

Efficient Pruning of Multilayer Perceptrons Using a Fuzzy Sigmoid Activation Function

Emilio Soria-Olivas^a, José D. Martín-Guerrero^a, Antonio J. Serrano-López^a,
Javier Calpe-Maravilla^a, Joan Vila-Francés^a and Gustavo Camps-Valls^a

^aDept. Enginyeria Electrònica. Universitat de València.
C/ Dr. Moliner, 50. 46100 Burjassot, València, Spain.

This paper presents a simple and powerful pruning method for multilayer feedforward neural networks based on the fuzzy sigmoid function presented in [1]. Successful performance is obtained in standard function approximation and channel equalization problems.

Key words: Pruning, fuzzy sigmoid, channel equalization, classification.

1. Introduction

The multilayer perceptron is one of the most popular artificial neural networks (ANNs) because of the universal approximation theorem [2]. This network is composed of a series of elements, neurons, arranged in layers and interconnected through synaptic weights, which are usually adjusted on the basis of iterative algorithms. Despite the good performance obtained in many fields, it is usually necessary to introduce some methods, such as early-stopping criteria, regularization, or pruning techniques, in order to control the capacity of the model and to prevent overfitting [3]. In this work, we focus on pruning, which basically consists in removing neurons or connections that are not significant in the functional mapping performed by the network [4]. Pruning is an important technique in the design of an ANN for many reasons: (1) a network with a high number of neurons needs a high number of training examples to learn a given problem, (2) the learning time increases significantly as a function of the number of nodes in the network, (3) the generalization capabilities decrease when the complexity of the network increases, and (4) low-sized networks are preferable for hardware implementation. Several pruning algorithms exist in the literature, such as Optimal Brain Damage, Optimal Brain Surgeon, Early Brain Damage, or Karnin's method [4]. However, the main problem with these algorithms is the high computational burden induced.

In this paper, we assess the performance of the activation function proposed in [1] in the context of multilayer perceptron pruning. In [1], a fuzzy sigmoid function was presented in the context of ANNs for regression and pattern recognition problems. The function offers low computational burden, high convergence speed, suitable hardware implementation, and straightforward interpretation of the results obtained by using the IF-THEN rules embedded in the ANN. Its use has also been shown to act as an efficient sigmoid kernel in the context of Support Vector Machines (SVM) [5], allowing more stable solutions and

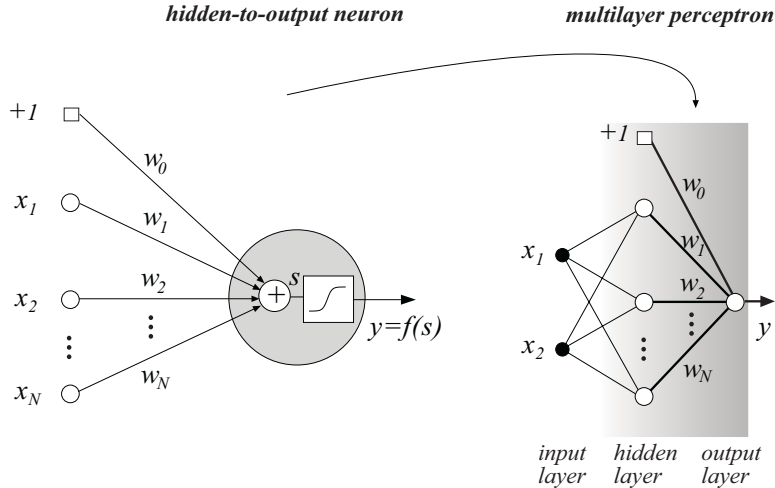


Figure 1. Example of a neuron with N inputs acting as the output node in the network (left), and a multilayer perceptron formed by two inputs, a hidden layer and an output layer (right).

lower computational efforts than classical methods.

The remainder of the paper is outlined as follows. Section 2 presents the theoretical development and learning rule of the proposed algorithm. Section 3 includes the experiments, and Section 4 presents some conclusions and a proposal for further work.

2. Algorithm development

A neuron in the classical multilayer perceptron (MLP) performs a weighted sum on its input signals x_i and a bias term, and passes the result through a nonlinear activation function $f(s)$, which is commonly a sigmoid-shaped function (see Fig. 1). In this paper, nevertheless, we take advantage of the fuzzy sigmoid activation function (*fuzzy tanh*) presented in [1] for pruning the network. The fuzzy tanh activation function has the form:

$$f(s) = \begin{cases} \text{sign}(s), & |s| \geq L \\ -\frac{s \cdot |s|}{L^2} + \frac{2s}{L}, & \text{otherwise} \end{cases} \quad (1)$$

where, for $L = 1$, a good approximation to the common hyperbolic tangent (sigmoid) activation function, $f(s) = \tanh(s)$, is obtained. When an ANN is trained using this activation function, the *sign* function in (1) forces the appearance of a high amount of saturated hidden neurons, i.e. many outputs in the network become ± 1 . This, in turn, induces high correlation coefficients among hidden outputs and, consequently, pruning appears naturally. In this way, weights corresponding to highly correlated hidden outputs can be removed from the network on the basis of a correlation coefficient threshold, ρ .

