COMPARISON OF LAVENBERG-MARQUARDT, SCALED CONJUGATE GRADIENT AND BAYESIAN REGULARIZATION BACKPROPAGATION ALGORITHMS FOR MULTISTEP AHEAD WIND SPEED FORECASTING USING MULTILAYER PERCEPTRON FEEDFORWARD NEURAL NETWORK

Dissertation in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE WITH A MAJOR IN ENERGY TECHNOLOGY WITH FOCUS ON WIND POWER

UPPSALA
UNIVERSITET

Uppsala University
Department of Earth Sciences, Campus Gotland

Orkhan Baghirli

June 2015

COMPARISON OF LAVENBERG-MARQUARDT, SCALED CONJUGATE GRADIENT AND BAYESIAN REGULARIZATION BACKPROPAGATION ALGORITHMS FOR MULTISTEP AHEAD WIND SPEED FORECASTING USING MULTILAYER PERCEPTRON FEEDFORWARD NEURAL NETWORK

Dissertation in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE WITH A MAJOR IN ENERGY TECHNOLOGY WITH FOCUS ON WIND POWER

Uppsala University
Department of Earth Sciences, Campus Gotland

Approved by:

Supervisor, Prof. Jens N. Sørensen

Examiner, Assoc. Prof. Simon-Philippe Breton

June 2015

ABSTRACT

Wind speed forecasting is critical for wind energy conversion systems since it greatly influences the issues such as scheduling of the power systems, and dynamic control of the wind turbines. Also, it plays an essential role for siting, sizing and improving the efficiency of wind power generation systems. Due to volatile and non-stationary nature of wind speed time series, wind speed forecasting has been proven to be a challenging task that requires adamant care and caution. There are several state-of-the-art methods, i.e., numerical weather prediction (NWP), statistical, and hybrid models, developed for this purpose. Recent studies show that artificial neural networks (ANNs) are also capable of wind speed forecasting to a great extent.

In this paper, 3-layer perceptron feedforward neural network is employed for comparison of three different training algorithms, i.e., Lavenberg-Marquardt (LM), Scaled Conjugate Gradient (SCG) and Bayesian Regularization (BR) backpropagation algorithms, in the view of their ability to perform 12 multistep ahead monthly wind speed forecasting. Horizontal wind speed, absolute air temperature, atmospheric pressure and relative humidity data collected between November 1995 - June 2003 and July 2007 – April 2015 for city of Roskilde, Denmark is used for training, validation and testing of the network model. The performed experiment shows that for 12 multistep ahead wind speed forecasting, SCG algorithm has obvious preference in terms of prediction accuracy with mean absolute percentage error (MAPE) of 3.717%, followed by LM and BR algorithms with MAPE of 4.311% and 4.587% accordingly. As a result, within the scope of this study, SCG algorithm is found to be more suitable to build a multistep ahead wind speed forecasting model.

Key words: Multistep ahead forecast, wind speed forecasting, backpropagation algorithms, neural networks.

ACKNOWLEDGEMENTS

The timely and successful completion of this research paper could hardly be possible without the helps and supports from a lot of individuals. I will take this opportunity to thank all of them who helped me either directly or indirectly during this important work.

First of all, I would like to thank Swedish Institute scholarship for funding my studies here at Uppsala University. I wish to express my sincere gratitude and due respect to my supervisor Professor Jens N. Sørensen. I am immensely grateful to him for his valuable guidance, continuous encouragements and positive supports, which helped me a lot during the period of my work.

I also express whole hearted thanks to my family, friends and classmates for their care and moral supports. The moments, I enjoyed with them during my time here in Sweden will always remain as a happy memory throughout my life.

I am also thankful to entire faculty and staff of Wind Power Project Management (WPPM) for their help during the course of my work. Lastly, I would like to congratulate and thank myself for my courage and hard work.

NOMENCLATURE

ACF	Autocorrelation function
ANN	Artificial neural network
ARIMA	Autoregressive integrated moving average
BMLP	Bridged Multilayer Perceptron
BP	Backpropagation
BR	Bayesian regularization
ERNN	Elman recurrent neural network
FFNN	Feed forward neural network
FCC	Fully connected Cascade
MLP	Multilayer Perceptron
MSE	Mean squared errors
MAE	Mean absolute error
MAPE	Mean average percentage errors
NARX	Nonlinear autoregressive
NWP	Numeric weather prediction
LM	Lavenberg-Marquardt
LNNTD	Linear neural network with time delay
LVQ	Learning vector quantization
RB	Resilient Backpropagation
RMSE	Root mean square errors
RNN	Recurrent neural network
EnKF	Ensemble Kalman Filter
SCG	Scaled conjugate gradient

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

Wind is a clean, ergonomic, sustainable and cost-effective alternative energy source to conventional fossil fueled power generation. Increasing level of wind power penetration into the electric grid requires accurate wind speed forecasting methods for the effective and efficient management of wind farms. Wind speed forecasting 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. For instance, long-term wind speed prediction is vital for siting and sizing of wind power applications, whereas short-term forecasting of wind speed is important for improving the efficiency of wind power generation systems (Li & Shi, 2010). Although wind energy may not be dispatched, related impact cost of wind power development can be substantially reduced if the wind energy can be scheduled using accurate wind forecasting (Wu & Hong, 2007). However, due to its intermittent and non-stationary nature, wind speed can be very challenging to predict (Lodge & Yu, 2014).

Wind speed is affected by large-scale atmospheric conditions and the morphology of the surface landscape. Several state-of-the-art techniques have been identified for wind speed forecasting. These techniques can be cataloged into numeric weather prediction (NWP) methods, statistical methods, methods based upon artificial neural networks (ANNs), and hybrid approaches. NWP methods could be the most accurate technique for short-term forecasting. However, in general, statistical, ANN methods, or several advanced hybrid methods based on observations perform more accurately over the very short-term forecast range (Wu & Hong, 2007). Averaging the original data over a longer time interval, ANNs are capable of making precise long-term predictions as well.

Statistical methods provide relatively inexpensive statistical forecasting models that do not require any data beyond historical wind power generation data. However, the accuracy of the prediction for these models drops significantly when the time horizon is extended (Saroha & Aggarwal, 2014). The advantage of the ANN is to learn the relationship between inputs and outputs by a non-statistical approach. These ANN-based methodologies do not require any predefined mathematical models. If same or similar patterns are met, ANNs come up with a result with minimum errors (Wu & Hong, 2007). In this paper, ANN method is employed for wind speed estimations.

Artificial neural network method has been tested on wind speed forecasting through different experimental set-ups in various research articles in the past years. In an earlier study, Ghanbarzadeh et al. (2009) used several input parameters such as air temperature, relative humidity and vapor pressure data during 1993-2004 for city of Manjil, Iran to estimate the wind speed using ANN. The measured data between 1993-2003 were used for training purpose, while the data from 2004 were used for testing purpose. The results show considerable agreement between actual and predicted data; wherein mean absolute percentage error (MAPE) were found to be 10.78%. Lodge and Yu (2014) proposed a multilayer neural network model by using air temperature, pressure and historic wind speed measurements as input data and found that predicted and actual wind speed measurements are in strong agreement with root mean square error (RMSE) of 0.5526, which is 6.14% of the mean wind speed. Wang et al. (2004) developed an ANN based algorithm that summarizes short-term pattern and long-term trend in the wind speed data and uses a non-linear filter for the noise reduction. The results show that ANNs outperform persistence and autoregressive integrated moving average (ARIMA) models by 2.3% and 2.1% correspondingly. Also, Sharma and Lie (2012) proposed a hybrid technique for error reduction and improvement of basic ANN model by integrating the method of Ensemble Kalman Filter (EnKF) to correct the output of ANN to find the best estimate of wind speed. The results show that the error can be reduced and very good accuracy can be obtained if this hybrid model is used for the prediction of wind speed.

The objective of this research paper is to present a comprehensive comparison study on the application of different artificial neural network training algorithms for multistep ahead monthly wind speed forecasting. Three types of training algorithms, namely, Lavenberg-Marquardt (LM), Scaled Conjugate Gradient (SCG) and Bayesian Regularization (BR) backpropagation algorithms for a given multilayer perceptron (MLP) feedforward neural network are investigated. The performance is evaluated based on several statistical metrics, namely, mean absolute error (MAE), mean square error (MSE), and mean absolute percentage error (MAPE). For this purpose, Matlab is chosen as an experiment environment to perform the required computations and visualizations.

The remainder of the paper is organized as follows. In chapter 2, related work from literature is explored and required knowledge regarding different ANN architectures and training algorithms is briefly provided. In chapter 3, a multiphase methodological framework is constructed, data used in this paper with its statistical properties are summarized and explained, and proposed network architecture is described. In chapter 4, the simulation parameters and the results are presented. In chapter 5, analysis of the results is discussed. Lastly, in chapter 6, final conclusive remarks are drawn and potential future work is inspected.

