Australian Journal of Basic and Applied Sciences, 7(8): 214-223, 2013
ISSN 1991-8178

Pseudo-Entropy Based Pruning Algorithm for Feed forward Neural Networks

Slim Abid, Mohamed Chtourou, Mohamed Djemel

[1]Control & Energy Management Lab (CEM LAB), National School of Engineering of Sfax, University of Sfax, Tunisia

Abstract: Design of artificial neural networks is an important and practical task:"how to choose the adequate size of neural architecture for a given application". One popular method to overcome this problem is to start with an oversized structure and then prune it to obtain simpler network with a good generalization performance. This paper presents a pruning algorithm based on pseudo-entropy of hidden neurons. The pruning is performed by iteratively training of network to a certain performance criterion and then removing the hidden neuron with individual pseudo-entropy greater than a preselected threshold which is slightly higher than the average value of network's pseudo-entropy until no one can further be removed. This approach is validated with an academic example and it is tested on induction motor modeling problem. Compared with two modified versions of Optimal Brain Surgeon (OBS) algorithm, the developed method gives interesting results with an easier computation tasks.

Key words: pruning, pseudo-entropy, neural networks.

INTRODUCTION

In recent years, many neural network models (J. M. Zurada, 1992; P. Mehra and B. W. Wah, 1992) have been elaborated for several applications (I. Rivals, and L. Personnaz, 2003) such as: function approximation, pattern classification (A. Sinha and D. N.Vishwakarma, 2013) and intelligent control (A. Mechernene *et al*., 2011; N. Zitouni *et al*., 2011). One of the most challenging issues in neural network applications is the choice of the adequate architecture for a given problem. In fact, too small network cannot learn the problem well and too large neural structure can lead to over-fitting and shows a poor generalization performance. Hence, to overcome this dilemma many topology optimization approaches have been introduced. They can be divided into three parts: the size of input, the number of hidden layers and the number of neurons in each hidden layer and the dimension of output. The first and the third ones are relatively easy to be selected when a priori domain knowledge is available but the second one is the most complicated choice. To overcome this problem, researchers have developed various methods which can be grouped into two categories: constructive and pruning approaches. The first strategy starts with an undersized structure and incrementally adds new hidden neurons and/or layers (D. Liu *et al*., 2002; L. Ma and K. Khorasani, 2004). The second one begins with an oversized architecture and eliminates the unnecessary weights and/or hidden neurons (R. Reed, 1993; M. Hagiwara, 1993). Recent interest has been growing on pruning approaches (H. Honggui and Q. Junfei, 2009; D. Juan and E. M. Joo, 2010; Z. Zhang, and J. Qiao, 2010). The most known methods are Optimal Brain Damage (OBD) (Y. Le Cun *et al*., 1990) and Optimal Brain Surgeon (OBS) (B. Hassibi *et al*., 1993) which eliminate weight with smallest saliency by estimating the second derivative of the error function with respect to the weights. In this paper, a pruning algorithm based on pseudo-entropy of hidden neurons is developed and compared with too existing ones derived from the OBS approach. The rest of paper is organized as follows. Section 2 presents brief description of two modified versions of OBS algorithm. A proposed pruning algorithm is introduced in section 3. Section 4 is devoted to the test of the proposed and its comparison with respect the two version of OBS algorithm. Finally, the conclusion is presented in section 5.

Related Works:

A Feedforward Neural Networks $(FNN)_s$ with a single output is represented in Figure 1.

This neural network is parameterized in terms of its weights, where:

$$w = [w_1, w_2, \dots, w_m]^T \in \Re^m \quad (1)$$

The training data consists of *N* patterns $\{x^j, y^j\}$, j=1,2,…,*N*.

The error function for a given pattern is defined as:

$$J_p = \frac{1}{2}(y - y^d)^2 \quad (2)$$

Corresponding Author: Slim Abid, Control & Energy Management Lab (CEM LAB), National School of Engineering of Sfax, University of Sfax, Tunisia
E-mail: abid_slim_enis@yahoo.fr

The global error function is described as:

$$J = \frac{1}{N}\sum_{p=1}^{N} J_p \tag{3}$$

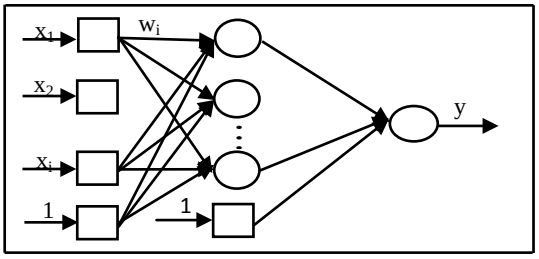

Fig. 1: *Feedforward Neural Network.*

1. OBS approach:

(B. Hassibi and D. G. Stork) proposed the (OBS) algorithm which finds and eliminates the weight that has the smallest relevance on the neural network performance and adjusts the remaining weights according to the error function gradient. The OBS algorithm assumes that the network has been trained to a local minimum of the error, so the second-order Taylor expansion of the error function with respect to the weights can be expressed as:

$$\delta J = \left(\frac{\partial J}{\partial w}\right)^T . \delta w + \frac{1}{2}\delta w^T . H . \delta w + O(\|\delta w\|^3) \tag{4}$$

Where H denotes the Hessian matrix composed by the coefficients of the second order derivatives of the error function with respect to weights.

