UM–SJTU Joint Institute

Fall 2018 VE593 Problem Solving with AI Techniques

Project 3 Report

Xinhao Liao

516370910037

Contents

1 MNIST Implementation Summary

The MNIST data set is going to be tested with the ANN network with different activation functions, with L2 regularization and early stopping applied. The parameters will be slightly modified for each activation function to obtain the best accuracy tested with the validation data. And finally, with the modified parameters for each function, the accuracy tested with the test data with each activation function will be measured and compared.

The ANN architecture is initially set to be [784,50,10]. Since one test image consists of 28*28=784 pixels as input, the input layer should have 784 neurons. The output layer of the ANN should contain 10 neurons to tell us which digit (0, 1, 2, 3, ..., 9) corresponds to the input image. Each neuron with index i from 0 to 9 gives a float in [0,1] which tells us how much it believes that the input image corresponds to the digit i. As for the hidden layers, the size is initially set to be 1 layer with 50 neurons.

For each activation function, before modifying the parameters, we can obtain a best range for the values of parameters with some quick test. The method is to use a network of the size [784,10] and quickly test the accuracy with only part of training data and part of validation data. This can minimize the time cost for testing. Based on the results, we can adjust the parameters. This process is omitted in this report.

Having obtained the rough values of the parameters, we still need to carefully modify them for the best results. The process is explained in the following sections.

Notice that the RELU function can easily bring *nan* and *inf* to the Numpy array and lead to problems, the learning rate need to be small enough to avoid problems. Also, in the implementation of RELU, a small constant 0.01 is used instead of 0 to avoid problems.

In *experiments.py*, the codes for the final comparison is given. The ANN networks, with the same initially randomly initialized weights and biases distributions and different activation functions, is trained with the selected parameters. The training accuracy in every iteration is shown, and the final test accuracy and time cost is also presented.

2 MNIST Activated with the Sigmoid Function

Initially we need to get a range of potential values of the parameters that is likely to produce good results. We can do this by constructing a network with no hidden layers, constantly modify the parameters, and try to quickly train and test it with only part of part of the dataset.

After some try, the parameters are initially set to be $T = 10$ indicating the maximum iteration number (noticing that early stopping is applied so it's likely to stop in advance once the accuracy drops to avoid over-fitting and save time), $n = 10$ indicating the size of the mini-batch, , $alpha = 1$ which is the learning rate, and $\lambda = 0.001$ for L2 regularization.

2.1 λ determination

Other parameters remain the same, and we adjust λ to be 0.000001,0.00001, 0.0001, and 0.001. 5 randomly initialized networks are used to check for the different λ. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below.

λ	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
0.000001	9496.8	7.6	87.84
0.00001	9468.2	6.8	77.32
0.0001	9503.4	4.2	46.35
0.001	9225	3	34.13

As we can see, when λ is set to be 0.0001, the results have the best final accuracy. So we set $\lambda = 0.0001$.

2.2 Learning Rate determination

Other parameters remain the same, and some are chosen. For the learning rate, if it's too small, the accuracy will grow slowly, which cost a lot of time for training. But if it's too big, the weights may fail to converge to the best values. And we adjust the learning to be 0.25, 0.5, 1, 1.5, and 2. 5 randomly initialized networks are used to check for the different learning rate. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below.

Learning Rate	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
0.25	9076.4	7.2	82.76
0.5	9582.8	6.2	67.50
1	9502.3	4.6	49.62

Considering the time cost the and the final accuracy, as we can see, the learning rate of 0.5 gives rise to the best performance. So alpha is set to be 0.5.

2.3 Batch Size determination

We have determined λ to be 0.0001 and the learning rate to be 0.5. This time we adjust the batch size n. 5 randomly initialized networks are used to check for the different n. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below.

n	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
5	9564.6	5.4	61.15
10	9591.4	6.6	70.21
20	9544	7	77.22

Considering the cost of the and the final accuracy, as we can see, the batch size 10 gives rise to the best performance.

3 MNIST Activated with the *tanh* Function

Initially we need to get a range of potential values of the parameters that is likely to produce good results. We can do this by constructing a network with no hidden layers, constantly modify the parameters, and try to quickly train and test it with only part of part of the dataset.

After some try, the parameters are initially set to be $T = 30$ indicating the maximum iteration number (noticing that early stopping is applied so it's likely to stop in advance once the accuracy drops to avoid over-fitting and save time), $n = 15$ indicating the size of the mini-batch, , $alpha = 0.01$ which is the learning rate, and $\lambda = 0.01$ for L2 regularization.

3.1 λ determination

And we adjust λ to be 0.006, 0.01, 0.03, and 0.05. 5 randomly initialized networks are used to check for the different λ. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below.

λ	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
0.006	9216.4	13.4	145.23
0.01	9202.6	10.4	112.93
0.03	9213	5.6	62.15
0.05	9148	5.6	67.32

Considering the final accuracy and the time cost, we set λ to be 0.03.

3.2 Learning Rate determination

For the learning rate, if it's too small, the accuracy will grow slowly, which cost a lot of time for training. But if it's too big, the weights may fail to converge to the best values. And we adjust the learning to be 0.001, 0.0025, 0.005, 0.01, and 0.02. 5 randomly initialized networks are used to check for the different learning rate. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below.

Learning Rate	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
0.001	9204	30	327.54
0.0025	9248.5	16.4	177.10
0.005	9234.2	9.2	100.90
0.01	9205	5.4	64.34
0.02	9133.4	3.6	39.54

Considering the time cost the and the final accuracy, alpha is set to be 0.005.

3.3 Batch Size determination

We have determined λ to be 0.03 and the learning rate to be 0.005. This time we adjust the batch size n. 5 randomly initialized networks are used to check for the different n. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below.

Considering the cost of the and the final accuracy, as we can see, the batch size of 25 gives rise to the best performance.

n	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
5	9110.4	5.2	56.13
10	9180	7	75.22
15	9242.6	9.2	101.12
20	9256	11.2	120.51
25	9260.4	12	128.47
30	9184	12	132.65

4 MNIST Activated with the RELU Function

Initially we need to get a range of potential values of the parameters that is likely to produce good results. We can do this by constructing a network with no hidden layers, constantly modify the parameters, and try to quickly train and test it with only part of part of the dataset.

After some try, the parameters are initially set to be $T = 30$ indicating the maximum iteration number (noticing that early stopping is applied so it's likely to stop in advance once the accuracy drops to avoid over-fitting and save time), $n = 20$ indicating the size of the mini-batch, , $alpha = 0.01$ which is the learning rate, and $\lambda = 0.01$ for L2 regularization.

4.1 λ determination

And we adjust λ to be 0.001, 0.01, and 0.1. 5 randomly initialized networks are used to check for the different λ. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below.

λ	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
0.001	3420	5.6	64.65
0.01	9517.4	15.2	174.28
0.1	8799.2	5.6	64.11

Considering the final accuracy and the time cost, we set λ to be 0.01.

4.2 Learning Rate determination

For the learning rate, if it's too small, the accuracy will grow slowly, which cost a lot of time for training. But if it's too big, the weights may fail to converge to the best values. And we adjust the learning to be 0.002, 0.005, and 0.01. 5 randomly initialized networks are used to check for the different learning rate. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below.

Learning Rate	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
0.002	9543	48	547.20
0.005	9568.4	23.8	278.27
0.01	9512	15.4	170.12

Considering the time cost the and the final accuracy, alpha is set to be 0.005.

4.3 Batch Size determination

We have determined λ to be 0.01 and the learning rate to be 0.005. This time we adjust the batch size n. 5 randomly initialized networks are used to check for the different n. The time cost and final accuracy are measured given the validation data set. The average of the measured results are shown below. Considering the time cost the

n	Final Accuracy (/10000)	Iteration Numbers	Total Time Cost (s)
2	9264.4	6.4	75.17
5	9549.8	10.8	124.66
10	9552	15.4	169.82
20	9566.4	23.2	274.22

and the final accuracy, the batch size is set to be 5.

5 Comparison

In this section, we use the obtained parameters to test the accuracy and time cost given the test data, and compared the results. The codes for this comparison is given in *experiments.py*. And it has been run 5 times for an average result. The result is given below.

Activation Function	Training Accuracy	Test Accuracy (/10000)	Time Cost (s)
Sigmoid	9521.8	9482.6	45.60
tanh	9292	9250	147.02
RELU	9554	9484.4	127.99

As we can see, Sigmoid function requires the least time cost and RELU function has the highest accuracy.