Practical Work **2**

OCR – Optical Character Recognition

Aprendizagem Computacional (MEI)

Adaptive Computation (MEI)

CNSD (MIEB)

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

September, 2014

(adapted from a previous work plan authored by Jorge Henriques)

Index

1. Digits recognition

1.1 Problem definition

Neuronal networks models will be developed for character recognition problems. The characters to be recognized are the 10 Arabic numerals:

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

1.2 Characters definition

It is assumed that each character is defined by a matrix composed of binary (0/1) elements. In particular, we assume that the digits are defined as a 16x16 matrix. For example, the following matrix can represent the digit **0** supposedly manually traced by some user in some device (for example in a PDA sensitive screen)

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

Note:

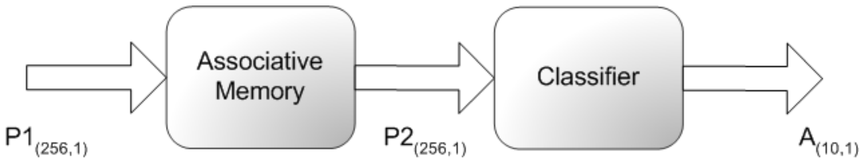
We will use the supplied Matlab function **mpaper.m** to convert a character into a such a binary matrix . See inside it the user instructions.

2. Neural network architectures

Two neural networks architectures will be comparatively studied:
i) both associative memory+classifier and
ii) only a classifier.

2.1 Associative memory + classifier

Two neural networks are considered, serially connected:



Associative memory

The first module, the associative memory, has as input the vector P1 (dimension 256,1), that defines the character to be classified, corresponding to the binary matrix (16,16).
This associative memory can be seen as a "filter" or "corrector": if the input character is not perfect, the associative memory has the capacity to provide an output P2 (dimension 256,1) that is a "more perfect character".
The associative memory is a neural network (see chapter 4) consisting of:
One single layer
Linear activation functions
Without bias

It is characterized by:

$$P_2 = W_P \times P_1$$

The weights, *Wp* (256,256), are evaluated using the pseudo-inverse method or the Hebb rule,

$$W_P = T \times pinv(P)$$

where *T* (256, Q) are the desired Q outputs, for a given *P inputs* (256,Q).

Classifier

The second neural network module is the classifier. The input (P2) is the output of the associative memory. The output (A) is the class where the digit belongs.

For the digits to be classified { '1' '2' '3' '4' '5' '6' '7' '8' '9' '0'}, the following classes are assumed { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.

It is assumed that the classifier is a neural network (see chapter 5) consisting of:

- One single layer
- A linear or non-linear activation function, namely
    - i) perceptron,
    - ii) linear,
    - iii) sigmoidal,
- With bias in each neuron.

It is characterized by:

$$A = f(W_N \times P_2 + b)$$

where matrix W_N has dimensions *(10*,256) and b dimensions (10,1). The input (P2) is a vector of dimension (256,1). The output (A) has dimension (10,1).

Concerning the activation function there are three alternatives:

Perceptron

$$A = \mathrm{hardim}(W_N \times P_2 + b)$$

Linear

$$A = W_N \times P_2 + b$$

Sigmoidal

$$A = \mathrm{logsig}(W_N \times P_2 + b)$$

For example, considering as input the digit defined in section 1.1 (zero) the output should belong to class 10 :

$$A = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The neural network parameters should be evaluated using the perceptron rule, if harlim is used, or the gradient method if purlin or logsig (see chapter 4) are used.

2.2 Classification

In this case only a neural network is considered, as shown in next figure.

Classifier

The characters (vector 256, 1) are directly provided to the classification system.

There is no pre-filtering of the data. The classifier must have a considerable generalization capability.

3. Matlab Implementation notes

One neural network is defined in Matlab, as an object, by a structure with fields to give all the needed data for training:

Architecture:

- sub-object properties define the architecture (inputs, layers, outputs, targets, biases, input weights, layer weights)
- network properties (number of inputs, number of layers, bias connect, input connect, layer connect, output connect, target connect)
- training properties (training function, training parameters). For each implemented training algorithm (and they are numerous) there are proper fields to give its needed parameters.

Numeric data is saved after Matlab 7 as .mat files. Save and load functions create and load .mat files by default. However there is a tool that can convert several other data types, namely .dat and ascii files.

(see Neural Networks Toolbox Manual, Chapter 10 (2012a or after)- Network Object Reference).