In order to minimize the error given by (3), OBS algorithm deletes the particular weight w_q which satisfies the following equation:

$$e_q^T . \delta w + w_q = 0 \tag{5}$$

where e_q represents the unit vector corresponding to weight w_q ($w^T e_q = w_q$).

The corresponding expression for the minimum error is modified caused by changing a given weight depicted as (B. Hassibi *et al.*, 1993):

$$\underbrace{min}_{q}\left\{min\left(\frac{1}{2}\delta w^T . H . \delta w\right) / e_q^T . \delta w + w_q = 0\right\} \tag{6}$$

So to resolve this constrained optimization problem, we introduce Lagrange's method which leads to find the corresponding optimum value of Lagrange function L after deleting w_q as:

$$L_q = \frac{1}{2}\frac{w_q^2}{[H^{-1}]_{qq}} \tag{7}$$

The remaining weights are updated according the following equation:

$$\delta w = -\frac{w_q}{[H^{-1}]_{qq}} H^{-1} e_q \tag{8}$$

with $[H^{-1}]_{qq}$ denotes the diagonal element (q,q) of the inverse of the Hessian matrix H^{-1}.

The OBS algorithm needs inverse Hessian matrix to update the weights which is the most disadvantage of this algorithm. Hassibi and Stork [16] proposed a procedure to calculate H^{-1} based on a recursive method as:

$$H_{n+1}^{-1} = H_n^{-1} + \frac{H_n^{-1} X^{n+1} \left(X^{n+1}\right)^T H_n^{-1}}{p + (X^{n+1})^T H_n^{-1} X^{n+1}} \tag{9}$$

with $\left(H_0 = \alpha I_n, H = H_p \text{ and } 10^{-8} \le \alpha \le 10^{-4}\right)$.

However, one of the main disadvantages of OBS approach is that it requires a great amount of computation and a huge time for pruning procedure.

2. Unit-OBS algorithm:

(A. Stahlberger and M. Riedmiller, 1997) developed the Unit-OBS pruning algorithm which eliminates the needless neuron in one step with minimal increase in error. This method reduces both the computation complexity and the running time. The details of the Unit-OBS algorithm are as follows:

Step 1: Train the neural network to minimum error based on some local optimization method.

Step 2: Compute H^{-1}.

Step 3: For each neuron *u*:

- Compute the number of output weights for the neuron u and record it as *m(u)*.
- Form the unit matrix M for neuron u which:

$$M = \left[e_{q1}, e_{q2}, \ldots, e_{qm\,(u)}\right] \tag{10}$$

where $q_1, q_2, \ldots, q_{m(u)}$ denote the sequence number of each output weight with:

$$e_{qi}^T = \left[0, \ldots, \underbrace{1}_{i}, 0, \ldots 0\right] \tag{11}$$

- Calculate the error deviation after deleting neuron *u*:

$$\Delta J(u) = \frac{1}{2} w^T M (M^T H^{-1} M)^{-1} M^T w \tag{12}$$

Step 4: Find the neuron u_0 that gives the smallest increase in error noted $\Delta J(u_0)$: if $(\Delta J(u_0) > E_t)$ then the algorithm will stop; otherwise go to *step 5*.

E_t indicates a preselected threshold.

Step 5: Remove neuron u_0 and update the other weights using the following equation:

$$\Delta w = -H^{-1} M (M^T H^{-1} M)^{-1} M^T w \tag{13}$$

The unavoidable drawback for this algorithm is the complexity of computing the Hessian matrix whose dimension is equal to the number of initial weights of the network. In order to overcome this disadvantage Jun-fei *et al*. have proposed a modified version of Unit-OBS algorithm known as First-Unit-OBS (J.-F. Qiao *et al*., 2008).

3. Fast-Unit-OBS algorithm:

In this case, the size of the Hessian matrix depends on the number of hidden neurons. The relevance of each hidden neuron is described by the following equation:

$$L_u = \frac{1}{2} \frac{\bar{w}_u^2}{[H^{-1}]_{uu}} \tag{14}$$

where the weight mean is given by:

$$\bar{w}_u = \frac{1}{s} \sum_{i=1}^{s} w_{ui} \tag{15}$$

where w_{ui} denotes the output weights for the hidden neuron *u* and *s* indicates the number of w_{ui}.

The Fast-Unit-OBS-algorithm is given by the following steps:

Step 1: Train the neural network to minimum error based on some local optimization method.

Step 2: Compute H^{-1}.

Step 3: For each hidden neuron u; compute the relevance L_u and find the neuron u_0 that gives the smallest value which will be noted *L*.

Step 4: If *L* is greater than the preselected threshold L_t, than go to *step 5*. Otherwise:

- delete the selected hidden neuron and adjust the other weights of the remaining neurons based on the following equation:

$$\Delta w = -\frac{\overline{w_u}}{[H^{-1}]_{uu}} H^{-1} e_u \tag{16}$$

$$\text{with: } e_u = \left[0, \ldots, \underbrace{1}_{u}, 0, \ldots 0\right] \tag{17}$$

- return to the s*tep 2*.

Step 5: Stop the pruning approach and it may be desirable to retrain the network at this stage.

Despite the improvements in algorithms Unit-OBS and Fast-Unit-OBS which have reduced the computation time of the pruning procedure, these methods still need the calculating of the Hessian matrix inverse. In this paper, a pruning algorithm based on pseudo-entropy of hidden neurons without using the inverse of the Hessian matrix is presented in the following section.

***Pruning Algorithm Based On Pseudo-Entropy Of Hidden Neurons*:**

1. Preliminary:

The concept of entropy was introduced by Shannon in information theory (C. Shannon and W. Weaver, 1948) to measure the disorder as:

$$Ent(x, C) = \sum_i f_i(x) log\big(f_i(x)\big) \tag{18}$$

where *Ent* is minimal when *x* is in one class and maximum if *x* is equidistributed in all classes of the partition *C*.

Several researches in the field of artificial intelligence have exploited the properties of the entropy concept. (R. Kamimura, 1993) developed a method of entropy minimization for competitive learning. Many improvements (R. Kamimura and T. Kamimura, 2000; R. Kamimura, 2002) have lead to present a new approach to modeling complex information processing in living systems.

On the other hand, the concept of entropy has been introduced to improving the generalization performances by information minimization (R. Kamimura *et al*., 1994). (K. H. Chan *et al*., 1999) proposed a new entropy learning approach for neural network based on an entropy penalty term allowing steering the direction of the hidden node's activation in the process of learning. Moreover, a lot of methods have been elaborated based on entropy minimization to increase the adequate variables selection and to obtain simple neural network architecture (R. Kamimura, 1995; R. Kamimura *et al*., 1995). In this context, (G. See Ng *et al*., 2003) presented an improved entropy learning method which uses relevance criteria for neural network pruning.

2. Proposed algorithm:

The basic idea of the algorithm proposed in this paper has been inspired from the Zhao's pruning approach (S. Zhao *et al*., 2010) which consists to eliminate the unnecessary weights using a Fast-OBS pruning algorithm based on the network's pseudo-entropy of weights. This algorithm presents a penalty term which can be used to control the weights distribution in the training phase: when the weights distribution is evenly, the penalty term is smaller, or else is larger. In this case, the objective cost function *E* for the three-layer feedforward networks with *n* input neurons, *k* hidden neurons and *h* output neurons, can be defined as follows:

$$E = J + G \tag{19}$$

where *J* is the error function given by (3) and *G* denotes the network's pseudo-entropy of weights defined as:

$$G = G^I + G^O \tag{20}$$

with:

$$G^I = -\sum_{i=1}^{k}\sum_{j=1}^{n} p_{ij}^I Logp_{ij}^I \tag{21}$$

$$G^O = -\sum_{l=1}^{h}\sum_{i=1}^{k} p_{li}^O Logp_{li}^O \tag{22}$$

p_{ij}^I and p_{li}^O are quasi-probabilities of the network input weights and network output weights, respectively:

$$p_{ij}^I = \frac{{w_{ij}^I}^2}{\sum_{i1=1}^{k}\sum_{j1=1}^{n} {w_{i1j1}^I}^2} \tag{23}$$

$$p_{li}^O = \frac{{w_{li}^O}^2}{\sum_{l1=1}^{h}\sum_{i1=1}^{k} {w_{l1i1}^O}^2} \tag{24}$$

Although this algorithm provides a simpler structure of the network and improves the speed of the pruning process, the complex calculation of the inverse of the Hessian matrix persists. In this paper, a pruning algorithm

based on pseudo-entropy of hidden neurons is developed. In the first step, the proposed algorithm consists to train the initial neural structure to obtaining the desired learning performance. In the second step, the contribution of each hidden neuron is estimated by calculating the relevance index which depends on hidden neuron's output amplitude and can be expressed as:

$$RI_u = \frac{1}{N}\sum_{p=1}^{N}\left(\frac{O_u^p}{\sum_{u=1}^{N_c} O_u^p}\right) \quad (25)$$

where O_u^p indicates the output of the uth hidden node for a given pattern and N_c denotes the number of hidden ones.

