Predicting Diabetes with Multilayer Perceptrons

Paridhi Awadheshpratap Singh
Student ID: a1865487
Email: a1865487@adelaide.edu.au

ABSTRACT

This research paper focuses on implementing and analyzing a Python based Multilayer Perceptron (MLP) algorithm to predict diabetes. Diabetes is a health concern that requires predictive models. The study investigates the potential of using MLP in this context. To carry out the analysis the researchers utilized the Pima Indians Diabetes Database, which was pre-processed for classification. The paper provides an overview of the MLP algorithm emphasizing its foundations, activation functions and layer configurations. For validation purposes the research code has been made available, on GitHub with a timestamp. In summary this paper enlightens us about how MLP plays a role in predicting diabetes. It discusses the strengths and limitations of this approach while suggesting areas, for improvement. Ultimately this research contributes to leveraging machine learning for healthcare outcomes.

I. Introduction

Public health is a top priority in protecting communities from diseases that can harm them. Governments allocate a significant portion of economic resources, known as GDP, to ensure the well-being of its people. Efforts such as vaccination help in alleviating such chronic diseases. However, in recent years, there has been a significant increase in chronic and genetic diseases that impact public health. Diabetes is especially dangerous because it can lead to other life-threatening illnesses, such as heart disease, kidney problems, and nerve damage. Diabetes is a metabolic disease that affects the body's ability to control blood sugar levels, also known as blood sugar. This condition is characterized by elevated blood sugar levels due to problems with insulin secretion, insulin action, or both. A complete lack of insulin secretion results in type 1 diabetes (T1D). In type 2 diabetes (T2D), the body is unable to effectively utilize the insulin it produces, resulting in the spread of the disease. Both types of diabetes are increasing, but the increase in T2D cases is more pronounced. T2D accounts for 90-95 percent of all diabetes cases.

Poorly treated diabetes can lead to serious health problems such as stroke, high blood pressure, and heart disease [1].

This paper focuses on the task of predicting diabetes using the Perceptron algorithm, a fundamental component of deep neural networks. We use the Pima Indian population dataset, which is publicly available through the Kaggle website [2]. We have used the pre-scaled version of this dataset from the LIBSVM database [3]. This dataset contains a wide range of clinical and demographic information, making it suitable for predictive modeling. The purpose of this article is to thoroughly investigate the effectiveness of the Perceptron algorithms in predicting diabetes. We delve into how the algorithm works, highlight its strengths and weaknesses, and provide real-world evidence of its performance through a series of experiments and analyses.

II. Method

The Multilayer Perceptron is a classification technique, within feed forward networks. It consists of layers. While the Single Layer Perceptron (SLP) can solve problems it falls short when it comes to handling nonlinear problems. To tackle these problems the Multilayer Perceptron (MLP) is employed. This approach, illustrated in Figure 1 [4] involves a feed forward network with one or more layers. MLP is commonly used for recognizing patterns and classifying input patterns to make predictions. Before the network undergoes training,

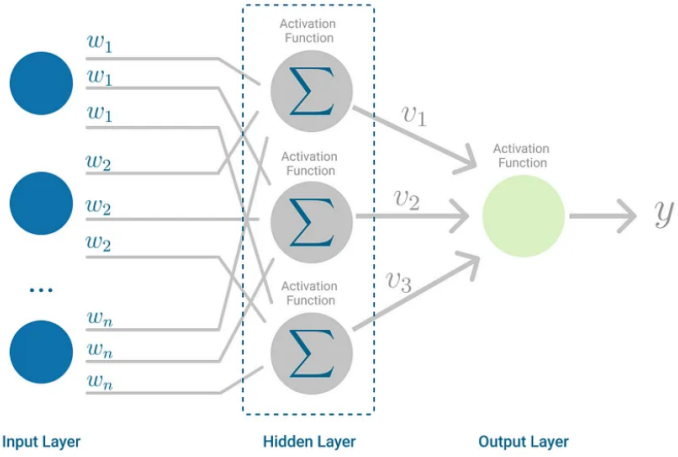

Fig. 1. Multilayer Perceptron Architecture

the weights w_1, w_2,,,,w_n are initialized randomly. Let W be the vector comprising of the randomly generated weights. The vector b represents the bias b_1, b_2,,,,b_n of each neuron in the network. Following this, the neurons learn from a training set X, comprising features x_1, x_2,,,,x_n and serve as input to the network, while y represents the output vector, the neuron's output depends on the weighted sum of all its neurons and can be expressed as:

$$z_i = w_i x_i + b_i \tag{1}$$

An activation function $a(z)$ is applied to the weighted sum z of each layer. The activation function $a(z)$ is typically

a non-linear function. The specific activation functions and the number of neurons in each layer depend on the network architecture and problem at hand [4].

The output is connected to the inputs of other neurons within the hidden layer and is not visible in the output. The data was already prescaled and the output was designated as -1 for absence of diabetes and +1 for presence of diabetes in the last column. The remaining columns represented the features related to the individuals in the study. To maintain quality of the data, we removed the rows with missing or 'nan' values from the dataset. To understand if some of the features can be combined for a better result,we conducted an Explotary Data Analysis by implementing PCA and confirmed using scatterplot. We verified the results with the help of a correlation matrix and they were inconclusive. Categorical values in the data were taken care of.

Before training the model, we split the data into training and testing sets. The partitioning of the data into sets ensures that the model can be implemented on data it has not seen during the training thereby providing a reliable assessment of its generalisation capacities. Here we split the data such that 80 percent of the data was used gor training while the remaining 20 percent was reserved for testing.

We implemented a multilayer perceptron model on the dataset using the PyTorch library in Python. The model architecture consists of three layers:

1. The input layer which has number of neurons equal to the number of features in the dataset.

2. A hidden layer with ReLU(Rectified Linear Unit) activation function for better efficiency and to avoid the vanishing gradient problem.

3. An output layer of a single neuron with a sigmoid activation function that maps the output to a probability score between 0 and 1, making it suitable for binary classification.

The model is then trained using BCELoss (Binary Cross-Entropy Loss) function as the optimisation criterion. BCELoss function measures the error between the predicted and the actual value(here -1 or +1) in the binary classification dataset, making it a suitable choice for binary classification.

We train the model in a loop with a number of iterations, where each iteration represents one complete pass through the training set. After each epoch the model's weights are updates based on the calculated loss, while the optimizer is used such that the weights are updated which results in minimum classification error.

We monitor the training loss throughout the entire process to assess the model's convergence. We also have plotted a loss curve that helps us understand how our model is learning and whether further iterations of training are needed. After training the Perceptron model, we evaluate it on the test set. We evaluate the results by testing its accuracy and calculating the F1 score. The accuracy is the ratio of the correctly classified samples to the total amount of samples in the dataset, while the F1 score determines if our model can classify samples into correct classes. A higher accuracy and F1 score indicates better performance of the model in correctly predicting the status of diabetes.

III. Experimental Analysis

We have used two major evaluation metrics to access the performance of the model namely accuracy and F1 score. While training the model we also experimented with different optimising functions to understand which optimizer suits the dataset best.

1. Accuracy is While training the model we experimented with different activation functions to understand which one suits our model and dataset better. Higher accuracy indicates that the model is efficient at making correct predictions. Accuracy is calculated as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{2}$$

Table 1 shows the accuracy obtained for different optimising functions for different learning rates.

TABLE I
Model Performance

Optimizer	Learning Rate	Accuracy	F1 Score
Adagrad	0.1	83.57	75.66
Adagrad	0.01	80.92	86.51
Adagrad	0.001	69.08	81.71
Adam	0.1	73.68	83.56
Adam	0.01	73.03	82.86
Adam	0.001	80.26	86.11
SGD	0.1	76.97	84.21
SGD	0.01	80.92	87.15
SGD	0.001	69.08	81.71

We have also studied the learning of our model by plotting the loss against the number of iterations. Fig 3, fig 4 and fig 5 represent the graphs of loss obtained using Adagrad, Adam and SGD optimising functions respectively.

F1 score is the other evaluation metric we have used for testing the performance of the model. F1 score combines two metrics: precision and recall. Precision is the ratio of true postive predictions to the total number of positive predictions and is a measure of the accuracy of positive predictions made by the model.

$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} \tag{3}$$

Recall is the ratio of true positive predictions to the total number of actual positive instances and is used to know about the ability of the model to identify all relevant instances of the positive class in the dataset.

$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} \tag{4}$$

The F1 score considers both false positives (precision) and false negatives (recall) and offers a single metric that balances these trade-off and is the harmonic mean of precision and recall. It provides a single value that balances both precision and recall. The harmonic mean is used instead of a simple average to give more weight to lower values.

