COMP SCI 7318 Deep Learning Fundamentals - Assignment 1

Lalitphan Sae-teoh (a1932456)

University of Adelaide

Abstract

Diabetes is a common health problem that is widespread in the world today. To prevent serious health issues, it is important to detect early to manage the disease. This assignment focuses on implementing the perceptron model, which is one of basic machine learning, to predict diabetes based on the pattern of patient data such as body mass index (BMI), glucose, and insulin levels. To experiment with implementing and tuning a simple neural network model to achieve the most efficient performance on this dataset.

1. Introduction

Millions of people worldwide are suffering from diabetes which is a chronic health condition. Diabetes affects the patient bodies failure to regulate blood sugar levels leading to serious problems such as cardiovascular diseases, kidney damage, and nerve disorders if not properly managed. As reported by the World Health Organization (WHO), the number of diabetes patients worldwide is constantly accelerating, meaning that early detection is essential to avoid long-term health issues.

Machine learning capabilities have drawn a lot of attention lately in the healthcare sector. It helps doctors analyze complex patterns of medical data and provides remarkably accurate disease diagnosis support. The perceptron model is one of the methods that is a simple form of the neural network, it is very helpful for binary classification tasks like diabetes prediction.

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 perform experiments on optimizing its performance through model tuning.

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 the description.

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

1.2. Perceptron Model

The perceptron model was first proposed by Frank Rosenblatt in 1958. One of the first types of artificial neural networks developed for binary classification tasks. It is a kind of linear classification where predictions rely on a linear combination of input features. The model function is started by giving weight to the input features and calculating the weighted sum of the inputs. Then generate a binary output by applying an activation function to the weighted sum. A specific threshold helps the model determine a class for the input from the total weighted sum, whether the total exceeds or falls under the threshold. [4]

The perceptron model is beneficial when data is a 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]

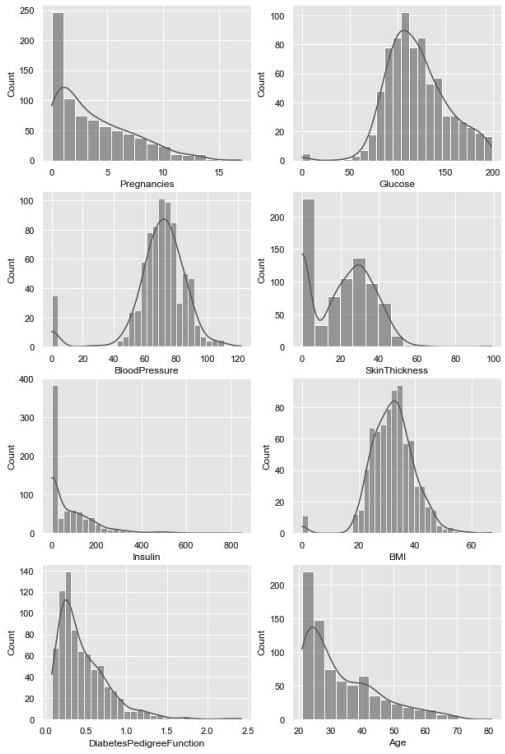

Figure 1. Features Distribution

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 developing a machine learning model is to preprocess the data to ensure it is suitable and proper for training. Also, it helps improve 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 during the data exploration 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:

The missing values are treated as zero in this dataset, especially insulin level and skin thickness features.

- Skin Thickness - Replace 0 with the average skin thickness from the age data.
- 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 between the predicted output $\hat{y}$ and the actual label y. The weight update rule was applied at each step:

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

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

, 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

The training was repeated for multiple iterations until the model converged, leading to the weights stop changing significantly. In other words, the model had learned the relationship between the input features and the class labels in the training set.

2.3. Evaluation

Once the model was trained and learned the patterns of the input features, 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:

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

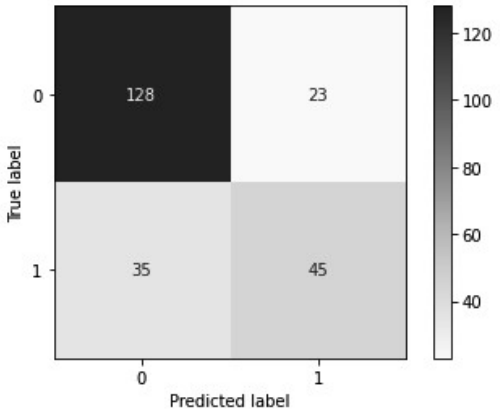

Figure 2. Base Model Confusion Matrix

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 differentiates between diabetic and non-diabetic patients.

2.3.3 Precision, Recall, and F1-Score

Additional metrics like 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. In this experiment, we aimed to improve the recall of diabetics class. 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

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 one of the hyperparameters in machine learning. The weights of the model are shifted regarding the loss of gradient at each iteration update by the learning rate. A high learning rate might help the model to converge faster. A poor result might occur when the learning is too high, it causes the risk of overshooting the ideal point. On the other hand, the possibility of falling into local minima can occur when the learning rate is too low. For that reason, the proper tuning of the learning rate can improve model performance. [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.

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

3.2.2 Epoch

The number of epochs in machine learning is a hyperparameter in which an epoch involves a complete training cycle in the entire training dataset. During model training, multiple epochs are required to sufficiently train a model to allow the algorithm to learn the underlying patterns of data as many times. The model should have sufficient time for training to allow it to understand the underlying pattern of training data. A proper number of epochs should be adjusted because too few epochs might cause underfitting while a massive number of epochs might result in overfitting. Therefore, tuning the number of epochs also helps in model performance. [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 is one method for enhancing model performance. Lasso (L1) and Ridge (L2) regularization are two examples of regularization. During the model training, they help to prevent overfitting by penalizing large weights and improving generalization which simplifies the model. In this experiment, L2 regularization is performed to the weight update rule by adding L2 penalty term which is the sum of the squared weights to the loss function. It results in the model keeping weights small and improving its generalization ability. To ensure that the model doesn't base 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,

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

changing epochs number affected the model's accuracy, the results can be found in Tab. 7.

3.4. Activation Function

Changing the activation function might help improve the model performance. A step function is applied traditionally to the perceptron as its activation function. The result of the step function is output in binary form which makes model learning more difficult. The step function in the activation function can be replaced with other complex functions such as Sigmoid or Tahn. Both example functions provide continuous outputs rather than strict binary. Also, they help smooth the gradients. [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.

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

4. Summary

In this assignment, a classification task is created by implementing a perceptron from scratch. The task is performed to predict diabetes from the Pima Indians Diabetes dataset. The implementation started with data exploration and data preprocessing to guarantee the quality of data. This included resolving missing values, standardizing features, and splitting the training data and testing data. A standard perceptron algorithm is applied to the model training. The base model was initialized and its performance was evaluated by using metrics such as accuracy, recall, and precision. The accuracy of the base model is 74.89%, while the recall and precision of the diabetic class are 56% and 66% respectively.

Many experiments were conducted to enhance model performance. First, new random weight initialization was applied to avoid issues like poor convergence, resulting in improving the recall of the diabetics class up to 10%. Second, the learning rate and epoch parameters were optimized, leading to an increase in accuracy to 78.79% and the recall of the diabetic class to 62%. Also, L2 regularization was tested to prevent overfitting, but the result doesn't show much improvement. From the test, the best regularization performance is an accuracy of 76.19% with 150 epochs. Additionally, a sigmoid activation function was introduced to handle non-linearities, which helps to refine the decision boundary. Despite, a few improvements in the overall accuracy, which is 75.32%, it can improve the recall of diabetic class to 64%.

Overall, the tests showed that the tuning can improve model performance remarkably. The final experiment included all the tuning techniques with optimized learning rates of 0.8, 50 epochs, L2 regulation, and sigmoid activa-

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

Table 10. Final Tuning Model Results

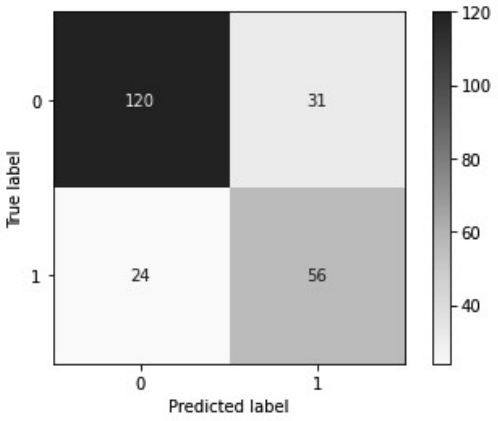

Figure 3. Final Tuning Model Confusion Matrix

tion. A significant improvement was achieved and brought the final accuracy to 76.19%, with 70% and 64% of recall and precision respectively Tab. 10 Fig. 3. Although it is not the best overall accuracy, compared to the base model the recall of diabetic class is improved by up to 14%. This demonstrates how hyperparameters and tuning techniques in the perceptron model can optimize performance in this application. However, there is room for further improvement, such as incorporating more advanced models like multi-layer perceptrons (MLPs) or handling imbalanced data by improving the prediction of diabetics class.

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. 1

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