We will consider a network with only one output neuron in order to simplify the calculations. Let us focus on the processing carried out by the output neuron in the network

(see Fig. 1), which is given by

$$y = f(w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_N \cdot x_N + w_0) \quad (2)$$

where N is the number of hidden neurons (in Fig. 1, $N = 3$), x_i represents the output of the i -th hidden neuron, and w_0 is the bias of the hidden layer. If the absolute value of the correlation coefficient $r_{i,j}$ between the outputs of hidden neurons i and j is higher than a fixed threshold ρ , then

$$x_i \simeq x_j \cdot \text{sign}(r_{i,j}) \quad (3)$$

Correlation coefficients are estimated every certain number of epochs; in particular, we calculated the correlation coefficients every 20 epochs in our experiments. Moreover, since (3) holds for every training pattern,

$$y \simeq f(w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + x_i \cdot (w_i + w_j \cdot \text{sign}(r_{i,j})) + \dots + w_0) \quad (4)$$

Therefore, the j -th hidden neuron can be removed, being the weight of the i -th hidden neuron updated, as follows:

$$w_i \simeq w_i + w_j \cdot \text{sign}(r_{i,j}) \quad (5)$$

This procedure is intrinsically simple, since it relies only on a tunable threshold parameter ρ and, thus, both low computational cost and flexible prunings are obtained.

3. Experiments

This section analyzes the performance of the proposed algorithm in terms of accuracy, robustness, computational cost, and convergence rate in standard function approximation and channel equalization problems.

3.1. Function approximation

In this section, we assess the performance of our method in the regression problems proposed in [6]: $f_1(\mathbf{x}) = x_1 + x_2$ and $f_2(\mathbf{x}) = x_1 \cdot x_2$, where $\mathbf{x} \in \mathbb{R}^2$ and $x_1, x_2 \in [0, 1]$. These problems are well-suited to test pruning procedures because the input-output relationships are very simple, and thus the number of hidden nodes should be very low.

The same training methodology was followed in both problems. We built the training sets by generating 100 samples of $\{x_1, x_2\}$ drawn from a uniform distribution in the range $x_1, x_2 \in [0, 1]$, while the validation set was formed by 1000 samples following the same distribution. We varied the number of hidden neurons between 4 and 9. In all cases, training of the networks was performed during 1000 epochs, and the best results were obtained by fixing $\rho = 0.9$. The convergence criterion for the training set consisted of analyzing whether the Mean-Squared Error (MSE) during the last 20 epochs was lower than 10% of the variance of the variable to be modeled ($f_1(\mathbf{x})$ or $f_2(\mathbf{x})$). The convergence criterion for the validation set was the same, but in this case we analyzed whether the MSE committed by the trained network in the validation set was lower than 10% of the variance of the variable to be modeled. These experiments were performed 50 times with different random weights initialization in order to avoid skewed conclusions, and average results are presented.

Table 1 shows the results achieved in both problems. It shows the rate of networks assuring convergence both in the training set and the validation set, and also the rate of pruned synaptic weights in the network as a function of the number of hidden nodes. It is important to note that the proposed methodology not only allows a great reduction in the computational burden (on average, half of the synaptic weights of the hidden nodes are pruned), but also the capabilities of the network to solve the problem are not reduced (similar convergence rates are achieved both with and without pruning).

3.2. Channel equalization

Transmission through a communication channel is affected by different kinds of interferences, such as power degradation and fades, multi-path time dispersions, and background thermal noise [7]. In this paper, we consider the general channel equalization scheme illustrated in Figure 2. The input to the channel is assumed to be a sequence, $\{y_i\}$, of independent symbols extracted from a specific alphabet. The channel output is corrupted by random noise, $\{n_i\}$, which is considered to be an additive white Gaussian process. The task of the equalizer is to recover the input sequence, $\{y_i\}$, from the received sequence $\{r_i\}$. A binary phase shift keying baseband communication system was considered in our experiments, in which data are encoded and transmitted as ± 1 .

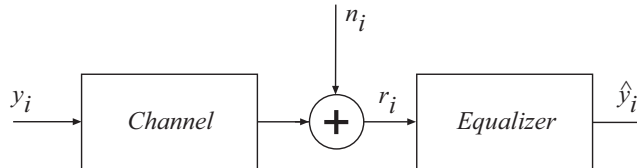


Figure 2. Channel equalization scheme used.

