Assignment 1, Deep Learning Fundamentals, 2022
Perceptron Algorithm

Duc Nhan Do
The University of Adelaide
Adelaide, SA, 5000
ducnhan.do@student.adelaide.edu.au

Abstract

The perceptron algorithm is a classic algorithm in Machine Learning and is often used for classification problems. In this experiment, as the goal is to gain deeper insights into the perceptron algorithm, this algorithm will be self-implemented from scratch and trained on the ***Pima Indians Diabetes*** *dataset. The algorithm will also be tested on different epochs and learning rates. Then, the best result will be compared with the perceptron algorithm from the sklearn library to inspect how well the self-implementation algorithm works.*

1. Introduction

Machine Learning is seen as a part of Artificial Intelligence and has recently become extremely popular in the technology community. Its algorithms and underlying architecture aim to create a model that can make predictions or decisions without being explicitly programmed to do so. This learning method is widely used to make predictions using its ability to generalise the input it has not seen before. The learning is proceeded continuously by minimising the difference between prediction and expected output from the training dataset. Machine learning can be applied in various areas, and one of its main tasks is classification [3]. In this experiment, one typical machine learning algorithm called ***Perceptron*** will be implemented to predict whether a patient has diabetes or not, using the ***Pima Indians Diabetes*** dataset. Furthermore, the perceptron algorithm will be tested and compared on different learning rates and numbers of epochs to find the optimal value that leads to the highest accuracy. The implementation code is available at: https://github.com/DoDucNhan/DL-assignment.git

2. Perceptron Alogrithm

In machine learning, the *Perceptron* is an algorithm for supervised learning of binary classifiers. This algorithm is a simplified biological neuron model that enables a computer to learn and process elements in the training set one at a time. After that, predictions will be made based on a linear predictor function combining a set of weights with the feature vector [5].

2.1. Loss function

Loss functions for classification are computationally feasible, representing the price paid for the inaccuracy of predictions in classification problems [2]. In the perceptron algorithm, the loss function is defined as follows:

$$l_{pern}(x, y, w) = max\{0, -y\langle x, w\rangle\} \tag{1}$$

To better understand this loss function, the first thing to notice is that the label of the data is $\{-1, 1\}$ and $y_{predict} = \langle x, w\rangle$. When the algorithm is misclassified, which means $y_{true} = 1, y_{predict} = -1$ or $y_{true} = -1, y_{predict} = 1$, the value of $-y_{true} \times y_{predict}$ will be greater than 0, and the max function will return the value of $-y_{true} \times y_{predict}$. Conversely, when the algorithm correctly predicts the label, the value of the max function returned will be 0. Therefore, this loss function can effectively record when the perceptron algorithm gives wrong predictions, and the goal of the learning process is to minimise calculated loss.

2.2. Learning algorithm

First is the mathematics notations for the algorithm:

- N: the size of the dataset
- w: the weights for making prediction
- (x, y): the input features and the corresponding labels, $y \in \{-1, 1\}$

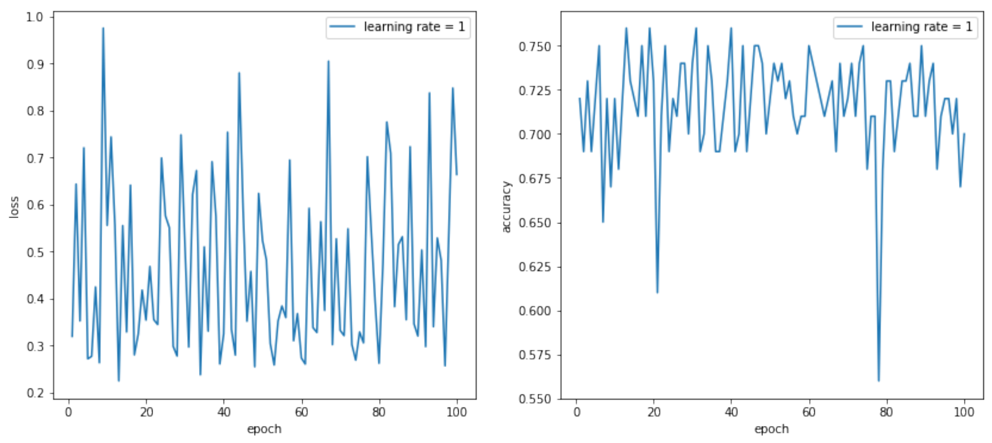


Figure 1. The *loss* and *accuracy* with no learning rate ($\alpha = 1$) after training for 100 epochs.

The learning process which is also known as updating weights of the perceptron algorithm is presented in the following steps:

Algorithm 1 Perceptron Algorithm

```
inputs: training data $\{(x_i, y_i)\}_{i=1}^{N}$, label $y \in \{-1, 1\}$,
learning rate $\alpha$, number of epochs $T$

initialization: initialize $w = 0$

procedure
    for $t := 1$ to $T$ do
        for $i := 1$ to $N$ do
            $y^* := \langle x_i, w\rangle$
            if $y_i \times y^* < 0$ then
                $w := w + \alpha(y_i \times x_i)$
            end if
        end for
    end for
end procedure

outputs: $w$
```

The $(y_i \times x_i)$ is the result of sub-gradient

$$\frac{\partial l_{pern}}{\partial w}$$

The class prediction of x is $\hat{y} = sign(\langle x, w\rangle)$, where $sign(z) = 1$ if $z \geq 0$, -1 otherwise.

