Forecasting of Wind Energy Generation using Self-Organizing Maps and Extreme Learning Machines

K. H. Tan
School of Electrical and Electronic Engineering,
Newcastle University, Singapore
k.h.tan3@newcastle.ac.uk

T. Logenthiran
School of Electrical and Electronic Engineering,
Newcastle University, Singapore
t.logenthiran@newcastle.ac.uk

W. L. Woo
School of Electrical and Electronic Engineering,
Newcastle University, England, UK
lok.woo@newcastle.ac.uk

***Abstract* - This paper aims to forecast wind energy generation. With accurate forecasting of energy generation, it will aid the energy sector in managing of stability and grid planning for supplied energy. The main focus of this project is Artificial Neural Network (ANN) while the training algorithms used in this project is a combination of Self-Organizing Maps (SOM) and Extreme Learning Machines (ELM). Furthermore, the training algorithm is applied into MATLAB and simulated several times in order to obtain the optimal parameters setting so as to accurately forecast wind energy generation.**

***Index Terms* – Wind energy Generation, Forecasting, Artificial neural network, Self-Organizing Maps, Extreme learning machine, MATLAB, Renewable energy resources**

I. Introduction

Overspending of the fossil fuels causes serious environmental pollutions all over the world. Since there is an increase in demand for electrical energy. Thus, the urgency to exploit into the renewable energy sector. Among the several renewable energies, wind is the most promising source of energy which can be connected to the electrical power system.

Despite wind energy generation having numerous benefits, it is a great challenge to accurately forecast wind energy as wind varies from the thermal exchanges between the surface and atmosphere. Not only does local climate and season affect the wind energy output, turbulences and chaotic processes further enhances the difficulty of forecasting. Fig. 1 shows the difference in wind energy generation in 3 consecutive days. Furthermore as power curves are nonlinear, any small errors in the forecasting of wind energy generation becomes a huge error in the power sent to the grid. Thus, an accurate algorithm to forecast wind energy generation is high in demand in the energy sector even today.

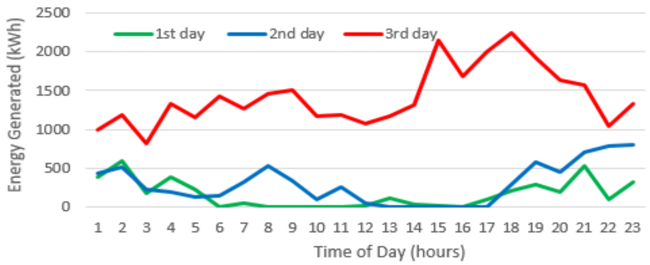

Fig. 1: Wind Energy Generation Example

The fundamental role of wind energy forecasting is to enable the program to successfully provide information of the expected wind power generation in the next few minutes, hours or days. Based on power system requirements, forecast are divided into 4 different horizons shown in Table I.

TABLE I. Forecasting Horizons Table

Types of Forecasting	Timing Horizons	Uses
Immediate Short-term	Few seconds – 30 minutes	Turbine Control and Load Tracking
Short-Term	30 minutes – 6 hours	Preload Sharing
Medium-Term	6 hours – 24 hours	Power System Management and Energy Trading
Long-Term	1 day – 7days	Maintenance Scheduling of Wind Turbines

In this paper, a methodology for forecasting wind energy generation using SOM and ELM as the training algorithm is presented. The developed forecasting model has been verified by some simulation studies. The rest of this paper is organized as follows: Section II presents an overview of the literature available on forecasting algorithm. Section III explains the proposed methodology used to implement the training. Section IV discusses the various simulations and results. Section V provides the discussion. Finally, the paper is concluded in Section VI.

II. Background Information

A. Machine Learning

Machine Learning is a computer-based algorithm which “learns” from the data. This algorithm is designed to handle forecasting models that can incorporate many kinds of data. Machine learning algorithms reduce variability by capturing and modelling all the relevant attributes while filtering out the random and unpredictable fluctuations. They are classified into three types, supervised learning, unsupervised learning and reinforced learning.

