Prediction of diabetes in female patients using perceptron model

Caquilpan Victor
The University of Adelaide
North Terrace, Adelaide, 5045
victor.caquilpanparra@student.adelaide.edu.au

Abstract

This study considers the implementation of a perceptron algorithm for predicting whether or not a patient has diabetes,based on previous diagnostic measurements. Data considers information of female people at least 21 years old of Pima Indian Heritage. A general machine learning process was done to train a model and do predictions in testing data. Final results allowed to achieve an accuracy of around 69%. An extensive analysis was done showing the main advantages and drawbacks of this algorithm.

1. Introduction

Diabetes is one of the most important diseases in the world,affecting a huge part of the population, being one condition which can show up in different stages of life, however, different studies point out that a timely identification of diabetes is possible by tracking specific characteristics of people [1].

On the other hand, artificial neural networks are a branch of artificial intelligence, which has been broadly developed in the last decades being useful in different industries. Perceptron is a binary classifier comprising two layers of neurons, an input layer which receives signals (called inputs) and an output layer produced by an activation function, which is useful to map input values to expected output. In this way, a model can be trained with labeled trained data (supervised learning) and be useful to do prediction in unknown data [2,3].

For this study a dataset of 768 females older than 21 years old of Pima Indian heritage is used. The dataset contains information about some medical predictors variables and one target variable,which is the presence of diabetes. In Table1 the predictors are described.

Name predictor	Description	Unit
Pregnancies	Number of times pregnant	#
Glucose	Plasma glucose concentration	-
BloodPressure	Diastolic blood pressure	mm/Hg
SkinThickness	Triceps skin fold thickness	mm
Insulin	2-hour serum insulin	mu/ml
BMI	Body mass index	-
DPF	Diabetes pedigree function	-
Age	Age (years)	#

Table 1. Descriptions of predictors.

Previous studies show that features presented in Table 1 are correlated to the presence of diabetes in people [4,5], thereby these would be appropriate to train a machine learning model. The data is numeric and is previously scaled in the range of 0 to 1 to their easier implementation, while that target variable is presented in values of 1 (presence of diabetes) and 0 (absence).

2. Perceptron algorithm

A perceptron is the basis model in the field of neural networks, which has been widely used to create more complex neural networks models. The architecture of a perceptron is with zero hidden layers and it could be used mostly for solving binary classification problems (however it could be used for regression, simulating a linear regression) [6]. Figure 1 represents the architecture of a classic perceptron, which corresponds to a McCulloch-Pitt neural network (which is considered to be the first neural network) [7].

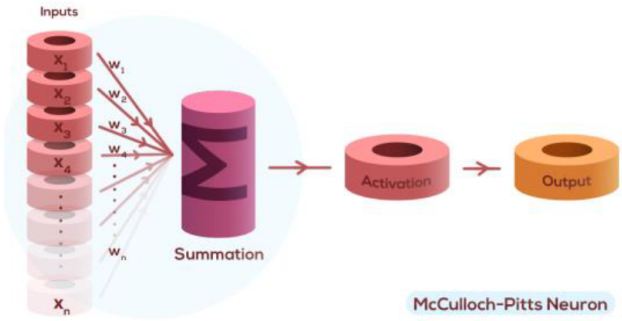

Figure 1. Perceptron architecture [6].

Having a set of features (X_1, X_2,..., X_n) is possible to estimate the value of a feature y called target, which is a binary class (as -1, 1 or on, off). So, each feature is multiplied by a certain weight value (w_1, w_2,..., w_3) and later each one of these calculations is summed and passed for an activation function. In addition to weights, a bias parameter (b) is considered. In this way, we can apply the next equation.

$$y_{out} = b + \sum_{i=1}^{n} X_i * w_i \qquad (1)$$

Where y_{out} is the output given the current weights w and bias b and n is the number of features. Then, an activation function is utilized to return an output of -1 and 1. Each one of these values correspond to a class. Equation 2 gives the representation of an activation function.

$$y_{pred} = f(y_{out}) = \{1,\ if\ y_{out} \geq 0, -1\ if\ y_{ou} < 0\} \quad (2)$$

Where y_{pred} is the estimation of y feature. Then if the target is not equal to the expected output, bias and weights need to be updated. For this, it is necessary to apply the next function.

$$w_i(n) = w_i(o) + lr * \sum_{i=1}^{n} (y_i * X_i * 1_{\{y_i < X_i, w_i > < 0\}}) \quad (3)$$

