ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 363-369

Orthogonal Least Squares Algorithm Applied to the Initialization of Multi-Layer Perceptrons

V. Colla[†], L.M. Reyneri[††], M. Sgarbi[†††]

[†] Scuola Superiore Sant'Anna, Pisa (I), e.mail: vale@mousebuster.sssup.it
[††] Politecnico di Torino (I), e.mail reyneri@polito.it
[†††] Scuola Superiore Sant'Anna, Pisa (I), e.mail mirkosg@tin.it

Abstract. An efficient procedure is proposed for initializing two-layer perceptrons and for determining the optimal number of hidden neurons. This is based on the Orthogonal Least Squares method, which is typical of RBF as well as Wavelet networks. Some experiments are discussed, in which the proposed method is coupled with standard backpropagation training and compared with random initialization.

1. Introduction

A suitable and efficient initialization procedure allows to set the initial values of the weights of a network not far from the optimal values determined by training: so doing, the training procedure takes a shorter time to reach the optimal values and therefore considerable saving of computation time is achieved.

In networks design, great importance must be attributed also to a correct choice of the number of hidden neurons, which helps avoiding problems of overfitting and contributes to reduce the time required for the training without significantly affecting the network performance.

Usually in literature these problems are faced separately for each neural paradigm, and very few are the attempts to apply techniques studied for a particular kind of network to a different paradigm. On the other hand, it has been recently pointed out [1] [2] that most of the neural paradigms, such as Multi-Layer Perceptron (**MLP**), Radial Basis Function (**RBF**) networks and Wavelet Networks (**WN**) [3], can be viewed under a unified perspective by means of the Weighted Radial Basis Function (**WRBF**) paradigm. One of the most important consequences of unification is that some initialization and training procedures studied for one particular neural paradigm could be applied with slight modifications to the other paradigms.

The focus of this paper is on the initialization of two-layer perceptrons and on the determination of the number of hidden neurons by applying the Orthogonal Least Squares method (**OLS**) [4], which is typically used by RBF networks as well as WNs.

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 363-369

2. Initialization of Two-Layer Networks

The issue of function approximation can be seen as a non-parametric regression problem, as there is no or very little *a priori* knowledge about the function to be estimated. The model can represent a very large class of functions and contains many free parameters which usually have no physical meaning in relation to the problem. The model obtained with two-layer WRBF feedforward networks with a linear output layer has the following form:

$$f(\mathbf{x}) = \sum_{i=j}^{M} a_j \cdot F\left(\mathbf{w}_j^T \boldsymbol{\Delta}_m(\mathbf{x} - \mathbf{c}_j) + b_j\right) + a_0 \qquad (1)$$

where $\mathbf{x} \in \mathbf{R}^N$ is the input vector; M is the number of hidden neurons, which are identified by a *weight vector* $\mathbf{w}_j$, a *center vector* $\mathbf{c}_j$ and a *bias* b_j; $\boldsymbol{\Delta}_m(\mathbf{x}-\mathbf{c})$ is a vector in $\mathbf{R}^N$ whose entries are $(x_j - c_j)$ $(j = 1, \ldots, N)$ for $m = 0$ and $|x_j - c_j|^m$ for $m \neq 0$; $F(z)$ is the *activation function*, which typically is a sigmoidal or exponential function, as well as a radial wavelet.

In practical applications, a set of sample input-output pairs $(\mathbf{x}_k, \mathbf{y}_k)$ $(k = 1, \ldots, K)$ is given, which should be fit by the model and, due to the high non-linearity in the input-output relation (1), the search for the optimal parameter values require iterative numerical procedures.

The idea which lays behind the OLS algorithm (introduced in [4] for RBF networks and adopted in [3] for WNs) is that, once m is chosen and $\mathbf{w}_j$, $\mathbf{c}_j$ and b_j are fixed, the model (1) depends linearly on the parameters a_j, which can therefore be determined by the standard Least Squares (**LS**) method. Therefore the problem is essentially divided in three steps:

- the construction of a *library* of candidate regressors $(F(\ldots))$ for model (1), namely a finite set of scaled and translated versions of $F(z)$, each one associated with a set of parameters $(\mathbf{w}_j, \mathbf{c}_j, b_j)$;
- the selection of a reduced number M of these regressors on the basis of the available sample data, by selecting among all the library functions those which give the greatest contribution to the approximation;
- the determination of the weights a_j of the output layer via LS technique.

