Assignment 1 - Logistic Regression and Neural Networks

Sagar Sonawane - ssonawan@buffalo.edu

10th October 2021

1 Assignment Overview

The task of the assignment is to perform the classification of data with the use of Machine Learning. We have to classify whether the patient has diabetes (class 1) or not (class 0), based on the diagnostic parameters in the dataset. In part 1 of the assignment, I have implemented a logistic regression model using the gradient descent algorithm from scratch. In part 2 of the assignment, I designed a neural network with regularization techniques. In part 3 of the assignment, I compared neural network models designed by L1 regularization and Dropout regularization techniques.

2 Dataset

The dataset used for this assignment is the Pima Indians Diabetes Database (diabetes.csv). The dataset contains medical data of female patients with 8 diagnostic measurements consisting of 768 instances. I have split the dataset into 3 parts - training, testing, and validation data. The training data constitutes 60% of the total data with testing and validation data both consist of 20% each. The training dataset has 460 samples whereas the testing and validation dataset both have 154 samples.

3 Python Editor

I have made use of Jupyter Notebook IDE for the implementation of the program.

4 Implementation

4.1 Pre-processing

I have split the dataset into three parts with the use of train-test-valid-split function from the fast-ml package. The dataset has been split into a 60-20-20 ratio with a randomness state value of 75. Check for any NaN values in the dataset was conducted with a result of 0 NaN values found. Later data was converted from pandas dataframe to numpy array for better implementation of the code.

4.2 Sigmoid function

Sigmoid function is a mathematical function that can take a value as input and map the value between 0 to 1 in an “S” shaped manner. Labels with an outcome of values more than 0.5 can be classified as class 1 and values less than 0.5 as class 0.
Sigmoid function can be expressed as

$$f(x) = 1 / 1+e^{-z}$$

Part 1:

4.2 Gradient Descent algorithm

Gradient Descent algorithm is used to find the optimal values of the parameters. It can be expressed as

w= w-lr*dw and b= b*lr -db

Where dw and db are partial derivatives of the loss function for w and b respectively.

4.3 Accuracy & Cost

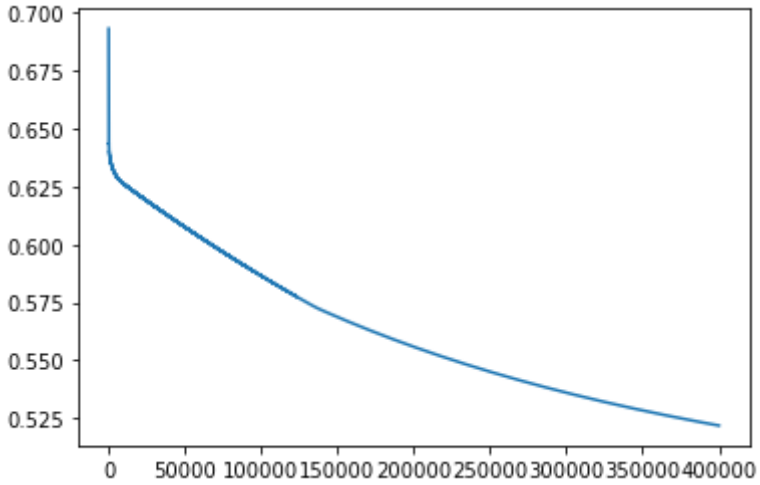
The accuracy of the logistic regression model was calculated by subtracting 1 with the error and multiplying by 100. Error is calculated by taking the absolute sum of (predicted sigma values - actual value) over the total number of observations.

Acc = 1 - ((predicted sigma - Y)/Total observations) * 100

The cost of the model is calculated by the following formula:

$$cost = -\frac{1}{m}\sum_{i=1}^{m}[y * log(a) + (1-y) * log(1-a)]$$

A graph of cost vs iterations is used to study the trend of cost during the training of the model.



5 Conclusion

In conclusion, the logistic regression model gives a 78.57% accuracy for the test data and a 76.62% for the validation data over 400,000 epochs with a learning rate of 0.00026.

PART 2:

4 Implementation

4.1.1 Deep Feedforward Neural Network

Feedforward Neural Network is a type of ANN in which information is passed from the input layer to hidden layers to the output layer. This NN doesn't form any cycle or loops in the network. Deep Feedforward NN makes use of multiple hidden layers.

