University of Coimbra

Machine Learning Report

Optical Character Recognition

Authors:

Adolfo Pinto, nº 2013138622, uc2013138622@student.uc.pt
Jani Hilliaho, nº 2016167354, uc2016167354@student.uc.pt

Faculty of Sciences and Technology
Department of Informatics Engineering

October 2016

Introduction

The purpose of this project was to develop neural network models for character recognition problems. The characters to be recognized by the neural network are the already known ten Arabic numerals, specifically, 1,2,3,4,5,6,7,8,9 and 0.

To aid in the identification of the digits, we assume that each character is defined by a matrix composed of binary elements, in this case 0 and 1. The digits are defined as a 16x16 matrix.

To illustrate the idea, the following matrix can represent the digit 0 (zero) supposedly manually traced by some user in some device:

```
0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 1 1 1 1 0 0 0 0
0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0
0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0
0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0
```

With the developed application, we will comparatively study two kinds of character recognition architectures, as shown in the pictures below:

- Associative memory + Classifier

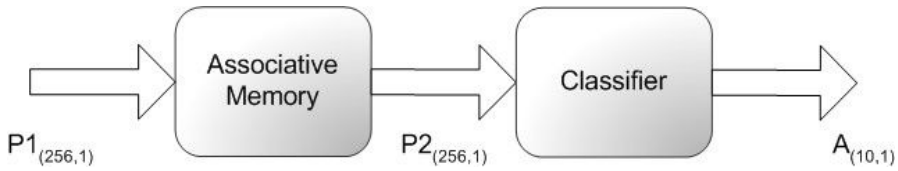

Figure 1. - Associative memory followed by a classifier

- Classifier

Figure 2. - A single classifier

Architecture

The main goal of this project is to correctly identify the characters drawn by the user, using for that a associative memory and classifier, in the first case, and only a classifier in the second one. Each character that the user draws, will belong to one of ten possible classes ([1,2,3,..,9,0]). Following this logic, for example, if the number drawn by the user is classified as the number five, then it will be placed in the class five.

The first model implemented in this project was **associative memory (AM) plus a classifier**. Here, the AM takes as input the characters to be classified, and tries to "correct" them, providing a better approximation to the perfect output. Then, the classifier takes the output of AM, and with the help of a trained artificial neural network, it will classify each number from the input and give to it a classification that is presented to the user.

The second and final model implemented was just a **classifier** that takes as a direct input the characters. Unlike the first step of the previous case, the characters are not "filtered" or "corrected" Here, the digits will be classified using an artificial neural network, same as the second step of the previous case.

The following picture describes the internal architecture of the application:

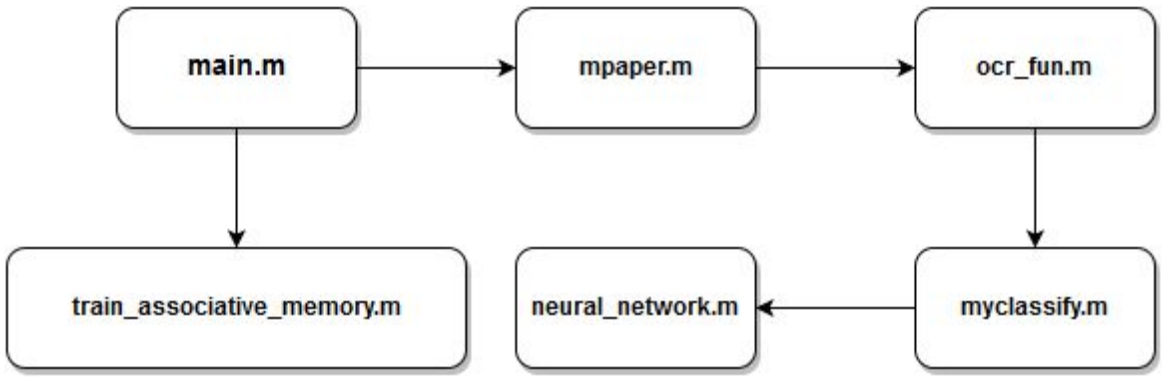

Figure 3. - internal architecture of the application

The application can be used by running the main.m file in MATLAB. Starting in the main.m file, the application will ask the user if he wants to run the experiment with or without the associative memory. If the user chooses to include the AM, its weights are evaluated using the pseudo-inverse method.

