# Neural Network

Computer Engineering, Universitat Jaume I

Castellón, Spain

David Dag Mora Zapata
Castellón, Spain
al152047@uji.es

Mihaita Alexandru Lupoiu
La Pobla Tornesa, Spain
al204332@uji.es

***Abstract*— In this article we are going to do a study on neural networks, and how we can create artificial neural networks by using the "PyBrain" modul of Python. We will also explain how to use that module and how we can train the artificial neural network so we can achieve the desired results.**

***Keywords*— artificial neural network; PyBrain; train**

## I. INTRODUCTION

An artificial neural network  is a mathematical model inspired by biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases a neural network is an adaptive system changing its structure during a learning phase. Neural networks are used for modeling complex relationships between inputs and outputs or to find patterns in data. [1]

## II. MULTILAYER PERCEPTRON

The multilayer perceptron is an artificial neural network (ANN) that consists of multiple layers.[Fig 1] That allows it to solve problems that are not linearly separable, which is the main limitation of the perceptron (also called single perceptron). The multilayer perceptron can be entirely or locally connected. In the first case, if it is entirely connected, each output of a neuron of layer "i" is input to all neurons of layer "i +1". While every neuron in the second layer, locally connected, "i" is a number of input neurons (region) layer "i +1". [2]

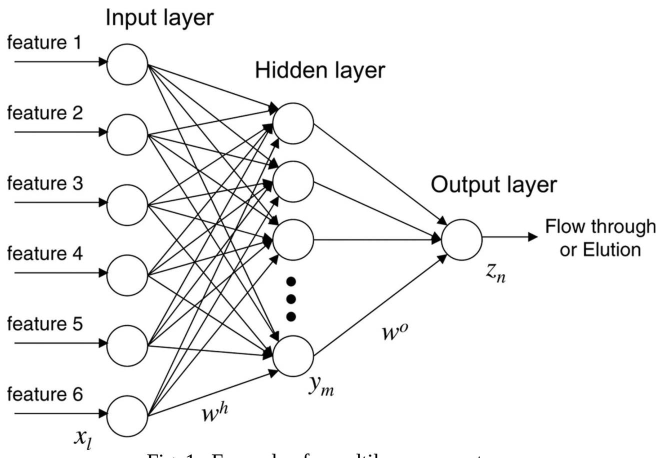


Fig. 1. Exemple of a multilayer perceptron.

The layers can be classified in  three types:

- Input Layer: Composed of neurons that enter the input patterns in the network. In these neurons no processing occurs.
- Hidden layers: Formed by those neurons whose inputs come from the outputs of previous layers and these neurons pass their outputs to the posterior layers.
- Output layer: Neurons whose output values corresponding to the outputs of the entire network.

The backpropagation is a algorithm used for training  these networks, and for that the multilayer perception is known as backpropagation network.

### *A. Limations:*

- The multilayer perceptron does not extrapolate properly, if the network doesn´t train correctly, or not enough, the outputs can be imprecise.
- The existence of local minimums in the error functions, dificults significantly the training, and then after it was reached the minimum workout, it stops. Iven if it diden't reach the convergence rate that was fixed

When we fall in a local minimum without satisfying the percentaje of error allowed, we have considered to change the tipology of network. For exameple we begin training with different initial weights, learned to modify the parameters, ghange the worout set or make another order.

### *B. Aplications*

The multilayer perceptron is used to solve problems of pattern matching, image segmentation, data compression, and so on.

## III. INTRODUCTION TO NEURAL NETWORKS WITH PYBRAIN

Python is a general-purpose, high-level programming language whose design philosophy emphasizes code readability. Its syntax is said to be clear and expressive. Python has a large and comprehensive standard library, and a lot of additional modules (like PyBrain) have been developed by third-parties.

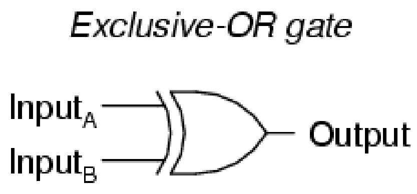
PyBrain[3] is a modular Machine Learning Library for Python. Its goal is to offer flexible, easy-to-use yet still powerful algorithms for Machine Learning Tasks and a variety of predefined environments to test and compare your algorithms (PyBrain is short for Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library).

Fig. 2 PyBrain and Python

Python(x,y) is a free scientific and engineering development software for numerical computations, data analysis and data visualization based on Python programming language, Qt graphical user interfaces and Spyder interactive scientific development environment.

## IV. The XOR Example

The XOR gate (sometimes EOR gate, or EXOR gate) is a digital logic gate that implements an exclusive or; that is, a true output (1) results if one, and only one, of the inputs to the gate is true (1). If both inputs are false (0) or both are true (1), a false output (0) results. [4]



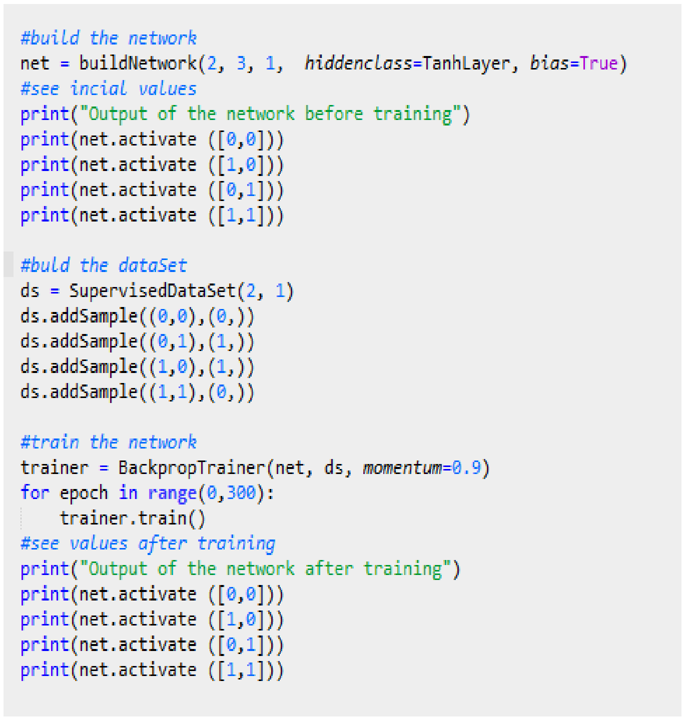
| A | B | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Fig. 3. XOR gate

With this logic gate we started our first contact with PyBrain. Here we learned how to use basic functions of this library (build a network, build a DataSet, train the Network with our dataset).

## V. Ejemplo de Iris

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Fisher (1936) as an example of discriminant analysis. It is sometimes called Anderson's Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species. Two of the three species were collected in the Gaspé Peninsula "all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus".[5]



```
#build the network
net = buildNetwork(2, 3, 1,  hiddenclass=TanhLayer, bias=True)
#see incial values
print("Output of the network before training")
print(net.activate ([0,0]))
print(net.activate ([1,0]))
print(net.activate ([0,1]))
print(net.activate ([1,1]))

#buld the dataSet
ds = SupervisedDataSet(2, 1)
ds.addSample((0,0),(0,))
ds.addSample((0,1),(1,))
ds.addSample((1,0),(1,))
ds.addSample((1,1),(0,))

#train the network
trainer = BackpropTrainer(net, ds, momentum=0.9)
for epoch in range(0,300):
    trainer.train()
#see values after training
print("Output of the network after training")
print(net.activate ([0,0]))
print(net.activate ([1,0]))
print(net.activate ([0,1]))
print(net.activate ([1,1]))
```

Fig. 4. Exeple code of the XOR

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other. t. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

TABLE I. Iris Information

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | ***Multivariate*** | **Number of Instances:** | ***150*** | ***Area:*** | ***Life*** |
| **Attribute Characteristics:** | ***Real*** | **Number of Attributes:** | ***4*** | **Date Donated** | ***1988-07-01*** |
| **Associated Tasks:** | ***Classification*** | **Missing Values?** | ***No*** | **Number of Web Hits:** | ***432322*** |

### A. *Attribute Information:*

1. Sepal length in cm

2. Sepal width in cm

3. Petal length in cm

4. Petal width in cm

5. Class (0 Iris Setosa, 1 Iris Versicolour, 2 Iris Virginica)

The first thing we had to do was to research the sample and see how many inputs and outputs the database has. In this case we have four inputs and one output, but it has three possible values. The only complication we had was to insert the new data, since we had to read a file and then manipulate the information.

Fig. 5. Iris Versicolor and Iris Setosa

The last thing you did in the practice, was to keep the weights of the neurons in an xml file, and then when we start the program, first it will read the status af the network and then improve it, rather than starting from scratch.

Fig. 6. Iris Virginica

## B. *Results:*

- Confusion matrix is:

```
[[ 16.   0.   0.]
 [  0.  11.   1.]
 [  0.   1.   8. ]]
```

- Percentage Correct: 94.5945945946

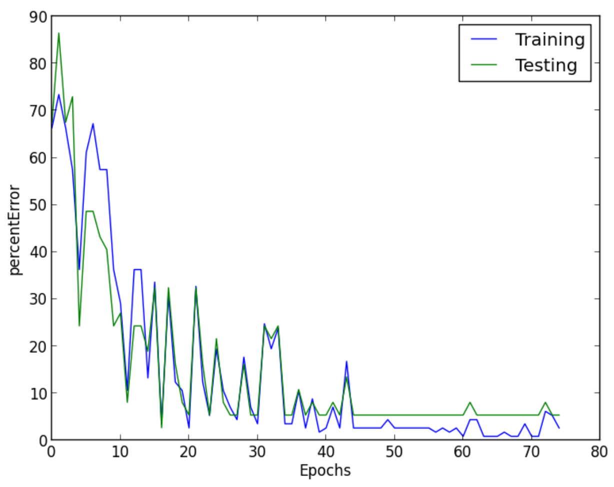


Fig. 7. Graphical display of the error versus the number of epochs in the iris case

The conclusion we have drawn is that the more we train the neurons, the better results we will get when we test it, because the weight of the edges will be closer to the optimum.

## VI. REPOSITORY UCI ML

In this session we had to choose from a group of data sets one with wich we will work. The data set we used is called "Haberman's Survival Data Set".

The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. [6]

TABLE II. HIBERMAN'S

| | | | | | |
|---|---|---|---|---|---|
| **Características del conjunto:** | Multivariable | **Cantidad de instancias:** | 306 | **Área:** | Vida |
| **Características del atributo:** | Entero | **Número de atributos:** | 3 | **Fecha Donación** | 1999-03-04 |
| **Tareas asociadas:** | Clasificación | **Faltan valores?** | No | **Número de visitas en la Web[1]:** | 44989 |

### A. *Attribute information:*

1. Age of patient at time of operation (numerical)
2. Patient's year of operation (year - 1900, numerical)
3. Number of positive axillary nodes detected (numerical)
4. Survival status (class attribute)
   -- 1 = the patient survived 5 years or longer
   -- 2 = the patient died within 5 year

When done the script, the survival status of the patients was modified as follows:

- 0 = the patient survived 5 years or more

- 1 = the patient died within 5 years

In this task we had to make a script very similar to Iris. The neural network was created and trained based on the data set "Haberman's Survival", we got the confusion matrix and the graph with the progress of the error and store it in an XML file network weights.

There were two versions of the program. The first version was done was without doing the normalization of the input and the results were very strange, because it gave the impression that the neurons did not train at any time.

### B. *Results:*

We tried to change the momentum, the hiden number of neurons or the number of epochs but the result was always the same. The script classified all input values as part of the group of type 0 (the patient survived 5 years or more), although in some cases had a slightly different.[ Fig. 8 ]

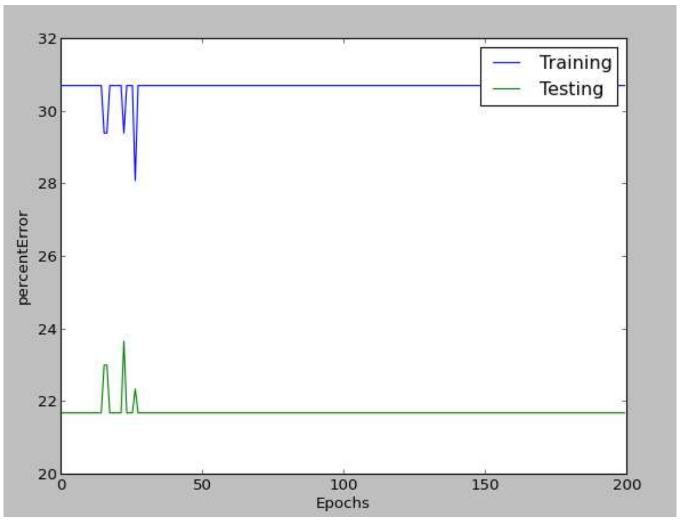


Fig. 8  Graphical display of the error versus the number of epochs in the Haberman's case I

- Confusion matrix is:

[[ 119.   33]

[  0.        0. ]]

- Percentage Correct:  78.22894736842

The reason of this results is because  the error of the output we obtained was very big, and therefore we maked a second version of the program where we use the normalization to improve the  results.

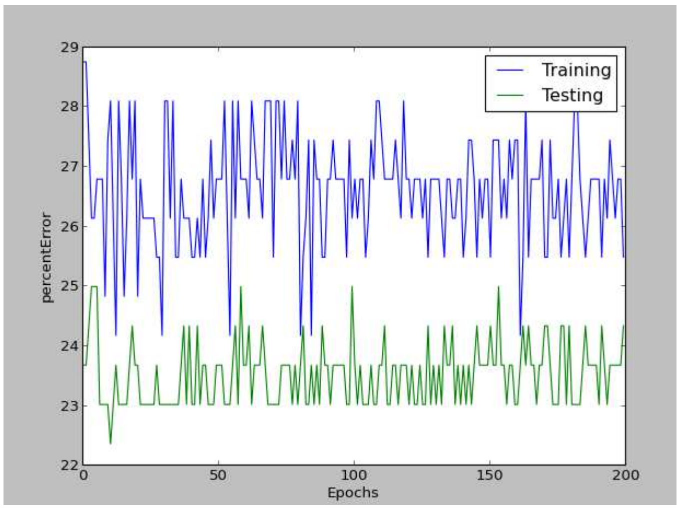


Fig. 9  Graphical display of the error versus the number of epochs in the Haberman's case II

- Confusion matrix is:

[[ 110.   31]

[  6.        5. ]]

- Percentage Correct:  75.657894368

This time we got a different resulting output. The neurons were training and did not classifyed all samples as part of the first type only, but even though many elements were classified correctly, the sample still had  a big  error rate.[Fig. 9]

As a final test we used the entire database to train and test of neurons, and the result was the one expected. Since we train and perform tests on the same data, the graph must overlap and the confusion matrix dose not have to change much with respect to the values obtained from previous tests.[Fig. 10]

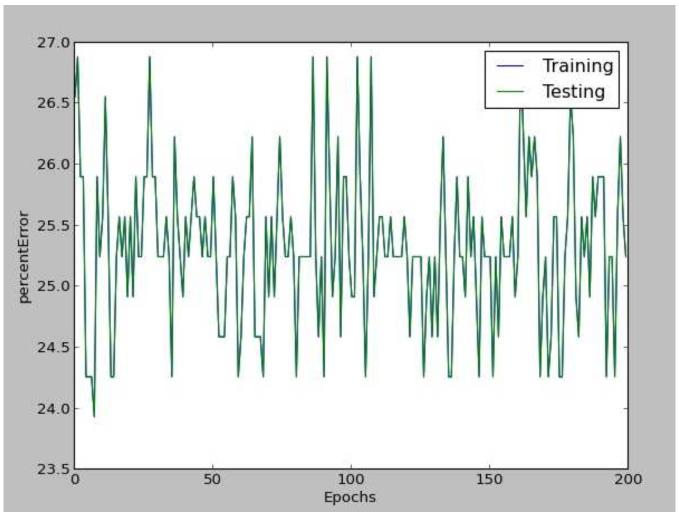


Fig. 10  Graphical display of the error versus the number of epochs in the Haberman's case III

- Confusion matrix is:

[[ 215.   67.]

[  10.      13. ]]

- Percentage Correct:  74.7540983607

With this test we confirmed that the standardization and classification process was done correctly.

The general conclusion at which we arrived was that the error of this problem is not easy to resolve. We have decided that the tests performed on the data sample is more than enough and it's not worth it to keep trying to improve the outcome because of our lack of knowledge.