3. Experiment Analysis

3.1. Dataset

The dataset used in this experiment is **Pima Indians Diabetes** (https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database). It consists of 768 rows and 9 columns. One column is for the label, which is $\{0, 1\}$ and others are information about *Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age*.

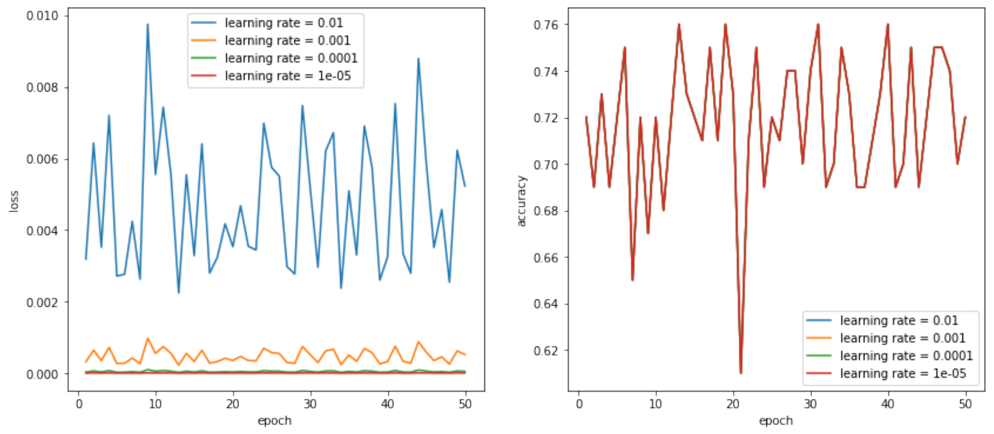
The scaled version of the dataset will be used, which is in the url: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html. The truth labels in this version are $y \in \{-1, 1\}$ indicating whether a patient has diabetes or not.

3.2. Learning rate

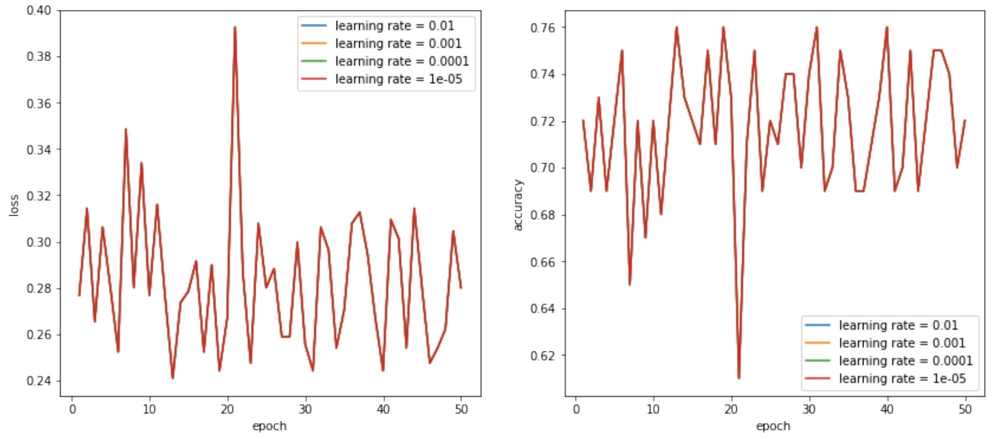
The perceptron algorithm was trained with four different learning rates, $\alpha \in \{0.01, 0.001, 0.0001, 0.00001\}$ and the same epochs of 50. After training, the low loss corresponds to a lower learning rate. However, the accuracy is the same regardless of different learning rates (Fig. 2a). Therefore, the $sign$ function was applied to $\langle x, w\rangle$ in Eq. 1 to investigate how the loss would change if the correctness of the prediction (value of $\{-1, 1\}$) was the only concerning aspect. As shown in Fig. 2b, after applying the $sign$ function, the loss is the same even if the learning rate changes. This result shows that the learning rate only scales the output value of the inner product $\langle x, w\rangle$ and has no effect on the sign of the output value [1,4].

