Diabetes Classification using Machine Learning

Dexson John D'Souza
Department of Computer Science
University at Buffalo
Buffalo, NY 14260
dexsonjo@buffalo.edu

1. Project Overview

To classify whether a person has diabetes, we have used Gradient Descent with Logistic Regression for part 1 of this project. The dataset was split into train(60%), test(20%) and validation(20%) sets. We have performed normalization on the dataset. In part 2, we have used a Neural Network with several layers to perform classification. We have used Relu and Sigmoid activation functions. We have also used L2 regularizers to improve the performance of the model. In part 3, we have implemented 2 separate Neural Networks; one using L2 regularizer and other using Droput.

2. Experimental Setup

To classify whether a person has diabetes, we have used Neural Networks. In deep neural network, input layer, hidden layer, and output layer is present. Pima Indians Diabetes Database dataset was used. We have used Jupyter Notebook which is an open-source software. We have used Scikit-learn which is open-source software for machine learning library in Python programming language to perform data partitioning. We have used Numpy which is library for the Python programming language. We have also used the Python library Pandas.

3. Result Analysis

3.1 Logistic Regression(Part 1)

3.1.1 Result

In this phase, we implemented Gradient Descent from scratch. We created a model that used Gradient Descent for learning. The model was able to achieve 81.16% accuracy for diabetes classification. The hyperparameters: learning rate was 0.01 and number of epochs were 1000.

3.1.2 Activation Function in Logistic Regression

For every iteration during training, we have used Sigmoid function to activate the input. Sigmoid function ensures that our output will be in the range from 0 to 1. This is very beneficial in binary classification problems. For any input 'Z', Sigmoid function is given as:

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

3.1.3 Visualization of training performance

The cost function with respect to the number of iterations for the diabetes classification system using Logistic Regression is depicted in Fig. 1. The X-axis denotes the iterations and Y-axis denotes the cost at a specific iteration.

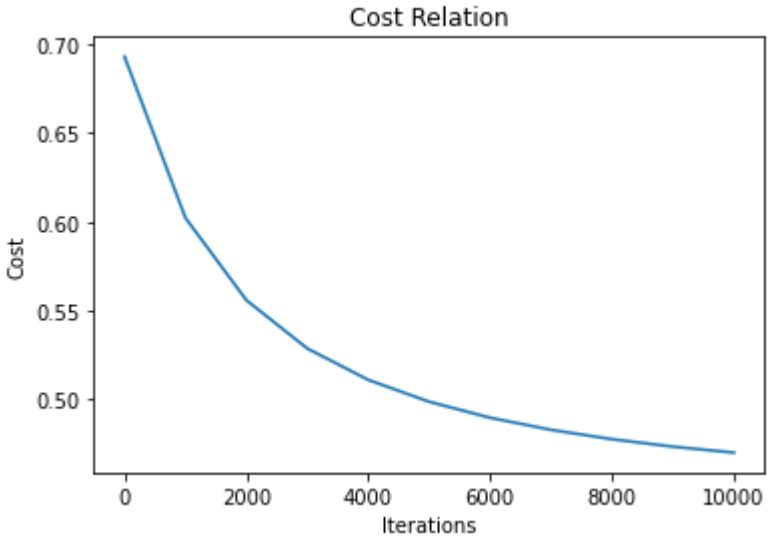

Figure 1: Cost-Iteration Relation for Logistic Regression

3.1.4 Comparative Study

Performance of the model for different hyper-parameter and model parameter is depicted in Table 1.

Table 1: Comparison of the model's performance for different parameter settings.

Learning Rate	No of Iteration	Accuracy Achieved
0.001	10000	81.16
0.001	100	78
0.000001	10000	77.9
0.000001	100	77.9

3.2 Neural Networks with L2 Regularizer(Part 2)

3.2.1 Result

We created a neural network model to perform classification. 2 Hidden layers were used. First layer had 512 nodes and used Relu activation. Second layer had 4 nodes and used Relu as well. The output layer consisted of a single node and used Sigmoid function. Adam optimizer was used for this model. We have used Binary cross entropy loss function to compute the loss. This model was able to achieve 82.46% accuracy for diabetes classification. The hyperparameters: batch_size was 20, number of epochs were 100 and learning rate of Adam optimizer was 0.005.

