Multi-layer Neural Networks on Fashion MNIST Dataset

Fangzhou Ai
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
faai@eng.ucsd.edu

Yue Qiao
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
yuq021@eng.ucsd.edu

Zunming Zhang
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
zuz008@eng.ucsd.edu

Abstract

This paper is a report for the programming assignment 2 for Lecture CSE 251B. In this report we build a simple neural network only relies on simple libraries like Numpy without any higher level machine-learning libraries invoked. This naive neural network can support multiple fully-connect layers and within each layer we can implement different activation functions, moreover, we also support regularization to prevent over-fitting from happening. We apply our model to the Fashion MNIST data-set and achieve expected precision. In our experiments, it shows that with 0.001 L2 penalty and using ReLU as activation function, the network has best performance.

1 Gradient check

2 Implment the model

We implemented our model on the Fashion MNIST dataset, we have 2 hidden layers here and each of them contains 50 nodes, we apply the tanh activation function for those layers and softmax at the output layer, we spited the training set into 80/20 train/validation set, and did cross-validation making sure each part of data would be at least be the validation set for 1 time, our batch size was 256, epochs limit was 300 here and the learning rate is 0.05, we also imposed a early stop limit on our training, once the validation set error kept going up for more than 10 epochs, we stopped training prcedure and recall the best parameters we saved before during the lowest validations set error, at the end of all trainnigs, we achieved around 76.37% accuracy. The loss and error curve are shown in Figure 1.

Preprint. Under review.

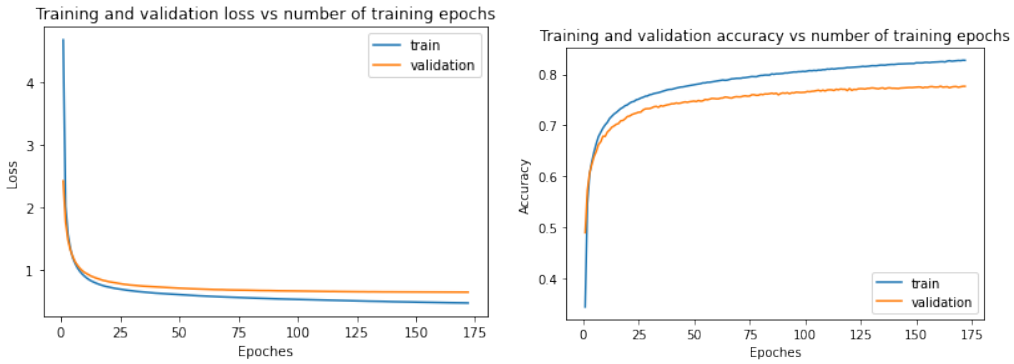

Figure 1: Traning and validation loss and accuracy for our model.

3 Experiment with regularization

We tried three different L2 penalty value: 0.01, 0.001, 0.0001. After all, we find that 0.001 is the best value for our model. The loss and accuracy can be found in Figure 2. The test set accuracy can be found in Table 1. We also find that with a smaller regularization value, more epoches are needed for converge.

Table 1: Test set accuracy for different regularization values

L2 penalty	Test set accuracy
0.01	0.7534
0.001	**0.8560**
0.0001	0.8027

4 Experiment with activations

We tried four different activation functions: sigmoid, tanh, ReLU and leakyReLU. Sigmoid is a nonlinear function that helps the network archives nonlinearity. However, it tend to respond very less to changes of X and has a problem of vanishing gradients. The derivatives of tanh are steeper than The derivatives of sigmoid, but it also has the vanishing gradients problem. ReLU does not have vanishing gradient problem and is simplier to calculate. However, since the gradient of ReLU for negative values are zero, some neurons will stop responding to the error. LeakyReLU tries to fix the dead neurons problem, but since it introduces some kinds of linearity, it cannot be used on complex networks. In our model, ReLU works best. LeakyReLU has similar performance compared to ReLU since our network is not complex enough. The loss and accuracy can be found in Figure 3. The test set accuracy can be found in Table 2.

