

Experimental investigation on the complexity-performance relations in Multilayer Perceptrons

Tautvydas Cibas^{*,**}, Patrick Gallinari^{*}, Olivier Gascuel[°]

* LAFORIA- IBP
Université Paris 6
4 Place Jussieu
75252 Paris cedex 05,
France
gallinari@laforia.ibp.fr

** LRI
Université Paris 11
91405 Orsay cedex,
France
cibas@lri.lri.fr

° LIRMM
Université de Montpellier
161 rue Ada
34392 Montpellier cedex 05,
France
gascuel@lirmm.fr

Abstract

This paper describes experimental investigations for exploring the dependence of Neural Networks behavior and capabilities on their complexity. Characteristic behavior patterns are worked out through an artificial problem. We analyze in particular the dependency of overfitting on NN complexity, characterize the bias-variance evolution of the error, and the effects of two regularization techniques. Results put in evidence the discrepancy between effective and theoretical capabilities of MLPs. It is also stressed that rules derived from general statistical results must be used with care in the case of neural networks.

1. Introduction

We investigate experimentally the behavior and capabilities of Neural Networks (NN) as a function of their complexity. We will not make use here of theoretical measures like e.g. VC-dimension or minimum description length, but will rather focus on some phenomena which characterize the actual behavior of NN.

The complexity trade-off for NN has been one of the main research theme in the domain over these last years. A well known problem is that of overfitting which has motivated several heuristics, e.g. early stopping, regularizing error terms, pruning. Well known statistical methods have been used for analyzing this phenomenon, e.g. structural risk minimization of Vapnik, bias-variance dilemma (Geman *et al.* 92). Although some well accepted general schemes do have emerged, many of them are often misused, others are simply not adapted to neural networks. These general ideas do not explain frequently observed phenomena, e.g. the many successful applications which have been reported with large networks and small training sets. We need to analyze NN in operational conditions and take into account besides the NN complexity, the training algorithm and the dataset characteristics, e.g. convergence or initialization conditions for the former and representativity or size for the latter. Similar arguments have been developed by (Kraaijeveld & Duin 94) who study the effective capacity of MLPs. We analyze here the dependency of overfitting on NN complexity, we characterize the bias-variance evolution of the error, show it is more complex than usually believed and use this framework

to analyze the effect and the potential of two regularizing techniques.

We will focus on the Multi-layer Perceptron (MLP), and use an artificial classification problem to illustrate our results. Although this last choice is questionable, we believe that such results also hold for a large number of other problems, namely for approximation and classification problems in large dimensional spaces. We use one hidden layer nets, model complexity is thus determined by the number of terms allowed in the model. As will be seen, for MLPs this seems to control the dimensionality of the approximation rather than the degree of smoothing. This should not be the case with local activation functions (e.g. Radial Basis Functions).

In 2 we present our test problem, in 3 we discuss the link between complexity and performance evolution during training. In 4 we use the bias-variance decomposition in order to analyze this behavior, we show in 5 how these quantities are influenced by regularization techniques.

2. A test problem

We have used a test problem (Gallinari & Gascuel 95) which is inspired from a well known waveform classification problem proposed in (Breiman *et al.* 84). This is a three class classification problem in \mathfrak{R}^{21} . Patterns from one class are a convex combination of two waveforms among three (h_1, h_2, h_3) (see Breiman *et al.* 84), each wave being defined in \mathfrak{R}^{21} , plus an additional gaussian noise. The three classes do have equal priors. Class 1 patterns are defined by:

$$x = uh_1 + (1 - u)h_2 + \varepsilon$$

with u a random variable in $[0,1]$ and $\varepsilon \sim N(0, 2 * I_{21})$, I_{21} being the identity matrix in \mathfrak{R}^{21} . Classes 2 and 3 are built in a similar way using respectively h_1-h_3 , $\varepsilon \sim N(0, 10 * I_{21})$ and h_2-h_3 , $\varepsilon \sim N(0, 26 * I_{21})$. This problem is non linear.

