COMP SCI 7318 Deep Learning Fundamentals - Assignment 1

Lalitphan Sae-teoh (a1932456)

University of Adelaide

Abstract

Diabetes is a common health problem that affects many people around the world. Early detection of diabetes is important to help manage the disease and prevent serious health issues. This assignment focuses on implementing a perceptron model, a basic machine learning technique, to predict whether a person has diabetes based on patient data such as body mass index (BMI), glucose, and insulin levels. It demonstrates how to implement and tune a simple neural network model to achieve the most efficient performance on this dataset.

1. Introduction

Diabetes is a chronic health condition affecting millions of people worldwide. It occurs when the body is unable to regulate blood sugar levels, leading to serious problems such as cardiovascular diseases, kidney damage, and nerve disorders if not managed properly. According to the World Health Organization (WHO), the number of global diabetes patients is continually rising, making early detection crucial in preventing long-term health problems.

Recently, the capabilities of machine learning have gained significant attention in the healthcare industry. It helps doctors analyze complex medical data by identifying patterns and assisting in disease diagnosis with impressive accuracy. One such method is the perceptron model, a simple form of artificial neural network, which is particularly useful for binary classification tasks like diagnosing diabetes.

This assignment focuses on implementing a perceptron model from scratch to classify patients as diabetic or non-diabetic based on key health indicators such as body mass index (BMI), glucose levels, and insulin levels. The objective is to assess the effectiveness of the perceptron model in predicting diabetes and to optimize its performance through model tuning.

Feature Name	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
Blood Pressure	Diastolic blood pressure (mm Hg)
Skin Thickness	Triceps skin fold thickness (mm)
Insulin	2-hour serum insulin (muU/ml)
BMI	Body mass index (weight(kg)/$(height(m))^2$)
Diabetes Pedigree Function	Diabetes pedigree function
Age	Age (years)

Table 1. Pima Indians Diabetes Database - Features

1.1. Dataset

The dataset for model training is Pima Indians Diabetes Database, which is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. There are 768 patients in this dataset, all of them are females at least 21 years old of Pima Indian heritage. Diabetes patients are classified as 1 while those who are non-diabetic are classified as 0 in the "Outcome" column. There are 8 features, please see Tab. 1 for descerption.

1.2. Perceptron Model

The perceptron model, introduced by Frank Rosenblatt in 1958, is one of the earliest types of artificial neural networks designed for binary classification tasks. It is a linear classification that makes predictions based on a linear combination of input features. The model works by assigning weights to input features and calculating the weighted sum of the inputs, which is then passed through an activation function to produce a binary output. If the weighted sum is greater than a certain threshold, the model classifies the input as belonging to one class; otherwise, it classifies it as belonging to the other class [4]

The perceptron model is beneficial when data is a

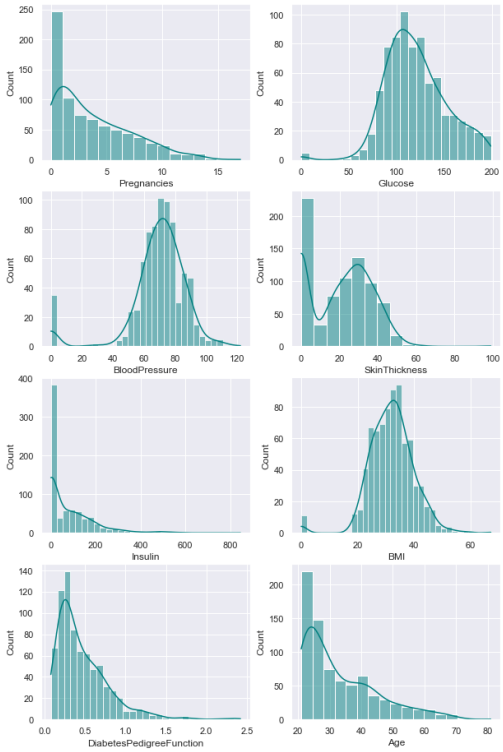

Figure 1. Features Distribution

straight line (or hyperplane in higher dimensions). It can effectively split the classes for linearly separable data. The XOR challenge illustrated how the perceptron could not handle input that was not linearly separable. [3]

2. Model Implementation

This section will include steps for building the perceptron model from scratch as a base model: data preprocessing, model training, and evaluation process. The implementation code can be found as follows source-code.

