Machine Learning Lab1: Construct a Back Propagation Neural Network

[redacted]

March 15, 2023

1 Purpose

From my perspective, implementing the back propagation algorithm in detail is challenging since I used to call the functions in packages. like PyTorch. However, I think this lab leads me to have a deeper understanding of back propagation neural networks. Back propagation is widely used in the training of neural networks. The purpose of back propagation is to use the Error Back Propagation Training to adjust the parameters (weights, bias) and minimize the difference with the true value.

In this lab, I will use BP neural network to implement a multiclassification task with 3 labels of the Iris dataset. In detail, I will write the code about core of the BP algorithm, including feed-forward, activation function, gradient descent, the structure of neural network and layers. Then I will train this neural network and test it using some metrics.

This work is mainly about using BP neural network to solve a classification problem. I'll try my best to duplicate the BP neural network and improve its performance.

2 Dataset

2.1 Basic Information

Iris[1] is a famous dataset used in classification problems. Here is the attribute information about it:

- Sepal length in cm
- Sepal width in cm
- Petal length in cm
- Petal width in cm
- Class: iris setosa, iris versicolour, iris virginica

The first four attributes are the input feature of our model while the last one, class, is the label. It contains 150 pieces of data without any missing data, Thus I can directly load the data.

2.1.1 Data Visualization

To have a better view of this dataset, I use matplotlib to visualize the dataset, which helps us to figure out the potential relations.

