Deep Learning
Assignment 2 Report

Tsinghua University

Albert Millan – 2019280366

Contents

1 Introduction

This report explains the setup and methodology involved to deploy a) softmax classifier on the MNIST dataset and b) a multilayer-perceptron employing sigmoid or ReLu activation functions, as well as softmax cross-entropy loss or euclidean loss functions.

2 Softmax Classifier

Upon successful implementation of the loss function and gradient computation, a accuracy on the test set of 79% was obtained with the provided hyperparameters. The model was further tuned, achieving a final accuracy of 91%. Table 1 outlines the values of the hypermarameters employed to obtain such result. Significant improvement was obtained upon reducing the regularization constant, increasing the learning rate and the number of training iterations. The loss fluctuates around 0.4 after 5000 batch iterations, while the accuracy reaches above 90% accuracy withint the initial 5000 batch iterations.

Type	**Initial Value**	**Tuned Values**
Batch size	100	100
Learning rate	0.001	0.004
Weight decay	0.5	1e-5
Max epoch	20	60

Table 1: Hyperparameters employed to train the model.

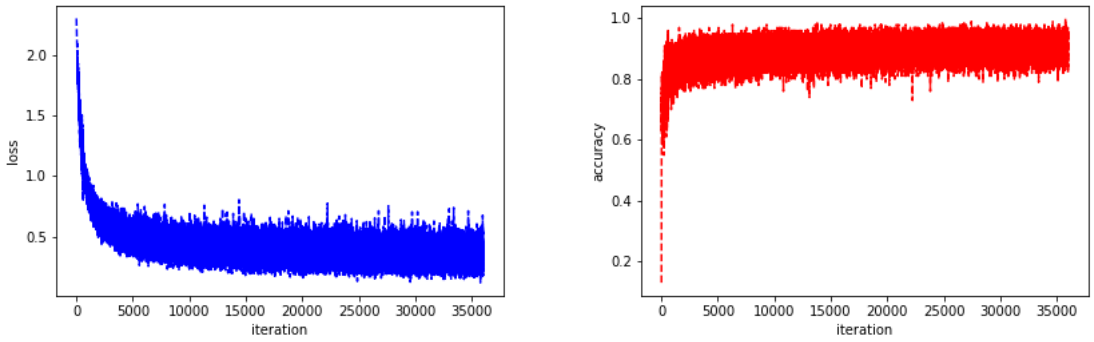

Figure 1: Softmax Classifier loss and accuracy over epochs.

3 Euclidean Loss MLP

Upon completing the implementation of the euclidean loss with both *Sigmoid* and *ReLu* activation functions, the accuracy obtained rounded to 84% and 92%. It appears that ReLu yields a superior performance under the initial parameters. Further tuning was performed, enhancing the accuracy of the Sigmoid and ReLu implementations to 94.02% and 96.66% respectively. While the improvement in the accuracy of the Sigmoid implementation was larger than that of the ReLu, the latter still outperformed the former. The rate of improvement in accuracy is highlighted in figure 2, comparing the accuracy and loss across implementations. The hyper parameters for the initial and final results for both implementations are provided in table 2.

Type	**Initial Value**	**Tuned Values**
Batch size	100	100
Layer size	128	128
Learning rate	0.001	0.005
Weight decay	0.1	0.001
Init std	0.01	0.01
Max epoch	20	40

Table 2: Hyperparameters employed to train the model.

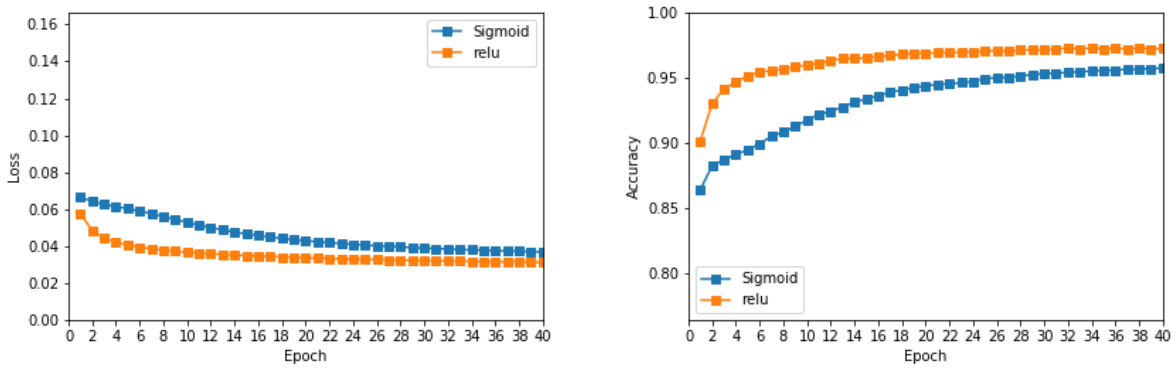

Figure 2: Loss and accuracy for Sigmoid and ReLu activation layers in Euclidean loss implementation of MLP.

