TI2736-A

Report

Delft University of Technology

Group 19
Rick Molenaar, ?????
Matthijs Klaassen, ????
Daniël Brouwer, 4288297

Sunday 20th September, 2015
Version 1.0

TU Delft Delft University of Technology

Challenge the future

Contents

1 Introduction

1.1 test

bla bla bla

2 Architecture

- 1. How many input neurons are needed for this assignment?

  Ten, as there are ten initial inputs (features). The network should classify 10 features into 7 classes.

- 2. How many output neurons do you require?
  As was described above the network goes from 10 features (inputs) to 7 classes (outputs), so there are 7 output neurons.

- 3. How many hidden neurons will your network have?
  The amount of hidden neurons is not fixed. It depends on a lot of different factors and it is almost impossible to tell the optimum amount. We started testing with 10 hidden neurons per layer, but eventually settled on 50 neurons per layer wich provided the best result.

- 4. Which activation function(s) will you use?
  We will use the Sigmoid activation function.

- 5. Give a schematic diagram of your complete network
  This schematic only has one hidden layer, ours has multiple but the setup is the same. Also included in the image is a schematic that shows how a neuron handles inputs to create an output. (using the sigmoid activation function).

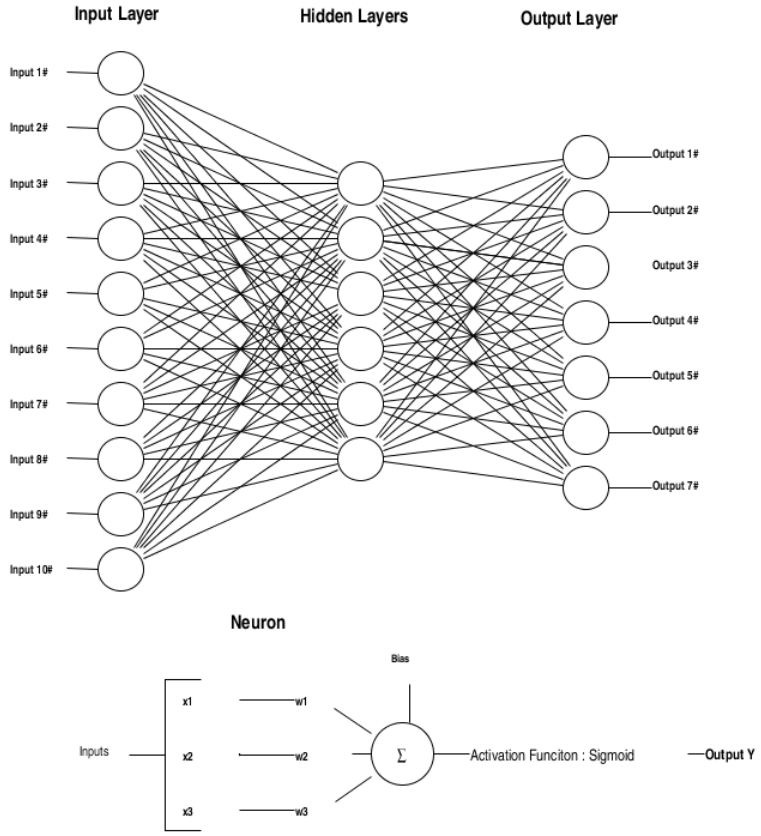

Figure 1: Schematic

3 Training

The neural network that is implemented has to be trained to the type of data it will be processing. To do this the data from targets.txt is divided into a training, validation and test set. By changing the parameters of the network and comparing error's in the validation layer, the "optimal" parameters can be found.

3.1 Dividing the data

The data (7854 features with 10 inputs) is divided into 7 folds of 7854/7=1122 input lines each. 5 of these folds will be used for training and the other two for the test and validation set respectively. The purpose of the test fold is to observe the error percentage when the network is untrained and thus uses random weight between -0.5 and -0.5. The training folds are used to train the network thus updating the weights of the network. After the network is trained the network processes the validation fold. For the validation fold the error percentage and sum of squared error's is calculated. Now the results from the test set and validation set can be compared and give insight in how well the network trained itself with the given parameters.

