Assignment report

First of all we will prepare the data to train the model given, with the function *str()* we will be capable to identify all the target attribute and have a brief view of the dataset.
The next step is to set a training and a test set to check the performance of the neural network, in this case we will use an 80% of the data set as training model and the remainder as test set.

So we set the next parameters to the function *neuralnet()* in order to train the data:

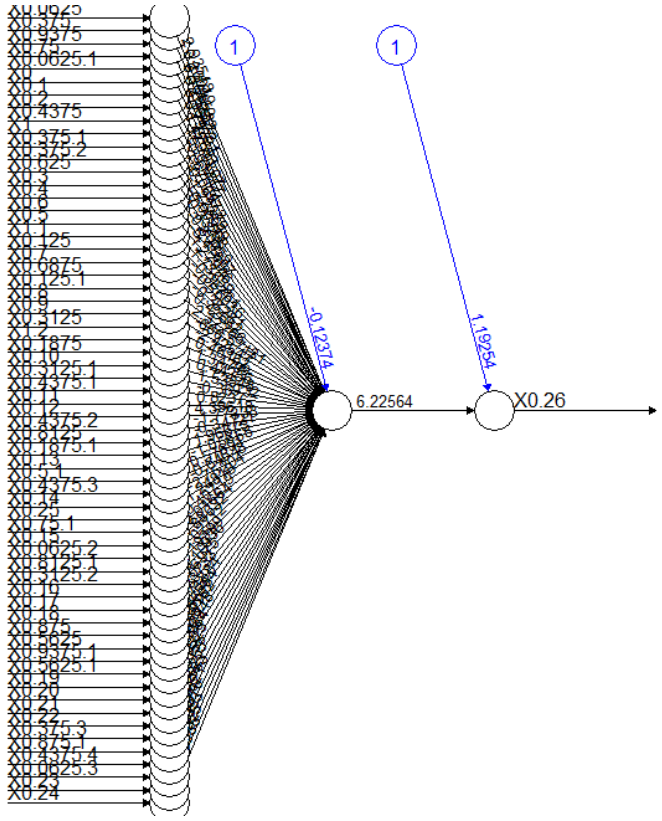
```
nn <- neuralnet(X0.26 ~ X0.0625 + X0.375 + X0.9375 + X0.75 + X0.0625.1 + X0 + X0.1
                + X0.2 + X0.4375 + X1 + X0.375.1 + X0.375.2 + X0.625 + X0.3 + X0.4
                + X0.6 + X0.5 + X1.1 + X0.125 + X0.7 + X0.6875 + X0.125.1 + X0.8
                + X0.9 + X0.3125 + X1.2 + X0.1875 + X0.10 + X0.3125.1 + X0.4375.1
                + X0.11 + X0.12 + X0.4375.2 + X0.8125 + X0.1875.1 + X0.13 + X0.5.1
                + X0.4375.3 + X0.14 + X0.25 + X0.75.1 + X0.15 + X0.0625.2 + X0.8125.1
                + X0.3125.2 + X0.16 + X0.17 + X0.18 + X0.875 + X0.5625 + X0.9375.1
                + X0.5625.1 + X0.19 + X0.20 + X0.21 + X0.22 + X0.375.3 + X0.875.1
                + X0.4375.4 + X0.0625.3 + X0.23 + X0.24,
                data = handwriting_train)
```

There are all the variables of the training data set, to do the first test we will leave the hidden nodes as default (1) to see the results and try to improve them if possible.

Results of the NN:

```
1 repetition was calculated.

        Error Reached Threshold Steps
1 6284.446747    0.008882733973  6674
```