[1] archive.ics.uci.edu/ml/datasets/iris

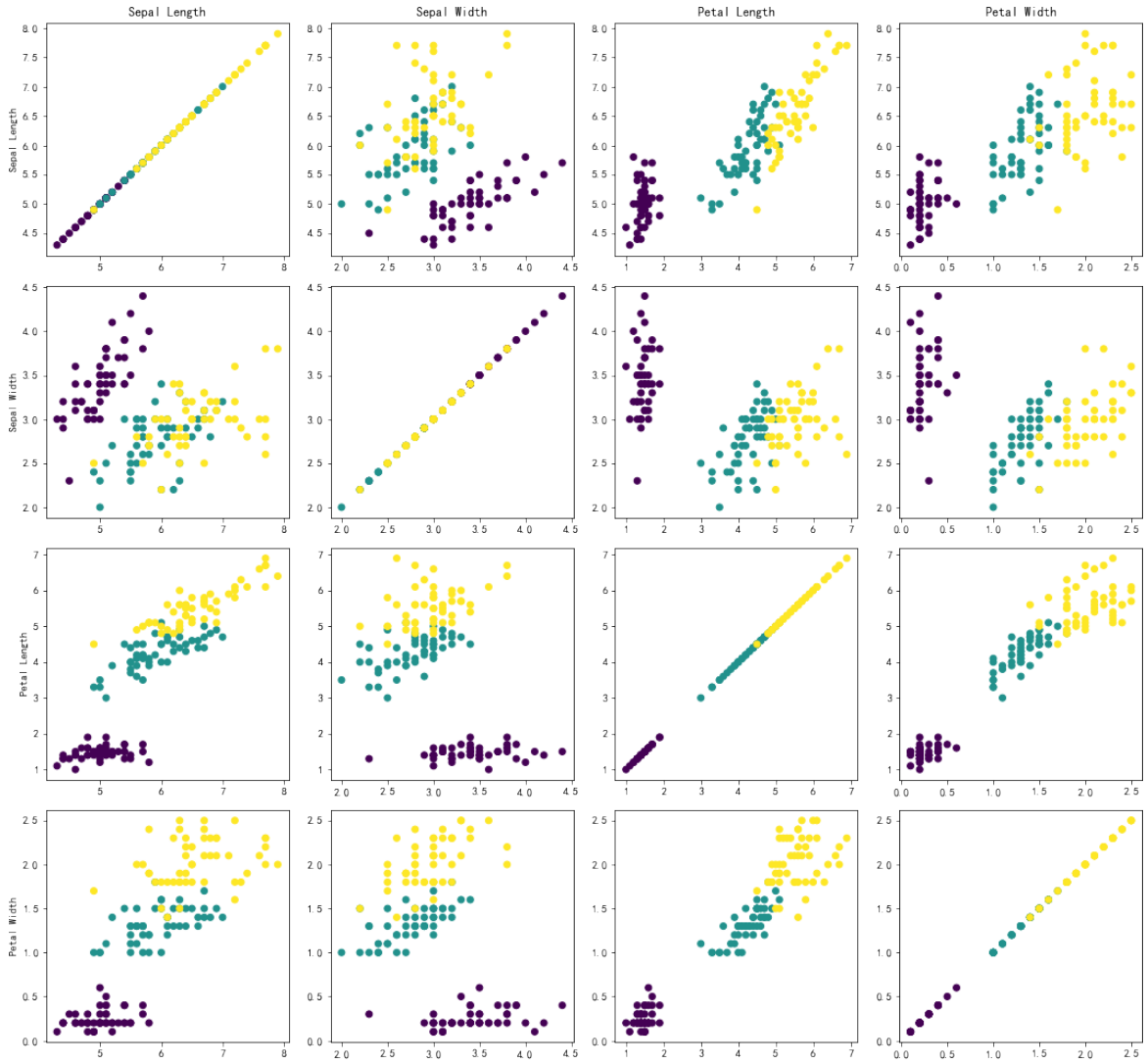

Figure 1: Correlation Map

From Figure 1, we can observe that these three classes can be divided by four features, which indicates the feasibility of our model.

3 Metric

To evaluate the performance of the model, I use accuracy and recall as the metrics.

Accuracy: It measures the percentage of correctly predicted outputs in the total number of inputs of the dataset. Accuracy is a straightforward metric that can measure the model's overall performance.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{1}$$

Recall: Compared to accuracy, recall is suitable when we want to separate all the positive instances correctly. However, only using recall maybe incorrectly classify negative instances as positive (false positives). Precision is a complementary metric of recall.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

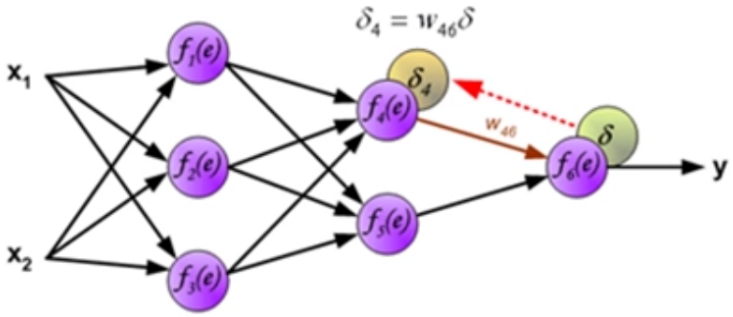

Figure 2: BP Neural Network[1]

4 Main Idea

In this section, I will explain the basic theory of back propagation and how it works in classification problems, especially in iris classification. This basic structure of BP neural network with one input layer three hidden layers and one output layer is as follows:

How does BP neural network work? Actually, BP neural network is an optimization of ANN in gradient descent. It will update the weights and bias of neural network based on the loss in last iteration, which is an efficient way to solve the gradient. It can be concluded into three steps:

- Propagate forward to calculate the objective function.

$$f_i(x) = \sum_{j=1}^{n} w_{ij}x_j + b$$

- Back propagation finds the gradient based on the loss function. In this case, we use cross-entropy loss function.

$$Loss = -\sum_{i=1}^{n}[y_i log(a_i) + (1 - y_i)log(1 - a_i)]$$

- Update the parameters according to gradient information.

$$w_{ij}* = w_{ij} - \alpha\frac{\partial l_j}{\partial w_i j}$$

We should repeat the three steps above until the error drops below our expectations.

4.1 Classification Problems

BP neural network can be used in many kinds of problems, such as regression and classification. For regression, it may only have one output while classification may have multi-choices.

In iris classification, we have tree outputs, meaning the size of output layer is 3. First, we should modify the label into digital form (0, 1,2). During processing, one-hot encoding is preferred. Thus we transform the label into three rows: $0 \longrightarrow [1, 0, 0]$. As to the prediction results formed in decimal form, we select the largest one assigning to 1 and others to 0 because the predictions reflect the probability of which kinds of iris the features represent.

5 Design

In this section, I mainly discuss the implementation details of my model. The pseudocode can be represented as follow:

Algorithm 1: Pseudocode of BP Neural Network

Data: Iris Data

foreach *i in iterations do* **do**

$a2, cache = forward\_propagation(X, parameters)$

$cost = compute\_cost(a2, Y, parameters)$

$grads = backward\_propagation(parameters, cache, X, Y)$

$parameters = update\_parameters(parameters, grads)$

end

5.1 Load Data

As the dataset doesn't need to preprocess, only two important steps are required:

- Separate the dataset into training data and test data with a ratio of 6:4
- Change the label into one hot encoding.

5.2 Set up Neural Network

I'm going to construct a BP neural network with 2 hidden layers.

5.2.1 Forward Propagation

For activation function, I choose tanh function. Because I have done experiments on sigmoid function and tanh and found that tanh performs better in stability and accuracy.

$$tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

5.2.2 Back Propagation

For two hidden layers, I have four parameters $w_1, W2, b_1, b_2$ that need to update. The update strategy is mentioned in the last section.

