Pattern neural networks: A case study

Călin-Adrian Popa, Cosmin Cernăzanu-Glăvan

"Politehnica" University of Timişoara, Romania
popacalin2002@yahoo.com, cosmin.cernazanu@ac.upt.ro

Abstract. The use of Matlab in the field of neural networks is common for a long time. Due to its performances, many studies have successfully used it for solving different types of problems. In this paper we will present a case study highlighting the best five pattern recognition neural networks that have been implemented in Matlab R2010b. To illustrate our classification we used three datasets: one containing data from a coastal observatory, one used for digit recognition and the last one from medical tests used to determine diabetes. For each type of neural network, we present its characteristics, its performance on the datasets and the advantages and disadvantages of its use. We conclude that the function fitting neural networks are the best for pattern recognition on the datasets we used, due to their high percentage of recognition and low resource consumption.

Keywords: pattern recognition, neural networks, Matlab

1 Introduction

The domain of artificial neural networks is of big interest in the last years, with several applications in a variety of domains, solving difficult problems such as estimation, identification, prediction or optimization. [1] They are inbuilt in everyday appliances (cell phones, TV sets, microwave ovens, etc.) or we interact with them in our daily life (form recognition, speech recognition, automatic diagnosis, etc.). [2]

For this reason, much research is made in finding the best types of neural networks that must be used in each type application, and what can be done to improve their performance. Matlab came in help of this type of research by providing a Neural Network Toolbox that offers many neural network implementations that can be easily and flexibly used in applications. [3]

Matlab® is a high-level language and interactive environment that enables the performing of computationally intensive tasks faster than with traditional programming languages such as C, C++, and FORTRAN. [4] This description proves that neural networks implemented in Matlab are faster than the ones implemented using the other programming languages. Studying their performance will help researchers use the best neural networks that fit their specific type of problem.

Our article is structured in the following way: after this Introduction, we will present the characteristics of the training of neural networks in Matlab. First, we will discuss the general characteristics, which apply for all types of networks, then a description of the datasets will be made, and then we will present each type of neural network we used, underlying their characteristics and their advantages and disadvantages. The third chapter will present the Experimental Results we obtained, specifying the recognition percentage, the time, memory and the performance of each neural network. The last chapter is dedicated to the Conclusions of our research and the classification we made between the networks, underlying our recommendations.

2 Training pattern neural networks in Matlab

2.1 General characteristics

There are some common characteristics of all the types of neural networks in Matlab. First of all, the dataset should be prepared before it is given to a certain type of neural network. The data should consist of two parts: one input part that contains the input data and one target part that contains the classes in which the data should be organized by the networks, corresponding to the patterns in which the input data is classified.

The target data should be represented as matrices, in which each row has a 1 in some position `i`, where `i` is the class they are to represent, and 0's in the rest. Our classification contains only supervised neural networks, in which the learning is done knowing in which class an input should be placed, but we also experimented with unsupervised neural networks, which try to classify the input data based only on its characteristics, not using target data. [5]

After preparing the data for use, the creation, training and testing of the neural network must be done. The creation is rather easy, having the form:

```
net = network(params);
```

Where `network` is the name of the network and `params` are the necessary parameters. Then the actual training takes place, usually in the following way:

```
net = train(net,x,t);
```

Where `x` are the inputs and `t` are the targets that have been prepared earlier. The network takes a certain percentage (usually 70%) of the input data, and uses it as the training set, another percentage (usually 15%) and uses it as the validation set, and the remaining percentage (usually 15%) as the testing set. After this operation, the neural network is trained.

Now, the testing of the trained network takes place. First, it is tested with the same dataset it was trained for, as shown below:

```
y = net(x);
```

Where `y` denots the outputs of the network, given the inputs `x`. After this, and in the same way, a new dataset is given as inputs for the network, which contains test input data, which hasn't been used to train the network.

To analyse the results given by the network, two methods can be used: one consisting of the predefined function vec2ind which determines the maximum value for each output, and classifies the output considering only that maximum, and the other, in which certain threshold constraints are imposed to the maximum in order to be able to classify the output; if the maximum doesn't meet the threshold, it is considered that the network hasn't recognised the input data corresponding to that output.

As performance measures, the first one is the time needed for the training of the network, and the second one is the performance, given in the following way:

```
perf = perform(net,t,y);
```

The notations being the ones used above. This function calculates the performance usually based on the Mean Squared Error performance function.

2.2 Datasets description

In order to classify the different types of neural networks implemented in Matlab according to their performance in pattern recognition problems, we used three datasets, which have very different characteristics.

The first dataset is taken from Martha's Vineyard Costal Observatory and contains data concerning oceanographic parameters [6]. The data was used to determine the risk of the occurrence of big waves in a certain period of the year. The input data contains 12,000 instances and the test data contains 1,000 instances. It contains 17 attributes and 3 classes.

The next two datasets are taken from the Neural Networks Benchmarks collection, which contains datasets used to evaluate the performances of different types of neural networks [7]. The first dataset that we used from this collection is the Pen-Based Recognition of Handwritten Digits, which contains 10,000 input instances and 1,000 test instances. It has 16 attributes and 10 output classes. The last dataset is the Diabetes, which contains 200 input instances and 67 test instances, 8 attributes and 3 classes.

As it can be easily seen, the datasets are very different from one another in almost every aspect, so we can expect that the results obtained are significant enough.

2.3 Types of neural networks in Matlab

We used seven types of neural networks from Matlab, five of which are supervised neural networks and two are unsupervised. Because the results given by the unsupervised networks were very poor for all of the three datasets, proving that they are not suitable for pattern recognition, in the following we will discuss only the five supervised networks which gave satisfactory results. These networks are:

Pattern recognition neural network. This type of network is created especially for the pattern recognition problems. It is a feed forward network that can be trained to classify inputs according to target classes. Its syntax in Matlab is:

```
patternnet(hiddenSizes,trainFcn),
```

where hiddenSizes is a row vector of one or more hidden layer sizes, and trainFcn is the training function, the default being the Levenberg-Marquardt backpropagation algorithm.

Feed forward neural network. These networks consist of a series of layers. The first layer is connected to the inputs. Each subsequent layer has a connection to the previous layer, and the last layer gives the network output. Their syntax is:

```
feedforwardnet(hiddenSizes,trainFcn),
```

where the parameters are the same as above.

Cascade forward neural network. Cascade forward networks are similar to feed forward networks, with the exception that they have a connection from the input and every previous layer to the following layers, and thus their name. Their syntax is:

```
cascadeforwardnet(hiddenSizes,trainFcn),
```

where the parameters are the same as above.

Function fitting neural network. The function fitting neural networks are also a type of feed forward networks, which are used to fit an input output relationship. Their syntax in Matlab is:

```
fitnet(hiddenSizes,trainFcn),
```

where the parameters are the same as above.

Learning vector quantization neural network. LVQ networks consist of two layers. The first one maps input vectors into clusters that are found by the network during training. The second layer merges groups found by the first layer into the classes defined by the target data. Their syntax in Matlab is:

```
lvqnet(hiddenSize,lvqLR,lvqLF),
```

where hiddenSize is the size of the hidden layer, lvqLR is the LVQ learning rate and lvqLF is the LVQ learning function.

3 Experimental results

Using the networks specified above and the presented datasets, we obtained the graphs listed in Fig.1, Fig.2 and Fig.3. Fig.1 lists the percentage of recognition of the trained network applied to the test set. Fig.2 lists the performance of each network, calculated using the Mean Squared Error performance function, when the network is applied to the test set. The last figure, Fig.3 lists the training time and memory consumption for each network and each dataset.

This last figure, as well as other resource measures done during training, showed that pattern network is the most effective, feed forward and fitting networks follow after it, and the most time and resource consuming is the cascade forward network and the LVQ network.

In terms of performance, experiments showed that the difference between pattern, cascade forward, feed forward and fitting networks is rather small, but the LVQ network has a very poor performance on this type of problem.

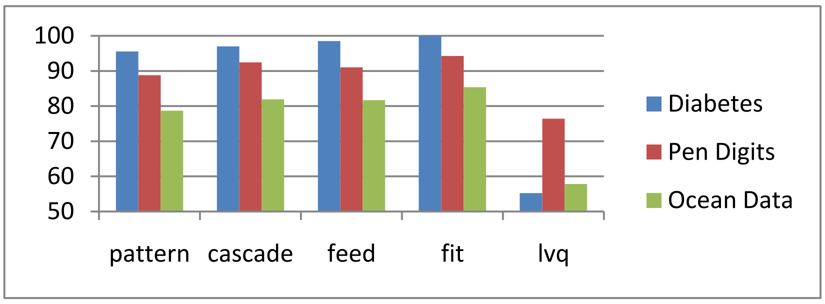

Fig. 1. Recognition percentage

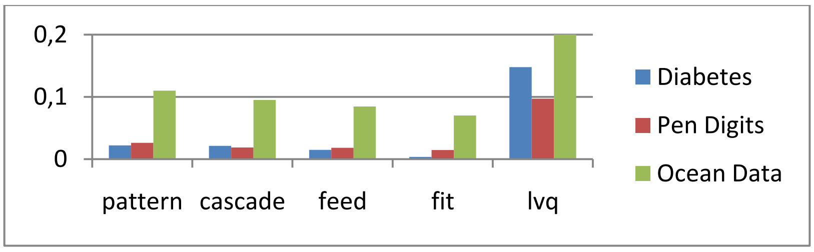

Fig. 2. Network performance (low is better)

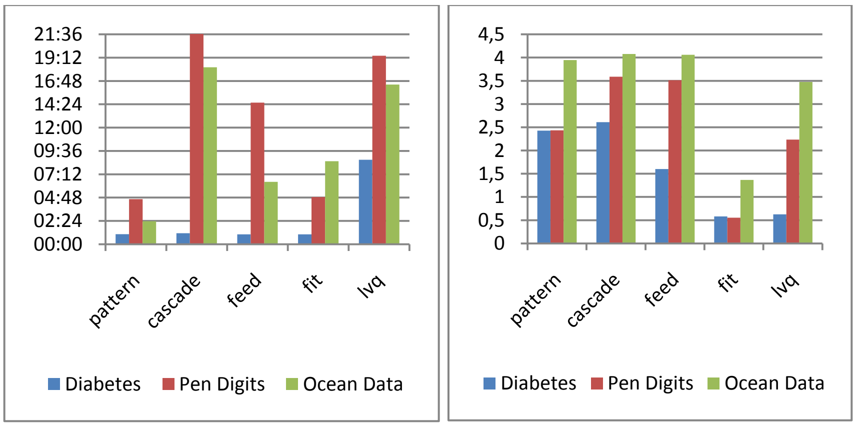

Fig. 3. Training time (low is better) and memory consumption (low is better)

The most important aspect is the percentage of recognition of the networks applied to the test sets. Fig.1 suggests the following classification: first, the fitting network, followed by the cascade forward and feed forward networks with almost the same percentage of recognition, then the pattern network, created especially for this type of problem, and last the LVQ network with such a poor recognition percentage that it is not recommended for such types of problems.

We also did experiments with competitive layer and self organizing map neural networks, which are unsupervised networks, but the results were also too poor to be considered in our classification.

4 Conclusion

This case study can be a starting point for whoever wants to use Matlab for pattern recognition problems, by giving a classification of the best five neural networks that can be used for these types of problems.

The experiments showed that the best choice that can be made using Matlab for this type of problem is the function fitting neural network, both in terms of recognition percentage as well as in terms of network performance. The runner-ups were the feed forward and the cascade forward neural networks, but the last one has poor time and resource performance. The pattern recognition neural network came only fourth in our classification, although it was especially designed in this scope by Matlab. LVQ networks are not recommended for use in pattern recognition problems.

The recommendation is thus to use function fitting networks for pattern recognition problems, but also not to neglect the feed forward, cascade forward and of course the pattern recognition neural networks, that are implemented in Matlab.

Acknowledgment. This work was partially supported by the strategic grant POSDRU/89/1.5/S/57649, Project ID 57649 (PERFORM-ERA), co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development 2007-2013.

References

1. Luger, G. F., "Artificial Intelligence, Structures and Strategies for Complex Problem Solving", Addison-Wesley, 2005
2. Rabunal, J. R., "Artificial Neural Networks in Real-life Applications", *Idea Group Pub*, 2005
3. Demuth H., Beale M. and Hagan M. (2006). Neural network toolbox for use with MATLAB. *Neural Network Toolbox*, IEE Savoy Place, London.
4. Matlab website, http://www.mathworks.com/products/matlab/
5. Haykin, S., "Neural Networks. A comprehensive Foundation", *Macmillan College Publishing*, New York, 1994
6. Martha's Vineyard Coastal Observatory: "Ocean Data", http://www.whoi.edu/mvco/data/oceandata.html
7. Carnegie Mellon University: "Neural Networks Benchmark Collection", http://archive.ics.uci.edu/ml/datasets