The experiment consisted of equalizing a standard channel given by the transfer function $H(z) = 0.5 + z^{-1}$ [8]. A set of 500 randomly generated samples were transmitted and networks were trained during 1000 epochs. We compared the performance of pruned networks with our algorithm versus unpruned networks using the standard hyperbolic tangent. In order to measure the bit error rate (BER), an independent test burst of

Table 1

Convergence rate (CR) [%] and Rate of Pruned Synaptic Weights (RPSW) [%] as a function of the number of hidden neurons. 'PR' indicates results using pruning while 'NPR' denotes results without pruning. Also, 'TS' and 'VS' indicate Training Set and Validation Set, respectively.

		Number of hidden neurons											
		4		5		6		7		8		9	
		PR	NPR	PR	NPR	PR	NPR	PR	NPR	PR	NPR	PR	NPR
$f_1(x)$	CR	TS	96	100	98	100	100	100	100	100	100	100	100
		VS	96	100	98	100	100	100	96	100	100	100	98
	RPSW		48.5		50.8		50.6		58.8		61.0		64.2
$f_2(x)$	CR	TS	100	98	100	98	100	98	100	100	100	98	98
		VS	98	100	98	100	98	100	100	100	98	100	98
	RPSW		48.0		49.2		53.0		51.1		56.0		60.2

Table 2

Rate of pruned synaptic weights [%] as a function of signal-to-noise ratio (SNR) and the initial number of hidden neurons.

Initial number of hidden neurons	SNR [dB]					
	5	8	11	14	17	20
4	52.5	30.0	15.0	17.5	17.5	15.0
5	32.0	38.0	30.0	14.0	28.0	18.0
6	46.6	33.3	28.3	30.0	26.6	36.6
7	48.5	42.8	42.8	38.5	38.5	37.1
8	50.0	47.5	37.5	42.5	40.0	40.0

50,000 samples was used. The experiment was repeated using different Signal-to-Noise Ratio (SNR) from 5 to 20 dB (3 dB steps). Figure 3 shows BER versus SNR for both methods. It is worth noting that our approach follows a very similar trend using a much simpler model.

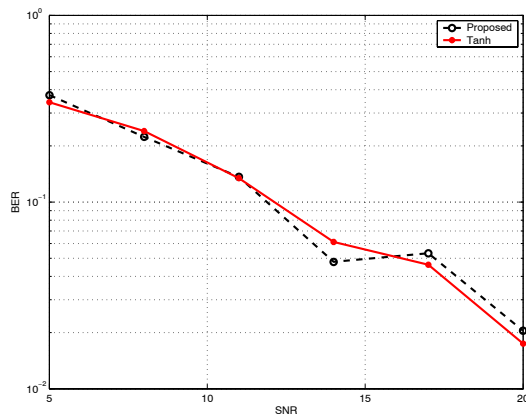


Figure 3. Bit error rate (BER) vs. signal-to-noise ratio (SNR) for the best network using the hyperbolic tangent activation function (solid line) and the proposed fuzzy sigmoid activation function for pruning (dashed line). Results are shown for the validation set.

Table 2 shows the rate of pruned synaptic weights [%] as a function of SNR and the initial number of hidden neurons. It is worth stressing that the proposed method considerably reduces model complexity. Similar results were observed with other channels, noise situations, and neural architectures. It should be pointed out that the relationship between the rate of pruned synaptic weights and the number of hidden neurons is different depending on the value of the SNR. In particular, results achieved in low SNR conditions are not significant since they correspond to very noisy situations, in which it is difficult to carry out any kind of pruning.

4. Conclusions

This paper presented a simple and powerful pruning method for multilayer feedforward neural networks. Experimental results in function approximation and channel equalization problems have demonstrated the validity of the approach in terms of both speed of convergence and reduction in the structural complexity of the pruned network. Future work will consider introducing adaptive threshold tuning in the pruning algorithm.

REFERENCES

1. E. Soria-Olivas, J. Martín-Guerrero, G. Camps-Valls, A. Serrano-López, J. Calpe-Maravilla, L. Gómez-Chova, A low-complexity fuzzy activation function for artificial neural networks, *IEEE Transactions on Neural Networks* 14 (6) (2003) 1379–1380.
2. K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* 2 (1989) 359–366.
3. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, New Jersey, 1999.
4. R. Reed, Pruning algorithms - a survey, *IEEE Transactions on Neural Networks* 2 (5) (1993) 740–747.
5. G. Camps-Valls, J. Martín-Guerrero, J. L. Rojo-Álvarez, E. Soria-Olivas, Fuzzy sigmoid kernel for support vector classifiers, *Neurocomputing Journal*. Accepted for publication, in press.
6. R. Sexton, R. Dorsey, N. Sikander, Simultaneous optimization network function and architecture algorithm, *Decision Support Systems* 36 (2004) 283–296.
7. S. Qureshi, Adaptive Equalisation. *Proc. IEEE*, Vol. 73, 1985.
8. G. J. Gibson, S. Siu, C. F. N. Cowan, The application of nonlinear structures to the reconstruction of binary signals, *IEEE Trans. Signal Proc.* 39 (8) (1991) 1877–1884.