CHAPTER 2. LITERATURE REVIEW

2.1 Introduction to ANN

Neural networks are the nonlinear parallel structure inspired by human brain system (Rojas, 1996). A more promising method for adaptive wind speed prediction is the use of artificial neural networks. Originally modeled from the biological central nervous system of human beings, an artificial neural network is a large-scale parallel-distributed information processing system that is composed of many inter-connected nonlinear computational units, i.e., neurons (Rojas, 1996). The network can perform many tasks such as function approximation, system identification, optimization, and adaptive control. A neural network based approach yields some valuable features over traditional methods, such as adaptive learning, distributed association, nonlinear mapping, as well as the ability to handle imprecise data. For wind speed prediction, a neural network model can be trained by taking a set of past measurement data. If there is a change in conditions, it can learn the change over time and adjust itself for more accurate predictions (Lodge & Yu, 2014). ANNs have been proved to be effective to simulate nonlinear systems. Hidden patterns, which could be independent of any mathematical models, can be found from the training data sets. If the same or similar patterns are met, ANNs come up with a result with minimum MSE (Wang et al., 2004). ANN maps the input vector into corresponding output vector and it is only imperative and other values need not be known. This makes ANNs very useful to mimic non-linear relationships without the need of any already existing models (Sharma & Lie, 2012).

2.2 Taxonomy of ANN architectures

Artificial neural network architectures can be divided into two categories such as supervised and unsupervised networks (Wilamowski, 2009). The supervised neural networks are trained to produce desired outputs in response to sample inputs, making them particularly well suited to model and control dynamic systems, classify noisy data, and predict future events. Some members of this family are 1) feedforward networks (FFNN): multilayer perceptron (MLP), bridged multilayer perceptron (BMLP), fully connected cascade (FCC), feedforward input-delay, linear network architectures 2) radial basis networks: generalized regression and probabilistic neural network architectures, 3) dynamic networks: nonlinear autoregressive (NARX), recurrent neural network (RNN), Elman, and Hopfield networks, and 4) learning vector quantization (LVQ) architectures. On the other hand, the unsupervised neural networks are trained by letting the network continually adjust itself to new inputs. They find relationships within data and can automatically define classification schemes. Members of this family include competitive layers and self-organizing maps. Souhaib Ben Taieb et al. (2012) gave a comparative review of existing strategies for multistep ahead forecasting and summarized the pros and cons of these strategies. According to this study, FFNN yields better result in terms of accuracy over RNN whereas compromising on the computation time. Afterwards, Saroha and Aggarwal (2014) presented a comparative analysis of different classes of ANNs for multistep ahead time-series forecasting of wind power. The three models, which have been used, are: linear neural network with time delay (LNNTD), FFNN and Elman recurrent neural network (ERNN). The results showed that for one step ahead forecasting, FFNN outperformed the others, whereas for multistep ahead forecasting LNNTD showed slightly better performance. Additionally, the same paper also mentions that FFNN is one of the most used, intuitional, and promising yet relatively less complex technique to use for wind speed time-series forecasting. In this paper, MLP FFNN architecture has been employed.

2.3 FFNN architecture

Rojas (1996) provided a comprehensive review of FFNN architecture and he stated that feedforward backpropagation network is one of the most popular techniques in the field of ANNs. A MLP is a FFNN model that maps input data onto output data. The MLP consists of three or more layers with each layer fully connected to the next one. The multilayered architectures are those where set of computing N units is subdivided into L subsets N1, N2, . . . N_L in such a way that only connections from units in N1 go to units in N2, and from units in N2 to units in N3, etc. The subsets N_i are called the layers of the network. The set of input nodes is called the input layer, and the set of output units is called the output layer. All other layers in between are called the hidden layers. A generic MLP architecture is illustrated in Figure 2.1.

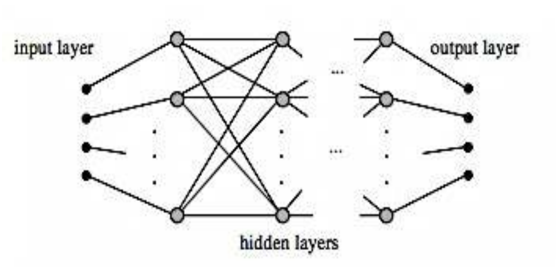

Figure 2.1: A generic architecture of MLP

Source: Rojas, 1996

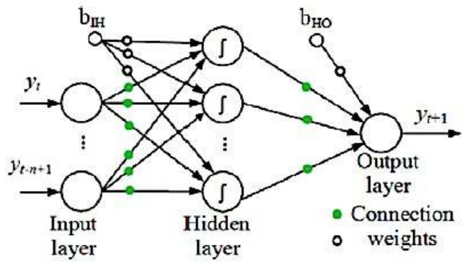

Figure 2.2: Topology of 3-layer FFNN

Source: Li & Shi, 2010

The source nodes in the input layer of the network, which are just the entry points for information into the network, do not perform any computation, yet supply respective elements of the activation pattern or input vector, which constitute the input signals applied to the neurons in the hidden layer. The output signals of the hidden layer are used as inputs to the output layer. The output signals of the neurons in the output layer of the network constitute the overall response of the network to the activation patterns applied by the input layer neurons.

A three-layer FFNN can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer (Saroha & Aggarwal, 2014). Figure 2.2 illustrates a FFNN with n input neurons, m hidden neurons and one output neuron. Each computing unit collects the information from n input lines with an integration function. The integration function is sum of the inputs. For j^{th} neuron of the network, the integration function can be described as in (2.1), where ω_{ij} is the connection weight from input i to hidden node j, y_i is the ith input with y_0 being the bias b_{IH}(with weight $\omega_{0j} = 1$). The total excitation computed in this way is then evaluated using a transfer function. Many kinds of transfer functions have been proposed in literature and one of the most popular hidden layer transfer functions is the hyperbolic tangent sigmoid function (2.2), denoted as f_H. Since learning algorithms require computation of the Jacobian (first order partial derivatives) and/or Hessian (squared second order partial derivatives) matrices of the network function, the continuity and differentiability of the transfer function must be guaranteed (2.3). Output of the jth neuron in the hidden layer wherein tangent sigmoid function is employed varies between -1 and 1 (2.4) and it serves as an input for the output neuron k (here k=1) in the output layer of the network (Rojas, 1996). Output of the output neuron can be calculated as shown in (2.5), where f_o is the output layer transfer function, usually a line function, ω_{jk} is the connection weights from hidden node j to an output neuron, z_j is the corresponding output of the jth neuron in the hidden layer with z_0 being the bias b_{HO}(with weight $\omega_{0k} = 1$).

$$net_j = \sum_{i=0}^{n} \omega_{ij} y_i \; (i = 0,1, \dots, n; j = 1, \dots, m) \tag{2.1}$$

$$f_H(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{2.2}$$

$$\frac{d}{dx} f_H(x) = 2f_H(x)(1 - f_H(x)) \tag{2.3}$$

$$-1 < z_j = f_H\left(net_j\right) = \frac{1-e^{-\left(\sum_{i=0}^{n} \omega_{ij} y_i\right)}}{1+e^{-\left(\sum_{i=0}^{n} \omega_{ij} y_i\right)}} < 1 \tag{2.4}$$

$$o_{out} = f_o\left(\sum_{j=0}^{m} \omega_{jk} z_j \; (j = 1, 2 \dots, m)\right) \tag{2.5}$$

The weights of the edges are real numbers selected at random. When the input pattern y_i from the training set is presented to this network, it produces an output $o_{out,i}$ different in general from the target t_i. The objective is to make $o_{out,i}$ and t_i identical for i = 1 . . . p, where p is the ordered pairs of n- and k-dimensional input and output vectors (here k=1), by using a learning algorithm to minimize the error function of the network (2.6).

$$E = \frac{1}{2} \sum_{i=1}^{p} \left\| o_{out,i} - t_i \right\|^2 \tag{2.6}$$

The adjustable parameters to minimize the error function are the interconnection weights and bias points. Thus, various iterative training algorithms can be utilized to minimize E. The three-layer ANN used in this study contains only one hidden layer. Multilayer ANN can have more than one hidden layer, however theoretical works have shown that a single hidden layer is sufficient for ANNs to approximate any complex non-linear function (Saroha & Aggarwal, 2014).

2.4 Taxonomy of Training Algorithms

Training is the process of determining the optimal weights and bias points of the ANN. This is done by defining the total error function between the network's output and the desired target and then minimizing it with respect to the weights. In this paper, we mainly consider the backpropagation (BP) training algorithms for FFNN. BP algorithms are the supervised learning method for MLP FFNN. The name refers to the backward propagation of error during the training of the network such that algorithms in this family use chain rule several times for the calculation of partial derivative of the network`s total error function with respect to weights. For this purpose, calculations begin at the output layer first and then propagate backward until each weight connection can be updated individually. There are different variations of BP training algorithms where each has certain advantages and disadvantages depending on the network architecture and complexity of the problem.

In a study conducted by Pan, Lee and Zhang (2013), BP algorithms have been categorized into six classes: 1) Adaptive Momentum 2) Self-adaptive learning rate 3) Resilient backpropagation (RB) 4) Conjugate gradient (CG) 5) Quasi-Newton 6) Bayesian regularization (BR). In the same paper, performances of these algorithms in the view of prediction accuracy, convergence speed and training time have been evaluated for the purpose of electricity load forecasting. Based on this study, it is concluded that BR algorithms perform well with an accuracy of 3.5% MAPE and it is preferred over other types of training algorithm. Due to its heavy processing load, it is also recommended that where the processing ability is limited, RB or CG can also be used to speed up the process and still acquire accurate results.

