Predict diabetes using Perceptron

Hu Xiaolong
The University of Adelaide
Adelaide SA 5005
A1912235@adelaide.edu.au

.

Abstract

This paper will provide process on implementing, describing, and testing a specific algorithm called Perceptron, which can be interpreted as a dense layer neural network, for predicting diabetes using the related diabetes dataset. And will compare the accuracy and analysis results between the single-layer perceptron and MLP models.

1. Introduction

Diabetes is a global health problem with an increasing prevalence rate, which has placed a heavy burden on individuals and society. According to the World Health Organization, diabetes has become one of the leading causes of death worldwide, and prevention and early diagnosis are essential to reduce the incidence [1]. Therefore, developing effective prediction models to identify high-risk individuals has become an important research direction in the field of public health.

1.1. Purpose

This paper aims to predict diabetes using machine learning algorithms, especially single-layer perceptrons and multi-layer perceptrons (MLPs). Perceptrons are simple linear classifiers that can effectively perform binary classification tasks by learning the relationship between input features and target labels. Despite their simple structure, they still show good results in many practical applications. On the other hand, MLPs, as a more complex neural network structure, can capture nonlinear relationships in data through multiple hidden layers, thereby improving classification performance.

1.2. Process

This paper will introduce basic single-layer perceptron to understand its working principle and limitations. Then, building an MLP model to explore its performance in diabetes prediction and compare it with the results of a single-layer perceptron. Through this process, hope to not only improve the prediction accuracy, but also gain a deeper understanding of the relationship between model performance and hyperparameter settings.

2. Method Description

The following will explain the algorithm and the model used in detail, and provide examples to illustrate its effects and shortcomings. Some possible improvements and their working principles are also attached to facilitate implementation by others.

2.1. Single-layer perceptron

The single-layer perceptron is one of the simplest neural network models suitable for linearly separable binary classification problems. Its basic structure includes an input layer and an output layer, where each node in the input layer corresponds to a feature, and the output node generates the prediction [2]. The functioning of the single-layer perceptron is as follows:

1. **Input Features**: The input data is represented as a feature vector x=[x_1,x_2,…,x_n].
2. **Weights and Bias**: Each input feature is assigned a weight w_i and a bias b.
3. **Linear Combination**: The weighted sum is calculated as $z = \sum_{i=1}^{n}(w_i x_i + b)$
4. **Activation Function**: The linear combination is passed through an activation function to produce the output. For binary classification tasks, the step function is commonly used:

$$\boldsymbol{y} = \begin{cases} \mathbf{1} \ \boldsymbol{if} \ \boldsymbol{z} \geq \mathbf{0} \\ \mathbf{0} \ \boldsymbol{otherwise} \end{cases}$$

5. **Weight Update**: The weights and bias are updated using the perceptron learning rule when the prediction does not match the true label:

$$w_i \underset{\square}{\leftarrow} w_i + \eta\left(y_{\text{true}} - y_{pred}\right)x_i$$
$$b \underset{\square}{\leftarrow} b_{\square} + \eta\left(y_{\text{true}} - y_{pred}\right)$$

where η is the learning rate.

While the single-layer perceptron is simple to implement, it can only handle linearly separable problems and cannot solve complex non-linear classification tasks.

2.2. Multi-Layer Perceptron

The multi-layer perceptron (MLP) is an extension of the neural network that includes at least one hidden layer, allowing it to capture non-linear relationships within the input data [3]. The structure and functioning of MLP is as follows:

1. **Network Structure**: An MLP consists of an input layer, one or more hidden layers, and an output layer. Each layer's nodes are interconnected by weights.
2. **Forward Propagation**: Input data is processed through each layer using weighted sums and activation functions, which may include ReLU [4], Sigmoid, or Tanh. The output layer produces the final prediction through an activation function.
3. **Loss Function**: An appropriate loss function (such as cross-entropy loss) is used to assess the model's prediction performance.
4. **Backpropagation**: Gradients for each weight are computed using the chain rule, and weights are updated using gradient descent. In addition, the backpropagation process involves calculating errors and updating weights layer by layer from the output layer back to the input layer.

By increasing the number of hidden layers and nodes, MLP can significantly enhance the model's expressive power, making it suitable for more complex tasks.

