MLP with MNIST

Juliane Weilbach

30 October 2018

1 Introduction

This exercise is about a classification using a multilayer perceptron (MLP) with a handwritten digit dataset, called MNIST. The implementation was done with a python template of the University of Freiburg.
The activation functions for the MLP were a hyperbolic tangent function, a sigmoid function or a rectified linear unit (RELU). The neural net uses a softmax function and a one-hot-encoding.

2 Results

2.1 First Neural Net Approach

First the preimplemented net contained [1] three fully connected layers. The first two layers were Relu's with 100 nodes, the last one is without an activation function and with 10 nodes. After 20 epochs of training the validation error is 0.0278 and the test error 0.0267, which is pretty good. To decrease the validation and test error seen in figure 1, it is necessary to change the neural net structure.

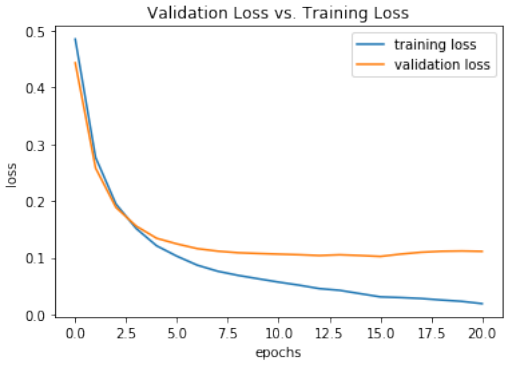

Figure 1: Validation vs. Training Loss

[1] batch-size:64, learning-rate:0.1, descent-type:sgd

2.2 Optimized Neural Net

For optimizing the first neural net, now the net [2] contains four layers. The first layer has a Relu activation, the second a hyperbolic tangent function, the third again a Relu activation and the last has no activation. All layers, except of the last layer with 10 nodes, including 150 nodes. Furthermore slightly different hyperparameters were used in this neural net, like bigger batch size of 128 and a higher learning rate of 0.15.

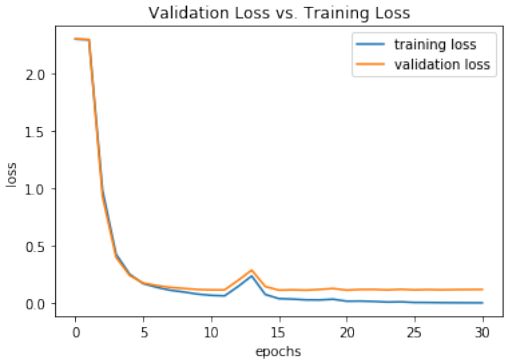

Figure 2: Validation vs. Training Loss optimized Neural Net

After 30 epochs of training the validation error decreases to 0.0227 and the test error to 0.0236.(Figure 2)
To reduce the overfitting of both nets, it would be possible to use e.g. early-stopping.

[2] nn2 called in the implementation