Here, $w_i(n)$ represents the new value of w_i meanwhile $w_i(o)$ is its old value. lr corresponds to the learning rate, a parameter which indicates how much weights need to be adjusted in each training step [7]. The last term of the equation is given by the loss function, which in this case is a zero-one loss function. This kind of function gives the rate of error between a current estimation and the expected target, being zero-one loss one of the most used. Equation 4 shows this function.

$$l_{0/1}(h) = \{1\ if\ y_i < X_i, w_i > \ < 0,\ 0\ otherwise\} \quad (4)$$

There are more different loss functions which can be implemented, such as perceptron loss or hinge loss functions (among others), whose functions are shown in Equation 5 and 6, respectively.

$$l_{pern}(h) = max\{0,\ -y < x, w >\} \quad (5)$$

$$l_H(h) = max\{0,\ 1 - y < x, w >\} \quad (6)$$

Each one of the steps mentioned previously needs to be repeated in an iterative cycle until the perceptron model converges or reaches a specific limit of iterations. Based on this procedure, in each iteration (or epoch), the model would tend to give a more accurate result. At the starting point, in general, weights and bias are set randomly or in some cases use zero values. In summary, it is necessary to set the next hyperparameters: learning rate, number of epochs and loss function to use. Other parameters could be included and in this way we can point out the perceptron model of Sklearn, a python library that provides different unsupervised and supervised learning algorithms [8]. The implementation of perceptron in this case includes parameters of 'alpha', which corresponds to a constant value that multiplies the regularization term when this is used: 'l1_ratio' which is a kind penalty condition and 'tol', a stopping criterion, which corresponds to a special condition to stop the training when the ratio between the previous and current loss function does not achieve a certain threshold [8].

Finally, when this iterative process is finished, it is possible to have more suitable w_i and b values. With these values, a prediction of target feature y is possible using Equation 1.

3. Experimental Analysis

For an appropriate implementation of a perceptron model, it is necessary to follow a pipeline (workflow). A very general overview of a pipeline considers the next steps [9]:

- Ingest of data and split in train and validation sets
- Preparation of data (preprocessing)
- Fit a model using training data
- Using validation set to find best parameters
- Implementation of model in test

For this experiment, only a set of data of 768 samples is available from UCI Machine Learning Repository [10], which was previously normalized. For the preparation of data, the preprocessing includes the checking of null data and, correct data type of features and how much target is balanced. Later, data was splitted in three sets, train (70 % of original data), validation (20%) and test data (10%),

which is a common approach used in machine learning projects. Validation data will be useful to identify the best hyperparameters for the model. Finally, a last step includes the estimation of the target variable in unknown data (test using the best model (best fit) [11,12].

Additionally, to support the validation step, there is a technique called 'grid search', which consists in testing a combination of different hyperparameters to identify which is the best setting [12].

For this process, the next hyperparameters were tested:

- Number of epochs: 30, 50, 100 and 150
- Learning rate: 1, 0.1, 0.01 and 0.001
- Loss function: 'zero-one', 'perceptron' and 'hinge' loss functions.

Finally, the results of predictions also could be evaluated. For this, the accuracy metric is considered, which corresponds to the rate between right predictions over the total number of samples. In this way, the allowed values of accuracy goes from 0 to 1, being values close to 1 preferable to indicate that one model fits mostly 'well' to a specific problem [12]. Besides, there is a confusion matrix which is basically a table where it shows the rate of correct predictions and mistakes given by the estimation of a model, which is useful to detect where the model is making the main misclassifications.

5. Code implementation

For the practical implementation of the algorithm, Python was used as a program language and code was deployed in a Jupyter notebook. The code of the creation of model and testing is available in the Github repository https://github.com/victorcaquilpan/PerceptronAlgorithm.

6. Results and discussion

The analysis of data indicates that our target class is unbalanced with 500 people with diabetes and only 268 without it. Also, visually it is possible to check that there are no outliers in data.

Regarding the creation of the perceptron algorithm, a base case was deployed in the first instance. This base case includes a learning rate of 0.01, 30 epochs, use of zero-one loss function and bias (intercept) not considered. The result of the training of this model is shown in Figure 2. This plot indicates how the accuracy changes during each epoch. In the plot both train and validation accuracy are presented to show whether the model has overfitting/underfitting in unknown data. In this case, it is possible to notice that both curves follow a similar behavior through the time demonstrated that this model can generalize in an 'appropriate well' to the data.

On the other hand, Figure 3 illustrates how weight values (w_i) go to change by each epoch. Each one of these weights goes to take different values as more epochs are considered. Overall, these weights converge in a specific point.