Table 2: Test set accuracy for different activations

Activation	Test set accuracy
sigmoid	0.7884
tanh	0.8560
ReLU	**0.8634**
LeakyReLU	0.8623

5 Individual contributions to the project

Fangzhou Ai Part (c)

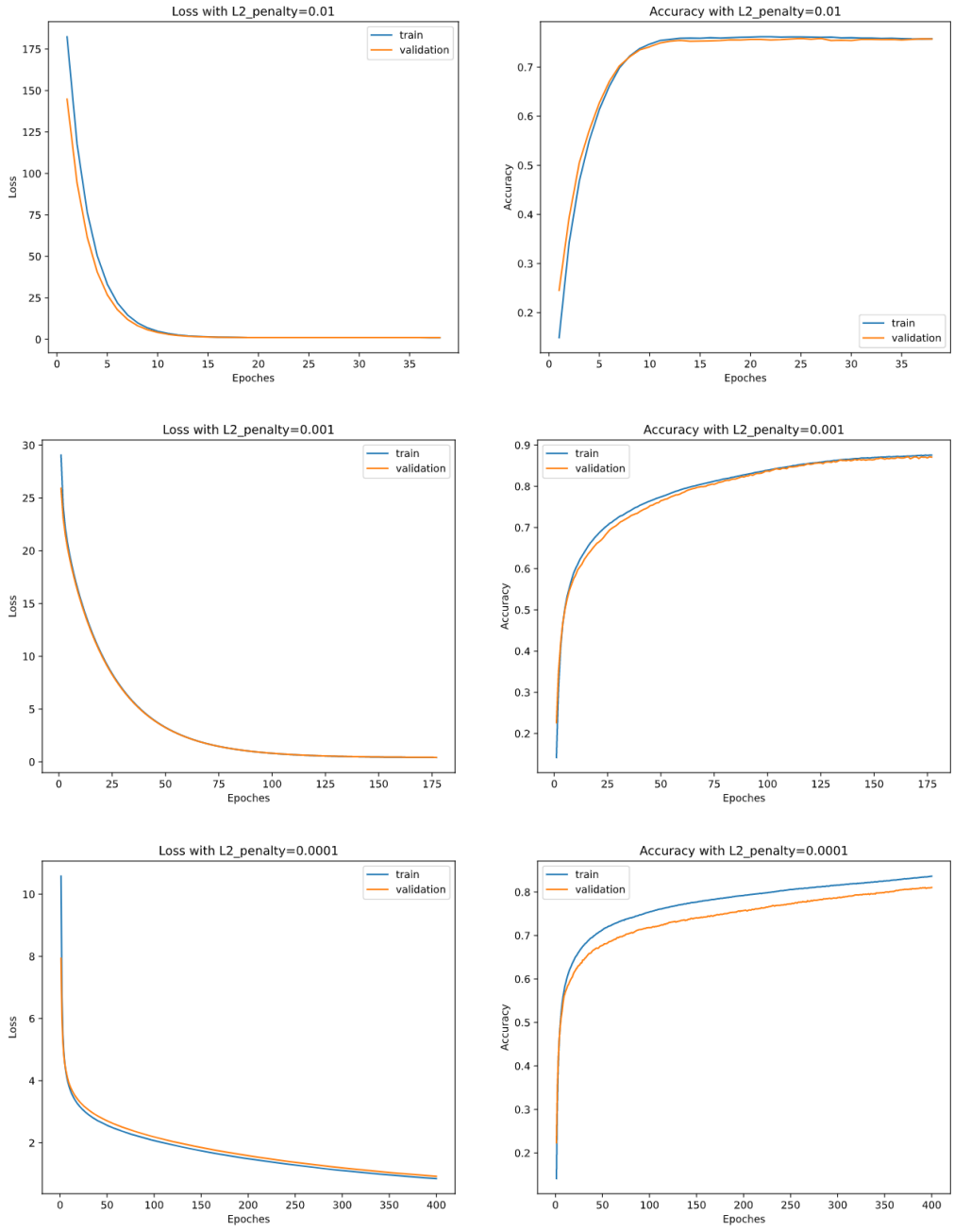

Figure 2: Traning and validation loss and accuracy for different regularization values

Yue Qiao Experiment with Regularization and Experiment with Activations.

Zunming Zhang

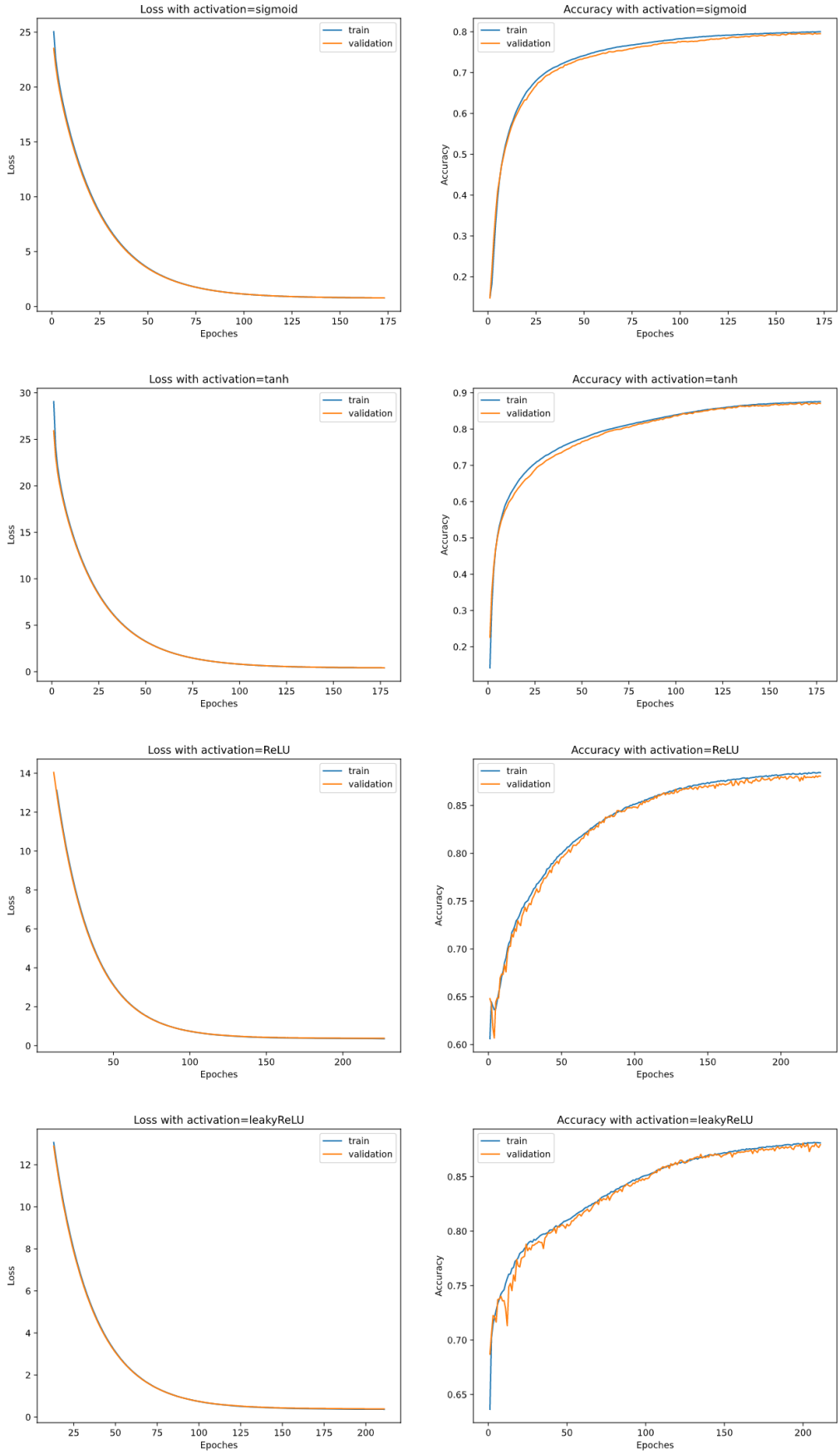

Figure 3: Traning and validation loss and accuracy for different activations