3.2 Evaluating the performace

To evaluate the performance of different neural networks, lots of different neural networks with all different parameters will follow the same procedure mentioned above. The error percentage, sum of squared errors and the parameters are then stored for each network in a single txt file.

The following parameters were changed for each network and checked for their influence:

- The learning rate α
- The amount of hidden layers
- The amount of neurons per hidden layer
- The amount of epochs of training

At the end, the parameters of the neural network which scored best,will be used for processing the unknown.txt inputs.

3.3 The amount of training epochs

Because the neural network is implemented in Java code, the training epochs run quite fast therefore lets say thousand epochs, 1 hidden layer with 100 neurons will only take about ten minutes of training although it seems that after 30-50 epochs the error rate is not really improving anymore. To be sure in deciding the best parameters for the neural network, 100 epochs were run for each individual network.

3.4 Impact of the initialization

The initialization where each weight gets a random value between -0.5 and 0.5 does seem to effect the performance to some extend. Networks with the same parameters can have different outcomes which is expected because were the final "solution" converges, depends on the initial weights.

4 Optimization

4.1 Performance of different networks

As described in section 3 different networks with different parameters can be compared to obtain the best parameters. fig. 2 shows the influence of changing the number of hidden neurons. Because

the performance of the network is somewhat random due to initializations each training and validation is done 10 times for each set of parameters. The parameters that were held constant were:

- Amount of hidden layers = 1
- Learning rate alpha = 0.1
- Amount of epochs = 20

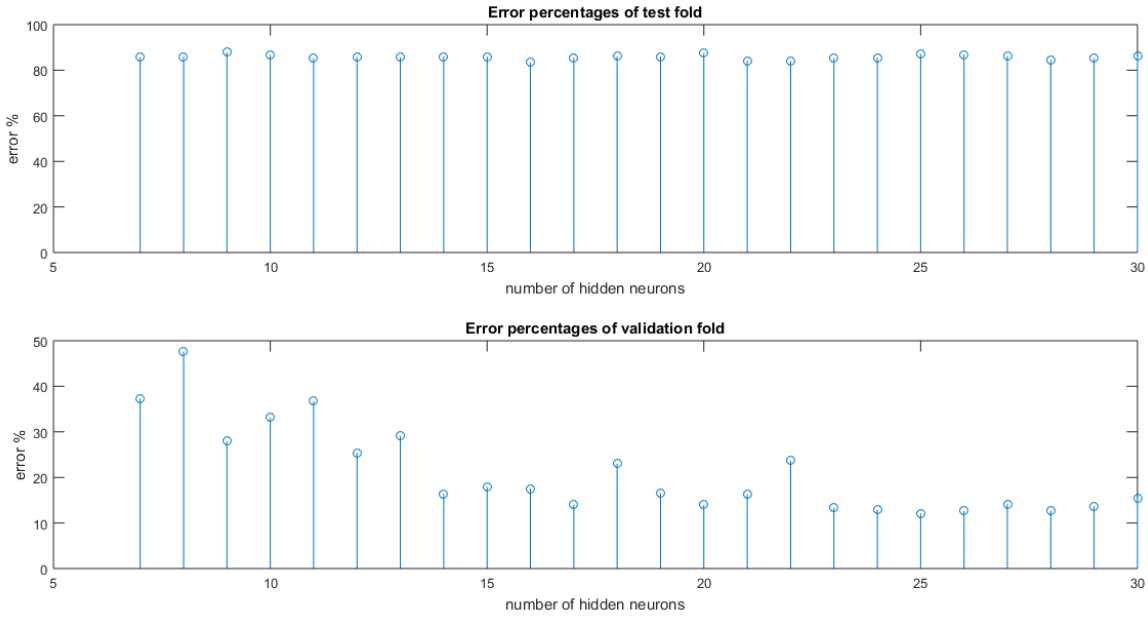

Figure 2: Error percentages of test and validation fold

The amount of neurons in the hidden layer is adjusted from 7 to 30 neurons and each time the data for the errors is recorded for 10 runs of the same network. Afterwards the errors of 10 runs are averaged per network and then the errors for the networks are compared with the help of Matlab. fig. 2 shows that the error percentage definitely decreases with more hidden neurons used, although the relation is not linear and the effect of adding more neurons becomes less and less effective.