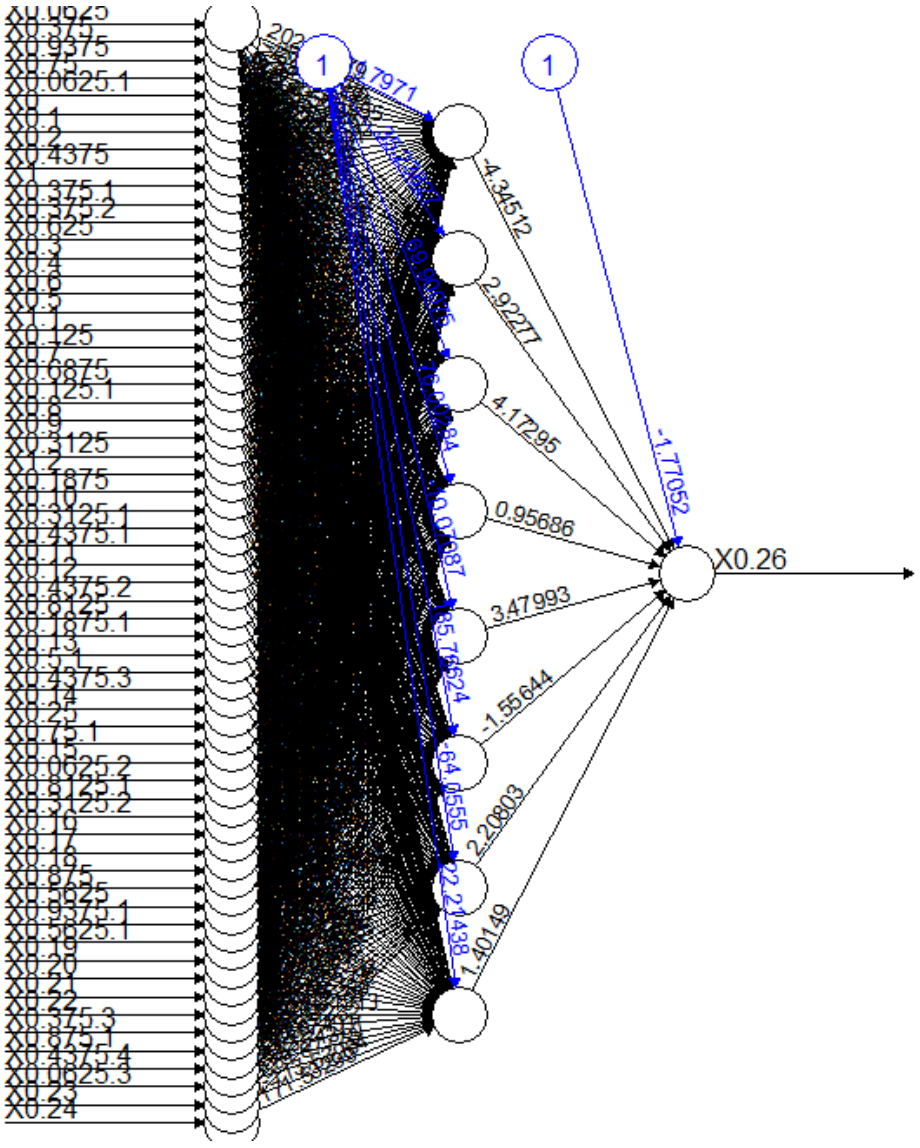


The results we get are as expected too poor given the fact that we have a large error rate and we will skip the performance analysis due to that.

Now we will try to make a new NN with more hidden layers that would be set as $\sqrt{n}$ where n will be the number of variables, so we will use 8 hidden layers this time. I have used this formula to measure the layers because I think it will be optimal to link the number of variables and the number of layers to obtain the performance, the results are:

```
1 repetition was calculated.

        Error Reached Threshold  Steps
1 529.0719921    0.009908659667 699291
```



As we can see, with the entry of the 61 variables and only 1 layer containing 8 hidden nodes and with an error calculation as default (sse) and the constant 1 (blue nodes) to make the result closer to the average of the entries for the multipliers gives us a result apparently much better than the neural network calculated before. To see more detailed information about the neural network we will output all the rates in a matrix:

```
                                  1
error                  5.290720e+02
reached.threshold      9.908660e-03
steps                  6.992910e+05
Intercept.to.1layhid1 -3.179710e+01
X0.0625.to.1layhid1    2.029808e+02
X0.375.to.1layhid1    -2.691610e+01
X0.9375.to.1layhid1   -3.822978e+01
X0.75.to.1layhid1     -2.508959e+01
X0.0625.1.to.1layhid1 -3.056073e+01
X0.to.1layhid1        -1.143986e+02
X0.1.to.1layhid1       9.889356e+01
X0.2.to.1layhid1       6.919859e+02
X0.4375.to.1layhid1   -4.641431e+01
X1.to.1layhid1         2.005324e+01
X0.375.1.to.1layhid1   2.942609e+01
X0.375.2.to.1layhid1  -3.016410e+01
X0.625.to.1layhid1    -1.719373e+01
```

This is a partial example of the matrix output view that allows us to identify every weight path from the variables entry to the hidden layer.

To estimate the accuracy of the net over unseen instances we will make use of the correlation between the obtained data and the test dataset that we created before.

```
> cor(predicted_target, handwriting_test$X0.26)
          [,1]
[1,] 0.8868396
```

As we can see, the correlation between both variables is very powerful and we can confirm that the neural network is fairly well trained and it would be capable of identifying most of the handwritten characters, in addition to these predictions we could use a function that given the string of characters predicted by the neural network determines the most common word, this way we can be perfectly sure that the neural network combined with that function could identify all kind of handwritten characters, words and sentences.

Dependence between the set of examples and the accuracy:

The dependence is given by the number of solutions that the neural network is able to calculate, but with more instances in the training set and more solutions calculated by the paths along the hidden nodes and layers the accuracy would increase. So we can conclude the accuracy depends equally on the training set and the number of layers and hidden nodes, because if we have a large number of instances to train, but the algorithm only trains a limited amount of cases the performance of the neural network decreases, and is the same case if we have a few examples to train and a lot of layers, because the performance would be the optimal for these instances, but different unseen instances could be useless.

To achieve an accuracy of 99% over unseen instances we may be able to know all the possible combinations of results for that problem and then we could calculate the optimal size with the most fitting examples to train the neural network, but in this case we don't know all the possible results, so we can estimate that if for 4495 objects we obtain a 88% accuracy approximately, to achieve a 99% could be 5056 instances, but this is only a speculation approximation, because the examples added could all be already trained by the neural network and don't affect to the performance.