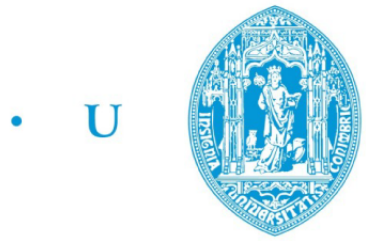
OCR – Optical Character Recognition

Francisco Jesus 2020140671
Pedro Louro 2020173394

October 2020



FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Teacher: António Dourado

Index

1 Introduction

With this project, we intend to develop neural network models for hand written character recognition, in particular the ten Arabic numerals: {1,2,3,4,5,6,7,8,9,0}.

Using the supplied **mpaper.m MATLAB** function, up to 50 hand-drawn characters can be converted into individual 16x16 matrices consisting of binary values, which are then converted to a 16x16 column vector of elements.

In order to achieve the goal of this work, two neural networks architectures were studied:

1.1 Filter + Classifier

The first architecture is comprised of two models, a filter, which could be implemented through **associative memory** or a **binary perceptron**, and a classifier.

The filter has the capacity to approximate the given input to the corresponding perfect character provided has the target, the result being given has the output. The target for the aforementioned approximation was constructed through several concatenations of the provided **MATLAB** file **PerfectArial.mat**, which contains perfect designs of the ten digits.

The associative memory filter consists of a single layer network, with linear activation functions and without bias. The input to be provided is a vector with 256 elements representing a character. The binary perceptron filter uses 256 hardlim neurons for the same effect and receives the same input.

The classifier consists of a single layer with ten neurons with bias in each one and one of three activation functions, they being linear, binary or sigmoidal. It receives as input the output provided by the associative memory or binary perceptron and provides a vector with ten elements as the output, which consists of zeros and a one identifying the class of the digit it belongs to

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 0$$

. The weight and bias are initialized with random values which are optimized by training the network with successive inputs. Its important to note that the optimizations are dependent on the activation function used and that if hardlim(binary) is used, it should be evaluated by the perceptron rule, and the gradient method for the others.

1.2 Classifier

The second architecture as only the classifier, that functions the same way as the classifier in the first architecture, with characters being directly provided to the classifier with a considerable generalization capability since there is no filtering.

2 Data Set

For training, the data set used, P_test, contains 500 cases, 50 unique cases concatenated 10 times, and for validation, testingP, another 50 unique cases were used, all obtained through the use of the function mpaper.m.

The first architecture uses the data set mentioned above and the matrix provided by the PerfectArial.mat file concatenated enough times to fit the necessary dimensions as the target.

The second architecture uses the same data set and as output the desired target for each character.

During testing, it can be deduced that the bigger the size of the data sets, the longer the network takes to train, although being too small can lead to bad performance since the network isn't apt for too many differences from the data set.

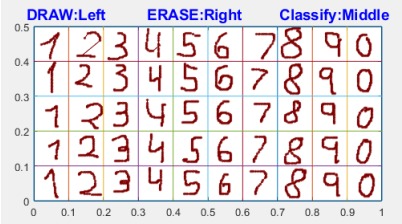


Figure 1: P_test

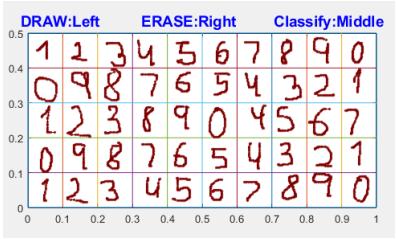


Figure 2: testingP

3 Neural Network Architectures and Results

3.1 Classifier with 2 layers (hidden + output)

In this architecture, it was implemented patternnet with one hidden layer consisting of 256 neurons using transig function (hyperbolic tangent sigmoid) and the output layer with 10 neurons. It was decided to use 256 neurons in the hidden layer, because it seemed interesting to replicate the first architecture, where the filter uses 256 neurons too.

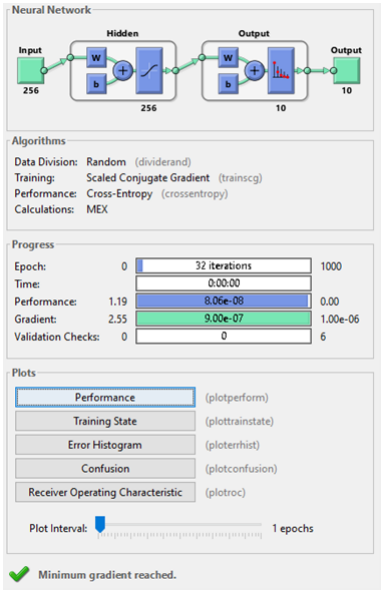


Figure 3: Patternnet with one hidden layer, transig function and output layer with 10 neurons