4.1.2 Layers & Activation functions

There are a total of 3 layers implemented in this NN - Input layer, Hidden layers, Output layer. The input layer consists of input neurons with the same dimensions as the input data in our case 8. I have used 3 hidden layers with 8, 5, and 3 neurons respectively all using the ReLU activation function.

ReLU (Rectified Linear Unit) is a type of non-linear activation function, which outputs the max value between 0 and the input. ReLU function can be expressed as

$$f(x) = \max(0,x)$$

The output layer has only one neuron and uses the sigmoid function as it outputs a binary classification value.

4.1.3 Optimizer

Optimizers are algorithms used to modify the attributes of the NN. I have implemented adam optimizer. Adam optimizer is an adaptive learning rate optimization algorithm designed to train deep neural networks.

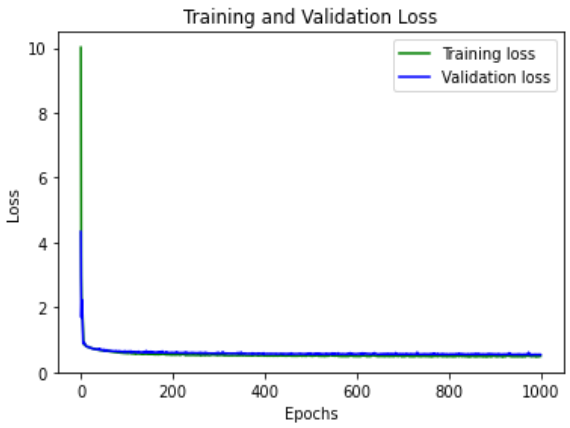
4.2 Regularization

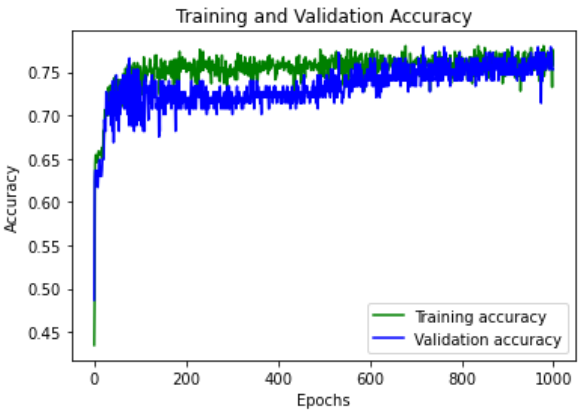
Regularization techniques are used to prevent the overfitting of the model. It adds some penalty to the model proportional to the complexity of the NN. L1 and L2 Regularization are the most common types of regularization.

L1 Regularization known as Lasso Regularization, adds the sum of the absolute weights as the penalty to the loss function. Whereas, L2 regularization known as Ridge Regularization adds the sum of the squared weights as the penalty to the loss function. I have implemented both L1 and L2 regularization to the hidden layers of the model.

4.3 Evaluation

Graphs of training vs validation loss and training vs validation accuracy are plotted to understand the behavior of the model.





5. Conclusion

The Neural Network gave a prediction accuracy of 80.51% for the data over 1000 epochs with a batch size of 128 and a learning rate of 0.01.

PART 3

4 Implementation

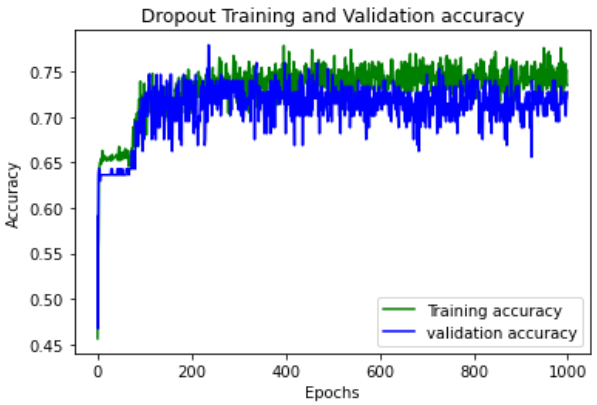
I have designed two models one using L1 regularization and the other using Dropout.