## VII.    THE RECOGNITION OF TRAFFIC SIGNALS PROBLEM

An automatic road sign recognition system first locates road signs within images captured by an imaging sensor on-board of a vehicle, and then identifies road signs assisting the driver to properly operate the vehicle.

Automated road sign recognition is a difficult task. There are a number of important issues that need to be taken into consideration. These include: illumination conditions, direction of sign’s face, status of paint on signs, placement of multiple signs near each other, torn and tilted signs, variations in sign’s scale, obstacles such as tree, image sensor’s properties, car vibrations, etc.

Assuming that the road sign has been previously located in the image, neural networks may be employed to implement the classification module because they have proven to be good classifiers and have been able to successfully solve several object recognition problems.

The first task we had to do, was to load the images of the traffic Sign in to the program. The imagenes we used are from "The German Traffic Sign Benchmark", that is a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) 2011.[2]

## A. *It has the following properties:*

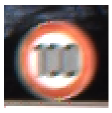
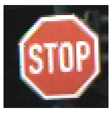
- Single-image, multi-class classification problem
- More than 40 classes
- More than 50,000 images in total
- Large, lifelike database

The training set archive is structured as follows:

- One directory per class
- Each directory contains the training images
- Training images are grouped by tracks
- Each track contains 30 images of one single physical traffic sign

The work was done with a small sub-set of this benchmark, by selecting only one track of each the following 4 classes:

TABLE III. CLASSES OF TRAFIC

| | | | |
|---|---|---|---|
| |  | |  |
| 00003 | 00007 | 00013 | 00014 |

The pixels of the image where the inputs of the neural network, but the resolution of the images are different and the input of the dataset is constant. So we need to make all the images have the same resolution.

Moreover, the number of inputs has a significant input on the cost of training. A compromise is needed, by keeping the resolution low while still making possible the distinction of road signs. The suggestion that we have received, was to use image sizes between 10x10 and 20x20 pixels.

In addition, the original images are stored in RGB format, that is, each images has three pixel planes. We used only the red channel because it contains the most interesting information about the road sign.

## B. *The image processing steps that were used:*

- Crop the image, i.e. select only the part of the image inside the ROI
- Scale the image to a fixed, small resolution
- Select the red channel
- Adjust the contrast with histogram equalization
- Finally, normalize the pixel values to the interval [-1,+1]

We created the classification data set, and the matrix images into a vector before adding them into the dataset. The next step was to build the neural network with the appropriate number of input and output units.

What we had to do was to discover which is the perfect number of hidden neurons. So for that we started with a few units and increase its number if the network did not converge.

The first tests were done with the next sample:

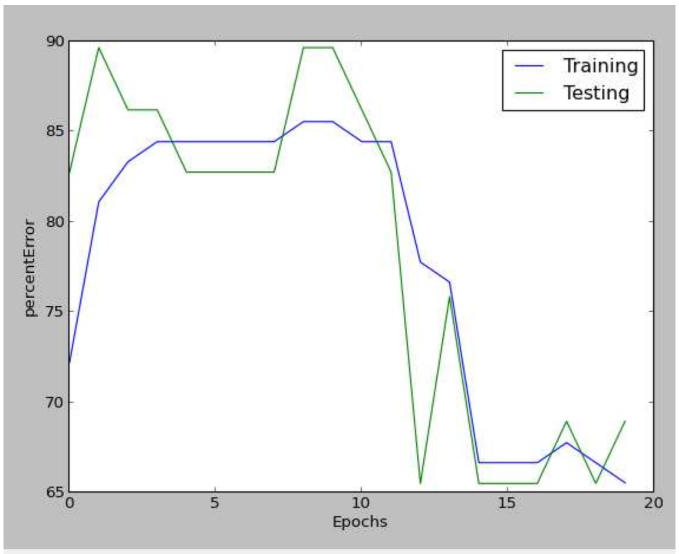
- classes = [3, 7, 13, 14]
- tracks = {3: 5, 7: 40, 13: 24, 14: 8}

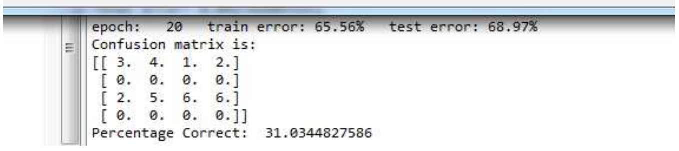
We used it until we found the best configuration and got a 100% success.

