Using a Perceptron Based Classifier to Predict Diabetes in Pima Indian Women

Brian Drake
University of Adelaide
THE UNIVERSITY OF ADELAIDE 5005 AUSTRALIA
brian.drake@student.adelaide.edu.au

Abstract

This paper explores the use of perceptron based machine learning models to predict the onset of Type 2 diabetes in individuals using the Pima Indians Diabetes Database. Three models were implemented: a single-layer perception (SLP), a multi-layer perceptron (MLP) and an optimised MLP with batch normalisation and dropout layers for improved generalisation. The models were trained and evaluated on the dataset, with results compared through accuracy and confusion matrix metrics. The optimised MLP demonstrated the highest predictive performance, achieving an accuracy of 82.96% and a false positive rate of 7.41%. Highlighting the potential of neural network based models for effective diabetes risk prediction and providing insights into model optimisation techniques, such as regularisation and normalisation.

1. Introduction

Diabetes Mellitus, commonly known as diabetes, is a condition that is symptomatic of uncontrolled blood glucose levels in the body. The two most prevalent types are Type 1 and Type 2. Type 1 is primarily a genetic condition where the pancreas produces little to no insulin, a hormone pivotal for blood glucose regulation. By contrast, Type 2 diabetes, while still influenced by genetic factors, is mostly associated with lifestyle choices [6].

To explore Type 2 diabetes, the Pima Indians Diabetes Database was used. This dataset was provided by the National Institute of Diabetes and Digestive and Kidney Diseases, under a Creative Commons 0 license. It contains information gathered from 768 female Pima Indian patients over the age of 21. The dataset is unbalanced in that 258 individuals were evaluated as positive for Type 2 diabetes, while the remaining 500 were negative for diabetes. It includes eight features: number of pregnancies, OGTT (Oral Glucose Tolerance Test), diastolic blood pressure, triceps skin fold thickness, blood insulin concentration, BMI (Body Mass Index), age, and pedigree diabetes function (a numeric value representing genetic factor). Alongside the target variable, Outcome, representing the binary classification of a patient's diabetes status.

This paper develops a predictive model using a perceptron-based supervised learning algorithm. The perceptron, introduced by F. Rosenblatt in 1958 [5] in "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," is one of the foundational machine learning algorithms. Briefly, the perceptron calculates a weighted sum of input features and their corresponding weights, which is then passed through an activation function to generate an output.

In modern machine learning, this structure is now used as an individual node in a multi-layered network of perceptron nodes, with weights updated through training epochs. Once trained, the model weights are saved and used to make predictions on new data. Allowing for this paper to not only explore the dataset but also simultaneously explore perceptron architecture by developing a predictive model using a single, multi, then an optimised multi layer perceptron.

2. Method

As explained prior, a perceptron takes an input, assigns initial weights, updates these weights based on data analysis, and produces either a classification or regression through an activation function. The perceptron is therefore attempting to find linear separations between classes. The computational aspect of this process is handled by PyTorch's Linear class. According to the PyTorch documentation [3], this operation is mathematically represented as:

$$y = xA^T + b$$

Where A represents the weights which are multiplied by x features, with b as the bias term. This typically results in an affine linear transformation. The initial weights are uniformly distributed as $U(-\sqrt{k}, \sqrt{k})$, where:

$$k = \frac{1}{numberofinputfeatures}$$

Dependant on the problem requirements, the perceptron can perform either regression or classification of data. In predicting Type 2 diabetes for Pima Indian women, the task was framed as a binary classification. To achieve this, the sigmoid activation function [3] was selected. The sigmoid function scales the input element-wise, producing an output between 0 and 1, and is defined as:

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

For this task, an output below 0.5 is considered a negative diabetes prediction, while an output of 0.5 or above is considered a positive diabetes prediction.

