Machine Learning - Robotics Engineering
Report Assignment 4
Neural Networks

Francesca Canale 4113133

9/12/2019

Abstract

This assignment is based on neural networks. A neural network is a computing model with a layered structure that resembles the one of neurons in human brain, with layers of connected nodes. A neural network can learn from data, so it can be trained to recognize patterns, classify data, and forecast future events [1].

1 Introduction

The goal of this lab assignment was to use MATLAB's own neural network tools, contained in a library called Neural Networks Toolbox [2].
The work was divided in three tasks:

- **Task 0:** Execute the introductory MATLAB tutorial "Fit Data with a Neural Network" [3], to get acquainted with the toolbox;
- **Task 1:** Execute the MATLAB tutorial "Classify Patterns with a Neural Network" [4], using different data sets and different values of the parameters;
- **Task 2:** Train a multilayer perceptron as an autoencoder.

2 Neural Networks

Neural networks are composed of layers and layers are made of nodes. A node is just a computational place, molded on a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines an input with a set of coefficients, or weights, that can amplify or dampen that input [5]. In Fig. (1) there is a diagram of what one node might look like.

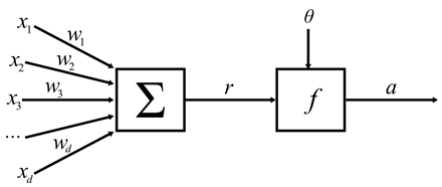

Figure 1: The formal neuron

In mathematical terms we have that:

$$r = x \cdot w$$

$$a = f(r - \theta)$$

where:

- **x** is a d-dimensional vector of input values;
- **w** is the corresponding (d-dimensional) vector of synaptic weights;
- **r** indicates the net input on the neuron membrane;
- **f()** is a non linear function (e.g. heaviside step, signum function, sigmoid, hyperbolic tangent..);
- θ is a threshold;
- **a** indicates the activation value of the membrane potential, is the output of the neuron.

A node layer is a row of those neuron-like switches that turn on or off when the input is fed through the net.
A *multi-layer neural network* (in Fig. (2)) is composed of:

- an input layer, that takes the external input;
- one (or more) hidden layer;
- an output layer, that provides the output of the network.

Each layer's output is simultaneously the subsequent layer's input.

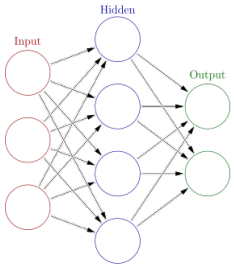

Figure 2: Structure of a multi-layer neural network

2.1 Autoencoder

Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data, then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible. Autoencoder, by design, reduces data dimensions by learning how to ignore the noise in the data [6].
The simplest autoencoder network is a multi-layer perceptron neural network which has one input layer, one hidden layer (with less units than the input one), and one output layer (with as many units as the input layer). The interesting output is the value of the hidden layer because we can expect that similar patterns will have similar representations; for instance, if we take into consideration images of handwritten digits, we hope that images representing a "1" will correspond to very similar representations, and quite similar to "7" but different from "0" or "8". In other words, that the network will learn the classes.

3 Task 1

To implement this task we had firstly to choose some data sets from some possible data repositories. Secondly, to use them to try "Classify Patterns with a Neural Network" MATLAB tutorial executing *NeuralNetworkPatternRecognition()* function. The data chosen are from UCI Machine Learning Repository [7]: Wine [8] and Poker Hand [9] data sets. Before using them we had to adapt their structure to a convention different from the one that we are used to: the function expects patterns as columns and variables as rows. Furthermore, also the target has a different standard: it is a matrix that has as many rows as the number of classes and as many columns as the number of

observations and where a 1 in position (i, j) means that the j-th instance belong to class i. This procedure is implemented in *target_computer()* function. Once the data are well organized, we can proceed executing the tutorial. The *NeuralNetworkPatternRecognition()* function implements a two-layer feedforward network that is used for pattern recognition, with a sigmoid transfer function in the hidden layer and a softmax transfer function in the output layer. The algorithm used to train the network is a scaled conjugate gradient. The data set is randomly divided in training, validation and testing set using respectively the 70%, the 15% and the 15% of the whole data set.

