Statistical Analysis of Multilayer Perceptrons Performances

Remus BRAD, Ioan MIHU and Macarie BREAZU

"Lucian Blaga" University of Sibiu
Computer Science Department
Bulevardul Victoriei 10, 2400 Sibiu, Romania
rbrad/mihuz/mac@cs.sibiu.ro

Abstract

The paper is based on a series of studies on the learning capabilities of multi-layered perceptrons (MLP). The complexity of these nonlinear systems can be varied, acting for instance on the number of hidden units, but we will be confronted with a choice dilemma, concerning the optimal complexity of the system for a given problem. By the mean of statistical methods, we have found that the effective number of hidden units is smaller than the potential size; some units have a "binary" activation level or a time constant activation. We also prove that weight initialization to small values is recommended and reduce the effective size of the hidden layer.

1 Introduction

We have study experimentally the problems of training different architectural sizes of MLPs. Investigations are based on statistical methods and don't make use of theoretical concepts concerning the system complexity [1] [2].

Being non-linear systems, transfer function is computing by composing the non-linear elementary functions of the cells. One can modify the complexity of the system by acting for instance on the hidden layer size. Therefore, we question about the optimal complexity of the system for a given task. This general problem is fundamental and exceeds the Neural Networks (NN) field; it requires the development of a complexity theory for the sample learning systems and leads to various research directions.

We have defined the potential size as given by the number of hidden units and the effective size as a result of network training on a given problem. Retrieving valuable information about the effective number of hidden units can be motivated by a great number of applications. It is difficult to find a compromise in terms of complexity, conducting to a high performance network.

In section 2, we present the framework of our experiments. Section 3 shows the result of our statistical investigations and defines the effective size of the MLP tested. A performance analysis is performed in section 4, by decomposing error in bias and variance terms.

2 Data Sets and MLP Architecture

We have employed in all our experiments, training and validation sets derived from the classification problem known as the Breiman waveforms [3]. This is a classification problem defined from three waveforms (h_1, h_2, h_3), each one expressed as convex combination of two waveforms with an additional gaussian noise. The first class elements are defined by:

$$x = rh_1 + (1-r)h_2 + \gamma \qquad (1)$$

where r is a random variable $r \in [0,1]$ and γ is the gaussian noise. The two other classes, 2 and 3, where obtained using the waveforms h_1, h_3 and h_2, h_3 respectively. The problem is nonlinear, each class having different variance arrays. Each waveform was sampled in a real number vector of size 21. A set of 3000 samples was used in the generation of training sets by a uniform distribution law. A set size varies between 300 and 1000 samples. Validation set consists of 5000 patterns.

The MLP employed has the following architecture: 21 input units, X hidden units and 3 output units. The size X of the hidden layer is modified during our investigations in the set (0, 5, 10, 15, 35, 60). The transfer function is hiperbolic tangent as recommended by [4]. MLPs were builded in the SN 2.8 NN simulator.

3 The Effective Number of Hidden Units

The number of hidden units governs the complexity of a MLP system. We will name it potential complexity of the system. In contrast, the effective capacity of the system will depend not only on the model but also on the system behavior on a given task [5].The last mentioned quantity could be specify the network architecture required for a special problem. The relationship between the potential complexity and the effective capacity is crucial to be apprehended, but difficult to measure and formulate. We have experimentally explored the MLP dynamic behavior, by tracing the hidden layer units activation and computing statistical measures.

3.1 Hidden Units Correlation

We first question about a possible redundancy between hidden units (h.u.) activations and the possibility to reduce hidden layer's size. The correlation between two h.u. can be measured by:

$$m_{ij} = \frac{\sum_t (x_i(t) - \overline{x_i})(x_j(t) - \overline{x_j})}{\sqrt{\sum_t (x_i(t) - \overline{x_i})^2 \sum_t (x_j(t) - \overline{x_j})^2}} \qquad (2)$$

with x_i^p - activation of the *i*-th h.u. for sample *p*, *p* – index on the total number of samples presented *N*, $\overline{x}_i$ - average of the *i*-th h.u. activations over all *N* samples.

Table 1 presents statistics over correlation values. As it can be observed, correlations are weak and redundancy could not be emphasized using this measure.

Table 1: Correlation statistics.

hidde nunits	Interval / Percentage					Average
	[0,0.1)	[0.1,0.2)	[0.2,0.3)	[0.3,0.4)	[0.4,0.5)	
5	44%	39%	14%	3%	0%	0.3736
15	55%	27%	11%	4%	2%	0.1839
35	42%	31%	16%	7%	3%	0.1698

Redundancy may be evidenced by the use of high order correlations, but result could be difficult to obtain and explain.

3.2 Hidden Units Activations

Each hidden unit is taking part in a certain manner and with a particular importance to the global network response. It is essential to identify cells behavior and evolution while training.

In the first test, weights where initialized using random values in [-1,1]. During the training stage, we will trace the hidden unit's activation values, on a test set of 1000 samples. For each unit, activation histograms where drawn, as show in figure 1. In this case, a 15 hidden cells MLP was considered. As we were expecting, after a number of epochs, cells behavior remain unchanged, some of them responding with the same activation, as depicted by histograms. Experiments have been concluded on different hidden layer networks and one could observe the same phenomenon.

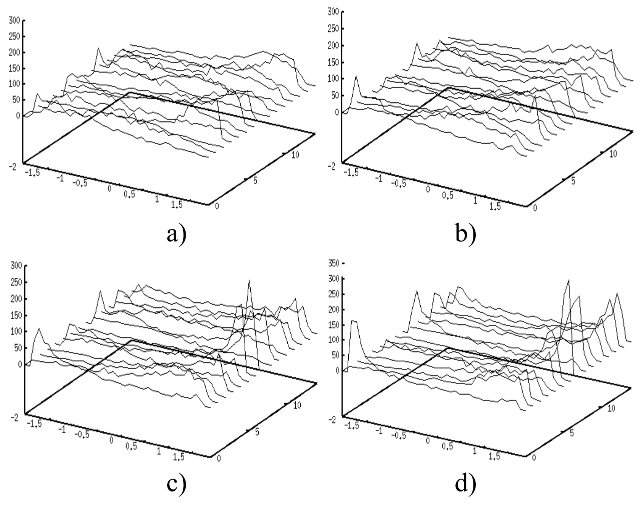


Figure 1: Hidden layer activation histograms after a) 5; b) 50; c) 200; and d) 1000 epochs, for 15 h.u., 300 training samples and weights initialization in [-1,1]

In the case of a weight initialization with small values, in the range [-0.01, 0.01], about 300 epochs will be required in "assigning" a special task to each hidden cell (see figure 2). One could observe a greater number of "constant" or "binary" h.u. activations. Starting from 300 epochs, training will only refine the role played by each unit.

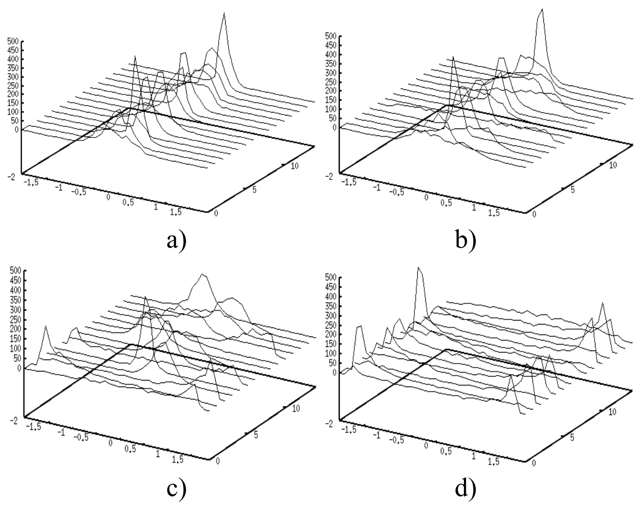


Figure 2: Hidden layer activation histograms after a) 5; b) 50; c) 200; and d) 1000 epochs, for 15 h.u., 300 training samples and weights initialization in [-0.01,0.01]

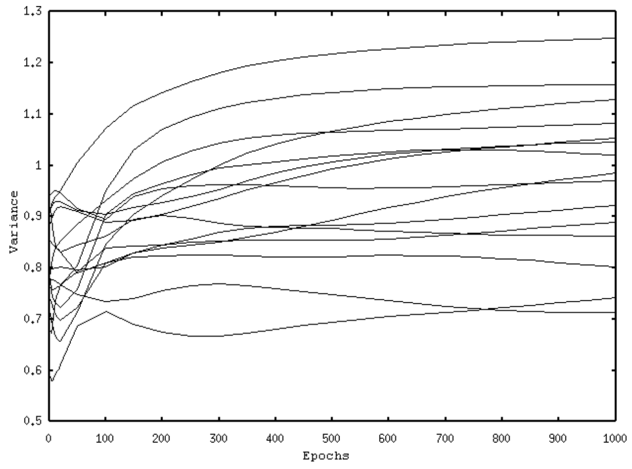
Investigations conducted on various architectures and training set sizes are reflecting the same conclusion: the "effective" number of h.u. is smaller than the potential one due to almost constant activations of some cells. A binary activation form could also be observed.

3.3 Activations Variance

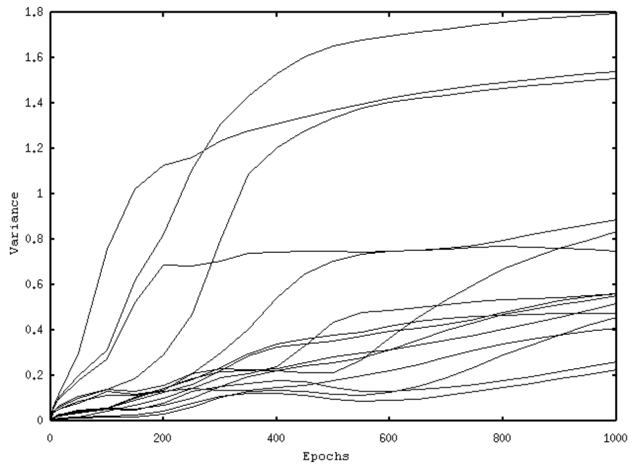
The histograms presented below are slides in time of the network state, acquired during training. A synthetic point of view could be obtained if each h.u. histogram will be represented by a value followed while training. Variance of the *i*-th h.u. is defined by:

$$\text{var}_i = \frac{1}{N}\sum_p (x_i^p - \overline{x_i})^2 \qquad (3)$$

where: p - index on the validation set, N- number of validation samples, x_i^p - activation value of the i-th h.u. on sample p, $\overline{x}_i$- average of i-th cell activations for all samples.



a)



b)

Figure 3: Hidden layer activation variances for 15 h.u., weights initialization in a) [-1,1] and b) [-0.01,0.01]

Analyzing the results shown in the figures below, one can deduce that cells with high order variances are the most active ones and a certain number of them could be considered as inactive. Comparing the two initialization cases, we can conclude that initialization with small weights is leading to a more accurate classification of units in the two categories. Between the most active h.u., high variance is also achieved by the binary response units.

3.4 Covariance eigenvalues

Hidden unit activation could be seen as a point in *n*-dimensional space. The size of this space can be retrieved by the rank of the covariance array of units activations. We are seeking a correlation of this points. In order to acquire the number of orthogonal axes for our space, we will first compute the covariance between two units, *i* and *j*:

$$\text{cov}_{ij} = \frac{1}{N}\sum_p (x_i^p - \overline{x_i})(x_j^p - \overline{x_i}) \qquad (4)$$

with the same notations as equation (3). Using the covariance array, which is symmetrically, we can estimate the effective size by the number of nonzero eigenvalues. Figures 4 show the time evolution of eigenvalues, for two MLP training cases. Values are sampled every 50 epochs.

Rank determination can be achieved by selecting a threshold. All eigenvalues below the given threshold are considered zero. Since for all architecture take into account, eigenvalues are very small, it is difficult to define the value of the threshold. Chosen as a fraction of the highest eigenvalue:

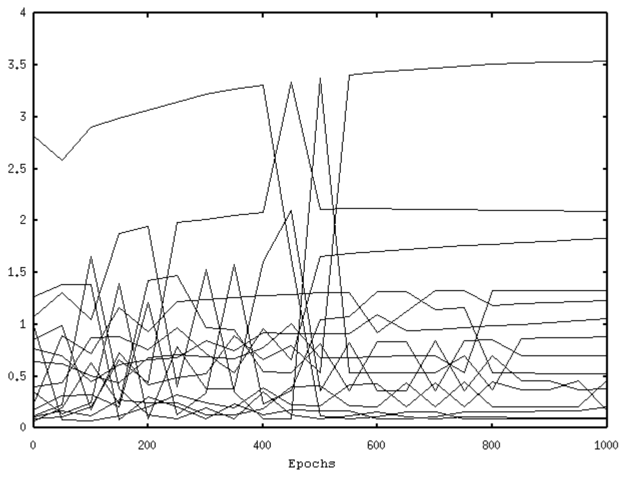
$$\text{Thr} = \frac{max\_eigenvalue}{f} \qquad (5)$$

for instance f=10, in the case of the 15 h.u. MLP, the effective number will be 12 and in the case of the 60 h.u., about 20. In table 2, we present the effective capacity of the hidden layer for several architectures and thresholds.

Table 2: The effective number of hidden units

f	number of hidden units		
	15	35	60
10	12	26	20
40	14	28	24
60	15	30	28

In any case, the effective capacity of the system is smaller than the potential one. The bigger is the size of the hidden layer, the smaller is the effective number. For the 15 h.u. MLP, no sensitive improvement was expected. This particular architecture seems to be suitable for the waveform problem. Information regarding the effective capacity could be used in pruning algorithms.


a)

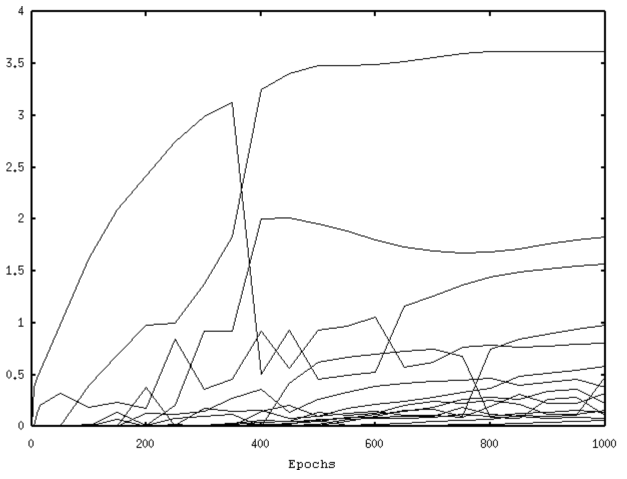

b)

Figure 4: Evolution of covariance eigenvalues for 15 h.u., weights initialization in a) [-1,1] and b) [-0.01,0.01]

4 Performance Analysis

In the previous section, we made a hidden layer analysis, but we never question about the network response when the number of hidden cells is varied. In this case we will describe some of the experiences that will hope explain the influence of the hidden layer in generalization and training error evolution [6].

4.1 Hidden Layer Size

To discover the performance dependence of the hidden layer, we will measure the error when the size will be changed from 0 to 60 hidden cells. The training set will consist of 300 different samples, used for all the 10 networks tested. The result will be the average of all network's behavior. The validation set will be identical in our case; 5000 samples will be tested. Figure 5 shows the results after 750 epochs, with a gradient descent error minimizing technique.

Generalization error decreases and has a minimum at about 100 epochs. For all cases, it's increasing with the number of training epochs. Distinction in evolution can be observed. The smallest error obtained is for the 5 hidden cells network, but the error value at the end of training is increasing with the size of the hidden layer. In conclusion, one can say that: the larger is the network, the worst are the results in generalization. However, this rule is inadequate if we consider the point of minimum generalization error.

If we examine the same figure, the over-training aspect seems to be emphasized for medium size networks (10-15 hidden cells) and imperceptible for large (60 hidden cells) and small networks (0-5 hidden cells).

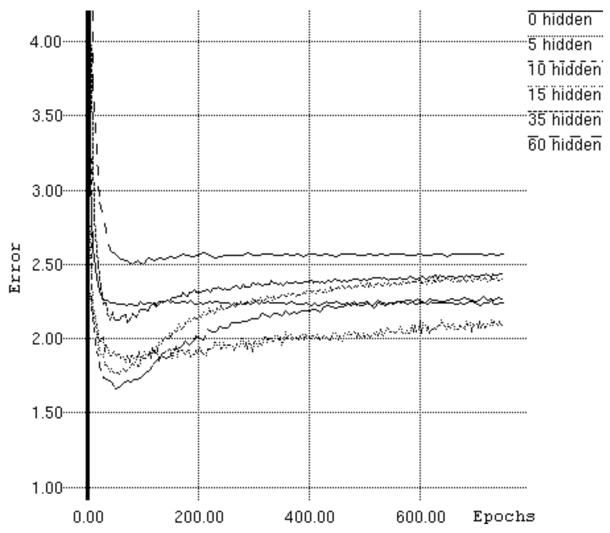


Figure 5: Validation error evolution corresponding to different architectures (from top to bottom, at the end, hidden layers have 60, 35, 15, 10, 0, 5 cells)

Training error evolution depicted in figure 6 is totally different. Networks without hidden layer have a constant and significant error; performance improved by the 5 h.u. network. For any moment after 100 training epochs, we can conclude that large size networks are learning faster and the error is decreasing with the number of hidden units. However, we will examine consciously this phenomenon.

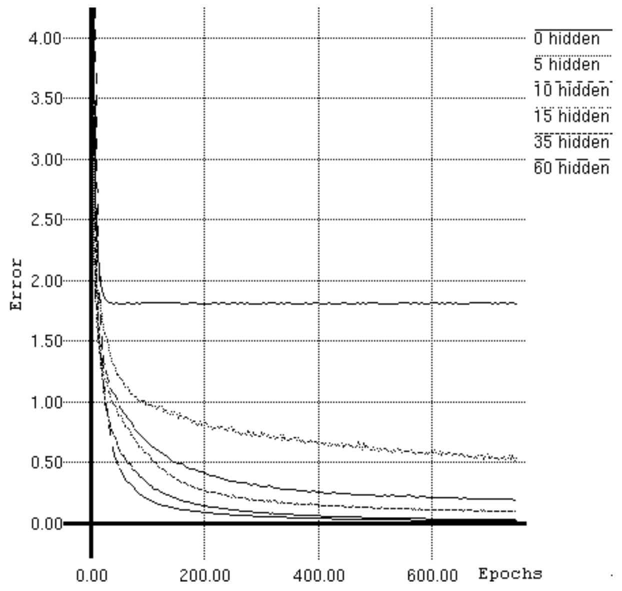


Figure 6: Training error evolution corresponding to different architectures (from top to bottom, at the end, hidden layers have 0, 5, 10, 15, 35, 60 cells)

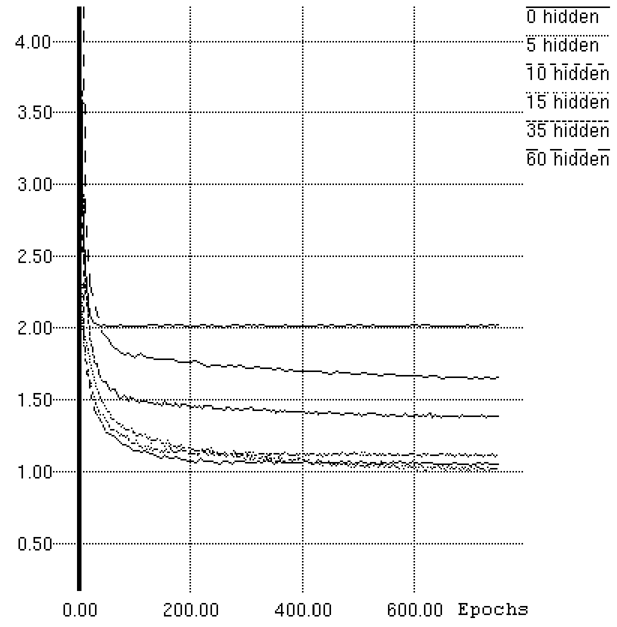


Figure 7: Bias evolution for different architectures (from top to bottom, at the end, 0, 60, 35, 15, 10, 5 h.u.)

4.2 Bias and Variance

The generalization error can be decomposed in two components: the *bias* and the *variance* [1]:

$$bias = \frac{1}{N}\sum_{i=1}^{N}\left\|\overline{y}(x_i) - c(x_i)\right\|^2 \qquad (6)$$
$$var = E_g - bias$$

where: N - number of samples; $c(x_i)$ - desired output for the i-th sample; $y(x_i)$ - the average of the 10 networks output for the i-th sample; E_g - generalization error

Theoretically, during the learning stage, the bias is decreasing and the variance is increasing with time. Due to their opposite evolution, a compromise concerning the two is difficult to consider. In our case, we have drawn for the 6 types of architecture, the bias (figure 7) and the variance evolutions (figure 8). At 750 epochs, MLPs with 5 hidden units have the smallest bias. For all networks with hidden layer, the variance is increasing, after a small hesitation at about 50 epochs.

Two points of interest are present in the previous pictures: the moment of minimum validation error and the final training moment at about 750 epochs. The curves drawn in figures 9 and 10 are obtained by saving weights values in the two cases discussed and by computing errors, bias and variance.

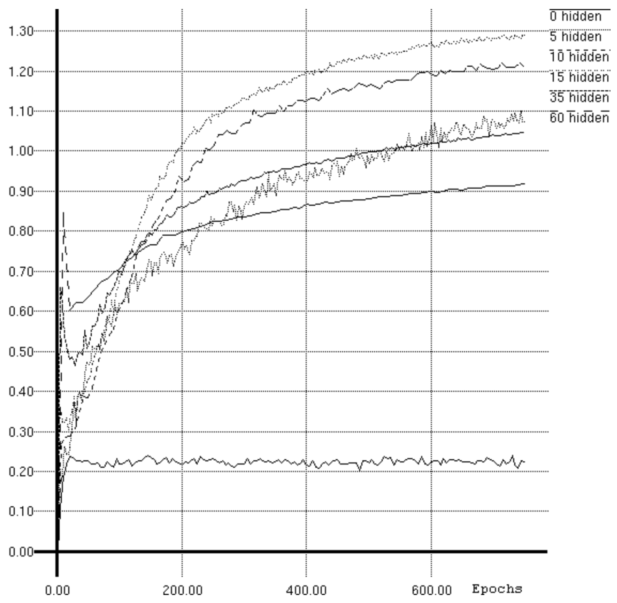


Figure 8: Variance evolution for different architectures (from top to bottom, at the end, 15, 10, 5, 35, 60, 0 h.u.)

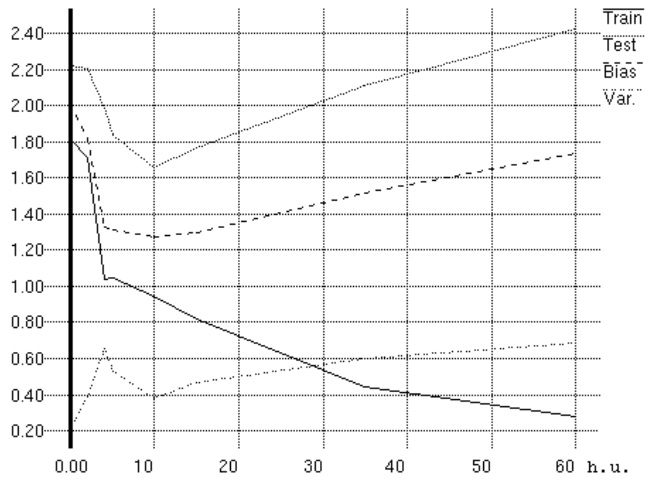


Figure 9: Errors, bias and variance depending of hidden layer width, for weights saved at validation error minimum

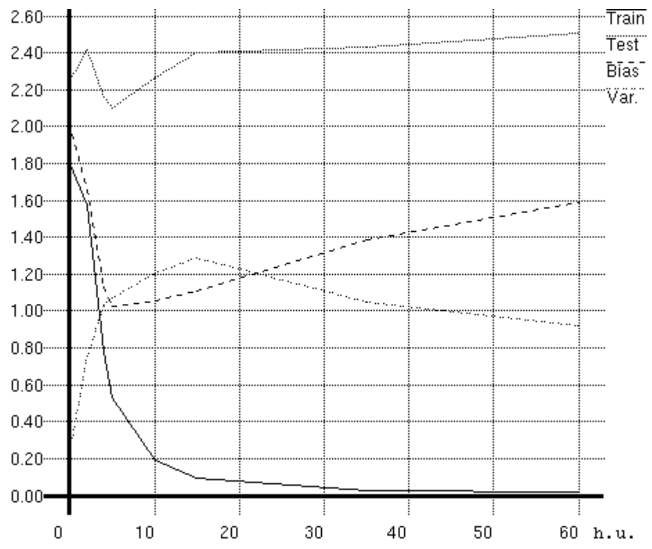


Figure 10: Errors, bias and variance curves depending of hidden layer width, for weights saved at the end of learning

In this case, bias and variance evolutions are different from those described by Gemann et al [1]. A MLP whose size was underestimated for a given task, will have limited degrees of liberty and will be too "rudimentary" to solve the problem. Therefore, the bias will be important and the variance weak. An overestimated MLP will have a large number of degrees of liberty and the process of optimization from different training sets will lead to different solutions. Thus, the corresponding variance will be significant.

Bias isn't always decreasing with complexity but after reaching a minimum, his further evolution is increasing. Following training and validation errors decreasing curves, we can conclude that large MLP have poor performances. Bias and variance have opposite evolution when hidden layer size is augmented. There is a moment when validation error reaches a minimum, corresponding to the information offered by the learning stage.

Compromise is achieved sometimes, by inserting in learning samples an important bias, who will support generalization. This phenomenon has no rigorous explanation, but some authors [7] have explored and analyzed deeply the problems discussed.

5 Conclusions

In the studies completed above, based on single hidden layer MLP, it has been possible to derive interesting results on network performances. In particular, we have been able to give some answers to the problems of MLP complexity. In addition, we show that it is possible to find a compromise between the size of hidden layer and the information of training samples. This analyses is only a first step in this direction and many questions remain unanswered. More work is required to test the significance of some of the observations. It would be of interest to analytically study the compromise of information versus complexity in learning. Finally, other possible directions of investigation include the dynamic behavior of MLP and the error surface.

6 References

[1] G. Geman, E. Bienenstock and R. Dourset, "*Neural Networks and The Bias Variance Dilemma*", Neural Computation 4, 1994, pp.1-58

[2] M.A. Kraaijveld and R.P.W. Duin, "The Effective Capacity of Multilayer Feedforward Network Classifiers, ICPR 94, 1994, pp.99-103

[3] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and Regression Trees*, Wadsworth Inc., Belmont CA, 1984

[4] G.B. Orr and K.-R. Muller, "*Neural Networks: tricks of the trade*", Springer, 1998

[5] J.E. Moody, "*The effective number of parameters: An analysis of generalisation and regularisation in nonlinear learning systems*", Advances in Neural Information Processing Systems 4, Morgan Kaufmann, 1992, pp.847-854

[6] R. Brad, "*Influences in Multilayer Perceptron Performances*", Acta Universitatis Cibiniensis, Seria Tehnica, vol XXVII, Editura Universitatii din Sibiu, 1997, pp.71-76

[7] T. Cibas, P. Gallinari and O. Gascuel, "*Experimental Investigation on the Complexity-Performance Relations in Multilayer Perceptrons*", ICANN'95, 1995