Report for Artificial Neural Network HW1

Jiayi Weng
Department of Computer Science
Tsinghua University
wengjy16@mails.tsinghua.edu.cn

Abstract

In HW1 we need to complete the Python code provided by TA. We use this code to run a multi-layer perception network to deal with MNIST dataset. We are required to finish different layers' forward and backward code, modify the the loss function and hyperparameters, and designed the network architecture by ourself. Also, there should be a compare with Sigmoid and ReLU layer. **It seems like the SGD+Momentum code written by TA is wrong.**

1 Network Architecture

In the following network designs, **all** of the parameters regarding to the SGD are set as:

- weight_decay: 5×10^{-3}
- momentum: 0.9

The total iteration in **all** network is equal to 12000, and the standard deviation in Linear layer is set to be 10^{-3}.

1.1 Affine Network

This network only includes nothing but a single Linear layer. The learning rate is equal to 10^{-5}, batch size is 50. Detailed network architecture and configuration can be found in line 34-46 of `run_mlp.py`.

1.2 Single Hidden Layer Network with Sigmoid Activation

The learning rate is equal to 10^{-2}, batch size is 100. Detailed network architecture and configuration can be found in Table 1 and line 48-63 of `run_mlp.py`.

Table 1: Single Hidden Layer Network with Sigmoid Activation

Layer	Type	In	Out
1	Linear	784	256
2	Sigmoid	256	256
3	Linear	256	10
4	Sigmoid	10	10

1.3 Single Hidden Layer Network with ReLU Activation

The learning rate is equal to 10^{-4}, batch size is 200. Detailed network architecture and configuration can be found in Table 2 and line 65-80 of `run_mlp.py`.

Table 2: Single Hidden Layer Network with ReLU Activation

Layer	Type	In	Out
1	Linear	784	256
2	ReLU	256	256
3	Linear	256	10
4	ReLU	10	10

1.4 Two Hidden Layers Network with Sigmoid Activation

The learning rate is equal to 10^{-2}, batch size is 300. Detailed network architecture and configuration can be found in Table 3 and line 82-99 of `run_mlp.py`.

Table 3: Two Hidden Layers Network with Sigmoid Activation

Layer	Type	In	Out
1	Linear	784	256
2	Sigmoid	256	256
3	Linear	256	128
4	Sigmoid	128	128
5	Linear	128	10
6	Sigmoid	10	10

1.5 Two Hidden Layers Network with ReLU Activation

The learning rate is equal to 10^{-4}, batch size is 300. Detailed network architecture and configuration can be found in Table 4 and line 101-118 of `run_mlp.py`.

Table 4: Two Hidden Layers Network with ReLU Activation

Layer	Type	In	Out
1	Linear	784	256
2	ReLU	256	256
3	Linear	256	128
4	ReLU	128	128
5	Linear	128	10
6	ReLU	10	10

2 Experiment Result

All of these experiments are conducted on my laptop, with 4 Intel i5-5200U 2.20GHz CPUs.

2.1 Loss

The loss in training procedure is illustrated in Figure 1, and the numeric result of training and testing loss is in Table 5.

2.2 Accuracy

The accuracy in training procedure is illustrated in Figure 2, and the numeric result of training and testing accuracy is in Table 5.