4 Task 2

To execute this task we had to use the MNIST database, a data set composed of images of handwritten digits from 0 to 9. From this very large data set, we selected a subset composed of only two of the ten digits and with reduced dimension (100 randomly chosen instances for each digit). After uploading and processing correctly the data set (adapting its format to the one described in section 3), we used MATLAB function *trainAutoencoder()* [10] to train the autoencoder. One necessary input of this function is the number of unit of the hidden layer: this has been set to 2, so to have a 2-dimensional plot of the output at the end. After this, we had to encode the different classes using MATLAB function *encode* [11]. At the end we had to plot the output of the two hidden layers using function *plotcl()* provided to us.

5 Results

5.1 Task 1

To obtain some experimental results the tutorial was executed for different values of the number of units in the hidden layer for each data set chosen. The results are shown in the form of confusion matrices (Fig. 3 - 4 - 5 - 6 - 7 - 8). The diagonal green cells show the number of cases that were correctly classified for each class, so the true positive results. The off-diagonal red cells show the misclassified cases for each class, so the false positive/negative results. The blue cell in the bottom right shows the total percent of correctly classified cases (in green), so the accuracy, and the total percent of misclassified cases (in red), so the total error.

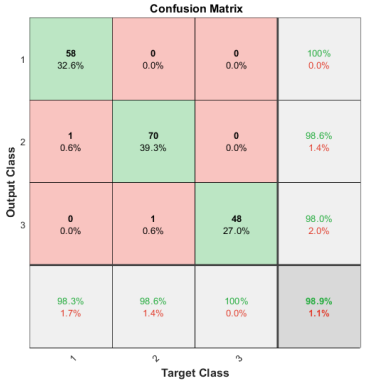

Figure 3: Wine data set with 10 hidden units

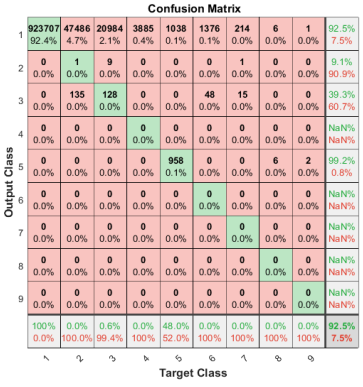

Figure 4: Poker data set with 10 hidden units

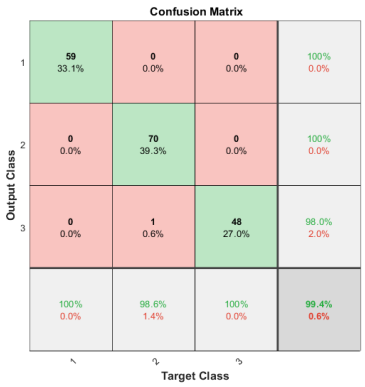

Figure 5: Wine data set with 30 hidden units

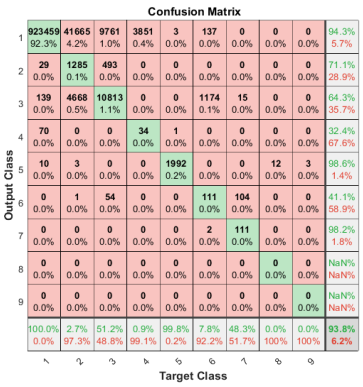

Figure 6: Poker data set with 30 hidden units

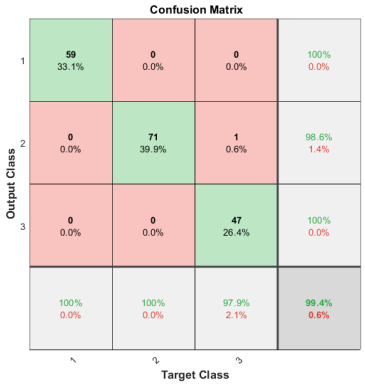

Figure 7: Wine data set with 50 hidden units

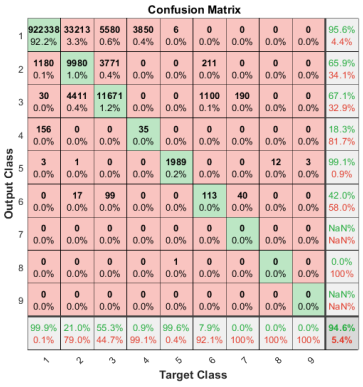

Figure 8: Poker data set with 50 hidden units

From the figure above we can notice that increasing the number of hidden units the results are generally better.

