Machine Learning
Assignment 1 - OCR – Optical Character Recognition

UNIVERSIDADE D COIMBRA

André Carvalho
andrecarvalho@student.dei.uc.pt
no. 2019216156

Paulo Cortesão
paulocortesao@student.dei.uc.pt
no. 2019216517

Class PL2

October 2022

Contents

1 Introduction

This report aims to describe the process undertaken in the development of a classifier for the recognition of written digits. The various relevant phases for the completion of this project will be described, including input generation, code development and structure, and results obtained.

2 Dataset Generation

In Machine Learning, the quality and quantity of data are very important if we want to achieve good performances and results. In this project, the authors used a set of 1000 digits, drawn using the *mpaper* function, as the dataset for the classifier's training. While drawing these digits, special care was given to producing irregular characters, so they could yield a greater diversity in the training process, hopefully making the neural network more robust against imperfect characters, very common in handwritten text.

3 Code development and structure

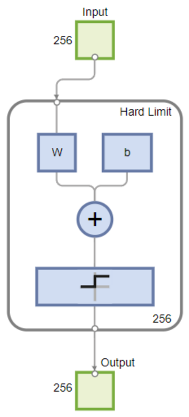
Considering what was given in the statement, the MATLAB API for neural networks was studied and the various classifiers and filters were assembled using it. For the classifiers, 4 main architectures were implemented: Filter + 1 layer NN; 1 layer NN without filtering; 2 layer NN; Pattern network (used as a reference). For the first 3 architectures, multiple activation functions were tested, including *hardlim*, *purelin* and *logsig*.

The possibility of including a softmax layer was also implemented, so that the effect this layer has on classifiers could be quantified.

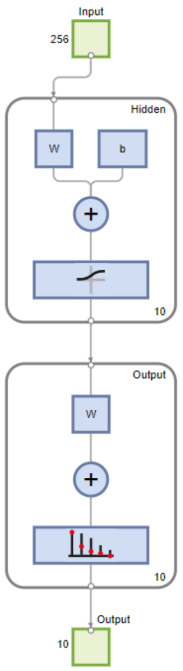
In most of the networks assembled, the MATLAB function *feedforwardnet* was used, since it created a simple feedforward network that could easily be adapted to what was intended. In the one-layer classifiers, the *linearlayer* network was used.

To ease the task of the classifiers, filters were implemented, aiming to obtain more standardized digits from the input. An associative memory filter was built using the pseudo-inverse method. As well as this, a binary perceptron digit was assembled. Their outputs to an example test input can be seen in 3.

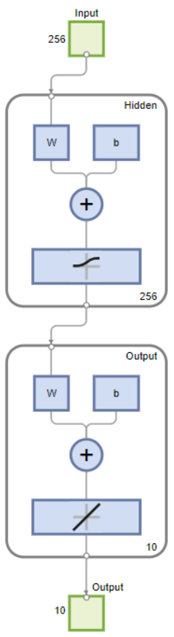
Some network architectures are shown below:



(a) Perceptron architecture



(b) One layer classifier + softmax layer architecture



(c) Two-layer classifier with a sigmoid activation function

Figure 1: Network architecture examples

After the testing phase, an GUI interface was developed to help classify digits after they were drawn using the *mpaper* function [4]. In this interface, the user could choose from the different classifiers proposed, their activation functions and the use of a softmax layer. After this step, the digits the user had drawn would be classified based on the networks trained with a 1000 epochs and its results would pop-up in a new window.

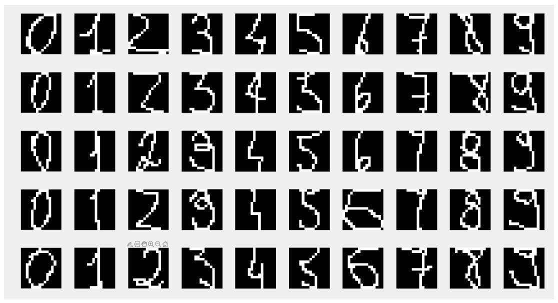
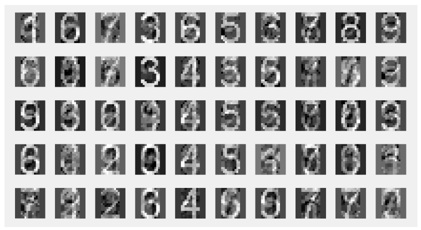


