A NEW NEURAL NETWORK PRUNING METHOD BASED ON THE SINGULAR VALUE DECOMPOSITION AND THE WEIGHT INITIALISATION ([1])

S. Abid* : ***IEEE Student Member*** **- F. Fnaiech* :** ***Senior Member IEEE*** **- M.Najim** :** ***Fellow IEEE***
***Laboratoire CEREP (CEntre de REcherche en Productique)**
E.S.S.T.T. 5 Av. Taha Hussein 1008 Tunis - TUNISIA.
Tel (216) 1 503 837 - Fax (216) 1 391 166, Email : Sab.Abid@esstt.rnu.tn; Farhat.Fnaiech@esstt.rnu.tn
****Equipe Signal & Image: Avenue du Dr. Albert Schweitzer - Domaine Universitaire. BP 99.**
33402 Talence Cedex -FRANCE, Email : najim@goelette.tsi.u-bordeaux.fr

Abstract
In this paper, we present an efficient procedure to determine the optimal hidden unit number of a feed-forward multi-layer Neural Network (NN) using the singular value decomposition (SVD) taking into account the function to be approximated by the NN and the initial values of the updating weights. The SVD is used to identify and eliminate redundant hidden nodes. Minimizing redundancy gives smaller networks, producing models that generalize better and thus eliminate the need of using cross-validation to avoid overfitting. Using this procedure we obtain a final model with fewer adjustable parameters and more accurate predictions than a network model with a fixed, a priori determined, size. We show these performances by applying this procedure to several problems such as function approximation and image recognition.

Keywords : multi-layer Neural Network (NN); Singular Value Decomposition (SVD); redundant neuron; NN size;

1. INTRODUCTION
The problem of determining the proper size of a multi-layer NN is recognized to be crucial, especially for its practical implications in such important issues as learning and generalization [5][6][7]. In general, it is desirable to have small NN size. This is true because increasing the number of hidden units in a multi-layer NN may improve its approximation quality with respect to its training patterns, but not always improves its generalization behavior to new patterns. An improperly chosen configuration may result in either overfitting of the training patterns or nonconvergence in learning. One way to improve the generalization behavior of a NN is to reduce its number of hidden units when convergence is reached. In addition, small NN are faster when deployed. In the literature many methods are used to optimize the NN structure. These methods include destructive, constructive, genetic-algorithm [8].

Destructive or pruning methods start from a fairly large network and dynamically remove unimportant connections or units, whereas constructive or growth methods start from a small network and dynamically grow the network. Since the latter usually require fewer computations, extensive research has been carried out in this area.

Another class of dynamic multi-layer NN is block-feedback NN that can be learned incrementally [11].

Moreover the majority of the researchers always ignore the problem of the dependence, of the optimal NN size, on the weight initialization. In this work and in [3], we demonstrate that the optimal NN size is highly dependent on the weight initialization. This problem is treated in section 3.

In this work, we propose a SVD pruning algorithm for network training and size selection that, starting from an arbitrary NN size, automatically determines the optimal number of hidden nodes, given the available data. Gradient descent approach is used to determine the internal weights of the net and the SVD is used to identify redundant hidden nodes. These nodes are eliminated and the reduced size network is retrained until convergence. The SVD procedure has the virtue of being both simple and rigorous.

The use of the SVD for optimizing the NN structure was proposed in [4]. In their paper [4], the authors consider a linear output multi-layer NN. This work is a development of the SVD algorithm to a multilayer NN with sigmoidal nonlinear neurons. Moreover we demonstrate that the optimal size of a multilayer NN depends on the initial choice of the synaptic coefficients.

This paper is organized as follows. In section 2 we present the SVD pruning algorithm, where in section 3, we discuss the problem of the initial weight NN size dependence. In section 4 we

([1]) This paper has been supported partially by a French/Tunisian project CMCU #00F1125

present some simulation results and comparisons between pruned networks and networks trained with the standard back-propagation algorithm. Finally in section 5, we give the main conclusions of the paper.

2. SVD PRUNING ALGORITHM

For simplicity purposes we shall suppose that the NN contains one output layer with n_2 neurons, one hidden layer with n_1 neurons and n_0 input units. However, without loss of generality, the pruning algorithm can be applied to multi-hidden layers.
For the hidden layer we have:

$$ue_k = \sum_{i=0}^{n_0} v_{ki} x_i = V_k^T X \quad \text{and} \quad y_k = f(ue_k) \qquad (1)$$

where $k = 1, \cdots, n_1$ and X is the input vector given by $X = [x_0 \ x_1 \ \cdots \ x_{n0}]^T$
For the output layer:

$$us_k = \sum_{i=0}^{n_1} w_{ki} y_i = W_k^T Y \quad \text{and} \quad z_k = f(us_k) \qquad (2)$$

where $k = 1, \cdots, n_2$ and Y is the input vector to the output layer given by $Y = [y_0 \ y_1 \ \cdots \ y_{n1}]^T$.
x_0 and y_0 are the bias terms and v_{ki}, w_{ki} are the linkweights to the corresponding k^{th} neuron. f is a nonlinearity assumed to be a sigmoidal function.
We define the weight matrixes for the hidden and the output layers as :

$$V = [V_1, V_2, \cdots, V_{n1}] \text{ where}$$
$$Vi = [vi0, vi1, \cdots, vin0]^T \qquad (3)$$
$$W = [W_1, W_2, \cdots, W_{n2}] \text{ where}$$
$$Wi = [wi0, wi1, \cdots, win1]^T \qquad (4)$$