For our experiments, we have used independent training sets of size 300, and an additional test set of size 5000. Performances on test and training sets are average performances of 10 experiments performed with nets trained on 10 different sets. In order to avoid additional perturbations, all nets with a given architecture have been initialized with the same set of weights.

3. Network complexity and performance evolution

Figure 1 shows the mean square error evolution during training for MLPs whose number of hidden units varies from 0 to 60 and for two algorithms: a stochastic gradient descent and a conjugate gradient (Möller 93). The two techniques do have similar behaviors. Since this similarity may be observed for all the other experiments we have performed, we thus conclude that the phenomena which are put in evidence are not heavily dependent on the optimization technique used for training and will focus only on the stochastic algorithm in the following. Figure 1 illustrates some well known NN behaviors, e.g. if we observe the performances at their best value for each architecture, it may be seen that they increase until some "optimal" architecture is reached after what they decrease. However linking this behavior to the NN complexity is more difficult. A comparison of the temporal evolution of the error for the different architectures brings complementary informations and reveals several underlying phenomena. The overtraining phenomenon becomes more pronounced when the network complexity is increased from 0 to 10 hidden units, after what the phenomenon is reversed and for large networks, it is barely visible for both algorithms. This may appear counter-intuitive, but this corresponds to frequently encountered situations: overtraining is not a simple increasing function of the network complexity. One explanation might be the restricted possibilities of gradient algorithms, however our algorithms have proven to be very efficient on a large set of tasks. There are probably other effects, e.g. large network develop redundant informations on their hidden space. However, all tests we have performed indicate that this correlation is distributed among all units and is therefore complex to put in evidence.

Note that it is often proposed in the literature to use an oversized network and to stop as soon as overtraining appears. It is clear from fig. 1 that care should be taken with this procedure since large nets do not always exhibit this characteristic peaking phenomenon, moreover, nets with different complexities do not lead to similar performances.

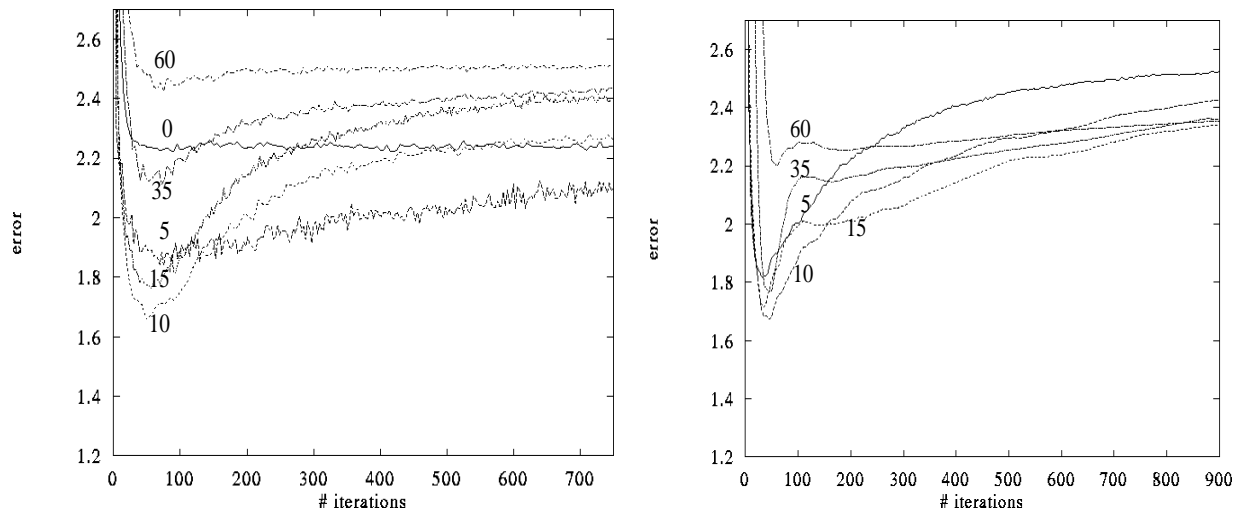


Figure 1 a (left), b (right): evolution of the performances (mean square error) during training for MLPs with a varying number of hidden units. (a) corresponds to a stochastic gradient descent and (b) to a conjugate gradient. Each curve corresponds to a two weight layer MLP, the number on the curve gives the size of the hidden layer.

