Prediction of Diabetes using Perceptron

Shavin Kaluthantri
The University of Adelaide
a1904121@adelaide.edu.au

September 29, 2023

Abstract

This study focuses on the development and optimization of Single Layer Perceptron (SLP) and Multi-Layer Perceptron (MLP) models to predict diabetes, utilizing a normalized diabetes dataset. A rigorous process of training, validation, and hyperparameter tuning was employed, with model efficacy evaluated based on accuracy and AUC. The results demonstrated that the simpler SLP model, with specific configurations, outperformed the more complex MLP, achieving a test accuracy of 82.24% and an AUC of 0.8515. The findings highlight the SLP's computational efficiency and suitability for less complex datasets, emphasizing the importance of appropriate model selection and the potential enhancements from advanced architectures, feature engineering, and diversified evaluation metrics in predictive model development.

1 Introduction

Diabetes, a chronic metabolic disorder characterized by elevated blood glucose levels, poses a significant global health challenge [3]. Predicting the onset of diabetes is crucial for timely intervention and effective management.

Machine learning techniques have demonstrated their efficacy in various medical applications, particularly in disease prediction systems [5]. For instance, studies have successfully employed artificial intelligence tools for diagnosing complications associated with diabetes, such as Diabetic Retinopathy [14]. Additionally, machine learning facilitates continuous monitoring of symptoms and bio-markers, enabling precise treatment plans tailored to individual patients [3].

In this study, this challenge is explored by employing machine learning techniques, specifically focusing on the implementation of a Perceptron model for diabetes prediction. The dataset used for this analysis originates from the Pima Indians Diabetes Database [13], providing a valuable resource for investigating and predicting diabetes incidence. To enhance the model's performance, pre-processed data made available by the CSIE at National Taiwan University was utilised.

The study is structured to facilitate a comprehensive exploration of the model's hyper-parameters. This includes variations in learning rates, activation functions, batch sizes, and optimizers, with a focus on refining the model's performance through an iterative process [2][14].

To benchmark the effectiveness of our approach, a thorough comparative analysis was conducted with leading competitors in diabetes prediction. This involved evaluating the model developed against state-of-the-art algorithms, emphasizing metrics such as Area Under the Curve (AUC) and accuracy to provide a robust assessment of predictive accuracy [9].

The perceptron model, when coupled with ensemble learning algorithm is shown to improve increase AUC by 3% without significant changes in execution time [10]. The framework proposed by Jehangir et al. uses auto-tunable multilayer perceptron combined with outlier detection algorithm. This frame-

work has been tested on the Pima Indian diabetes dataset and have reported highest accuracy of 88.7% [6] Similarly, use of Multi-Layer Perceptron (MLP) algorithm is shown to out perform other machine learning algorithms: K-means,Fuzzy C-means, Artificial Neural Network, Convolutional Neural Networks on the Pima Indian diabetes dataset obtaining accuracy score of 86%. [12]

Through an extensive hyper-parameter tuning process, the most effective combination of parameters was identified, leading to a significantly improved predictive accuracy. The refined model demonstrates noteworthy capabilities in accurately identifying instances of diabetes within the dataset.

To further enhance the predictive power of the model, an extension beyond the traditional Perceptron architecture was pursued, leading to the implementation of a Multi-Layer Perceptron (MLP). This neural network, characterized by multiple layers of interconnected nodes, exhibits an enhanced capacity for capturing complex relationships within the data [5].

This research contributes to the ongoing efforts to develop more accurate and accessible tools for diabetes prognosis and management.

2 Methodology

2.1 Data Preparation

The Pima Indians Diabetes Database is a widely used dataset for developing predictive models, specifically for diabetes. It contains records from a study carried out on women of Pima Indian heritage aged 21 and above. Here are the features of this dataset:

1. **Pregnancies:** Number of times pregnant
2. **Glucose:** Plasma glucose concentration over 2 hours in an oral glucose tolerance test
3. **BloodPressure:** Diastolic blood pressure (mm Hg)
4. **SkinThickness:** Triceps skinfold thickness (mm)
5. **Insulin:** 2-Hour serum insulin (mu U/ml)
6. **BMI:** Body mass index $\left(\frac{\text{weight in kg}}{(\text{height in m})^2}\right)$
7. **DiabetesPedigreeFunction:** Diabetes pedigree function (a function which scores likelihood of diabetes based on family history)
8. **Age:** Age (years)
9. **Outcome:** Class variable, 0 denotes the absence of diabetes, and 1 denotes the presence of diabetes.