Wilamowski (2009) in his study of neural networks and learning algorithms compared different network architectures and training algorithms and concluded that with fewer number of neurons, the neural network should have better generalization abilities. If too many neurons are used, then the network can be overtrained on the training patterns, but it will fail on patterns never used in training. With fewer number of neurons, the network cannot be trained to very small errors, but it may produce much better results for new patterns.

Kisi and Uncuoglu (2005) investigated the use of three BP training algorithms, i.e., LM, CG and RB, for stream flow forecasting and determination of lateral stress in cohesionless soils. Based on the study results, although the LM algorithm is found to be faster and to have better performance than the other algorithms for the training set, the resilient backpropagation algorithm showed the best accuracy for the testing set.

In this present paper, LM, SCG and BR algorithms are employed and their performances pertaining to accuracy of multistep ahead monthly wind speed forecasting are compared.

2.5 Lavenberg-Marquardt backpropagation algorithm

LM algorithm was 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, then the Hessian matrix can be approximated and the gradient can be computer as in (2.7) (2.8) (Kisi & Uncuoglu, 2005; Hagan & Menhaj, 1994):

$$H = J^T J \tag{2.7}$$

$$g = J^T e \tag{2.8}$$

Where J is a Jacobian matrix, which contains first order derivatives of the network errors with respect to the weights and biases, e is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix. The LM algorithm uses this approximation to the Hessian matrix in the following Newton-like update (2.9), where x represents connection weights.

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \tag{2.9}$$

When the scalar μ is zero, this is just Newton`s method, using the approximate Hessian matrix. When $\mu$ is large, this becomes gradient descent with a small step size. Newton`s method is faster and more accurate near a minimum error, so the aim is to shift towards Newton`s method as quickly as possible. Thus, μ is decreased after each successful step and is increased only when a tentative step would increase the performance function. In this way, the performance function (also known as network error function) will always be reduced at each iteration. The LM optimization technique is more powerful than the conventional gradient descent techniques (Wilamowski, 2009; Hagan & Menhaj, 1994).

2.6 Scaled conjugate gradient backpropagation algorithm

The basic backpropagation algorithm adjusts the weights in the steepest descent direction, i.e., the most negative of the gradient. This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence (Hagan, Demuth & Beale, 1996). In the conjugate gradient algorithms a search is performed along such a direction which produces generally faster convergence than the steepest descent direction, while preserving the error minimization achieved in all previous steps (Kisi & Uncuoglu, 2005). This direction is called the conjugate direction. In most of the CG algorithms the step-size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size, which will minimize the performance function along that line. All of the CG algorithms start out by searching in the steepest descent direction at first iteration (2.10). Frequently, CG algorithms are used with line search. That means the step size is approximated with a line search technique, avoiding the calculation of the Hessian matrix to determine the optimal distance to move along the current search direction (2.11). Then the next search direction is determined so that it is conjugate to previous search direction (2.12). The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction (Hagan, Demuth & Beale, 1996).

$$p_o = -g_0 \tag{2.10}$$

$$x_{k+1} = x_k + \alpha_k g_k \tag{2.11}$$

$$p_k = -g_k + \beta_k p_{k-1} \tag{2.12}$$

The various versions of CG algorithms are distinguished by the manner in which the factor β_k is computed (Kisi & Uncuoglu, 2005).

It is also possible to use another approach in estimating the step size than the line search technique. The idea is to combine the model trust region approach, known from the LM algorithm with the CG approach. This approach is known as SCG and introduced to literature by Møller (1993). In this method, as it is described in (2.13), where s is the Hessian matrix approximation, E is the total error function and $E`$ is the gradient of E, scaling factors λ_k and σ_k are introduced to approximate the Hessian matrix and initialized by user at the beginning of the algorithm such that 0< λ_k< 10^{-6} and 0<σ_k<10^{-4}. For SCG, β_k factor calculation and direction of the new search can be shown as in (2.14) (2.15) (Møller, 1993):

$$s_k = \frac{E`(w_k+\sigma_k p_k)-E`(w_k)}{\sigma_k} + \lambda_k p_k \tag{2.13}$$

$$\beta_k = \frac{(|g_{k+1}|^2 - {g_{k+1}}^T g_k)}{{g_k}^T g_k} \tag{2.14}$$

$$p_{k+1} = -g_{k+1} + \beta_k p_k \tag{2.15}$$

Design parameters are updated at each iteration user independently, which is crucial for the success of the algorithm. This is a major advantage compared to the line search based algorithms.

2.7 Bayesian regularization backpropagation algorithm

BR is a training algorithm that updates the weights and bias values according to LM optimization (Foresee & Hagan, 1997; MacKay, 1992). It minimizes a combination of squared errors and weights, and then determines the correct combination so as to produce a network that generalizes well (Pan, Lee & Zhang, 2013). BR introduces network weights into the training objective function which is denoted as F(ω) in (2.16) and further explained by Yue, Songzheng & Tianshi (2011).

$$F(\omega) = \alpha E_\omega + \beta E_D \tag{2.16}$$

Where E_ω is the sum of the squared network weights and E_D is the sum of network errors. Both α and β are the objective function parameters. In the BR framework, the weights of the network are viewed as random variables, and then the distribution of the network weights and training set are considered as Gaussian distribution.

The α and β factors are defined using the Bayes' theorem. The Bayes' theorem relates two variables (or events), A and B, based on their prior (or marginal) probabilities and posterior (or conditional) probabilities as in (2.17) (Li & Shi, 2012):

$$\mathrm{P(A|B)} = \frac{\mathrm{P(B|A)P(A)}}{\mathrm{P(B)}} \tag{2.17}$$

Where P(A|B) is the posterior probability of A conditional on B, P(B|A) the prior of B conditional on A, and P(B) the non-zero prior probability of event B, which functions as a normalizing constant. In order to find the optimal weight space, objective function (2.16) needs to be minimized, which is the equivalent of maximizing the posterior probability function given as in (2.18):

$$P(\alpha, \beta | D, M) = \frac{P(D|\alpha,\beta,M)P(\alpha,\beta|M)}{P(D|M)} \tag{2.18}$$

Where α and β are the factors needed be to optimized, D is the weight distribution, M is the particular neural network architecture, $P(D|M)$ is the normalization factor, $P(\alpha, \beta|M)$ is the uniform prior density for the regularization parameters and $P(D|\alpha, \beta, M)$ is the likelihood function of D for given α, β, M. Maximizing the posterior function $P(\alpha, \beta|D, M)$ is equivalent of maximizing the likelihood function $P(D|\alpha, \beta, M)$. As a result of this process, optimum values for α and β for a given weight space are found. Afterwards, algorithm moves into LM phase where Hessian matrix calculations take place and updates the weight space in order to minimize the objective function. Then, if the convergence is not met, algorithm estimates new values for α and β and the whole procedure repeats itself until convergence is reached (Yue, Songzheng & Tianshi, 2011).

CHAPTER 3. METHODOLOGY AND DATA

3.1 Introduction

A multi-phase methodological framework is constructed in order to build a platform for the performance evaluation of three different backpropagation training algorithms, i.e., LM, SCG and BR. The proposed framework contains phases such as: 1) collecting the data 2) preprocessing the data 3) data transformation 4) processing the data and 5) postprocessing the data. A systematic methodological flow chart of the study is presented in Figure 3.1. An explicit description for each module and sub-module is given later in this section.

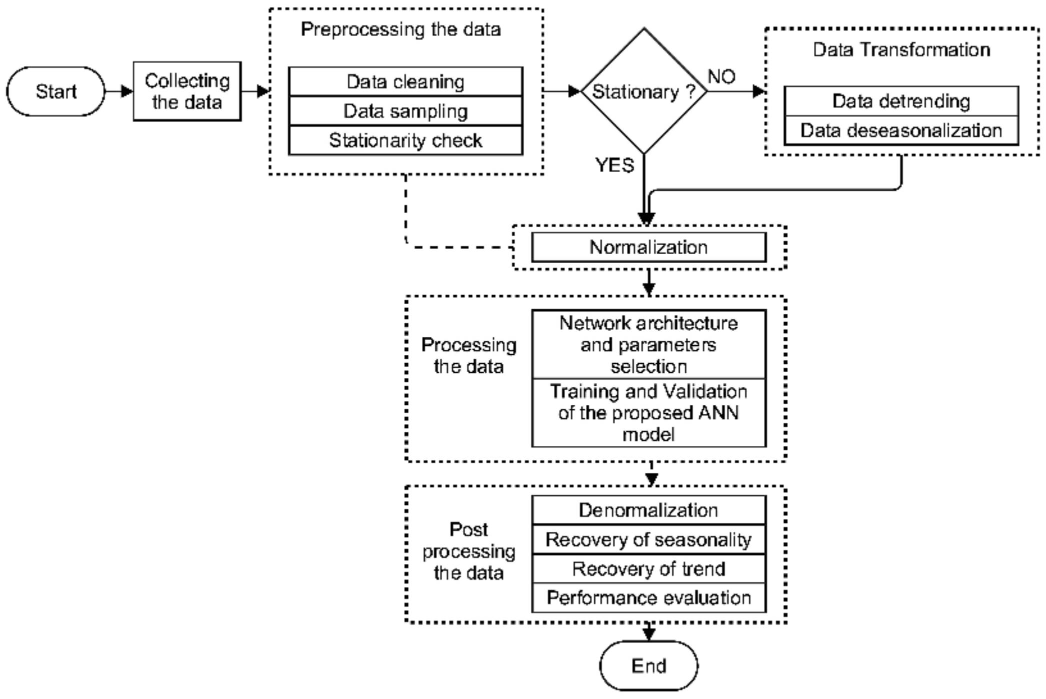

Figure 3.1: Multi-phase methodological framework

3.2 Collecting the data

For this study, 10 minute averaged horizontal wind speed, atmospheric pressure, relative humidity and absolute air temperature data taken from the met mast located at Risø Campus near Roskilde, Denmark between November 1995 - June 2003 and July 2007 – April 2015 have been used. Atmospheric pressure, air temperature and relative humidity constitute the input datasets and the purpose of the neural network is to use these input data to map the output data, which is the wind speed. Time-series diagrams and statistical properties of the original weather data, in terms of minimum, maximum, mean, standard deviation (std.), relative standard deviation (RStd.) and skewness values, are presented in Figure 3.2 and Table 3.1 accordingly. Standard deviation is a measure used to quantify the level of variations in a dataset. Relative standard deviation is the ratio of std. and mean values and it is shown in percentage. Skewness is a measure of the asymmetry of the probability distribution of a given dataset.

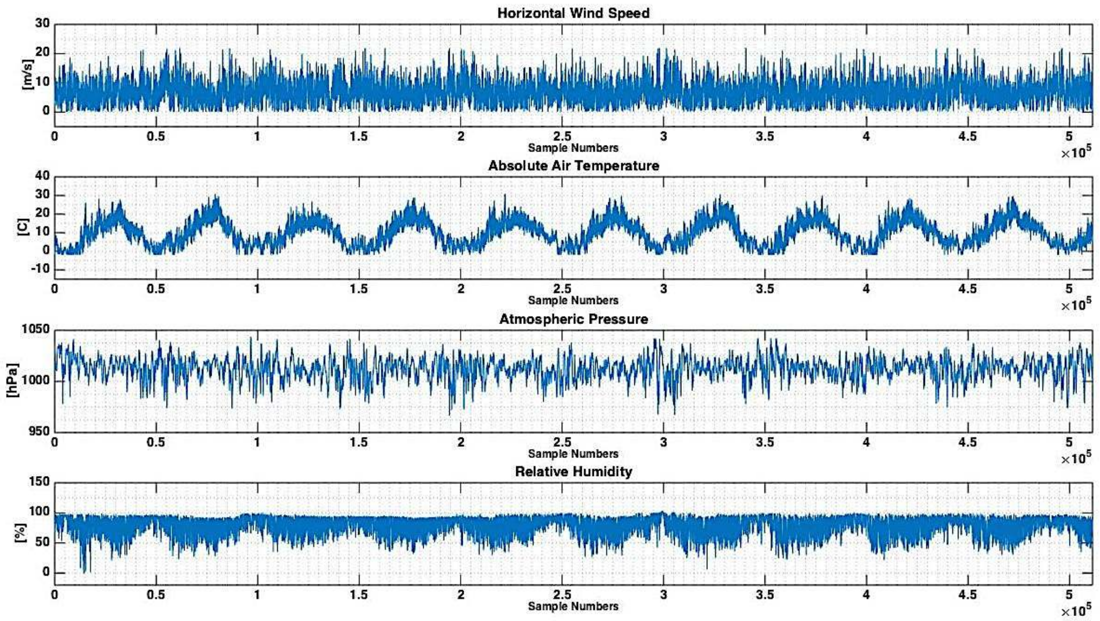

Figure 3.2: 10 minutes averaged original weather data

Table 3.1: Properties of the original weather data

Label	Unit	Altitude	Sample	Interval	Min.	Max.	Mean	Std.	RStd.	Skew
Wind Speed	m/s	76.6 m	511332	10 min.	0.8	21.98	7.00	3.25	46.43	0.57
Temperature	°C	2 m	511332	10 min.	-2.00	30.90	9.61	6.57	68.37	0.25
Pressure	hPa	2 m	511332	10 min.	966.5	1043.7	1012.7	10.4	1.03	-0.4
Humidity	%	2 m	511332	10 min.	-1.00	103.00	79.97	13.67	17.09	-0.9

3.3 Preprocessing the data

3.3.1 Data cleaning

Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a dataset. The anomalies observed within the provided datasets are removed and the resulting cleansed time-series for each of the weather features is now ready for data sampling.

3.3.2 Data sampling

In this study, future wind speed is forecasted in a monthly basis. For this reason, after months with incomplete records are removed from the cleaned datasets, the remaining datasets are sampled into monthly averages and presented in Figure 3.3. Statistical properties of monthly averaged datasets are listed in Table 3.2.

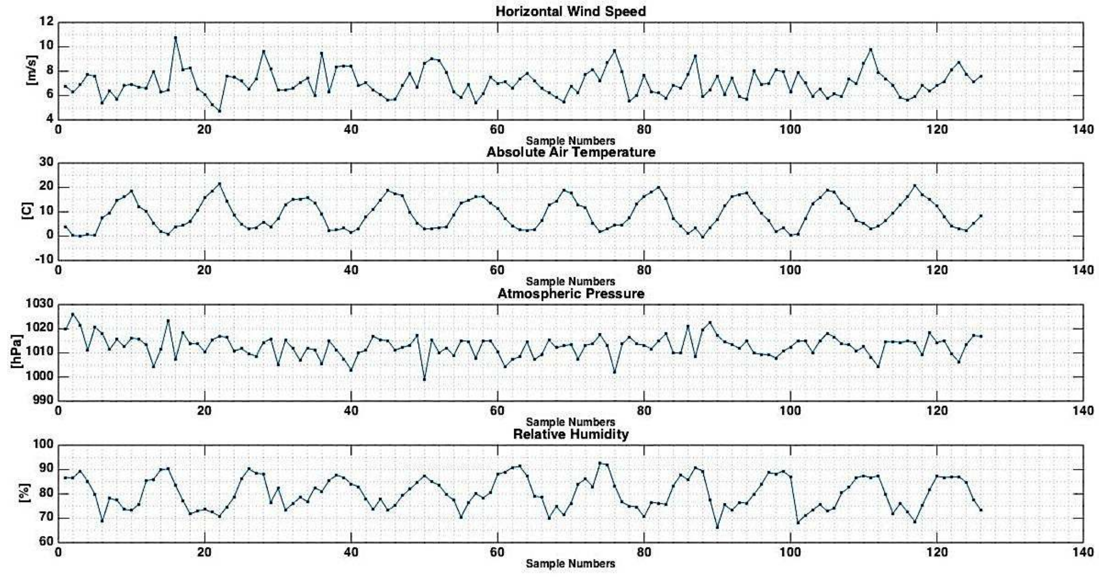

Figure 3.3: Monthly averaged weather data

Table 3.2: Statistical properties of the monthly averaged weather data

Data Label	Sample	Min.	Max.	Mean	Std.	RStd.[%]	Skew
Wind Speed	126	4.72	10.73	7.04	1.088	15.45	0.69
Temperature	126	-0.38	21.42	9.07	5.97	65.78	0.23
Pressure	126	999.05	1026.10	1012.82	4.40	0.43	-0.13
Humidity	126	66.315	92.63	80.417	6.44	8.01	-0.013

As it is seen from Table 3.2, absolute air temperature data suffers from high dispersion level whereas wind speed suffers from the degree of skewness. Therefore, datasets need to be further processed until they reach stationarity.

3.3.3 Stationarity check

Stationarity is another measure that characterizes the nature of a time-series, i.e., time-series with constant mean and standard deviation over time. The time-series forecast is based on the assumption that provided datasets are stationary (T. Kim, Oh, C. Kim & Do, 2004). Therefore, it is essential to verify the stationarity assumption of datasets before they are deployed into neural network. For this purpose, autocorrelation function (ACF), i.e., indicator of stability of a time-series, is utilized to check whether weather datasets used in this study are stationary. Corresponding results from ACF are provided in Figure 3.4.

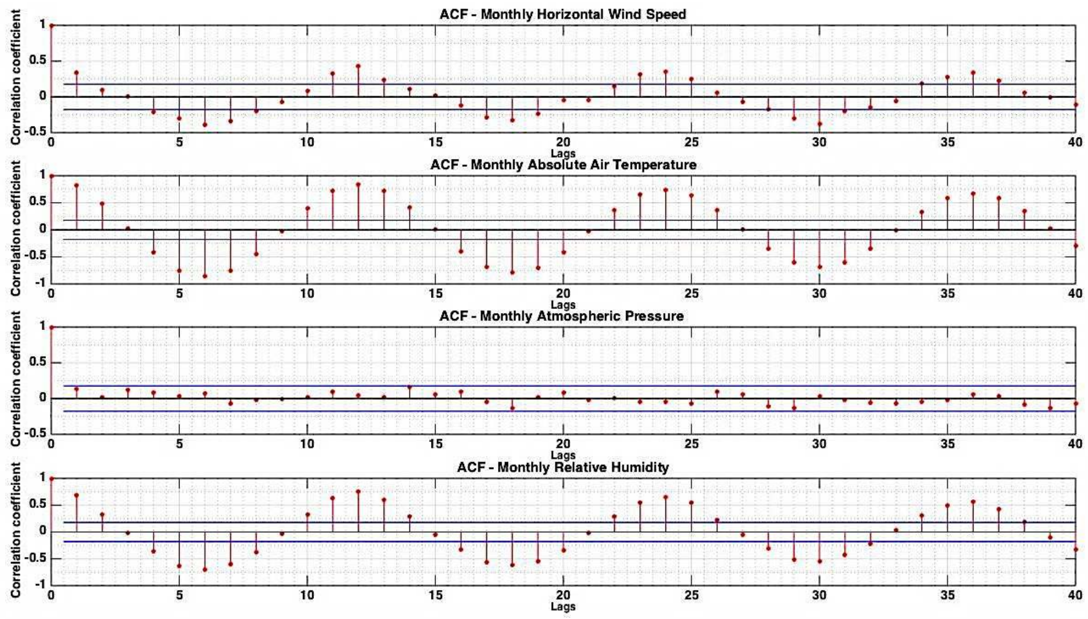

Figure 3.4: Autocorrelation function results for horizontal wind speed, absolute air temperature, atmospheric pressure and relative humidity datasets.

Figure 3.4 shows that horizontal wind speed, absolute air temperature and relative humidity datasets exhibit significantly large ACF values at the increasing lags (lags in this context mean months and number of months used in this test is 40, which is chosen arbitrarily), which do not diminish quickly. This indicates the non-stationarity of these datasets (Brockwell, 2003; Montgomery, Jennings & Kulahci, 2008). It is also clear from the ACF diagrams that, these datasets have a cyclic pattern with periodicity of 12 samples. On the other hand, pressure data follows rather stationary behavior with no obvious periodic fluctuations. In order to have better accuracy of forecast, non-stationary datasets need to be further processed until they are stationary.

3.3.4 Data transformation

The objective of the data transformation is to produce a series with no apparent deviations from stationarity, and in particular with no apparent trend or seasonality. The next step is to model the estimated noise sequence, i.e., the residuals obtained by estimating and subtracting the trend and seasonal components (Brockwell, 2013).

Classical decomposition model suggests that a time-series can be described as a sum of trend, seasonal and random noise components (3.1):

$$X_t = m_t + s_t + Y_t \tag{3.1}$$

Where X_t is the observation at time t, m_t is the trend component, s_t is the seasonal component, and Y_t is the random noise component.

3.3.5 Data detrending

Data detrending, in this context, means removing the mean trend from a time-series. All provided datasets including the atmospheric pressure (even it shows stationary behavior) are subject to this process. This means that each of the weather datasets is set to have a zero mean value and the corresponding statistical properties that are matter of particular interest, i.e., std. and skewness, of the detrended datasets stayed unchanged as expected. Data detrending is the preliminary stage for data deseasonalization.

3.3.6 Data deseasonalization

Many time-series exhibit cyclic variation known as seasonality. Seasonal variation is a component of a time-series, which is defined as the repetitive and predictable movement around the trend line. The objective of data deseasonalization is to eliminate these seasonal periodic variations from the detrended datasets in order to find a nonparametric relationship between irregularities (residuals, noise) exist in the input and output data, which is a challenging task (Montgomery, Jennings & Kulahci, 2008). For this purpose, the method of least squares is used to determine the best-fit line to data. To reflect the seasonal changes, Fourier series, i.e., sum of sine and cosine functions that describes a periodic signal, is used as a fit line model. The atmospheric pressure data is exempt from this stage since it did not show any obvious seasonal pattern. The remaining time-series have gone through this process and the corresponding results are provided in Figure 3.5 and Table 3.3.

Table 3.3: Fit lines and residual statistics

Label	Fit line	Std.	Skew
Wind Speed	$f(x) = -0.1264 - 0.1712 \cdot \cos(0.5236 \cdot x) + 1.01 \cdot \sin(0.5236 \cdot x)$	0.86	0.37
Temperature	$f(x) = 0.2471 + 1.132 \cdot \cos(0.5235 \cdot x) - 8.03 \cdot \sin(0.5235 \cdot x)$	1.60	0.086
Pressure	N/A	4.41	-0.13
Humidity	$f(x) = -0.1701 + 2.869 \cdot \cos(0.526 \cdot x) + 7.435 \cdot \sin(0.526 \cdot x)$	3.08	-0.55

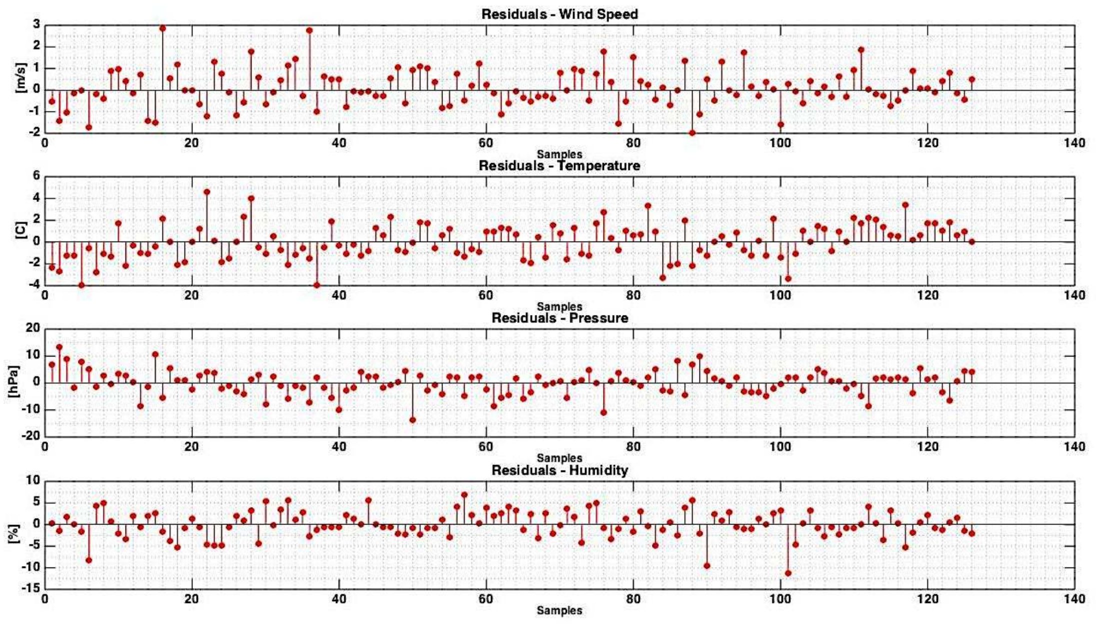

Figure 3.5: Weather data residuals after deseasonalization

3.3.7 Data normalization

To increase the efficiency of the network, all of the datasets are normalized. For this purpose, the feature scaling method described in (3.2) is adopted so as to bring all values into the range [-1,1]; where x is a value within an initial dataset before normalization takes place, x_N represents the normalized value, and x_{min} and x_{max} represent the minimum and maximum value of the initial dataset. The reason behind the normalization step is for that if input values as they appear in the initial datasets are fed into the neuron, integration process may yield very high results, which would cause the transfer function (tangent sigmoid) to exhibit low performance to resolve between small changes in input data and lose its sensitivity. Corresponding results of this stage with final statistical evaluation of datasets just before they are fed into the network are provided in Figure 3.6, Tables 3.4 and 3.5. Figure 3.7 shows the ACF of normalized weather data and explained later in this paper.

$$x_N = 2 \cdot \frac{x - x_{min}}{x_{max} - x_{min}} - 1 \tag{3.2}$$

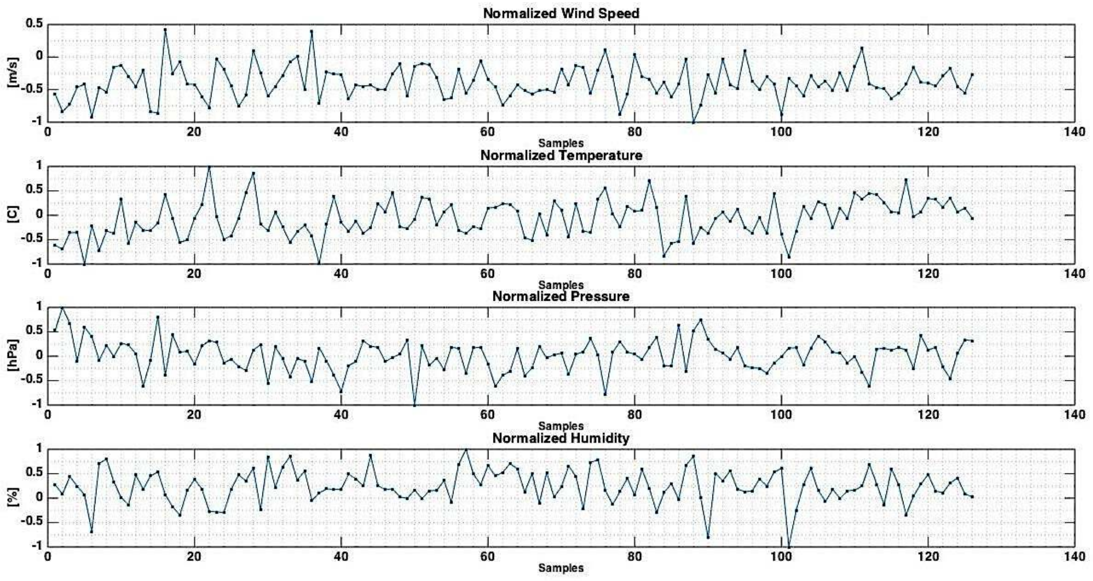

Figure 3.6: Normalized weather data

Table 3.4: Statistical evaluation of the normalized weather data

Data Label	Sample	Min.	Max.	Mean	Std.	Skew
Wind Speed	126	-1	1	-0.38	0.25	0.37
Temperature	126	-1	1	-0.07	0.37	0.08
Pressure	126	-1	1	0.02	0.32	-0.13
Humidity	126	-1	1	0.24	0.34	-0.55

Table 3.5: Evolutionary statistical properties of the weather datasets

	Original data		Sampled data		Deseasonalized		Normalized	
Label	Std.	Skew.	Std.	Skew.	Std.	Skew.	Std.	Skew.
Wind Speed	3.25	0.57	1.088	0.69	0.86	0.37	0.25	0.37
Temperature	6.57	0.25	5.97	0.23	1.60	0.086	0.37	0.086
Pressure	10.4	-0.4	4.40	-0.13	4.41	-0.13	0.32	-0.13
Humidity	13.67	-0.9	6.44	-0.013	3.08	-0.55	0.34	-0.55

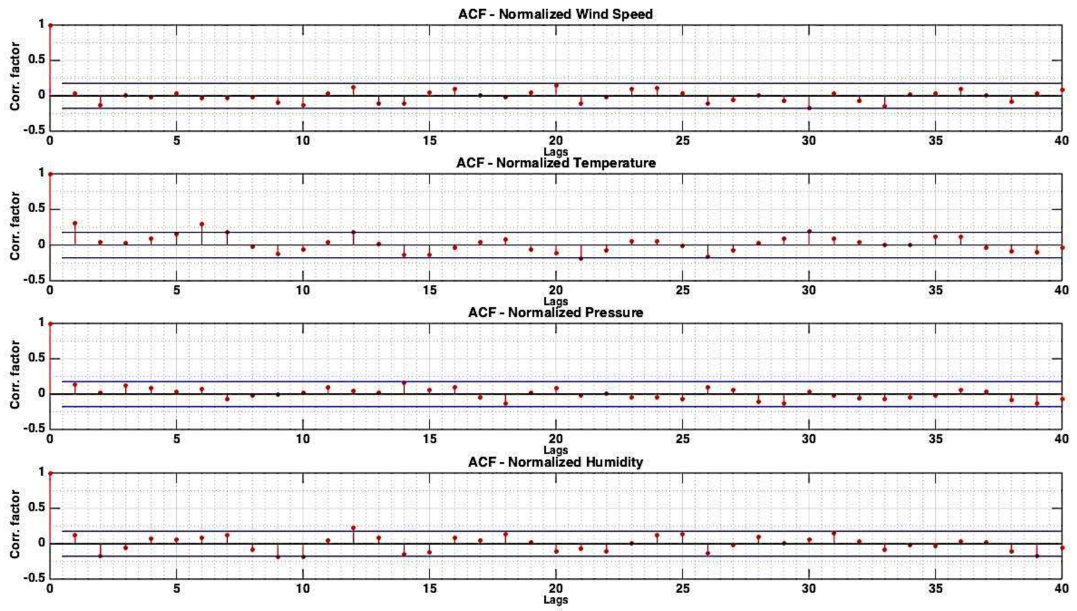

Figure 3.7: Autocorrelation function of the normalized weather data

3.4 Processing the data

3.4.1 Network architecture and parameters selection

In this study, three-layer FFNN architecture is employed. Matlab neural network toolbox is utilized to build the network model and then the corresponding Matlab code is generated and further developed to serve the purpose. The proposed FFNN architecture consists of 1 input, 1 hidden and 1 output layer. The number of neurons in the hidden layer is adjusted to give the best performance for each training algorithm used. The neurons in the hidden layer use the tangent sigmoid transfer function whereas the output neuron uses the pureline transfer function. The shorthand notation for this proposed network topology is 3-N-1 where 3 is the number of nodes in the input layer, N is the number of neurons in the hidden layer, and 1 is the number of neurons in the output layer. The proposed network model is provided in Figure 3.8.

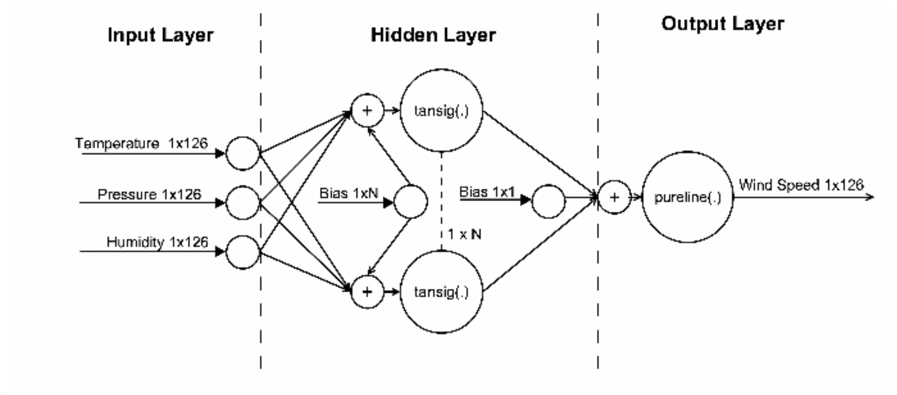

Figure 3.8: 3-layer perceptron FFNN architecture

3.4.2 Training and Validation of the proposed ANN model

The objective of training and validation stages is to generate an optimum weight space in order to establish the mapping of the extracted noise components from input and target datasets. After the neural network is built, previously normalized datasets are deployed into the network. There are total 126 samples for each weather feature. The datasets are divided into three subsets, i.e., training set, validation set, and testing set. During the training phase, training algorithms attempt to correct the randomly distributed initial weight space until the performance goal of the validation phase is archived or no further correction can be made after several consecutive iterations. The validation set is set up to avoid the overfitting on the training data, as an ANN without validation set is likely to be overfitted on the training data (T. Kim, Oh, C. Kim & Do, 2004). When overfitting occurs, the network loses its ability to find an underlying relationship between training and testing sets, rather focuses on the training set performance, which brings the testing set performance significantly lower. All subsets are composed of the three input vectors, i.e., absolute air temperature, atmospheric pressure and relative humidity, and one target/output vector, i.e., the horizontal wind speed data. The breakdown of samples for each subset is chosen arbitrarily such as: 90 samples for the training set, 24 samples for the validation set, and 12 samples (a year) for the testing set. Different training algorithms, i.e., LM, SCG and BR backpropagation algorithms, are used to train the network and the corresponding configuration parameters are as shown in Table 3.6. Thereafter, the trained network is used to forecast 1 to 12 multistep ahead target values in the testing set.

Table 3.6: Training algorithms configuration parameters

Configuration Parameters	**LM**	**SCG**	**BR**
Maximum number of epochs to train	1000	1000	1000
Performance goal	0	0	0
Maximum validation failures	2	2	2
Initial μ	0.001	N/A	0.005
μ decrease factor	0.1	N/A	0.1
μ increase factor	10	N/A	10
Maximum μ	1e10	N/A	1e10
σ_0	N/A	5e-5	N/A
λ_0	N/A	5e-7	N/A

3.5 Postprocessing the data

After the normalized datasets being processed by the proposed ANN model, generated network output is gone through the postprocessing procedure. This includes the denormalization of the network`s output, recovery of seasonality, recovery of trend and performance evaluation steps.

3.5.1 Denormalization

Before the datasets were fed into the network, it was normalized. Thereby, after all calculations are finalized, the output of neural network is denormalized using (3.3).

$$y_D = y_N(x_{max} - x_{\min}) + x_{min} \tag{3.3}$$

Where y_N is the network output, y_D is the denormalized network output, x_{max} and x_{min} are the normalization parameters of the input datasets as they are described in (3.2).

3.5.2 Recovery of seasonality

During the preprocessing phase, seasonalities have been modeled as a best fit of Fourier series and removed. Here the removed part, i.e., seasonal variations of wind speed, is added on to the denormalized network output.

3.5.3 Recovery of trend

Recovery of trend is the opposite of detrending and it is proceed as adding the removed mean value of the monthly wind speed data back to the denormalized network output.

3.5.4 Performance evaluation

After denormalization and recovery phases, actual and forecasted testing datasets are evaluated. The performances of three proposed training algorithms (LM, SCG and BR) are examined for 12 multistep ahead wind speed forecasting in the view of accuracy of predictions. For this purpose, statistical tools such as MAPE, MAE and MSE are employed to evaluate the measure of accuracy (3.4, 3.5 & 3.6).

