Neural Networks Trained with the EEM Algorithm: Tuning the Smoothing Parameter

JORGE M. SANTOS [1,2], JOAQUIM MARQUES DE SÁ [1] AND LUÍS A. ALEXANDRE [3]
[1] Intituto de Engenharia Biomédica, Porto, Portugal
[2] Instituto Superior de Engenharia do Porto, Porto, Portugal
[3] IT – Networks and Multimedia Group, Covilhã, Portugal
jms@isep.ipp.pt, jmsa@fe.up.pt and lfbaa@di.ubi.pt

Abstract: - The training of Neural Networks and particularly Multi-Layer Perceptrons (MLP's) is made by minimizing an error function usually known as "cost function". In our previous works, we apply the Error Entropy Minimization (EEM) algorithm in classification and its optimized version using, as cost function, the entropy of the errors between the outputs and the desired targets of the neural network. One of the difficulties in implementing the EEM algorithm is the choice of the smoothing parameter, also known as window size, in the Parzen Window probability density function estimation for the computation of the entropy and its gradient. We present here a formula yielding the value of the smoothing parameter, depending on the number of data samples and on the neural network output dimension. Several experiments with real data sets were made in order to show the validity of the proposed formula.

Key-Words: - Entropy, Parzen, Smoothing Parameter, Cost Function

1 Introduction

The use of entropy and related concepts in learning systems is well known. The Error Entropy Minimization concept was introduced in [1] and we use the same approach in [2] with the goal of performing classification. The EEM Algorithm for classification use, as cost function, the Renyi's Quadratic Entropy of the error, between the output of the neural network and the desired targets. We have made a further optimization of the EEM algorithm in [3] introducing a variable learning rate to achieve both a faster convergence of the algorithm and an automatic adaptation of the value of the learning rate to the specific problem.

In order to compute the Renyi's Quadratic Entropy of the error one must choose a value for the smoothing parameter in the Parzen Window method, that is best suited for a specific data set. In all known works using this method, giving as example ours in [2,3] and other authors in [1,4], the value of the smoothing parameter is always experimentally selected. Knowing that this approach is not the most appropriate, in this paper, we study the behavior of the algorithm with different data sets and propose a formula to compute the smoothing parameter for a specific data set and neural network. In section 2 the EEM algorithm and further optimizations are presented. In section 3 the smoothing parameter for entropy and gradient computation is discussed and a specific formula is presented. In section 4 the results of several experiments are presented showing the validity of the formula. In the final section, we present the conclusions.

2 The EEM Algorithm

Let $y=\{y_i\}\in\Re^m$, $i=1,..,N$, be a set of samples from the output vector $Y\in\Re^m$ of a mapping $\Re^n\rightarrow\Re^m : Y=g(w,X)$, where w is a set of Neural Network (NN) weights, X is the input vector and m is the dimensionality of the output vector. Let $d=d_i\in\{-1,1\}^m$ be the desired targets and $e_i=d_i-y_i$ the error for each data sample. In order to compute the Renyi's Quadratic Entropy of e we use the Parzen Window probability density function (pdf) estimation. This method estimates the pdf as

$$f(e)=\frac{1}{Nh^m}\sum_{i=1}^{N}K(\frac{e-e_i}{h}) \qquad (1)$$

where K is a kernel function and h the bandwidth or smoothing parameter. We use the Gaussian kernel with zero mean and unitary covariance matrix:

$$G(e,I)=\frac{1}{(2\pi)^{m/2}}\exp\left[-(\frac{1}{2}e^Te)\right] \qquad (2)$$

The Reniy's Quadratic Entropy of the error e can be estimated, applying the integration of gaussian kernels [5], by

$$\hat{H}_{R2}(e)=-\log\left[\frac{1}{N^2h^{2m-1}}\sum_{i=1}^{N}\sum_{j=1}^{N}G(\frac{e_i-e_j}{h},2I)\right] \qquad (3)$$
$$=-\log V(e)$$

The gradient of $V(e)$ is:

$$F_i = -\frac{1}{2Nh^{2m+1}} \sum_{j=1}^{N} G(\frac{e_i - e_j}{h}, 2I)(e_i - e_j) \quad (4)$$

This gradient is back-propagated into the MLP the same way as in the MSE algorithm. The update of the neural network weights is performed using $\Delta w = \pm\eta \, {\partial V}/{\partial w}$. The $\pm$ means that we can minimize (−) or maximize (+) the entropy. We also proved in [2] that we can use this cost function for classification if the output of the neural network is on $[-a, a]$ and the targets are in $\{-a, a\}$, $a \in \Re$. Since one of the problems of the EEM algorithm is its slow convergence, we introduced in [3] an optimization of the algorithm based on the use of a variable learning rate (VLR), during the training phase, which applies the rule:

$$\eta^{(n)} = \begin{cases} \eta^{(n-1)}u & \text{if} \quad H_{R2}^{(n)} < H_{R2}^{(n-1)} \\ \eta^{(n-1)}d & \text{if} \quad H_{R2}^{(n)} \geq H_{R2}^{(n-1)} \end{cases} \quad (5)$$

being u and d the increasing (**up**) and decreasing (**down**) factors of the learning rate η in consecutive iterations.