B. ANN Architecture

Inspired by the nervous system, ANN is formed under class of artificial intelligence which works like a brain-like tool with highly interconnected simple processing units designed to model how the human brain performs a particular task [1]. Since ANN is a subset of Machine Learning.

In this paper, the proposed methodology would be a combination of both unsupervised learning and supervised learning.

C. *Kohonen Self-Organizing Maps*

Originally proposed by Kohonen in 1982, SOM is a type of ANN using unsupervised learning method, trained to cluster and visualize high-dimensional data into low dimensional data (mostly two-dimensional) through the means of grouping similar data together to form a map [2].

Formation of SOM network involves three essential processes: Firstly, competitive process where for each input pattern, each neuron in the output layer compute its value for the distance in between to find the Best Matching Unit (BMU). Let *m* denote the dimension of the input pattern *x*

$$x = [x_1 \quad \cdots \quad x_m]^T \tag{1}$$

For each neuron, the weight vector, *w* has dimension *m*.

$$w_j = [w_{j1} \quad \cdots \quad w_{jm}]^T \tag{2}$$

In order to find the winning neuron, there is a need to calculate the Euclidean distance for each neuron. The shortest distance is declared the BMU.

$$d_{w,x} = \sqrt{\sum_{m=1}^{n} (x_m - w_{jm})^2} \tag{3}$$

Secondly, cooperative process where the BMU determines the spatial location of a topological neighborhood of excited neurons causing the neighboring neurons to cooperate. The lateral distance between two neurons *i* and *j* are denoted as d_{ji} and h_{ji} as a topological neighborhood to measure how close the neurons *i* and *j* are.

$$h_{j,i} = exp^{(\frac{-d_{j,i}^2}{2\sigma^2})} \tag{4}$$

where σ is the effective width of neighborhood, and it decreases over time.

$$\sigma(n) = \sigma_0 exp^{(\frac{-n}{\tau})} \tag{5}$$

where σ_0 is the initial value of σ, *n* is the iteration number and τ being the time constant determining the slope of graph.

Finally, the adaptive process where the excited neurons shorten their distance in relation to current input pattern via weight adjustments. After selecting of BMU and neighborhood neurons, the synaptic weights w_j are updated by

$$w_j(n+1) = w_j(n) + \eta(n) h_{ji(x)}(n)(x - w_j(n)) \tag{6}$$

where x is the input pattern, $\eta(n)$ is the rate of learning, $h_{ji(x)}(n)$ is the function of neighborhood and -$w_j(n)$ is the forgetting term in Hebbian hypothesis which stop the weight from going towards infinity. Table II discusses the details of each step needed to obtain the output map.

TABLE II. SOM ALGORITHM STEPS

Steps	Topologies	Applications
1	Initialization	Initializing of small random values for initial weight values of neurons.
2	Sampling	Selecting of input vector *x* at random.
3	Matching	Locating of winning neuron which distance is smallest form input *x*.
4	Updating	Adjusting of synaptic weight vectors of winning neuron and neighbors.
5	Iteration	Repetition of steps 2-4 until no noticeable map changes occurs.

D. *Single Hidden Layer Feed Forward Network (SLFN) – ELM*

SLFN is one of the many different types of Neural Networks that have been used extensively for the ability to directly approximate complex nonlinear mappings from input samples and provide large class of natural and artificial phenomena which are difficult to handle. However SLFN is not only tedious, it also relies too much on the learning algorithm for optimization results.