The OLS procedure can lead to a sub-optimal solution, which can correspond to a local (not global) minimum of the approximation error: this is not necessary a drawback if the achieved error is small enough for the considered application.

2.1. Creation of the MLP Library

As far as standard RBF networks [4] are concerned, the creation of the function library consists in centering a function with fixed spreading factor in each data point; if the number of data is too large, suitable criteria must be applied to eliminate some useless functions. In [3] an almost similar procedure is applied to

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 363-369

WN; in that case the library elements, generated by distributing the parameters on a so-called dyadic grid, are grouped in *levels* on the basis of the dilation parameters and the functions of each level are orthogonal to those belonging to other levels (due to the properties of wavelet functions). Moreover, in [3] a method for reducing library size is proposed, which eliminates the functions of the library whose support contains less than a fixed number N_p of sample points. In practice, by support we mean the *hyper-cubic* subset of input space where the function is higher than a given threshold.

In MLPs, internal activation is a linear combination of its inputs, which means $\mathbf{c}_j = \mathbf{0}$ and $m = 0$ in (1), and $F(z)$ is a hyperbolic tangent:

$$F(z) = \frac{e^{z/\gamma} - e^{-z/\gamma}}{e^{z/\gamma} + e^{-z/\gamma}} \tag{2}$$

where γ is a factor depending on the dimension of the input vector which helps, together with the input normalization, to maintain the value of the neuron activation within a limited range. We adopt $\gamma = 10N$.

The parameters to be discretized are $(\mathbf{w}_j, b_j)$: the weight vector affects the direction and steepness of the hyperbolic tangent, while the bias determines translation. In [5] an OLS-based initialisation procedure was alredy proposed for MLP, but the weight vectors included in the library of candidate regressors were randomly chosen and no attempt was made to determine an optimal network dimension. The approach proposed here is a compromise between those in [4] and [3]: for each sample point, a set of weight vector (distributed on a regular lattice) is created depending on a integer parameter called for analogy *level*.

In the proposed method, at level l, each vector entry assumes all the 2^l odd integer values in the range $[-(2^l - 1), (2^l - 1)]$, while the admissible values for the bias are all the $2^l - 1$ even integers in $[-(2^l - 2), (2^l - 2)]$. Such ranges apply when the input values are scaled in the range $[-1, 1]$.

In practice, it is not needed to consider opposite weight vectors, because they introduce in the library two functions of opposite slope and identical position on the input space and therefore obtainable by multiplying each other by -1 (thanks to the symmetry of $F(z)$), therefore having $-a_j$ instead of a_j in the output layer. This is automatically handled by the Least Squares algorithm (third step). As a consequence, the first entry of the input vector is bound to assume only the $2^l - 1$ positive odd values in the range $[1, (2^l - 1)]$.

All possible combinations of biases and weights are considered, which means $N_{TOT} = (2^l - 1)2^{ln-1}$ possible sets of parameters $(\mathbf{w}_j, b_j)$. The set of parameters at level l contains also all the sets associated with the lower levels.

To select only the useful regressors among all library functions, one should consider the "region of interest" (instead of the support) of the hyperbolic tangent, namely the region where ∇F is not negligible, as in the other regions, where the function is almost "flat", sample points can lie everywhere without affecting the evaluation of the model.

Thus, the procedure proposed in [3] can be adapted to hyperbolic tangent functions: a library function is eliminated if less than N_p sample points are

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 363-369

L	N_p	N_{TOT}	N	M	Initialization		After 30s	After 1200s
					T_{in} (s)	NRMSE	NRMSE	NRMSE
4	1	120	18	11	1	$2.33 \cdot 10^{-2}$	$2.33 \cdot 10^{-2}$	$2.33 \cdot 10^{-2}$
5	1	496	42	12	2	$2.36 \cdot 10^{-3}$	$2.36 \cdot 10^{-3}$	$2.36 \cdot 10^{-3}$
6	1	2016	90	12	1	$2 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	$2 \cdot 10^{-2}$
7	1	8128	200	14	17	$1.77 \cdot 10^{-3}$	$1.77 \cdot 10^{-3}$	$1.77 \cdot 10^{-3}$
Random initialization				11	0.1	1.09	0.1	$4.26 \cdot 10^{-2}$
Random initialization				12	0.1	1.17	$8.96 \cdot 10^{-2}$	$4.3 \cdot 10^{-3}$
Random initialization				14	0.1	1.03	0.12	$4.52 \cdot 10^{-3}$

Table 1: Results of some experiments for approximating function (4); T_{in} is the time required for initialization. The network performances are compared after initialization and for two fixed values of CPU time elapsed.

such that the absolute value of the activation $z_j = (\mathbf{w}_j^T \mathbf{\Delta}_m(\mathbf{x} - \mathbf{c}_j) + b_j)$ associated with its parameters set is lower than γ. So doing, the standard fast OLS procedure [4] can be applied to a restricted set of $N < N_{TOT}$ functions.

The regressors selection could be terminated when a maximum number of functions is reached or when a minimum error value is obtained; a compromise between model complexity and approximation accuracy is achieved with more sophisticated stop criteria, such as to reach the minimum of the *Akaike Final Prediction Error* [6], which is used in the experiments described below.

3. Numerical Results

In order to assess the performance of the proposed initialization method, some experiments have been performed in which a standard MLP network is initialized and trained for function approximation; training is performed with standard backpropagation.

As a performance index, we adopt the *Normalized Root Mean Square Error* (NRMSE) defined as:

$$\text{NRMSE} = \frac{1}{\sigma_y} \sqrt{\sum_{j=1}^{K} [f(\mathbf{x}_j) - y_j]^2} \tag{3}$$

where σ_y is the standard deviation of $f(\mathbf{x})$.

The first function we have worked on is:

$$g(x) = (x - 0.2)^2 cos[10(x - 0.3)^2] + 0.4x \tag{4}$$

200 samples uniformly distributed in $[0, 1]$ have been used for initialization and as training set, while the validation set is composed of 1000 uniformly distributed samples.

Table 1 reports some results obtained with different levels and Fig. 1 depicts the original function and the network output after initialization for the second

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 363-369

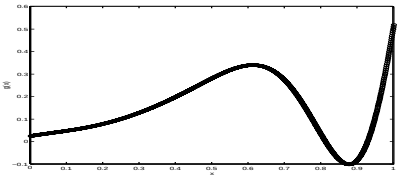


Figure 1: Function (4) (plain line) and its estimate (dots) after network initialization.

L	N_p	N_{TOT}	N	M	Initialization		After 1200s	After 3000s
					T_{in} (s)	NRMSE	NRMSE	NRMSE
4	1	1920	232	17	60	0.127	0.127	0.127
5	1	15872	961	21	378	$9 \cdot 10^{-2}$	$9 \cdot 10^{-2}$	$9 \cdot 10^{-2}$
5	5	15872	960	30	683	$6.33 \cdot 10^{-2}$	$6.33 \cdot 10^{-2}$	$6.33 \cdot 10^{-2}$
Random initialization				17	0.3	1.2	$9.26 \cdot 10^{-3}$	$6.5 \cdot 10^{-3}$
Random initialization				21	0.3	1.08	$9.9 \cdot 10^{-3}$	$7.71 \cdot 10^{-3}$
Random initialization				30	0.3	1.21	$1.12 \cdot 10^{-2}$	$6.03 \cdot 10^{-3}$

Table 2: Results of some experiments for approximating function (5); T_{in} is the time required for initialization. The network performances are compared after initialization and for two fixed values of CPU time elapsed.