This optimization improved the performance of the original algorithm and we called it the EEM-VLR algorithm.

3 Smoothing Parameter

In the EEM algorithm, the error entropy estimation is different according to the vector e dimension. This dimension depends on the number of MLP output neurons that depends on the number of classes, C, and their coding. If we use binary coding, the dimension of vector e will be $\lceil \log_2 C \rceil$ whereas if we use the 1-out-of-C coding the number of output neurons as well as the dimension of vector e will be equal to the number of classes. In the experiments that we performed, we obtained good results for both codings but we suggest the use of binary coding only when the number of classes is a power of 2; otherwise, we get an excessive number of outputs compared with the number of classes.

One of the problems of pdf estimation using the Parzen Window method, besides the choice of the kernel, is the choice of the smoothing parameter h.

In the EEM algorithm the value of h depends on: the different coding of the number of classes; the number of data samples; the dimension m of the vector e.

For continuous $f(x)$ the estimated density function will converge to the true density as $N \to \infty$, when:

$$h \to 0 \text{ and } Nh \to \infty \quad (6)$$

Silverman proposed in [6], for unidimensional cases assuming Gaussian distributed samples, and using a Gaussian kernel, the following formula for the smoothing parameter:

$$h_{op} = 1.06 \sigma N^{-0.2} \quad (7)$$

where σ is the sample standard deviation and N is the number of samples. For multidimensional cases, also assuming normal distributions and using the normal kernel, Bowman and Azzalini [7] proposed the formula:

$$h_{op} = \sigma \left(\frac{4}{(m+2)N} \right)^{\frac{1}{m+4}} \quad (8)$$

where m is the dimension of vector x.

An important fact that impedes, in our case, the use of formulas 7 and 8 is that our algorithm uses the entropy of e as a control variable, i.e., the algorithm progresses only if the entropy at a given iteration is smaller than at the previous one. Since the entropy value is proportional to the smoothing parameter value, used to compute it, if one uses a value for h proportional to the variance of e, one might be increasing, by this simple fact, the entropy value and the algorithm fails to reach the minimum. This means that we are limited to use a value for h that does not depend on σ.

Considering that variable e takes values, in the unidimensional case, in the interval $[-2,2]$, and that the maximum standard deviation in this case is 2, we considered to use this value to replace the standard deviation in formulas 7 and 8. So, these formulas became:

$$h_{op} = 2.12 N^{-0.2} \quad (9)$$

and

$$h_{op} = 2 \left(\frac{4}{(m+2)N} \right)^{\frac{1}{m+4}} \quad (10)$$

Note that, in the EEM algorithm, we only need to compute the entropy and its gradient; we do not

need to estimate the probability density function of e. This is a relevant fact because, in the gradient descent method, more important than computing with extreme precision the gradient is to get with relative precision its direction. In addition, the computation with extreme accuracy of the probability density function causes the entropy to have high variability. This fact could lead to the occurrence of local minima. Given these considerations, we do not hesitate in using h values higher than the ones usually proposed for pdf estimation. Taking into account the experimental results with several data sets we tried to formulate a rule that yields higher values of h for smaller data sets than those obtained with formula 10 and still inducing the same behavior.

We then arrived at the following proposed new formula with similar behavior as (10)

$$h_{op} = 25\sqrt{\frac{m}{N}} \tag{11}$$

Notice the decreasing behavior with N and increasing behavior with m as should be desired. A comparison between the values of h obtained with formulas 10 and 11 for different values of m is shown in Fig. 1.

In the several experiments that we performed using formula 11 the results were very satisfactory, as we will see in the next section.

4 Experiments

We performed several experiments with different real data sets, with different number of samples and different number of classes, summarized in Table 1

Table 1 – Real data sets used for the experiments.

Data sets	*Samples*	*Features*	*Classes*
Ionosphere	351	33	2
Sonar	208	60	2
wdbc	569	30	2
Iris	150	4	3
Wine	178	13	3
Pb12	608	2	4

(The Pb12 data set can be found in [8] and all the others can be found in the UCI repository of machine learning database). We also produced an artificial data set and used it as a 2-class and as a 4-class classification problem to try to establish the influence of the number of classes in the value of the smoothing parameter. The two versions of the artificial data set (that we call XOR-n), similar to an XOR problem, but with some noise added, are shown in Fig. 2.

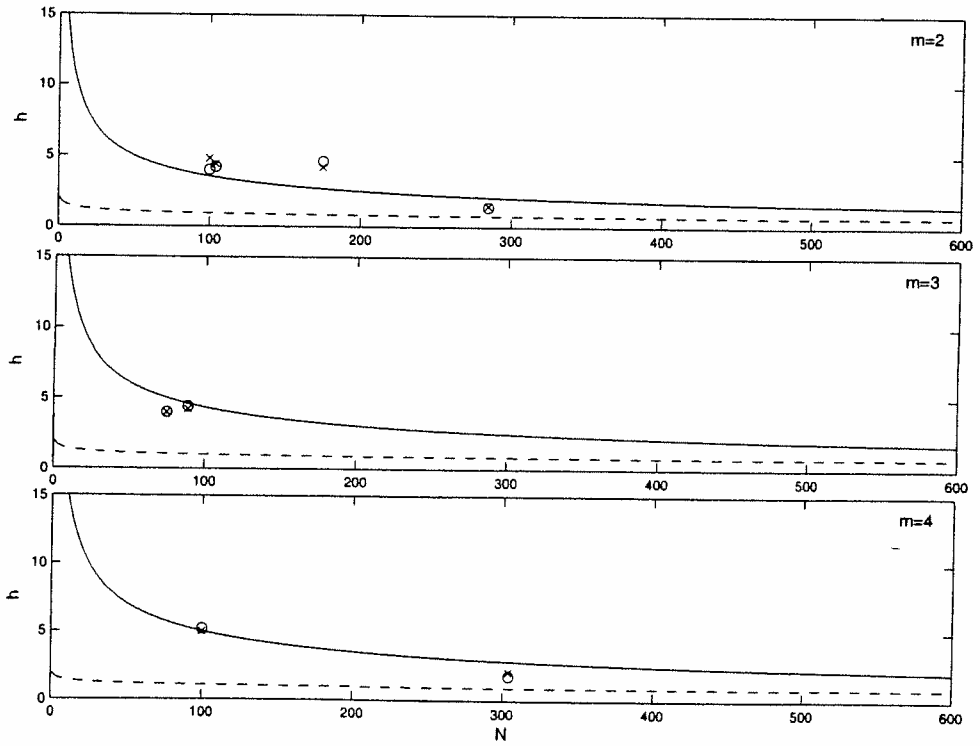

Fig. 1 - Value of h for formulas 10 (dashed) and 11 (solid) for m=2, 3 and 4. (Marked points refer to experiments with data sets mentioned in section 4).

In all experiments we used (I,n,m) MLP's, where I is the number of input neurons, n is the number of neurons in the hidden layer and m is the number of output neurons. We applied the cross-validation method using half of the data for training and half for testing. The experiments for each data set were performed varying the number of neurons in the hidden layer, the value of the smoothing parameter and using different number of epochs.

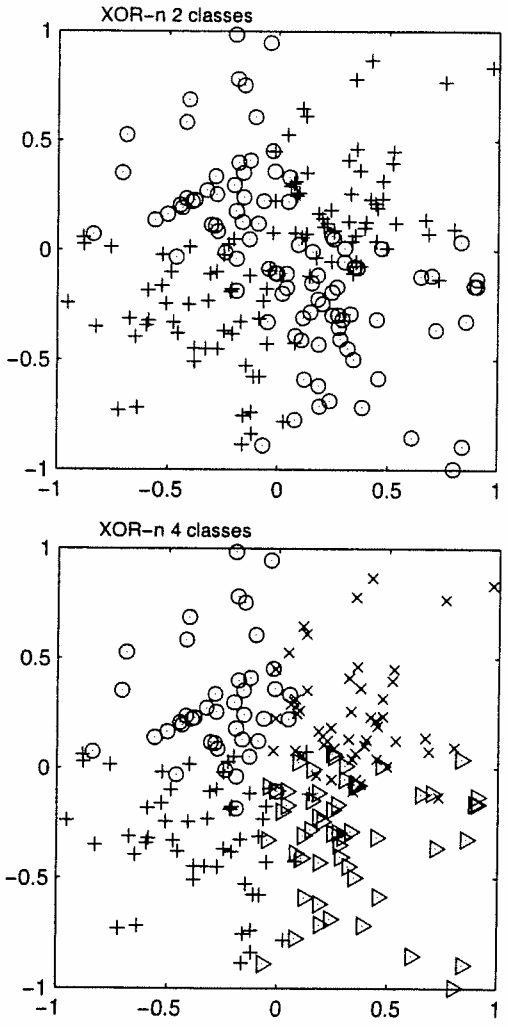

Fig. 2 - Artificial data set for the 2 first experiments.

The results are shown in Table 2 and Table 4. Each result is the mean error of 20 repetitions. For each experiment we highlighted the 10 best results performed with the same number of epochs in order to get the needed guidance about the optimum value for the smoothing parameter.

In Table 2 we show the results of the classification errors for the XOR-n data sets.

Table 2 – Errors (%) for XOR-n data sets, 2 and 4 classes (Resp. 80, 120 and 160 epochs from top to bottom)

XOR-n 2 classes

	Number of Hiden-Layer Neurons					
h	*2*	*4*	*6*	*8*	*10*	*12*
2,0	36,50	28,25	24,40	**23,10**	**23,38**	27,30
2,4	35,43	30,38	26,43	23,93	27,00	28,15
2,8	37,90	27,18	24,45	26,65	26,38	26,48
3,2	36,28	26,33	26,78	23,88	25,65	23,68
3,6	35,93	26,30	27,10	**22,43**	27,08	27,20
4,0	35,40	25,75	26,83	24,93	**21,10**	24,80
4,4	35,33	27,75	27,25	**22,30**	26,65	**22,20**
4,8	35,63	24,70	24,98	23,73	25,33	**22,65**
5,2	34,45	28,60	26,75	**23,05**	**21,20**	24,68
5,6	36,35	25,68	23,75	24,93	25,50	24,10
6,0	34,20	27,53	23,70	**22,38**	25,55	23,60
2,0	35,10	24,30	24,43	**22,75**	24,48	23,55
2,4	35,78	29,68	25,08	23,88	25,75	26,38
2,8	34,00	26,25	25,00	25,45	26,75	27,13
3,2	33,68	25,05	24,80	28,55	26,63	25,63
3,6	33,83	24,65	24,53	25,03	24,70	26,43
4,0	34,95	25,18	**21,20**	**22,88**	26,23	**22,15**
4,4	33,20	23,38	24,10	25,80	23,30	24,15
4,8	33,98	24,10	23,68	23,88	23,43	**23,00**
5,2	32,93	**22,60**	23,63	24,08	24,20	25,80
5,6	33,78	23,40	**22,75**	23,80	24,93	**23,25**
6,0	33,98	25,18	23,33	**22,75**	**22,63**	24,05
2,0	36,28	27,03	**22,78**	23,43	23,70	**22,55**
2,4	34,68	26,28	24,93	24,30	26,60	27,05
2,8	35,10	27,10	24,40	27,43	25,50	26,55
3,2	33,65	26,58	24,75	25,25	26,60	25,83
3,6	34,90	**22,00**	26,48	24,18	24,98	25,00
4,0	33,78	25,63	**21,28**	24,33	25,80	23,48
4,4	34,03	24,40	23,58	25,53	23,73	26,73
4,8	33,68	24,00	23,38	**22,55**	**22,78**	23,70
5,2	35,10	**21,90**	24,45	23,25	23,38	23,00
5,6	33,03	**21,33**	23,18	24,33	23,60	23,60
6,0	34,70	**22,03**	**22,08**	24,33	22,95	23,33

XOR-n 4 classes

	Number of Hidden-Layer Neurons					
h	*2*	*4*	*6*	*8*	*10*	*12*
2,0	19,00	19,08	18,13	19,30	18,88	19,10
2,4	17,65	18,13	19,10	18,73	**17,40**	18,83
2,8	18,90	17,98	17,90	**17,85**	19,15	18,23
3,2	17,75	18,18	17,73	18,30	17,75	18,58
3,6	19,10	18,50	17,95	18,30	18,53	18,30
4,0	**17,38**	**17,73**	18,65	18,20	19,13	18,38
4,4	18,18	18,78	18,00	**18,13**	18,08	18,70
4,8	18,68	17,98	18,48	19,18	18,50	18,25
5,2	18,83	**17,70**	**17,30**	19,53	18,70	18,70
5,6	18,68	18,33	**17,65**	18,40	19,15	19,13
6,0	**17,40**	19,10	18,10	19,53	**17,73**	18,45
2,0	18,63	19,43	18,63	20,18	19,70	20,63
2,4	20,38	19,28	18,55	18,95	19,73	20,45
2,8	18,33	18,98	19,30	18,48	18,88	18,38
3,2	18,28	18,23	**17,75**	18,15	18,53	18,65
3,6	18,20	**18,05**	18,95	18,73	**17,75**	18,80
4,0	18,48	18,23	18,80	18,70	18,15	**17,78**
4,4	17,68	18,18	18,98	18,48	18,05	19,15
4,8	17,60	18,60	18,93	18,35	18,08	18,50
5,2	**17,38**	**18,08**	17,80	**17,95**	18,63	18,78
5,6	**17,50**	18,23	**17,68**	18,65	18,78	19,15
6,0	17,85	18,08	17,85	18,18	**17,68**	18,55
2,0	18,35	19,45	19,80	19,20	**18,83**	19,48
2,4	18,60	18,63	19,00	19,30	19,85	19,08
2,8	17,93	19,50	19,75	19,60	19,35	19,38
3,2	18,88	19,38	19,68	19,35	19,38	19,58
3,6	18,08	**17,95**	18,90	18,90	19,90	19,05
4,0	**17,33**	18,85	19,03	**18,28**	19,65	19,28
4,4	18,43	20,08	19,28	19,60	19,45	19,53
4,8	18,38	19,13	19,30	19,05	**18,98**	19,18
5,2	**17,40**	19,68	**18,85**	**18,78**	19,98	19,33
5,6	17,60	**18,45**	**18,33**	19,30	19,58	19,18
6,0	18,18	18,70	19,45	19,03	19,38	19,08

Comparing the two tables, we can see that, an increased number of classes demand an increased value of the smoothing parameter. In the first case, (2-class problem), the optimum value for h is about 4.0 and in the second case, (4-class problem), the optimum value for h is about 4.8.
The classification errors for the real data sets are shown in Table 4. For each data set, we performed experiments with different number of epochs. In Table 3 we present the values of h for the minimum classification errors ("Best h" column) and the value of h that represents, in our opinion, those 10 smallest classification errors ("Suggested h" column). We also present the values proposed for each data set by formulas 10 and 11.

Table 3 – Values of h for each data set driven by the experiments and defined by formulas 10 and 11.

Data sets	Nr. of Classes	Nr. of Samples	Best h	Suggested h (10 min. Errors)	Form. 11	Form. 10
Ionosphere	2	351	4,6	4,2	2,67	0,85
Sonar	2	208	4,2	3,9	3,47	0,92
wdbc	2	569	1,4	1,6	2,10	0,78
Xor200	2	200	4,0	4,6	3,54	0,93
Iris	3	150	4,0	3,8	5,00	1,05
Wine	3	178	4,6	4,2	4,59	1,02
Pb12	4	608	1,8	2,1	2,87	0,93
Xor200	4	200	5,2	4,8	5,00	1,07

The experiments clearly show that, for small values of h, the classification errors are significantly large and, therefore, the use of formulas like (10), used for density estimation, are not appropriate for the EEM algorithm.
Figure 1 shows (crosses and open circles), the h values obtained in the experiments with all data sets where we can easily see their relation with the h values of the proposed formula 11. For each data set the crosses are the suggested h values given by the 10 smallest classification errors and the circles are the h values for the minimum classification error.

5 Conclusions

In this paper, we have presented a formula for the value of the bandwidth parameter, h, of the entropy estimation, to use in the EEM-VLR algorithm.
This formula, contrary to the ones proposed by Silverman and Bowman, does not depend on σ due to the iterative nature of the EEM-VLR algorithm.
We showed that it produces very good results using a set of experiments where the values proposed by our formula are much closer to the best ones found empirically, than the values obtained using the Silverman and Bowman formulas.
We expect that this contribution may save time and effort to many practitioners using this type of iterative entropic algorithms involving Parzen Window density estimation.

Acknowledgment

This work was supported by the Portuguese Fundação para a Ciência e Tecnologia (project POSI/EIA/56918/2004).

References:

[1] D. Erdogmus and J. Principe, "An Error-Entropy Minimization Algorithm for Supervised Training of Nonlinear Adaptive Systems", Trans. On Signal Processing, Vol. 50, No. 7, 2002, pp. 1780-1786.

[2] J. M. Santos, L. A. Alexandre, and J. Marques de Sá, "The error entropy minimization algorithm for neural network classification," in Proceedings of the 5th International Conference on Recent Advances in Soft Computing, A. Lofti, Ed. Nottingham Trent University, 2004, pp. 92–97.

[3] J. M. Santos, J. Marques de Sá, L. A. Alexandre, and F. Sereno, "Optimization of the error entropy minimization algorithm for neural network classification," in Intelligent Engineering Systems Through Artificial Neural Networks, C.H.Dagli, A. L. Buczak, D. L. Enke, M. J. Embrechts, and O. Ersoy, Eds., vol. 14. ASME Press, 2004, pp. 81–86.

[4] D. Erdogmus and J. Principe, "Entropy Minimization Algorithm for Neural Networks," Proceedings of the Intl. Joint Conf. on Neural Networks, 2001, pp. 3003-3008.

[5] D. Xu and J. Princpe, "Training mlps layer-by-layer with the information potential," in Intl. Joint Conf. on Neural Networks, 1999, pp. 1716–1720.

[6] B. W. Silverman, Density Estimation for Statistics and Data Analisys. Chapman & Hall, 1986, vol. 26.

[7] A. W. Bowman and A. Azzalini, Applied Smooting Techniques for Data Analysis. London: Oxford University Press, 1997.

[8] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton, "Adaptive mixtures of local experts," Neural Computation, pp. pp.79–87, 1991.

Table 4 - Errors (%) for the data sets Ionosphere (resp. 40, 60 and 80 epochs), Sonar (resp. 50, 100 and 150 epochs),Wdbc (resp. 40, 60 and 80 Epochs), Iris (resp. 40, 60 and 80 epochs),Wine (resp. 40, 60 and 80 epochs) and Pb12 (resp. 200, 250 and 300 epochs) for different number of hidden-layer neurons and several values of the smoothing parameter *h*.

Ionosphere - 40, 60 and 80 epochs

	Number of Hidden Layer Neurons																	
h	4	8	12	16	20	24	4	8	12	16	20	24	4	8	12	16	20	24
1,4	33,84	19,54	20,04	15,29	16,26	17,13	12,70	12,76	12,97	13,24	13,54	13,27	13,07	13,02	13,00	13,09	13,17	13,49
1,8	**12,47**	12,70	13,21	13,13	12,69	12,61	13,13	13,19	12,91	13,01	12,80	13,14	13,47	13,28	13,40	13,18	12,87	13,20
2,2	12,71	12,86	12,81	13,20	13,03	13,01	13,41	13,06	12,88	12,91	12,80	12,83	14,20	13,14	13,40	13,00	12,80	13,50
2,6	13,07	**12,40**	**12,37**	12,82	12,93	13,17	13,07	12,83	13,16	12,94	13,16	12,71	13,33	12,83	12,94	**12,53**	13,06	12,69
3	12,70	12,97	12,74	12,77	12,70	13,23	13,17	13,03	13,23	12,86	12,84	13,19	13,00	12,71	12,70	**12,33**	12,90	12,90
3,4	13,10	12,80	12,80	12,67	12,64	12,84	12,91	**12,50**	12,77	12,66	12,94	12,60	13,10	13,00	13,11	12,73	12,83	**12,59**
3,8	12,73	12,79	12,70	**12,26**	**12,46**	**12,26**	13,07	13,10	13,36	12,89	**12,24**	12,87	12,81	12,62	12,88	**12,24**	12,69	12,63
4,2	13,09	**12,39**	12,67	12,73	**12,59**	12,97	13,06	13,29	12,82	**12,46**	**12,41**	**12,41**	13,04	**12,40**	12,64	**12,34**	12,94	12,99
4,6	12,96	12,81	**12,06**	12,80	**12,43**	12,63	12,77	12,70	12,61	**12,44**	**12,29**	12,96	12,61	**12,41**	13,29	12,67	12,73	12,72
5	12,71	12,61	12,90	12,70	12,69	12,87	13,11	**12,44**	**12,44**	**12,29**	13,16	12,87	13,16	**12,56**	**12,60**	**12,60**	12,63	13,06

Sonar - 50, 100 and 150 epochs

	Number of Hidden Layer Neurons																	
h	2	4	6	8	10	12	2	4	6	8	10	12	2	4	6	8	10	12
1,0	49,47	48,63	49,18	48,97	48,56	50,65	48,36	49,90	49,30	49,04	48,41	49,81	50,82	50,48	50,00	49,86	50,82	49,95
1,3	48,49	45,77	45,67	43,22	41,37	43,73	36,71	35,36	37,74	35,53	34,52	31,42	31,63	28,10	27,81	29,88	26,90	25,51
1,6	24,90	23,65	23,61	22,96	22,88	23,17	24,64	23,44	23,41	24,06	23,94	23,80	25,53	24,52	23,92	24,18	25,00	23,61
1,9	24,78	23,27	22,96	**22,40**	**22,65**	23,03	24,02	24,52	23,51	22,86	23,29	23,08	24,23	24,52	23,08	22,33	24,13	24,52
2,2	23,17	23,68	23,53	24,28	23,77	23,66	24,16	24,33	23,49	22,84	24,42	23,08	25,31	24,45	23,27	**22,67**	22,74	23,15
2,5	24,37	23,05	**22,28**	23,08	23,34	24,18	23,99	**22,40**	24,52	24,35	22,81	**22,74**	24,06	24,06	**22,26**	22,67	23,41	23,25
2,8	23,82	23,70	23,61	**22,72**	**22,76**	**22,36**	23,51	24,47	23,00	22,81	23,10	23,87	24,33	23,85	23,99	23,00	22,86	**22,67**
3,1	24,04	24,06	22,98	22,98	22,98	**22,81**	24,64	24,04	**22,45**	23,82	23,29	**22,74**	24,98	23,17	23,29	23,05	**22,26**	22,88
3,4	24,11	23,82	24,18	24,04	23,08	23,13	24,06	24,35	23,15	**22,12**	23,27	**21,32**	24,42	23,56	**22,14**	23,32	23,17	**22,00**
3,7	24,50	23,82	**21,73**	22,81	**22,21**	22,81	23,56	23,58	23,37	23,44	23,44	**21,61**	25,41	23,58	23,51	22,84	23,27	**22,07**
4,0	24,26	23,15	23,42	23,56	**22,12**	22,96	23,58	23,77	23,13	**22,40**	**22,67**	**22,14**	23,73	23,89	23,08	**21,95**	**22,07**	**22,60**

Wisconsin breast cancer - 40, 60 and 80 epochs

	Number of Hidden Layer Neurons																
h	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10		
1,0	50,34	52,31	51,22	49,44	46,88	51,54	51,16	49,32	47,58	45,97	48,80	46,52	49,48	48,60	48,58		
1,2	8,83	10,37	19,04	19,74	21,80	10,71	11,46	11,48	9,74	19,47	4,41	**2,89**	5,49	5,83	6,11		
1,4	**2,53**	**2,33**	**2,65**	**2,57**	3,01	**2,79**	**2,80**	3,07	2,95	**2,82**	**2,77**	**2,81**	3,08	3,00	**2,92**		
1,6	2,77	**2,70**	**2,61**	**2,71**	2,77	2,97	**2,83**	**2,91**	**2,83**	2,96	**2,95**	3,23	3,75	3,06	**2,91**		
1,8	2,83	2,75	2,90	2,84	2,78	3,05	3,06	3,06	2,97	3,05	**2,96**	3,04	3,11	3,08	3,08		
2,0	2,84	2,87	**2,64**	**2,58**	2,97	2,97	3,01	3,06	3,07	3,08	3,26	**2,87**	3,12	2,99	3,06		
2,2	2,91	2,77	2,91	3,07	2,90	**2,72**	3,06	**2,81**	2,97	3,23	3,13	3,11	3,17	**2,95**	3,05		
2,4	2,67	2,76	2,92	2,84	2,59	2,94	3,04	3,04	**2,88**	3,24	3,06	3,26	3,22	**2,99**	3,36		
2,6	3,28	2,88	3,31	2,97	3,28	3,07	**2,93**	3,02	3,06	3,49	3,03	3,08	3,06	3,21	3,34		
2,8	2,89	2,85	2,79	2,94	2,95	3,06	3,02	3,25	3,07	3,28	3,15	3,05	3,28	3,20	3,40		
3,0	**2,69**	2,91	3,09	3,19	2,98	2,98	3,14	3,16	3,17	3,37	3,30	3,49	3,49	3,35	3,35		

Iris - 40, 60 and 80 epochs

	Number of Hidden Layer Neurons																
h	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10		
2,0	7,97	7,20	6,33	9,20	7,27	5,07	5,23	5,07	4,40	4,90	5,13	4,57	4,10	4,23	4,13		
2,4	7,17	6,63	4,60	4,63	4,24	4,80	**3,67**	4,50	**3,87**	4,30	7,83	**3,73**	4,80	4,23	4,23		
2,8	6,77	4,43	4,77	**3,83**	4,70	5,87	4,67	4,27	4,07	4,13	4,60	4,50	**3,77**	**3,57**	4,10		
3,2	7,37	4,87	**4,10**	4,40	4,40	8,40	4,10	**3,93**	4,27	**3,80**	5,27	4,40	**3,73**	3,83	3,93		
3,6	4,70	6,67	4,67	**4,10**	**4,03**	6,37	5,00	**3,97**	**3,97**	4,13	4,57	4,07	4,63	4,00	4,10		
4,0	5,50	4,30	4,83	5,50	4,53	6,60	**3,97**	**3,97**	**3,87**	4,53	5,57	5,73	**3,83**	**3,50**	**3,80**		
4,4	4,80	**4,20**	4,37	4,73	4,23	6,60	4,47	**3,80**	4,20	4,37	7,30	4,23	**3,67**	4,30	4,57		
4,8	5,50	**4,10**	4,33	4,57	4,37	6,37	**3,80**	4,40	4,50	4,07	5,17	4,53	4,03	4,20	4,00		
5,2	7,40	4,67	4,63	4,97	4,50	4,60	4,23	4,07	4,13	4,00	6,10	4,53	4,53	3,90	3,87		
5,6	5,54	**3,90**	**4,20**	4,47	4,27	5,40	4,53	4,20	4,40	4,10	4,70	4,30	4,57	4,10	**3,73**		
6,0	5,00	4,37	**4,07**	**4,00**	4,53	5,70	4,03	4,43	4,13	4,10	5,00	5,00	4,30	3,83	**3,80**		

Wine - 40, 60 and 80 epochs

	Number of Hidden Layer Neurons																		
h	4	6	8	10	12	14	16	4	6	8	10	12	14	16	4	6	8	10	12	14	16
1,4	46,41	34,92	29,30	22,89	17,89	16,60	17,75	6,77	2,25	3,79	2,30	2,81	2,61	2,56	**2,30**	2,92	2,67	2,56	2,39	2,67	2,39
1,8	8,54	2,61	2,28	2,28	2,28	2,53	2,45	**2,16**	2,64	2,33	2,61	2,39	2,50	2,36	2,47	2,64	2,42	2,75	2,64	2,95	2,59
2,2	2,28	2,44	2,19	2,56	2,50	2,67	2,36	2,95	2,28	2,42	2,56	2,61	2,59	2,30	2,53	2,64	2,61	2,73	2,59	2,28	2,64
2,6	2,30	2,84	2,25	**1,94**	2,36	2,42	2,33	2,28	2,39	2,19	**2,14**	2,36	2,33	2,28	2,47	2,81	2,64	2,78	2,47	2,64	2,89
3,0	2,47	2,42	2,36	**1,97**	2,39	2,31	2,22	2,42	2,39	2,44	2,08	2,56	2,28	2,22	2,81	2,56	2,42	2,84	2,61	2,87	**2,31**
3,4	2,33	2,59	2,19	2,05	2,78	2,08	**1,94**	2,50	2,22	2,89	2,22	2,30	**2,16**	2,45	**2,17**	**2,33**	2,44	2,42	**2,25**	2,47	2,42
3,8	2,64	2,59	2,14	2,11	2,30	2,30	2,33	**2,05**	2,39	2,64	**1,94**	2,42	2,64	2,22	2,81	2,59	2,84	2,70	2,78	**2,36**	2,75
4,2	2,14	2,28	2,22	2,11	2,64	**1,94**	2,42	2,45	2,36	**1,88**	2,75	2,45	**2,19**	2,44	2,78	**1,88**	2,67	2,36	2,87	**2,30**	2,53
4,6	2,56	2,28	2,16	2,39	**2,00**	**1,83**	**1,91**	2,47	2,31	2,78	2,39	**2,16**	2,28	2,61	3,01	**2,36**	2,50	2,95	2,39	2,89	2,87
5,0	2,05	**1,86**	2,42	2,16	2,22	**1,97**	**2,02**	2,89	2,53	**2,11**	2,36	**2,17**	2,39	2,28	3,06	2,84	2,61	2,44	**2,02**	2,42	2,70

Pb12 - 200, 250 and 300 epochs

	Number of Hidden Layer Neurons																	
h	2	4	6	8	10	12	2	4	6	8	10	12	2	4	6	8	10	12
1,6	24,09	16,17	13,26	14,46	14,71	14,62	22,29	15,07	12,76	10,21	11,65	11,23	18,73	12,10	10,63	10,78	10,68	9,03
1,8	21,25	11,53	**8,19**	9,29	8,88	**8,12**	22,89	**7,80**	8,50	8,29	**7,76**	8,36	25,80	8,00	**7,51**	**7,89**	8,39	8,25
2,0	25,49	9,11	**7,82**	**8,11**	**8,51**	8,92	24,94	**7,65**	**7,82**	8,40	**7,64**	9,08	25,71	8,36	8,43	**7,55**	8,08	8,17
2,2	28,72	8,79	**8,66**	**8,49**	8,85	9,31	28,46	10,35	**8,24**	**7,80**	**7,58**	8,92	27,02	9,40	8,17	**7,94**	**8,05**	**7,60**
2,4	28,64	11,65	8,77	9,18	**8,35**	9,88	30,83	12,38	9,61	8,53	**8,21**	8,81	29,65	10,77	8,41	8,70	9,76	**7,76**
2,6	28,94	9,73	9,19	9,34	9,70	**8,73**	30,30	13,22	8,77	10,58	9,08	9,26	29,03	10,78	9,43	**7,72**	8,31	**8,09**
2,8	28,77	11,99	11,28	9,36	9,61	9,36	28,75	10,97	8,90	**8,22**	9,89	9,38	28,63	10,21	8,77	9,83	9,37	8,82
3,0	29,35	11,51	11,37	9,05	**8,30**	9,89	28,29	10,81	9,60	10,43	10,15	8,63	28,31	11,55	**7,55**	8,26	8,51	8,89