From [3] and [4], ELM was introduced in as an implementation to SLFN. Furthermore, unlike backpropagation discussed in [5], ELM is not only faster but it also obtains better generalization performance. The algorithm for ELM is separated into two parts:

1. Training of ELM

Input variables, output and number of hidden nodes are needed to be defined at the start. *x* is the input matrix with *m* variables and *n* samples, *y* being the output with m samples and *k* as the number of hidden nodes.

$$x = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \cdots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}_{n \times m} \tag{7}$$

$$y = [y_{11} \quad \cdots \quad y_{1mm}]^T \tag{8}$$

After defining of inputs, next is randomizing weights *W* which connects the input nodes to hidden nodes, which is a combination of input variables and hidden nodes.

$$W = \begin{bmatrix} W_{11} & \cdots & W_{1k} \\ \vdots & \ddots & \vdots \\ W_{m1} & \cdots & W_{mk} \end{bmatrix} \tag{9}$$

Calculation of the hidden layer matrix H by multiplying x with W. and using activation function g to find the hidden layer output matrix H_{out}.

$$H = x\,W \tag{10}$$

$$H_{out} = g\,H \tag{11}$$

The estimated output $\hat{y}$, is connected to the hidden layer output matrix H_{out}, through and output layer weight ß. Using least squares solution, ß can be obtain by using ${H_{out}}^{\dagger}$ which is the Moore-Penrose generalized inverse of H_{out}.

$$ß = {H_{out}}^{\dagger}.y \tag{12}$$

$$\hat{y}, = H_{out}.ß \tag{13}$$

2. Testing of ELM

After the system is being trained by the inputs, there is a need to test the performance of the system. Thus using w and ß obtained in the training phase, repeat (7 to 13, excluding 9 and 12) with the new input variables and outputs.

III. PROPOSED METHODOLOGY

A. *Overview of Proposed Methodology*

The overview of the proposed methodology is shown in Fig. 2. SOM algorithm is done with the aid of MATLAB-GUI [6].

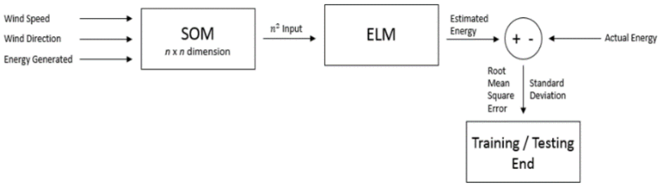

Fig. 2: Proposed Methodology

B. *Preprocessing of Data*

The dataset is acquired from [7] from 1st December 2015 to 1st January 2016 and are sampled at an interval of 10 minutes. These include the wind speed (m/s), wind direction, and energy generated (kWh).

C. *Activation Function*

When implementing activation functions in neural network to learn, the updating of contributions depends on the slope of the activation function. Is this paper, the chosen activation functions are log sigmoid (soft step), $g(x) = (1 + e^{-x})^{-1}$ and hyperbolic tangent, $(x) = (e^{x} - e^{-x})(e^{x} + e^{-x})^{-1}$.

D. *Performance Measure*

The performance of the proposed methodology is measured by the root mean square error (RMSE) achieved on the testing set data over 30 repetitive simulations.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{16}$$

where, $\hat{y}_i$ is the estimated output of the model, y_i is the actual output, and n is the total number of samples.

IV. SIMULATION AND RESULTS

The simulation is done by running MATLAB on a single core of a 64-bit system ASUS personal laptop running on Windows 8.1 with dual core 2.3 GHz CPU and 8.0 GB RAM. Before optimal parameters are established, and initial simulation is done using random parameters shown in Table III. After obtaining RMSE, the simulation is repeated 30 repetitions to check how much error deviates from each simulation. Different simulations will be tested in order to obtain the optimal parameters for forecasting of wind energy generation.

TABLE III. INITIAL PARAMETERS PROPOSED METHODOLOGY

Dataset	**Parameters**
Days for Training	2
Days for Testing	1
Activation Function	Log-Sigmoid
Size of SOM	10-by-10
No. Of Hidden Nodes	100

A. *Simulation 1: Number of Hidden Nodes*

The initial simulation used 2 days of dataset for training and 1 day for testing for short-term forecast of 4 hours. A total of up to 29 days of training and 1 day of testing will be simulated to acquire the optimal size of dataset for forecasting horizons. The simulations and results are shown in Table IV and Fig. 3.

