Use of some sensitivity criteria for choosing networks with good generalization ability

Yannis Dimopoulos[1], Paul Bourret[2], Sovan Lek[1]

[1] *UMR 9964, Equipe de Biologie Quantitative, Université Paul Sabatier*
118 Route de Narbonne, F-31062 Toulouse Cedes, France
[2] *Département d'Etudes et de Recherches en Informatique, ONERA-CERT*
2, av. Edouard Belin, F-31055 Toulouse Cedex, France

Abstract. In most applications of the multilayer perceptron (MLP) the main objective is to maximize the generalization ability of the network. We show that this ability is related to the sensitivity of the output of the MLP to small input changes. Several criteria have been proposed for the evaluation of the sensitivity. We propose a new index and present a way for improving these sensitivity criteria. Some numerical experiments allow a first comparison of the efficiencies of these criteria.

1. Introduction

Let us consider a MLP as a function $f:R^n \rightarrow R^m$,

$$f(x) = \phi_n[W_n\phi_{n-1}[W_{n-1}\phi_{n-2}[...\phi_1[W_1x]]]] \qquad (1)$$

where W_i stands for the weight matrix and ϕ_i stands for diagonal non-linear operators; the elements of which are activation functions.

It has been proved [1] that, under some weak assumptions, any function g: $R^n \rightarrow R^m$ can be approximated with an arbitrary accuracy by an MLP. Given the architecture, the learning process consists in computing weight matrices minimizing a given citerion:

$$J = (1/2)<\| g(x_j)-f(x_j) \|^2> \qquad (2)$$

where $<>$ is a statistical expectation with respect to input x_j, $j=1...N$, while $\|.\|$ denotes a Euclidian norm. This minimization is performed by the well known backpropagation method.

In practice we have to take into account that system inputs and outputs may be noise-contaminated. The real value of x_j and g are unknown, we only know $x_j + \varepsilon_j$ and $g(x_j) + e_j$, where ε_j and e_j are random noise. Thus, the best data interpolation may not be the best solution. What is expected is that small input variations may not induce too large an output variation and that MLP does not model the noise. The well known phenomenon of *overfitting* amounts to noise modelling. This *overfitting* appears when the complexity of the network is greater than the complexity of the system. In this case a small input variation causes a large output variation. Thus, methods of optimization of the network size have been proposed [2].

Usually *overfitting* is controlled by using a subset of the data for validation. This subset is not used for the computation of the weight matrix but for stopping the training process checking the model generalization ability. Separation of the data into the two subsets is not straight-forward. Several methods have been proposed belonging to the class of cross-validation methods. In this class of methods several questions arise. They are discussed in [3]:

- Part of the data is not directly used for estimating the weights.
- If the stop criterion is a given threshold for the generalization ability on the validation set, then this threshold may be reached from several initial conditions, corresponding to several solutions for W. On the other hand several local minima of (2) may be reached in the learning set after various learning phases which all meet the stop criterion, leading to different performances with the validation set.
- The solution is highly dependent on the dividing up of the data and on the initial conditions.

On the other hand there is not just one solution for the minimization of (2). The number of solutions is related to the size of the network and to the number and the distribution of the inputs. Thus, common sense leads to the choice of the local minimum with the lowest sensitivity.

2. Sensitivity criteria

The link between the modification of inputs, x_j , and the variation of outputs, $y_j = f(x_j)$, is the Jacobian matrix $dy/dx^T = [\partial y/\partial x]_{mxn}$. It represents the sensitivity of the network outputs according to small input perturbations. For a network with n layers the Jacobian is:

$$dy/dx^T = D_n W_n D_{n-1} W_{n-1} \ldots D_i W_i \ldots D_1 W_1 \qquad (3)$$

where D_i stands for a diagonal matrix the elements of which are the first partial derivatives of each activation function ϕ_i with respect to its input. The upper bound of the Jacobian norm measures the sensitivity of the network. This norm depends upon the number of weights which is related to the number of layers and the number of nodes per layer [4]. Matsuoka [5] proposed that this norm be added to the function J thus modifying the learning rule of the backpropagation. For a network with n inputs, one hidden layer with ni nodes, and one output (i.e. m=1), the gradiant vector of y_j with respect to x_j is $d_j=[d_{j1},\ldots,d_{je},\ldots,d_{jn}]^T$, with:

$$d_{je} = s_j \sum_{i=1}^{ni} w_{is} I_{ij} (1-I_{ij}) w_{ei} \qquad (4)$$

(under the assumption that a logistic sigmoid function is used for the activation). When s_j is the derivative of the output node with respect to its input, I_{ij} is the output of the ith hidden node for the input x_j, the scalars w_{is} and w_{ei} are the weights between the output node and the ith hidden node, and between the eth input node and the ith hidden node.