3.2.2 Visualization of accuracy during training

The accuracy of the Neural Network model with respect to the number of epochs during the training is depicted in Fig. 2. The X-axis denotes the epochs and Y-axis denotes the accuracy at a specific epoch. We have plotted the

training accuracy for training data set as well as training accuracy for validation dataset.

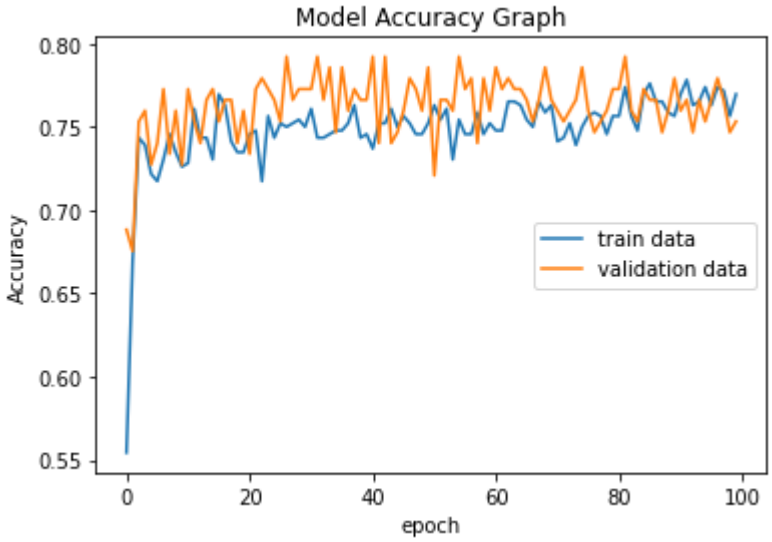

Figure 2: Accuracy vs Epochs

3.2.3 Visualization of loss during training

The loss of the Neural Network model with respect to the number of epochs during the training is depicted in Fig. 3. The X-axis denotes the epochs and Y-axis denotes the loss at a specific epoch. We have plotted training loss for training dataset as well as training loss for validation dataset.

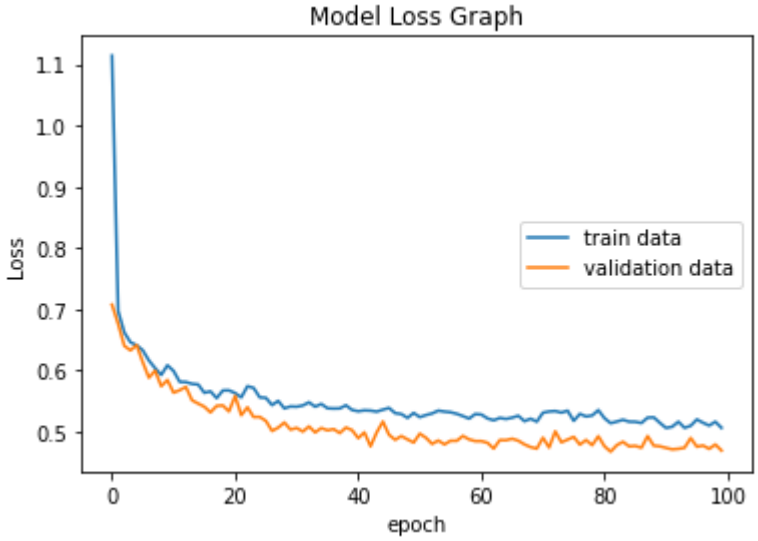

Figure 3: Loss vs Epochs

3.2.4 Comparative Study

Performance of the Neural Network for different hyper-parameter and model parameter is depicted in Table 2.

Table 2: Comparison of the model's performance for different parameter settings.

Learning Rate	L2	Epochs	Accuracy
0.005	0.1	100	82.46
0.005	0.1	1000	81.16
0.005	0.001	100	72.27
0.1	0.001	1000	64.28
0.1	0.1	100	64.28
0.1	0.001	1000	64.28

3.3 Neural Networks with different Regularization techniques (Part 3)

3.3.1 Result

We created a Neural Network model having 1 hidden layer to perform classification. This model used L2 regularizer and consisted of 256 nodes. The accuracy achieved using this model was 76.30%. We created another Neural Network model similar to the first model. The only difference is instead of L2 regularizer, Dropout was used in this model. This mode was able to achieve 87.17% accuracy.

