University of Freiburg

Deep Learning Lab

Neurorobotics/Robotics

Assignment 1

Author:
Mostafa Hussein

Supervisor:
Andreas Eitel

November 5, 2017

Abstract

This report is summarizing the task of Assignment 1 which was implementing a feed-forward neural network from scratch. A MLP was built and applied to the MNIST dataset. The report is divided into 3 sections as follows: Section 1 represent a simple explanation of the implementation, section 2 represents the designed neural network and section 3, represents the results.

1 Feed-Forward Neural Network

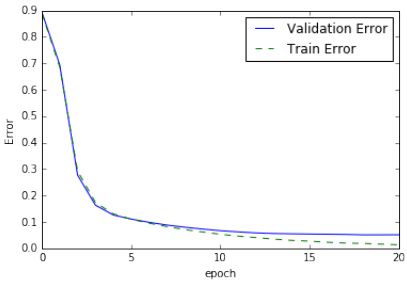
This exercise was an implementation of a Neural Network (or MLP) to classify the MNIST digits with it. Firstly, a function is implemented to download and load the MNIST dataset. Afterthen, the Neural Network Layers are implemented as follows:
Each layer has a constructor that takes an input layer plus some additional arguments such as layer size and the activation function name. The layer then uses the provided input layer to compute the layer dimensions, weight shapes, etc. and setup all auxilliary variables. Each layer then has to provide three functions: $output\_shape()$, $fprop()$ and $brop()$. The $output\_shape$ function is used to figure out the shape for the next layer and the $fprop()/bprop()$ functions are used to compute forward and backward passes through the network.

2 Neural Network Design

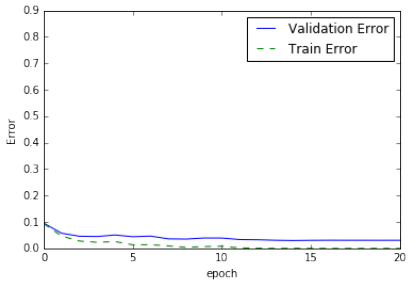
The designed neural network consists of 5 Fully Connected Layers including the input and the output layers. The first hidden layer consists of 250 num_units and a $relu$ activation function. The second one consists of 200 num_units as well as a $relu$ activation function too. However, the last hidden layer consisted of 10 num_units and it used a $linear$ activation function.
The parameters used to train the network was a $learning\_rate = 0.4$, $max\_epochs = 20$ and $batch\_size = 32$

3 Results

The designed neural network which was already provided had nearly the same design of 5 Fully Connected Layers and the same activation functions. However, it was using different parameters which were passed to the network as well as the *num_units* for each hidden layer.
To be clear, in the given network, the *num_units* in each layer was 100, however, this was different in my designed network. As well as the parameters, the $learning\_rate = 0.1$, $max\_epochs = 20$ and $batch\_size = 64$ and my designed network had parameters of $learning\_rate = 0.4$, $max\_epochs = 20$ and $batch\_size = 32$.
Accordingly, the results between the two networks are different. The result from the first network are shown in Figure 1a. The training and validation erros are higher than the second network, they start from 88.86% and 89.36% respectively, then dropping down along the 20 epochs to 1.35% and 5.15%. In contrast, as shown in Figure 1b, the results of the designed network are better throughout the 20 epochs. The training error starts from around 9.35% and converging to 0.00% along the epochs, and as for the validation error, it is also decreasing with the same behaviour by starting from 9.54% and decreasing to 3.08%.
To conclude, increasing the *num_units* of each hidden layer as well as having a smaller *batch_size* and a higher *learning_rate* yielded in a better overall training and validation error as well as better convergence than the ready implemented MLP.



(a) Provided Neural Network.



(b) Designed Neural Network.

Figure 1: Results of the provided and designed Neural Networks