## C. *Configuración:*

The first test were made with this configuratión[Fig. 11]:

- división de datos 0.25
- hiden 1
- epoch 20



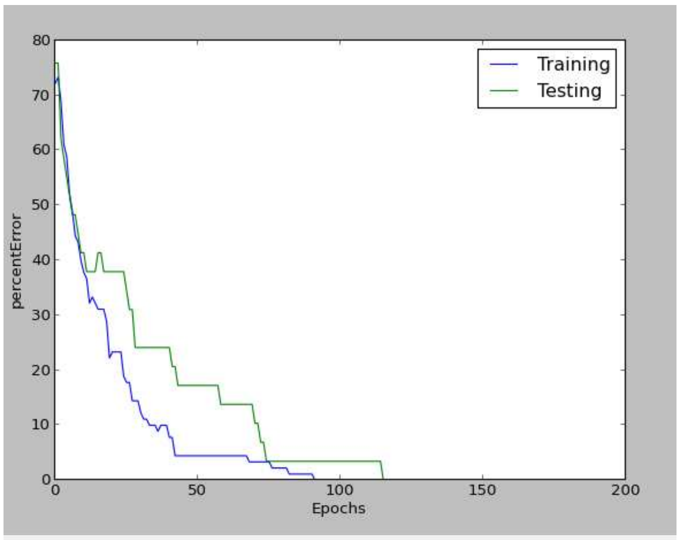


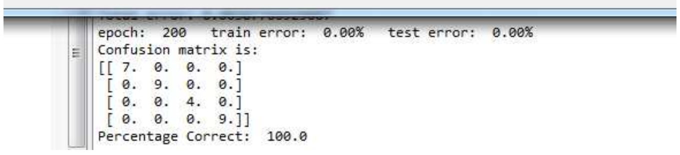
```
epoch:   20   train error: 65.56%   test error: 68.97%
Confusion matrix is:
[[ 3.  4.  1.  2.]
 [ 0.  0.  0.  0.]
 [ 2.  5.  6.  6.]
 [ 0.  0.  0.  0.]]
Percentage Correct:  31.0344827586
```

Fig. 11 Graphical display of the error versus the number of epochs in the Trafic Signals case and the confusion matrix I

The best configuration we found [Fig. 12]:

- división de datos 0.25
- hiden 7
- epoch 200





```
epoch:  200   train error:  0.00%   test error:  0.00%
Confusion matrix is:
[[ 7.  0.  0.  0.]
 [ 0.  9.  0.  0.]
 [ 0.  0.  4.  0.]
 [ 0.  0.  0.  9.]]
Percentage Correct:  100.0
```

Fig. 12  Graphical display of the error versus the number of epochs in the Trafic Signals case and the confusion matrix II

## D. *First Conclusion:*

The more tests performed the better the neuronal network was, and that the optimal number of hidden neurons is seven.

## E. *After finding the optimal configuration, we made some tests with other samples to compare and the result was:*

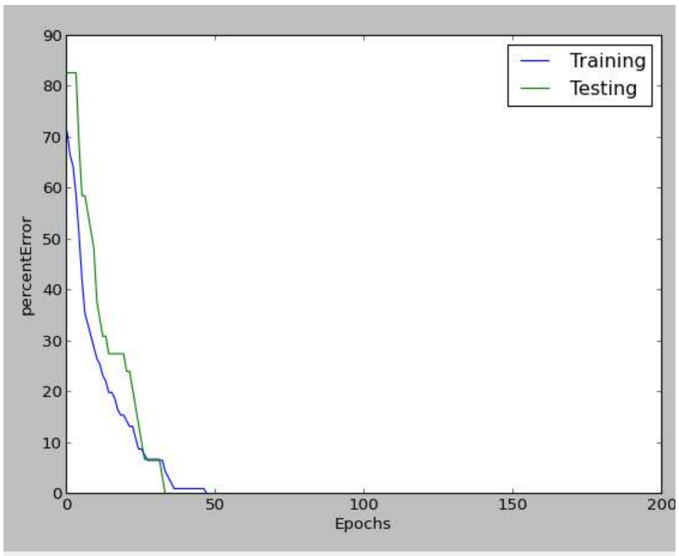
Configuration[Fig. 13]:

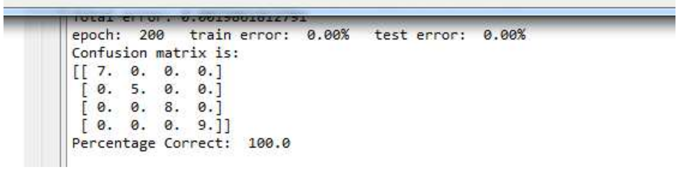
- classes = [3, 7, 13, 14]
- tracks = {3: 15, 7: 20, 13: 35, 14: 4}
- división de datos 0.25
- hiden 7
- epoch 200

Configuration[Fig. 14]:

- classes = [3, 7, 13, 14]
- tracks = {3: 30, 7: 10, 13: 20, 14: 8}
- división de datos 0.25
- hiden 7
- epoch 200

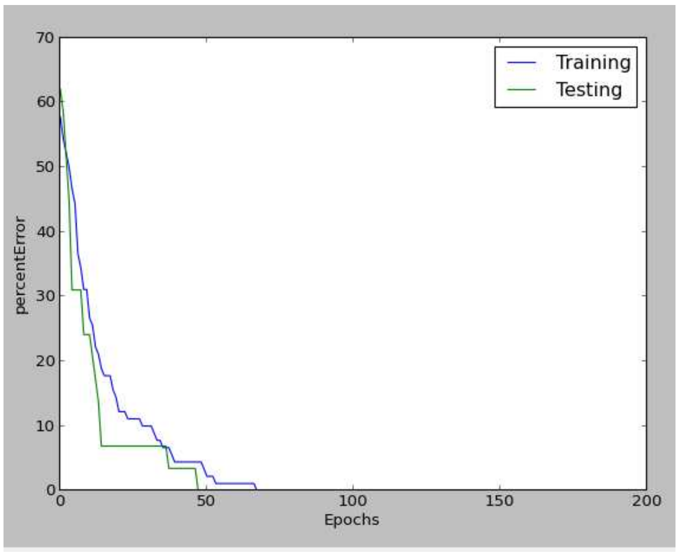
And the result of these test were the one we desired.

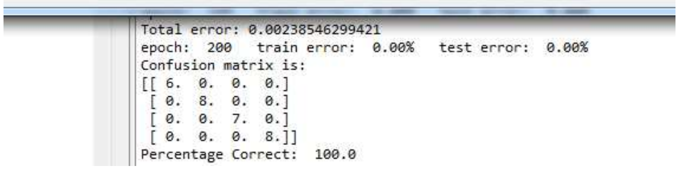




```
epoch:  200   train error:  0.00%   test error:  0.00%
Confusion matrix is:
[[ 7.  0.  0.  0.]
 [ 0.  5.  0.  0.]
 [ 0.  0.  8.  0.]
 [ 0.  0.  0.  9.]]
Percentage Correct:  100.0
```

Fig. 13  Graphical display of the error versus the number of epochs in the Trafic Signals case and the confusion matrix III





```
Total error: 0.00238546299421
epoch:  200   train error:  0.00%   test error:  0.00%
Confusion matrix is:
[[ 6.  0.  0.  0.]
 [ 0.  8.  0.  0.]
 [ 0.  0.  7.  0.]
 [ 0.  0.  0.  8.]]
Percentage Correct:  100.0
```

Fig. 14  Graphical display of the error versus the number of epochs in the Trafic Signals case and the confusion matrix IV

## VIII. THE FINAL CONCLUSION

The computing world has a lot to gain from neural networks. Their Ability to learn by example makes them very flexible, and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task, and there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture.

Neural networks Also Contribute to other areas of research: such as neurology and psychology. Regularly they are used to model parts of living organisms and to investigate the internal mechanisms flag of the brain.

## REFERENCES

[1] http://en.wikipedia.org/wiki/Artificial_neural_network

[2] http://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicap a

[3] http://www.aulavirtual.uji.es

[4] http://en.wikipedia.org/wiki/XOR_gate

[5] http://en.wikipedia.org/wiki/Iris_flower_data_set

[6] http://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival