Neural Network: Two Layers Classifier via Numpy

22110980014 Xu Shi

April 9, 2023

Abstract

In this project, we build a neural network two layers classifier only using Numpy on handwriting dataset of MNIST. We firstly train a classifier model, then, seek the optimal parameters $i.e.$, learn rate, hidden layer size and regularization strength. Finally, we load the model, test it with the optimal parameters, and output the classifier accuracy in train, validation and test data, respectively.

1 Introduction

Neural network classifier is a very powerful tool in machine learning, deep learning and other fields, the two layers classifier is a special case. Here we use the handwriting dataset in MNIST which is one of the most popular datasets to train the neural network model. It contains 50000, 10000, 10000 figures for training, validating and testing, respectively. We train our model by two matrices, weight and biases matrix, respectively. Meanwhile, we compare two different activation functions sigmoid and ReLU for output layer.

The readers can get the **detail codes in GitHub**[1]. Meanwhile, the **trained model** can be found in Baidu Netdisk[2] with fetch code is {t5jp} (If you can't download it from Baidu Netdisk, you can find it in GitHub, as well).

2 Algorithms

Firstly, we denote the model loss $L(x, \delta)$, where x is data, δ is parameter. Meanwhile, the model loss also contains the regular term $\frac{1}{2}\lambda\|\delta\|_2^2$. Then we can propose the important stochastic gradient descent (SGD) algorithm(Algorithm 1). Then, we know that to train a model on MNIST datasets, we should sample a

Algorithm 1 Stochastic Gradient Descent (SGD)

Input: Data x_0, model loss function L, parameter δ_k, learn rate η, regular parameter λ.
for $k = 0 : m$ **do**
  Compute x_{k+1} in some rules.
end for
for $k = m : 0$ **do**
  Compute $\frac{\partial L}{\partial \delta_k}$ in chain rule.
  Update $\delta_k = \delta_k - \eta\frac{\partial \mathcal{L}}{\partial \delta_k}$.
end for

minimal batch from the whole datasets and using the SGD method every iteration. So we should first randomly shuffle the datasets for stochastic updates. Then, after 1 epoch or more, we validate the model on the validation data and check its accuracy. If accuracy decreases, we should decrease the learn rate descent strategy accordingly. We propose our training model strategy as following (Algorithm 2),

[1] https://github.com/Shixu-max/Numpy-neural-network-two-layers-classifier
[2] https://pan.baidu.com/s/1hzkMCHe8OqT3ZuXBRyiuug

Algorithm 2 Training Model Strategy

Input: Data x, data label y, validation data $\hat{x}$, data label $\hat{y}$, learn rate η, constant $0 < \rho < 1$, iteration n.
for $k = 1 : n$ **do**
    Randomly shuffle x, y.
    for choose a batch of x, y **do**
        Use SGD method on the choosen batch.
    end for
    Compute the accuracy on validation data.
    if Accuracy descent **then**
        $\eta = \rho\eta$.
    end if
end for

3 Parameters Seek

In this section, we firstly choose the best loss and activation functions, the we propose how to seek the optimal parameters *i.e.*, learn rate, hidden layer size and regularization strength.

3.1 Loss Function

Above all, we should choose a suitable loss function, here we consider two loss functions, the mean square error (MSE) and the cross entropy (CE) loss. We test which is better under activation function is ReLU, fixed learn rate = 1e-2, hidden size = 500, batch size = 40 and epochs = 10, we have following results, From this Figure 1 we can see that, MSE performs better than CE, so in following part, we choose MSE as loss function. (If you want to see detailed results, you only need to change loss function in **Change Loss Function** in the code file **twolayerclassifier**).

3.2 Activation Function

Firstly, we know that our neural network have two layers, so we should have two activation functions, the second activation function is fixed by sigmoid, since it should scale the output to $[0, 1]$. So we need to test ReLU and sigmoid functions only for the first activation function under loss function is MSE, fixed learn rate = 1e-2, hidden size = 500, batch size = 40 and epochs = 10, we have following results, From this Figure 2 we can see that, ReLU performs better than sigmoid, so in following part, we choose ReLU for the first activation function (If you want to see detailed results, you only need to change the first activation function = 'ReLU' or 'sigmoid' in **Change Activation Function** in the code file **twolayerclassifier**).

3.3 Learn Rate

In Algorithm 2, the learn rate descent strategy is important. If the validation accuracy decreases, we should reduce the learn rate. We train our model with learn rate is one of $[1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128]$ under loss function is MSE, activation function is ReLU, hidden size = 500, batch size = 40 and epochs = 10, the result is as following, Then, from Table 1 we know that **the optimal learn rate is 0.03125**

Table 1: Learn Rate

Learn rate	1	1/2	1/4	1/8	1/16	1/32	1/64	1/128
Accuracy	0.0991	0.0991	0.0991	0.9729	0.9813	0.9815	0.9815	0.9787

with corresponding accuracy is 0.9815.

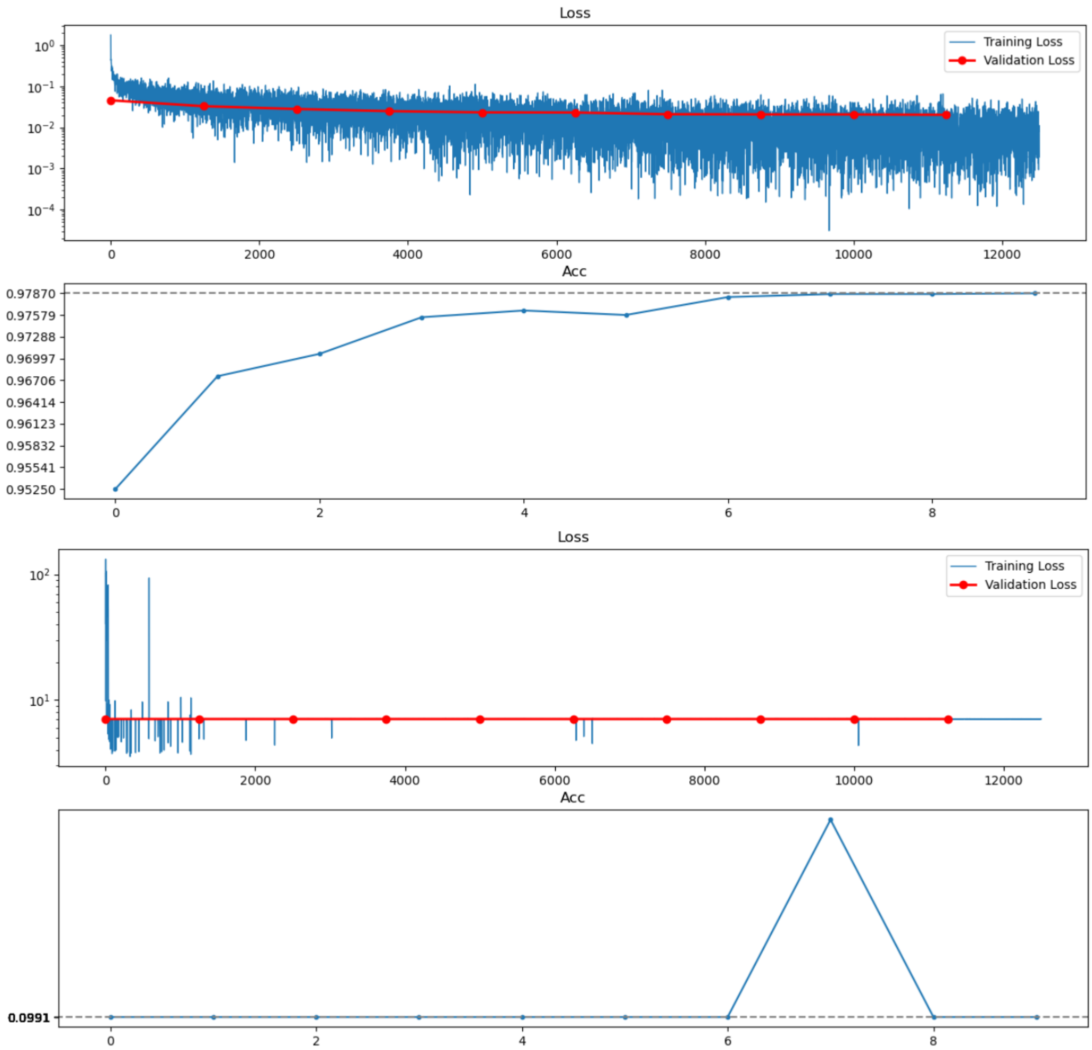

Figure 1: MSE vs BCE

3.4 Hidden Layer Size

The hidden layer size is also very important. We train our model with hidden layer size is one of $[10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]$ under loss function is MSE, activation function is ReLU, learn rate $= 0.03125$, batch size $= 40$ and epochs $= 10$, the result is as following, Then, from Table