The result obtained using the data in testingP.mat was the following:

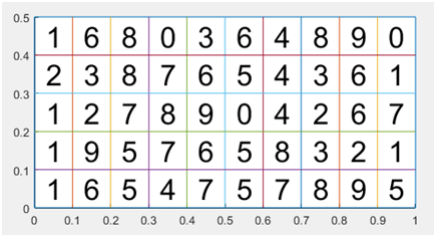


Figure 4: Classifier with 2 layers (hidden + output) results

3.2 Classifier with 1 layer (purelin + traingd)

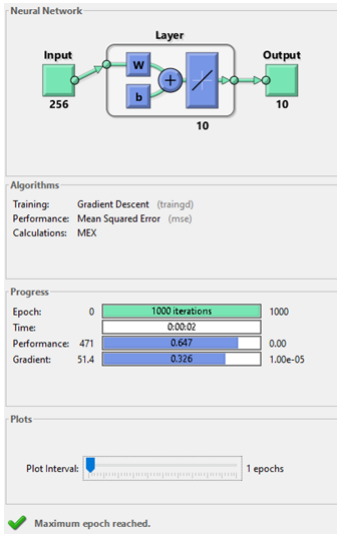


Figure 5: Single layer with 10 neurons, using purelin transfer function and traingd training function

In this architecture it was used a single layer with 10 neurons, that uses purelin as its transfer function (linear) and uses as training function traingd (gradient descent backpropagation). The results were as follow:

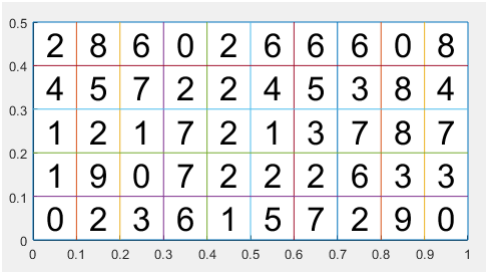


Figure 6: Classifier with 1 layer (purelin + traingd) results

Which corresponds to 24% of correct numbers. It was also observed that it used all the epochs to train, which can mean that it did not reach the best optimization.

3.3 Classifier with 1 layer (purelin + trainlm)

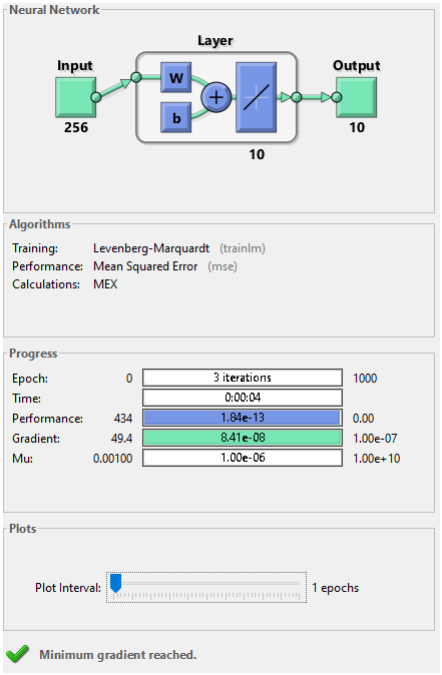


Figure 7: Single layer with 10 neurons, using purelin transfer function and trainlm training function

In this architecture, the training function was changed to trainlm (Levenberg-Marquardt) and the following results were obtained:

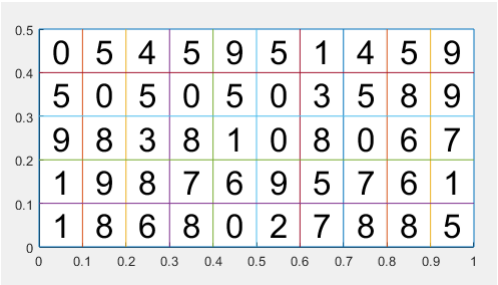


Figure 8: Classifier with 1 layer (purelin + trainlm) results

Which corresponds to 26% of the correct numbers. It can be deduced that the architecture with gradient descent backpropagation could achieve better results than this one, since it couldn't be completely optimized.

3.4 Classifier with 1 layer (hardlim)

Figure 9: Single layer of q0 neurons with hardlim transfer funtion

In this architecture it was used one layer of 10 neurons with the hardlim transfer function. The results obtained were:

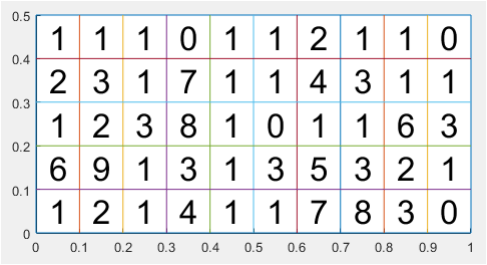


Figure 10: Classifier with 1 layer (hardlim) results

Which corresponds to 44% of correct numbers. This appeared to be strange, since that this layer was expected to be the worse compared to logsig and purelin.

3.5 Classifier with 1 layer (logsig + trainscg)

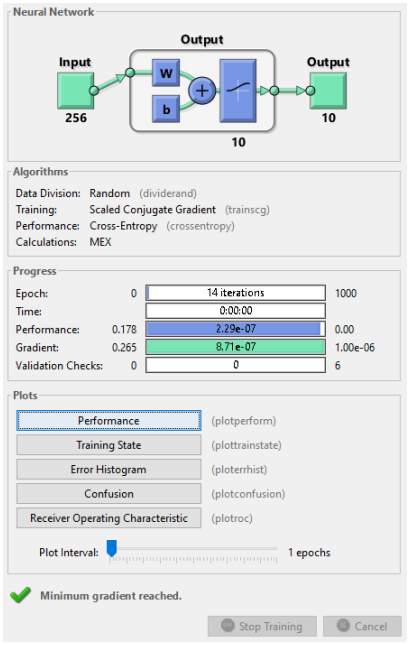


Figure 11: Single layer with 10 neurons, logsig transfer funtion and trainscg training function

In this architecture, it was used a neural network with 10 neurons and one layer, using logsig as its transfer function (sigmoidal) and trainscg as its training function (scaled conjugate gradient). Trying to use traingd and trainlm for this architecture, yielded worse results. So, it cam be deduced that trainscg may be better for this type of architecture of the three functions used.

The results obtained were:

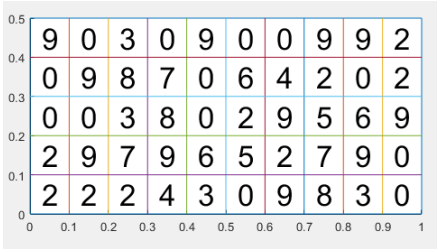


Figure 12: Classifier with 1 layer (logsig + trainscg) results

Which corresponds to 36% of correct numbers.

3.6 Classifier with 1 layer (softmax + trainlm)

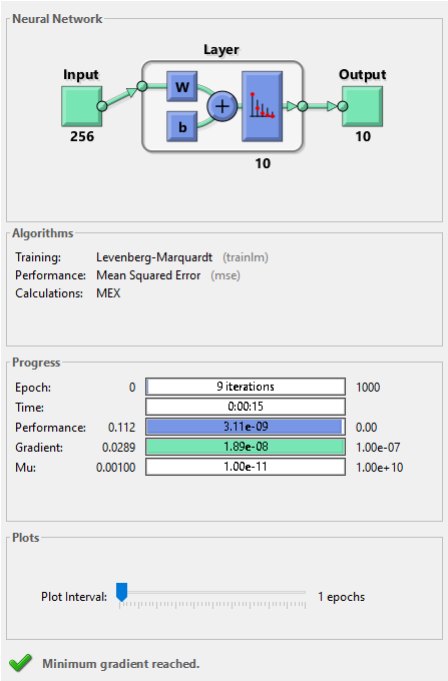


Figure 13: SIngle layer with 10 neurons, softmax transfer function and trainlm training function

In this architecture, it was used one layer with 10 neurons, which uses softmax as its transfer function and trainlm as its training function. This architecture took 15 seconds to finish training, but of all the architectures with one layer that were studied, it was the one the best results.
The results were:

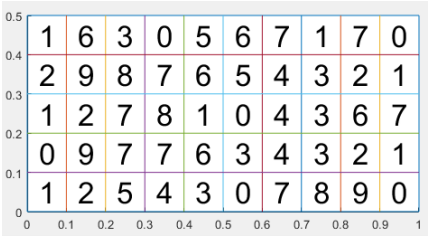


Figure 14: Classifier with 1 layer (softmax + trainlm) results

Which corresponds to 74% of correct numbers. We can deduce that the use of softmax may improve the network.

3.7 Filter + Classifier (Binary Perceptron filter + classifier)

Firstly, we have binary perceptron filter with 256 neurons and a linear classifier with 10 neurons and traingd. In this case the linear classifier reached the 1000 epochs and can conclude the same that was said previously for this classifier. The results for this architecture were:

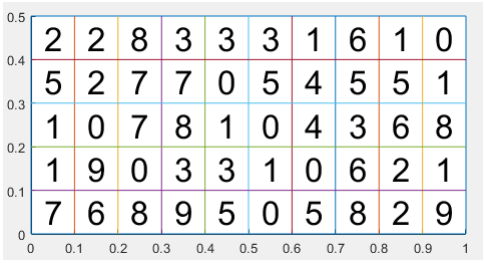


Figure 15: Binary Perceptron filter with 256 neurons and linear classifier with 10 neurons results

Which means, that there are 32% of correct numbers, which means that compared to the architecture with only the linear classifier, this one is better, since in this one the values that come from the input are adjusted to be close to the ones from the target, so the classifier has a better change at giving correct values.