The SVD alternates between estimation of the weights from input to hidden nodes V and the weights from hidden to output nodes W in a two stage process as follows: 1) given a set of values for the W coefficients, the weights V are adjusted using a nonlinear optimization method to minimize the sum of squared errors. During this stage the coefficients W remain fixed. 2) given the new values for the weights from input to hidden nodes, the outputs of the hidden nodes are calculated. These outputs, which are nonlinear transformations of the inputs, are used to calculate a new set of coefficients W using linear least squares by inverting f in (2). Use of least squares techniques, more specifically SVD, allows well-developed linear methods to be applied to the training and selection of appropriate size of NN. We use SVD as an explicit model selection method (i.e., to identify and remove redundant hidden nodes during network training) and not a priori. The advantage of such an approach is that it adapts the model structure (number of hidden units) to the available data and avoids the need for potentially expensive cross-validation procedures, thus allows all available data to be used for training.

2.1. Updating the hidden weights

The steepest descent method is used to adjust the hidden weights V. For a detailed discussion on the intuition and theoretical derivation of the method see [1][2].
In our formulation, the objective function is the sum of squared errors:

$$E = \frac{1}{2} \sum_{p=1}^{N} \sum_{k=1}^{n_2} (z_k - dz_k)^2 \qquad (5)$$

where the sum is over all N training examples and n_2 outputs; dz_k represents the desired NN response for the k^{th} output neuron. Iterations continue until the change in the objective function between iterations is below a prespecified threshold or the gradient is exactly zero, which implies that the optimum has been reached.

2.2. Singular Value Decomposition.

Once the hidden weights V are Known, the output of each hidden node for each input pattern p can be calculated. This gives an $N \times n_1$ matrix. Then, we can formulate the following linear least squares problem: Calculate W_j such that the residual

$$E_j = \left\| AW_j - ldz \right\|_2 \qquad (6)$$

is minimized, where A is the $N \times n_1$ matrix having as columns the outputs of each hidden node for all N input vector, and ldz is the linear desired output vector defined as:

$$ldz = f^{-1}(dz) \qquad (7)$$

This linear least squares formulation can be obtained for all n_2 responses.
The basic premise behind SVD is that a general $N \times n_1$ matrix can be decomposed in three matrices R, S, and T as

$$A = R \, . \, S \, . \, T^T \qquad (8)$$

where R is an $N \times N$ column-orthogonal matrix, S is an $N \times n_1$ diagonal matrix with positive or zero elements and T is an $n_1 \times n_1$ orthogonal matrix. The elements of the S matrix are called the singular values of the A matrix. The matrix A may be singular (or nearly singular) because of row or column degeneracies. Column degeneracies imply that the hidden node outputs are correlated. This means that the problem is overdetermined with the given set of hidden nodes; i. e., the available data cannot help distinguish between them. In practice, it corresponds the presence of redundant hidden nodes.

This observation motivates the use of SVD as the least squares solution technique in our network training method. The development of a model network involves choice of a priori fixed number (typically large) of hidden nodes, with the hope that they will be adequate to approximate the unknown underlying function. However, it may well be the case that this initial model is over-parameterized (redundant). Redundant hidden nodes cause singularities in the *A* matrix, which can be identified through inspection of its singular values. A nonzero number of small singular values indicates redundancy in the initial choice for the number of hidden nodes. The algorithm then eliminates these hidden nodes and retains the "pruned" network model.

3. DEPENDENCY OF THE OPTIMAL NN SIZE ON THE WEIGHT INITIALIZATION

We have shown in (6) that the construction of the *A* matrix depends on the initial weight selection. Then the SVD decomposition is highly dependent of the initial weights. For different initial weights the element of S matrix change and the number of small singular values change also then the redundant hidden nodes change in turn. This leads in the end of training to different optimal NN sizes. Which one of these networks is the better?

To avoid this problem we shall search the optimal range of the initial weights or the optimal standard deviation for a chosen function distribution of the initial weights. To do so we inspect the singular values of *A* for many initial weights, and then we select the optimal range of the initial weights that gives the maximum number of redundant units for removing.

4. APPLICATION EXAMPLES

The SVD pruning algorithm was applied on two variations of a test problem. We present in this section the numerical results of two problems. In each problem, we compare the performances of the pruned neural network (PNN) with respect to the network trained with the standard back propagation algorithm (SBPNN). These performances are the final hidden unit number with respect to the initial number used for the SBPNN, the iteration number needed for convergence for a priori chosen threshold, and the convergence time for each network.

