A Pruning Algorithm Based on Relevancy Index of Hidden Neurons Outputs

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

Mohamed Chtourou
Control & Energy Management Lab
(CEM LAB)
National School of Engineering of Sfax
University of Sfax, Tunisia
mohamed.chtourou@enis.rnu.tn

Mohamed Djemel
Control & Energy Management Lab
(CEM LAB)
National School of Engineering of Sfax
University of Sfax, Tunisia
mohamed.djemel@enis.rnu.tn

***Abstract*—Choosing the training algorithm and determining the architecture of artificial neural networks are very important issues with large application. There are no general methods which permit the estimation of the adequate neural networks size. In order to achieve this goal, a pruning algorithm based on the relevancy index of hidden neurons outputs is developed in this paper. The relevancy index depends on the output amplitude of each hidden neuron and estimates his contribution on the learning process. This method is validated with an academic example and it is tested on a wind turbine modeling problem. Compared with two modified versions of Optimal Brain Surgeon (OBS) algorithm, the developed approach gives interesting results.**

Keywords- pruning algorithm; OBS approach; relevancy index; hidden neurons

I. INTRODUCTION

The Feedforward Neural Networks (FNNs) [1, 2] have been successfully used to solve many problems, such as: dynamic system identification, signal processing, pattern classification [3, 4] and intelligent control [5]. One of the difficulties of a FNNs applying process is the determination of the optimal network architecture. In general, if the network structure is too small, it may not be able to learn the training samples. On the other hand, large-sized networks learn easily but show poor generalization capacities due to over-fitting. Thus, algorithms that can determine the appropriate network architecture automatically are highly desirable. Research in this field can be classified in two categories: constructive [6] and pruning approaches [7]. Recent interest has been growing on pruning strategies [8-10] that start with large-sized network and remove unnecessary hidden neurons or weights either during the training phase or after convergence to a local minimum. The most known methods are Optimal Brain Damage (OBD) [11] and Optimal Brain Surgeon (OBS) [12] which eliminate the neurons or weights with the smallest saliency one by one, which significantly increases the complexity of the procedure computation and the running time.

In this paper, an improved pruning algorithm based on hidden neurons' outputs is investigated and compared with two algorithms derived from the OBS method. The rest of paper is organized as follows. Section 2 presents briefly analysis of OBS algorithm and the description of two modified versions of this algorithm namely (Unit-OBS) [13] and (Fast-Unit-OBS) [14]. A pruning algorithm based on relevancy index of hidden neurons outputs is introduced in section 3. Section 4 illustrates the obtained simulation results. Finally, the conclusion is presented in section 5.

II. RELATED WORKS

A simple (FNN)s with a single output is represented in figure 1 (the generalization to more outputs units is straightforward).

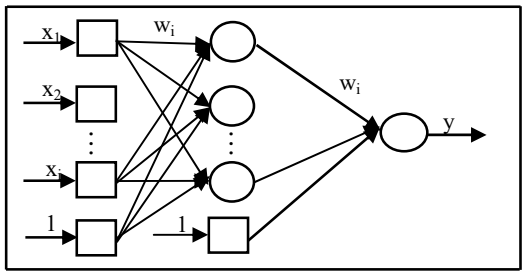

Fig. 1. Feedforward Neural Network.

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

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

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

The error function for a given pattern is defined as:

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

The global error function is described as:

$$J = \frac{1}{N}\sum_{p=1}^{N} J_p \quad (3)$$

A. OBS approch

In [11], an OBD method which calculates the saliency only with the pivot elements of the Hessian matrix without retraining after the pruning process was introduced. To overcome this problem, the (OBS) algorithm which determines and removes the weight that has the smallest effect on the neural network performance and adjusts the remaining weights according to the error function gradient was proposed [12]. 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:

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

where H denotes the Hessian matrix composed by the coefficients of the second order derivatives of the error function.

In order to minimize the error given by (3), OBS algorithm deletes one of the weights having a value tending to 0 and removes the particular weight wq which satisfies the following equation:

$$e_q^T \delta w + w_q = 0 \quad (5)$$

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:

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

So as 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:

$$L_q = \frac{1}{2}\frac{w_q^2}{\left[H^{-1}\right]_{qq}} \quad (7)$$

The remaining weights are updated according the following equation:

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

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. It has been proposed a procedure to calculate H^{-1} based on a recursive method as [12]:

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

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

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

B. Unit-OBS algorithm

The Unit-OBS pruning algorithm removes the unneeded neuron in one step with minimal increase in error [15]. This approach 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:

$$M = [e_{q1}, e_{q2}, \cdots, e_{qm(u)}] \quad (10)$$

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

$$e_{qi}^T = \left[0, \cdots, \underset{i}{1}, 0, \cdots 0\right] \quad (11)$$

  - Calculate the error deviation after deleting neuron u:

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

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

  where E_t indicates a preselected threshold.

