Short term wind speed prediction using artificial neural networks based onLevenberg-Marquardt Optimization Method

BabakDehghan[1], RanaAmirzehni[2],BurakBarutçu[3]

Istanbul Technical University, Energy Institute, 34469 Maslak, Istanbul, Turkey

dehghanbakhshay@itu.edu.tr

Abstract

Wind speed prediction is critical for wind energy conversion systems since it greatly influences the issues such as the scheduling of a power system, and the dynamic control of the wind turbine.In this paper the short term wind speed forecasting in the region of Istanbul, Turkey, applyingthe technique of artificial neural network (ANN) based on Levenberg-Marquardt Optimization Methodto the 10 minute interval time series representative of the site ispresented and also we present a comprehensive comparison study on the application of different artificial neural networks architecturesin wind speed forecasting and compare triangular reduction method data with Hecht-Nielsen Approximation method data.The developed model for short term wind speed forecasting showed a very good accuracy and agreement between predicted and real data.The results are validated and the effectiveness of the triangular reduction method andHecht-Nielsen Approximation method is demonstrated.

1- MSc student at Istanbul Technical University, Istanbul, Turkey
2-MSc student at University of Tabriz, Tabriz, Iran
3-Assistant professor at Istanbul Technical University, Istanbul, Turkey

Keywords: Artificial neural networks, Wind speed forecasting, Time series, Levenberg–Marquardt algorithm

1-Introduction

Neural Networks have been widely used as time series predictors: most often these are feed-forward networks which employ a sliding window over the input sequence.

The artificial neural networks (ANN) is able to handle theuncertainties that appear in the solution of problems related to thereal world offering robust solutions and easy implementation ofa different way compared with traditional solutions.

Its use has been successful in a great variety of tasks in diversefields such as industrial processes, sciences and business [1]. Theforecasting has shown to be one of the most interesting applications [2,3]. The ability to predict accurately the future is fundamentalin the planning and decision making in any activity; theinherent nonlinear structure is useful to manage the complex relations.

The models for wind forecasting and power generation are valuable support tools to support the operators of the ElectricUtility Control Centre (EUCC) [9]. The measurements of the windspeed are generally reported in the form of time series (minutes,hours, months, etc.), which are adequate to use ANN for predictionpurposes [5,6].

In the present study, different ANN models have been developedfor the short term wind speed prediction in Canakkale, Turkey, using data measurements of year 2009 - 2010 obtained from 60 meter meteorology mast by 10 minutes time interval in the site.

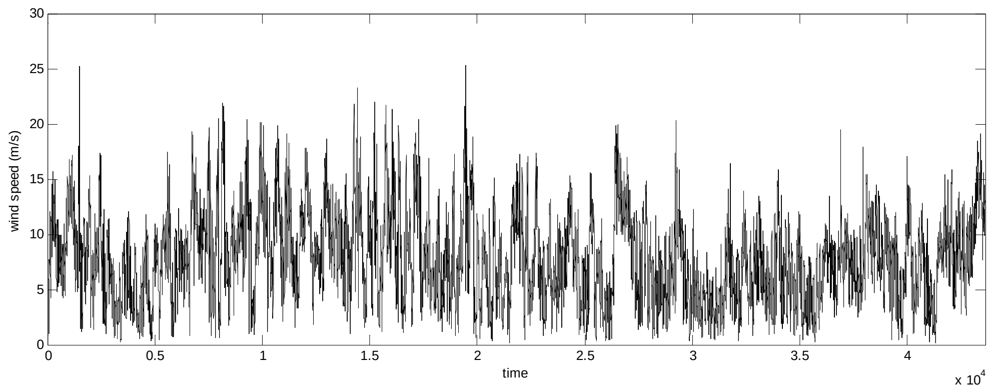

Fig.1. Wind speed data during 1 year collected by Canakkale, Turkey

2- Time series

A time series is a sequence of vectors, $\mathbf{x}(t)$, $t = 0, 1, \ldots,$ where $t$represents elapsed time. For simplicity we will consider here only sequences of scalars, although the techniques considered generalize readily to vector series. Theoretically,x may be a value which varies continuously with t, such astemperature. In practice,for any given physical system, x will be sampled to give a series of discretedata points, equally spaced in time. The rate at which samples are taken dictatesthe maximum resolution of the model; however, it is not always the case that themodel with the highest resolution has the best predictive power, so that superiorresults may be obtained by employing only every n^{th} point in the series. Work in neural networks has concentrated on forecasting future developmentsof the time series from values of x up to the current time. Formally this can bestated as: find a function $f: N \rightarrow N+d$ such as to obtain an estimate of x at time$t + d$, from the N time steps back from time t, so that:

$$x(t+d)=f(x(t), x(t\text{-}1), x(t\text{-}2),\ldots,x(t\text{-}N+1)) \qquad (1)$$

3- Artificial neural networks

An Artificial neural network (ANN), usually called “neural networks” (NN), are mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. ANN are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

4-Feed-Forward Neural Network Training Based on Levenberg-Marquardt Optimization Method

A Feed-forward multi-layer artificial neural networks consist of three basic components; an input layer, one or more hidden layers and an output layer Due to the fact; there isn't any connection between the nodes in the same layer. The input layer of ANN transfers the information it receives from outside to the internal processing unit. Hidden layers are used for feature extraction during the flow of information. Output layer gives the system output. There are two major steps in the use of a feed-forward ANN one is teaching and the other is the recall [7,8].

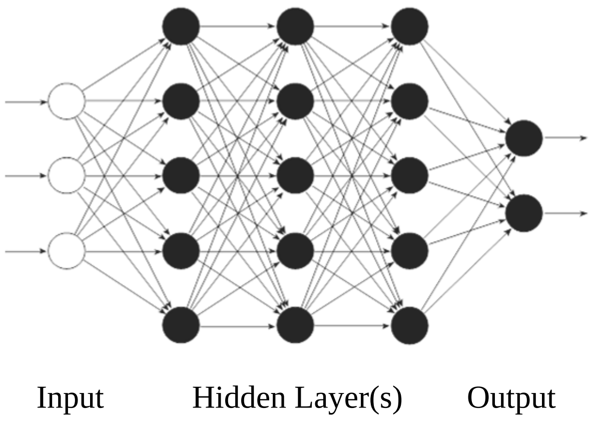

Fig.2. Artificial Neural Network

A Feed-forward neural network converts a series of input data $x(x_1,\ldots,x_n)$ in to one or more output data $\mathbf{y}(y_1,y_m)$. In this manner, a mathematical relationship can be given as follows,

$$y_j = f_j(x) \qquad j = 1,\ldots\ldots,m \tag{2}$$

However, a neural network does not express such a relationship in an explicit manner, but it internally contains weigh factors in an implicit manner. Figure 2 shows a feed-forward, three layer neural network that is defined as input layer, hidden layer and output layer respectively. In calculating the net input of a neuron *j*, the inputs from all *m* neurones of the proceeding layer are considered. The values of input data set are usually normalised within the range from 0 to 1 or from –1 to +1. The net value of neuron *j*, Net_j, is calculated from all *n* input data, x_i.

$$Net_j = \sum_{i=1}^{n} w_{ji} \, x_i \tag{3}$$

Here the elements w_{ji} express the weight values between neurones *i* and *j*. The output, y_j, of the j^{th} is given by

$$y_j = f\,(Net_j) = 1 \Big/ \left[1 + \exp(-\alpha\; Net_j + \theta_j)\right] \tag{4}$$

The function*f* is called as an activation function and it is here a sigmoid function. θ_j is a threshold value for neuron *j*. During the training process the input patterns are fed to the network to give a result at the output side y_j. This result is compared with the target value of the output, t_j. Hence network's weights are changed to minimise the following error term.

$$J(w) = \frac{1}{2} \sum_j \left(y_j - \hat{y}_j\right)^2 \tag{5}$$

Using the error term, the weight factors are adjusted by means of a parameter optimisation technique [7,8]. For this purpose, Levenberg-Marquardt method is used as a parameter optimisation technique.

5-Levenberg-Marquardt Algorithm in Parameter Optimisation

Levenberg-Marquardt method uses a search direction that is a solution of the linear set of equations [9,10]. The Levenberg-Marquardt method uses a search direction that is a cross between the Gauss-Newton direction and the steepest descent.

Like the quasi-Newton methods, the Levenberg-Marquardt algorithm is designed to approach second-order-training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed forward networks), then the Hessian matrix can be approximated and the gradient can be computed as a Jacobian matrix.The Jacobian matrix contains first derivatives of the network errors with respect to the weights and biases, and ***e*** is a vector of network errors. The Jacobian matrix can be computed through a standard back-propagation technique that is much less complex than computing the Hessian matrix. Levenberg-Marquardt Algorithm can be summarised as follows.

Algorithm: Given a function *J*(*w*) of the sum of prediction errors

$$J(w) = \frac{1}{2}\sum_{k=1}^{K}\left[y(k) - \hat{y}(k)\right]^2 \qquad (6)$$

the minimising parameters

$$\hat{w} = \arg\min_{\theta} J(w) \qquad (7)$$

can be found by the following algorithm.

1.Initialise:

Set iteration index *l*=1.

Initialise $\hat{w}(1)$ and $\mu(1)$ andspecify η.

2. Evaluate the model and the residuals:

Evaluate $\hat{y}(k)$ and $\frac{\partial \hat{y}(k)}{\partial w_p}$ for all patterns *k* and parameters *p*. Compose the residual vector **R**

$$r_k = \hat{y}(k) - y(k) \qquad (8)$$

and compute $J(\hat{w}(l))$

$$J(\hat{w}(l)) = \frac{1}{2}\mathbf{R}^{\mathrm{T}}\mathbf{R} \qquad (9)$$

Compose the Jacobian matrix **G**

$$g_{k,p} = \frac{\partial \hat{y}(k)}{\partial \theta_p} \qquad (10)$$

3. Solve the parameter update

$$\Delta\hat{w}(l) = -\left[\mathbf{G}^{\mathrm{T}}\mathbf{G} + \mu(l)\mathbf{I}\right]^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{R} \qquad (11)$$

4. Repeat Step 2 using $\hat{w}(l) + \Delta\hat{w}(l)$ instead of $\hat{w}(l)$

compute $J(\hat{w}(l) + \Delta\hat{w}(l))$.

If $J(\hat{w}(l)+\Delta\hat{w}(l)) < J(\hat{w}(l))$ then increase the step size

$$\mu(l+1)=\mu(l)/\eta \tag{12}$$

and update the parameters

$$\hat{w}(l+1)=\hat{w}(l)+\Delta\hat{w}(l) \tag{13}$$

Otherwise reduce the step size

$$\mu(l+1)=\eta\mu(l) \tag{14}$$

5. Set *l*=*l*+1 and return to Step 2, or quit.

The main drawback of the Levenberg-Marquardt algorithm is that it requires the storage of some matrices that can be quite large for certain problems. Also, the size of the Jacobian matrix can be very large, which includes the number of training sets and the number of weights and biases in the network. It turns out that this matrix does not have to be computed and stored as a whole.[10, 12]

6- Proposed model and results

For the generation of the ANN models, it is convenient to know the following parameters: (i) the number of input vectors, (ii) the number of layers, (iii) the number of output vectors, (iv) thenumber of neurons, etc. In general rules do not exist, nevertheless, it is accepted that a feed-forward network with one hidden layer, with a transference function identified in the output unit and a logic function in the units of the intermediate layer, can approximate any continuous function in a reasonable way [7].

In this study a group of artificial neural networks (ANN) applied for the purpose of one step ahead prediction. The ANN architectures which applicable to this goal can be grouped under two main branches. In the first group which called SISO (Single Input Single Output) ANN, there is 1 node as input, 1 neuron as output layerand a group of (e.g. 5, 7 etc.) neurons are applied as the number of neurons in their hidden layer. In the second group which called MISO (Multi Input Single Output) ANN, a group of nodes (e.g. 3 or more) are set as input layer and 1 neuron applied to the output layer. In this group, sliding window technique is also used to test the improvement [13,14] and 2 different selection criteria are used separately for determining the number of neurons in the hidden layers,one is building as a triangular architecture (n+1)/2 and other isHecht-Nielsen's 2n+1 approximation[15] (which based on Kolmogorov's proof[16]). In both groups, ⅔ of the signal is set for the training and the rest ⅓ is used for the test. Training is performed by using Levenberg-Marquardt algorithm for in 500 steps. Logarithmic sigmoid function is used for all hidden layers and pure linear activation function is applied to all output layers. SISO ANN's could never achieve success levels as high as MISO architectures because of MISO networks are designed by the embedding approach in prediction theory but SISO's are not use any information about the past values of signal more than one step before. Because of this to concentrate on two main MISO architecture and give a comparison of them is chosen as the main aim of this study.

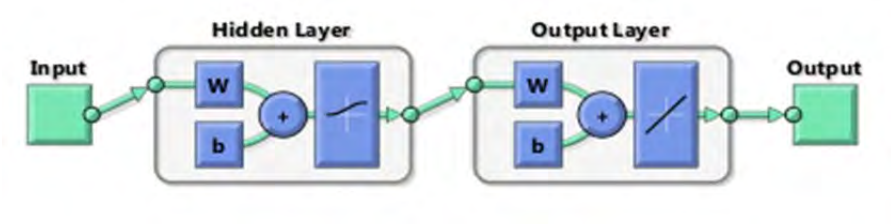

Fig.3. Computer Model of Neural Network

The two proposed MISO ANN architectures are shown in Fig. 4for three input nodes. Both group have logarithmic sigmoid activation functions in hidden layers and pure linear activation functions in output layers.

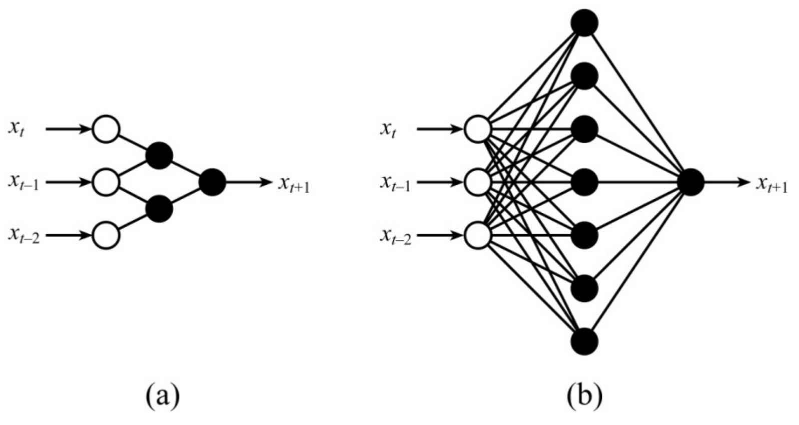

Fig. 4.MISO ANN architectures: (a)is triangular ANN with (3+1)/2=2 neurons in hidden layer and (b) is Hecht-Nielsen Approximation with 2×3+1=7 neurons in hidden layer.

As creating a suitable comparison scheme 4 models (in other words 4 different embedding dimensions) proposed for both architectures:

Model (1): 3 input nodes, (a)2neurons in hidden layer fortriangular reduction method(b)7 neurons in hidden layer forHecht-Nielsen Approximation, and 1 output neuron.

Model (2): 5 input nodes, (a) 3 neurons in hidden layer for triangular reduction method (b) 11 neurons in hidden layer for Hecht-Nielsen Approximation, and 1 output neuron.

Model (3): 7 input nodes, (a) 4 neurons in hidden layer for triangular reduction method (b) 15 neurons in hidden layer for Hecht-Nielsen Approximation, and 1 output neuron.

Model (4): 9 input nodes, (a) 5 neurons in hidden layer for triangular reduction method (b) 19 neurons in hidden layer for Hecht-Nielsen Approximation, and 1 output neuron.

It is pretended that in the course of the training, validation andprediction processes, other models can be identified and used.

For the sake of comparison 5 ANN created for 4 models in both architectures and these ANNs were trained separately by Lewenberg-Marquardt training algorithm with 500 steps. All feed-forward ANNs have logarithmic sigmoid activation functions in their hidden layers (because of wind speed is a non-negative signal) and pure linear activation functions in their output layers. Success levels of these ANNs are calculated by well-knownR^2 goodness of fit test. Results are presented in the following section.

7- Results and conclusion:

R^2 values of 5 different ANN trials for each architecture philosophy and 5 embedding dimensions are given in the following tables (Table. 1 and 2). Left hand columns of both tables are indicate the number of input nodes, number of hidden layer neurons and number of output neurons respectively. For instance 5 3 1 means an ANN which have 5 input nodes, 3 neurons in its hidden layer and 1 output neuron. ANN 01, 02, ... are indicate the number of trials. Comparison between two different architecture pairs with same embedding dimensions is depicted in the following Fig. 5.

ANN arc. ↓ / ANN # →	R^2 ANN 01	ANN 02	ANN 03	ANN 04	ANN 05
3 2 1	0.86768	0.86322	0.86610	0.86828	0.86387
5 3 1	0.84008	0.82056	0.82616	0.82979	0.82709
7 4 1	0.90294	0.90877	0.90103	0.90247	0.90221
9 5 1	0.91571	0.91767	0.90757	0.92418	0.91936

Table.1. R^2 (goodness of fit test) values of ANN Architectures which hidden layer neuron numbers determined by triangular reduction

ANN arc. ↓ / ANN # →	R^2 ANN 01	ANN 02	ANN 03	ANN 04	ANN 05
3 7 1	0.86048	0.86010	0.86111	0.86419	0.85584
5 11 1	0.83155	0.82728	0.82496	0.80374	0.83394
7 15 1	0.90453	0.88747	0.89346	0.89372	0.92119
9 19 1	0.91242	0.91369	0.91050	0.91314	0.91817

Table.2. R^2 values of ANN Architectures which hidden layer neuron numbers determined by Hecht-Nielsen Approximation

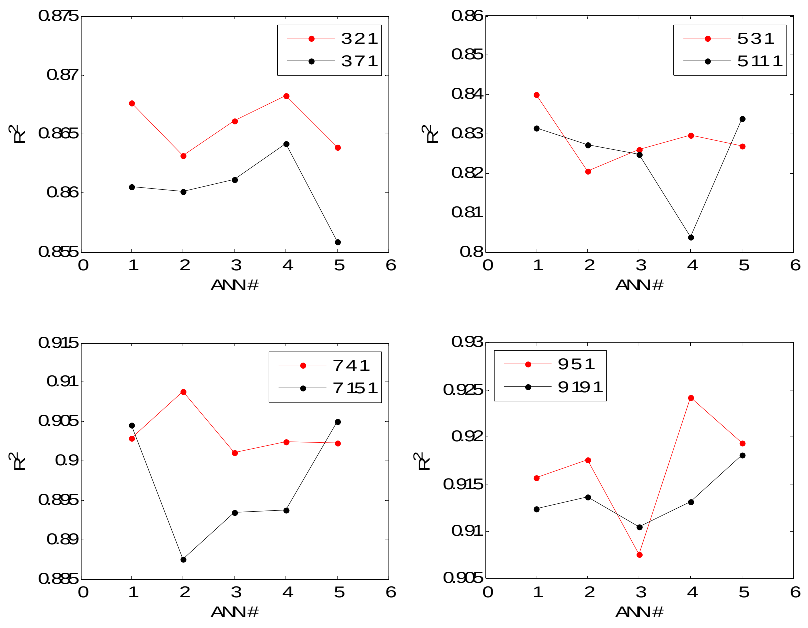

Fig. 5. R^2 values of 5 trials of each ANN architectures

For both architecture ANN's performance generally increase by increasing embedding dimensions (like the decreasing prediction error of AR models by increasing model order) but also it can be seen that performance of triangular ANN's are generally higher than the "2*n*+1" ANN's (Hecht-Nielsen Approximation). In this case triangular reduction can be defined as a better hidden neuron number determination algorithm than the Hecht-Nielsen approximation for this purpose. Also small number of hidden neurons means smaller weight matrices accordingly shorter processor times.

References

[1] Widrow B, Rumelhart D, Lehr MA. Neural networks: applications in industry, business and science. Communications of the ACM 1994; 37(3):93–105.

[2] Peter Zhang G. Neural networks in business forecasting. Georgia StateUniversity, USA. IRM Press, ISBN 1-59140-215-8; 2004.

[3] Kua-Ping L, Fildes R. The accuracy of a specifying feed forward neural networks forecasting. Computers & Operations Research 2005; 32:2151-69.

[4] Holttinen H. Optimal electricity market for wind power. Energy Policy 2005; 33:2052-63.

[5] Kariniotakis GS, Nogaret EF. Wind power forecasting using advanced neural networks models. IEEE Transactions on Energy Conversion 1996; 11(4):762‑7.

[6] Li S, Wunsch DC, O’Hair E, Giesselmann MG. Comparative analysis of regression and artificial neural network models for wind turbine power curve estimation. Journal of Solar Energy Engineering 2001;123:327‑32.

[7] Beale, R. and Jackson, T., 1998. Neural Computing: An Introduction, Adam Hilger Publishing. Ltd., GB.

[8] Haykin, S., 1999. Neural Networks: A Comprehensive Foundation, Prentice Hall Inc., GB.

[9] Levenberg, K., 1944. A Method for the Solution of Certain Problems in Least Squares, Quarterly Applied Math., 2, 164-168.

[10] Marquardt, D., W., 1963. An Algorithm for Least Squares estimation of Nonlinear Parameters, SIAM Journal Applied Math., 11, 431-441.

[11] Hornik K, Stinchcombe M, White H. Multilayer feed forward networks are universal approximations. Neural Networks 1989; 2:359‑66

[12] Li G, Shi J. On comparing three artificial neural networks for wind speed forecasting. Apply Energy 2010;87:2313–20.

[13]Dorffner, G., 1996, Neural Networks for Time Series Processing, Neural Network World 4/96, 447-468.

[14] Frank, R.J., Davey, N., Hunt S.P., 2001, Time Series Prediction and Neural Networks, J. Intell. Robot. Syst., 31, 1-3 91-103.

[15] Hecht-Nielsen, R., 1990, Neurocomputing, Addison-Wesley, USA.

[16] Kolmogorov, A. N., 1963, On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition, American Mathematical Society Translations, Series 2, vol. 28, 55-59