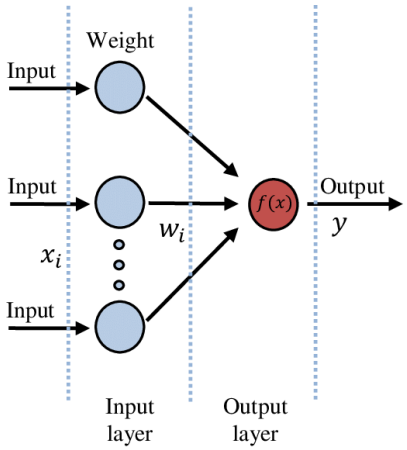

Figure 1. Graphical representation of a generic perceptron [1]

.

For this paper, figure 1 represents a single layer perceptron. Specifically, the input would be the 768x8 dataset, the f(x) function being the sigmoid activation function, and the output would be the diabetes class prediction.

2.1. Optimisers

Optimisers are pivotal components in machine learning models, responsible for adjusting the weights during training to minimise the loss function and improve the overall performance.

For proof of concept, Stochastic Gradient Descent (SGD) was initially used for the single layer and multi layer perceptron models. This optimiser updates the weights during training by calculating the gradient of the loss function with respect to each weight and then adjusting the weights in the opposite direction of the gradient [3]. The weight update rule for SGD is:

$$w = w - \eta \nabla L(w)$$

Where w represents the models' weights, η represents the learning rate and $\nabla L(w)$ is the gradient of the weighted loss function.

While SGD is computationally efficient, it can be limited by slow convergence or by getting stuck in saddle points, where the gradient is small but not optimal. Causing convergence to be stuck in a local minima and not the global minima. To overcome this potential issue, the Adam optimiser was used for the final model.

Adam (Adaptive Moment Estimation) uses adaptive learning rates for each parameter by maintaining two moving averages [3]. The first moment is the gradient and the second moment is the squared gradient. Mathematically, Adam is expressed as follows.

$$m_t = \beta_1 m_(t-1) + (1 - \beta_1) \nabla L(w)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla L(w))^2$$

$$(m_t) = \frac{m_t}{1 - \beta_1^t}, (v_t) = \frac{v_t}{1 - \beta_2^t}$$

$$w = w - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$$

Where m_t is the first moment and v_t is the second moment. β_1 and β_2 are decay rates where, β_1 is usually equal to 0.9 and β_2 is usually equal to 0.999. Finally, ϵ is a small constant added to prevent division by zero.

2.2. Loss Function

In binary classification tasks, the Binary Cross-Entropy Loss (BCELoss) function is commonly used to measure the difference between the predicted output and the actual target label [3]. This loss function in particular is suitable for the classification of diabetes Type 2 task being performed. BCELoss is defined as:

$$l(x, y) = -[y \cdot \log(x) + (1 - y) \cdot \log(1 - x_n)]$$

Where x is the predicted probability (the output formt he sigmoid function), y is the actual label (0 for negative, 1 for positive), and $\log(x)$ ensures that the function penalises confident incorrect prediction strongly.

BCELoss computes the error for each prediction and averages it over the batch. In PyTorch, the log is clamped to be greater than or equal to -100, to prevent undefined values due to extreme predictions, ensuring stability during backpropagation and keeping the loss finite.

2.3. The Multi Layer Perceptron

The multi-layer perceptron requires more than one activation function, due to the dense (fully connected) hidden layers. This function will be a ReLu (a rectified linear unit function). ReLu is an element wise function that maps the output to the maximum of either 0 or the output itself [3].

The purpose of this activation function is to enable a non-linear output between layers, assisting in learning complex data patterns. The aim of which is to develop better performing model than the single layer perceptron. The ReLu function is mathematically represented as:

$$ReLu(x) = max(0, x)$$

2.4. Additional Techniques For Optimisation

Building on these concepts and mathematical ideas, an optimised model using the multi-layer perceptron architecture can be developed to enhance predictive performance. This optimised model would contain two additional techniques: batch normalization and dropout.