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \times 100 \tag{3.4}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|F_t - A_t| \tag{3.5}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(F_t - A_t)^2 \tag{3.6}$$

Where F and A are the actual and forecasted values, n is the number of samples. The final comparisons of algorithms are evaluated using MAPE values since it is more intuitional measure with its percentage-wise analysis, however, ranking of the algorithms does not depend on the choice of utilized statistical tool. Therefore, experiment results are valid for all tools of accuracy measurements. The experiment results are presented in visual and tabular forms in the following section.

CHAPTER 4. APPLICATION OF THE METHODOLOGY AND RESULTS

The main objective of this study is to compare the different types of backpropagation training algorithms in their ability to build a wind speed forecasting model, and then select the most suitable training algorithm to train the model especially for multistep ahead monthly wind speed forecasting. The used datasets are based on the historical weather records for Roskilde, Denmark, which contains 126 monthly samples for each of the 4 features; those are horizontal wind speed, absolute air temperature, atmospheric pressure and relative humidity. The first 90 samples are the training dataset and the latter 24 samples are the validation dataset for building the neural network models with different training algorithms and the remaining 12 samples are the test dataset to evaluate the models. For this purpose, statistical measures such as MAPE, MSE and MAE are adopted. The proposed methodological framework is applied on this case study. The experiment results are presented as follows.

After numerous iterations and adjustments to the number of neurons in the hidden layer (for each trial at different number of hidden neurons, from 3 to 13, more than 30 cases are run with different initial weights), the best performance records for each algorithm are found to be as in Table 4.1 where H is the number of neurons in the hidden layer. Figures 4.1, 4.2 and 4.3 present the 1 to 12 multisteps ahead wind speed forecast and actual data, accordingly for the cases wherein LM, BR and SCG algorithms are employed. Figure 4.4 shows forecast and actual data for all algorithms superimposed on each other in order to visually interpret the accuracy of predictions. Afterwards, in Table 4.2, detailed statistical measurements of the prediction accuracy for each training algorithm versus forecast horizon are given for the quantitative evaluation of performance. Finally, in Figure 4.5, graphical representation of data in Table 4.2 is given for the reason of comparison between performances of different training algorithms used.

Table 4.1: Training and validation performance results

	LM [H=8]		SCG [H=9]		BR [H=8]	
	Training	Validation	Training	Validation	Training	Validation
MAPE [%]	8.523	8.057	9.186	7.959	10.488	8.448
MSE	0.582	0.471	0.664	0.493	0.828	0.570
MAE	59.106	57.667	63.365	57.572	74.451	60.580

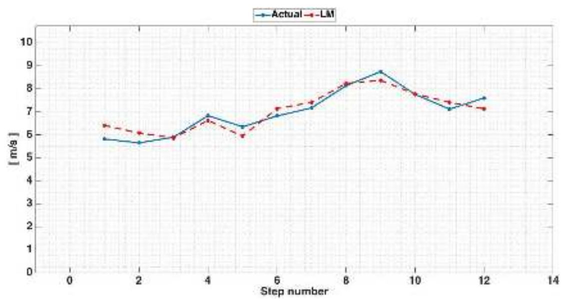

Figure 4.1: 1 to 12 steps ahead wind speed forecast using the LM algorithm

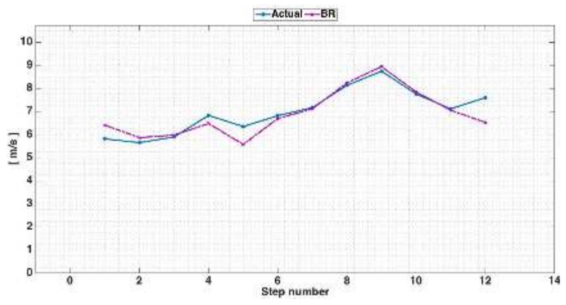

Figure 4.2: 1 to 12 steps ahead wind speed forecast using the BR algorithm

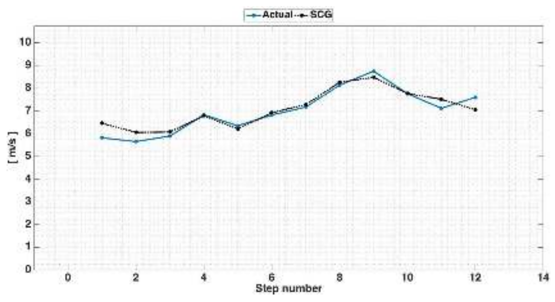

Figure 4.3: 1 to 12 steps ahead wind speed forecast using the SCG algorithm

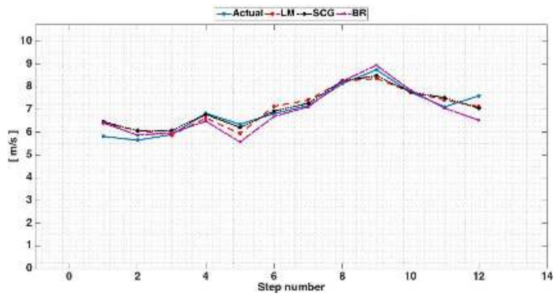

Figure 4.4: 1 to 12 steps ahead wind speed forecast results for all training algorithms

Table 4.2: Prediction results of ANN from step 1 to step 12

Steps	LM			SCG			BR		
	MAPE	MSE	MAE	MAPE	MSE	MAE	MAPE	MSE	MAE
1.	10.182	0.350	59.213	11.073	0.414	64.379	10.216	0.353	59.410
2.	8.861	0.266	50.902	9.117	0.289	52.422	7.049	0.200	40.670
3.	6.066	0.177	34.873	7.081	0.203	40.865	5.159	0.135	29.823
4.	5.299	0.143	31.274	5.493	0.153	31.892	5.156	0.132	31.162
5.	5.521	0.148	33.152	4.828	0.126	28.271	6.567	0.226	40.425
6.	5.375	0.140	32.912	4.275	0.107	25.270	5.772	0.191	35.736
7.	5.085	0.128	31.634	3.876	0.093	23.178	5.057	0.164	31.414
8.	4.603	0.113	28.929	3.583	0.083	21.839	4.589	0.145	28.823
9.	4.572	0.116	29.923	3.522	0.082	22.364	4.332	0.133	27.831
10.	4.134	0.105	27.078	3.181	0.073	20.219	4.001	0.120	25.897
11.	4.139	0.103	27.323	3.398	0.0815	21.983	3.718	0.110	24.074
12.	4.311	0.113	28.982	3.717	0.099	24.728	4.587	0.197	31.025

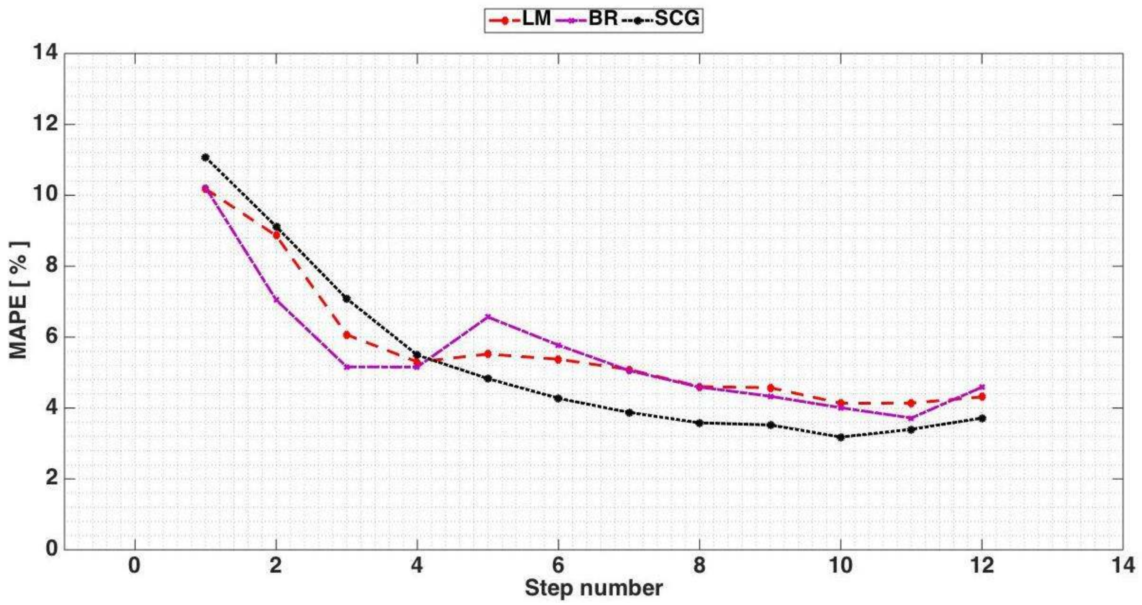

Figure 4.5: Performance evaluation

CHAPTER 5. DISCUSSION AND ANALYSIS

ANNs are sensitive to input and output datasets and their performance is highly dependent on the nature of time-series fed into the network. The non-stationary time-series with high values of standard deviation and skewness may lead to poor performance of ANNs (Kisi & Uncuoglu, 2005). As it is seen from Table 3.5 and Figure 3.7, after the preprocessing phase, the resulting statistical measures such as standard deviation and skewness are within the acceptable range and all datasets now exhibit the stationary behavior.

Table 4.1 shows that for each algorithm, validation set has higher prediction accuracy compared to the training set and therefore, no obvious overfitting on the training data is observed and high degree of generalization is achieved. The validation set performance reflects the accuracy of forecast in the testing set better than the training set performance and therefore, taking the results in Table 4.1 as a basis, it also can be claimed that SCG algorithm showed better generalization with MAPE of 7.959% followed by LM and BR algorithms with 8.057% and 8.448% values accordingly.

As it is seen from Table 4.2 and Figure 4.5, all algorithms start with low prediction accuracy. Up to step 4, BR outperforms LM and SCG. In step 4, all algorithms show very similar results. Step number 4 is a stage of transition where from step 4 to step 5, BR falls behind LM, and SCG shows the best result. Starting from step 6 up to step 10, performance of SCG gets even better whereas performances of LM and BR are almost the same. Step 10 is another transitory stage where performances of SCG and LM start to decrease and one step later, i.e., step 11, all algorithms exhibit decreasing performance.

Based on these results, it can be concluded that BR has better capability of a short-term forecast, however in the long run, it loses its accuracy and follows similar performance to that of LM. On the other hand, SCG shows less preferable performance for a short-term forecast, however, in the long run, it yields the best results. Within the scope of this study, for 12 multistep ahead monthly wind speed forecast, SCG outperformed LM and BR by MAPE of 0.594% and 0.87% accordingly in terms of overall prediction accuracy.

In a similar study, conducted by Ghanbarzadeh et al. (2009), 3-layer FFNN with 12 neurons in the hidden layer using the LM algorithm yielded MAPE of 10.78% for the same length of forecasting horizon used in this study. However, this study shows that in conjunction with the proper usage of preprocessing, LM algorithm with 8 neurons in the hidden layer is capable of the better estimation with MAPE of 4.311%. Another similar study carried out by Kisi and Uncuoglu (2005) for the lateral stress prediction which used 3-layer FFNN, trained by LM and CGF (a similar variation of SCG) algorithms with 176 training and 88 testing samples and featured 5 input/output datasets yielded similar results with MAPE of 4.13% and 4.27% respectively.

Based on these present research findings, SCG algorithm in conjunction with 3-layer FFNN, which showed the best result with MAPE of 3.717% is found to be superior to cases where LM and BR algorithms correspondingly resulted in MAPE of 4.311% and 4.587% and therefore it is suggested to build a multistep ahead wind speed forecasting model using the SCG algorithm.

CHAPTER 6. CONCLUSIONS

Wind speed forecasting plays a significant role in spatial planning of wind farms, such that accurate wind speed predictions can substantially reduce the impact cost pertaining to wind power development. Therefore, wind speed forecasting has always attracted special attention from both academia and industry. Due to volatile and non-stationary nature of wind speed time-series, wind speed forecasting has been proved to be a challenging task that requires adamant care and caution. There are several state-of-the-art methods, i.e., numerical weather prediction (NWP), statistical and hybrid models, developed for this purpose. Recent studies show that artificial neural networks (ANNs) are also capable of wind speed forecasting to a great extent.

In this paper, the 3-layer perceptron FFNN is used as an ANN architecture and the accuracy-wise comparison of three different backpropagation training algorithms, i.e., LM, SCG and BR is investigated. A multi-phase methodological framework is constructed and applied in order to build a 12 multistep ahead monthly wind speed forecasting model. An input matrix that contains 114 out of total 126 monthly averaged samples of absolute air temperature, atmospheric pressure and relative humidity and an output vector with the same length that is composed of the corresponding horizontal wind speed data gathered between November 1995 - June 2003 and July 2007 – April 2015 for city of Roskilde, Denmark is used for training and validation purposes whereas the remaining 12 samples of monthly averaged input data is used so as to perform 12 multistep ahead wind speed prediction, i.e., an equivalent of a year.

The performed experiment shows that for 12 multistep ahead wind speed forecasting SCG algorithm has obvious preference in the view of prediction accuracy with MAPE of 3.717%, followed by LM and BR algorithms with MAPE of 4.311% and 4.587% accordingly and therefore it is suggested to build a wind speed forecasting model within the scope of this study.

A difficult task involved in the application of ANNs on wind speed forecasting includes choosing the suitable network architecture, training algorithm, and configuration parameters since they directly affect the accuracy of the predictions. MLP FFNN suffers from the limitation of static input output mapping and non-stationarity of time-series (Anbazhagan & Kumarappan, 2013). Furthermore, ANNs are sensitive to statistical properties of the input and output datasets and therefore their performance varies accordingly.

Therefore it is a tedious task to prepare the datasets to feed into the neural network and optimize the configuration parameters such as number of the neurons in the hidden layer. Hence, the findings of this research should not be generalized and further research that includes optimization of the network parameters, several datasets from different sources with different size and statistical properties together with more sophisticated data preprocessing & postprocessing methods should follow this study.

REFERENCES

Lodge, A.; Xiao-Hua Yu, "Short term wind speed prediction using artificial neural networks," Information Science and Technology (ICIST), 2014 4th IEEE International Conference on, vol., no., pp.539,542, 26-28 April 2014

Ghanbarzadeh, A.; Noghrehabadi, A.R.; Behrang, M.A.; Assareh, E., "Wind speed prediction based on simple meteorological data using artificial neural network," Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on , vol., no., pp.664,667, 23-26 June 2009

Yuan-Kang Wu; Jing-Shan Hong, "A literature review of wind forecasting technology in the world," Power Tech, 2007 IEEE Lausanne, vol., no., pp.504,509, 1-5 July 2007

Wang, X.; Sideratos, G.; Hatziargyriou, N.; Tsoukalas, L.H., "Wind speed forecasting for power system operational planning," Probabilistic Methods Applied to Power Systems, 2004 International Conference on , vol., no., pp.470,474, 16-16 Sept. 2004

Sharma, D.; Tek Tjing Lie, "Wind speed forecasting using hybrid ANN-Kalman Filter techniques," IPEC, 2012 Conference on Power & Energy , vol., no., pp.644,648, 12-14 Dec. 2012

Saroha, S.; Aggarwal, S.K., "Multi-step ahead forecasting of wind power by different class of neural networks," Engineering and Computational Sciences (RAECS), 2014 Recent Advances in , vol., no., pp.1,6, 6-8 March 2014

Souhaib Ben Taieb, Gianluca Bontempi, Amir F. Atiya, Antti Sorjamaa, A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition, Expert Systems with Applications, Volume 39, Issue 8, 15 June 2012, Pages 7067-7083

Rojas, Raul. Neural Networks: A Systematical Introduction. Vol. 1. Berlin: Springer, 1996. 1 vols.

Wilamowski, B.M., "Neural network architectures and learning algorithms," Industrial Electronics Magazine, IEEE , vol.3, no.4, pp.56,63, Dec. 2009

Xinxing Pan; Lee, B.; Chunrong Zhang, "A comparison of neural network backpropagation algorithms for electricity load forecasting," Intelligent Energy Systems (IWIES), 2013 IEEE International Workshop on , vol., no., pp.22,27, 14-14 Nov. 2013

Ozgur Kisi; Erdal Uncuoghlu, “Comparison of three backpropagation training algorithms for two case studies,” Indian Journal of Engineering&Materials Sciences, Vol. 12, October 2005, page 434-442

Hagan, M.T., and M. Menhaj, "Training feed-forward networks with the Marquardt algorithm," IEEE Transactions on Neural Networks, Vol. 5, No. 6, 1999, pp. 989–993, 1994.

Martin Fodslette Møller, “A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks,” Volume 6, Issue 4, 1993, Pages 525-533

Hagan, M.T., H.B. Demuth, and M.H. Beale, Neural Network Design, Boston, MA: PWS Publishing, 1996.

MacKay, Neural Computation, Vol. 4, No. 3, 1992, pp. 415–447

Foresee and Hagan, Proceedings of the International Joint Conference on Neural Networks, June, 1997

Zhao Yue; Zhao Songzheng; Liu Tianshi, "Bayesian regularization BP Neural Network model for predicting oil-gas drilling cost," Business Management and Electronic Information (BMEI), 2011 International Conference on , vol.2, no., pp.483,487, 13-15 May 2011

Gong Li, Jing Shi, “Applications of Bayesian methods in wind energy conversion systems”, Renewable Energy, Volume 43, July 2012, Pages 1-8

Gong Li, Jing Shi, On comparing three artificial neural networks for wind speed forecasting, Applied Energy, Volume 87, Issue 7, July 2010, Pages 2313-2320

Tae Yoon Kim, Kyong Joo Oh, Chiho Kim, Jong Doo Do, Artificial neural networks for non-stationary time series, Neurocomputing, Volume 61, October 2004, Pages 439-447, ISSN 0925-2312

P. J. Brockwell, R. A. Davis, “Introduction to Time Series and Forecasting”, 2nd edition, Springer Publication, March 2003

Douglas C. Montgomery, Cheryl L. Jennings, Murat Kulahci,“Introduction to Time Series Analysis and Forecasting”, Wiley Series In Probability And Statistics, 2008, ISBN 978-0-4 71-653974

S. Anbazhagan and N. Kumarappan, "Day-Ahead Deregulated Electricity Market Price Forecasting Using Recurrent Neural Network", IEEE Systems Journal, vol. 7, no. 4, pp. 866-872, 2013.