3.3.2 Visualization of accuracy during training

The accuracy of both Neural Network models during training is depicted in Fig. 4. The X-axis denotes the epochs and Y-axis denotes the accuracy at a specific epoch.

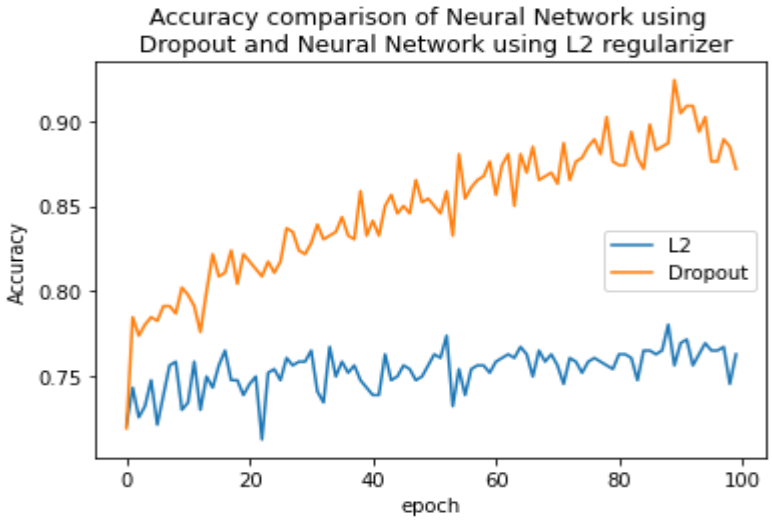

Figure 4: Accuracy vs Epochs

3.3.3 Visualization of loss during training

The loss of both Neural Network models during training is depicted in Fig. 5. The X-axis denotes the epochs and Y-axis denotes the loss at a specific epoch.

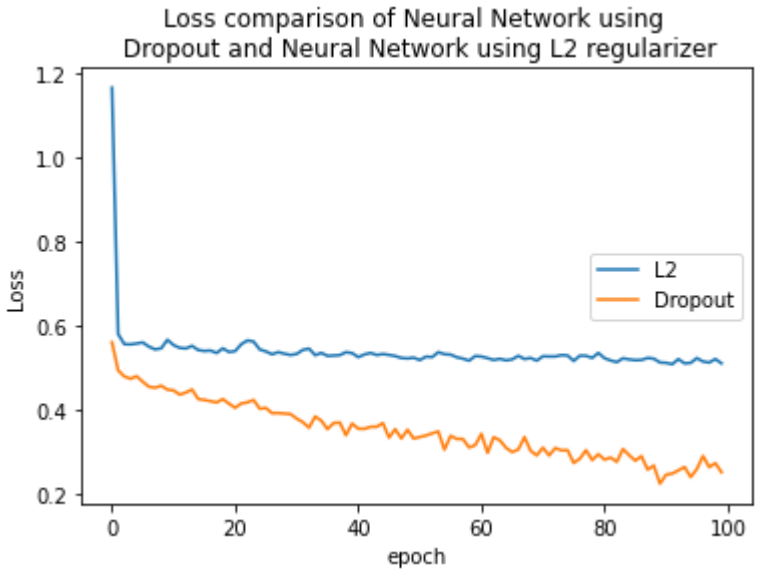

Figure 5: Loss vs Epochs

3.3.3 Comparison of Regularization Methods

Regularizers allow us to apply penalties on layers of Neural Network during training. L2 regularizer adds an L2 penalty which is equal to the square of the magnitude of coefficients. In Dropout, the basic idea is to run each iteration of the algorithm on randomly modified versions of the hidden layers. It improve the robustness of the model by using different neural network configuration for learning at different epochs. Dropout is very effective in preventing the overfitting problem. Therefore, Dropout is more effective than L2 regularizer in improving the performance of the neural network.