Batch normalisation is a function whereby inter-layer outputs of a neural network are transformed to maintain a mean of approximately 0 and a standard deviation of approximately 1 [3]. This enables faster resolution of epochs, enabling more efficient training through normalisation. Mathematically, this looks like:

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta$$

Where the mean and standard deviation are calculated per-dimension, with and being learnable parameter vectors of size C. C equal to the number of features or channels of the input.

Dropout is where during model training some elements of the input tensor are randomly changed to zeroes. Achieved as a probability that each element will turn into a zero [3]. For this experiment, the dropout was set to a 50% probability. This methodology is effective in regularisation of data and preventing the co-adaptation of neurons [2]. Meaning that the increased complexity of the model will be less likely to create redundant or silent nodes. In turn allowing for greater generalisation on new data.

2.5. Model Assessment

The models developed in this paper were assessed on two key metrics: accuracy and false negative rate.

Accuracy allows a rapid insight into the proportion of correct predictions the model makes, however, is limited regarding characterising the performance of the model. Accuracy was measured as the percentage of correct predictions against the total number of predictions.

A confusion matrix is a tool used to characterise the predictive behaviour of a model. Achieved by categorising predictions against either a true or false outcome. Where false positive, true positive, true negative, or false negative are the outcome categories. False negative is the defining attribute for this experiment as it places emphasis on reducing potential harm to patients in a clinical setting.

Therefore, using these evaluation metrics in tandem will enable a justified assessment of the model's performance.

3. Experimental Analysis

The experimental analysis was conducted using Python, along with the Pima Indian Diabetes Dataset provided. A boxplot was used to visualise features and identify potential outliers. Although the serum insulin and plasma glucose concentration features were visualised as having outliers, no outliers were removed. Owing to Type 2 diabetes affecting both blood glucose and insulin levels, leading to high variability in these features. For all other features, data points were dropped if they were 1.5 times the interquartile range (IQR) above the upper quartile or below the lower quartile minus 1.5 times the IQR. The features after having outliers removed were then visualised in a boxplot, as shown in Figure 2.

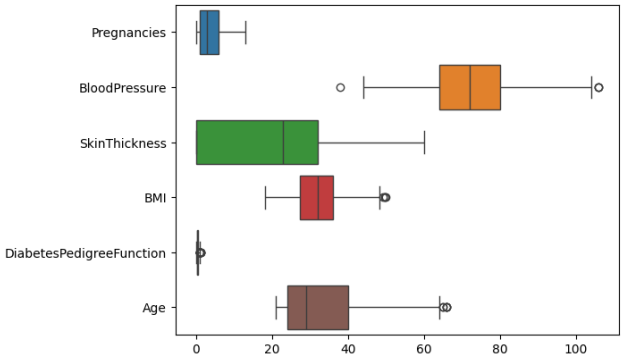

Figure 2. Boxplot displaying data categories that had outliers removed, displaying that some had low impact outliers remaining.

The target labels were already in a [0, 1] binary format, so they remained unchanged. All of the features were then scaled using the sklearn.preprocessing StandardScaler function. This function standardises features by removing the mean and scaling feature values to their unit variance [4]. This enables all of the features to be comparable to each other. Without scaling the data, data ranges with higher numerical values will be disproportionately weighted against features with lower numerical ranges. This can lead to poor convergence and suboptimal learning. After Scaling, the data was then randomly split into an 80:20 training:testing proportion, using sklearn's train_test_split function. The purpose was so that later or earlier recorded data is not more representative than the rest as well as seeding the split for posterity. Following this, the data was converted into float32 tensors to ensure compatibility with the PyTorch models.

