Assignment 1: Predict Diabetes using Perceptron

Sanchit Goel
The University of Adelaide
Adelaide, South Australia, Australia
sanchit.goel@student.adelaide.edu.au

Abstract

One of the most common diseases in the world is diabetes. Even though medical tests for diagnosing diabetes are highly accurate, using Artificial Intelligence (AI) to diagnose diabetes can significantly reduce time and assist doctors thus reducing mis-classifications. This project aims at doing this by using multiple basic neural network techniques and also comparing them to find out the best way in which AI can help in predicting diabetes. Due to the dataset being small and imbalanced, the main metric of comparison was the F1 score. After modelling using cross validation to avoid overfitting, it was found that the most simple neural network actually outperforms the complex ones, though it may not be significant since the difference seen was below 1%. Overall, this project acts as a very good start in finding a suitable AI technique in solving this problem.

1. Introduction

Diabetes is a disease that occurs when the body is unable to produce enough insulin or use the insulin produced efficiently. Research shows that diabetes is often misdiagnosed in children and adults [1, 2]. Having an AI system can be helpful in assisting doctors when it comes to diagnosing diabetes. On top of that, an AI system that takes in very few parameters to detect diabetes would be even more helpful. The PIMA Indian Diabetes dataset [3] is a widely used diabetes dataset that contains various features that can be used to predict whether a patient (female of Pima Indian heritage who is at least 21 years old specifically) has diabetes or not. The authors of [4] use various machine learning algorithms along with oversampling methods such as SMOTE to predict this. The goals of this project are listed as follows:

- To create a model using the data available in the PIMA Indian Diabetes dataset, i.e., without using oversampling to create artificial data points.
- To see how a simple perceptron performs and compare it with a single-layered perceptron and artificial neural network.

2. Method Description

This section deals with the description of the method used to create various models using the PIMA dataset. Before starting with the method, the following sub-section provides information regarding the dataset and the preprocessing done before modelling.

2.1. Dataset and Preprocessing

The PIMA Indian Dataset contains information of 768 females of Pima Indian heritage that are at least 21 years old. It consists of the following 8 features: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function and Age. A description of the dataset is shown in Table 1. The target variable has 2 labels, 0 and 1 with 0 indicating that the patient does not have diabetes and 1 indicating otherwise. This makes the problem a binary classification problem.

Feature	**Description**
Pregnancies	Number of times patient got pregnant
Glucose	Plasma glucose concentration in a 2 hours oral glucose tolerance test
Blood Pressure	Diastolic blood pressure (mm Hg)
Skin Thickness	Triceps skin fold thickness (mm)
Insulin	2-Hour serum insulin (mu U/ml)
BMI	Body mass index (weight in kg/(height in m)^2)
Diabetes Pedigree Function	A scoring function based on the probability of having diabetes based on family history
Age	Age of the patient
Outcome	0 or 1 indicating diabetes

Table 1. PIMA Indian Dataset and their features

Dataset preprocessing required first converting feature names to snake_case based on the tame data con-

ventions. For example, SkinThickness was converted to skin_thickness. Following this, data analysis revealed that five features, namely glucose, blood pressure, skin_thickness, insulin and bmi had some rows with values as 0, which isn't possible in the real world. Therefore, it was best to just drop these rows. Another thing that analysis revealed was the imbalance present in the data. About 35% of the rows belonged to the positive class (1), while the rest belonged to the negative class (0). Even though a lot of researchers perform undersampling/oversampling to have more training data [**?**], they weren't used in this project to preserve the integrity of the dataset and make sure that all the data used to train the system is reliable [5].

After this, correlations between features were checked to see if any two features were highly correlated or not. If two features are highly correlated, one needs to be dropped to reduce bias (keeping in mind that the features aren't highly correlated with the target variable). Gladly, no two features were highly correlated. Glucose had the highest correlation with outcome. Once the analysis was done, the last step was to standardise all the features so that no feature gets assigned a higher or lower weight since gradient descent makes use of the feature values. Standardisation was done using eq 1.

$$\bar{x} = \frac{x - \mu}{s} \tag{1}$$

where:

$\bar{x}$ = standardised data
μ = mean of the feature
s = standard deviation of the feature

2.2. Models

Before talking about models, let's clarify how the training and validation splits were done. K-Fold cross validation was used with five as the number of splits, giving 5 different sets of training and testing data containing 80% and 20% of the data respectively. This was done to see how well does the model generalise and to look at the impact of imbalanced data. Three different variants of the perceptron were used. There are two operations that occur at a neuron. The first section of a neuron computes a pre-activation value. This pre-activation value is passed on to the second section that uses an activation function to compute the output of a neuron. In all the models, the activation function used is the sigmoid activation function which is calculated as shown in 2.