Scaling is crucial in machine learning modeling, especially for datasets containing features with different scales. The features "Glucose," "Blood Pressure," "SkinThickness," "Insulin," and "BMI" have different units and scales. When features are on different scales, the gradient descent in the perceptron models may take longer to converge and significantly slow down training, and features with larger scales can dominate the cost function causing the model to mostly learn from those features and possibly ignore the smaller scale features. Scaling also helps reduce numerical instability by ensuring excessive large or small values do not occur during calculations.

Therefore, the dataset used is for this analysis 'diabetes_scale' It is normalized and split into features and labels, the labels in this dataset are converted to 0 and 1 instead of -1 and 1. There were 9 missing values from the Age column, which were dropped since it only accounts for 1% of the dataset and thereby does not significantly impact results. The dataset is further split into training and test datasets, with a training size of 80% and a test size of 20%.

2.2 Model Construction

Two types of models are constructed:

2.2.1 Single Layer Perceptron

A Single Layer Perceptron (SLP) is one of the simplest forms of a feedforward neural network and is used primarily for binary classification problems. [11] The structure of an SLP is comprised of a single layer

of nodes, where the inputs are directly connected to the output via a series of weights. SLP is suited to fit data where the classes are linearly separable due to its inherent linear nature and the utilization of a linear activation function.

The learning process of an SLP involves iteratively updating the weights and bias to minimize the error between the predicted and the actual output. The learning process for a perceptron can be summarized as follows[1]:

Mathematically, the output y of a perceptron is represented as:

$$y = \phi\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{1}$$

where:

- w_i represents the weights,
- x_i represents the input features,
- b is the bias,
- m is the number of features,
- and ϕ is the activation function.

The learning algorithm can be summarized as follows [1]:

1. **Initialization:** The weights are initialized with small random values or zeros.
2. **For each training example:** (x, d), where x is the input vector and d is the desired output:
   (a) Compute the actual output y:
   $$y = \phi\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{2}$$
   (b) If the prediction is incorrect, update the weights and bias:
   $$w_i = w_i + \eta(d - y)x_i \tag{3}$$
   $$b = b + \eta(d - y) \tag{4}$$
3. η is the learning rate, controlling the step size at each iteration.

The weights and biases are updated iteratively via a process known as gradient descent which uses the Empirical Risk Minimalisation (ERM) to minimize the loss function based on misclassifications. The process continues until the specified number of iterations or until convergence is achieved, resulting in the final weights used for binary classification. This process can be broken down as follows:

Assume:

$$g(\mathbf{x}; \mathbf{w}) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle) \tag{5}$$

where $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$, and $y \in \{-1, 1\}$.

1. **Input:** Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, step size η, and number of iterations T.
2. **Initialization:** $\mathbf{w}_1 = \mathbf{0}$
3. **Iterative Update:** For each iteration $t = 1$ to T:
   $$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \sum_{i=1}^{n} \left(y_i \mathbf{x}_i \mathbb{1}_{\{y_i \langle \mathbf{x}_i, \mathbf{w}_t \rangle < 0\}}\right) \tag{6}$$

Output:

$$\mathbf{w}^* = \mathbf{w}_T \tag{7}$$

The class of $\mathbf{x}$ is predicted via:

$$y^* = \text{sign}(\langle \mathbf{x}, \mathbf{w}^* \rangle) \tag{8}$$

The Empirical Risk is given by:

$$R_n(\mathbf{w}, \ell_{\text{pern}}) = \frac{1}{n} \sum_{i=1}^{n} \ell_{\text{pern}}(\mathbf{x}_i, y_i, \mathbf{w}) \tag{9}$$

with Loss:

$$\ell_{\text{pern}}(\mathbf{x}, y, \mathbf{w}) = \max\{0, -y\langle \mathbf{x}, \mathbf{w} \rangle\} \tag{10}$$

The sub-gradient of $R_n(\mathbf{w}, \ell_{\text{pern}})$ is:

$$\frac{\partial R_n(\mathbf{w}, \ell_{\text{pern}})}{\partial \mathbf{w}} = \frac{1}{n} \sum_{i=1}^{n} \left(-y_i \mathbf{x}_i \mathbb{1}_{\{y_i \langle \mathbf{x}_i, \mathbf{w}_t \rangle < 0\}}\right) \tag{11}$$

For each training example, if the model misclassifies, the weights are updated according to the learning rate η and the input features. This process continues for a specified number of iterations, or until convergence, yielding the final weights $\mathbf{w}^*$ used for classification.