The network's pseudo-entropy of hidden neurons can be defined as:

$$Ent = -\sum_{u=1}^{N_c} RI_u Log(RI_u) \quad (26)$$

The individual pseudo-entropy of each hidden neuron *u* is introduced as:

$$Ent_u = -RI_u Log(RI_u) \quad (27)$$

Our goal is to minimize the network's pseudo-entropy to obtain simpler neural network architecture. The proposed strategy consists to remove each hidden neuron with individual pseudo-entropy greater than a preselected threshold which is slightly higher than the average value of network's pseudo-entropy.

It is to be noted that we are interested in a multi input-single output (MISO) model (the generalization to more outputs units is straightforward) and the weights update law is based on the back-propagation algorithm (D. E. Rumelhart *et al*., 1986) which is given by:

$$\delta w = -\varepsilon \frac{\partial J_p}{\partial w} \quad (28)$$

where ε denotes the learning rate.

The proposed pruning algorithm can be described as follows:

Step 1: Train the neural network based on the back-propagation algorithm to obtain a tolerated value of the training criterion noted J_t.

Step 2: Calculate network's pseudo-entropy of hidden neurons using equation (26).

Step 3: Compute individual pseudo-entropy using equation (27) and remove each hidden neuron u which satisfies the following test:

$$Ent_u > \lambda\, Ent^{av} \quad (29)$$

with:

- Ent^{av} indicates the average value of network's pseudo-entropy defined as

$$Ent^{av} = \frac{1}{N_c}\, Ent \quad (30)$$

- λ is a preselected constant slightly greater than 1.

If there is no neuron that satisfies the test (29), than go to *step 5*.

Step 4: Train the obtained structure with a reduced number of iterations noted *t* and return to the *step 2*.

Step 5: Stop the pruning approach and retrain the network at this stage.

It can be seen that the proposed algorithm is based on a pruning approach avoiding the computation the inverse of Hessian matrix which consists the main contribution of this strategy.

RESULTS AND DISCUSSIONS

In this section, the simulation results related to the proposed algorithm are presented in order to test its performance. In first time, the capacities of the elaborated algorithm are verified on an academic example. In second time, we use this algorithm for induction motor neural modeling. Through two simulation examples the proposed algorithm will be compared with the Unit-OBS and the Fast-Unit-OBS algorithms according to some performances criteria.

1. First example:

This example consists on using a given neural network to generate a given learning set. The studied algorithms will be applied to retrieve the initial neural structure having generated the learning set.

Figure 2 shows the neural network generating the learning set with an input vector composed by:{y(k-1), y(k-2), u(k-1), u(k-2)}, 3 hidden neurons and y(k) as the output where the activation function in the hidden neurons and the output neuron is the sigmoid one.

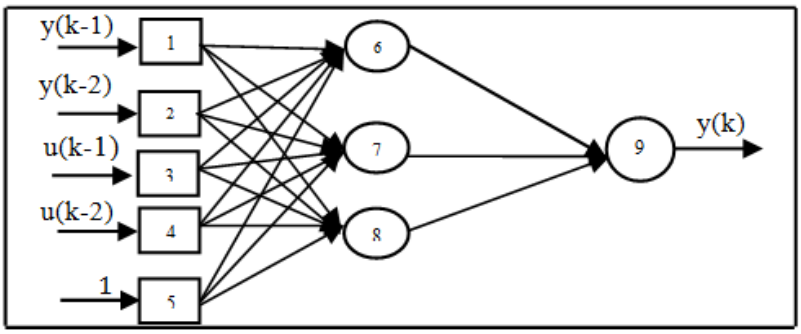

Fig. 2: *Neural network generating the learning set.*

Figure 3 and 4 represent the input signal used respectively for the training and validation sets.

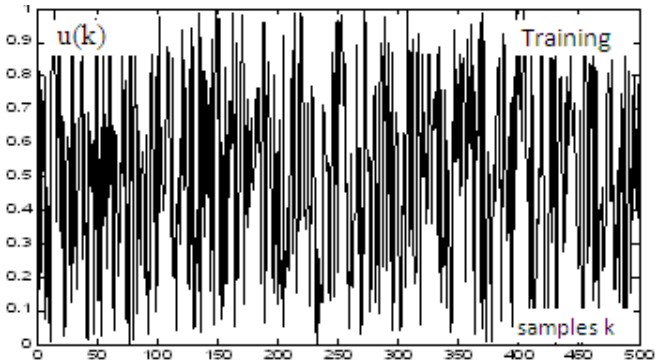

Fig. 3: *Training input signal*

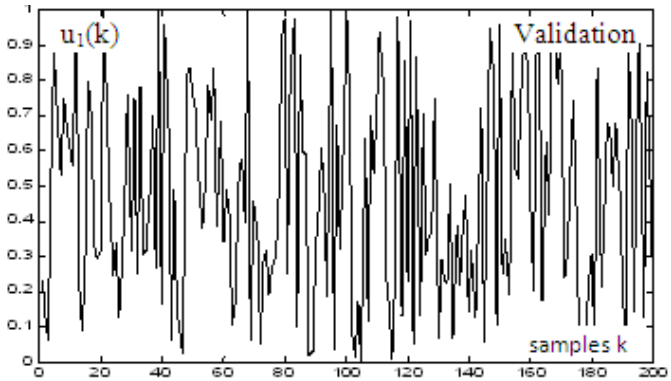

Fig. 4 *Validation input signal.*

The efficiency of each algorithm is computed as:

$$eff\% = \left(1 - \frac{\left|N_c^f - 3\right|}{N_c}\right) \times 100 \tag{31}$$

with N_c^f indicates the final number of hidden neurons for the obtained structure after the pruning procedure.

The initial values of weights and the numerical parameters are selected based on several simulations to obtain the best performances in terms of training capacities and convergence time. The simulation results

describing the performances of Unit-OBS, Fast-Unit-OBS and the proposed algorithms are given in table 1 where for each algorithm a set of initial neural structures has been considered.

Table 2 shows the average of performances for the studied algorithms.

It can be seen that the proposed approach provide better performances. On the other hand, the proposed algorithm avoids the complex calculation of the inverse of the Hessian matrix which considers the main advantage of this method. To confirm these interpretations, we test the investigated algorithms on neural modeling problem of an induction motor.

Table 1: Performances of the studied algorithms.

Algorithm	Numerical simulation Parameters	N_c	N_c^f	Efficiency (%)	Generalization error	Run time
Unit-OBS	ε=0.6, E_t=8.10^{-5} J_t=0.003, α=9.10^{-5}	5	3	100	0.0036	0'11"
		8	5	75	0.0032	0'11"
		11	4	90.91	0.0035	0'18"
		14	3	100	0.0035	1'11"
		20	4	95	0.0037	1'12"
		25	3	100	0.0035	2'08"
Fast-Unit-OBS	ε=0.6, L_t=J_t=0.003, α=10^{-8}	5	3	100	0.0036	0'10"
		8	3	100	0.0033	0'09"
		11	4	90.91	0.0035	0'14"
		14	2	92.86	0.0034	1'36"
		20	5	90	0.0037	0'46"
		25	4	96	0.0036	0'43"
Proposed algorithm	ε=0.6, J_t=0.003, t=20, λ=1,05	5	3	100	0.0034	0'15"
		8	3	100	0.0034	0'11"
		11	4	90.91	0.0034	0'14"
		14	3	100	0.0034	1'05"
		20	3	100	0.0033	0'47"
		25	4	96	0.0032	0'31"

Table 2: Average of performances for the studied algorithms.

Algorithm	Efficiency (%)	Generalization error	Run time
Unit-OBS	93.49	0.0035	0'52"
Fast-Unit-OBS	95.11	0.0035	0'36"
Proposed algorithm	97.82	0.0034	0'31"

2. Second example:

Induction motor description and control are difficult engineering problems, which are still not completely resolved. This is due to the complexity, highly nonlinear and time-varying parameters of the dynamic model of the machine.

Induction motor dynamics are described by means of space vector. This is a common procedure employed within the electrical machine in order to reduce the number of equations needed to describe the motor model. In fact, the three-phase stator windings can be transformed, using the Park transformation, into equivalent quadrature-phase windings. In this case, the dynamics of induction motor (W. Leonhard, 1996; P. Vas, 1990; M. Chtourou *et al*., 1996) are then given as follows:

$$\begin{cases} \dot{I}_{ds} = \lambda_s V_{ds} + \omega_s I_{qs} - \alpha_s I_{ds} + p\omega_m \lambda_m \psi_{qr} + \delta\psi_{dr} \\ \dot{I}_{qs} = \lambda_s V_{qs} - \omega_s I_{ds} - \alpha_s I_{qs} - p\omega_m \lambda_m \psi_{dr} + \delta\psi_{qr} \\ \dot{\psi}_{dr} = \beta I_{ds} - \alpha_r \psi_{dr} + (\omega_s - p\omega_m)\psi_{qr} \\ \dot{\psi}_{qr} = \beta I_{qs} - \alpha_r \psi_{qr} - (\omega_s - p\omega_m)\psi_{dr} \\ \dot{\omega_m} = \frac{1}{J_r}\left[\frac{pm}{L_r}\left(\psi_{dr} I_{qs} - \psi_{qr} I_{ds}\right) - C_r\right] \end{cases} \quad (32)$$

where: d and q denote direct and quadrature. Subscripts r and s stand for rotor and stator. I, V and ψ denote current, voltage and flux respectively. J_r and p represent rotor inertia and number of pole pairs respectively. ω_m and ω_s are machine and stator speeds, respectively. C_r is the load torque.

With:

- $\lambda_s = \frac{L_r}{D}, \lambda_r = \frac{L_s}{D}$ and $\lambda_m = \frac{m}{D}$ $(D = L_s L_r - m^2)$, where L and m are the auto inductance and the mutual inductance respectively.
- $\alpha_r = \frac{R_r}{L_r}$ and $\alpha_s = R_s\lambda_s + \frac{R_r \lambda_m m}{L_r}$, where R is the resistance.
- $\beta = m\alpha_r$ and $\delta = \frac{R_r \lambda_m}{L_r}$.

The system of equations (32) can be rewritten in the condensed form:

$$\dot{X} = F(X, U) \tag{33}$$

with:

$$X = \left[I_{ds}\ I_{qs}\ \psi_{dr}\ \psi_{qr}\ \omega_m\right]^T \tag{34}$$

$$U = \left[V_{ds}\ V_{qs}\ \omega_s\right]^T \tag{35}$$

The control vector *U* can be reduced to stator voltage V_{ds} if $V_{qs} = 0$ and the stator speed ω_s is constant. The model output is the machine speed ω_m.

Machine parameters used in simulations:

$$\begin{cases} \omega_s = 100\pi\ rads^{-1},\ V_{dsn} = 220\sqrt{3}\ V, p = 2, J_m = 0.5\ Kgm^2 \\ R_s = 0.29\ \Omega, R_r = 0.38\ \Omega, L_s = L_r = 0.05\ H, m = 0.0473\ H \\ \text{and } \mathrm{C_r} = \mathrm{k}\omega_m,\ k = 0.1070 \end{cases}$$

The input-output neural model describing the induction motor is presented in Figure 5.

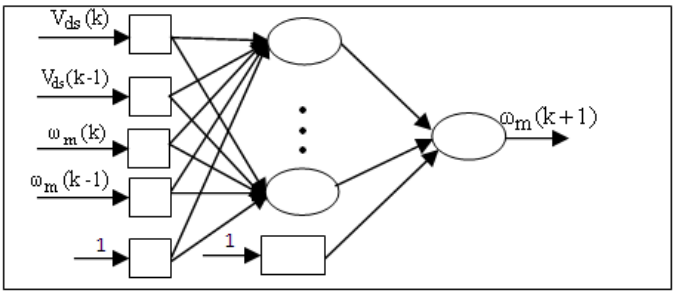

Fig. 5: *Input-output neural model of the induction motor.*

The model input vector is constituted by the actual and previous voltage ($V_{ds}(k)$ and $V_{ds}(k-1)$) and the actual and previous machine speed ($\omega_m(k)$ and $\omega_m(k-1)$). The model output is the future value of the machine speed $\omega_m(k+1)$.

The selection of the input vector and the value of the sampling period (*Δt=0.001s*) have been done after several simulations.

Figure 6 represent respectively the input signal $V_{ds}(k)$, used in the training (6.a) and validation (6.b) phases.

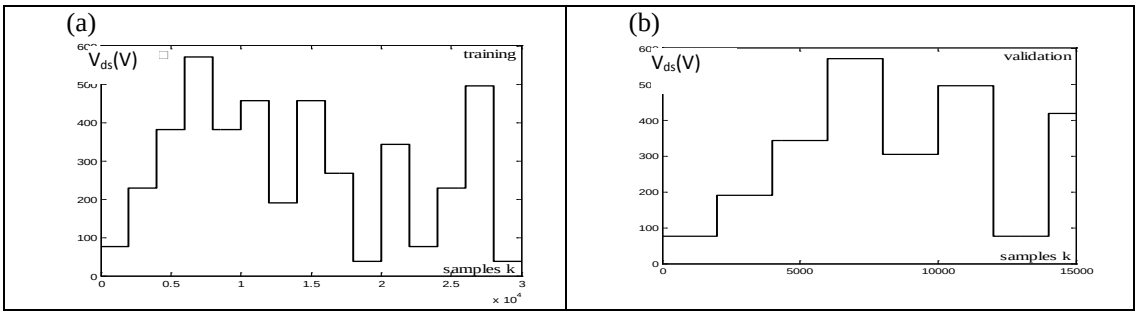

Fig. 6: *$V_{ds}(k)$ used in the training and validation phases ((a): training, (b): validation).*

It is to be noted that each training algorithms have been executed for different initial neural structures.

The simulation results describing the performances of the algorithms studied in this paper are illustrated in Table 3.

It can be seen that the proposed algorithm provides better performances when compared to Unit-OBS and Fast-Unit-OBS algorithms. For better illustration of the results presented in Table 3, they can be summarized in Table 4 which presents the average performance on final architecture, generalization capacities and convergence time.

Table 3: Algorithms Performances for the induction motor neural modeling problem.

Algorithm	Numerical simulation Parameters	N_c	N_c^f	Generalization error	Run time
Unit-OBS	ε=0.2, E_t=8.10^{-5} J_t=0.07, α=9.10^{-5}	10	5	0.9434	3'25"
		15	7	1.1787	4'15"
		20	4	1.5536	8'29"
		25	7	1.8714	10'46"
Fast-Unit-OBS	ε=0.2, J_t=0.07, L_t= 0.002, α=10^{-8}	10	7	1.0277	2'44"
		15	9	1.3090	8'27"
		20	12	1.1896	3'33"
		25	19	0.9417	3'38"
Proposed algorithm	ε=0.2, J_t=0.07, t=10, λ=1,15	10	4	0.9535	2'57"
		15	6	0.9725	4'09"
		20	4	1.1035	5'26"
		25	7	1.1294	7'00"

Table 4: Average of performances.

Algorithm	N_c^f	Generalization error	Run time
Unit-OBS	6	1.3868	6'44"
Fast-Unit-OBS	12	1.1170	5'06"
Proposed algorithm	5	1.0397	6'08"

Table 4 shows the contribution of the proposed algorithm. In fact, we note that the algorithm based on pseudo-entropy of hidden neurons leads to the simplest neural structure and presents the least generalization error with satisfactory convergence time. Moreover, the proposed method avoids the complex computation of the inverse of the Hessian matrix which considers the major drawback of the OBS approach.

Figure 7 gives the evolution of the identified neural model output for the training (7.a) and validation (7.b) sets using the proposed algorithm.

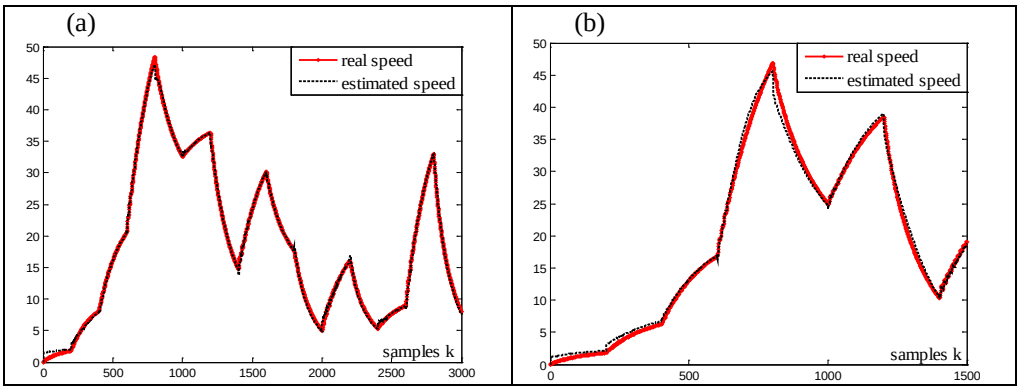

Fig. 7: *Training and validation performances for the proposed algorithm ((a): training, (b): validation).*

Conclusion:

In this paper, the OBS pruning approach and its modified versions have been studied. In second time, a pruning algorithm based on pseudo-entropy of hidden neurons has been introduced allowing to obtain the adequate neural structure. The main contribution of this method is that it avoids the calculating of the inverse of the Hessian matrix which is indispensable for using any pruning algorithm based on OBS approach. To confirm the validity of the elaborated algorithm, it has been applied for an academic example and the neural modeling of induction motor. The simulation results have demonstrated the good performance of the proposed algorithm for the neural network architecture optimization. The obtained network has acceptable generalization ability with a good efficiency what shows the utility of this algorithm.

REFERENCES

Chan, K.H., G. See Ng, S.S. Erdogan, H. Singh, 1999. New entropy learning method for neural network. IEEE Conference Proceedings on Systems, Man, and Cybernetics, 3: 412-417.

Chtourou, M., N. Derbel and M.B.A. Kamoun, 1996. Control of a loaded induction machine using a feedforward neural network. Int Journ. of Syst. Sciences, 27(12): 1267-1295.

Hagiwara, M., 1993. Removal of hidden units and weights for backpropagation networks. Proc. Int. Joint Con Neural Networks, 1: 351-354.

Hassibi, B. and D.G. Stork, 1993. Second-order derivatives for network pruning: Optimal brain surgeon. Advances in Neural Information Processing Systems, 5: 164-171.