5.2 Task 2

To evaluate the results the task was executed for different combination of two digits. In the following plots (Fig. 9 - 10 - 11 - 12) each different (in color and style) point represent an output of one of the two neuron of the hidden layer and so a different digit.

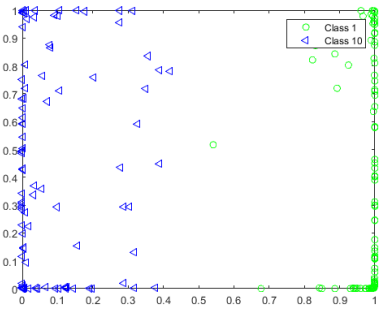

Figure 9: Digit 1 VS digit 10 (0)

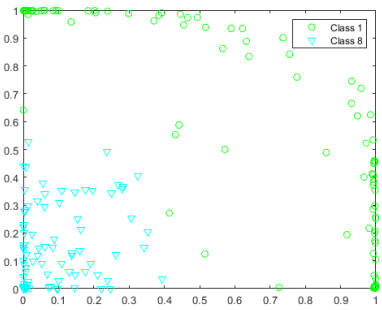

Figure 10: Digit 1 VS digit 8

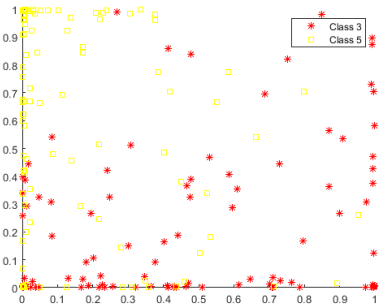

Figure 11: Digit 3 VS digit 5

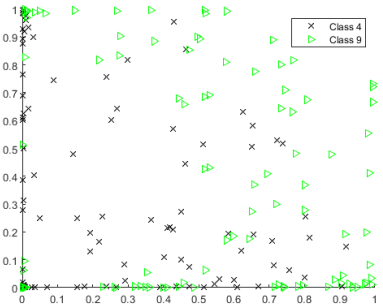

Figure 12: Digit 4 VS digit 9

From the figure above we can notice that when the two digits are very different in shape (e.g. "1" and "0" in Fig. (9) and "1" and "8" in Fig. (10)) the plot produces points that are linearly separable. Instead for more similar handwritten digits (e.g. "3" and "5" in Fig. (11) and "4" and "9" in Fig. (12)) the results are much worse.

References

[1] "What Is a Neural Network?" - MathWorks - https://uk.mathworks.com/discovery/neural-network.html

[2] "Shallow Networks for Pattern Recognition, Clustering and Time Series" - MathWorks - https://it.mathworks.com/help/deeplearning/gs/shallow-networks-for-pattern-recognition-clustering-and-time-series.html

[3] "Fit Data with a Shallow Neural Network" - MathWorks - https://it.mathworks.com/help/deeplearning/gs/fit-data-with-a-neural-network.html

[4] "Classify Patterns with a Shallow Neural Network" - MathWorks - https://it.mathworks.com/help/deeplearning/gs/classify-patterns-with-a-neural-network.html

[5] "A Beginner's Guide to Neural Networks and Deep Learning" - skymind - https://skymind.ai/wiki/neural-network

[6] "Auto-Encoder: What Is It? And What Is It Used For?" - Towards Data Scinece - https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726

[7] UCI Machine Learning Repository - http://archive.ics.uci.edu/ml/index.php

[8] "Wine data set" - UCI Machine Learning Repository - http://archive.ics.uci.edu/ml/datasets/Wine

[9] "Poker hand data set" - UCI Machine Learning Repository - http://archive.ics.uci.edu/ml/datasets/Poker+Hand

[10] "trainAutoencoder" - MathWorks - https://it.mathworks.com/help/deeplearning/ref/trainautoencoder.html

[11] "encode" - MatWorks - https://it.mathworks.com/help/comm/ref/encode.html