4.1. Sine function approximation problem

In this application the NN has been used to approximate the sine function of the form

$$f(x) = \sin(x) \,;\, 0 < x < 4p \qquad (9)$$

The networks contain one input unit and one output unit. For the SBPNN we have fixed the hidden unit number n_1 at 15. This value is chosen after 50 learning trials to maximize the learning speed and finding a good initial weight sets. By application of the SVD algorithm and the research of an optimal range for the initial weight coefficients we can reduce the hidden unit number n_1 to 11. Table 1 resumes the comparison results of this problem.

It is clear that the SVD algorithm removes successfully the redundent neurons and improves the performances of the neural network.

4.2. Pattern recognition problem

The second experiment we have considered is the training of a NN to recognize patterns presented to its input given in Figure 1. Although many different experiments were performed with various data sets

	n_1	Convergence iteration number	Convergence time (s)
SBPNN	38	632	34.25
PNN	29	479	29.36

Table 2: Comparison results of the performances of the PNN with respect to the SBPNN for the pattern recognition problem

	n_1	Convergence iteration number	Convergence time (s)
SBPNN	15	2238	12.13
PNN	11	1776	10.88

Table 1: Comparison results of the performances of the PNN with respect to the SBPNN for the sine function approximation problem

and different network structures, we present here only a limited number because of space considerations.
The input pixels are set to a level of -0.5 or +0.5. It is important to note that these levels are considered to be analog values rather than digital binary values. Although, we present here experiments with patterns of two levels, similar results are obtained with patterns of various gray levels.
The output is likewise treated as analog signal.. Therefore, the network is considered trained only when the output agrees in sign with the desired output but also in absolute value.
The networks contain 49 inputs, 4 output units. For the SBPNN, we have fixed n_1 at 38 in the same manner that in the first application. Table 2 shows the comparison results.

5. CONCLUSION

In this paper we have proposed an efficient procedure to determine the optimal hidden unit number of a feed-forward multilayer NN. This procedure is based on the SVD and takes into account the function to be approximated and the initial values of the updating weights. We have shown that the optimal NN size is highly dependent on the initial synaptic coefficient set. The new proposed algorithm requires less number of neurons and less number of iterations than the standard back propagation algorithm for a suitable choice of the initial weights.

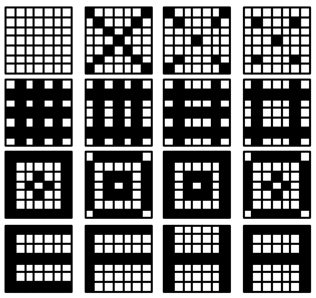
Figure 1: 7x7 patterns used for training

6. REFERENCES

[1] S. Abid ; F. Fnaiech and M. Najim, "A fast Feed-Forward Training Algorithm Using A Modified Form Of The Standard Back-Propagation Algorithm" *IEEE Transactions on Neural Network*. pp. 424-430, March 2001.

[2] S. Abid F. Fnaiech and M. Najim, "Evaluation Of The Feedforward Neural Network Covariance Matrix Error" *IEEE International Conference on Acoustic Speech and Signal Processing* (ICASSP'2000) : June 5-9, Istanbul, Turkey.

[3] F.Fnaiech, N. Fnaiech and M. Najim "A new feedforward neural network hidden layer's neurons pruning algorithm" *IEEE International Conference on Acoustic Speech and Signal Processing* (ICASSP'2001) : Salt-lake-city, USA.

[4] D. C. Psichogios and L. H. Ungar, "SVD-NET: An algorithm that automatically selects network structures" *IEEE Trans. on Neural Networks*, Vol. 5, No. 3, pp. 513-551, May 1994.

[5]N. Murata, S. Yoshizawa and S. I. Amari, "Network information criterion determining the number of hidden units for an artificial neural network model" *IEEE Trans.* on *Neural Networks*, Vol. 5, No. 6, pp. 865-871, Nov. 1994.

[6] C. C. Teng and B. W. Wah, "Automated learning for reducing the configuration of a feedforward neural network" *IEEE Trans.* on *Neural Networks*, Vol. 7, No. 5, pp. 1072-1085, Sep. 1996.

[7] G. Castellano, A. M. Fanelli and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks" *IEEE Trans.* on *Neural Networks*, Vol. 8, No. 3, pp. 519-531, May 1997.

[8] R. Setino and L. C. K. Hui, "Use of quasi-Newton method in a feedforward neural network construction algorithm" *IEEE Trans.* on *Neural Networks*, Vol. 6, No. 1,pp. 273-277, Jan. 1995.

[9] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *advances in Neural Information Processing systems* 5, S. Hanson, J. Cowan, and C. Giles, Eds. San Mateo, CA: Morgan Kaufmann, 1993.

[10] M. C. Moze and P. Smolensky, " Using relevance to reduce network size automatically,' *Connect. Sci.*, vol. 1, No. 1, pp. 3-16, 1989.

[11] S. Santini, A. D. Bimbo and R. Jain, "Block structured recureent neural networks," *IEEE Trans.* on *Neural Networks*, Vol. 8, No. 1, pp. 135-147, 1995