2.1. Data Preprocess

The first step in building a machine learning model is to preprocess the data to ensure it is suitable and proper for training. Also, it improves the quality and consistency of the dataset, which is crucial for better model performance. However, it is essential to explore data before preprocessing data.

From a total of 768 patients in this dataset, there are 35 % of diabetes patients and 65 % of non-diabetes patients. Missing data is not found in this dataset but some features might be handled with zero, for example almost half of the patients have zero insulin levels which should not be true. Most features are not in normal distribution form and they have different scales. Fig. 1

These are the processes to manipulate this dataset for model training.

2.1.1 Handling Missing Values:

As the missing values are treated as zero in this dataset, especially insulin level

- Skin Thickness - replace 0 with the average skin thickness from the group of age
- Insulin level - replace 0 with the average insulin level from the group of glucose level

2.1.2 Feature Scaling

The input features were scaled using standard normalization. This manipulation is significant because the perceptron algorithm is sensitive to the magnitude of feature values. The formula used for feature scaling is:

$$x' = \frac{x - \mu}{\sigma}$$

- x' is the normalized feature
- x is the original feature
- μ is the mean of the feature
- σ is the standard deviation of the feature

2.1.3 Data Splitting

The dataset was split into 80% for training the model and 20% for testing sets to evaluate the model performance.

2.2. Model Training

After preprocessing the data, the perceptron model was implemented from scratch. The goal was to train the model to classify patients as diabetic or non-diabetic based on the given features.

2.2.1 Model Initialization

The key components to build perceptron can be found in Tab. 2. Begin by initializing the weights $w_1, w_2, ..., w_n$ and the bias b set to zero. The learning rate η and epoch are set to 0.1 and 10 respectively. For the activation function, the step function (or threshold function) is used which outputs either 1 or 0.

$$\hat{y} = \begin{cases} 1 & x > 1 \\ 0 & x \leq 0 \end{cases}$$

2.2.2 Training Process

The training process involved iterating through the dataset and adjusting the weights w based on the prediction error e

Components	Description
Input Features x_i	data points or features
Weights w_i	weights determine the importance of each feature in making the prediction
Bias b	additional input that helps model makes better predictions by shifting the decision boundary
Weighted Sum z	linear combination computes from the input features $z = w_1x_1 + w_2x_2 + w_nx_n + b$
Activation Function	determine the output by passing weighted sum z
Learning Rate η	hyperparameter to determines the size of the steps the model takes to minimize loss function
Epoch	the number to complete cycle through the entire training dataset

Table 2. Perceptron Components

between the predicted output $\hat{y}$ and the actual label y. The weight update rule was applied at each step:

$$w_i \leftarrow w_i + \eta * (y - \hat{y}) * x_i$$

, where the variables definition can be found in Tab. 2. The bias was also updated similarly. The perceptron model training steps are taken as follows:

1. Compute the weighted sum for each input
2. Apply the activation function to get the predicted output
3. Calculate the prediction error

$$e = y - \hat{y}$$

4. Update the weights and bias based on the error
5. Repeat for a set number of epochs

2.3. Evaluation

Once the model was trained and learned the relationship between the input features and the class labels in the training data, its performance was evaluated using the test set.

2.3.1 Accuracy

The accuracy of the perceptron model was calculated as the ratio of correctly predicted samples to the total number of samples in the test set:

$$Accuracy = \frac{Numberofcorrectpredictions}{Totalnumberofpredictions}$$

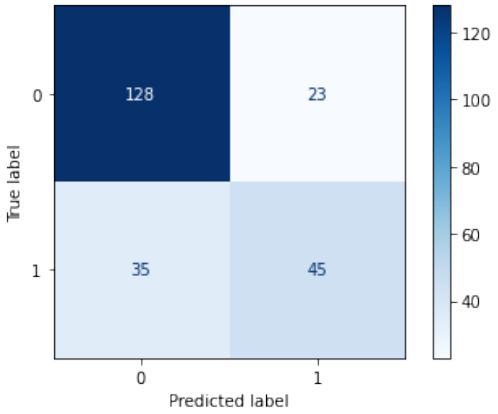

Figure 2. Base Model Confusion Matrix

2.3.2 Confusion Matrix

This matrix provides a detailed breakdown of true positives (TP), true negatives(TN), false positives(FP), and false negatives(FN), helping to assess how well the model distinguishes between diabetic and non-diabetic patients.