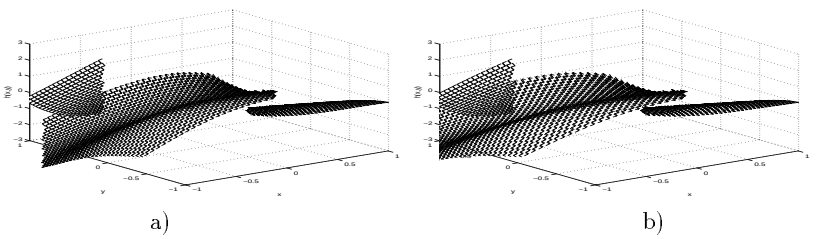
case of Tab. 1. As a comparison, Tab. 1 also presents the results obtained by training similar networks with randomly initialized weights. For the same CPU time elapsed, the performance of the networks initialized with the proposed method are better than those achieved with traditional methods in most cases.

The second function has the following expression:

$$h(x,y) = (x^2 - y^2)e^{\frac{x+y}{2}} + 2e^{-2[(x+.5)^2+(y+.5)^2]} \tag{5}$$

400 samples uniformly distributed in the square domain $[0,1] \times [0,1]$ have been used for initialization and as training set, while the validation set is composed of 2500 uniformly distributed samples.

Table 2 reports some results obtained with different levels and values of N_p and Fig. 2 depicts the original function and the network output after initialization for the third case of Tab. 2. As a comparison, Tab. 2 presents the results



a) b)

Figure 2: a) The function (5); b) its estimate after the network initialization.

obtained by training similar networks with randomly initialized weights. In the present case for randomly initialized networks the training is able to obtain an error lower with respect to the network initialized with the proposed method, which at once achieves low values of NRMSE but never receive (as in the previous case) a significant improvement from training.

4. Discussion and Conclusions

The paper proposes a novel method for initializing two-layer perceptrons, which has been derived from procedures currently applied to RBF networks. The results show that considerable saving of computation time is obtained without greatly affecting the network performances. Moreover the number of hidden neurons can be automatically chosen. Anyway, the proposed methodology still presents some drawbacks: it seems not completely exempt from problems deriving from local minima of the surface error, as well as random initialization, even if the latter absolutely needs training in order to obtain acceptable values of the NRMSE. The number of parameter vectors depends exponentially on the input dimension; therefore, when many inputs are treated, memory and computation power of the computer on which the procedure runs can heavily limit the maximum level which can be considered. Moreover, in the procedure of elimination of useless functions from the library, the number of sample points covered by the "non flat portion" of a single function cannot be determined in a very efficient way, as it happens in the case of RBF and WN. Future work concerns these aspects of the initialization and has a twofold aim: to increase the computation efficiency and to study if there exist other criteria of selecting a more limited but yet uniformly distributed set of weights vector.

References

[1] L.M. Reyneri, "Unification of Neural and Fuzzy Computing Paradigms", in *Proc. of AT-96, 1-st Int'l Symp. Neuro-Fuzzy Systems*, Lausanne, August 1996.

[2] M. Sgarbi, V. Colla, L.M. Reyneri: "A Comparison Between Weighted Radial Basis Functions and Wavelet Networks ," *Proc. European Symp. on Artificial Neural Networks ESANN'98*, pp. 13-19, Bruges, Belgium, April 22-24, 1998.

[3] Q. Zhang: "Using Wavelet Network in Nonparametric Estimation," *IEEE Trans. Neural Networks*, Vol. 8, No. 2, pp. 227–236, March 1997.

[4] S. Chen, C.F.N. Cowan, P.M. Grant: "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Trans. Neural Networks*, Vol. 2, No. 2, pp. 302–309, March 1991.

[5] M. Lehtokangas, J. Saarinen, K. Kaski, P. Huuhtanen: "Initializing Weights of a Multilayer Perceptron Network by Using the Orthogonal Least Squares Algorithm," *Neural Computation*, Vol. 7, pp. 982-999, 1995, Massachusetts Institute of Technology.

[6] H. Akaike: "Fitting autoregressive models for prediction," *Ann. Inst. Stat. Math.*, Vol. 21, pp. 243-347, 1969.