2.2.2 Multilayer Perceptron

A Multilayer Perceptron (MLP) is a class of feedforward artificial neural network model. Unlike Single Layer Perceptrons, MLPs have at least one hidden layer between the input and output layers, allowing them to model more complex, non-linear relationships between inputs and outputs.[4] Each node in these layers employs a non-linear activation function, like the sigmoid or ReLU, enabling the network to learn from the error and make adjustments to the weights.

MLPs are universal function approximators and have the capability to model intricate patterns and make predictions about previously unseen data. They are suited for tasks where the relationship between the input and output variables is non-linear and complex.

The construction of an MLP model involves defining the architecture, which includes the number of hidden layers, the number of nodes in each layer, and the activation functions used in the hidden layers and the output layer. The training involves using algorithms like backpropagation in conjunction with optimization methods like gradient descent to minimize the error between predicted and actual outputs on the training data.

2.3 Activation functions

The models were trained with activation functions: Sigmoid, ReLU, and Tanh. The sigmoid activation are commonly used in binary classification problems and was used to set the base model. Relu was selected for its simplicity and efficiency. The Relu function is computationally inexpensive and helps minimise the likelihood of vanishing gradient problems. It has also been found to accelerate the convergence of SGD compared to the Sigmoid and Tanh activation functions [8]. Tanh was chosen because it maps the training data between -1 and 1, centering the data around 0, and is therefore zero-centered, making it easier for the model to learn patterns.

2.4 Optimization functions

The optimization algorithms tested for this model were Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam), and Root Mean Square Propagation (RMSprop). SGD was used for its simplicity and efficiency. Adam computes adaptive learning rates for each parameter and is especially effective when dealing with sparse gradients on noisy problems[7]. RMSprop adapts the learning rates during training and is very effective for datasets with unpredictable patterns.

2.5 Hyperparameter Tuning

:

The models are trained and validated using a combination of 5-fold cross-validation. The cross-validation method provides a robust estimate of the model's performance and generalization ability on unseen data, by averaging the results over five different splits. The models' hyperparameters including the learning rate, batch size, optimizer, and activation function were tuned using a grid-tuning approach. In the initial stage of hyperparameter tuning, coarse tuning was performed with smaller epochs to quickly identify promising regions in the hyperparameter space. This helps eliminate and narrow the search spaces making the tuning process more manageable and less computationally intensive. The hyperparameters choosen for the grid search are shown below:

- **Learning Rates:**

$$\{0.001, 0.01, 0.1, 1\}$$

- **Optimizers:**

'SGD', 'Adam', 'RMSprop'

- **Activation Functions:**

ReLU(), Sigmoid(), Tanh()

- **Batch Sizes:**

$$\{10, 30, 90\}$$

The course tuning was followed by fine-tuning with increased epochs and varying learning rates to pinpoint the best hyperparameter values. the learning rates were choosen around the best learning obtained from the coarse tuning process for both the types of models

The combination of 5-fold cross-validation with coarse and fine-tuning enables a systematic and efficient approach to hyperparameter optimization. This approach ensures that the model is not only well-tuned but also capable of generalizing well to unseen data, enhancing its reliability and robustness in practical applications.

2.6 Evaluation

The evaluation of the models focuses on two critical metrics: Accuracy and the Area Under the Receiver Operating Characteristic Curve (AUC). These metrics are integral for understanding the model's capability to correctly classify instances and its discriminative power.

2.6.1 Accuracy

Accuracy is a prevalent metric in the realm of classification models. It is represented by the formula:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{12}$$

$$= \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

In this equation:

- TP denotes True Positives.
- TN denotes True Negatives.
- FP denotes False Positives.
- FN denotes False Negatives.

2.6.2 AUC - ROC Curve

The AUC is the area under the Receiver Operating Characteristic (ROC) curve, a graphical representation of the model's diagnostic ability. It is a quantifier of the model's capability to discriminate between the positive and negative classes. The AUC ranges from 0 to 1. A value of 0.5 suggests that the model has no discrimination capability, and a value of 1 denotes a perfect model. It is calculated as:

$$\text{AUC} = \int_0^1 \text{TPR}(t) d\text{FPR}(t) \tag{14}$$

In this equation:

- t denotes the threshold.
- $\text{TPR}(t) = \frac{\text{TP}}{\text{TP+FN}}$ is the True Positive Rate.
- $\text{FPR}(t) = \frac{\text{FP}}{\text{FP+TN}}$ is the False Positive Rate.

