Deep Learning Course 2018 *Vanditha Chandrashekar Rao*
Autonomous Intelligent Systems
University Freiburg *Date: 30.10.2018*

Deep Learning Lab Course 2018 - Exercise 1

In this exercise we have to implement a neural network and classify the MNIST digits. The MNIST problem is a dataset developed by Yann LeCun, Corinna Cortes and Christopher Burges for evaluating machine learning models on the handwritten digit classification problem. The MNIST digit dataset is used to evaluate and compare models, where 50,000 images are used to train a model, 10,000 images for validation and a separate set of 10,000 images are used to test it. This report will give a short overview over the implementation and the resulting classification of the MNIST data set.

Implementation

In this exercise we had to implement a feed-forward neural network by completing the provided stub.

Activation Functions and Layer

The stub provided different types of layers with different activation functions and their derivatives. Activation functions like sigmoid, tanh and ReLU was used for the hidden layer.

The different types of layers used in this neural network are: input layer, fully connected layer, linear output, and softmax output. The weights and biases are propagated through the network. When we are making predictions about categorical data, the best practice is to use a “one-hot encoded” vector. This means that we create a vector as long as the number of categories we have, and force the model to set exactly one of the positions in the vector to 1 and the rest to 0

The Neural Network

As our Neural Network expects flat vectors of size 28*28 as input, hence we must reduce the images down into a vector of pixels for input layer. We can do this transform easily using the reshape() function on the NumPy array.

The hidden layers consist of three Fully Connected Layers with ReLU activation functions. The first layer has 100 hidden units, the second layer has 100 hidden units and the last layer has 10 hidden units, which is the number of classes to classify. The weights and biases are initialized with a normal distribution with a standard deviation of 0.01.

The softmax activation function is used on the output layer to turn the outputs into probability-like values and allow one class of the 10 to be selected as the model's output prediction.

Results

The MNIST data set was divided in a training, validation, and test set. The network was trained and evaluated with the training set. Stochastic gradient descent was chosen as the optimization strategy. The network was trained with 20 training epoch and batch size of 64. **Figure 1a and Figure 1b** shows the training error and validation error with learning rate, **α = 0.1 and α= 0.01 respectively**. As shown in the figure, the training error and the validation error decreases with the increasing number of epochs

As the last part of the task, we had to setup a network that works well and gets reasonable accuracy. The network was trained on the complete data (60,000 digits) and the test error was computed.

Figure 2a and Figure 2b shows the training error of the final network with **α = 0.1 and α = 0.01 respectively**. **Figure 3a and Figure 3b** shows correctly and incorrectly classified digits from the MNIST data set with **α = 0.1 and α = 0.01 respectively**. The test error with α = 0.1 and α = 0.01 is approximately 17.72% and 20.43% respectively

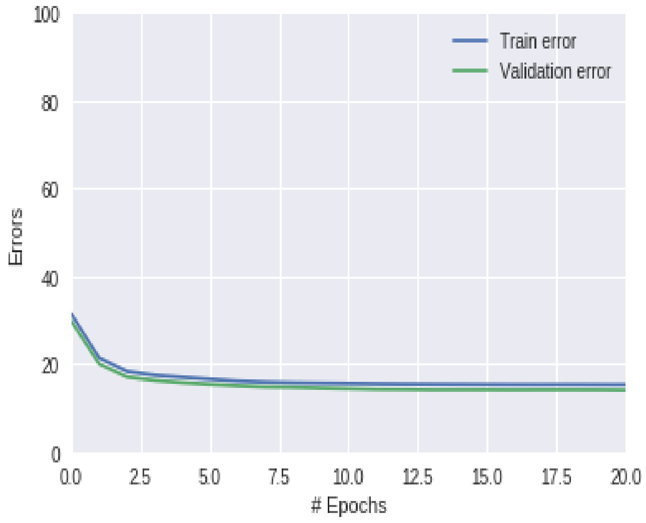

Figure 1a: α= 0.1

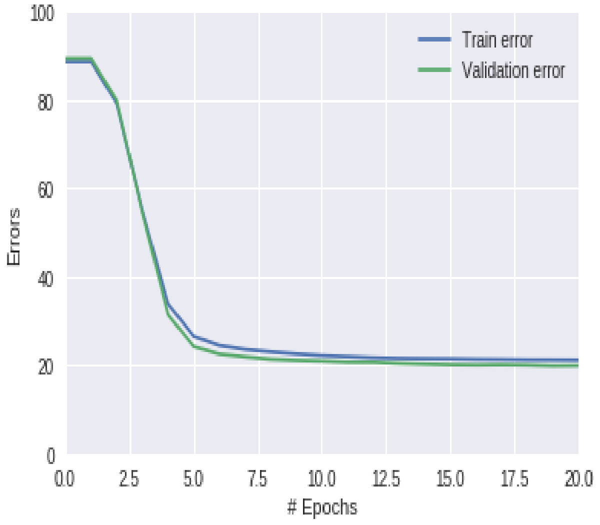

Figure 1b: α= 0.01

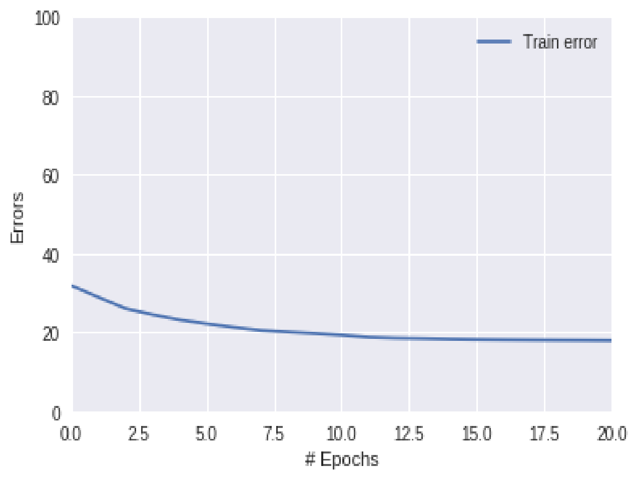

Figure 2a: α= 0.1

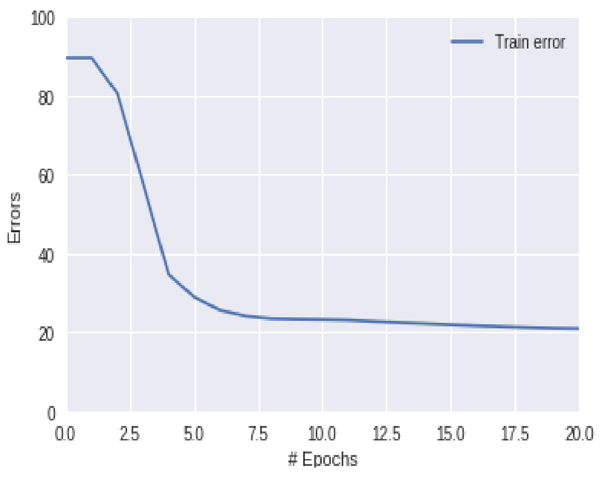

Figure 2b: α= 0.01

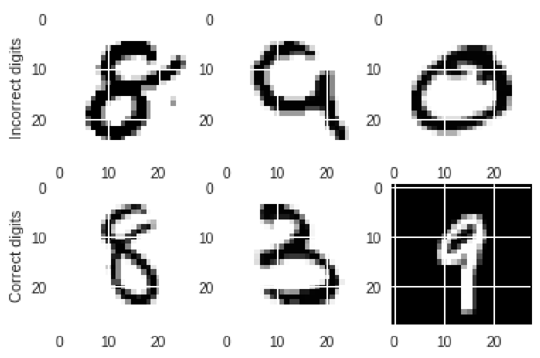

Figure 3a: α= 0.1

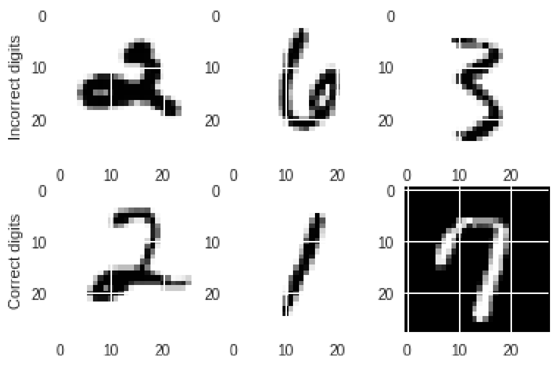

Figure 3b: α= 0.01

The accuracy of correctly classified digit with α = 0.1 is 81.7% and α = 0.01 is 78.38%

Reference

1. https://towardsdatascience.com/gradient-descent-in-python-a0d07285742f
2. https://github.com/kdexd/digit-classifier
3. http://neuralnetworksanddeeplearning.com/chap1.html
4. https://www.python-course.eu/neural_network_mnist.php