6 Results and Analysis

6.1 Results

Here are the results of accuracy, recall and loss function:

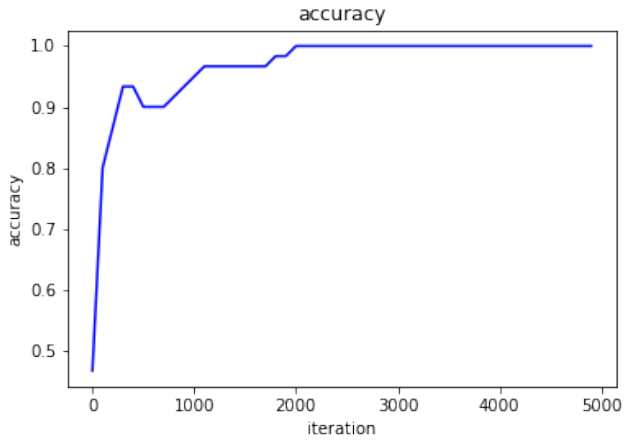

Figure 3: Accuracy

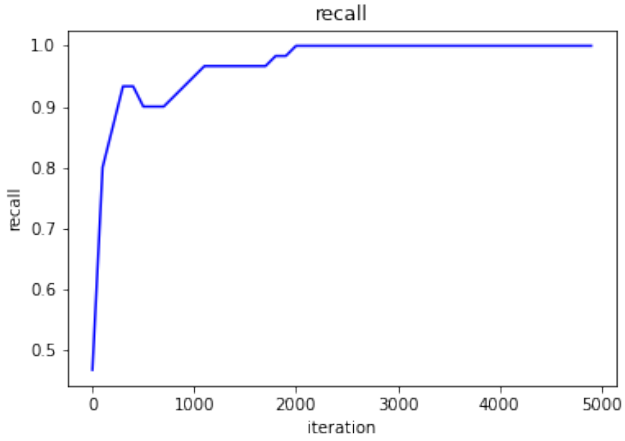

Figure 4: Recall

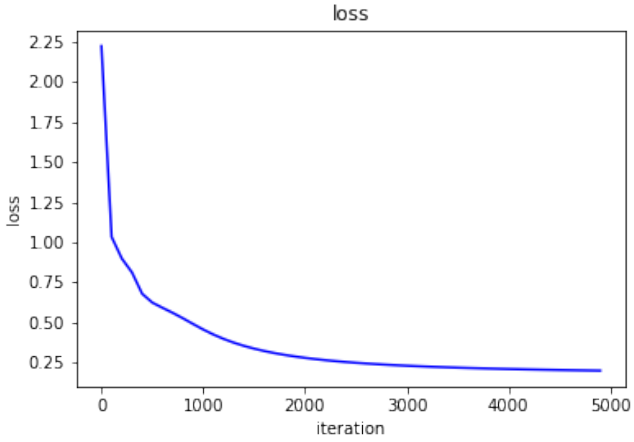

Figure 5: Loss Function

From figures above, with the training epoch growing, the accuracy and recall increase while the loss decreases. Straightforwardly, it demonstrates the effectiveness of our model and the accuracy and recall can reach up to 0.983 and 0.981 respectively.

6.2 Analysis

I achieve a relatively good result and I need to explore further to explain this situation. I collect the data of accuracy versus epoch: [0.3, 0.6833333333333333, 0.85, 0.8666666666666667, 0.9166666666666666, 0.9166666666666666, 0.9166666666666666, 0.9333333333333333, 0.9333333333333333, 0.9333333333333333]. From this data, I observe that accuracy suddenly increases in the first 200 iterations and becomes stable and then maintains at 0.93. To explain this situation, I think the property of gradient plays a role. As our results approach the optimal solution, the slope around it is small. Thus, the step size of gradient descent becomes smaller, meaning the update of parameters becomes slow and stable. The hyperparameter, learning rate, also affects the step size.

6.3 Weakness

It takes too many epochs to train. Usually, the number of epoch is around several hundred. However, in this case, this number comes to several thousand which is abnormal. According to my study, the fact is that the batch size in this lab is large. Every time I train the model, I put all the training set

into it which means one epoch is one iteration and the batch size is the size of data. Usually, a larger batch size means you need more epoch to get the same accuracy but the time to process is faster[2][2].

At the same time, the accuracy of this model may not very high. I make a comparison between the PyToch version. We can find that the accuracy surprisingly come to 1. I think there are many things that I can optimize.