In the next step the user will be asked for the number of training cases to run with. The application has two options: 150 and 500 numbers.

In the next step the user can choose to draw the characters manually with a mouse, or to load pre-drawn characters. Pre-drawn characters should be in a file named 'user_input.mat'. The user can create this input file with *mpaper* function, which saves characters to file 'P.mat'. The user have to rename that file manually to 'user_input.mat' before using it with the application.

After selecting the input data, the user is asked to select an activation function to use in the neural network. The user has to select it from three choices: hardlim (binary), purelin (linear) and logsig (sigmoidal).

After selecting the activation function, the neural network will be trained with selected amount of training cases, and the input characters will be evaluated using the *ocr_fun* function.

As the figure 3 suggests, the *ocr_fun* function will call *myclassify* to apply the trained classifier neural network to the data (*neural_network.m*). Some functions from the Neural Network Toolbox, MATLAB, were used to create, train, visualize, and simulate neural networks.

Results

The neural networks with and without associative memory were trained with 150 and 500 cases and with activation functions hardlim, logsig and purelin. Networks were tested by giving the characters in image 1 as input and counting correct, incorrect and undetected numbers from the output data. An example of output data is shown in image 2.

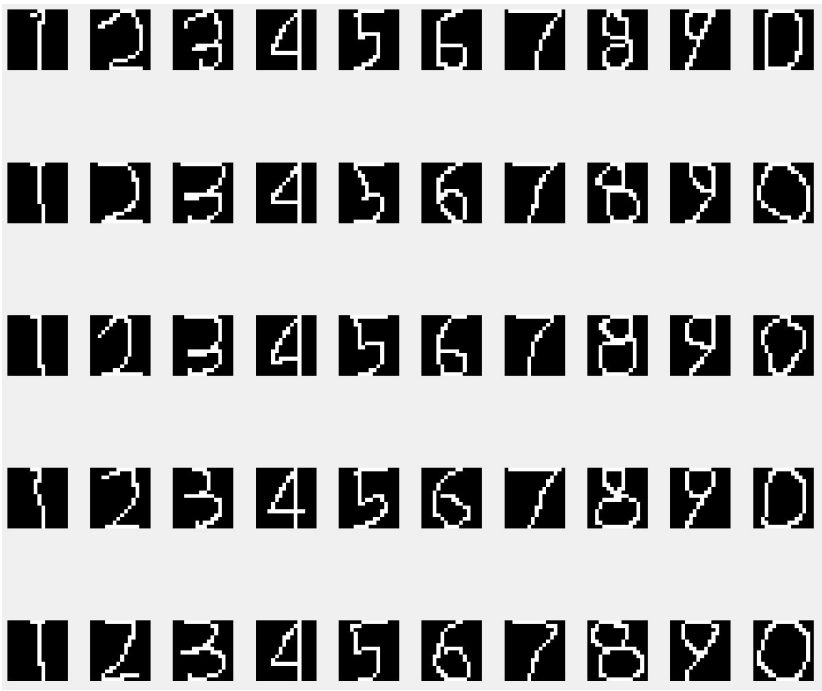

Image 1: Input characters

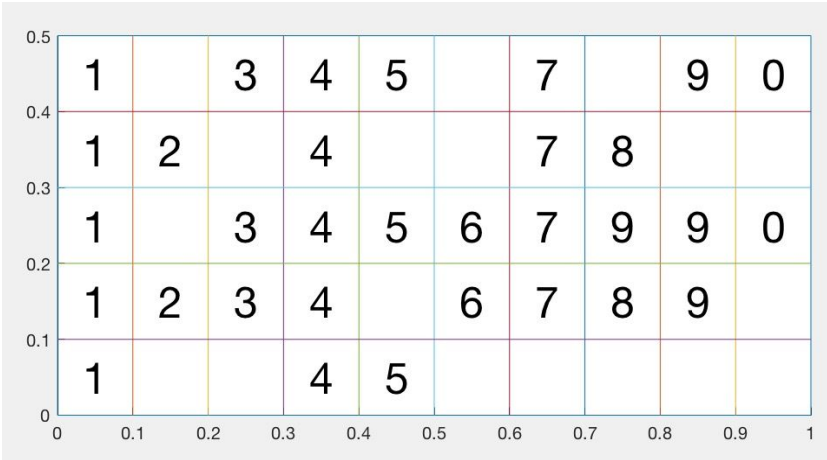

Image 2: Output data example

The neural network was tested with all possible combinations of three activation functions, two architectures and two different amount of training cases. The neural network was therefore tested with 12 different set of properties, each time with 50 input characters. The network was trained with max. 1000 iterations. In some cases the training achieved the performance goal and stopped before 1000 iterations. Table 1 shows the results with associative memory, and table 2 shows the results without associative memory.

RESULTS WITH ASSOCIATIVE MEMORY						
Amount of training cases	**150**			**500**		
Activation function	**Hardlim**	**Logsig**	**Purelin**	**Hardlim**	**Logsig**	**Purelin**
Iterations	**1**	**1000**	**9**	**20**	**1000**	**1000**
Time	**00:00:00**	**00:03:35**	**00:00:02**	**00:00:07**	**00:12:39**	**00:10:12**
Correct numbers	**10**	**28**	**27**	**31**	**46**	**43**
Incorrect numbers	**10**	**22**	**23**	**2**	**4**	**7**