Let $\|d_j\|$ be the norm of the gradient in x_j. Then the sensitivity of the network for the learning data set is:

$$SSD = \sum_{j=1}^{N} \|d_j\|^2 \qquad (5)$$

However, this norm does not account for the variation of the sensitivity in the neighbourhood of x_j. Thus, networks with small first derivatives should be preferred even if the derivatives have a wide dispersion on the learning set. The dispersion may be represented by the second derivatives. Assuming that between several approximators the smoothest is the best one Bishop [6] proposed a modification of function J by adding a term which represents the curvature. Thus, the learning process minimizes the curvature of f as well as the error function (2). The curvature at x_j is:

$$k_j = (dd_j)^2/[1+(d_j)^2]^3 \qquad (6)$$

where dd_j stands for the derivatives of d_j (i.e. the second derivatives of y_j with respect to x_j). Deriving the gradient leads to the Hessian matrix with diagonal elements:

$$dd_{je} = y_j(1-y_j)(1-2y_j)\Big[\sum_{i=1}^{ni} w_{is} I_{ij}(1-I_{ij}) w_{ei}\Big]^2 + y_j(1-y_j) \sum_{i=1}^{ni} w_{is} I_{ij}(1-I_{ij})(1-2I_{ij}) w_{ei}^2 \qquad (7)$$

Webb [7] proposed to add to J a term dependent on dd_j. The coefficient of this term is proportional to the variance of the input noise.
An estimate of the curvature of f is:

$$CURV = \sum_{j=1}^{N} k_j \qquad (8)$$

Minimizing this index leads, of course, to smooth functions but their sensitivity may not be minimal. For instance the curvature of the exponential function $y = e^x$ is small (for $x > 0$) but it is obvious that its sensitivity is high; a large first derivative in comparison with the second derivative implies a small curvature. Thus, a good index of the sensitivity must take into account the norm of the Jacobian matrix but also the sensitivity of this matrix itself with respect to small disturbances of the network inputs. The smaller d_j and dd_j (the first and second derivatives of y_j with respect to x_j) the less the network is sensitive. Therefore,

$$D_j = \sum_{e=1}^{n} (dd_{je} d_{je})^2 \qquad (9)$$

indicates the sensitivity of the network in x_j given that each term of the sum is only large if both d_j and dd_j are large.
Let

$$GSI = k \sum_{j=1}^{N} (D_j)^2 \qquad (10)$$

where k is the number of dd_j sign modifications divided by N. We propose it be used to estimate the sensitivity of the network.

Obviously none of the indices *SSD*, *CURV* or *GSI* may be used as ending criteria for the learning phase, because they are not related to the behavior of the error function J. For instance if the weights of the network are initialized with values close to zero, *GSI* is small before the begining of the learning phase. Many of the well known approaches propose to add

a penalty term to J and thus optimize a linear combination of the two criteria: accuracy and sensitivity. However, in this way we can obtain several solutions which are local minima and there is no criterion for choosing one of them. Thus, we propose a hierarchical aggregation of the two criteria: several solutions which are each local minima of J are computed (all satisfying a given stopping rule) and the sensitivity index allows us to select the solution which should have the best generalization ability. In the following section we compare the three sensitivity indices on a few numerical examples.

3. Examples

The four experiments consisted of the approximation of two functions: $g_1(x) = 0.8 \sin(2\pi x)$ and $g_2(x) = \exp(-x^2)$ without and with an added noise. For these experiments we used the same network architecture with one hidden layer. Each experiment consists in computing the synaptic weights of the network 50 times with randomly chosen initial weights. The 50 learning phases are performed on the same learning set and are checked by a test which stops the learning phase before the begining of *overfitting*. This test is based on the average relative variance (*ARV*) computed on the learning set (a given part of the data). The learning phase is stopped when the *ARV* on the learning set falls under a given threshold. For a subset S *ARV* is:

$$ARV(S) = \frac{1}{\sigma_N^2 \, N_S} \sum_j \left(g(x_j) - y_j \right)^2 \qquad (11)$$

where σ^2_N is the variance computed over all the data and where N_S is the size of set S. This allows *ARV* to be independent on the size of the data set belonging to S. The generalization ability of the networks is estimated by *ARVval* (*ARV* computed on the validation set).

For the approximation of g_1, x is equaly spaced in [0,1] and the learning set consists of 25 values randomly chosen among 1025 in [0,1]. For the approximation of g_2 we used 100 values of x randomly distributed in [-2,2] and 20 values among them were randomly chosen for the learning set. The relationship between one sensitivity index and the generalization ability (*ARVval*) of the obtained networks is shown in figures 1 and 2 for the three presented definitions of sensitivity. Figure 1 is related to the approximation of g_1 (without and with noise 1a and 1b respectively), while figure 2 is related to the approximation of g_2 (without noise 2a and with noise 2b). The correlation coefficients between the three indices of sensitivity and the generalization ability where computed and are presented in table 1.