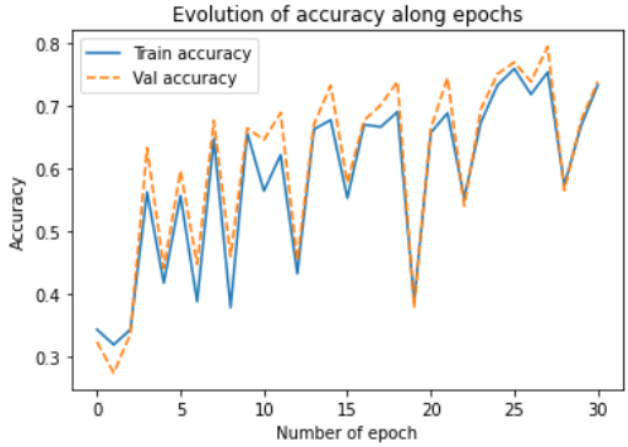

Figure 2. Implementation of perceptron algorithm (base case) and its accuracy values.

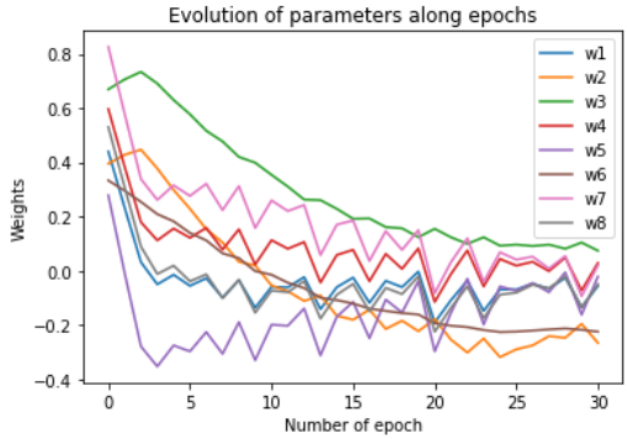

Figure 3. Evolution of weight values (w_i) through epochs in perceptron algorithm (base case).

During experimentation, several combinations of hyperparameters were tested. The main results are obtained using diverse loss functions and learning rates. Figure 4 presents the performance of a perceptron model using different loss functions. Figure 2 exposes the performance of a 'zero-one' loss function being clear that accuracy rises as the number of epochs progresses, however, a zigzagging behavior is shown up in the last epochs. Contrarily, using perceptron and loss functions is possible to get a steady increase of the accuracy.

Moreover, at 30 epochs, the model which gets the best accuracy is the last one reaching around 0.8.

In Figure 5, the performance of a perceptron algorithm with different learning rates values is presented. For this case, 'zero-one' loss function and 30 epochs were considered. It is observable that using a high value of learning rate (0.01) the performance of the model is erratic and zigzagging, while that employing a low value, the accuracy increase is steady but slower.

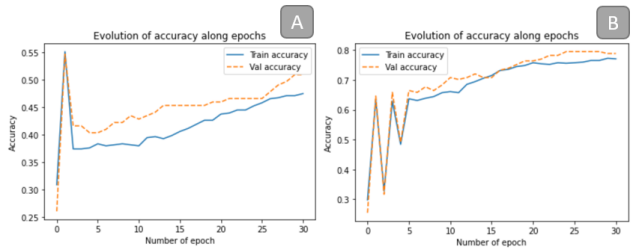

Figure 4. Experiments with perceptron algorithm testing loss functions. A = 'Perceptron' and B = 'Hinge'

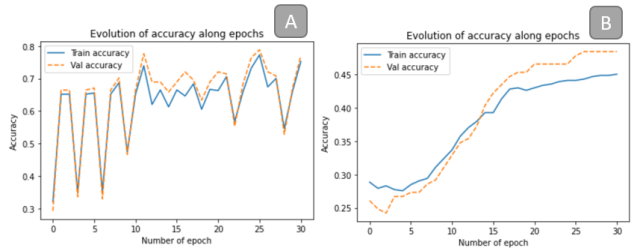

Figure 5. Experiments with perceptron algorithm testing learning rates. A = 0.01, B = 0.0001.

A summary of the validation step using grid search is shown in Table 2. 48 different combinations of hyperparameters were tested, however only the best five models are presented.

Ranking of model	Number of epochs	Learning rate	Loss function	Accuracy
1	150	0.001	Hinge	0.77
2	50	0.001	Hinge	0.77
3	30	0.001	Hinge	0.77
4	30	0.01	Zero-one	0.77
5	100	0.001	Hinge	0.76

Table 2. Summary of results in validation step.

