Performance comparison of ANN training algorithms for classification

F. Darío Baptista[1], Sandy Rodrigues[1]
[1]Madeira Interactive Technologies Institute
University of Madeira
Funchal, Portugal

Fernando Morgado-Dias[1,2]
[2]Competence Center of Exact Sciences and Engineering
University of Madeira
Funchal, Portugal
morgado@uma.pt

***Abstract*—The Artificial Neural Network research community has been actively working since the beginning of the 80s. Since then many existing algorithm were adapted, many new algorithms were created and many times the set of algorithms was revisited and reinvented. As a result an enormous set of algorithms exists and, even for the experienced user it is not easy to choose the best algorithm for a given task or dataset, even though many of the algorithms are available in implementations of existing tools. In this work we have chosen a set of algorithms which are tested with a few datasets and tested several times for different initial sets of weights and different numbers of hidden neurons while keeping one hidden layer for all the Feedforward Artificial Neural Networks.**

Keywords*—*classification, artificial neural networks, algorithm, performance

I. INTRODUCTION

The Artificial Neural Network (ANN) research community has been actively working since the beginning of the 80s. A lot of the work developed concerned different ANN architectures, hardware implementations but training algorithms always received a large share of attention. Since the beginning many existing algorithm were adapted, many new algorithms were created and many times the set of algorithms was revisited and reinvented. As a result an enormous set of algorithms exists and, even for the experienced user it is not easy to choose the best algorithm for a given task or dataset, even though many of the algorithms are available in implementations of existing tools.

For this work we decided to test some of the existing algorithms and test them with benchmark datasets in order to compare them and help the user to choose the best algorithms for their applications.

One of these tools is Neural Network Toolbox of Matlab. Matlab is today the reference tool in many engineering fields and its new version of the ANN toolbox [1] will undoubtedly become a reference for the ANN community for its ease of use.

Another tool used in this test is the NNSYSID [2-4]. This toolbox, which is also used with Matlab, was the first tool to have a Levenberg-Marquart [5-6] algorithm implementation that exhibited a good performance and is still a reference in the ANN community. This tool also contains two other training algorithm, namely a Batch version of Backpropagation and Recursive (incremental) version of backpropagation

Another tool tested is the Neurosolutions for Matlab [7]. This software is associated with a very pedagogical book about ANN and exhibits some relevant characteristics, such as a graphical tool, fast algorithms and good theoretical background that tempted us to test it also.

II. TRAINING SET SIZE

The use of the ANN considers three aspects: train the network, validate and finally test it. During training and

validating, the ANN acquires knowledge and later uses this knowledge to test on the data.

A feed-forward network is a non-parametrical classifier so, it requires a lot of data for appropriate training due to not having a priori assumptions about the data. The number of training patterns, N, required for classifying test samples with an error δ is approximately given by [7]:

$$N > \frac{W}{\delta} \qquad (1)$$

Where W is the number of weights in the ANN. A rule of thumb based on this states that N should be approximately 10W. Where δ is an accuracy parameter and a good value for this parameter is 0.1, which corresponds to an accuracy level of 90%. For this rule of thumb to be true, we should always choose samples for the train dataset that cover all possible conditions. If the train dataset does not contain data from some areas of pattern space, the machine classification in those areas will be based on extrapolation [4].

III. Test Description

In this work four different datasets were used, namely *Parkinsons Data Set, Car Evaluation Data Set, Cardiotocography Data Set* I and II. Table 1 shows the features of each one of the datasets [9].

Table 1. Features of each dataset used in this work.

Data Set	Number of inputs	Number of classes	Number of samples			
			Train N_{train}	Validation N_{val}	Test N_{test}	Total
Parkinsons	22	1	137	30	30	197
Car Evaluation	6	4	1208	261	261	1730
Cardiotocography I	21	3	1488	319	319	2126
Cardiotocography II	21	10	1488	319	319	2126

These datasets were chosen because they are of upper average or high complexity for classification and that will allow us to take conclusions about which algorithm performs better. Table 2 shows the information that leads us to conclude the previous statement. It is possible to see the number of weights of each ANN(W) and the number of training patterns required classifying test samples an accuracy level of 90% (N~$10W$).

In the same table it is possible to analyze the difference between the number of sample available for each dataset and the number of sample of thumb states' rule, ΔN, in other words:

$$\Delta N = N_{train} - N \qquad (2)$$

If $\Delta N>0$ it means that there are enough sample to obtain 90% accuracy in the sample test otherwise there are not enough samples. The networks where $\Delta N<0$ are the ones where the training algorithm will find difficulties and it will be possible to verify the efficiency of each of these algorithms.

In this work 20 algorithms were tested and for each algorithm 20 ANNs of one hidden layer with 4, 7, 10, 12, 15 and 20 neurons in that layer were tested in a total of 140 ANN per algorithm. In the total, 1820 ANN were trained and tested for the Neural Network Toolbox and 560 ANN for the NNSYSID.

Table 2. Features of each dataset used in this work.

Data Set	Number of neurons	W	N	ΔN	Number of classes	Dataset Level of complexity according to	
						thumb states' rule	number of classes
Car Evaluation	4	48	480	729	4	Average	High
	7	81	810	399			
	10	114	1140	69			
	12	136	1360	-151			
	15	169	1690	-481			
	20	224	2240	-1031			
Cardiotocography I	4	103	1030	458	3	High	High
	7	178	1780	-292			
	10	253	2530	-1042			
	12	303	3030	-1542			
	15	378	3780	-2292			
	20	503	5030	-3542			
Cardiotocography II	4	138	1380	108	10	High	Very High
	7	234	2340	-852			
	10	330	3300	-1812			
	12	394	3940	-2452			
	15	490	4900	-3412			
	20	650	6500	-5012			
Parkinsons	4	97	970	-833	2	Very High	Easy
	7	169	1690	-1553			
	10	241	2410	-2273			
	12	289	2890	-2753			
	15	361	3610	-3473			
	20	481	4810	-4673			

The algorithms used for training are summarized in table 3.

Table 3. Algorithms tested. (Name of the algorithm on the tool)

Tool	*Training algorithm*	Description
Neural Network Toolbox of Matlab	*trainbfg*	*BFGS quasi-Newton backpropagation*
	trainbr	*Bayesian regulation backpropagation*
	traincgb	*Conjugate gradient backpropagation with Powell-Beale restarts*
	traincgf	*Conjugate gradient backpropagation with Fletcher-Reeves updates*
	traincgp	*Conjugate gradient backpropagation with Polak-Ribiére updates*
	traingd	*Gradient descent backpropagation*
	traingda	*Gradient descent with adaptive learning rate backpropagation*
	traingdm	*Gradient descent with momentum backpropagation*
	traingdx	*Gradient descent with momentum and adaptive learning rate backpropagation*
	trainlm	*Levenberg-Marquardt backpropagation*
	trainoss	*One-step secant backpropagation*
	trainrp	*Resilient backpropagation*
	trainscg	*Scaled conjugate gradient backpropagation*
NNSYSID	*batbp*	*Batch version of backpropagation*
	incbp	*Recursive (incremental) version of backpropagation*
	marq	*Basic Levenberg-Marquardt method*
eurosolutions for Matla	*Step*	*Step*
	Momentum	*Momentum*
	DeltaBarDelta	*Delta-Bar-Delta*
	Quickprop	*Quickprop*

The first algorithms belong to the ANN Matlab toolbox while the following 3 are from the NNSYSID toolbox and the last 4 are from Neurosolutions for Matlab.

IV. RESULTS

The ANNs were trained using nonparametric training because the weights of the classifier are directly estimated from the input data. For that reason the measures for nonparametric data (median and the percentiles) are used. If a parametric measure is used, mean and standard deviation, the results are likely to be inaccurate because there are some cases with extreme values (outliers and far-outliers) that would elevate the mean.

A way to illustrate the median, percentiles and the dispersion of data is using boxplot. Figures 1, 2, 3 and 4 present the boxplot of correct classification of all training algorithms for each dataset. The line in the middle of the boxes is the median of correct classification. The bottom of the box indicates the 25% of cases that have values between the 50th and the 25th percentile and the top of the box represents the 25% of the cases that have values between the 50th and the 75th percentile. This means that 50% of the cases lie within the box. The T-bars that extend from the boxes are called whiskers. The whisker below of 25th percentile represents 25% of the cases and the other whisker above of 75th percentile represents the remaining 25% of the cases. It is possible to find also outliers and far-outliers. The outliers are defined as values that do not fall in the inner fences and the far-outliers represent cases that have values more than three times the height of the boxes. It is possible to verify that there are some outliers and far-outliers that allowed the best classification and others that allowed the worst classification. This variance is caused by the random initial weights that can benefit or not the training algorithm.

Figure 1 present the performance of training algorithms of the car evaluation dataset which was classified as a dataset of average complexity according to thumb states' rule and high complexity according to the number of classes.

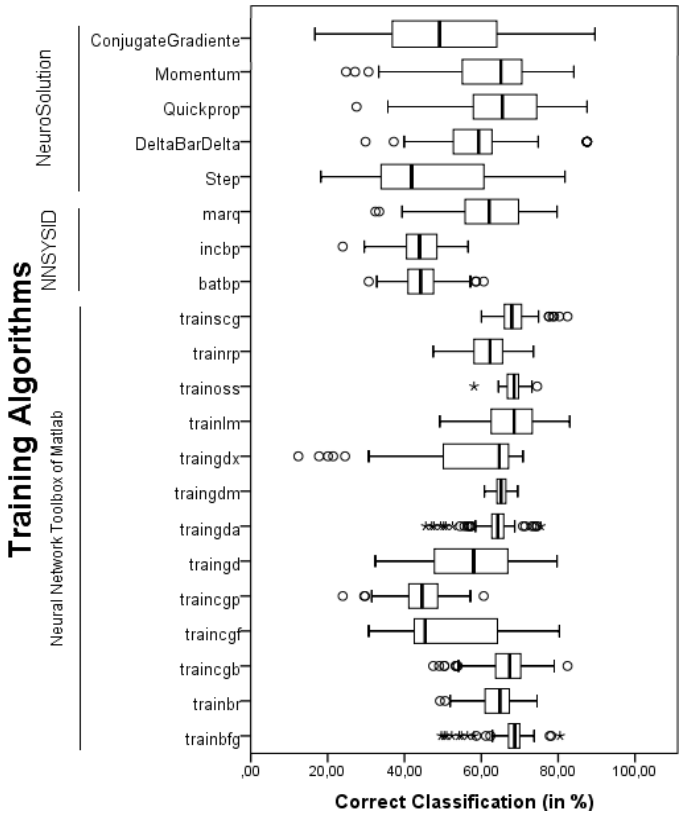

Figure 1. Results of the performance of the different training algorithms using the Car evaluation dataset.

When analyzing figure 1 it can be verified that the training algorithms trainbfg, trainscg, traingda, traingdm and trainoss have little variation in the percentage of correct classification. This means that these algorithms have little efficiency for this dataset whose complexity is average. Comparing the algorithms between the two different tools it can be observed that the NNSYSID presents lower efficiency except on the marq algorithm. Analyzing the Neurosolution algorithms it can be verified that they present a good performance, especially QuickProp and Momentum.

Figure 2 presents the performance of the training algorithms for the Cardiotocography I dataset which was classified as a dataset of high complexity according to thumb states' rule and high complexity according to the number of classes.

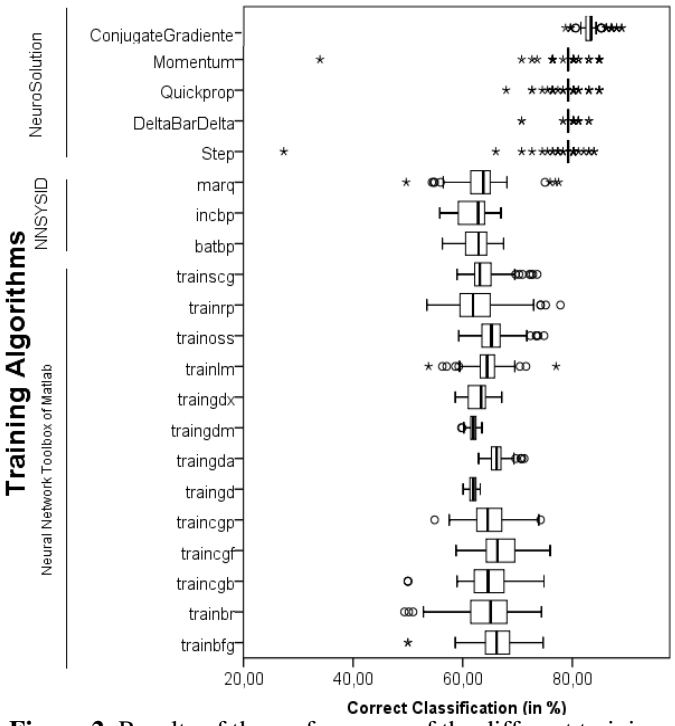

Figure 2. Results of the performance of the different training algorithms using the Cardiotocography I dataset

Analyzing figure 2, we verify that the training algorithms traingd and traingdm have lower media and little variation in the percentage of correct classification. This was predictable because in the previous results where the dataset had average complexity they already presented some difficult to classify correctly.Also, it is possible to see that the training algorithms of Neurosolutions have a good performance, exhibiting the best results. This fact was verified in the previous dataset too. The NNSYSID algorithms did not performed very good on this dataset obtaining below average results.

Figure 3 presents the performance of the training algorithms of Cardiotocography II dataset which was classified as a dataset of high complexity according to thumb states' rule and very high complexity according to the number of classes.

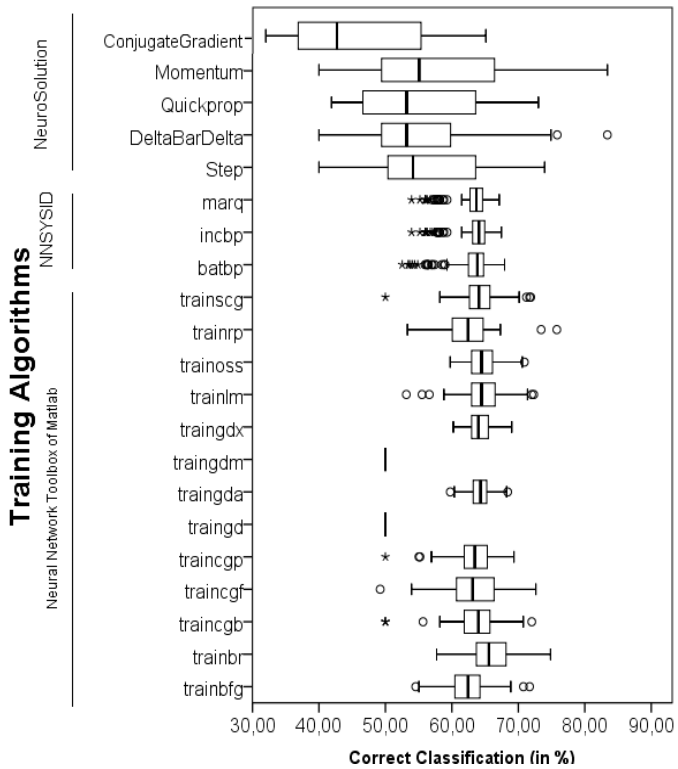

Figure 3. Results of the performance of different training algorithms using the Cardiotocography II dataset

Analyzing figure 3, one verifies that the training algorithms *traingd* and *traingdm* have lower media and no variation in the percentage of correct classification. Again, it was verified that these training algorithms do not have good capacity to train an ANN to find the best performance. It is also possible to verify that trainbr and traincgf can get the best result for the dataset with higher complexity. The NNSYSID algorithms show an average performance for this dataset and the Neurosolutions algorithms, although their average values are low, present either outliers or the top of the whiskers with very good values.

Figure 4 presents the performance of the training algorithms of Parkinson dataset which was classified as a dataset of high complexity according to thumb states' rule and very easy complexity according number of classes.

In figure 4 it is possible to verify that the algorithms traincgb, traincgf, traincgp, trainlm and trainoss present the value of 25th percentile equal to the median, in other words, there are 35 ANN with the same capacity to make classification. A similar situation can be found for the Neurosolutions algorithms. However, in the algorithms of Neurosolutions, the results were worst when compared with the Matlab counterpart.

Analyzing in particular the boxplot of traingdx it is possible to verify that the value of 25th percentile and the minimum extreme of whiskers are equal to the median. So, 50% of ANNs have the same capacity to make classification.

Another detail for the Parkinson dataset shows that the training algorithms traincgb, traincgf, trainscg and marq achieved an accuracy level equal or higher than 90% with a

dataset where the train samples do not respect a rule of thumb which states that N ~ 10W. This means that these algorithms have a good efficiency. However, it is very difficult for traincgb and traincgf to obtain an ANN with this performance. For these algorithms, the probability of getting the ANN with this result is 1 ANN in 140 ANN. Nevertheless, with the algorithms trainscg and marq it is more likely to obtain an ANN with this higher performance.

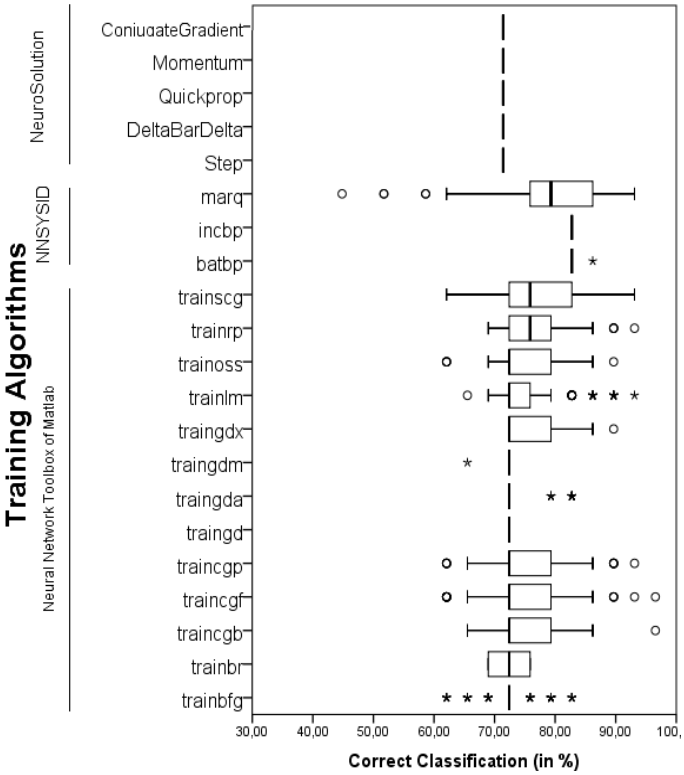

Figure 4. Results of the performance of different training algorithms using the Parkinson dataset

Finally, to find the best result among datasets and have a general perspective we show the algorithm's rank for each dataset. To determinate this rank we used a statistic rule for nonparametric data [10]. In this procedure, the rank for nonparametric data is calculated by ranking all scores as if they belonged to one algorithm, then adding the separate rank totals for each algorithm.

In this rank, the highest value points to the best algorithm. The General column adds the ranks obtained for each dataset to give a general perspective.

Considering individual datasets the best and worst performance are obtained by:

- Best: trainoss / worst: traincgf in car dataset;
- Best: Conjugate Gradient / worst: traingd and traingdx in CTG I dataset;
- Best: trainbr / worst: traingdm in CTG II dataset;
- Best: batbp / worst: Step, DeltaBarDelta, Quickprop, Momentum and Conjugate Gradient in Parkinson dataset;

Table 4. Rank for each algorithm

	car	*CTG I*	*CTG II*	*Parkinson*	*General*
trainbfg	*1784,29*	*1227,73*	*1211,04*	*1178,59*	*5401,65*
trainbr	*1382,25*	*1017,03*	*1776,08*	*922,35*	*5097,71*
traincgb	*1599,26*	*1043,87*	*1475,78*	*1417,99*	*5536,9*
traincgf	*702,58*	*1269,68*	*1389,85*	*1401,01*	*4763,12*
traincgp	*384,55*	*1063,08*	*1382,70*	*1458,63*	*4288,96*
traingd	*995,19*	*461,26*	*282,00*	*1206,00*	*2944,45*
traingda	*1306,98*	*1320,49*	*1542,70*	*1292,37*	*5462,54*
traingdm	*1412,17*	*461,48*	*282,00*	*1196,24*	*3351,89*
traingdx	*1208,18*	*712,54*	*1531,19*	*1495,47*	*4947,38*
trainlm	*1673,10*	*1020,95*	*1594,20*	*1408,57*	*5696,82*
trainoss	*1821,28*	*1153,43*	*1588,99*	*1474,10*	*6037,8*
trainrp	*1204,05*	*670,86*	*1231,86*	*1583,14*	*4689,91*
trainscg	*1759,24*	*857,53*	*1523,30*	*1534,48*	*5674,55*
batbp	*380,20*	*678,21*	*1367,08*	*2160,07*	*4585,56*
incbp	*367,82*	*595,05*	*1410,66*	*2158,50*	*4532,03*
marq	*1446,00*	*2137,19*	*1055,86*	*523,90*	*5063,55*
Step	*622,99*	*2062,96*	*888,18*	*424,50*	*3998,63*
DeltaBarDelta	*1078,19*	*2100,24*	*733,34*	*424,50*	*4336,27*
Quickprop	*1505,57*	*2085,00*	*839,98*	*424,50*	*4855,05*
Momentum	*1369,88*	*2071,37*	*903,08*	*424,50*	*4768,83*
Conjugate Gradient	*652,65*	*2210,22*	*845,58*	*424,50*	*4132,95*

As can you see in table 4, in a general way, the *trainoss* is the best training algorithms. In the second and third position lie *trainlm* and *trainscg*.

Regarding the worst performances we can find traingd algorithm with a very low rank when compared to other algorithms.

V. CONCLUSION

Comparing the performance of different algorithms is a complex task. For this work we have selected a set of datasets with upper average or high complexity and we have tested the algorithms using one hidden layer ANN and freedom to use different numbers of neurons according to a predefined set of values.

Considering the results obtained for the datasets chosen, the best training algorithms are the trainoss, trainlm and trainscg considering the general ranking presented in table 4 and the

worst algorithm is traingd. All these algorithms belong to the Matlab ANN toolbox. It should be noted that traingd corresponds to what is usually known as Backpropagation which is still used by many practitioners in ANN community, but is the simplest first order training algorithm.

When looking at the results obtained by the other tools at the general ranking we can see that the quality of the algorithms is not very different from the Matlab ones. The best result outside of Matlab is obtained by marq from NNSYSID toolbox.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Portuguese Foundation for Science and Technology for their support for this work through project PEst-OE/EEI/LA0009/2011.

REFERENCES

[1] MATLAB Neural Network Toolbox – User's Guide R2012b

[2] Nørgaard, M. (1996b), "Neural Network System Identification Toolbox for MATLAB", Technical Report.

[3] Nørgaard, M. (1996c), "Neural Network Control Toolbox for MATLAB", Technical Report.

[4] Nørgaard, M., O. Ravn, N. K. Poulsen, and L. K. Hansen, "Neural Networks for Modelling and Control of Dynamic Systems. Springer, 2000.

[5] Levenberg, K., "A method for the solution of certain problems in least squares". Quart. Appl. Math., 2:164—168, 1944.

[6] Marquardt, D., "An algorithm for least-squares estimation of nonlinear parameters", SIAM J. Appl. Math., 11:431—441, 1963.

[7] Principe, J. C., Euliano, N. R., Lefebvre, W. C. Neural and Adaptive Systems: Fundamentals through Simulations, Wiley, New York,1999.

[8] Haykin S., Artificial Neural Networks: A Comprehensive Foundation. In: *IEEE Press*, New York.1995.

[9] Bache, K. & Lichman, M. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. 2013.

[10] Hugh Coolican, "Research Methods and Statistics in Psychology", 5th edition, Routledge, 2009.