Handwriting Recognition With the MNIST Database Report

By Wai Kwan Shum

December 6, 2017

Abstract

By using the MNIST dataset, I was able to investigate backpropagation learning algorithm and techniques such as early stopping to achieve better performance from the neural network. I found out that incremental training gave better performance than mini-batch training. MSE and accuracy do not necessarily agree on the choice of optimal model. Randomizing training data per epoch seemed to affect the network performance positively.

Introduction

The goal of this assignment was for us to apply the ANN architectures, learning algorithms and machine learning techniques we have acquired in class on the widely used and preprocessed MNIST dataset. Since the dataset is put together very well, instead of spending time to processing the data (e.g. normalization), I was able to focus on investigating the performances achieved by various learning algorithms and performance improvement techniques. I investigated how varying the number of neurons (network complexity) and training with incremental and mini-batch methods affect the performance of the network.

Methods

I used backpropagation with one hidden layer to train the network. Log-sigmoid transfer function was used in both hidden layer and output layer; sigmoid functions are often used in pattern recognition problems, and multiple layers networks take advantage of nonlinear transfer functions (such as log-sigmoid) to learn nonlinear relationships between the input and output.

Since the target outputs (labels) from the MNIST dataset are literal digits from 0 to 9, I transformed the digits 0 to 9 into a 10 by 10 identity matrix for the network training. That was also the reason why I chose log-sigmoid as it is a nonlinear transfer function that gives output values in [0,1].

In the MNIST dataset, there are 60,000 data in the training set and 10,000 data in the testing set. I took out 10,000 data (~15%) from the training set as validation set. I also used early stopping for improving generation during training; I evaluated the network performance using the validation set after each epoch, and stopped the training when the mean square error (MSE) of the validation set increased three times consecutively. Although overfitting is very unlikely to happen in this dataset as the data are uniformly distributed in the both training set and testing set, early stopping prevents increasing the complexity of the resulted network unnecessarily.

I mainly used incremental training to train the network with various number of neurons. I also trained the network by using the same set of training data and random sets of training data in each epoch.

I briefly used mini-batch training to train the network with various batch sizes with the same given iterations.

To calculate the accuracy on testing set, I checked the output vector (10 by 1) to see which index has the highest value, if that index in the target output vector also has the highest value (which is 1), that is considered a match. Accuracy = number of matched / total number of testing data.