Hassibi, B., D. Stork, and G. Wolff, 1993. Optimal brain surgeon and general network pruning. IEEE Int. Conf. Neural Networks, 1: 293-299.

Honggui, H. and Q. Junfei, 2009. A Novel Pruning Algorithm for Self-organizing Neural Network. Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, pp: 1245-1250.

Juan, D. and E.M. Joo, 2010. A Fast Pruning Algorithm for an Efficient Adaptive Fuzzy Neural Network. 8th IEEE International Conference on Control and Automation Xiamen, China, 1031-1035.

Kamimura, R., 1993. Competitive learning by entropy minimization. Conference: Algorithmic Learning Theory - ALT, 743: 111-122.

Kamimura, R., 1995. Selective responses by entropy minimization. Mathematical and Computer Modelling, 21(1–2): 143-157.

Kamimura, R., 2002. Information theoretic neural computation. World Scientific.

Kamimura, R., T. Toshiyuki, N. Shohachiro, 1994. Improving generalization performance by information minimization. IEEE International Conference on Computational Intelligence, Neural Networks, 1: 143-148.

Kamimura, R. and T. Kamimura, 2000. Information theoretic rule discovery in neural networks. IEEE International Conference on Systems, Man, and Cybernetics - SMC, 4: 2569-2574.

Kamimura, R., S. Nakanishi, Kernel Hidden, 1995. Unit Analysis Network Size Reduction by Entropy Minimization. EICE TRANSACTIONS on Information and Systems, E78-D(4): 484-489.

Le Cun, Y., L.S. Denker, and S.A. Solla, 1990. Optimal brain damage. Advances in Neural Information Processing systems, D.S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 2: 598-605.

Leonhard, W., 1996. Control of electrical drives. Springer-Verlag.

Liu, D., T.S. Chang, and Y. Zhang, 2002. A Constructive Algorithm for Feedforward Neural Networks with Incremental Training. IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications, 49(12): 1876-1879.

Ma, L. and K. Khorasani, 2004. New training strategies for constructive neural networks with application to regression problems. Journal Neural Networks, 17(4): 589-609.

Mechernene, A., M. Zerikat, S. Chekroun, 2011. Indirect Field Oriented Adaptive Control of Induction Motor based on Neuro-Fuzzy Controller. J. Electrical Systems, 7(3): 308-319.

Mehra, P. and B.W. Wah, 1992. Artificial Neural Networks Concepts and Theory. IEEE Comput. Society Press.

Qiao, J.-F., Y. Zhang, H.-G. Han, 2008. Fast unit pruning algorithm for feedforward neural network design. Applied Mathematics and Computation, 205: 622-627.

Reed, R., 1993. Pruning algorithms-A survey. IEEE Trans. Neural Net., 4(5): 740-747.

Rivals, I. and L. Personnaz, 2003. Réseaux de neurones formels pour la modélisation, la commande et la classification. Collection sciences et techniques de l'ingénieur dirigée par Suzanne LAVAL.

Rumelhart, D.E., G.E., Hinton and R. J.Williams, 1986. Learning internal representations by error Propagation. Parallel Distributed Processing. MIT Press, Cambridge, MA, 1: 318-362.

See Ng, G., Abdul Wahab And D. Shi, 2003. Entropy learning and relevance criteria for Neural network pruning. International Journal of Neural Systems, 13(5): 291-305.

Shannon, C. and W. Weaver, 1948. The mathematical theory of communication. Bell System Technical Journal, 27: 378-423.

Sinha, A., D.N. Vishwakarma, 2013. Pattern Classification based Intelligent Numerical Protection of Salient-pole Synchronous Generator using Neural Networks. J. Electrical Systems, 9(1): 125-136.

Stahlberger, A. and M. Riedmiller, 1997. Fast network pruning and feature extraction using the unit-OBS algorithm. Advances in Neural Information Processing Systems, Denver, 9: 2-5.

Vas, P., 1990. Vector control of AC machine. Ed. Oxford University Press.

Zhang, Z. and J. Qiao, 2010. A Node Pruning Algorithm for Feedforward Neural Network Based on Neural Complexity. International Conference on Intelligent Control and Information Processing, - Dalian, China, 406-410.

Zhao, S., Q. Liu, B. Zhang, 2010. A Fast OBS Pruning Algorithm Based on Pseudo-Entropy of Weights. 2nd IEEE International Conference on Advanced Computer Control (ICACC), 3: 451-455.

Zitouni, N., R. Andoulsi, A. Sellami, A. Mam, A. Hassen, 2011. Sliding-Mode Control for an artificial neural network Analysis and Design of treatment water system. J. Electrical Systems, 7(2): 237-257.

Zurada, J.M., 1992. Introduction to Artificial Neural Systems. St Paul, MN: West.