Faculty of Engineering
Albert-Ludwigs-Universitat
Autonome Intelligente Systeme - AIS
Wintersemester 2017/18

Deep Learning Lab Course 2017

Assignment 1

Author:	Markus Merklinger
E-mail:	merklingermarkus@gmail.com
Tutors:	Aaron Klein, Artemij Amiranashvili, Mohammadreza Zolfaghari, Maria Hgle, Jingwhei Zhang, Andreas Eitel
Data of submission:	November 6, 2017

Deep Learning Lab Course 2017
Assignment 1

November 6, 2017

Abstract

In this Assignment we studied a fundamentals of deep learning. With the implement and training supervised feed-forward neural network the underlying conspectus can be fully understand. In addition the with Gradient Checking it could be shown that the implementation of the neural network was correct. By training the Neural network on the MNIST dataset the importance and difficulty of choosing good the hyper parameter optimization was realized. With a hyper parameter optimization with Tree of Parzen Estimators good parameters were found for the MNIST dataset.

Contents

1 Implementation

The implementation of the MLPs (feed-forward neural networks) is divided in two main parts: a Layers and Neural Network class. The Neural Network class can contain different layers and computes the forward backward passes and loss of the layers form weight updates while training. The wights are updated with Gradient descent or Stochastic Gradient descent (SGD).

With gradient checking the functionality of the implementation could be checked. The Table 1 shows the result of the Gradient Checking for random data and three fully connected layer with relu, tah as activation.

Table 1: Results Gradient Checking for three fully connected layer

	error 1	error 1
layer 1	6.76e-09	6.74e-09
layer 2	1.65e-08	1.43e-07
layer 3	2.82e-08	3.49e-06

2 Evaluation and Results

Now the network implementation is evaluated on the MNIST dataset. A classification task perform will be performed . The dataset consists of images of handwritten digits and a label for the corresponding number. After shuffling the data and using different parameters for the learningrate activation function, number of fully connected layers and hidden units and number of epochs. The performance of the trained network is measured on validation data which are not used for training. The most important one in the learning rate in combination with the layers and hidden units. Same learning rates with different models are leading to different performances on the validation set and vice versa. The Figure 1 and Figure 2 are are illustrating the importance of the learning rate and model. If we compare these two can see that the complexer model with 500 hidden units overfits the training data and there is no significant improvement for the validation set. The complexity of the model can lead to the problem of overfitting. Also it can be seen that the simple model can not pass the 5% error. Therefor these hyperparmeters are not a good choice. In addition the initialization of the weights can lead to a very different performance after the same epochs. Since we are using a SGD for the training and shuffle the data, we can have different results during training.

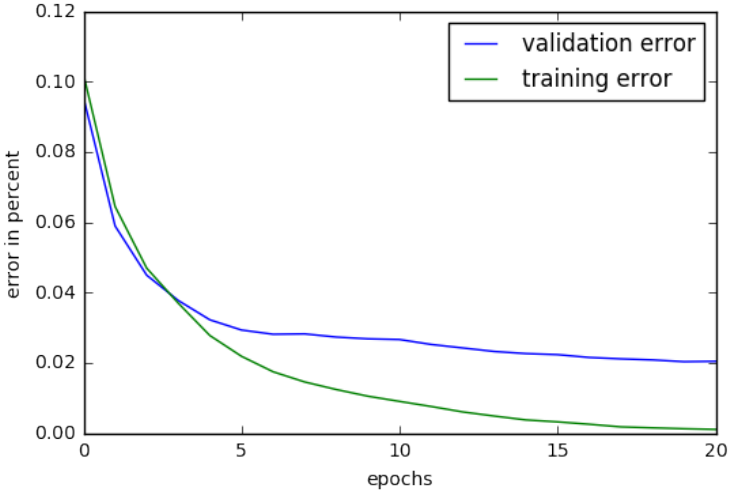

Figure 1: MLP with two fully connected layer, with 500 units each. Trained on on the MNIST dataset with SGD, learning rate 0.1