3.3. Number of epochs

Since the learning rate does not affect the perceptron algorithm's performance, it will be assigned to 1, which can be understood that there is no learning rate during training. As shown in Fig. 1, after 100 epochs, the accuracy is fluctuated and hard to identify the pattern. The algorithm has also been trained on various epochs, and accuracy's constant ups and downs are still happening. This phenomenon happens because the dataset is non-linearly separable, the algorithm continues updating, and the inconsistent rise and fall of loss

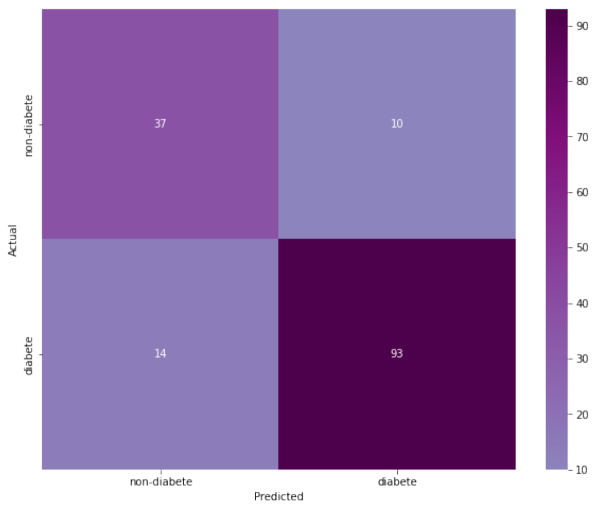


(a) The *sign* function is not applied to the output.

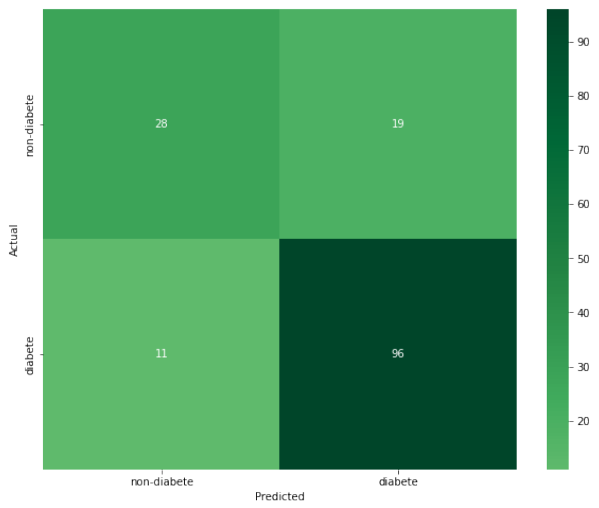


(b) The *sign* function is applied to the output.

Figure 2. The *loss* (the left side) and *accuracy* (the right side) of different learning rates when applying and not applying *sign* function to the output.



(a) The confusion matrix of the implemented perceptron.



(b) The confusion matrix of the perceptron using library.

Figure 3. The confusion matrix comparison between the implemented perceptron and the perceptron from *sklearn* library.

and accuracy occur. Therefore, a straightforward approach to address this difficulty is finding the peak of accuracy (or the minimum loss). The epoch corresponding to that accuracy will be the best number of epochs for the training process.

	Precision		Accuracy
	non-diabete	diabete	
self-implementation	0.73	0.90	**0.84**
sklearn	0.72	0.83	0.81

Table 1. The precision and accuracy comparison between self-implementation percetron and *sklearn* perceptron.

The self-implementation algorithm was compared with the perceptron algorithm from the *sklearn* library with default parameters to evaluate the performance of the self-implementation algorithm. After retraining with the best epoch found earlier (the best epoch in the experiment is 13), the self-implementation algorithm's accuracy is 84%, which is 3% higher than the *sklearn* library (Table. 1). The confusion matrix in Fig. 3 also shows more details about class predictions of the *sklearn* and self-implementation algorithms. Thus, this comparison has confirmed that determining the peak of accuracy and its matching epoch will bring promising results. In this case, just training with a small number of epochs has brought excellent performance instead of continuing to train over 100 epochs but still needs to be determined if the accuracy is increasing.

Based on the above analysis, the limitations of this simple perceptron algorithm are apparent. First, perceptron is used only for *Binary Classification* problems. Second, the algorithm can only work well with linearly separable sets of vectors.

4. Conclusion

Through experimentation, it can be seen that the algorithm is not affected by the learning rate because multiplying the update by any constant simply rescales the weights but never changes the sign of the prediction [4]. In this classification problem, the best performance of the algorithm can be found through the peak of the accuracy record. However, the loss and accuracy of the algorithm are very inconsistent in this non-linearly separable data, so it is quite challenging to find the optimal epoch for the algorithm if training with a larger and more complex dataset. Since the perceptron algorithm will never converge for non-linearly separable data, classifiers using non-linear decision functions such as *Quadratic Discriminant Classifier, Multi-Layer Perceptron (MLP), Decision Trees, Random Forest, and K-Nearest Neighbours (KNN)* might be better solutions to the classification problem in this context.

References

[1] Learning rate in the perceptron proof and convergence - data science stack exchange. https://datascience.stackexchange.com/questions/27909/learning-rate-in-the-perceptron-proof-and-convergence. (Accessed on 09/17/2022). 2

[2] Loss functions for classification - wikipedia. https://en.wikipedia.org/wiki/Loss_functions_for_classification. (Accessed on 09/18/2022). 1

[3] Machine learning - wikipedia. https://en.wikipedia.org/wiki/Machine_learning. (Accessed on 09/17/2022). 1

[4] neural network - perceptron learning rate - data science stack exchange. https://datascience.stackexchange.com/questions/16843/perceptron-learning-rate. (Accessed on 09/17/2022). 2, 4

[5] Perceptron - wikipedia. https://en.wikipedia.org/wiki/Perceptron. (Accessed on 09/17/2022). 1