4.2 Optimum parameters

After many tests, 1 hiddenlayer containing 100 neurons resulted in the lowest stable errorrate. And therefore these parameters are used for determining the outputs of the unknown.txt inputs.

5 Evaluation

5.1 Succes rate of the network

Using the 1 hidden layer and 100 hidden neurons the error rate in the validation set was down to 2.67% after 1000 epochs which corresponds to 30 errors of the (7854/7) targets in the validation set. In the test set the error percentage was 85.38% with 958 errors. To show the dispersion of the errors a confusion plot was made from the outputs and targets using Matlab. The confusion matrix is show in fig. 3 and a plot of the values is show in fig. 4.

```
C =

   163     0     2     1     1     0     0
     0   166     1     0     0     1     2
     4     1   144     0     0     0     0
     0     2     0   150     0     0     1
     0     1     6     0   151     1     0
     0     0     0     0     0   159     1
     0     2     0     1     1     1   159
```

Figure 3: Confusion matrix

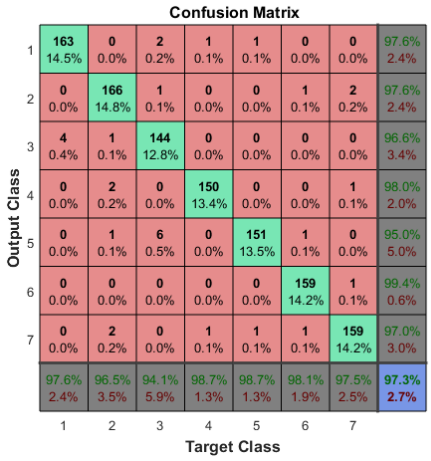

Figure 4: Confusion plot

From the matrix and plot it can be concluded that the trained network works quite well on the validation set. The few errors that exist are somewhat evenly distributed over the whole set. Output number 3 has the highest error rate of 5.88% with 9 error values from the 144+9=153 values.

6 Matlab's Toolbox

6.1 Comparing results

Using the nprtool from Matlab a similair network with also 100 hidden neurons is run to compare it to the network made using Java code. To show the performance of the nprtool generated network, a confusion plot was made (see fig. 5). The nrptool gave an error percentage of 6.2% for the validation fold which is bigger than the error using the Java code (2,67%). Both the nrptool and the Java code will give different results each run, depending on the initialization weights. Also the Java code was run for 1000 epochs while the nrptool only runs about 20-30 epochs every time. These factors have a significant effect on the outcome therefore the Java code performed slightly better.

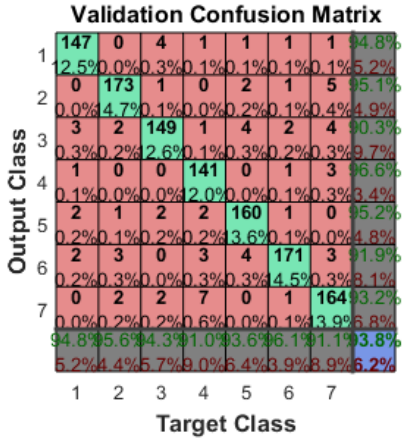

Figure 5: Confusion plot using network trained with nprtool

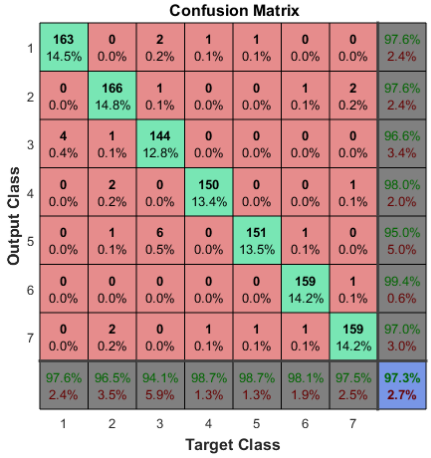

Figure 6: Confusion plot using network trained with Javacode

References

[1] M. Negnevitsky, *Artificial Intelligence A Guide to Intelligent Systems, Second Edition*, Pearson Education, Harlow, 2005.