recent run:

```
Weights after training:  [[ 4.16068018  8.07628811 -2.17441742 -0.1893491  -1.07384808  5.6506535
   2.86423397 -0.15882371 -0.64035763]]
```

```
Validation dataset accuracy: 77.92207792207793
Training dataset accuracy: 77.39130434782608
Testing dataset accuracy: 75.32467532467533
```





5. **Credits/source:**

Below links were used for the above images. Also the code was designed to implement the algorithms/methods mentioned in the below links:

a. https://medium.com/analytics-vidhya/understanding-logistic-regression-b3c672deac04
b. https://medium.com/data-science-group-iitr/logistic-regression-simplified-9b4efe801389
c. https://www.internalpointers.com/post/cost-function-logistic-regression
d. **Numpy and pandas website for taking help for different methods**
e. **Stackoverflow for frequently occurring errors**