2.3.3 Precision, Recall, and F1-Score

Additional metrics such as precision, recall, and the F1-score were used to evaluate the model's effectiveness in handling imbalanced classes, where one class, for example non-diabetic, might have more instances than the other. These metrics are calculated as:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

2.3.4 Base Model Results

The result of the base model can be found in Fig. 2 and Tab. 3. Overall accuracy is 74.89%, the recall is 56% which is quite risky to use this model prediction for medical data. The precision, recall, and F1-score will be focused on class diabetes (class = 1) because it is more important for performance tuning in this prediction.

3. Performance Tuning

There are several strategies to apply for increasing the accuracy of the basic perceptron model. Here are the approaches to improve the model performance.

Class	Precision	Recall	F1-Score	Accuracy
Diabetic	66%	56%	61%	
Non-Diabetic	79%	85%	82%	
All				74.89%

Table 3. Base Model Results

Class	Precision	Recall	F1-Score	Accuracy
Diabetic	66%	66%	66%	
Non-Diabetic	82%	82%	82%	
All				76.62%

Table 4. Weight Initial Optimization Model Results

Number of Epochs	Accuracy
10	74.89%
50	78.79%
100	75.32%
150	75.76%
200	73.59%
300	77.92%
500	74.89%
800	76.62%
1000	74.03%

Table 5. Epoch Tuning Results

3.1. Weight Initialization

Weight initialization plays an important role in optimizing the performance of machine learning models, especially neural networks. Proper initialization can speed up convergence and help improve the model performance. If initialized weights are poor, the learning can be slow or ineffective. [2]

In this experiment, the initial weights were adjusted to $randomnumber * 0.01$, where a random number is random floats sampled from a univariate Gaussian distribution of mean 0 and variance 1. The result of weight initial optimization is shown in Tab. 4. Compared to the base model, it has a better prediction on diabetes class, the recall is increased by 10% from 56% to 66%.

3.2. Hyperparameter Tuning

The learning rate and the number of epochs can significantly affect model performance.

3.2.1 Learning Rate

The learning rate is a hyperparameter in machine learning models that controls how much the model's weights are adjusted concerning the loss gradient during each update. Proper tuning the learning rate can improve accuracy because a high learning rate can make the model converge faster but risks overshooting the optimal point which might result in poor performance. A low learning rate allows more precise convergence but may lead to slower training time and may get stuck in local minima. [1]

After applying several learning rates (example: 100,10,1,0.01,0.001) to the base model without changing epoch number (epoch =10), the accuracy result, which is 74.98%, is similar to the 0.01 learning rate. There is no effect from the experiment of changing only the learning rate parameter for this dataset.

3.2.2 Epoch

An epoch refers to one complete cycle through the entire training dataset. In practice, multiple epochs are required to adequately train a model because the algorithm often needs to see the data many times to learn the underlying patterns. Tuning the number of epochs helps ensure the model has sufficient time to understand patterns in the data without overfitting. However, too few epochs may result in underfitting and a massive number of epochs can lead to overfitting. [1]

After applying several epochs to the base model without changing the learning rate (learning rate = 0.1), the accuracy rate of the model is changed. The best performance for this experiment is 50 epochs, where accuracy is increased by almost 4% from 74.89% the base model to 78.79%. Also, the recall is increased by 6% from 56% to 62%. The experiment results are shown at Tab. 5.

3.3. Regularization

Regularization techniques can help reduce overfitting and improve generalization such as L2 (Ridge) or L1 (Lasso) regularization. They help to penalize large weights and encourage simpler models. L2 regularization is added to the weight update rule in this experiment. The L2 penalty term adds the sum of the squared weights to the loss function, encouraging the model to keep weights small and improving its generalization ability. This helps ensure that the model doesn't rely too heavily on any single feature. [1]

From the experiment, Tab. 6 the accuracy is similar to the base model, which is 74.89%, but there is a slight improvement in the recall of diabetic class by 5% from 56% to 61%. The several learning rates are applied after regularization, there is no effect on the performance. In contrast, changing epochs number affected the model's accuracy, the results can be found in Tab. 7.