4 Softmax Cross-Entropy Loss MLP

Upon completing the implementation of the softmax cross-entropy loss with both *Sigmoid* and *ReLu* activation functions, the accuracy obtained rounded to 31.42% and 92.71%. It appears that ReLu yields a superior performance under the initial parameters. Sigmoid implementation appears to overshoot the optima with the initial parameters. Further tuning was performed, enhancing the accuracy of the Sigmoid and ReLu implementations to 90.48% and 94.51% respectively. While the improvement in the accuracy of the Sigmoid implementation was larger than that of the ReLu, the latter still outperformed the former. The rate of improvement in accuracy is highlighted in figure 3, comparing the accuracy and loss across implementations. The hyper parameters for the initial and final results for both implementations are provided in table 3.

Type	**Initial Value**	**Tuned Values**
Batch size	100	100
Layer size	128	128
Learning rate	0.001	0.004
Weight decay	0.1	1e-5
Init std	0.01	0.01
Max epoch	20	40

Table 3: Hyperparameters employed to train the model.

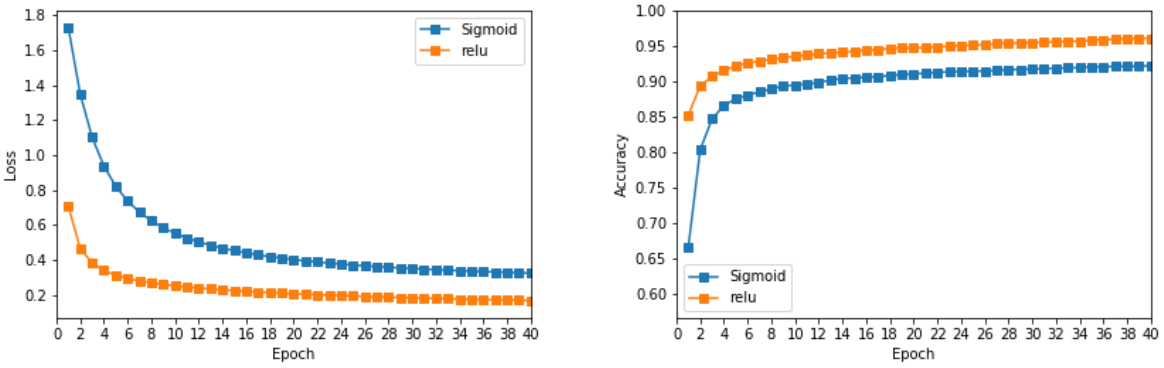

Figure 3: Loss and accuracy for Sigmoid and ReLu activation layers in Softmax cross-entropy loss implementation of MLP.

5 Two Hidden Layer MLP

Following the modular architecture employed in the previous sections, a MLP containing two hidden layers was implemented. The layers architecture involves a FCLayer + ReLu + FCLayer + ReLu + FCLayer + Softmax. Layers were stacked on top of each other. The accuracy obtained on the test set after training is 96.56%. The hyperparameters are displayed in table 4 and the rate of change of loss and accuracy per epoch are shown in figure 4.

Type	**Initial Value**	**Tuned Values**
Batch size	-	100
Layer size(1)	-	128
Layer size(2)	-	64
Learning rate	-	0.005
Weight decay	-	1e-3
Init std	-	0.01
Max epoch	-	50

Table 4: Hyperparameters employed to train the model.

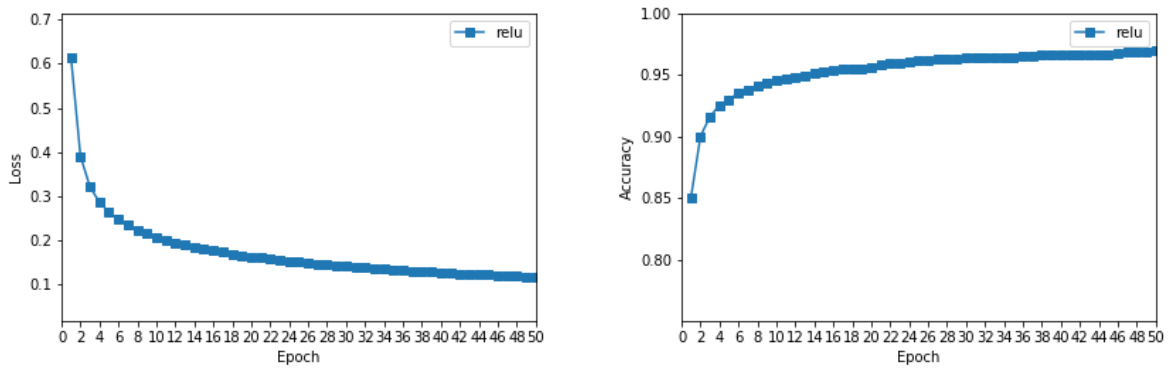

Figure 4: MLP training loss and accuracy.