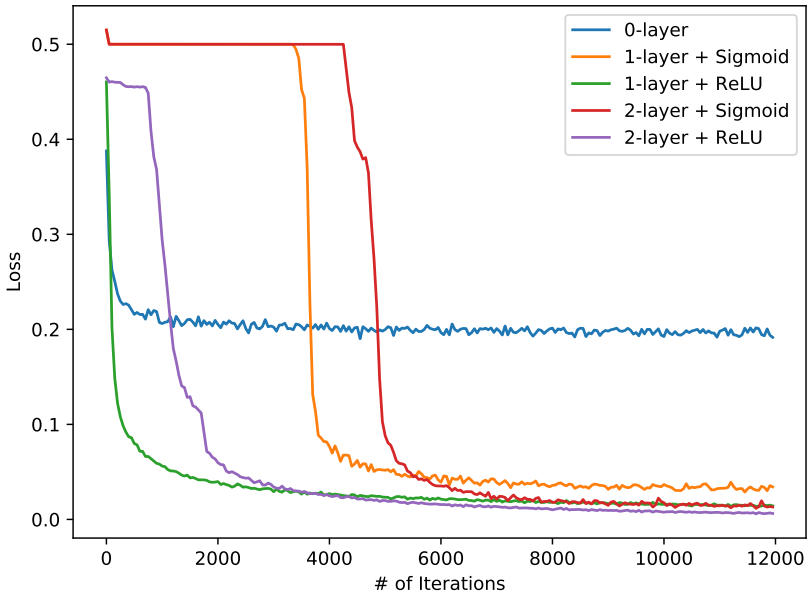

Figure 1: Training loss of different network.

Table 5: Training/Testing Loss/Accuracy after 12000 Iterations

Metric	0-layer	1-layer + Sigmoid	1-layer + ReLU	2-layer + Sigmoid	2-layer + ReLU
Train Loss	0.19015	0.02873	0.01364	0.01166	**0.00585**
Train Acc.	87.00%	97.62%	99.28%	98.92%	**99.61%**
Train Time	**11.283s**	138.210s	164.739s	309.243s	275.991s
Test Loss	0.19417	0.03224	0.02306	0.01759	**0.01675**
Test Acc.	86.33%	96.77%	**98.13%**	98.01%	98.05%

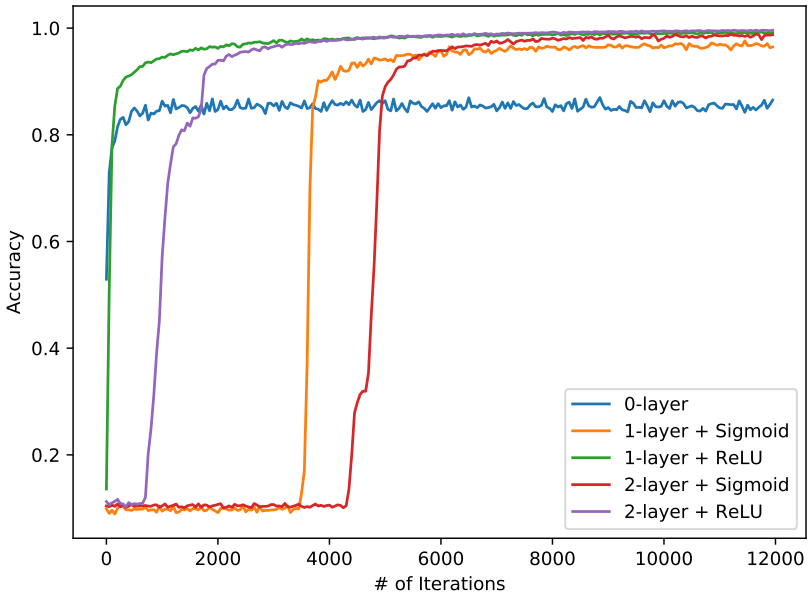

Figure 2: Training accuracy of different network.

2.3 Analytics

In single layer network, ReLU activation is much better than Sigmoid activation both in converge speed and accuracy. ReLU's time is longer than Sigmoid's due to the different batch size. In two-layer network, we can see ReLU is much better than Sigmoid in all metrics when in the same configuration.

The Sigmoid activation may cause the problem of gradient vanish. From Figure 1 and Figure 2, the network with Sigmoid activation in the beginning of training procedure is stuck on a platform, however the ReLU activation does not have this kind of disadvantage. In the last few steps, Sigmoid activation can not achieve the same performance of ReLU. I think it is caused by gradient vanishing problem, thus the model get stuck on a local minimum.

As for the computational consumption, the ReLU only needs superfast $\max$ operation. But Sigmoid needs $\exp$ and $\div$ operation, which is very slow. This is also happened on back propagation.

3 Question on SGD+Momentum Code

See suppmental material `readme.md` for more details.