Table 2: Hidden Layer Size

Hidden Size	50	100	200	300	400	500	600
Accuracy	0.9391	0.9744	0.9793	0.9802	0.9825	0.9829	0.9815
Hidden Size	700	800	900	1000			
Accuracy	0.9826	0.9821	0.9834	0.9833			

2 we know that **the optimal hidden layer size is 900 with corresponding accuracy is 0.9834**.

3.5 Regularization Strength

For the L2 regularization strength, we train our model with hidden layer size is one of $[1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 0]$ under loss function is MSE, activation function is ReLU, learn rate $=$

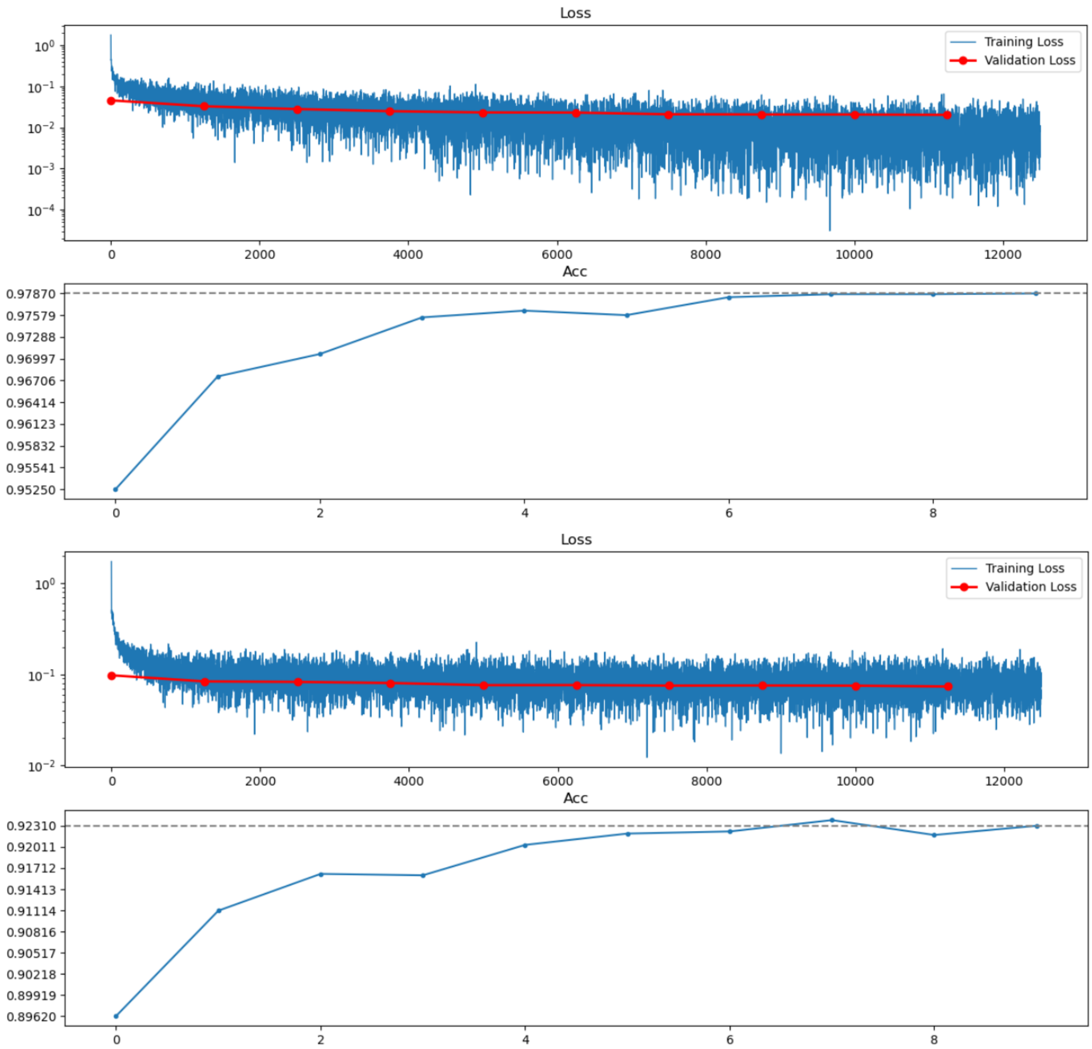

Figure 2: ReLU vs sigmoid

0.03125, hidden layer size = 900, batch size = 40 and epochs = 10, the result is as following, Then, from

Table 3: Regularization Strength

Regularization	1e-1	1e-2	1e-3	1e-4	1e-5	1e-6	1e-7	0
Accuracy	0.0967	0.0989	0.7724	0.9803	0.9832	0.982	0.9829	0.9834

Table 3 we know that **the optimal regularization strength is 0 with corresponding accuracy is 0.9834**.

Finally, we conclude that the optimal parameter of our model is **learn rate = 0.03125, hidden layer size = 900, regularization strength = 0**. Then we will use these parameters to make the visualization.

4 Visualization

In this section, we will use the optimal parameters: learn rate = 0.03125, hidden layer size = 900, regularization strength = 0, batch size = 40 and epochs = 10. Moreover, we use MSE as the loss function, ReLU as the first activation function.

Firstly, we propose the figure of the training loss, validation loss and accuracy as following (Figure 3), Then, we get the trained model at optimal parameters, and you can find it in Baidu Netdisk[3] with

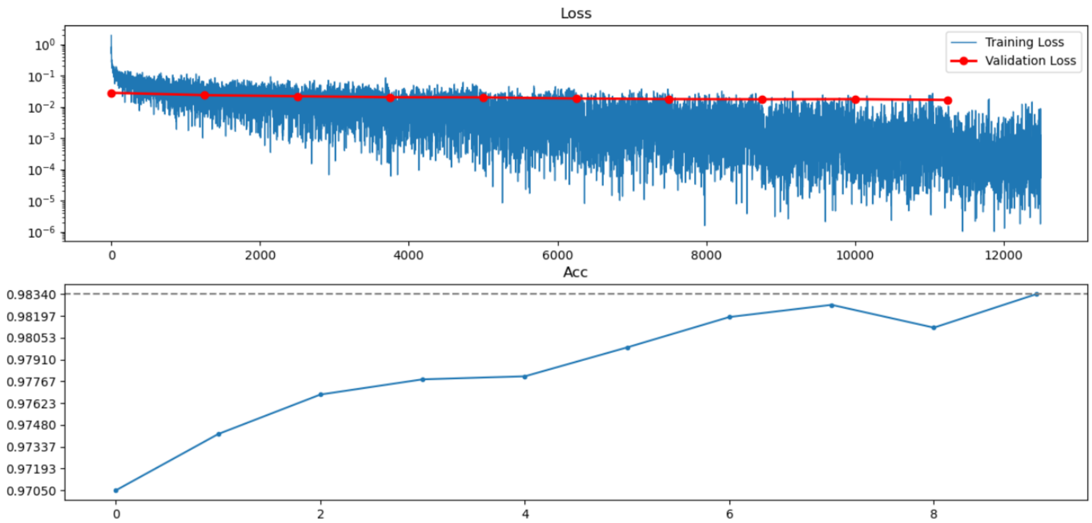

Figure 3: Model of Optimal Parameters

fetch code is {t5jp}.

Moreover, this model have accuracy 99.996% **on train data,** 98.34% **on validation data,** 98.38% **on test data**. So we know that we have accuracy 98.38% on handwriting dataset of MNIST.

Although our model have a high accuracy, we may fail in some datasets, there are still shortages of our model.

5 Conclusion

In this paper, we build a neural network two layers classifier only using Numpy. We use different parameters to seek the optimal parameters: learn rate, hidden layer size and regularization strength. The trained model have an accuracy 98.38% on handwriting dataset of MNIST. Although we have a high accuracy, there are also some shortages can be avoided in our model, such as, more intensive parameters search and another adjustment technique.

[3] https://pan.baidu.com/s/1hzkMCHe8OqT3ZuXBRyiuug