Neural network

Alice Maria Catalano 5157341

Abstract—(Artificial) neural networks are networks of several interconnected units (multi-layer) with a simple behaviour used to classify things and make predictions. They're usually trained iteratively. In this tasks execution MATLAB's neural network tools will be used.

INTRODUCTION

a. Task description

The aim of this laboratory is to test the different functions of the Neural Network Tool using the dataset available in different suggested repositories like:

- UCI Machine Learning Repository https://archive.ics.uci.edu/ml/index.php ;
- NIST handwritten digits data set https://www.nist.gov/srd/nist-special-database-19 which contains the already used MNIST handwritten character database.
- Kaggle https://www.kaggle.com/datasets hosts machine learning competitions and has many datasets available, including links to UCI

The first two tasks were two tutorials to get started using the MATLAB tool, using "shallow networks".

The third one is the implementation of the simplest autoencoder network, as example of unsupervised training, since the target set is the input pattern itself. Is required to use the MNIST data set, known from the previous work, in particular the 10-class MNIST digits problem.

The workflow of the experiment is:

- Split the data into subsets of different classes x1, x2, ... , x10
- Create a training set with only 2 classes
- Train an autoencoder on the new, reduced training set
- Encode the different classes using the encoder obtained
- Plot the data using the "*plotc*l" function given.

IMPLEMENTATION

b. Task0: Fitting task

This problem can be approached in 2 ways:

- with the neural net fitting app
- or the command line functions

as suggested by the tutorial I started using the app and then generated the command-line scripts.

The "Body fat data set", which structure is shown in the **model summary** section (Figure 1.a), is the dataset chosen for this tutorial from the sample dataset provided by the tool.

To define the fitting problem, we need to arrange a set of inputs as a vector (predictors), which will be

```
predictors = [0 1 0 1; 0 0 1 1];
responses = [0 0 0 1];
```

the columns in a matrix; then arrange a set of responses in a second matrix.

Once this was defined, training and plotting phase started and the result of the different outputs will be shown in the result section.

c. Task1: Feedforward multi-layer networks (multi-layer perceptron)

In this section a pattern recognition task is implemented. Here too we have two ways to solve the problem and one of them is using the **Neural Net Pattern Recognition** app.
To define a pattern recognition problem, arrange a set of input vectors (predictors) as columns in a matrix. Then arrange another set of response vectors indicating the classes to which the observations are assigned. When there are only two classes, each response has two elements, 0 and 1, indicating which class the corresponding observation belongs to.
After opening the *nprtool* is possible to start the task choosing dataset, trying to choose two different dimensions of them, in this case will be the **Glass Data set** (Figure 1) and the **Thyroid Data Set** (Figure 2).
The thyroid can be used to create a neural network that classifies patient's thyroid as normal, hyperfunctioning or subnormal functioning

In the results section we can find the different scenario generated by the plotting functions.

d. Task2: Autoencoder

Train a multilayer perceptron as an autoencoder for the MNIST data. Following the workflow explained above (section a.), due to the massive amount of data a subset of 2 digits has been chosen, taking 500 random samples for each of them. It was interesting to chose different digits com compare the load of work in learning them for the autoencoder algorithm.
The training and encoding phase are completed thanks to the given functions "*trainAutoencoder()"* and "*encode()"*.
Afterwards the output will be plotted in a 2-dimensional plot given by the 2 units of the hidden layer; also the plotting function was given by the project "*plotcl()"*.

RESULTS

a. Fitting task results

As specified before these results are for the body fat dataset.

I chose as hidden layer size 5, a test data of 25 and a validation data of 15 the training progress result was:

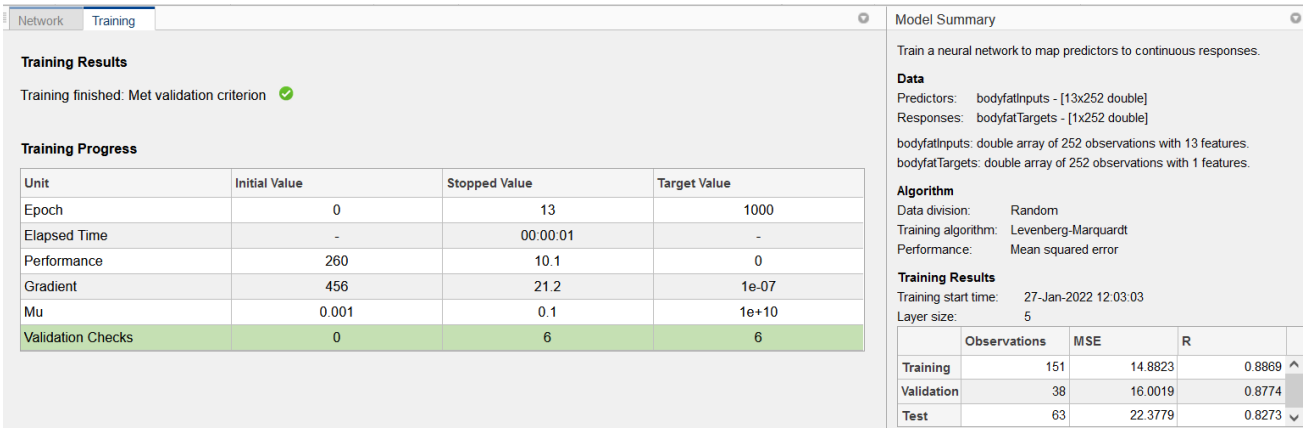


Figure 1.a Training process results for the Body Fat database.

This table shows what happens at the beginning, what happened when the training stopped (after 13 epochs) and what would have happened at the actual end of the training process.

The information given are:

- *The epoch* is the number of times the process passed the whole training set
- *Performance* is the accuracy of the training
- *Gradient* value will never be 0 but we'll see it decrease during the process.
- *Validation checks* how many times during the process the validation is calculated.

After that is possible to evaluate the **performance plot** (Figure 2.a):

Here is possible to notice how the train and test accuracy are a bit distant from each other which means that the random split generated two set with some consistent statistical differences.

For the validation line we can say that the circled point is the minimum value, after that the validation plot is raising and after a bit stopping.

b. Pattern Recognition results

A comparison between the two-dataset chosen will be developed here.
Starting from training the two sets. The same settings for both will be used. Staring with:

- Split data in training, validation, and test set, namely 65%, 10%, 25%.
- Layer size 15 and then 5.

The evaluation of this set will be done on the confusion matrices, to show the classification results for training, testing and validation sets and the results of those data combined, shown in Figure 1.b and Figure 2.b for the Thyroid dataset, and in Figure 3.b and Figure 4.c for the Glass dataset. the accuracy of the outputs can be evaluated by the number of correct classifications

c. Autoencoder Results

In the plots we see the representation of the two digits problem from MNIST database, and the different colors represent the different neurons (2).

As we can see, the bigger is the difference between the shapes of the digits the easier will the be the linear separation (for instance 1-8 and Figure 2.d), otherwise, the points will result mixed, and to hardly separable (for instance 4-9 and 3-8)