The accuracy and the AUC are calculated for each fold during the k-fold cross-validation and subsequently averaged to attain a more robust estimate of the model's performance. Lastly, these metrics are evaluated on a separate test set to assess the model's generalization capabilities on unseen data. This dual-metric evaluation ensures a comprehensive understanding of the model's performance by combining insights into correct classification from accuracy with insights into discriminative power and robustness from the AUC.

3 Results

The results in Tab. 1 and Tab. 2 lists the best hyperparameters obtained for the SLP and MLP models during the coarse cross-validation tests. The number of iterations (epochs) was set to a value of 50 during this grid search process to speed up the tuning since grid search had 108 different combinations of hyperparameters for the model to be tested on.

As shown in Eq. (12) The SLP model had highest validation accuracy of 0.761 with a learning rate of 0.1 and Batch size of 30 and with Relu activation function and Adam optimiser. The SLP outperformed the MLP for these hypermarameters. The highest accuracy obtained for MLP was 0.728 with the Tanh activation function, Adam optimiser, learning rate of 0.01 and batch size of 90.

Table 1: Best Performing Models - Sorted by SLP validation Accuracy

Act.	Opt.	L.R.	B. S	SLP	MLP
ReLU()	Adam	0.1	30	0.761	0.644
ReLU()	RMSp	0.1	10	0.756	0.644
ReLU()	RMSp	0.1	30	0.755	0.644
ReLU()	RMSp	0.1	90	0.755	0.644
ReLU()	Adam	1.0	90	0.748	0.644

Table 2: Best Performing Models - Sorted by MLP validation Accuracy

Act˙	Opt.	L. R.	B. S.	SLP	MLP
Tanh()	Adam	0.1	90	0.649	0.728
Tanh()	SGD	1.0	90	0.713	0.712
Tanh()	RMSp	0.01	90	0.722	0.680
Tanh()	RMSp	0.01	10	0.689	0.661
Tanh()	Adam	0.001	30	0.482	0.659

Next, using the best hyperparameters for each model. The models were fine-tuned at higher epoch of 200 to obtained best learning rate.

Both models obtained highest accuracy for learning rate of 0.1. The loss function and accuracy plots obtained during final stage of tuning SLP is shown in Fig. 1 and Fig. 2 respectively.

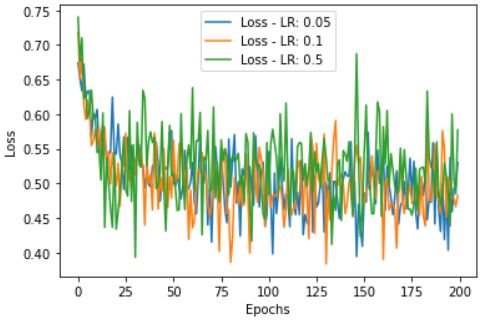

Figure 1: Loss of SLP

Finally, the best hyperparameters obtained from

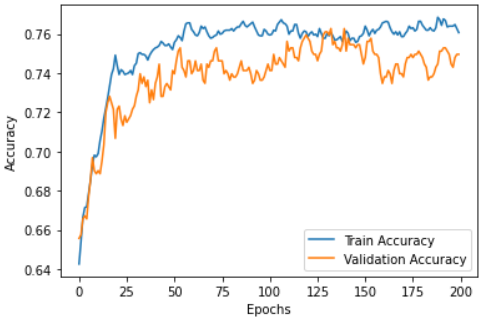

Figure 2: Training and validation accuracies of SLP

fine cross-validation test is used to train model with entire train dataset and this model is evaluated against the test dataset. The SLP model achieved a training accuracy of 76.11% and a test accuracy of 82.24%, with an AUC of 0.8515. In contrast, the MLP model achieved a training accuracy of 74.63%, a test accuracy of 78.95%, and an AUC of 0.8490.

4 Discusion

The SLP model with Relu activation function, learning rate of 0.1, batch size of 30, and Adam optimizer demonstrated superior performance over the MLP model. It had higher test accuracy and a slightly better AUC score compared to the MLP model, which was optimized using the Adam optimizer. The results indicate that for this specific dataset, a simpler model like SLP can outperform a more complex model like MLP.