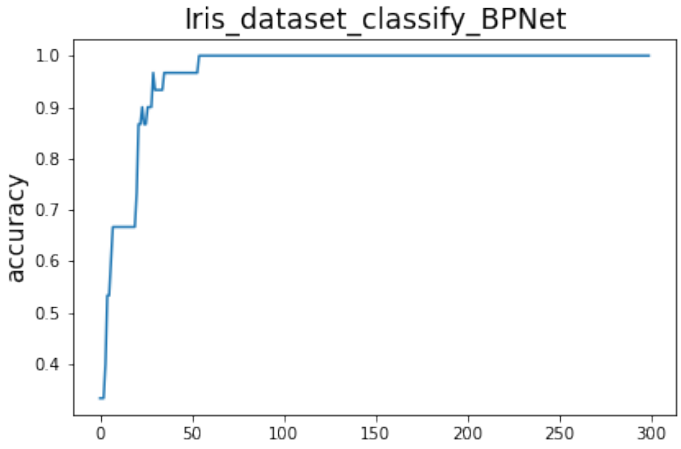

Figure 6: Use PyTorch to Construct

And another weakness is that the prediction results of this model aren't stable. Every time I rerun the program, the results change differently. I don't know how to solve this problem yet. (This problem has been solved in Section 8.)

7 Conclusion

In this lab, I implement the basic components of BP neural network and successfully complete iris classification with the accuracy reaching 0.98. For further study, I think I can improve the following aspects:

1. Divide the training data into different batches to decrease the training epochs.
2. Improve calculation accuracy. Since PyTorch is written in C++ which has higher precision than NumPy, the results in my model may have less accuracy.
3. Parameters should be adjusted. In the future, I may try to change the number of hidden layers, nodes in the hidden layers, and learning rate to find out the best parameter combination.

8 Update

On March 15th, I adjust the parameters and some details, the model achieves a better performance. Both recall and accuracy reach 1 and performance more stable.

[2]https://stats.stackexchange.com/questions/164876/what-is-the-trade-off-between-batch-size-and-number-of-iterations-to-train-a-neu

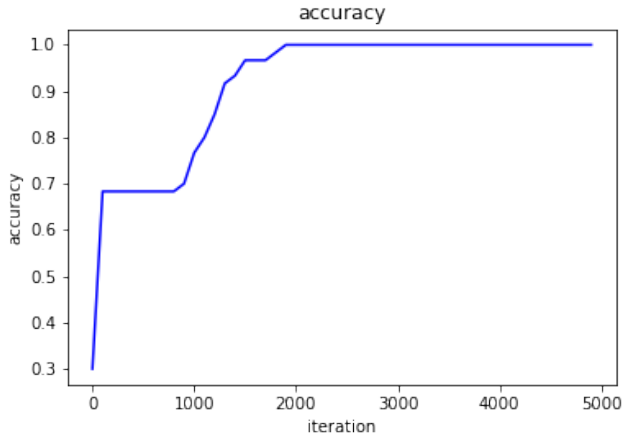

Figure 7: Accuracy Updated

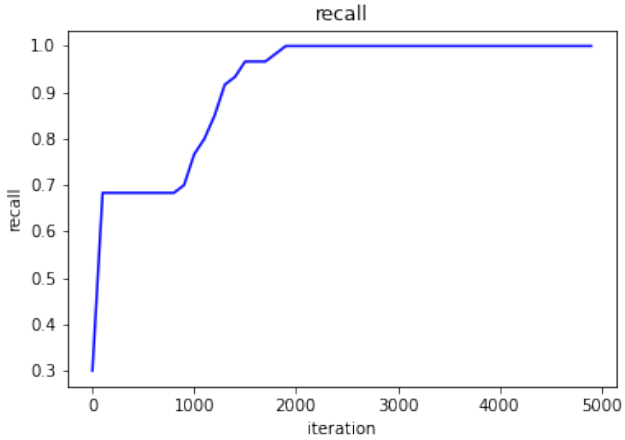

Figure 8: Recall Updated

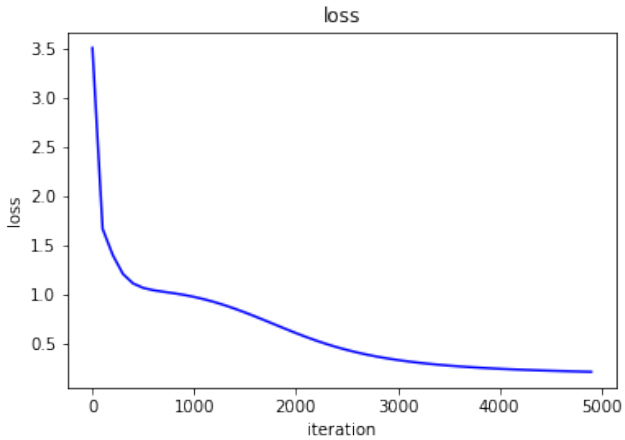

Figure 9: Loss Updated

References

[1] Li K, Machine Learning and Knowledge Discovery, 2023 Spring

[2] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.