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

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

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 a modified version of Unit-OBS algorithm has been proposed in [14].

C. 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:

$$L_u = \frac{1}{2}\frac{\overline{w}_u^2}{\left[H^{-1}\right]_{uu}} \quad (14)$$

where the weight mean is given by:

$$\overline{w}_u = \frac{1}{m}\sum_{i=1}^{m} w_{ui} \quad (15)$$

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

The steps 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 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:

$$\Delta w = -\frac{\overline{w}_u}{\left[H^{-1}\right]_{uu}} H^{-1} e_u \quad (16)$$

with :

$$e_u = \left[0,\cdots,\underset{u}{1},0,\cdots 0\right] \quad (17)$$

  - return to the *step 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 of calculating the inverse of the Hessian matrix .To overcome this problem, a pruning algorithm that eliminates the hidden neurons having a low impact on learning performances without using the inverse of the matrix is proposed in the next section.

III. PROPOSED ALGORITHM

The basic idea of the algorithm proposed in this paper has been inspired from the pruning approach in [16] which consists of estimating the sensitivity of the global error changes with respect to each connection during the training phase and removing the weight which presents the smallest sensitivity value. The neural network will be retrained and the pruning process will be executed as capacities learning are satisfactory. However, the success of this approach depends on the size of the initial neural structure. Indeed, for the case of large-sized network, the number of weight increases hugely which degrades the performance of the approach significantly.

The proposed method consists of finding the contribution of each hidden neuron in the network which reduces the complexity of the procedure and eliminates the one having the least effect on the cost function. This effect is estimated by calculating the relevancy index which depends on the output amplitude of each hidden neuron defined as follows:

$$RI_u = \frac{1}{N}\sum_{p=1}^{N}\left(\frac{O_u^p}{\sum_{u=1}^{N_c} O_u^p}\right) \quad (18)$$

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

The average value of relevancy index is defined as:

$$RI_u^{av} = \frac{\sum_{u=1}^{N_c} RI_u}{N_c} \quad (19)$$

Each hidden neuron with a relevancy index smallest than the average value should be deleted.

It is to be noted that we are interested in a multi input-single output (MISO) model and the weights update law is based on the back-propagation algorithm which can explained as:

$$\delta w = -\varepsilon\frac{\partial J_p}{\partial w} \quad (20)$$

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 the relevancy index for each hidden neuron using equation (18).
- Step 3: Remove each hidden neuron u which satisfies the following test:

$$RI_u < RI_u^{av} \quad (21)$$

If there is no neuron that satisfies the test (21), than go to step 5.

- Step 4: Maintaining the same average value of the relevance index, train the obtained structure with a reduced number of iterations noted *m* and return to 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 strategy avoiding the computation the inverse of Hessian matrix which consists the main contribution of this approach.

IV. SIMULATIONS RESULTS AND DISCUSSIONS

In this section, we present the simulation results. Initially, the capacities of the proposed algorithm are verified on an academic example. Then, we use this algorithm for wind

turbine 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.

A. First example

The goal is to present the learning set to the studied algorithms and to show that they can determine a number of hidden neurons close to the one of the neural network used to generate the learning set.

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

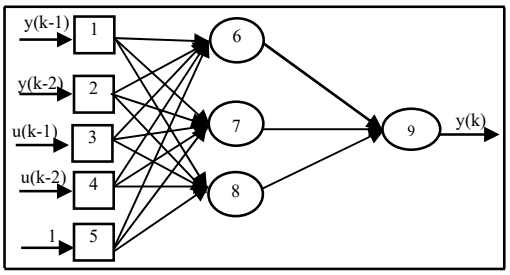

Fig. 2. Neural Network generating the learning set.

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

The efficiency of each algorithm is computed as:

$$\text{eff}\% = \left(1 - \frac{N_c^f - 3}{N_c}\right) \times 100 \qquad (22)$$

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

The initials 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.

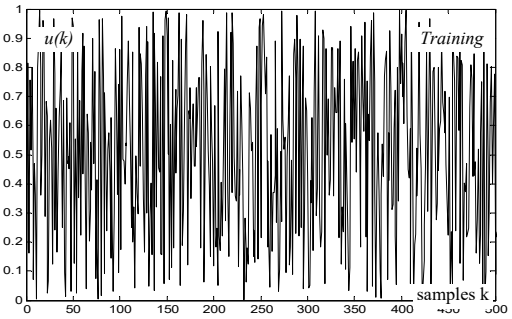

Fig. 3. Training input signal.

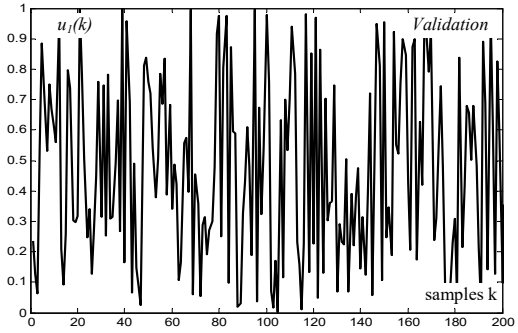

Fig. 4. Validation input signal.

The simulation results describing the performances of Unit-OBS, Fast-Unit-OBS and the proposed algorithms are given in Table I where for each algorithm a set of initial neural structures has been considered. Table II shows the average of performances for the studied algorithms.

It can be seen that the investigated approaches provide similar and satisfactory 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 studied algorithms on neural modeling problem of wind turbine.

B. Second example

In the last years, a growing interest in renewable energy has been evident [17]. Wind energy has become competitive and considerable technological progress has been achieved in the field of wind turbines. The energy production depends on many factors essentially the choice of robust methods for controlling wind turbines [18]. This step can be accomplished as well as the model describes the system dynamics correctly, which shows the importance of the selection of modeling strategy.

In this work, modeling of the turbine is provided via a neural model whose architecture is selected based on a pruning approach. The rigid model with only one degree of freedom [19-21] is given by:

$$I_t \dot{\omega}_a = T_a - T_g - k_t \omega_a \qquad (23)$$

The point (.) designates the first order time derivative, T_g is the generator torque and I_r, I_g, k_r, k_g are the moment of inertia of rotor side masses, the moment of inertia of generator side masses, the mechanical damping in the rotor side and the mechanical damping in the generator side respectively, where:

$I_t = I_r + n_g^2 I_g$ is the total inertia of generator side masses,

$k_t = k_r + n_g^2 k_g$ is the equivalent mechanical damping.

For model (23) $\omega_g = n_g \omega_a$ is satisfied, where: $\omega_a = \dot{\theta}_a$ is the rotor rotational speed and $\omega_g = \dot{\theta}_g$ is the rotational speed of the high speed shaft, while n_g designates the gear ratio between the primary shaft and the secondary shaft, θ_a and θ_g are the azimuthally rotor position and the azimuthally position of the

high speed shaft. The captured aerodynamic torque Ta is given in terms of the power coefficient $C_p(\lambda,\beta)$ as:

$$T_a(k)=\frac{1}{2}\phi\pi R^2\frac{v^3}{\omega_a(k)}C_p(\lambda,\beta) \quad (24)$$

where λ is the specific speed defined as:

$$\lambda=\frac{R\omega_a}{v} \quad (25)$$

v is the effective wind speed, ϕ is the air density, and R denotes the blades rotor radius. The power coefficient $C_p(\lambda,\beta)$ is estimated using aerodynamic data obtained from wind tunnel measurements. It is generally represented under the form of an analytical formula which gives $C_p(\lambda)$ for various values of the pitch angle β.

In the literature [22] one finds the following approximation:

$$C_p(\lambda,\beta)=A\,\mathbf{exp}[B] \quad (26)$$

where:

$$\begin{cases} A=c_1G+c_4B+c_5 \\ B=c_6G \\ G=\dfrac{1}{\lambda+c_2\beta}+\dfrac{c_3}{\beta^3+2} \end{cases} \quad (26)$$

with the coefficients c_i, i=1... 6 are identified from real Cp curves.

Using the Euler approximation and for small values of the sampling period, the discrete model describing the wind turbine can be expressed as follows:

$$\omega_a(k+1)=\omega_a(k)+\Delta t\left[\frac{T_a(k)}{I_t}-\frac{T_g(k)}{I_t}-\frac{k_t}{I_t}\omega_a(k)\right] \quad (27)$$

Δt denotes the sampling period.

The goal of this study is to determine an adequate neural model of the wind turbine. This model should be able to learn nonlinear dynamics of the wind turbine and gives good generalization ability.

TABLE I. 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, $\alpha=9.10^{-5}$	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, $\alpha=10^{-8}$	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, m=20	5	2	80	0.0033	0'23"
		8	3	100	0.0034	1'09"
		11	3	100	0.0032	0'29"
		14	3	100	0.0035	0'19"
		20	5	90	0.0035	0'40"
		25	5	92	0.0032	1'43"

The wind turbine parameters used in simulations are the following [23, 24]:

$$\begin{cases} \phi=1.225kgm^{-3},\ R=21.38m,\ n_g=43.165, \\ I_r=3.25.10^5\,kgm^2,\ I_g=34.4kgm^2,\ \beta=-1°, \\ k_r=1.5Nmrad^{-1}s,\ k_g=3.7Nmrad^{-1}s,\ \lambda_{opt}=7.5, \\ c_1=1.1023.10^2,\ c_2=-0.02,\ c_3=0.003.10^2, \\ c_4=-0.309082,\ c_5=-9.636.10^2\ and\ c_6=-18.4 \end{cases}$$

The input-output neural model describing the wind turbine is presented in Figure 5. The model input vector is constituted by the actual and previous generator torque ($T_g(k)$ and $T_g(k-1)$), the actual and previous rotor rotational speed ($\omega_a(k)$ and $\omega_a(k-1)$) and the actual value of wind speed v(k). The model output is the future value of the rotor rotational speed $\omega_a(k+1)$. The selection of the input vector and the value of the sampling period have been done after several simulations. The chosen mean wind speed was set to vmoy=12ms-1 and Δt=0.1s. Figures 6 and 7 represent respectively the input signals $T_g(k)$ and v(k), used in the training (6.a and 7.a) and validation (6.b and 7.b) phases.

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 III. 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 III, they can be summarized in Table IV which presents the average performance on final architecture, generalization capacities and convergence time.

Table IV shows the contribution of the proposed algorithm. In fact, we note that the algorithm based on relevancy index leads to the simplest neural structure and presents the least convergence time with satisfactory generalization abilities. 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 8 gives the evolution of the identified neural model output for the training (8.a) and validation (8.b) sets using the proposed algorithm.

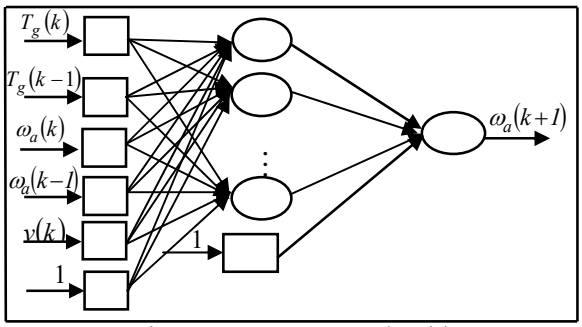

Fig. 5. Input-output neural model.

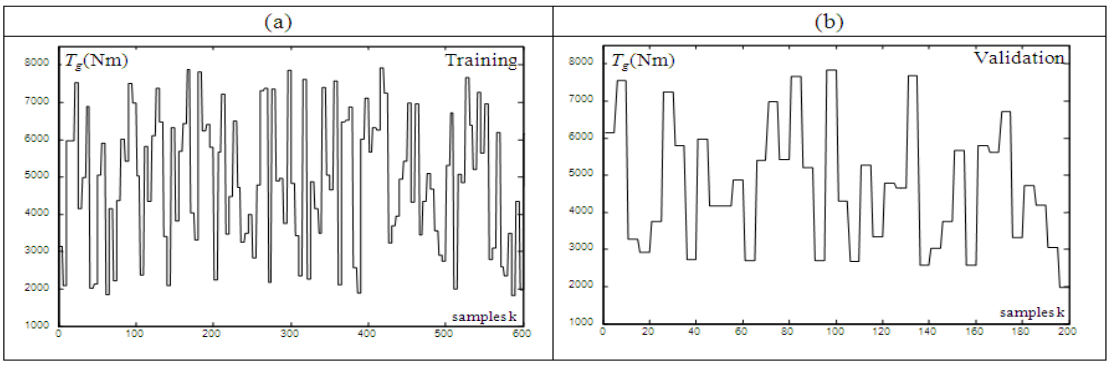

Fig. 6. Tg (k) used in the training and validation phases ((a): training, (b): validation).

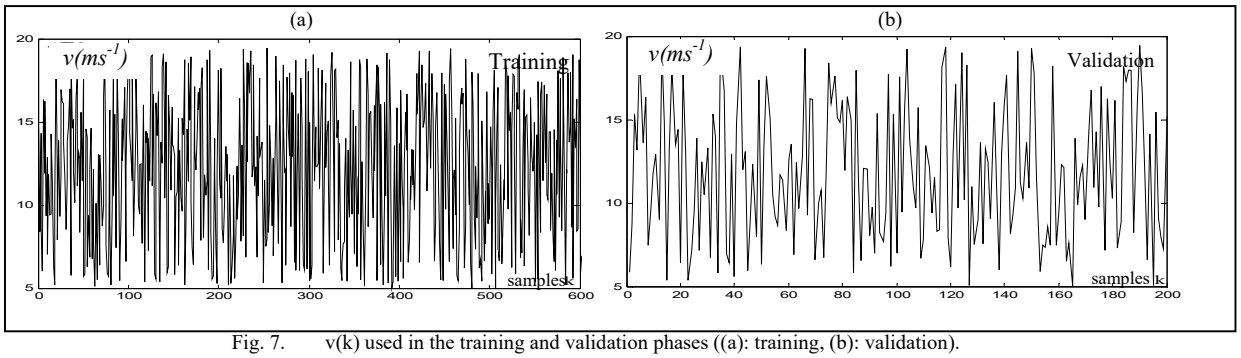

Fig. 7. v(k) used in the training and validation phases ((a): training, (b): validation).

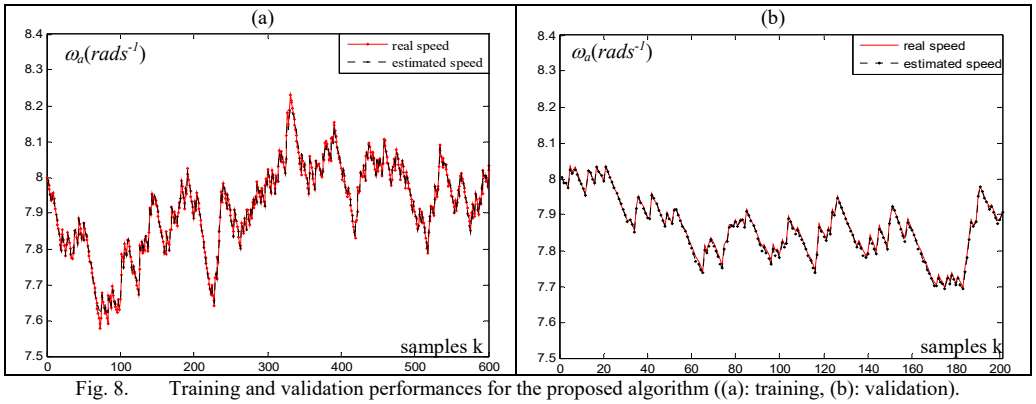

Fig. 8. Training and validation performances for the proposed algorithm ((a): training, (b): validation).

TABLE II. 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	93.67	0.0033	0'47"

TABLE III. ALGORITHMS PERFORMANCES FOR THE WIND TURBINE NEURAL MODELING PROBLEM.

Algorithm	Numerical simulation Parameters	N_c	N_c^f	Generalization error	Run time
Unit-OBS	ε=0.4, E_f=8.10^{-5} J_f=0.01, α=9.10^{-5}	10	9	0.0058	1'42"
		15	9	0.0080	2'01"
		20	6	0.0056	2'44"
		25	7	0.0048	2'41"
Fast-Unit-OBS	ε=0.4, J_f=0.01, L_f= 0.002, α=10^{-8}	10	8	0.0044	1'53"
		15	14	0.0048	1'34"
		20	13	0.0057	2'37"
		25	13	0.0067	2'02"
Proposed algorithm	ε=0.4, J_f=0.01, m=10	10	6	0.0063	1'34"
		15	6	0.0084	1'39"
		20	7	0.0048	2'04"
		25	5	0.0084	1'52"

TABLE IV. AVERAGE OF PERFORMANCES.

Algorithm	N_c^f	Generalization error	Run time
Unit-OBS	8	0.0061	2'17"
Fast-Unit-OBS	12	0.0054	2'02"
Proposed algorithm	6	0.0070	1'47"

V. CONCLUSION

This paper presents a pruning algorithm based on relevancy index that allows obtaining the adequate neural network structure. The main advantage 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 the OBS approach. To confirm the effectiveness of the developed algorithm, it has been applied on an academic example and on a wind turbine model. The simulation results demonstrated that the proposed algorithm, compared with the Unit-OBS and the Fast-Unit-OBS algorithms, not only ensures the same training and generalization performance but also shortens the runtime and simplifies considerably the obtained neural structure after pruning process what proves the potential utility of this algorithm.

REFERENCES

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

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

[3] V. E. Neagoe, C.T. Tudoran, "A neural machine vision model for road detection in autonomous navigation", U.P.B. Sci. Bull., Series C, Vol. 73, No. 2, pp. 167-178, 2011

[4] E. Şuşnea, "Using artificial neural networks in e-learning systems", U.P.B. Sci. Bull., Series C, Vol. 72, No. 4, pp. 91-100, 2010

[5] A. Mechernene, M. Zerikat, S. Chekroun, "Indirect field oriented adaptive control of induction motor based on neuro-fuzzy controller", J. Electrical Systems, Vol. 7, No. 3, pp. 308-319, 2011

[6] D. Liu, T. S. Chang, Y. Zhang, "A constructive algorithm for feedforward neural networks with incremental training", IEEE Transactions on Circuits and Systems. Fundamental Theory and Applications, Vol. 49, No. 12, pp. 1876-1879, 2002

[7] R. Reed, "Pruning algorithms-A survey", IEEE Trans. Neural Net., Vol. 4, No. 5, pp. 740-747, 1993

[8] H. Honggui, Q. Junfei, "A novel pruning algorithm for self-organizing neural network", International Joint Conference on Neural Networks, Atlanta, Georgia, USA, pp. 22-27, 2009

[9] D. Juan, E. M. Joo, "A fast pruning algorithm for an efficient adaptive fuzzy neural network", 8th IEEE International Conference on Control and Automation Xiamen, China, pp. 1030- 1035, 2010

[10] Z. Zhang, J. Qiao, "A node pruning algorithm for feedforward neural network based on neural complexity", International Conference on Intelligent Control and Information Processing, Dalian, China, pp. 406- 410, 2010

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

[12] B. Hassibi, D. Stork, G. Wolff, "Optimal brain surgeon and general network pruning", IEEE Int. Conf. Neural Networks, Vol. 1, pp. 293–299, 1993

[13] A. Stahlberger, M. Riedmiller, "Fast network pruning and feature extraction using the unit-OBS algorithm", Advances in Neural Information Processing Systems, Denver, Vol. 9, pp. 2-5, 1996

[14] J. -F. Qiao, Y. Zhang, H. -G. Han, "Fast unit pruning algorithm for feedforward neural network design", Applied Mathematics and Computation, Vol. 205, pp. 622–627, 2008

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

[16] E. D. Karnin, "A simple procedure for pruning backpropagation trained neural networks", IEEE Trans. Neural Networks, Vol. 1, pp. 239-242, 1990

[17] N. Ciprian, M. Florin, "Operational parameters evaluation for optimal wind energy systems development", U.P.B. Sci. Bull., Series C, Vol. 74, pp. 223-230, 2012

[18] A. Pintea. D. Popescu, "A comparative study of digital IMC and RST regulators applied on a wind turbine", U.P.B. Sci. Bull., Series C, Vol. 74, No. 4, pp. 27-38, 2012

[19] P. Christou, "Advanced materials for turbine blade manufacture", Reinforced Plastics, Vol. 51, No. 4, pp. 22-24, 2007

[20] L. Fingersh, M. Hand, A. Laxson, "Wind turbine design cost and scaling model", National Renewable Energy Laboratory, Technical Report NREL/TP-500-40566, 2006

[21] Y. D. Song, B. Dhinakaran, X. Y. Bao, "Variable speed control of wind turbines using nonlinear and adaptive algorithms", Wind Engineering and Industrial Aerodynamics, Vol. 85, pp. 293-308, 2000

[22] K. Reif, F. Sonnemann, R. Unbehauen, "Nonlinear state observation using H∞-filtering filtering riccati design", IEEE Transactions On Automatic Control, Vol. 44, No. 1, pp. 203-208, 1999

[23] A. Khamlichi, B. Ayyat, M. Bezzazi, L. El Bakkali, V. C. Vivas, C. L. F. Castano, "Modelling and control of flexible wind turbines without wind speed measurements", Australian Journal of Basic and Applied Sciences, Vol. 3, No. 4, pp. 3246-3258, 2009

[24] S. Abid, M. Chtourou, M. Djemel, "Incremental and Stable Training Algorithm for Wind Turbine Neural Modeling", Engineering Review (ER), Vol. 33, No. 3, pp. 165-172, 2013