Class	Precision	Recall	F1-Score	Accuracy
Diabetic	64%	61%	63%	
Non-Diabetic	80%	82%	81%	
All				74.89%

Table 6. Regularization Model Results

Number of Epochs	Accuracy
10	74.89%
50	74.46%
100	71.86%
150	76.19%
200	73.59%
300	75.76%
500	74.03%
800	75.76%
1000	76.19%

Table 7. Regularization and Epoch Tuning Results

Class	Precision	Recall	F1-Score	Accuracy
Diabetic	65%	64%	64%	
Non-Diabetic	81%	81%	81%	
All				75.32%

Table 8. Sigmoid Activation Function Model Results

Learning Rate	Accuracy
100	71.00%
10	73.16%
1	74.89%
0.1	75.32%
0.01	72.73%
0.001	71.00%
0.0001	70.13%

Table 9. Sigmoid and Learning Rate Tuning Results

3.4. Activation Function

The perceptron traditionally uses a step function as its activation function, which can make learning harder when updates are based on strict binary decisions (0 or 1). The step function can be replaced with more complex activation functions like sigmoid or tanh. They allow the model to learn using continuous outputs rather than just binary output. Moreover, they smooth the gradients and improve training for problems with insufficient strict binary output. [2]

This experiment will be performed by replacing sigmoid as an activation function. Sigmoid function is commonly used in binary classification problems because it outputs values to the range between 0 and 1, making it ideal for interpreting the output as probabilities.

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

After applying the sigmoid function to the base model with 0.1 learning rate and 10 epochs, the model accuracy is maintained. However, the recall of diabetic class is increased by 8% from 56% to 64% Tab. 8. The performance varies depending on learning rates but slightly changes in different numbers of epochs. Tab. 9 shows the sigmoid experiment result of 10 epochs.

4. Summary

In this assignment, a perceptron model was implemented to classify individuals as diabetic or non-diabetic based on health indicators such as BMI, glucose levels, and insulin levels. The dataset used was carefully explored and preprocessed to ensure data quality, which involved handling missing values, normalizing feature scales, and splitting the data into training and testing sets. The base model was initialized and trained using a standard perceptron algorithm, and its performance was evaluated using metrics such as accuracy, recall, and the confusion matrix. The base model achieved an accuracy of 74.89%, but had room for improvement in recall and precision of diabetic class which are 56% and 66% respectively.

To improve performance, several experiments were conducted. First, random weight initialization strategies were tested to avoid issues like poor convergence, which can improve recall of diabetic class up to 10%. The learning rate and epoch parameters were optimized, leading to an increase in accuracy to 78.79% and recall of diabetic class to 62%. Additionally, L2 regularization, was applied to prevent overfitting, and a sigmoid activation function was introduced to handle non-linearities, which refined the decision boundary.

Overall, the performance tuning significantly improved the model's classification accuracy and recall. The final experiment incorporating optimized learning rates 0.8, epochs 50, L2 regularization, and sigmoid activation achieved a notable improvement, bringing the final accuracy to 76.19%, with enhanced recall and balanced precision, 70% and 64% respectively Tab. 10 Fig. 3. Although, the overall accuracy is not the best but the recall is increased by 14% from the base model. This demonstrates the importance of tuning and regularization in the perceptron model for optimal performance in healthcare applications.

Class	Precision	Recall	F1-Score	Accuracy
Diabetic	64%	70%	67%	
Non-Diabetic	83%	79%	81%	
All				76.19%

Table 10. Final Tuning Model Results

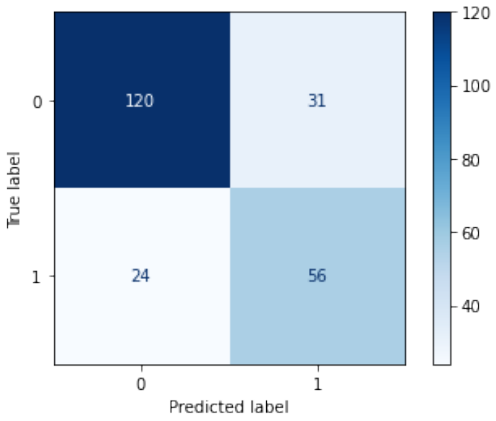

Figure 3. Final Tuning Model Confusion Matrix

References

[1] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2019. 4

[2] Aaron Courville Ian Goodfellow, Yoshua Bengio. *Deep Learning*. The MIT Press, 2016. 4, 5

[3] Seymour A. Papert Marvin Minsky. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969. 2

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