Helper train_model, evaluate_model, and model_confusion_matrix functions were made. The train_model function takes the proposed model specified

and runs a forward training calculation for a default 100 epochs using the training data subset. An epoch being one complete pass through the entire training dataset by a machine learning model during the training process. The quantity of 100 epochs was chosen as a limiting factor to prevent overfitting the model to the training data, given its size. As well as providing a standard point of comparison for all models developed. The evaluate_model function takes the testing dataset and checks whether the prediction matches the label and prints an accuracy percentage to the console. The model_confusion_matrix plots a confusion matrix by executing the classifier against the dataset and categorising its outcome. The plot was made using seaborn and the confusion matrix function itself from sklearn. These helper functions were designed to be modular so that they could be used for all models developed.

Three total models were developed as enumerated below. Table 1 represents all results collated for ease of reading.

Model	Accuracy	False Negative
Single Layer	66.67%	15.56%
Multi Layer	69.93%	30.37%
Optimised Multi Layer	82.96%	7.41%

Table 1. Collated accuracy and False negative proportion of each model.

A single layer perceptron (SLP) was developed first, using nn.Linear module within PyTorch. The SLP was then trained and evaluated using the binary cross entropy loss and SGD optimiser resulting in an accuracy of 66.67%.

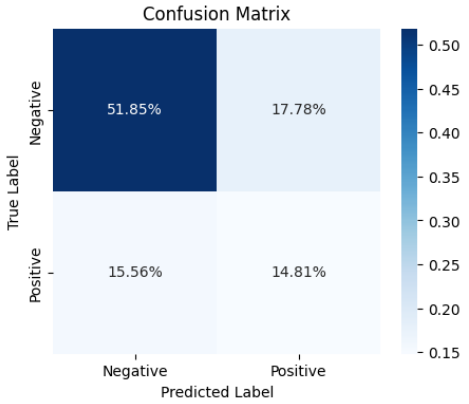

Figure 3. Confusion Matrix plot for the SLP, displaying a limited prediction capacity

The prediction accuracy is adequately characterised by the confusion matrix in figure 3, displaying that the model is predicting both positive and negative outcomes. Despite being able to guess either outcome the model is more often than not identifying a negative outcome. This is reflective of the imbalanced dataset and displays that some some learning is occurring. Therefore this model displays how shallow architecture can lead to limited performance.

A multi-layer perceptron (MLP) was then developed to mitigate the shallow architecture problem highlighted. The resultant MLP architecture included one hidden layer with 32 fully connected perceptron nodes. This model for comparability is also using the same optimiser and loss function as the SLP. Despite adding a large amount of complexity to the model, the MLP resulted in a small accuracy gain of 3.26%, for a total accuracy of 69.93%.

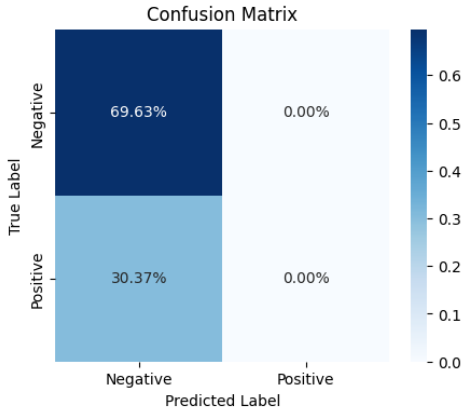

Figure 4. Confusion Matrix plot for the MLP, displaying a similar prediction behaviour as the SLP.

Despite the increase in accuracy, the MLP displays a worse predictive behaviour than the SLP, displayed in figure 4. Whereby the model is only making negative predictions. As such, despite the increased complexity and accuracy, the MLP performs worse than the SLP overall.

Considering the MLP's performance, an optimised multi layer perceptron was then developed. The optimised model introduced several additional layers and improvements over the MLP's architecture. Inclusive of increased hidden layer depth from 32 to 256 nodes, batch normalisation, 50% dropout, and an Adam optimiser. The increased complexity in tandem with the dropout designed to increase generalisation and predictive behaviour, while batch normalisation implemented to reduce computational load. Enabling Adam to detect the global minima in a faster and more accurate manner than SGD, increasing the impact of the set 100 epochs. Retaining the BCELoss function, the resultant optimised model displayed an 82.96% accuracy. This is a 16.29% accuracy improvement in comparison to the SLP and a 13.03% accuracy improvement against the MLP.