TABLE IV. SIMULATION 1 BEST PERFORMANCE RESULT

Training	**Testing**	**RMSE Test**	**STD Test**
1	1	558.64	7.81
7	**1**	**277.30**	**5.84**
14	1	809.94	10.29

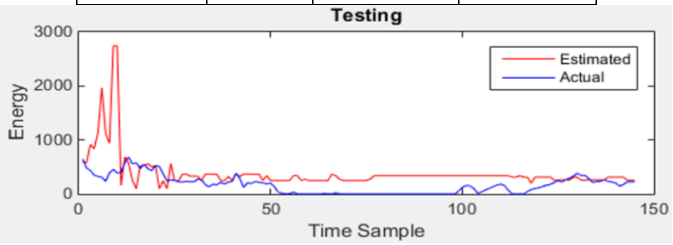

Fig. 3: Predicted Energy (kWh) in Simulation 1

For simulation with an equal amount of training data, the parameters estimates have greater variance, while for simulations with less training data, the performance statistic will have greater variance. Thus, out of the 29 simulations conducted, the simulation which yields the best result, has an approximate of 70:30 split.

B. *Simulation 2: Size of Dimension in SOM*

This simulation is allowing the network to run a total of 100 different patterns of SOM dimension ranging from 1-by-1 to 10-by-10. The size of dimension in SOM which yields the best results is 7-by-7 shown in Table V and Fig. 4.

TABLE V. SIMULATION 2 BEST PERFORMANCE RESULT

Dimension	**No. of Input Variables**	**RMSE Test**	**STD Test**
1-by-1	1	476.20	12.52
7-by-7	**49**	**238.76**	**11.36**
10-by-10	100	331.18	13.32

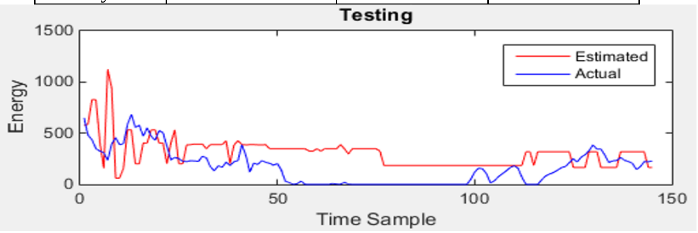

Fig. 4: Predicted Energy (kWh) in Simulation 2

Comparing the results, it can be concluded that increasing of input variables does not necessarily means that the system may generate a better performance as “overfitting” may occur. In addition “underfitting” also occurs when there is a lack in input variables.

C. Simulation 3: Number of Hidden Nodes in Hidden Layer

The hidden layer must be wide enough for the forecasting to be accurate, and the number of hidden nodes in the hidden layer defines the width [8]. 1 to 100 hidden nodes are set for simulation 3 to determine the optimal number of hidden nodes. The simulations and results are shown in Table VI and Fig. 5.

TABLE VI. SIMULATION 3 BEST PERFORMANCE RESULT

No. Hidden Nodes	RMSE Test	STD Test
2	480.07	15.33
3	414.97	16.87
70	**225.53**	**14.96**

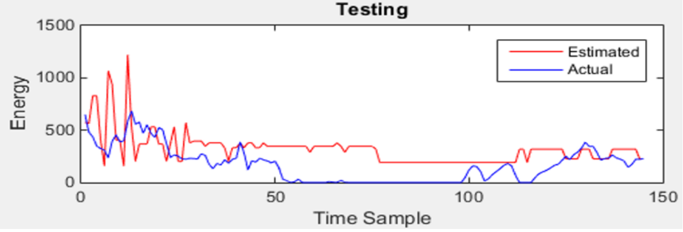

Fig. 5: Predicted Energy (kWh) in Simulation 3

In order to prevent “over-fitting” [9], the optimal number of hidden nodes for best performance has been trial and it is observed that the number of hidden neurons is about $n = \log(T)$, where T is the number of training samples inputted into the system. Using the formula, the estimated theoretical optimal number of hidden nodes should be 2 to 3 hidden nodes. However the simulation results proved that number of hidden nodes set to 70 yields the best results which shows that theoretical approach and practical approach differs.

