IT3105 Module 5

Johan Slettevold
Iver Egge

November 2015

1 Theano

As requested, we used the theano framework in Python to implement this module. One of the student assistants recommended a theano tutorial consisting of a Youtube video [1] and a Github repository [2]. We followed this guide and used the code as a start point for the network implementation.

2 Network variations

The exercise text mentions five variables that could be modified to create the five network variations required. We decided to define a standard configuration, on which we could modify one variable in at the time. These five variables were:

1. The number of hidden layers;
2. The size of the hidden layers;
3. The activation function;
4. The learning rate;
5. The back-propagation function.

We did some research to find a simple but good standard configuration, primarily by reading the papers provided by this course. The standard number of hidden layers was set to one, as the exercise text says that one layer should be sufficient.

The size of the hidden layers, or rather, the standard total amount of weights, was set to ten times the total amount of observations. This number was based on a tip from a Youtube tutorial [3]. If the network has 784 input weights, 10 output weights, and we have 60000*784 observations, the number of nodes in the hidden layer should equal 600.

The exercise text says that the ReLU activation function is rather superior. We decided to use this as the standard function in combination with noise generation between the layers. This counter-works growth stagnation.

We were quickly able to figure out that the learning rate which the theano tutorial provided was great, and defined this as the standard value. This value is 0.001.

Stochastic gradient descent is a rather straight forward way of solving the back-propagation. However, it does not try to accelerate learning, like for example RMSprop; Therefore, we decided on rather using RMSprop as default.

3 Statistics

The five plots in figure 1 describes the average scores of our five variation configurations through ten learning epochs, respectively. Although we realized that some networks might need to train for longer than 10 epochs, our lack of machine power combined with the impressive results of some configurations persuaded us to limit the training. The first variation describes the hidden layer variation. In this configuration, the network was initialized with two hidden layers, each containing 600 nodes. This variation achieved an average score of 98.11% correct guesses on the MNIST test set.

The second variation describes the layer size variation. We changed the number of nodes in the hidden layer from 600 to 1200 which achieved a score of 98.21% on average.

In the third variation, we changed the activation function from ReLU to Sigmoid and removed the noise factor. Here, we can clearly notice that the noise factor is absent due to the much straighter line. This variation achieved a score of 96.39% on average.

The fourth variation consisted of changing the learning rate. We decided to plot the scores when changing this variable by a factor of plus ten. This variation achieved a score of 96.96% on average.

The last variation describes the difference between SGD and RMSprop. This variation scored poorly, probably because this back-propagation function, SGD, does not learn as quickly as RMSprop. The variant achieved a score of 76.99% on average, but would probably have done better with more epochs.

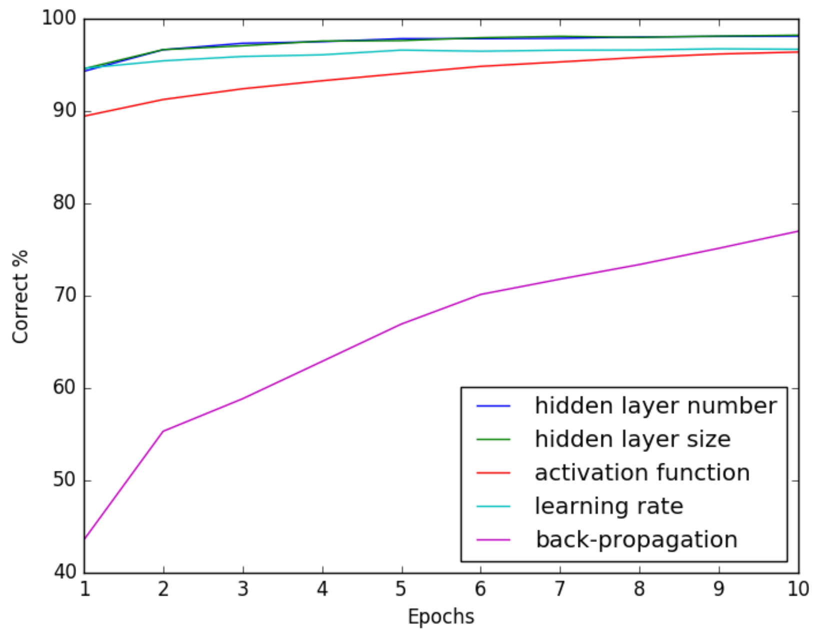

Figure 1: Variation scores

4 Sources

[1] https://www.youtube.com/watch?v=S75EdAcXHKk
[2] https://github.com/Newmu/Theano-Tutorials
[3] https://www.youtube.com/watch?v=S4ZUwgesjS8