Results

Same dataset in each epoch

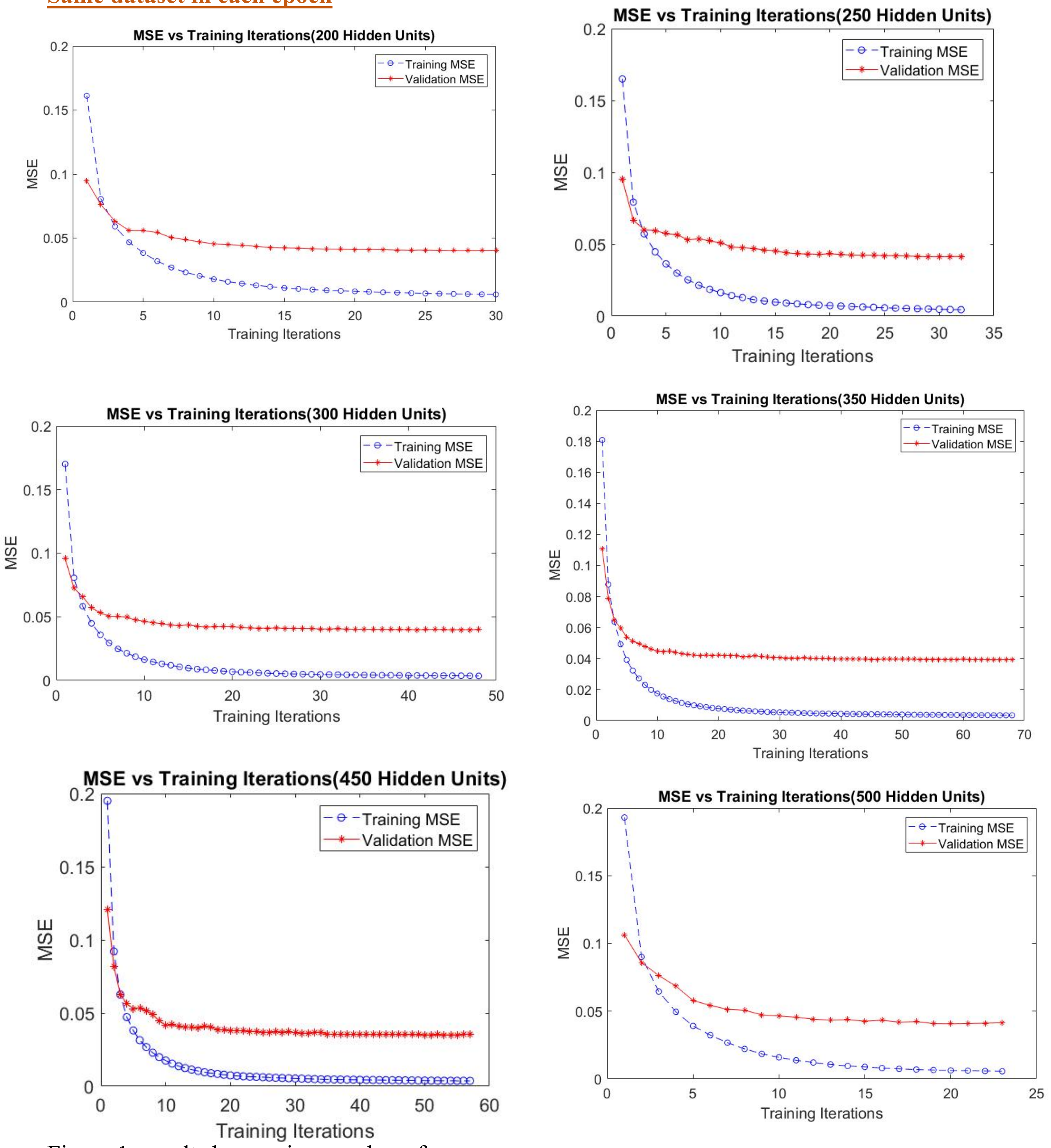

Figure 1: results by varying number of neurons

Random dataset in each epoch

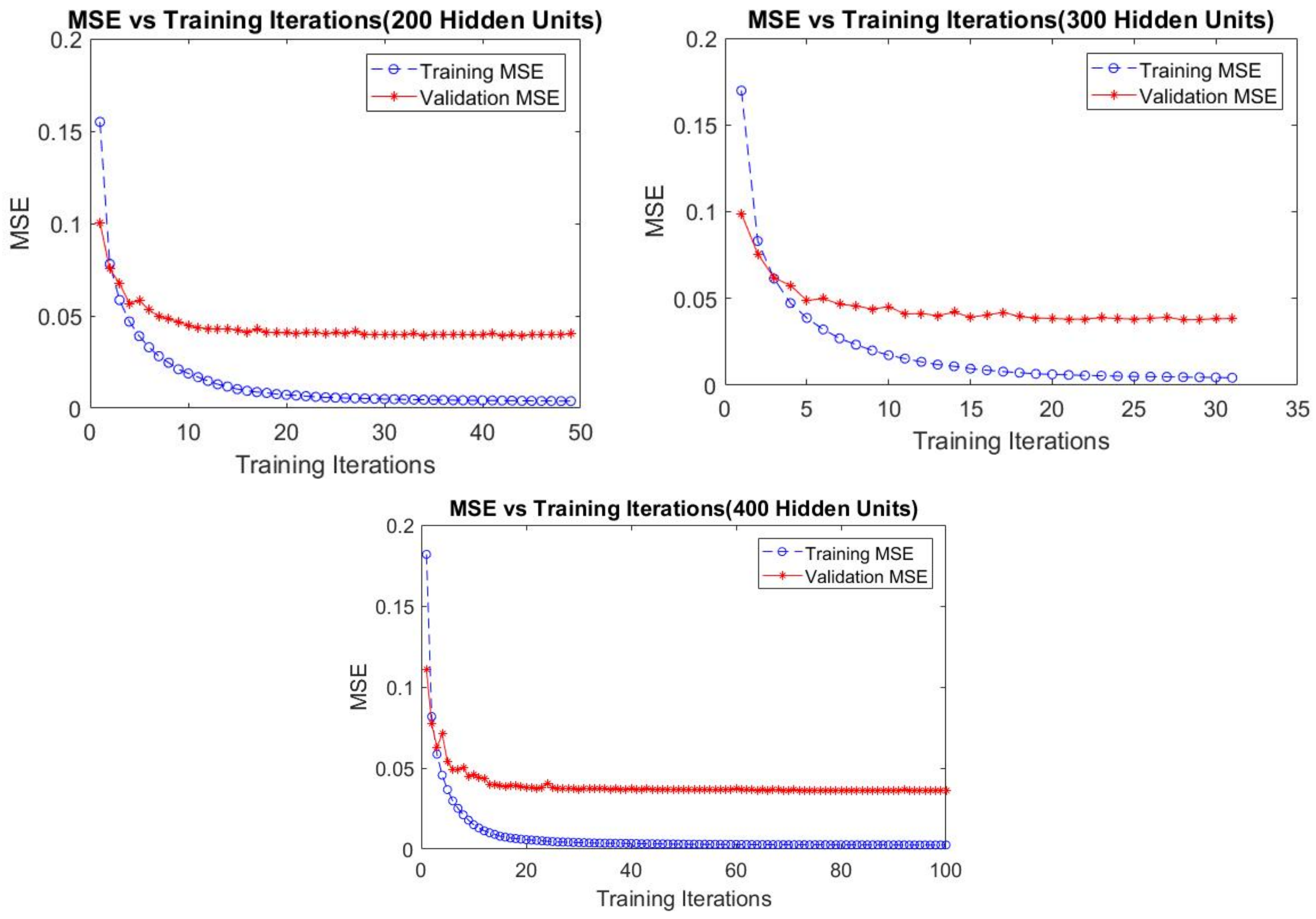

Figure 2: results by varying number of neurons (random dataset)

From testing set	Number of Neurons						
	200	250	300	350	400	450	500
MSE	4.17%	4.12%	3.78%	3.74%	3.72%	3.69%	3.64%
MSE(random)	4.01%		3.7%		3.54%		
Accuracy	97.86%	97.86%	98.05%	98.03%	98.09%	98.05%	98.07%
Accuracy(random)	97.99%		98.12%		98.14%		

Figure 3: MSE and accuracy with various number of neurons (incremental training)

From testing set	Batch size (using 300 neurons)		
	100	1000	10000
MSE	12.21%	23.71%	74.24%
Accuracy	93.43%	87.52%	41.12%

Figure 4: MSE and accuracy with various batch sizes (mini-batch training)

Observation

As shown in figure 1 and figure 2, the training set MSEs start steadily converging to zero at around the 20th iteration and the validation set MSEs moving around the 4% - 5% range at around the 10th iteration.

From figure 3, we can see that the training set MSEs are lower when datasets are randomized in each epoch than the corresponding ones when the dataset stays the same. This phenomenon applies to the accuracy as well.

From figure 4, we see that the network performs better when the batch size is smaller.

Conclusion

Although mini-batch training took less time than incremental training, the incremental training performed better on the testing set. I believe that it was due to weights and biases were updated more frequently in incremental training, all inputs contributed to the weights and biases update; in other words, the network learnt more frequently on different patterns compared to the mini-batch training.

Testing set MSEs got lower when the number of neurons used was higher, it was because higher number of neurons covered more possible input patterns than lower number of neurons in the network; in other words, the network had better prediction on testing data. However, lower MSE does not necessarily mean higher accuracy (as shown in figure 3).

Randomizing the training dataset during training in each epoch seemed to have better performance on testing data. Essentially the training set is the same but they are in different order during training, but the network might possibly be adapted to randomness by training it with random orders of data (better generation in a sense) since testing data is totally new (random) to the network.

For mini-batch training, training with smaller batch size gave better performance on the testing data, it was due to the fact that the weights and biases got updated more frequently per epoch than bigger batch size; the network learnt better with smaller batch size in the given number of iterations.

Bibliography

Hagan, M., Demuth, H., Beale, M. and De Jesús, O. (2016). *Neural network design*. [S. l.: s. n.].

Ufldl.stanford.edu. (2017). *Using the MNIST Dataset - Ufldl*. [online] Available at: http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset [Accessed 6 Dec. 2017].

Yann.lecun.com. (2017). *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. [online] Available at: http://yann.lecun.com/exdb/mnist/ [Accessed 6 Dec. 2017].