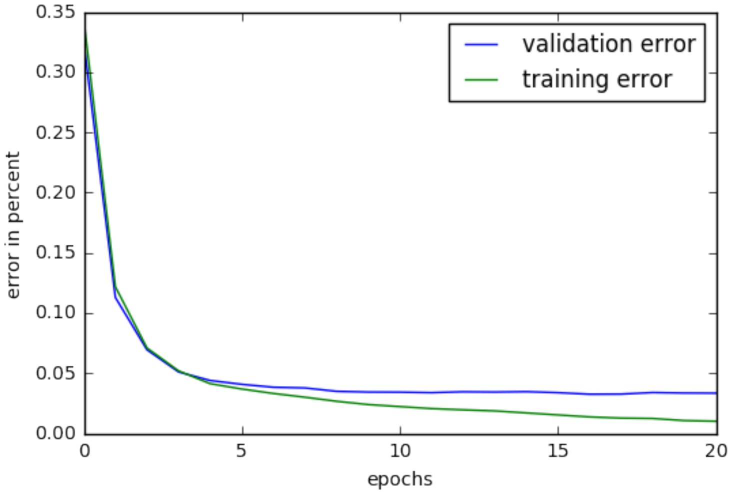

Figure 2: MLP with two fully connected layer, with 50 units each. Trained on on the MNIST dataset with SGD, learning rate 0.1

A good performing could be achieved by:

- increase the number of training samples form 10000 to 40000 data points (10000 samples for the validation data)
- increase the number of units to a higher value to fit the data better
- therefore the model is more complex so a lower the learning rate was reduced to get a good fit
- the disadvantage is that training takes way longer

To find even better parameter a hyperparameter optimization with Tree of Parzen Estimators (TPE) was used for the: learningrate, number of epochs, activation function, number of fully connected layers and hidden units. Running the optimization it was observed that the relu activation function lead to the best results after the same number of steps. Figure 3 show the best performing model. The best run was with a valid error of 0.014% and in the worst of 0.028%. The Figure 4 show some miss classified digits of the network. Here it can be seen that some digits are even hard for a human to classify.

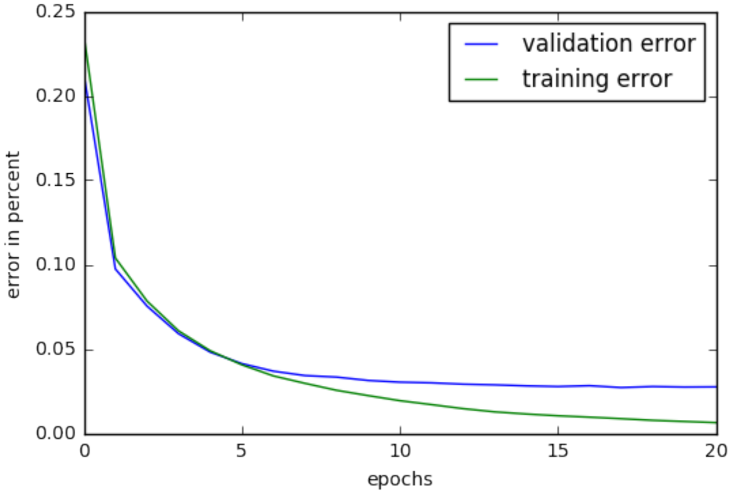

Figure 3: MLP with two fully connected layer, with 50 units each. Trained on on the MNIST dataset with SGD, learning rate 0.1

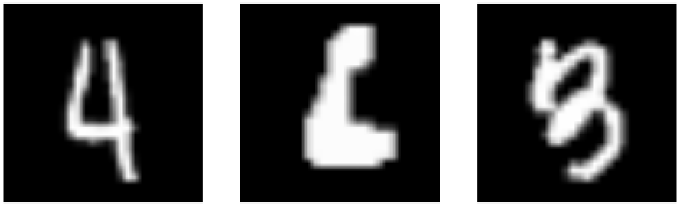

Figure 4: Wrong classified digits for the hyperparameter optimization

3 Summary and Conclusion

In the first exercise of the lab course the we learned the properties and how to implement MLPs. Gradient Checking was a good way show that the implementation was correct and mistakes could be fixed. With the training a real dataset and experimenting with different hyperparameters we could see how difficult it is to find the optimal parameter for the dataset. By a hyperparameter optimization with Tree of Parzen Estimators explore how different parameters influence the performance.

List of Figures

List of Tables