Figure 2: Digits drawn


(a) Associative memory filter output

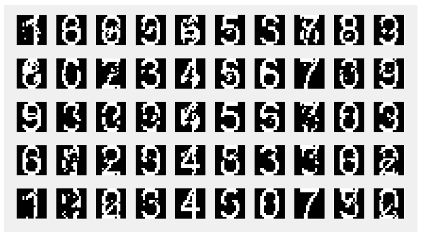
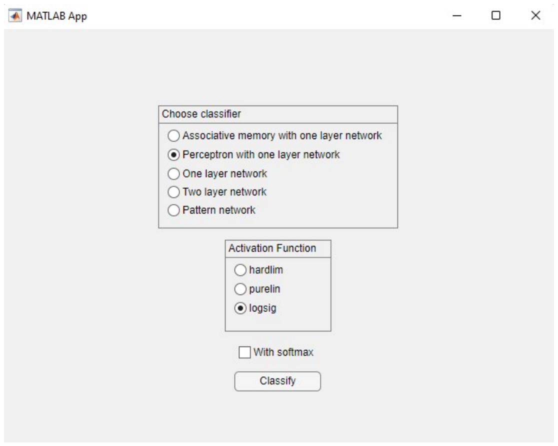

(b) Binary perceptron filter output

Figure 3: Filtering outputs



Choose classifier

- Associative memory with one layer network
- [✓ Perceptron with one layer network]
- One layer network
- Two layer network
- Pattern network

Activation Function

- hardlim
- purelin
- [✓ logsig]

☐ With softmax

Figure 4: GUI interface

4 Discussion

From the results in the appendix, some principles of machine learning were empirically obtained by the development of this project:

- The larger the size of the population, the better the classification accuracy and the generalization capability. As can be seen from the table, doubling the number of training examples led to significant improvements in test accuracy (62% vs. 76% - best results for 500 and 1000 examples, respectively). It is reasonable that these results were obtained, since a larger dataset corresponds to a more representative sample of the problem space, which means the learning algorithm will be able to take into account a greater amount of variations being more adaptable as a result.

- There has to be a significant number of epochs before the accuracy becomes acceptable. As the learning algorithm uses the examples and some kind of error function or gradients to update its weights, convergence is only attained by exposing the algorithm to the examples for many times, so that it is possible to adequately adjust these parameters. What could be seen by these tables is that an increase in the number of epochs led to a significant increase in test accuracy (68% vs. 76% if we consider the best results for 100 and 1000 epochs).

- An excellent accuracy in the training phase does not correspond to an excellent accuracy in testing (and in general use). As can be seen from the various tables, various classifiers achieved a perfect score in the training phase, but then only had a 50-60% score in the testing phase. This can have several meanings: on one hand, there can be insufficient data to represent the space of the classification problem, meaning that it is impossible to attain a good result in this task with the given training dataset. On the other, the network might have reached a minimum in the error function when considering its training inputs, which would reflect an overfitting condition.

- From the results of each filter, we can see that both filters try to improve the imperfect calligraphy of the digits, but in some cases the digits are totally changed, which could result in some misclassification. Overall, the results of the binary perceptron seem to mismatch less digits, which can also be seen in the tests with the 1-layer neural network.

- From the testing accuracy data, it can be said that the main objective of the classifiers is achieved in most cases. Apart from two cases, it becomes evident that the networks learned their task, since their performance is superior to a random classification (significantly greater than 10%).

- From the tests performed, the best architecture is the binary perceptron with one-layer classifier, associated to a sigmoid activation function and a softmax layer; tied with the one-layer classifier with sigmoid activation function and no filter. The occurrence of the sigmoid in these two cases could be attributed to the possibility it has of leading to the representation of nonlinear relationships between inputs and outputs, something that cannot occur with linear and hardlim functions. With 1000 training epochs, these architectures yielded 76% testing accuracy - as such, it can be stated that this is a relatively robust system, with some generalization capability. Bearing in mind what was stated in the previous points, if the number of training epochs and samples were increased (for example, five times), it is expected that this architecture achieved human-like performance.