The optimised model also displays a substantial improvement in predictive behaviour. As seen in figure 5, the model displays a 7.41% false positive prediction rate and contains a performance reflective of its 82.96% accuracy. Characterised by the model's ability to guess both

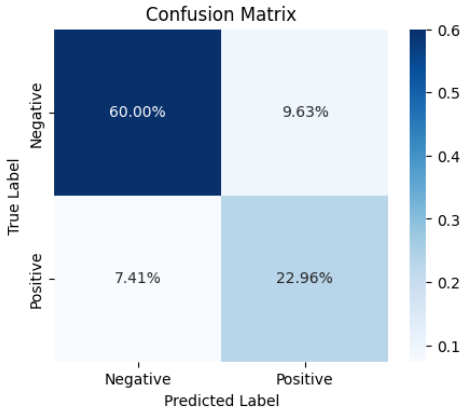

Figure 5. Confusion Matrix plot for the optimised MLP, displaying a superior prediction behaviour.

categories accurately without reflecting the imbalance that exists within the dataset, indicating reduced overfitting.

4. Code

Please find the code here.

5. Conclusion

Through the process of developing and evaluating various perceptron based models, several insights can be made about model architecture, optimisation, and regularisation.

The single layer perceptron (SLP) demonstrated the limitations of shallow models. Despite the elegance in its simplicity, the model achieved a modest accuracy of 66.67%. This accuracy, however, is hindered by the predictive behaviour of the model, guessing negative in 67.41% of all cases. This is in similar proportion to the unbalanced dataset itself. Displaying a limited capacity to meaningfully predict outcomes.

The later transition to a multi layer perceptron yielded slightly improved accuracy. An overall accuracy of 69.93%. The improvement marred by the predictive behaviour of this model being worse than the SLP; guessing negative in all cases. Thus, highlighting the importance of selecting appropriate optimisers and regularisation techniques. Where adding hidden layers without considering issues such as overfitting or slow convergence can result in under performance.

As such, an optimised MLP model was developed by increasing the hidden layer depth, applying batch normalisation and dropout, and changing to the Adam optimiser. Implementing such, improved the model's accuracy to 82.96%. Creating a 13.03% improvement over the original MLP. The model also demonstrated predictive performance consistent with its accuracy while maintaining a false positive rate of 7.41%. This suggests the potential utility of the model as a tool for recommending individuals to seek further medical attention.

In future, additional factors could be explored for optimisation. These include hyperparameter tuning, PCA (Principal Component Analysis), using advanced optimisers, data augmentation, or even different architectures entirely. Hyperparameter tuning such as learning rates, further epochs, batch sizes and dropout rates could lead to enhanced performance. PCA could be performed to reduce the dimensionality of the data, increasing the ease of the perceptron model in drawing linear boundaries between dimensions. Using more advanced optimisers such as RMSProp or ADAgrad could also provide a more accurate global minima calculation. Data augmentation for a larger dataset could also allow for more granular feature characterisation. Lastly, an alternative architecture like a CNN or a LSTM could provide greater prediction accuracy than a perceptron. However, exploration of these factors was out of scope for this paper.

Overall, this experiment has reiterated the importance of model design, optimiser selection, and regularisation, while simultaneously displaying how far machine learning and artificial intelligence have come since the original perceptron.

References

[1] Hakob Grigoryan. *Machine Learning-Based Techniques for Financial Data Analysis and Forecasting Purposes*. PhD thesis, Bucharest Academy of Economic Studies, 02 2018. 2

[2] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012. 3

[3] The Linux Foundation. *PyTorch 2.4 Documentation*, 2023. Accessed: September 12, 2024. 1, 2, 3

[4] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 3

[5] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. 1

[6] Amit Sapra and Priyanka Bhandari. Diabetes, 2023. 1