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

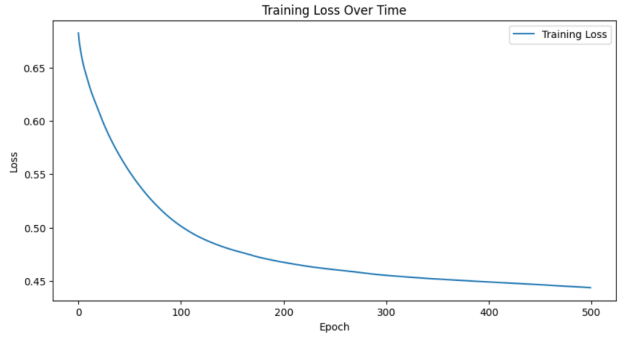

Fig. 2. Training loss using Adagrade

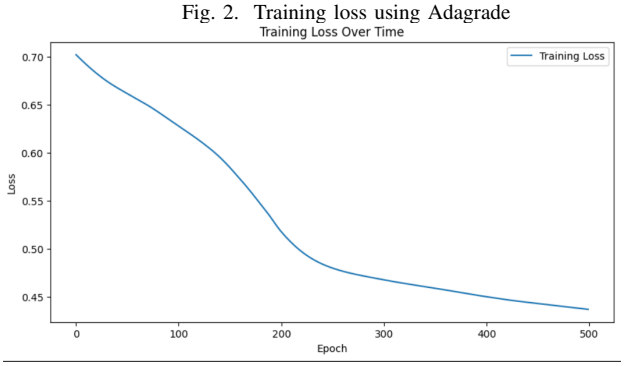

Fig. 3. Training loss using Adam

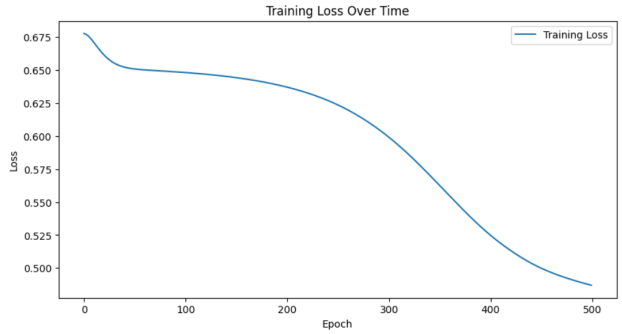

Fig. 4. Training loss using SGD

IV. Code

The code with the implementation of Perceptron algorithm is Python is provided in a GitHub repository accessible here. The code includes all the steps of our perceptron algorithm including pre-processing of dataset, creating the model, training of the model and evaluation of the model by testing it on the test dataset. The access has been given to the tutor jinan.zou@adelaide.edu.au as per the requirements of the assignment and has been timestamped for verification.

V. Conclusion

We conducted experiments where we trained the Perceptron model using various optimization functions. Our findings indicate that Adagrad emerged as the most effective optimizer. Although all optimizers yielded nearly identical peak accuracy levels for different iteration counts, as demonstrated in Figure 2, the training loss curve obtained with Adagrad exhibited the highest level of precision. Moreover, while the SGD optimizer achieved the highest F1 score at 0.01 learning rate (87.15), Adagrad also achieved a comparable F1 score of 86.51 for a learning rate of 0.01 for the same number of iterations, as illustrated in fig 3. Given Adagrad's consistent performance across all three scenarios, we conclude that it is the optimal optimization function for our model.

The results obtained after this study has paved way for more opportunities that are not related just to the world of machine learning. The high accuracy in this study has helped us understand that early detection can help in improving healthcare outcomes. Early detection of diabetes and intervention at the right time can enhance the patients' well-being by introducing personalised plans for every patient. We still believe a lot of improvements can be made by further research. Future research can include using advanced neural network architectures, fine-tuning of hyperparameters or using more features to broaden the scope of the research.

References

[1] U. M. Butt, S. Letchmunan, M. Ali, F. H. Hassan, A. Baqir, H. H. R. Sherazi *et al.*, "Machine learning based diabetes classification and prediction for healthcare applications," *Journal of healthcare engineering*, vol. 2021, 2021.

[2] K. w. UCI Machine Learning, "Pima indians diabetes database."

[3] L. D. C. class), "Pima indians diabetes dataset."

[4] C. Bento, "Multilayer perceptron explained with a real-life example and python code: Sentiment analysis," *Towards Data Science*, 2021.