D. Simulation 4: Comparing Performance of SOM-ELM with ELM

In this section, the optimal parameters obtained from the previous simulations will be used to compare with ELM algorithm to determine which algorithm is more feasible. Since the dataset only has three input variables, the dimensions of SOM will be changed to either 1-by-3 or 3-by-1 for fair comparison. Table VII shows the comparison between the three cases.

TABLE VII. COMPARISION OF PROPOSED METHODOLOGY

Algorithm	Dimension of Map	No. of Input Variables	RMSE Test
ELM	-	3	660.92
SOM-ELM	3-by-1	3	356.22
SOM-ELM	1-by-3	3	356.43

V. DISCUSSION

The simulations carried out have shown improvement of the RMSE from the initial test results. Through the three simulations, there is no standard procedure of fixing the parameters that can be followed to achieve the best results for wind energy generation. However, upon comparing the proposed methodology of SOM-ELM with just ELM, it is proven that SOM-ELM yields the better results between the two. Since SOM clusters the data into maps which causes the system to improve the forecast of wind energy generation. One technical merit of the work is the versatility of the system. As long as there are sufficient data, other renewable sources can also be forecasted. However, accuracy can slowly be improved by increasing variables and parameters which is also one of the limitations of this project.

VI. CONCLUSION

This work investigates the forecasting of wind energy generation using SOMs and ELM training algorithm. Firstly a forecasting model has been successfully setup using MATLAB software. Secondly efficiency of different parameters has been analyzed for the best result. Lastly, a comparison between the performance of SOM-ELM and ELM has been discussed. The model is evaluated using RMSE. Future works includes obtaining of a more detailed dataset which includes more parameters, integrating other ANN techniques to determine a more accurate forecasting algorithm and using other software to test the forecasting algorithm.

REFERENCES

[1] C. Maschio and D. J. S. , "Bayesian history matching using artificial neural network and Markov Chain Monte Carlo," *Journal of Petroleum Science and Engineering*, vol. 123, pp. 62-71, 2014.

[2] T. Heskes, "Self-Organizing Maps, Vector Quntization, and Mixture Modelling," *IEEE transactions on neural networks ,* vol. 12(6), pp. 1299-1305, 2001.

[3] C.-Y. Liu, C. Chen, C.-T. Chang and L.-M. Shih, "Single-hidden-layer feed-forward quantum neural network based on Grover learning," *Neural networks : the official journal of the International Neural Network Society*, vol. 45, pp. 144-150, 2013

[4] T. T. Teo, T. Logenthiran and W. L. Woo, "Forecasting of photovoltaic power using extreme learning machine," in *2015 IEEE Innovative Smart Grid Technologies*, Asia, 2015..

[5] C.-S. Chen, B. P.-T. Chen, F. N.-F. Chou and C.-C. Yang, "Development and application of a decision group Back-Propagation Neural Network for flood forecasting," *Journal of Hydrology*, vol. 385(1), pp. 173-182, 2010.

[6] S. H. Oudjana, A. Hellal, and I. H. Mahamed, "Short term photovoltaic power generation forecasting using neural network," *11th International Conference on Environment and Electrical Engineering (EEEIC)*, pp. 706-711, 2012.

[7] "Real Time Data," Sotavento Galicia, [Online]. Available: http://www.sotaventogalicia.com/en/real-time-data/historical.

[8] W. Zou, Y. Li and A. Tang, "Effects of the number of hidden nodes used in a structured-based neural network on the reliability of image classificatio," *Neural Computing and Applications*, vol. 18(3), pp. 249-260, 2009.

[9] N. Wanas, G. Auda, M. Kamel and F. Karray, "On the optimal number of hidden nodes in a neural network," in *Conference Proceedings*, 1998.