Deep Learning Lab – Assignment 1
Hakan Yilmaz

Introduction

The goal of this assignment is the implementation of a simple feed-forward neural network and the application thereof to image classification (MNIST digits). The observations during training with different parameter sets and the training/testing results with the final parameter set are presented in this document.

Observations during training

During training networks with different parameter sets, the following parameters were varied in order to familiarize with their individual effects on final validation accuracy, update accuracy of individual epochs, training time etc.:

- Number of hidden layers (2-4)
- Activation functions of hidden layers (tanh, sigmoid or ReLU)
- Number of units in each layer
- Weight initialization (e.g. standard deviation of normal distribution)
- Optimization technique (gradient descent or stochastic gradient descent)
- Parameters of optimization technique (batch size, learning rate)
- Number of epochs

One could observe that the training error generally decreases faster with respect to epochs, and the final training error is smaller using a higher number of hidden layers and/or a higher number of units in each layer, i.e. the more complex the network the better is the adaptation to the training set. Nevertheless, complex networks do not generalize well on unseen data (overfitting) so that either regularization techniques can be taken advantage of (not in this assignment) or the number of hidden layers/units must be chosen adequately small in a manual way as it was done in this assignment. Additionally, a high number of layers/units directly increases the training time.

Because of their horizontal asymptotes, the squashing activation functions such as tanh or sigmoid are attributed to the so-called “vanishing gradient problem”, i.e. the gradient with respect to certain weights vanishes. Using ReLU(x) = max(0, x) instead, not only tackles this problem, but also decreases training time because of computationally simple applications of the function or its derivative.

Using a randomized weight initialization, e.g. zero-mean-normally distributed numbers, helps to avoid the optimization algorithms “getting stuck” in local minima. Small standard deviations contribute to a fast convergence.

The used optimization technique has implications on the convergence speed and the accuracy of parameter updates. In general, using all training data for one weight update (gradient descent) leads to highly accurate parameter updates but also to long training times. In order to shorten training time, stochastic gradient descent is used where randomly chosen training batches are trained on. The smaller the batch size, the less the accuracy of the weight updates.

The learning rate of the used optimization technique (gd or sgd) critically determines the convergence of the algorithm and its convergence speed. High learning rates accelerate the procedure, but can lead to problems close to local minima. Here, the learning rate is set manually.

The number of epochs determines the final training/validation error. In early stopping techniques (stop training before validation error increases) the number of epochs is determined automatically. Here, this parameter is set manually [http://colinraffel.com/wiki/neural_network_hyperparameters].

Parameter choice and training/testing results

The final neural network to classify the MNIST data with, has the following parameter set:

--- Input (784)

– HL1 (Units: 1028, ReLU)

– HL2 (Units: 512, ReLU)

– HL3 (Units: 256, ReLU)

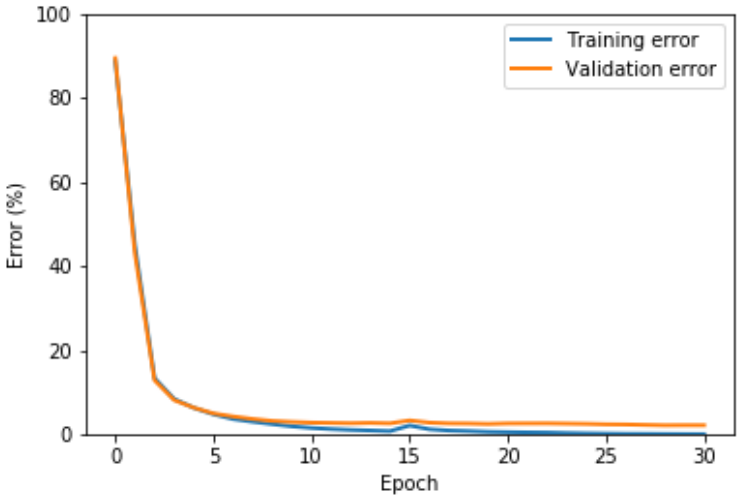
--- Output (Units: 10, Softmax)

Standard deviation of normal distribution for weight initialization: 0.01

Optimization technique: Stochastic gradient descent (batch size: 256, learning rate: 0.15)

Number of epochs: 30

Learning Curve:



Correct/incorrect classified digits: