Optical Character Recognition (OCR)

Aprendizagem Computacional / Machine Learning (MEI)

Computação Neuronal e Sistemas Difusos / Neural Computation and Fuzzy Systems (MIEB)

André Correia (MEI) – 2020158188

Ennio Malvati (MEI) – 2020187992

Departamento de Engenharia Informática.

Faculdade de Ciências e Tecnologia

Universidade de Coimbra

October 2020

Contents

Introduction

It is proposed, with this practical assignment, to develop neuronal networks in order to recognize handwritten characters, being those characters the 10 Arabic numerals:

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

The main goal was to create random characters (up to 50) and classify them.

In order to achieve the proposed goal, we had to apply the theoretical knowledge acquired during classes and develop a model using MATLAB software.

Three neural networks were created (a single layer one, a double layer one, and single layer with a binary hardlim perceptron acting as a filter) in order to analyze different architectures and compare their performances.

Different datasets were used (a first one with 500 characters and a second one with 1000 characters) as well as different training algorithms. The first dataset with 500 characters was created only by one person and the one with 1000 characters was created by two different persons. Also, the one with 500 characters has almost perfectly written numbers, while the other one has numbers with some defects.

It must be noted that, for testing purposes, a third person created an input matrix in order for us to test the models. The changes, due to this, were huge.

1.Network Architecture

As mentioned before, three models were created in order to fulfill the proposed objective.

After some testing with different training algorithms, the final chosen architecture was the following:

For the binary perceptron as filter, we used the hardlim activation function with perceptron learning algorithm and incremental training:

```
net.numInputs = 1;
net.numLayers = 1;
net.layers{1}.size = 256;
net.layers{1}.transferFcn = 'hardlim';
net.trainFcn = 'trainc';
net.adaptFcn = 'learnp';
net.outputConnect = [1];
net.inputConnect = [1];
net.inputs{1}.size = 256;
net.inputWeights{1,1}.learnFcn = 'learnp';
```

For the single layer classifier, we used the logsig activation function with gradient descent learning algorithm, and the default training method (Batch):

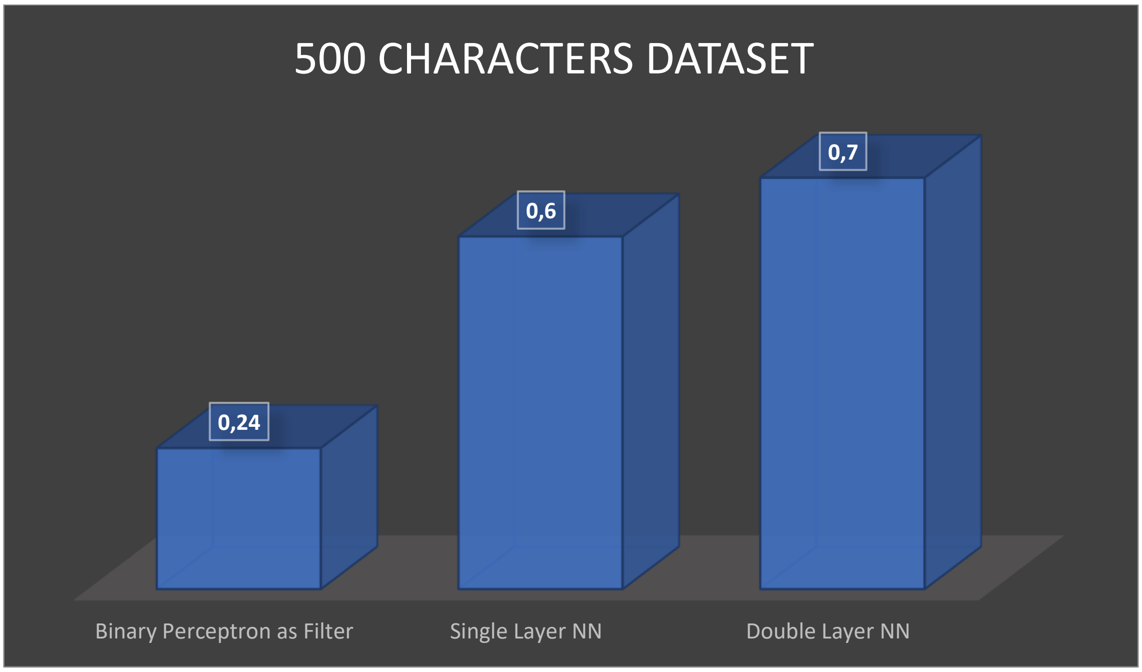
```
net.numInputs = 1;
net.numLayers = 1;
net.trainFcn = 'trainscg';
net.adaptFcn = 'learngd';
net.outputConnect = [1];
net.inputConnect = [1];
net.biasConnect = [1];
net.inputs{1}.size = 256;
net.layers{1}.transferFcn = 'logsig';
```

For the double layer classifier, we used a feedforward network with 120 neurons in the hidden layer, with logsig as activation functions in the first layer and purelin as activation function in the second layer, scaled conjugate gradient as the training algorithm and gradient descent as the learning algorithm. For training, only training (85%) and validation (15%) sets were used:

```
net = feedforwardnet(120);

net.trainFcn = 'trainscg';

net.trainParam.epochs = 4000;
net.trainParam.goal = 0;
net.trainParam.min_grad = 1e-100;
net.trainParam.max_fail = 500000;
net.divideFcn = 'dividerand';
net.performFcn = 'mse';
net.adaptFcn = 'learngd';

net.divideParam.trainRatio = 0.85;
net.divideParam.valRatio = 0.15;
net.divideParam.testRatio = 0.0;

net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'purelin';
```

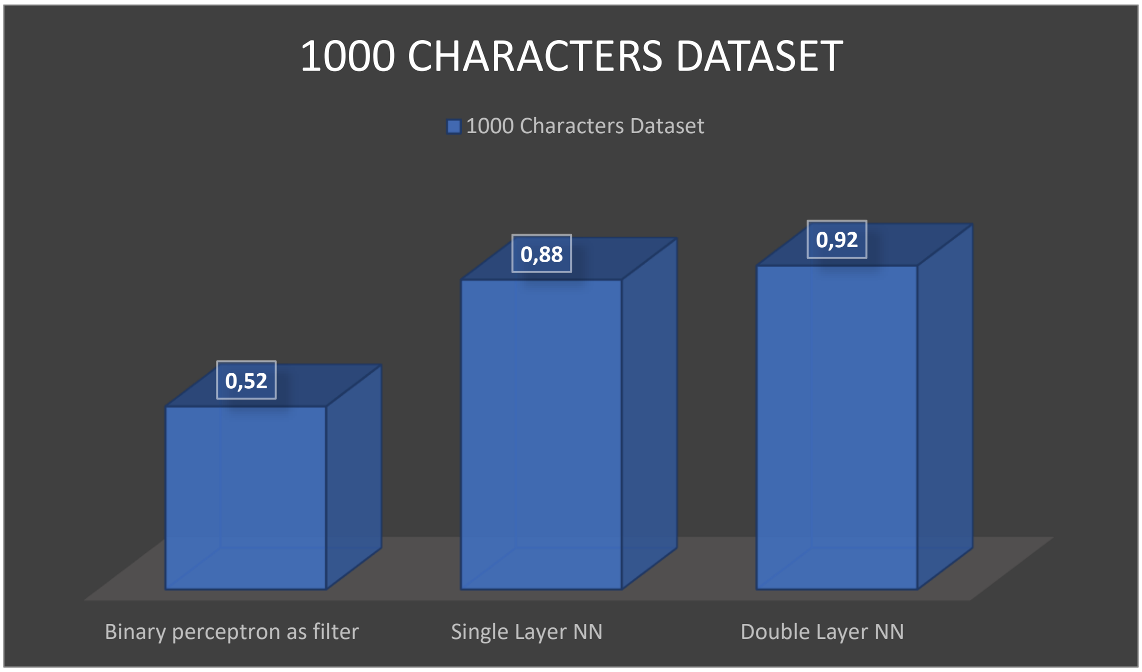
For all this networks, the default learning rate was used, and we were satisfied with the results that it provided

2.Performance Review



From the graphic above, we can see that the best performing neural network was the Double Layer one. In fact, it was the best performing network with any learning and training algorithm (taking into consideration that the activation functions from the first layer were always sigmoidal).

The double layer network gave us an accuracy around 70%, the Single layer one gave us an accuracy of around 60% while the binary perceptron used as a filter gave us an accuracy of around 25%.



From the above chart, we can clearly see improvements in every network due to the increase of the dataset. The Double Layer network achieve around 92% of accuracy, the Single Layer NN was not too far behind with around 88% and the Binary perceptron as filter was able to correctly classify around half of the testing sample with around 52% of accuracy.

Once again, the Double Layer NN was the network with the better performance, but the Single Layer NN was not too far behind. The biggest gains, were clearly in the Binary perceptron as filter

2.1 Additional Performance

With another training algorithms, we could see some changes in the performance, but clearly the above-mentioned parameters were the best.

The softmax layer could give an increase of 1 or 2 per cent, but the training time was clearly longer, so we decided no to go with softmax layer since the gains were not compensating the training time (clearly a personal choice).

In the Double Layer NN, the best approach was to use sigmoidal activation functions in the first layer and linear activation functions in the second layer.

Using both sigmoidal functions in both layers, showed us a huge decrease in performance

The training algorithm ‘traincgp’ was showing a decrease of 4 of percent in the classification.

Using ‘tansig’ instead of ‘logsig’ also showed a decrease of around 4% in accuracy.

3.Software Architecture

The architecture of our software is developed in the following way:

initTrainNetworks.m:

This function will create the three networks discussed in this project, and will train them with the mentioned parameters.

myClassifier.m:

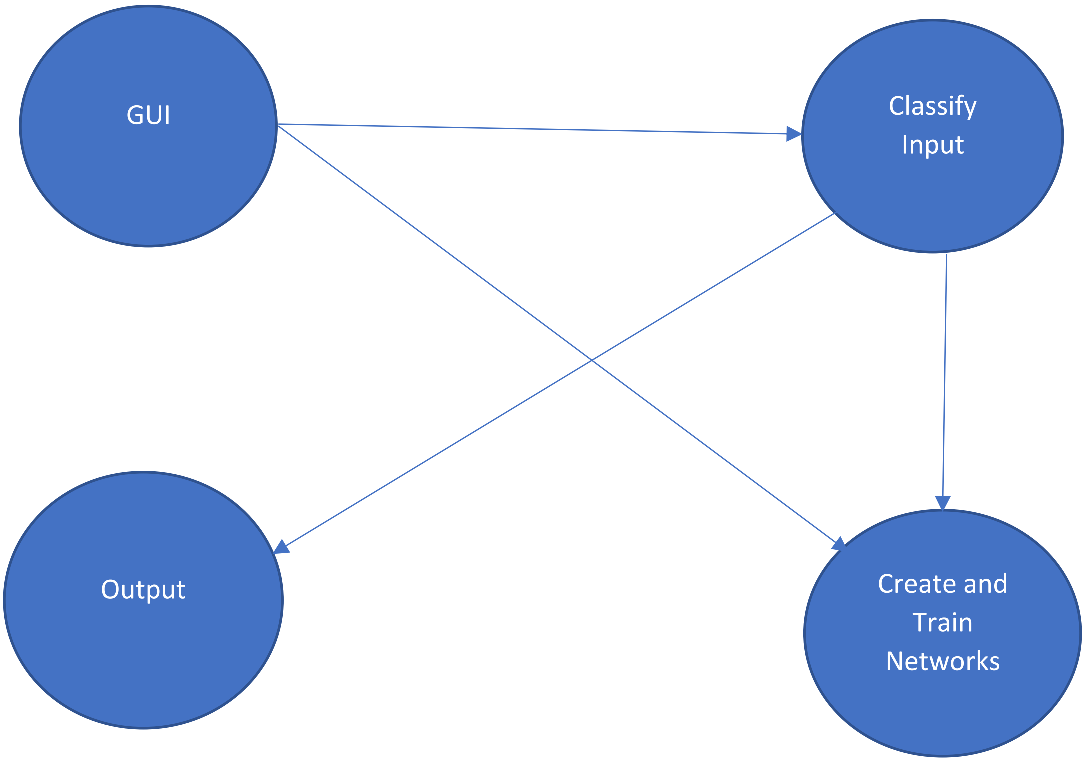
This function will look for networks in the current file system, and if it cannot find them, it will invoke initTrainNetworks and will save the output of that function in order to create freshly trained NNs.

mpaper.m and ocr_fun.m had small changes in order to fit our project.

Training dataset, Target matrix, and other related files are already available in the file system.

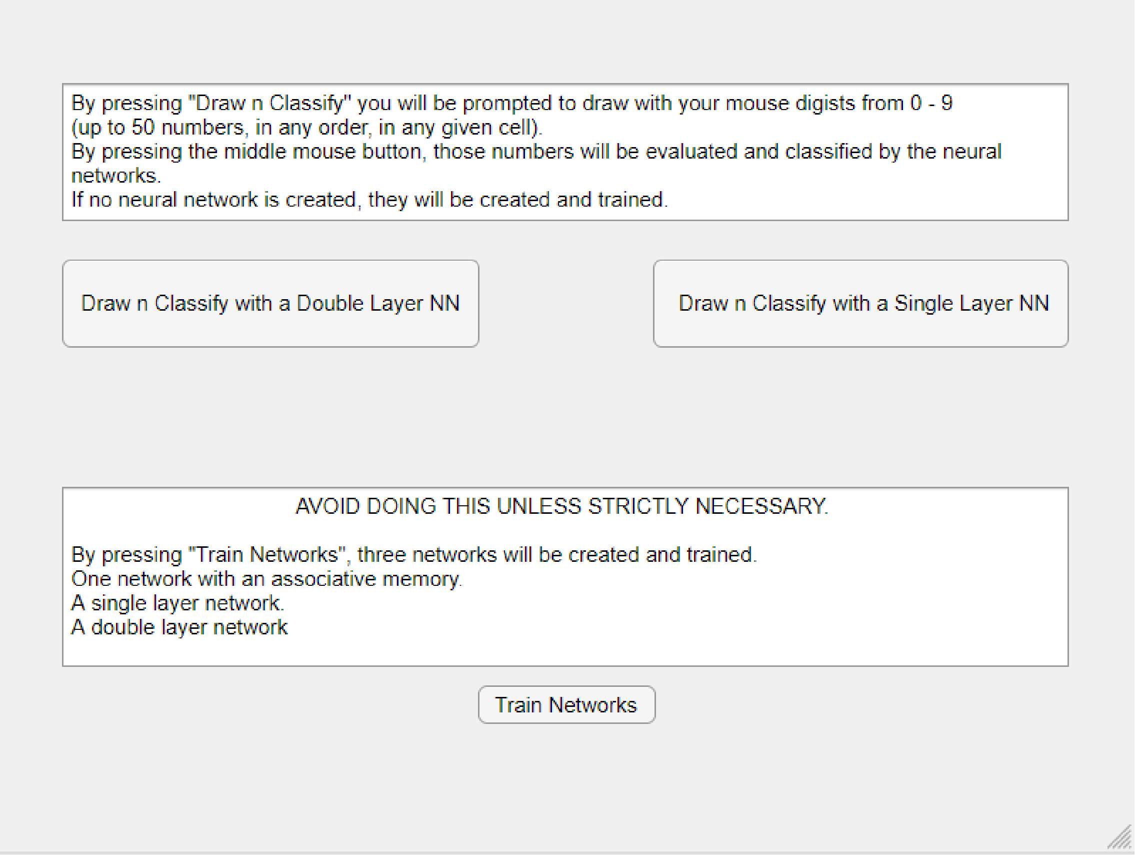
An additional function (test.m) was used to calculate the models performance

The flow of the application can be traced to the following:



4.GUI

An easy and intuitive GUI was developed for this project.

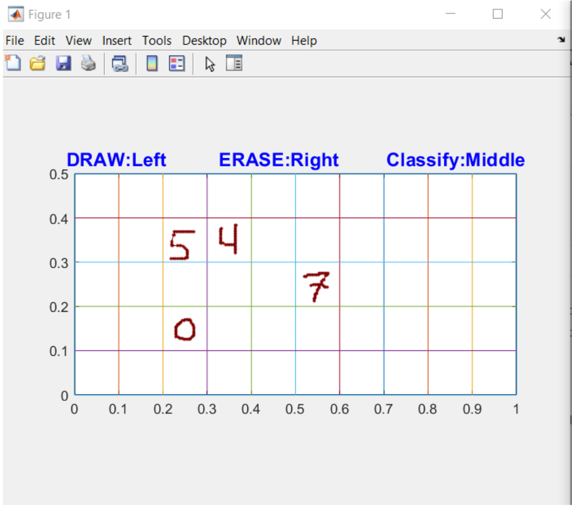


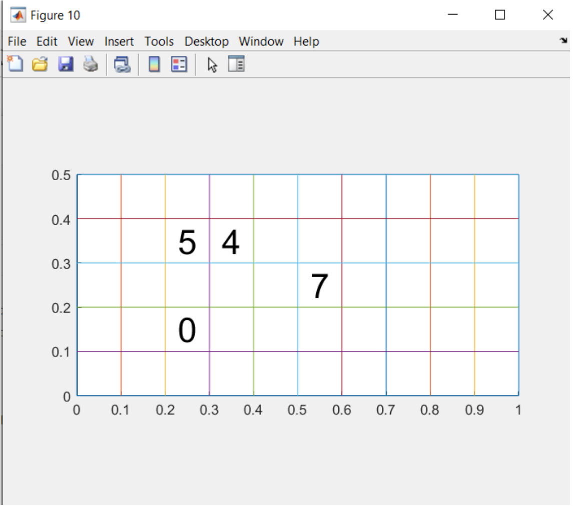
By pressing "Draw n Classify" you will be prompted to draw with your mouse digists from 0 - 9 (up to 50 numbers, in any order, in any given cell).
By pressing the middle mouse button, those numbers will be evaluated and classified by the neural networks.
If no neural network is created, they will be created and trained.

Draw n Classify with a Double Layer NN
Draw n Classify with a Single Layer NN

AVOID DOING THIS UNLESS STRICTLY NECESSARY.

By pressing "Train Networks", three networks will be created and trained.
One network with an associative memory.
A single layer network.
A double layer network





In order to start this GUI, please follow the steps mentioned below:

1. Start your MATLAB software
2. Execute the app ‘OCR’
3. An interactive window will popup with three buttons
4. By pressing ‘Draw n Classify with a Double Layer NN’, you will be prompted to draw with your mouse up to 50 Arabic numerals and it will be classified (pressing the middle mouse button) using the Double Layer Network with the parameters described in this report
5. By pressing ‘Draw n Classify with a Single Layer NN’, you will be prompted to draw with your mouse up to 50 Arabic numerals and it will be classified (pressing the middle mouse button) using the Single Layer Network with the parameters described in this report
6. By pressing “Train Networks”, three networks will be created and trained (Binary Perceptron as filter, Single Layer NN, Double Layer NN).
   Avoid using this option unless strictly necessary. It is meant to be used as troubleshoot in case the networks cannot be created / loaded in the other buttons

To be noted that, if there are no NN available, the GUI will automatically created them and train them before prompt you to drawn any Arabic numerals.

5.Conclusion

With this practical assignment, we were able to get a good grasp in the basic knowledges of a neural network and the related training and learning algorithms.

A few points are worth to mention regarding findings during this assignment:

- The dataset presented for training plays a huge role. We noticed a huge difference in accuracy increase since we switched from a 500 character dataset to a 1000 character dataset.
- The dataset should include different types of numerals, meaning this that not only one person should draw the numerals for the dataset. We noticed, again, a huge increase in accuracy from the moment that more that one person draw numerals into the data set.
- The best combination for a double layer NN, is having sigmoidal activation functions in the first layer, and having a linear activation function in the second layer.
- A classifier with one layer only, although it is less accurate than a double layer one, it can do the job with really good results.
- The binary perceptron acting as a filter could not give us any improvement in accuracy. It dropped the accuracy of a sigmoidal single layer NN from around 90% to 50%
- We can conclude that in order to have a good classifier, the dataset is something that needs a lot of focus. It is mandatory that the dataset is extensive, and with enough data to cover unusual cases. We can also conclude that the linear activation functions have a huge decrease in performance when used a activation from first layers.

There are still some points that we will explore in the future, as for instance, trying to apply a classifier to recognize similarities between imagines and see it can classify correctly the classes. We are definitely looking forward to see if the performance of the algorithms will be in the same order.