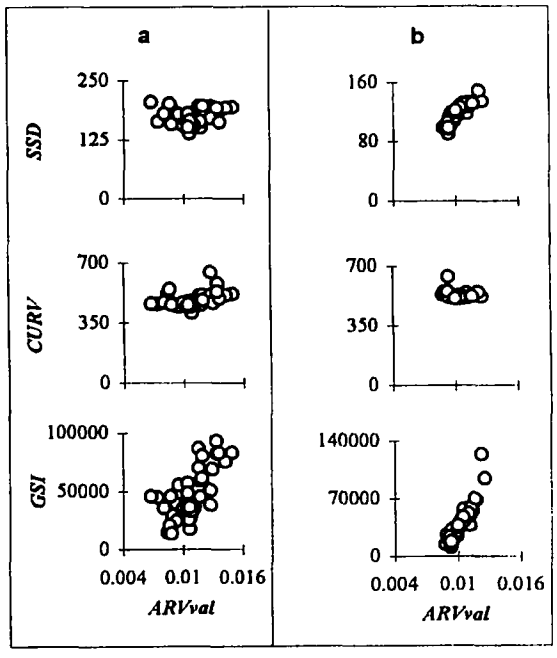

Fig. 1. Scattergrams of the three sensitivity indices versus the generalization ability, for g_1 without noise (a), and with noise (b).

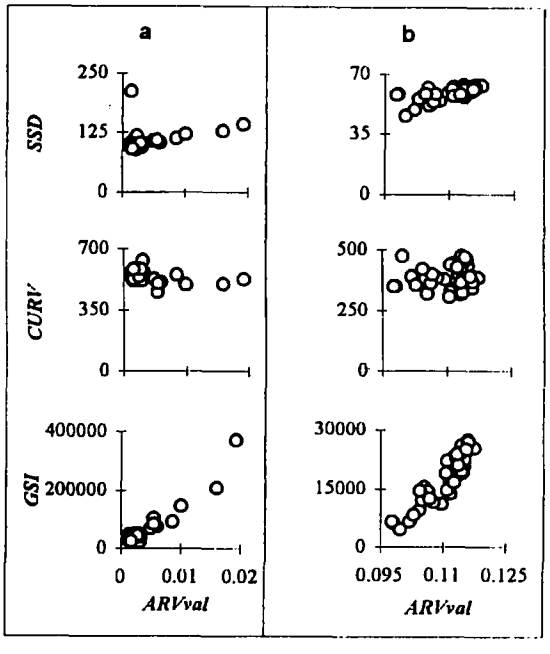

Fig. 2. Scattergrams of the three sensitivity indices versus the generalization ability, for g_2 without noise (a), and with noise (b).

	Sensitivity indices		
Approximations	SSD	CURV	GSI
g_1	0.089	0.406	0.664
g_1 + noise	0.874	-0.254	0.907
g_2	0.394	-0.337	0.958
g_2 + noise	0.694	-0.001	0.921

Fig. 1. Correlation coefficients between the three indices of sensitivity and the generalization ability for the four function approximations.

We performed too few experiments to be able to claim that the lower *GSI* is, the better *ARVval*. Nevertheless, we can compare the means of two sets of values: the first one consists of solutions such that their *ARVval* values are smaller than the *ARVval* median and the other one consists of solutions such that their *ARVval* values are larger than the median. For the four examples the difference of the two means is significant with a probability larger than 98%. The same statistical test was performed for *SSD* and *CURV* leading to the acceptance of the null hypothesis in all examples for *CURV* and in half of the examples for *SSD*. Moreover, we can estimate a threshold of *GSI*: for instance if we only choose a network such that its *GSI* is smaller than the first quartile of the *GSI* values then we can say that its *ARVval* would have more chance of being smaller than network with larger values of *GSI*.

4. Conclusion

Using a sensitivity index to choose a network allows us to avoid the local minima corresponding to the most sensitive networks. This computation can be easily implemented in the backpropagation algorithm. Obviously, the computation of the set of W matrices can be performed in parallel. This method, used together with an approach optimizing the size of the network, should be very efficient for designing networks with good generalization ability.

References

[1] G. Cybenko. Approximations by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303-314, 1989.

[2] J.H. Friedman. An overview of predictive learning and function approximation, in V. Cherkassky, J.H. Friedman, H. Wechsler eds., *From Statistics to Neural Networks, Theory and Pattern Recognition Applications*, Springer-Verlag, pp. 1-61, 1993.

[3] A. S. Weigend, B. A. Huberman and D. E. Rumelhart. Predicting sunspots and exchange rates with connectionist networks, in M. Casdagli and S. Eubank eds., *Nonlinear Modeling and Forecasting*, Addison-Wesley, pp. 395-432, 1992.

[4] L. Fu and T. Chen. Sensitivity analysis for input vector in multilayer feedforward neural networks, *IEEE Internatioal Conférence on Neural Networks, San Francisco, California*, pp. 215-218, 1993.

[5] K. Matsuoka. An approach to generalization problem in back-propagation learning, *Proc. INNC'90, Paris*, pp. 765-768, 1990.

[6] C. M. Bishop. Curvature-driven smoothing in backpropagation neural networks, *Proc. INNC'90, Paris*, pp. 749-752, 1990.

[7] A. R. Webb. Functional approximation by feed-forward networks: a least-squares approach to generalization, *IEEE Trans. on Neural Networks*, vol. 5, no. 3, pp 363-371, 1994.