4. Bias and variance evolution

Error may be decomposed into a bias and a variance term as it is shown below for the theoretical mean square error:

$$\begin{aligned}
 E_{X^*D, S_n} \left[\left(y_{S_n}(x) - d \right)^2 \right] &= E_{X^*D} \left[\left(E(d/x) - d \right)^2 \right] + E_{X^*D} \left[\left(E_{S_n} \left[y_{S_n}(x) \right] - E(d/x) \right)^2 \right] \\
 &\quad + E_{X^*D, S_n} \left[\left(y_{S_n}(x) - E_{S_n} \left[y_{S_n}(x) \right] \right)^2 \right]
 \end{aligned} \tag{1}$$

where subscripts "X*D" and "S_n" indicate respectively expectation with respect to the probability distribution of the data $p(X^*D)$ and to the training set S of size n . y_S is the transfer function of the network trained on data set S , x and d denote respectively the input and the desired output. The three right terms of (1) are respectively the Bayes, bias and variance components of the error. Each of them may contribute to the final error, the Bayes component is inherent to the problem and is a lower bound of the error. (Geman *et al.* 92) discuss the interpretation of this decomposition for controlling the degree of smoothing of non parametric methods. They point out the necessity of a compromise between the bias and variance terms for obtaining an adequate smoothing and good generalization performances. This has proven useful for setting parameters in techniques like kernel and nearest neighbor density estimation or curve fitting (Fukunaga 90). (Geman *et al.* 92) propose to control the complexity of NN through their number of hidden cells. There is a common belief that when one increases the number of hidden cells, the bias term is likely to be reduced whereas the variance should increase. Things are actually more complicated as will be seen below.

In our experiments, we have used the following estimations for the bias and variance:

$$\text{Bias}^2 = \frac{1}{N} \sum_{i=1}^N \left\| d_i - \bar{y}(x_i) \right\|^2 \quad \text{Var} = \frac{1}{N \cdot K} \sum_{i=1}^N \sum_{k=1}^K \left\| \bar{y}(x_i) - y_k(x_i) \right\|^2$$

where y_k is the output of the net trained on set S_k , \bar{y} is the mean output function for the K nets, and N is the test set size. Figure 2 represents the temporal evolution of these quantities during learning. For any architecture, bias decreases and stabilizes, whereas variance rapidly moves during the first iterations, reaches a minimum and then increases before stabilizing. Best performances of the different models do correspond indeed to a compromise between these two terms which is located near the minimum of the variance term.

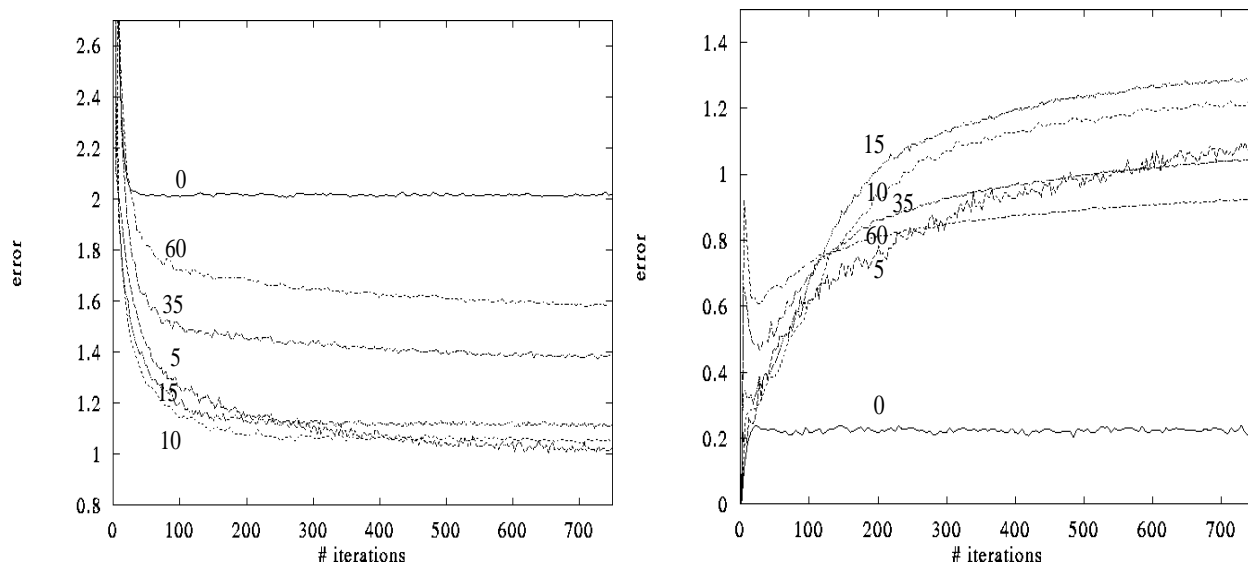


Figure 2 a (left), b (right): bias (a) and variance (b) evolution for the different architectures. Each curve corresponds to a two weight layer MLP, the number on the curve gives the size of the hidden layer.

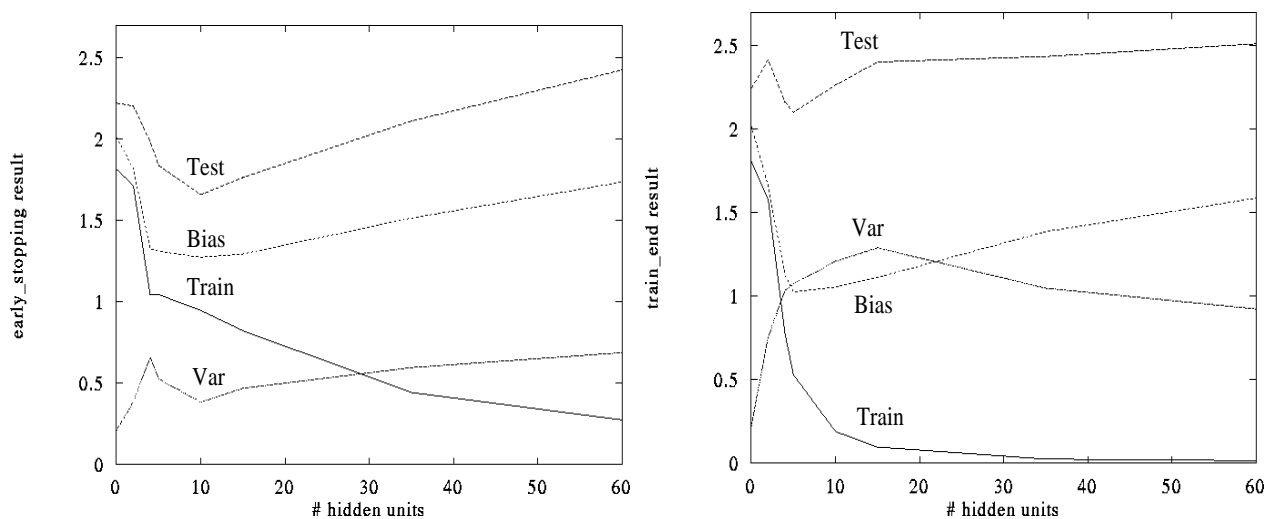


Figure 3 a (left), b (right): bias and variance evolution for the different architectures, the hidden layer size of which is indicated on the x axis. From top to bottom, curves correspond to performances on the test set, bias, variance and mean performance on the training sets. These quantities have been computed at the minimum of the error on the test set for each net (a) and after stabilization of the test error (b).

When comparing bias and variance for different architectures, we can see that bias first decreases when the complexity is increased from 0 to 10 hidden cells, after what it increases again until 60 hidden cells. This evolution is summarized in figure 3 a&b where we have plotted the values of the

bias and variance terms at the minimum of the test error (a) and after stabilization (b). The two figures exhibit the same bias behavior. Evolution of the variance term is more complex (fig.3-b). When measured after stabilization, it increases with the number of hidden units and then decreases. For early stopping, its evolution reflects directly the performances of the models on the test set.

5. Smoothing control

Varying the number of hidden units in a MLP does not appear to be a precise tool for the control of generalization performances. Smoothing may be controlled more effectively via early stopping or regularization. Early stopping allows to reduce the variance of the transfer function (fig.3) and increases the bias at least for networks of intermediate complexity. The early stopping solution lies on the path leading from the initial transfer function to the stabilized solution and thus depends on the initial values of the parameters and on the algorithm. Direct regularization gives more freedom to constrain the solution. We have investigated here the behavior of a simple weight decay procedure which applies different constraints to three sets of weights: threshold weights, input to hidden and hidden to output, they are denoted respectively W_1 , W_2 , W_3 . The reason for this is that these weights play different roles for the transfer function (Williams, 95). We have used the following cost function for training:

$$C = \alpha C_S + \sum_{i=1..3} \frac{\beta_i}{z_i} C_{W_i} \quad (2)$$

where C_S is the mean squared error and $C_{w_i} = 1/p_i \sum_{j=1}^{p_i} w_j^2$ with p_i the number of weights in set W_i . Performances are highly dependent on the choice of hyperparameters α and β . They have been set using either a validation set or the method of integrating over hyper-parameters (Buntine & Weigend 91). The latter allows an automatic determination of these parameters, they are being reestimated on line after each sweep through the training set by:

$$\alpha = \frac{kN}{\sum_{i=1}^N \|d_i - y_i\|^2} \quad \beta_i = \frac{p_i}{\sum_{j=1}^{p_i} w_j^2}$$

where k is the number of classes.

However because the α term in (2) becomes rapidly very small we had to use additional parameter (z_i in (2) which have been set by cross validation to $z_1 = z_2 = 500$, $z_3 = 100$) in order to compensate this effect. Note that these values are highly dependant on the weight initialization. Similar results have been obtained with the simpler form:

$$C = C_S + \lambda C_{W_3} \quad (3)$$

where λ has been set by cross validation to 0.2 for 15 hidden units and 0.5 for 60, the coefficients of the other weight groups being near 0, they have been neglected. The integration scheme also led to very small values for these coefficients. Note that (3) controls directly the variance of the weight distribution for the second weight layer.

These regularization techniques have been used on two architectures (15 and 60 hidden cells) with different characteristic behaviors. Performance evolution on figure 4 shows that overfitting has been suppressed for the two nets. Moreover, performances for both architectures are better than the best performances obtained via early stopping which confirms that adequate regularization may be more efficient than early stopping or hidden units number optimization.

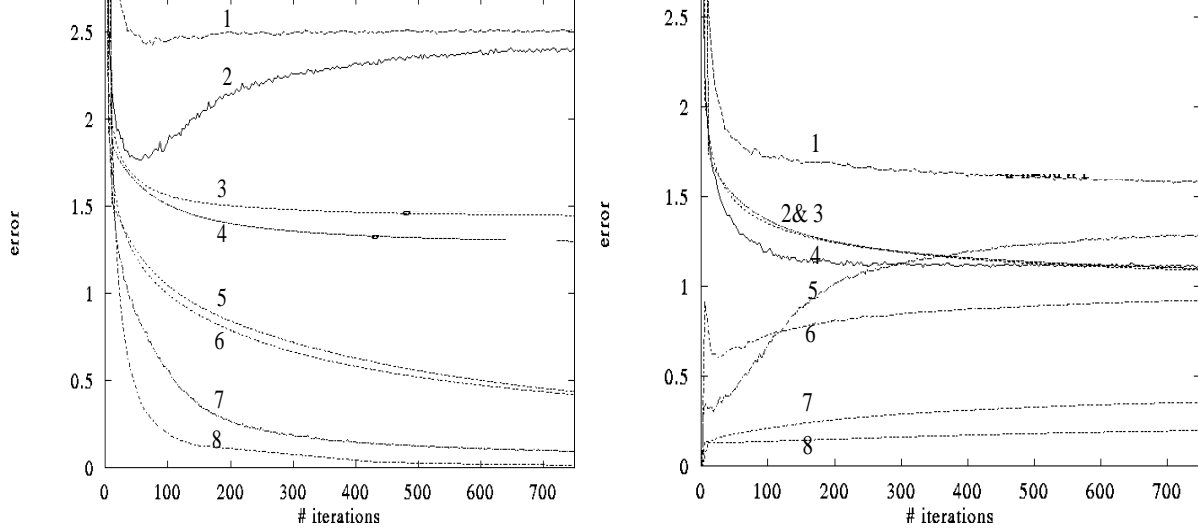


Figure 4 a (left), b (right): (a) shows the performance evolution during training of regularized 15 and 60 hidden units nets. The 4 curves in the center of (a) are respectively from top to bottom, test performances for 15 (3) and 60 (4) hidden units, training performances for 60 (5) and 15 (6) hidden units. For comparison we have plotted the performances of the corresponding unconstrained nets ((1) and (2) test for 60 and 15; (7) and (8) train for 15 and 60). (b) shows the bias and variance evolution: (1) bias for 60 hidden units, (2) regularized 60, (3) regularized 15, (4) 15, and variance for 15 (5), 60 (6), regularized 15 (7), regularized 60 (8).

We have been able to exploit the superior potential of the 60 hidden units architecture which gives now the best performance. Figure 4-b shows a comparison of bias and variance terms of these nets with those of the unconstrained versions. Bias reaches the same value as that of the 15 hidden units unconstrained net, which is near the optimal bias of the best net in figure 2. The variance has been considerably reduced (fig.4-b) for both systems which explains the performance increase. Constraining the solution via regularization terms may be pretty efficient. Note that hyper-parameter determination is not trivial, even with the integration scheme and has been performed using a validation set.

6. Conclusion

We have worked out through an example characteristic behavior patterns of MLPs. They reveal that care should be taken when using intuition gained from general results for analyzing NN behavior and that classical techniques for controlling the generalization performances may operate very differently and should be carefully chosen. Of course, theoretical investigation must be developed now in order to analyze these characteristic patterns.

References

- Breiman L., Friedman J.H., Olshen R.A., Stone C.J. (1984): Classification and Regression Trees, Wadsworth Inc.
- Buntine W.L., Weigend A.S. (1991): Bayesian Back-propagation, Complex Systems 5, 603-643.
- Fukunaga K. (1990): Statistical Pattern Recognition, 2nd ed., Academic Press.
- Gallinari P., Gascuel O. (1995): Statistique, apprentissage et généralisation; application aux réseaux de neurones, to appear in Revue d'Intelligence Artificielle.
- Geman S., Bienenstock E., Doursat R. (1992): Neural Networks and the Bias Variance Dilemma, Neural Computation 4, 1-58.
- Kraaijeveld M.A., Duin R.P.W. (1994): The Effective Capacity of Multilayer Feedforward Network Classifiers, ICPR 94, 99-103.
- Möller M., (1993): A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning, Neural Networks 6(4), 525-533.
- Williams P.M. (1995): Bayesian regularization and Pruning Using a Laplace Prior, Neural Computation 7, 117-143.