$$sigmoid(x) = \frac{1}{1 + \exp^{-x}} \tag{2}$$

When training a neural network, we first perform a forward pass that computes the output of the neural network. In this case, since the problem is a binary classification problem, we use the binary cross entropy loss which is calculated as shown in eq 3.

$$J(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \tag{3}$$

where:

y = true class label
p = predicted class probability

Once the loss has been computed, we backpropagate the loss through the network to update the weights. This is usually done using gradient descent. In this project, Adam optimiser [6] has been used instead which is an upgraded version of gradient descent and tends to perform better because it makes use of momentum to push weights in the direction they are going faster, while also decaying the learning rate based on history. In addition to this, Adam also performs bias correction so that the weights aren't biased towards zero (or whatever the initial guess was) which might result in slower updates in the beginning. Equations 4 to 8 show how the Adam optimiser works.

$$\theta_t = \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{5}$$

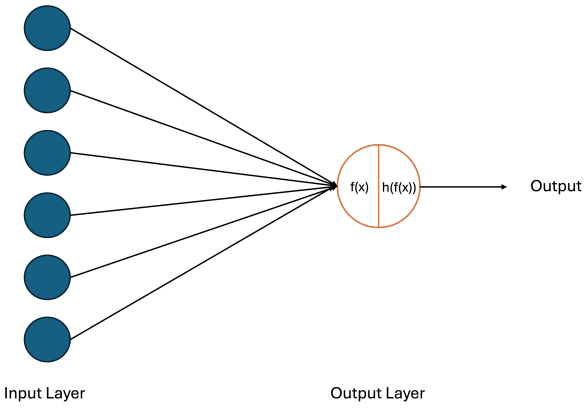
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{6}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{7}$$

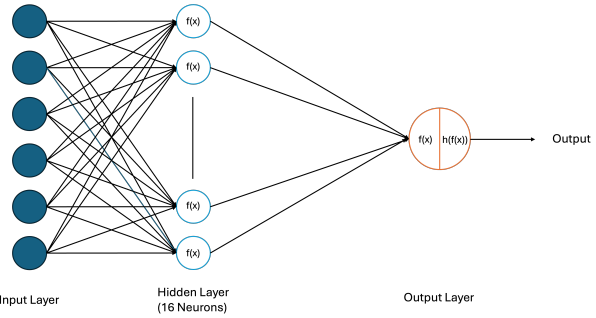
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{8}$$

where:

θ_t = parameters at time step$t$
α = learning rate
β_1, β_2 = exponential decay rates for the moment estimates
m_t = history used for momentum
v_t = history used to decay the learning rate
$\hat{m}_t$ = bias-corrected 1st moment estimate
$\hat{v}_t$ = bias-corrected 2nd moment estimate
g_t = gradient at time step$t$
ϵ = small constant to prevent division by zero



(a) A single-layered perceptron that outputs the sigmoid of the weighted average of the inputs.



(b) A single-layered perceptron with a single hidden layer. The sigmoid activation function has only been used in the output layer.

Figure 1. Architecture of models used in this project. $f(x)$ is the pre-activation value which is a weighted sum of the inputs. $h(f(x))$ computes the sigmoid of the pre-activation value. The artificial neural network architecture is not depicted here, but is similar to the architecture of the single-layered perceptron (consisting of 3 hidden layers instead).

2.2.1 Perceptron

A perceptron neuron [7] simply passes the weighted average of the inputs to an activation function in order to compute an output. It is one of the simplest types of neural networks. Since the dataset consisted of eight features, the input layer was made up of eight neurons that are used to compute the pre-activation value. This pre-activation value is fed into the sigmoid activation function to output a single value as the output of the network. The architecture of this model is shown in figure 1a

2.2.2 Single-layered Perceptron (SLP)

A Single-Layered Perceptron (SLP) is an architecture of a level higher than the perceptron. The difference between an SLP and a perceptron is that an SLP consists of a single hidden layer that contains multiple perceptron neurons. The input layer consists of eight neurons followed by a hidden layer of sixteen neurons. This hidden layer is followed by the output layer consisting of a single neuron. Note that the sigmoid activation function was only used in the output layer. The hidden layer simply passed on the pre-activation value that was computed using the weights and the inputs. This was done to reduce the complexity. Figure 1b shows the architecture of this model.

2.2.3 Artificial Neural Network (ANN)

An Artificial Neural Network (ANN) is an architecture of a level higher than the single-layered perceptron. The difference between an SLP and an ANN is that an ANN consists of multiple hidden layers. It is also called a feedforward neural network. In the architecture of the ANN used, the input layer consists of eight neurons followed by three hidden layers. The first hidden layer consists of sixteen neurons. The second hidden layer consists of eight neurons and the third hidden layer consists of four neurons. The third and last hidden layer is followed by an output layer consisting of a single neuron. Like the single-layered perceptron, the sigmoid activation function was only used in the output layer. The hidden layers simply passed on the pre-activation value that was computed using the weights and the inputs. Figure 1b shows the architecture of this model.

3. Experiments and Results

3.1. Experiments

The main goal of this project was to use a perceptron to predict whether a female patient has diabetes or not. Therefore, the main point of analysis was to compare the perceptron with the SLP and ANN. Another experiment done was related to the way the data was split. K Fold cross validation shuffles the dataset and splits it, so the ratio of classes won't be same in the training and the testing sets. Results for all the models using K Fold cross validation were compared with the results using the stratify method for data splitting. The stratify method splits the dataset in a way such that the ratio of classes is the same in both the training and testing splits. This analysis is important considering that the dataset was imbalanced. Note that in all the experiments, the models were trained for 100 epochs.

3.2. Results

To compare the models and experiments, the main metric would be the F1 Score owing to an imbalance present in the dataset. Accuracy, precision, recall and AUROC

scores were also calculated and are presented for comparison. Note that these metrics make use of the confusion matrix which consists of True Negatives (TN), False Positive (FP), False Negatives (FN) and True Positives (TP). Table 2 shows the scores for all the experiments.

Model	Stratify	Accuracy	Precision	Recall	F1 Score	AUROC
PER	False	77.06	64.42	70.80	67.04	85.51
SLP	False	78.33	71.24	59.05	64.37	85.11
ANN	False	78.33	71.24	59.05	64.37	85.15
PER	True	75.51	62.49	67.69	64.57	85.28
SLP	True	77.80	77.20	56.15	62.15	84.95
ANN	True	77.80	77.20	56.15	62.15	85.00

Table 2. Scores for the Perceptron (PER), Single-Layered Perceptron (SLP) and Artificial Neural Network (ANN) for different stratify experiments. The scores presented are the mean scores for 5 folds.

From the results, it is evident that the perceptron trained on unstratified data splits is the best model based on the F1 score. The F1 scores of all the other experiments are lower by 3% to 5%. Even the recall is better for the first experiment which means that the model is quite good at predicting the positive class which was already less in the dataset compared to the negative class. Even though the precision for that experiment is less than some of the other experiments, it is not a metric as important as recall over here since we have few positive instances and want our model to be good at predicting those. Additionally, AUROC is also an important metric here which measures a model's ability to discriminate between the classes at different threshold levels. Even though the AUROC score for the first experiment is higher than the rest, the difference between these scores between all the experiments is negligible. Therefore, it can be said that increasing the model complexity is not the right solution to this problem.

4. Conclusion and Future Work

This project aims to make use of basic neural network techniques, namely perceptrons to try and solve the problem of diabetes detection based on eight features derived from medical tests. The dataset used focuses on doing the said prediction for female patients of PIMA Indian heritage. Using various data visualisation techniques and some data preprocessing techniques, missing or erroneous values were removed, data was standardised and the necessary features were selected, i.e., all eight of them. Once this was done, three different neural networks were trained, all having difficult complexity. This was done to compare their performances. Cross Validation techniques were used to make sure that no model overfits. The simple single perceptron neuron performed slightly better than the models with more complexity in terms of the main metric which was the F1 Score. It also had the best AUROC score.

This project was a good introduction to neural networks and their power to capture patterns in datasets that are not even large. Future work can explore the use of Machine Learning in solving this problem because in most cases that involve tabulated data, traditional machine learning algorithms outperform neural networks. Additionally, focus needs to be put on a larger dataset that contains diverse information that can benefit people from all kinds of backgrounds and cultures.

References

[1] The Lancet Regional Health – Europe. Misdiagnosis of type 1 and type 2 diabetes in adults. *The Lancet regional health. Europe*, 29:100661–100661, 2023. 1

[2] Souad Larabi-Marie-Sainte, Linah Aburahmah, Rana Almohaini, and Tanzila Saba. Current techniques for diabetes prediction: Review and case study. *Applied sciences*, 9(21):4604–, 2019. 1

[3] Jack W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings - Symposium on Computer Application in Medical Care*, pages 261–265, 1988. 1

[4] Isfafuzzaman Tasin, Tansin Ullah Nabil, Sanjida Islam, and Riasat Khan. Diabetes prediction using machine learning and explainable ai techniques. *Healthcare technology letters*, 10(1-2):1–10, 2023. 1

[5] Sasidhar Duggineni. Data integrity and risk. *Open journal of optimization*, 12(2):25–33, 2023. 2

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv (Cornell University)*, 2017. 2

[7] F Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958. 3