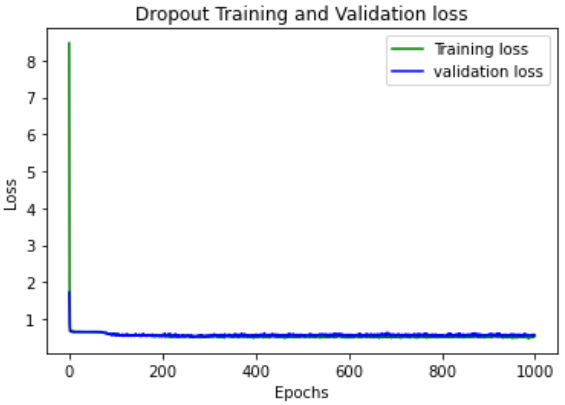
4.1 Dropout Regularization

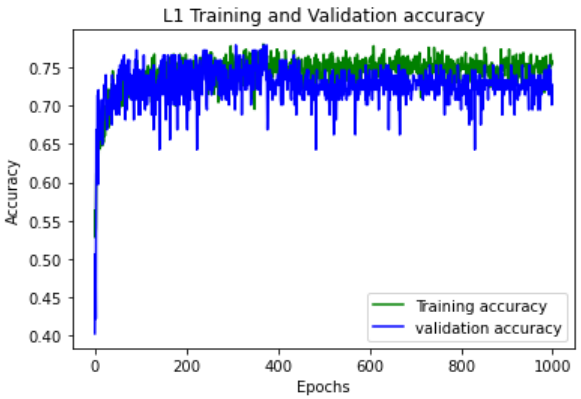
Dropout Regularization is a regularization technique that randomly selects and removes nodes from the model at every iteration, thus having a unique combination of nodes and result after every iteration.

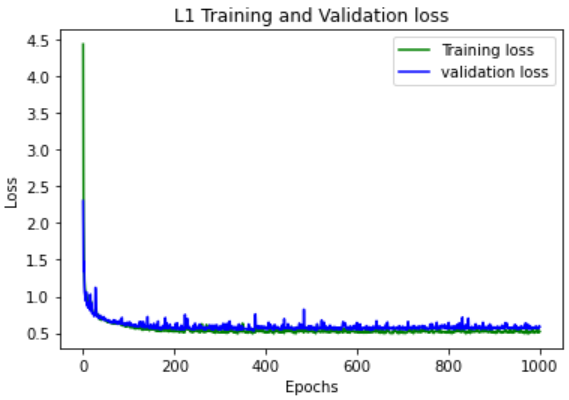
5. Evaluation

Both the models are then compared and evaluated with the help of training vs validation loss and accuracy graphs.

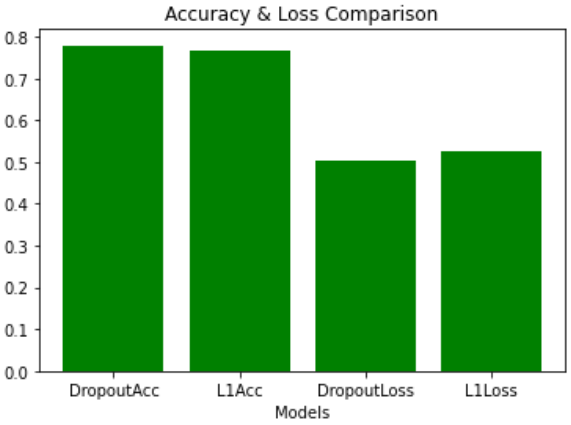








Dropout model showed a plateau of both training and validation accuracy around 50-100 epochs then had a sharp increase as compared to the L1 model which had a gradual increase and stayed more or less the same after 50-100 epochs.
L1 model showed a gradual decrease in both the training and validation loss around 50-100 epochs with some spikes at random intervals. Dropout model again plateaued till around 90 epochs then had a small decrease in both losses with no spikes at random intervals.
A graph comparing the accuracies and losses of Dropout Model and L1 model was plotted.



6. Conclusion

To conclude, the Dropout Model performed better as the Dropout model gave an accuracy of 77.92% and a loss of 50.32% as compared to the L1 model which gave an accuracy of 76.62% and a loss of 52.55%. Both the models ran for 1000 epochs with a batch size of 32 and a learning rate of 0.01.