Given the results presented in Table 2, it is possible to see a clear tendency where Hinge loss function and a learning rate of 0.001 give the best performance achieving an accuracy value of 0.77. Also, it is important to point out that in these cases, the number of epochs does not give an enhancement in the accuracy, thereby around 30 epochs is a enough number of iterations for this model. Furthermore, the fourth best performance is given by a model with Zero-one loss function, learning rate of 0.01 and 30 epochs, however as is indicated in Figure 2, this loss function gives a zigzagging behavior as iterations increase. Utilizing the best model, it obtained an accuracy of 0.76 in test data. Table 3 shows the confusion matrix by the implementation of the best fit.

Prediction **True**	**Diabetes**	**No diabetes**
Diabetes	13	3
No diabetes	14	40

Table 3. Matrix confusion for best model (best fit).

Here, it is possible to observe that from a total of 70 samples in test data (27: No diabetes, 43: Diabetes), the main mistakes are associated with the misclassification of people without diabetes. 14 cases of people without diabetes (20 %) were misclassified as diabetes cases, which is in practical terms the least serious scenario (type I error) in medical field. Although 76% of cases are predicted correctly, the rate of type I error is relatively high and 4% is misclassified as typeI error. This last is more serious for medical applications and in some cases is recommended to minimize this value as much as possible thereby other metrics could be suggested in future works, such as recall, which penalyses more false positive cases [13].

As a final part, a comparison with a perceptron model from sklearn library was presented. For this comparison, the same random initial values of w_i perceptron loss function and 0.0001 learning rate were utilized. The behavior of accuracy values during the training step are similar in both cases, however the accuracy in the test set is much better in sklearn model (0.6) than the created perceptron algorithm (0.38). This may be because sklearn has a larger number of hyperparameters which modify the performance of the algorithm.

7. Conclusion

This study consists in the implementation and experimentation of a perceptron algorithm. It was considered experiments testing the main hyperparameters identified in a perceptron model, such as learning rate, number of epochs and kind of loss functions.

From the tests carried out, it was evidenced that an extensive searching of hyperparameters is necessary to get a more robust performance. Despite, a perceptron is a simple neural network model, a correct selection of a loss function or learning rate is important to get a better fit.

The best result corresponds to a perceptron model using a hinge loss function, a learning rate of 0.001 and 150 epochs, obtaining an accuracy of 0.76 in testing data, however, it was identified a rate of type II error (4%), which impacted negatively the potential use of this model in a real context. Therefore, for future experiments, it is recommended to consider other metrics as recall to avoid having a high rate of type II error. Additionally, it was suggested the use of a more complex model (a neural network with hidden layers) to be able to fit in a better way to the behavior of this data.

8. References

[1] Awab Habib, et al., 2022. An efficient prediction of diabetes using artificial neural networks. AIP Conference Proceedings 2393.

[2] Zhi-Hua Zhou. 2021. Machine Learning. Springer, Gateway East, Singapore.

[3] Gopinath Rebala, Ajay Ravi and Sanjay Churiwala. 2019. An Introduction to Machine Learning. Springer Cham.

[4] Maniruzzman, Jahanur Rahman, Benojir Ahammed and Menhazul Abedin. 2020. Classification and prediction of diabetes disease using machine learning paradigm. Health Information Science and Systems. 2020. 8:7.

[5] Dutta Debadri, Paul Debpriyo and Ghosh Parthajeet. 2018. Analysis feature importances for diabetes prediction using machine learning. 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).

[6] Jaswinder Singh and Rajdeep Banerjee. 2019. A study on single and multi-layer perceptron neural network. Proceeding of the Third International Conference on Computer Methodologies and Communication (ICCMC 2019).

[7] Snehashish Chakraverty, Deepti Sahoo and Nisha Mahato. 2019. McCulloch-Pitts Neural Network Model. In: Concepts of Soft Computing. Springer, Singapore.

[8] Pedregosa Fabian et al., 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. Vol 12.

[9] Santosh Rao. 2019. Building a Data Pipeline for Deep Learning. White paper - NetApp.

[10] Dua Dheeru and Graff Casey. 2019. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[11] Yun Xu and Royston Goodacre. 2018. On Splitting Training and Validation Set: A comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. Journal of Analysis and Testing. Vol 2:3.

[12] Aurelien Geron. 2019. Hands-on machine learning with Scikit-learn, Keras and Tensorflow: concepts, tools and techniques to build intelligent systems. O'Reilly Media, Incorporated, 2019. ProQuest Ebook Central.

[13] Steven A. Hicks. 2021. On evaluation metrics for medical applications of artificial intelligence. Scientific Reports, Vol 12:5972.