Secondly, we have binary perceptron filter with 256 neurons and a sigmoidal classifier with 10 neurons and trainsgd. The results for this architecture were:

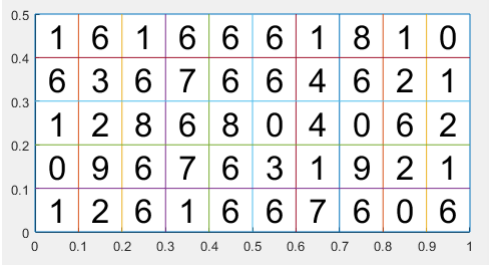


Figure 16: Binary Perceptron filter with 256 neurons and sigmoidal classifier with 10 neurons results

Which corresponds to 48% of correct numbers, which again is better than the classifier alone.

Lastly, binary perceptron filter with 256 neurons and a hardlim classifier with 10 neurons. The results were:

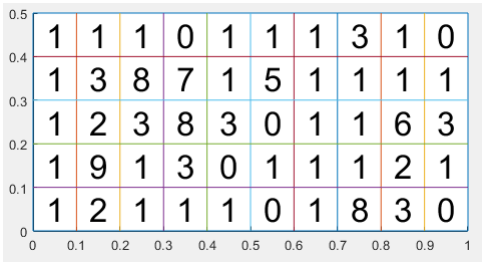


Figure 17: Binary Perceptron filter with 256 neurons and hardlim classifier with 10 neurons results

Which means that there are 38% of correct numbers. In comparison with the classifier with hardlim, it was worse. Probably this classifier works better without a filter, but like said previously, this is strange and could be something wrong with the hardlim classifier.

3.8 Filter + Classifier (Associative Memory Filter + Classifier)

As for the binary perceptron filter, the associative memory filter with the three classifiers (hardlim, linear, sigmoidal) were also combined. This filter as proven to be slower in comparison with the binary perceptron filter.

First, associative memory filter and a linear classifier. In this architecture there is a filter that uses associative memory and a linear classifier with 10 neurons and transfer function traindg. The results for this architecture were:

Figure 18: Associative Memory filter and linear classifier with 10 neurons results

Which corresponds to 14% of correct numbers.

The next architecture is the associative memory filter combined with a sigmoidal classifier. The results for this were:

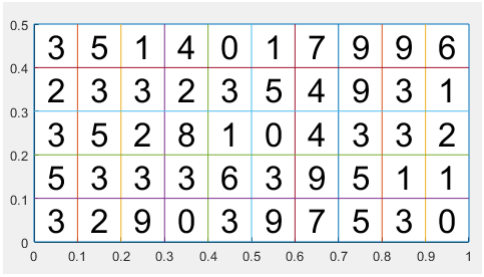


Figure 19: Associative Memory filter and sigmoidal classifier with 10 neurons results

Which corresponds to 28% of correct numbers.

The final architecture is the associative memory filter combined with a hardlim classifier. The results were:

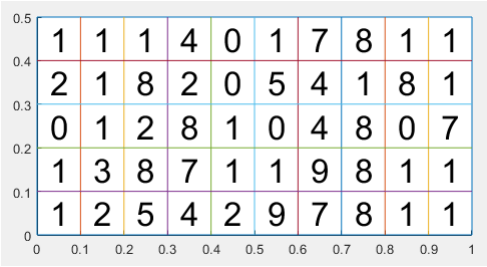


Figure 20: Associative Memory filter and hardlim classifier with 10 neurons results

Which means that there are 40% of correct numbers. It can be deduced that associative memory may work better combined with the hardlim classifier.

4 Conclusion

To conclude, after a thorough comparison, the worst architecture was filter + classifier using associative memory, since it presents the worst results of all the architectures and it gave worst results than classifiers alone.

The architecture with perceptron and classifier seems to improve the results, since the classifiers alone give worst results than this one.

In relation to the best transfer function (hardlim, sigmoidal and linear), it was observed that, based on the results and the fact that the hardlim classifier seems to not be working appropriately, the sigmoid transfer function might be the best one.

In relation to the classifier with two layers, it was observed that this one gave better results than all the architectures, except the classifier with softmax architecture. Which means that using two layers, one hidden and one for output is a better option than using only one layer to classify.

The use of softmax improved the results, since both the best architectures use this transfer function (classifier with two layers - 64% and classifier with one layer with softmax - 74%). Also using this two architectures could be viable to interpret numbers that are not perfect.