IMAGES

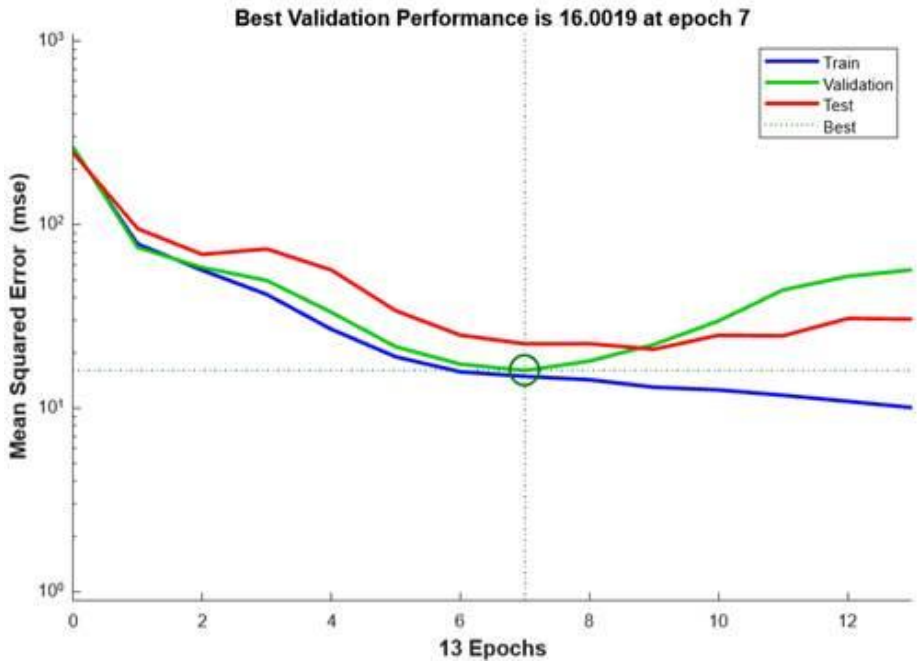


Figure 2.a Validation performance of the fitting problem

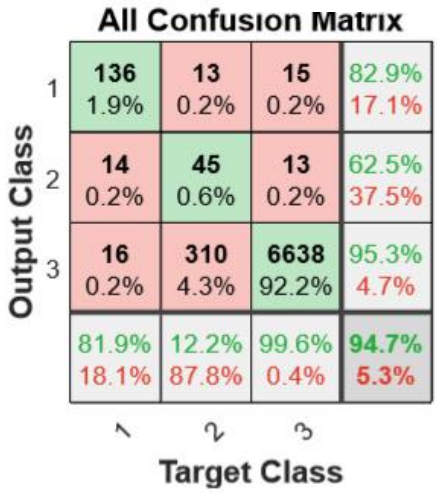


Figure 1.b Confusion matrix "Thyroid set" 15 layers

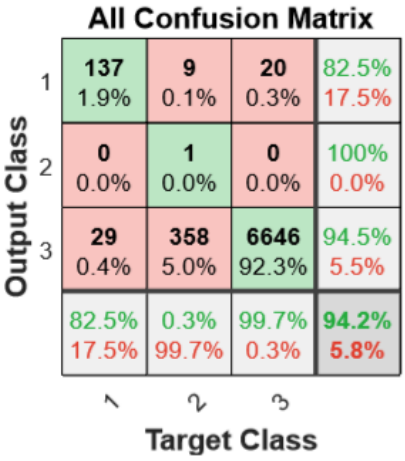


Figure 2.b confusion matrix "thyroid set" 5 layers

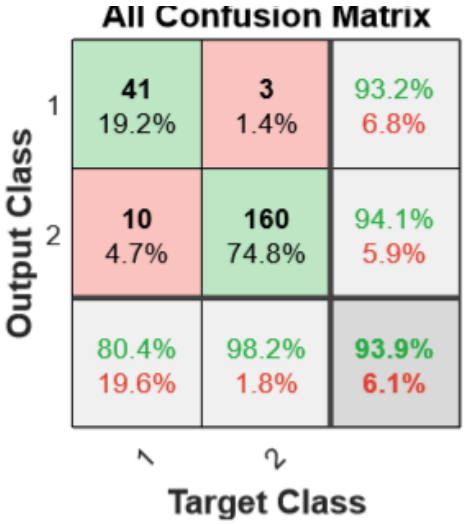


Figure 3.b confusion matrix "glass set" 15 layers

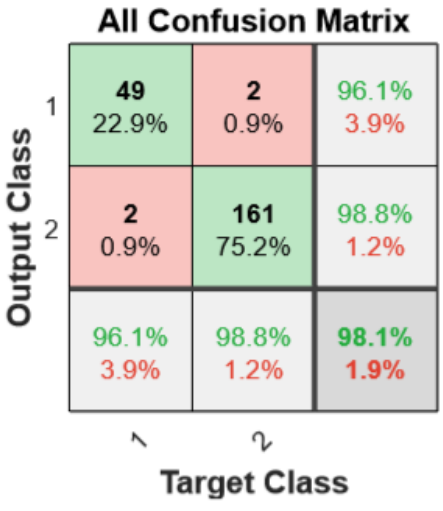


Figure 4.b confusion matrix "glass set" 5 layers

4

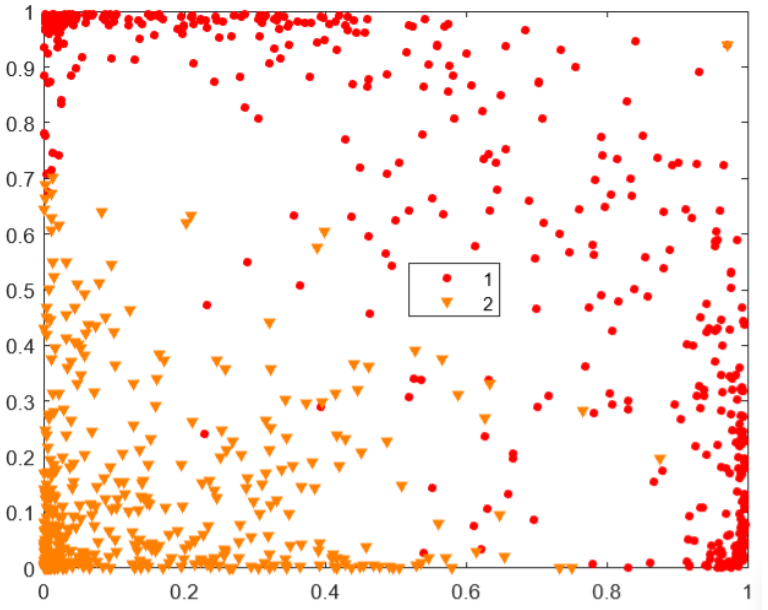


Figure 1.d Autoencoder classes 1 and 8

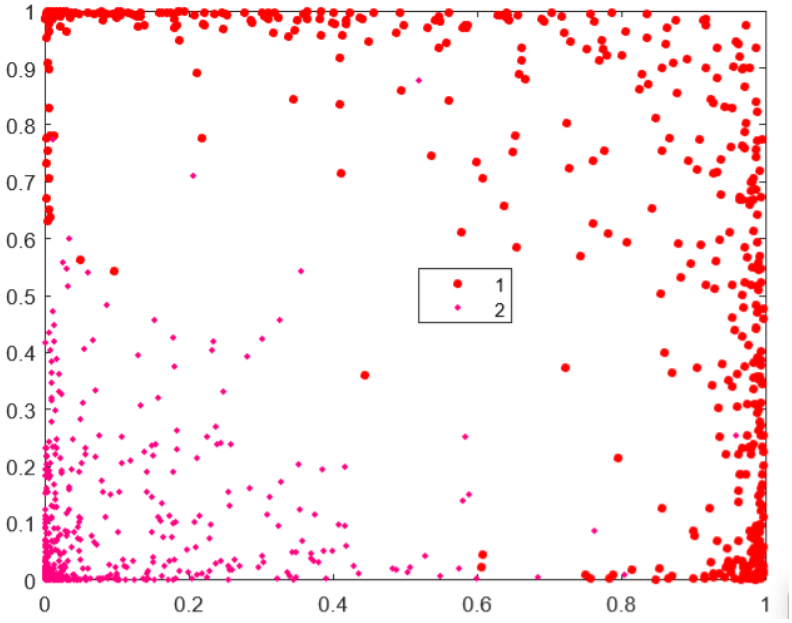


Figure 2.d autoencoder of class 1 and 4

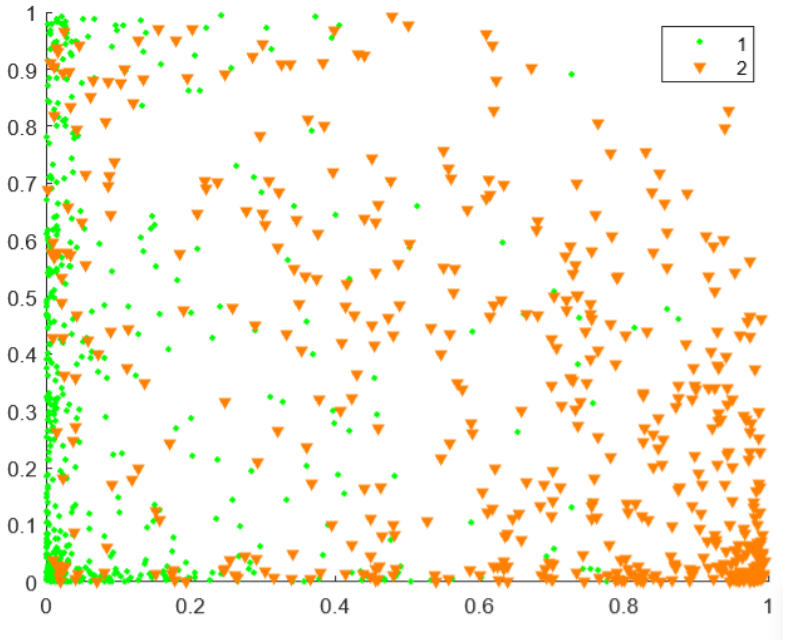
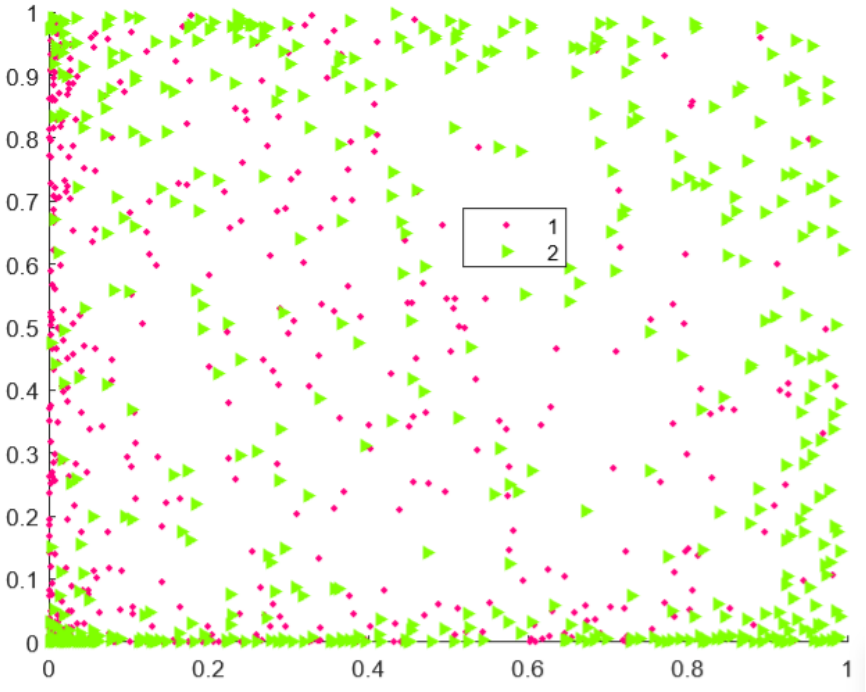


Figure 3.d autoencoder between 3 and 8



4.d Autoencoder between class 4 and 9