Undetected numbers	**30**	**0**	**0**	**17**	**0**	**0**

Table 1: Test results with associative memory

RESULTS WITHOUT ASSOCIATIVE MEMORY						
Amount of training cases	**150**			**500**		
Activation function	**Hardlim**	**Logsig**	**Purelin**	**Hardlim**	**Logsig**	**Purelin**
Iterations	**6**	**1000**	**1000**	**9**	**1000**	**1000**
Time	**00:00:00**	**00:02:55**	**00:03:28**	**00:00:03**	**00:10:53**	**00:10:39**
Correct numbers	**26**	**40**	**26**	**36**	**44**	**43**
Incorrect numbers	**6**	**10**	**24**	**3**	**6**	**7**
Undetected numbers	**18**	**0**	**0**	**11**	**0**	**0**

Table 2: Test results without associative memory

Results in tables 3 and 4 are calculated from the data in tables 1 and 2. Table 3 shows the amounts of correct numbers by different neural network properties.

Because of 12 different combinations, the network was tested with 12 different set of properties with 50 input characters and 12 tests * 50 characters makes total of 600 tests with single characters. This amount of tests was divided by different properties to create the table 3.

Examples:

- From all characters tested with logsig activation function, 103 / 200 were detected correctly.
- From all characters tested with AM, 185 / 300 were detected correctly.
- From all characters tested with network trained with 150 cases, 157 / 300 were detected correctly.

Amount of correct numbers					
By activation function		**By structure**		**By the amount of training cases**	
Hardlim:	**103 / 200**	**With AM**	**185 / 300**	**150:**	**157/300**
Logsig:	**158 / 200**	**Without AM**	**215 / 300**	**500:**	**243/300**
Purelin:	**139 / 200**				

Table 3: Amount of correct numbers

As shown in table 3, the total amount of correct numbers was 55% higher with 500 training cases than with 150 training cases. As shown in table 4, the training time varies with different amounts of training cases. The total training time with 500 training cases was 4,45 times as long as with 150 training cases.

Total training time with different amount of training cases	
150 training cases	**600 s**
500 training cases	**2673 s**

Table 4: Total training time with different amounts of training cases

The average amount of correct numbers was higher with associative memory than without it. However, based on results in tables 1 and 2, the highest amount of correct numbers was a result of neural network with associative memory, logsig activation function and 500 training cases, which detected correctly 92% (46/50) of the input characters. As shown in tables 1, 2 and 3, the logsig activation function always detected correctly much more numbers than the other activation functions.