In this work some auxiliary functions may be used. Read the files and pay attention to the several comments and instructions in them.

mpaper: to write using the mouse a set of digits and to convert them to a binary vector of inputs (256, Q).

grafica (X,Y,Z): to show until three draft digits **X, Y, Z** (256, 1)

showim(P): to show all Q digits - **P**: (256, Q) or T (256, Q) in a 5x10 image matrix.

ocr_fun (data): calls the classifier and plots the numerals of the result in a 5X10 grid.

The file PerfectArial.mat contains the perfect coding of the ten digits in a structure called *Perfect*. To see them type showim(Perfect).

(**mpaper, showim and ocr_fun** have been adapted from the Statistical Pattern Recognition Toolbox, (C) 1999-2003, written by Vojtech Franc and Vaclav Hlavac, http://www.feld.cvut.cz, Faculty of Electrical Engineering, Czech Technical University Prague)

Some scripts **should be implemented** by the user:

myclassify: to perform the classification

3.1 Training Process

The user must define a training set with a sufficiently high number of cases to allow an effective training. The function mpaper allows to define 50 inputs at once.

The user must also define the target data set, with the desired output for each of the inputs.

In the case of pre-filtering, she/he must trace the perfect numerals using mpaper and then code them in a **T** .mat file or alternatively use the perfect digits present in PerfectArial (note that the structure inside is named Perfect; if you want to see them try showim(Perfect), or grafica (Perfect(:,1), Perfect(:,2), Perfect(:,3)).

In the case of direct classification (without pre-filtering), then the **T** matrix is a set of ten 0/1 digits corresponding to the good answer for each of the 50 inputs (in each *t* vector only one digit is 1, the others are 0).

With 100, 150, etc. inputs the training is more effective. But with 50 inputs, five for each of the ten numerals, the network must already show a good performance.

To show the digits two Matlab functions are available (the first one gives more clear results):

grafica(P(:,k)), will show the k digit

showim(P), shows all 50 digits (this one from the referred Statistical Pattern Recognition Toolbox).

Associative memory training

Matlab provides the **pinv** function for pseudo-inverse of Moore-Penrose.

```
pinv(P)
```

Classsifier definition and training

(some of the following may depend on the NN toolbox version)

Neural network definition

Use **feedforwardnet/newp** function.

```
R2012a:

New Feedforward net:net=feedforwardnet(hiddenSizes,trainFcn)
New Perceptron: net =newp(P,T)

Being necessary to define after the network properties.
```

nA= number of outputs (10 in the present case);

function – activation function

purelin – linear

logsig - sigmoidal

learning – learning method (one layer network)

learngd – gradient rule

learnh – hebb rule

learnhd- hebb rule with decaying weight (see help)

Initialization

Initially the network parameters can be set by:

```
» W=rand(10,256);
» b=rand(10,1);
» net.IW{1,1}=W;
» net.b{1,1}= b;
```

Training parameters

It is possible to specify several parameters related to the training of the neural network. Some of them are: (depends on the NN Toolbox version).

```
» net.performParam.ratio = 0.5;   % learning rate
» net.trainParam.epochs = 1000;   % maximum epochs
» net.trainParam.show = 35;       % show
» net.trainParam.goal = 1e-6;     % goal=objective
» net.performFcn = 'sse';         % criterion
```

Training

To evaluate the network parameters Matlab provides the **train** function.

```
net = train(net,P,T);
```

where P is the (256,Q) inputs and T the desired outputs (10,Q).

After the training phase, the final weighs and bias can be accessed by

```
» W = net.IW{1,1};
» b = net.b{1,1};
```

Validation

To validate the neural network it is available the **sim** function. Pt is the testing set.

```
 a = sim(net,Pt)
```

The test set must be different from the training set. Il allows to measure the generalization capabilities of the network. The user must define the test set with the support of mpaper.

4. Conclusions

The report should focus the following:

- **1. Data set**
  - How does the data set influence the performance of the classification system?

- Note that some additional digits (not perfect) should be included into the data set. The classification system should be able to classify perfect digits but also imperfect digits.

- **2. Neural network architecture**

  - Which architecture provides better results: only the classifier or the associative memory+classifier ?
  - Which is the best activation function: hardlim, linear or logsig?
  - Does the Hebb rule perform well ?

- **3. Results**

  - Is the classification system able to achieve the main objectives (classification of digits)? Which is the percentage of well classified digits)
  - How is the generalization capacity? Is the classification system robust enough (to give correct outputs when new inputs are not perfect)? Which is the percentage of well classified new inputs ?

Enjoy yourself with this interesting work. Many devices of daily use recognize characters (hand-written or not) based on a neural network classifier.

15.September.2014.