The perceptron model presented a more computationally efficient alternative, converging faster and requiring fewer resources compared to the MLP. Given the perceptron's performance and efficiency on this dataset, it was deemed unnecessary to pursue the optimization of a more complex and resource-intensive model like the MLP.

Moreover, MLPs, with their added complexity and flexibility, have a higher risk of overfitting, especially on datasets with simpler relationships and lower dimensionality. Overfitting results in models that perform well on the training data but have poor gener-

alization capabilities on unseen data.

Due to the aforementioned reasons and after weighing the trade-offs between model complexity, computational efficiency, and generalization capability, the decision was made to optimize the perceptron model for this dataset. The perceptron's simplicity, faster convergence, and satisfactory performance made it a more suitable choice for our specific needs and constraints.

Fig. 1 shows that a learning rate of SLP of 0.05 causes the loss to drop slower compared to rest of the plot during the initial epochs slowing down convergence. Learning rate of 0.5 causes the loss function to converge at a higher rate but ends at a relatively higher overall loss. The learning rate of 0.1 provides the optimal setting with high convergance initially followed by saturation towards the end at a lower loss value.

The difference between training accuracy and validation accuracy as shown in Fig. 1 is very small for most of iterations of the SLP model,this indicates the SLP model accurately learned the underlying features of the dataset enabling it to accuately make classifications on unseen data. This assumption is re-validated by high performance on the test dataset by the SLP.

The loss curve and accuracy plots both validates the fact the the SLP model has been tuned well for the given range of hyperparameters.

5 Code

The code can be accessed here:
https://github.com/shavinkalu23/7318$_{a}ssignment_{1}$

6 Conclusion

This study aimed to construct and optimize Single Layer Perceptron (SLP) and Multi-Layer Perceptron (MLP) models for predicting diabetes. The models were trained, validated, and tested using the normalized diabetes dataset, and their performance was evaluated based on accuracy and AUC.

The Single Layer Perceptron model outperformed the Multi-Layer Perceptron model in terms of both accuracy and AUC. This suggests that for certain datasets, especially when they are not highly complex, simpler models like SLP could be more effective and efficient.

While both models achieved respectable accuracies and AUC, it's important to explore more sophisticated model architectures, feature engineering, and data augmentation methods for further improvements. The application of regularization can potentially lead to substantial improvements in model performance, reinforcing their robustness and reliability in varied and dynamic environments. Additionally, considering other performance metrics like precision, recall, and F1-score, especially in imbalanced datasets, would provide a more comprehensive understanding of the models' performance.

This study provides insights into the importance of selecting appropriate model architectures, hyperparameter tuning, and model evaluation techniques in building effective and efficient predictive models.

References

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. 3

[2] Ricardo Casanova, Adolfo Correa, Solomon K. Musani, and Donna K. Arnett. Prediction of incident diabetes in the jackson heart study using high-dimensional machine learning. *PLOS ONE*, 11(10):e0163942, 2016. 1

[3] Samer Ellahham. Artificial intelligence: the future for diabetes care. *The American journal of medicine*, 133(8):895–900, 2020. 1

[4] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning, volume 1, 2016. 4

[5] Mohammed Abdullah Hossain, Rumana Ferdousi, and Mohammad F. Alhamid. Knowledge-driven machine learning based framework for early-stage disease risk prediction in edge environment. *Journal of Parallel and Distributed Computing*, 146:25–34, 2020. 1, 2

[6] Maham Jahangir, Hammad Afzal, Mehreen Ahmed, Khawar Khurshid, and Raheel Nawaz. An expert system for diabetes prediction using auto tuned

multi-layer perceptron. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 722–728, 2017. 2

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 4

[9] Seungjae Lee and Hojung Song. Implementation of diabetes incidence prediction using a multilayer perceptron neural network. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2021. 1

[10] Roxana Mirshahvalad and Nastaran Asadi Zanjani. Diabetes prediction using ensemble perceptron algorithm. In *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 190–194, 2017. 1

[11] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013. 2

[12] Sivasankari S S, J. Surendiran, N. Yuvaraj, M. Ramkumar, C.N. Ravi, and R.G Vidhya. Classification of diabetes using multilayer perceptron. In *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, pages 1–5, 2022. 2

[13] UCI Machine Learning Repository. Pima indians diabetes database, n.d. 1

[14] Emanuele Vocaturo and Eugenio Zumpano. The contribution of ai in the detection of the diabetic retinopathy. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1516–1519. IEEE, 2020. 1