2.3. Improvement

This paper gives result of impact of different hyperparameter settings (such as learning rate and number of hidden layers) on model performance. By systematically adjusting these parameters, it aim to improve predictive accuracy and gain deeper insights into how hyperparameter selection affects model training.

3. Experimental Test

The following shows the data preprocessing steps, the experimental setup for testing the models, and the results obtained from different configurations of the Perceptron and Multilayer Perceptron (MLP) models.

3.1. Data Preprocessing

The dataset used for this experiment was preprocessed before training the models. First, the labels with format of +1 and -1, were converted into 0 and 1 for binary classification. Since the number of features in each sample was expected to be 8, rows with a different number of features were filtered out. Additionally, no missing values were found in the data, to ensure that the data was clean and ready for model training. Finally, the dataset was split into training and test sets, using an 80/20 ratio (test_size=0.2) and a fixed random seed (random_state=42) to ensure reproducibility.

3.2. Model Testing and Hyperparameters

Both Single Layer Perceptron (SLP) and Multilayer Perceptron (MLP) were extensively tested with different hyperparameters such as learning rate and number of epochs. The following settings were tested:

Learning rate: 0.1, 0.01, and 0.001
Number of epochs: 200 and 500

The models were evaluated based on the loss reduction during training and the accuracy achieved on the test set.

3.3. SLP Results

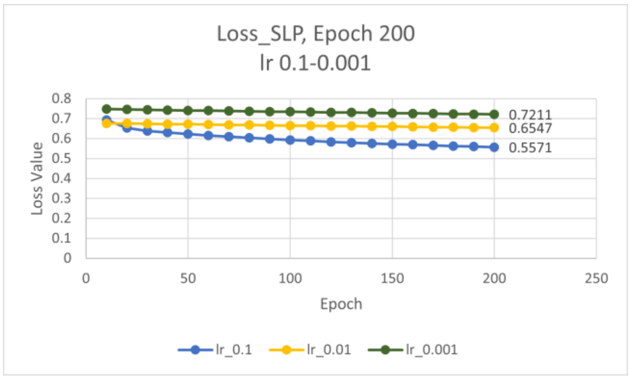

Fig.1. Observe the loss curve change of the SLP model from 0.1 to 0.001 at 200 epochs (spacing 10)

The SLP model shows rapid convergence, the loss drops from 0.6930 in the 10th epoch to 0.5571 in the 200th epoch, and the test set accuracy reaches 0.7632. However, as lr changes, the accuracy also decreases. At 0.01, the accuracy is 0.6908. At 0.001, it is only 0.3289. At the same time, the loss is high at the beginning, and then it decreases slowly, from 0.7474 in the 10th epoch to 0.7211 in the 200th epoch.

3.4. MLP Result

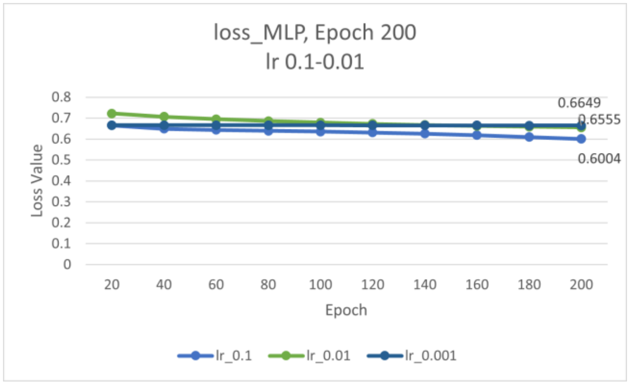

Fig.2. Observe the loss curve change of the MLP model from 0.1 to 0.001 at 200 epochs(spacing 20)

It can be seen that at lr 0.1, MLP performs well, with a faster reduction in loss and a test accuracy of 0.6908, which is comparable to SLP. However, when the lr is 0.01 and 0.001, the loss rate and accuracy rate have almost no change. The loss rate is almost the same, while the accuracy rate is consistent in 0.6908, indicating that at this relatively small learning rate, the learning effect of MLP is very poor.

3.5. Compare both model in Epoch 500 lr 0.01

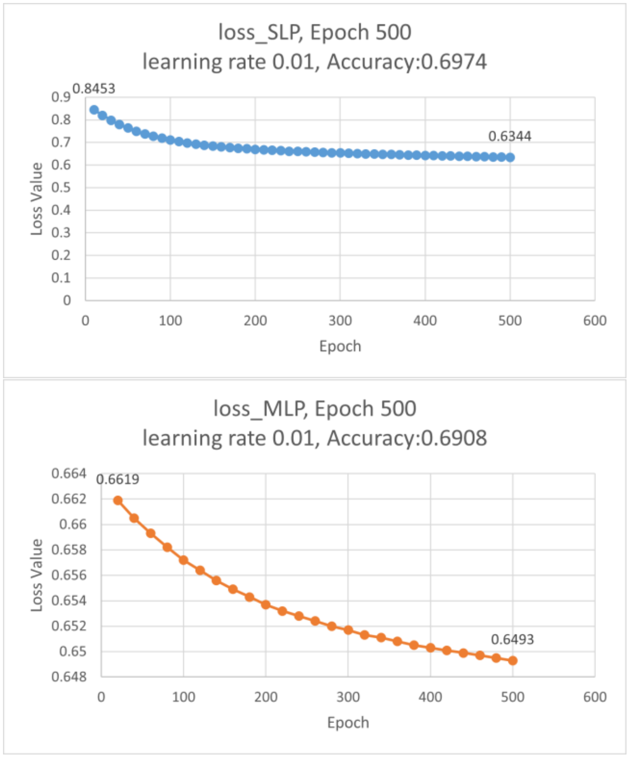

Fig.3. Compare the loss curve change of the SLP and MLP model in lr 0.01 at 500 epochs(SLP spacing 10, MLP 20)

It can be observed that the SLP model has a significant convergence, reaching 0.6344 at 500 epochs, and the accuracy has reached 0.6974. However, the MLP converges slowly, and the final loss and accuracy are basically the same as SLP.

4. Results Analysis

The single-layer perceptron performed best at a learning rate of 0.1 and 200 epochs, with an accuracy of 76.32%. This shows a lower learning rate leads to slower convergence and lower accuracy. And despite its complexity, the multilayer perceptron does not perform significantly better than SLP. The accuracy of MLP is stable at 69.08% under any lr.

These means that a higher learning rate helps speed up convergence, but a learning rate that is too high may cause the model to skip the optimal point. In addition, in terms of model performance, the comparison between MLP with SLP did not show significant improvement, which may be due to overfitting or insufficient model complexity to meet the requirements of the data set.

Although both SLP and MLP models show good performance on the dataset, SLP with a simpler structure achieves higher accuracy with the tuning of hyperparameters. MLP may need further tuning of hidden layers or hyperparameters to prevent overfitting and realize its potential for better performance.

5. Conclusion

This paper provides an in-depth discussion of the Perceptron model and its performance in binary classification tasks, especially the impact of different learning rates and training cycles (epochs). The basic perceptron method is stable in terms of accuracy and loss convergence, but its limitation is that it is difficult to deal with more complex nonlinear classification problems. Model performance can be effectively improved by adjusting the learning rate, which has been verified in experiments. Although multi-layer perceptrons fail to significantly surpass the performance of single-layer perceptrons, their improvements are effective under certain conditions.

Considering the future optimization of the model structure, nonlinear activation functions such as ReLU and more advanced optimization methods such as Adam [5] will be good methods. These can effectively improve training efficiency and accuracy. Furthermore, the generalization ability of the model can be further improved by increasing the size of the data set and enriching the feature processing methods.

GitHub links: https://github.com/SpiderJockey7/Perceptron_Predict_diabetes.git

References

[1] WHO, “Diabetes,” *World Health Organization*, 2024. https://www.who.int/health-topics/diabetes#tab=tab_1

[2] H. D. Block, "The Perceptron: A Model for Brain Functioning," *Rev. Mod. Phys.*, vol. 34, pp. 123–134, Jan. 1962.

[3] A. J. Shepherd, *Second-order Methods for Neural Networks: Fast and Reliable Training Methods for Multi-layer Perceptrons*, 1st ed., Perspectives in Neural Computing, Springer, 1997.

[4] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, Z. Liu, H. Bischof, J.-M. Frahm, T. Brox, and A. Vedaldi, "Dynamic ReLU," in *Computer Vision – ECCV 2020*, vol. 12364, 2020, pp. 351–367.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1412.6980.