- The softmax function can have a positive impact on the classification of these digits. As it reduces the output range from $\mathbb{R}$ to $[0, 1]$, theoretically, it allows for a smooth convergence to the optimum configuration in layers whose activation function has that range, such as the linear layer. Still, this was not verified by the experimental results, where the absence of this layer proved beneficial(52% average accuracy when absent vs. 33.2% when present, considering the 3 tables in the appendix).

5 Conclusion

This project allowed for a better understanding of various machine learning concepts, more specifically, neural networks, and the impact of various factors, such as dataset size, number of training epochs, and types of activation functions, in their applicability and utility. As well as this, MATLAB skills in this subject were improved. Finally, in this learning process, performant classifiers were obtained. As such, it is considered that the objectives for this project were attained.

Appendix A

Result tables - training and testing accuracies

In order to test the classification capabilities of the various networks, some independent variables, such as dataset size, activation function, number of training epochs, and type of network were tested. The respective accuracy results are displayed in the following tables.

Classifier	Activation function	Softmax	Train accuracy (%)	Test accuracy (%)
Associative memory with one-layer NN	Harlim	Yes	86.4	54
		No	96.2	**68**
	Purelin	Yes	89.9	58
		No	85.6	64
	Logsig	Yes	95.3	**68**
		No	58.6	46
Binary perceptron with one-layer NN	Harlim	Yes	99.3	62
		No	99.9	64
	Purelin	Yes	98.3	60
		No	57.4	40
	Logsig	Yes	99.7	64
		No	68.7	60
One-layer NN	Harlim	Yes	99.5	66
		No	98.6	64
	Purelin	Yes	24.9	26
		No	88.7	**68**
	Logsig	Yes	47.3	40
		No	98.7	**68**
Two-layer NN	Harlim	Yes	21.9	16
		No	34.5	20
	Purelin	Yes	8.7	2
		No	25.6	10
	Logsig	Yes	15.2	14
		No	87.5	54
Pattern network	-	-	88.7	**68**

Table A.1: Results with 1000 samples and 100 training epochs

Classifier	Activation function	Softmax	Train accuracy (%)	Test accuracy (%)
Associative memory with one-layer NN	Harlim	Yes	88	54
		No	96.9	70
	Purelin	Yes	90.2	58
		No	85.6	68
	Logsig	Yes	96.4	70
		No	86.7	66
Binary perceptron with one-layer NN	Harlim	Yes	100	62
		No	100	74
	Purelin	Yes	100	64
		No	69.7	34
	Logsig	Yes	100	**76**
		No	100	70
One-layer NN	Harlim	Yes	100	66
		No	100	66
	Purelin	Yes	32.3	32
		No	90.4	62
	Logsig	Yes	78.8	46
		No	99.9	**76**
Two-layer NN	Harlim	Yes	61.4	42
		No	34.6	22
	Purelin	Yes	10.7	14
		No	86.1	70
	Logsig	Yes	46.1	32
		No	93.7	62
Pattern network	-	-	87.4	62

Table A.2: Results with 1000 samples and 1000 training epochs

Classifier	Activation function	Softmax	Train accuracy (%)	Test accuracy (%)
Associative memory with one-layer NN	Harlim	Yes	100	52
		No	98	50
	Purelin	Yes	78	44
		No	95.8	50
	Logsig	Yes	98.9	50
		No	99.4	54
Binary perceptron with one-layer NN	Harlim	Yes	100	50
		No	100	30
	Purelin	Yes	20	12
		No	100	60
	Logsig	Yes	100	50
		No	100	54
One-layer NN	Harlim	Yes	100	**62**
		No	100	56
	Purelin	Yes	14.4	18
		No	94.6	52
	Logsig	Yes	52.8	44
		No	100	56
Two-layer NN	Harlim	Yes	57.2	28
		No	80.2	32
	Purelin	Yes	8.2	10
		No	84.4	46
	Logsig	Yes	90.4	44
		No	91.2	54
Pattern network	-	-	89.4	54

Table A.3: Results with 500 samples and 1000 training epochs