Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

Computer Science Engineering Degree
Mention: Computation
Fourth year. First quadrimester

Introduction to Computational Models

Assignment 1: Multilayer Perceptron

Ventura Lucena Martínez
31008689C
i72lumav@uco.es

Academic Year 2021-2022
Cordoba, October 27, 2021

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

This lab assignment serves as familiarisation with neural network computational models, in particular, with the multilayer perceptron. To do this, an implementation of the basic back-propagation algorithm for the multilayer perceptron has been carried out with an analysis checking the effect of the different parameters: network architecture or topology, moment factor (μ), use of validation set, decrease of the learning rate (η) for each layer and so on).

1 Architecture

The concept of **architecture** referred to neural networks makes mention not only of the number of neuronal layers or the number of neurons in each of them, but also the connection between neurons or layers, the type of neurons present and even the way in which they are trained.

1.1 Layers and Neurons

On the first hand, a **layer** is a set of neurons whose inputs come from a previous layer (or from the input data in the case of the first layer) and whose outputs are the input from a later layer. We will denote each layer as l_i.

On the second hand, the basic unit of the neural network is the **perceptron** or **neuron**. Each neuron has **inputs** with x_i values, each one weighted with its corresponding **weight** w_i to be optimized.

1.2 Connetions

The neural network has its **neurons fully connected**; this is, each neuron of the layer i is fully connected with each neuron of the layer $i + 1$, for any neuron from $i = 0$ to $i = n\_neurons$ - 1. In the experiments we will see the following cases:

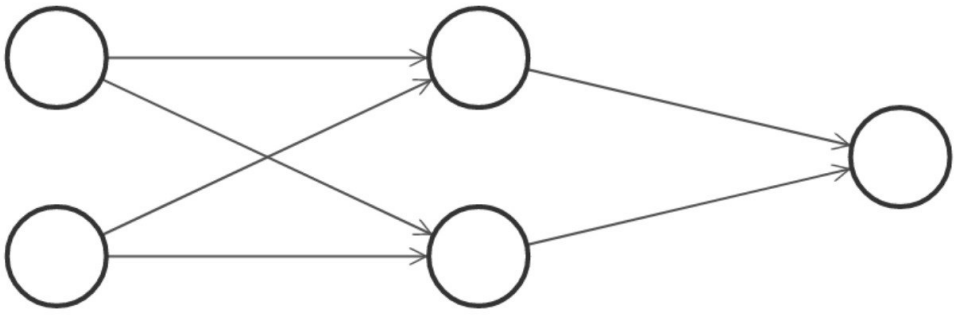

Figure 1: Topology based in 1 hidden layer and 2 neurons/each.

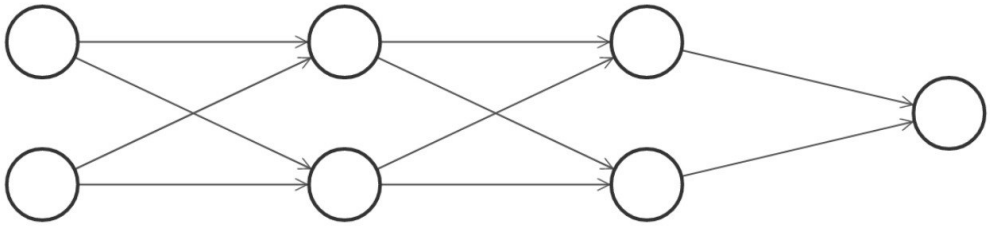

Figure 2: Topology based in 2 hidden layer and 2 neurons/each.

1.3 Type of neurons

The neurons used in the analysis use a **sigmoid activation function**:

$$\sigma(x) = \frac{1}{1 + e^{-(x-t)}} \tag{1}$$

1.4 Training

The neural network training is based on the **Minimum Square Error (MSE)** cost function. We recall that the MSE is defined as follows:

$$MSE = \frac{1}{N} \sum_{p=1}^{N} \left(\frac{1}{k} \sum_{o=1}^{k} (d_{po} - o_{po})^2 \right) \tag{2}$$

- N: number of pattern in the considered dataset.
- k: number of outputs.
- d_{po}: target value for pattern p and the output variable o.
- o_{po}: predicted value.

2 Back-propagation algorithm

The pseudo code that adheres to the backpropagation algorithm is the following:

Algorithm 1 Back-propagation

```
i ← 1
while StopCondition == false do
    while i < nPatterns − 1 do
        feedInputs();
        forwardPropagate();
        backpropagateError();
        accumulateChange();
        weightAdjustment();
    end while
end while
```

- *feedInput()*: feed the input neurons of the network with a vector passed as an argument.
- *forwardPropagate()*: calculate and propagate the outputs of the neurons, from the first layer until the last one.
- *backpropagateError()*: backpropagate the output error wrt a vector passed as an argument, from the last layer to the first one.
- *accumulateChange()*: accumulate the changes produced by one pattern and save them in Δ_w.
- *weightAdjustment()*: update the network weights, from the first layer to the last one.

3 Experiments

We will test different neural network configurations and run each configuration with five seeds (1, 2, 3, 4 and 5). Based on the results obtained, the standard and mean deviation of the error will be obtained. The MSE error has been calculated for both the training set and the test set.

To assess how the implemented algorithm works, we will use four datasets, in which we will use the following nomenclature:

- n: number of inputs.
- h: number of neurons in each hidden layer.

- k: number of outputs.
- l: number of hidden layers.

3.1 XOR problem

This dataset represents the problem of non-linear classification of the XOR. The same file will be used for train and test:

	Train	**Test**
Network Architecture {n:h:k}	**Mean +- Std**	**Mean +- Std**
{2 : 2 : 1}	0.169924 +- 0.0535004	0.169924 +- 0.0535004
{2 : 4 : 1}	0.126922 +- 0.0862173	0.126922 +- 0.0862173
{2 : 8 : 1}	0.0504197 +- 0.0276255	0.0504197 +- 0.0276255
{2 : 32 : 1}	0.0149084 +- 0.00494185	0.0149084 +- 0.00494185
{2 : 64 : 1}	0.0581697 +- 0.0967619	0.0581697 +- 0.0967619
{2 : 100 : 1}	0.105724 +- 0.118576	0.105724 +- 0.118576

Table 1: XOR test with l=1.

	Train	**Test**
Network Architecture {n::h::k}	**Mean +- Std**	**Mean +- Std**
{2 : 2 : 2 : 1}	0.249957 +- 7.72707e-05	0.249957 +- 7.72707e-05
{2 : 4 : 4 : 1}	0.248135 +- 0.00312095	0.248135 +- 0.00312095
{2 : 8 : 8 : 1}	0.229608 +- 0.0062313	0.229608 +- 0.0062313
{2 : 32 : 32 : 1}	0.00736378 +- 0.00039509	0.00736378 +- 0.00039509
{2 : 64 : 64 : 1}	0.00284092 +- 0.000271725	0.00284092 +- 0.000271725
{2 : 100 : 100 : 1}	0.0511583 +- 0.0992609	0.0511583 +- 0.0992609

Table 2: XOR test with l=2.

3.2 Sine function

This dataset is composed by 120 training patterns and 41 testing patterns. It has been obtained adding some random noise to the sine function[1].

[1]See the figure in the assignment statement, page 4 [1].

	Train	Test
Network Architecture {n:h:k}	**Mean +- Std**	**Mean +- Std**
{2 : 2 : 1}	0.0296493 +- 0.00013139	0.036267 +- 0.000237388
{2 : 4 : 1}	0.0294346 +- 0.000405016	0.0364018 +- 0.000186411
{2 : 8 : 1}	0.0293432 +- 0.000488771	0.0363062 +- 0.000706818
{2 : 32 : 1}	0.029185 +- 0.000484976	0.0357259 +- 0.00066081
{2 : 64 : 1}	0.0282931 +- 0.000266491	0.0353251 +- 0.000436835
{2 : 100 : 1}	0.0279422 +- 0.000854061	0.035787 +- 0.00170421

Table 3: Sine test with l=1.

	Train	Test
Network Architecture {n::h::k}	**Mean +- Std**	**Mean +- Std**
{2 : 2 : 2 : 1}	0.0297324 +- 1.32882e-05	0.0361349 +- 7.98215e-05
{2 : 4 : 4 : 1}	0.0297757 +- 3.71914e-05	0.0361989 +- 0.00017494
{2 : 8 : 8 : 1}	0.0298237 +- 0.000105279	0.0361776 +- 0.000221546
{2 : 32 : 32 : 1}	0.0296029 +- 0.000660249	0.0362081 +- 0.000547148
{2 : 64 : 64 : 1}	0.0277694 +- 0.000288064	0.0349257 +- 0.000408011
{2 : 100 : 100 : 1}	0.0236055 +- 0.00361911	0.031486 +- 0.00461592

Table 4: Sine test with l=2.

3.3 Quake dataset [2]

This dataset is composed by 1633 training patterns and 546 testing patterns. It corresponds to a database in which the objective is to find out the strength of an earthquake (measured on the Richter scale). As input variables, we use the depth of focus, the latitude and longitude at which it occurs.

	Train	Test
Network Architecture {n:h:k}	**Mean +- Std**	**Mean +- Std**
{2 : 2 : 1}	0.0301504 +- 5.9553e-05	0.0272633 +- 7.75238e-05
{2 : 4 : 1}	0.0300978 +- 7.31425e-05	0.0271794 +- 8.5991e-05
{2 : 8 : 1}	0.0299311 +- 4.25856e-05	0.027022 +- 6.82137e-05
{2 : 32 : 1}	0.0298313 +- 6.41134e-05	0.0269965 +- 3.89243e-05
{2 : 64 : 1}	0.0297725 +- 4.39388e-05	0.026944 +- 5.79112e-05
{2 : 100 : 1}	0.0297574 +- 1.8829e-05	0.0269815 +- 5.3433e-05

Table 5: Quake test with l=1.

	Train	Test
Network Architecture {n::h::k}	**Mean +- Std**	**Mean +- Std**
{2 : 2 : 2 : 1}	0.0301905 +- 9.57208e-05	0.0272986 +- 0.00010887
{2 : 4 : 4 : 1}	0.0301433 +- 5.26797e-05	0.0272438 +- 6.21493e-05
{2 : 8 : 8 : 1}	0.0301133 +- 4.10226e-05	0.0272139 +- 3.93082e-05
{2 : 32 : 32 : 1}	0.0298069 +- 5.01197e-05	0.0269754 +- 3.87634e-05
{2 : 64 : 64 : 1}	0.0295227 +- 6.07135e-05	0.0270543 +- 6.38149e-05
{2 : 100 : 100 : 1}	0.0294049 +- 4.31967e-05	0.0271286 +- 4.12738e-05

Table 6: Quake test with l=2.

3.4 Parkinsons dataset [3]

This dataset is composed by 4406 training patterns and 1469 testing patterns. It contains, as inputs or independent variables, a series of clinical data from patients with Parkinson's disease, including biometric measurement data from their voice. Furthermore, as output or dependent variables, it includes the motor value and the UPDRS (Unified Parkinson's Disease Rating Scale).

	Train	Test
Network Architecture {n:h:k}	**Mean +- Std**	**Mean +- Std**
{2 : 2 : 1}	0.0348651 +- 0.000815127	0.0369673 +- 0.000444937
{2 : 4 : 1}	0.0318177 +- 0.00532028	0.0324739 +- 0.00558102
{2 : 8 : 1}	0.0240912 +- 0.00343674	0.0243361 +- 0.00331208
{2 : 32 : 1}	0.0149779 +- 0.0009777	0.0164731 +- 0.000969022
{2 : 64 : 1}	0.0143679 +- 0.000383266	0.0164768 +- 0.000546589
{2 : 100 : 1}	0.0142593 +- 0.000346112	0.015767 +- 0.000852286

Table 7: Parkinsons test with l=1.

	Train	Test
Network Architecture {n:h:k}	**Mean +- Std**	**Mean +- Std**
{2 : 2 : 2 : 1}	0.0415226 +- 0.00941726	0.0420234 +- 0.00880434
{2 : 4 : 4 : 1}	0.026611 +- 0.00326153	0.0279339 +- 0.00428357
{2 : 8 : 8 : 1}	0.0208918 +- 0.00406526	0.0210816 +- 0.00405457
{2 : 32 : 32 : 1}	0.00830808 +- 0.00124543	0.00989245 +- 0.00160873
{2 : 64 : 64 : 1}	0.00613586 +- 0.00107343	0.00877766 +- 0.00166647
{2 : 100 : 100 : 1}	0.00593478 +- 0.00105771	0.00861298 +- 0.00159673

Table 8: Parkinsons test with l=2.

4 Extra experiments

Known the best network architecture for each problem, we will try some combinations for parameters v and F, that help our neural network to not over-learn. Although we could get worse results, over-learning would be a problem if we want a general approach (instead of a concrete one):

4.1 Sine function

	Train	**Test**
Network Architecture {n:h:k}, v, F	**Mean +- Std**	**Mean +- Std**
{2 : 100 : 100 : 1}, v=0, F=1	0.0236055 +- 0.00361911	0.031486 +- 0.00461592
{2 : 100 : 100 : 1}, v=0.15, F=1	0.0294976 +- 0.000804972	0.0391876 +- 0.00142406
{2 : 100 : 100 : 1}, v=0.25, F=1	0.0292843 +- 0.000643391	0.0388381 +- 0.00118085
{2 : 100 : 100 : 1}, v=0.15, F=2	0.0294976 +- 0.000804972	0.0391876 +- 0.00142406
{2 : 100 : 100 : 1}, v=0.25, F=2	0.0292843 +- 0.000643391	0.0388381 +- 0.00118085

Table 9: Sine test with l=2, v and F.

4.2 Quake dataset

	Train	**Test**
Network Architecture {n:h:k}, v, F	**Mean +- Std**	**Mean +- Std**
{2 : 100 : 100 : 1}, v=0, F=1	0.0294049 +- 4.31967e-05	0.0271286 +- 4.12738e-05
{2 : 100 : 100 : 1}, v=0.15, F=1	0.0300118 +- 1.75272e-05	0.0271158 +- 1.89527e-05
{2 : 100 : 100 : 1}, v=0.25, F=1	0.0300123 +- 1.69179e-05	0.0271157 +- 2.06249e-05
{2 : 100 : 100 : 1}, v=0.15, F=2	0.0300118 +- 1.75272e-05	0.0271158 +- 1.89527e-05
{2 : 100 : 100 : 1}, v=0.25, F=2	0.0300123 +- 1.69179e-05	0.0271157 +- 2.06249e-05

Table 10: Quake test with l=2, v and F.

4.3 Parkinsons dataset

	Train	**Test**
Network Architecture {n:h:k}, v, F	**Mean +- Std**	**Mean +- Std**
{2 : 100 : 100 : 1}, v=0, F=1	0.00593478 +- 0.00105771	0.00861298 +- 0.00159673
{2 : 100 : 100 : 1}, v=0.15, F=1	0.00956598 +- 0.00126631	0.0106193 +- 0.00121578
{2 : 100 : 100 : 1}, v=0.25, F=1	0.00951922 +- 0.00104379	0.0104841 +- 0.00113511
{2 : 100 : 100 : 1}, v=0.15, F=2	0.00717236 +- 0.00144832	0.0085327 +- 0.00137681
{2 : 100 : 100 : 1}, v=0.25, F=2	0.00615798 +- 0.00167699	0.0080563 +- 0.00169684

Table 11: Parkinsons test with l=2, v and F.

5 Conclusion

After all the experiments carried out, we can see how the neural networks with 2 layers and 100 neurons in each layer perform better results, independently the used parameters.

However, although this results are the best, we have to consider the amount of time that each configuration spend executing the algorithm. In this sense, due to the fact that most of the configurations perform good results, we could consider that the best ones are not that ones with the minimum errors. For instance, if we calculate the relation between the error obtained and the time spent, we can obtain a good approach to a phenomenal neural network:

$$Best_i = \frac{Error_i}{Time_i} \tag{3}$$

Finally. it would have been interesting to expose some convergence charts and the performance of the metric commented previously based on the error and time, but due of to lack of time, it has not been possible.

References

[1] Pedro Antonio Gutiérrez Peña. *Lab assignment 1: Multilayer perceptron implementation*. URL: https://moodle.uco.es/m2122/mod/resource/view.php?id=49010.

[2] *Quake dataset*. Knowledge Extraction based on Evolutionary Learning. URL: https://sci2s.ugr.es/keel/dataset.php?cod=75.

[3] Athanasios. Tsanas **and** Max. Little. *Parkinsons Telemonitoring Data Set*. URL: http://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring.