2009 Ninth International Conference on Intelligent Systems Design and Applications

Hyperbolic Tangent Basis Function Neural Networks Training by Hybrid Evolutionary Programming for Accurate Short-Term Wind Speed Prediction (*)

C. Hervás-Martínez[1], P.A. Gutiérrez[1], J.C. Fernández[1], S. Salcedo-Sanz[2], A. Portilla-Figueras[2], A. Pérez-Bellido[2] and L. Prieto[3]

[1]Department of Computer Science and Numerical Analysis,
University of Córdoba, Campus de Rabanales, 14071, Spain
{chervas, pagutierrez,jcfernandez}@uco.es,
[2]Department of Signal Theory and Communications.
Universidad de Alcalá, Campus Universitario, 28871, Spain
sancho.salcedo@uah.es
[3]Department of wind resource, Iberdrola Renovables Inc.

Abstract

This paper proposes a neural network model for wind speed prediction, a very important task in wind parks management. Currently, several physical-statistical and artificial intelligence (AI) wind speed prediction models are used to this end. A recently proposed hybrid model is based on hybridizations of global and mesoscale forecasting systems, with a final downscaling step using a multilayer perceptron (MLP). In this paper, we test an alternative neural model for this final step of downscaling, in which projection hyperbolic tangent units (HTUs) are used within feed forward neural networks. The architecture, weights and node typology of the HTU-based network are learnt using a hybrid evolutionary programming algorithm. This new methodology is tested over a real problem of wind speed forecasting, in which we show that our method is able to improve the performance of previous MLPs, obtaining an interpretable model of final regression for each turbine in the wind park.

1. Introduction

The use of alternative sources of energy such as wind and solar energy is becoming more and more important in developed countries, as an important factor to mitigate the impact of the current crisis and the impact of petroleum's high prices. One of the main problems that arise in wind power generation is the accurate forecasting of the power that will be injected in the distribution network: a good forecasting of the power produced is crucial for the management of wind parks. Recently, several models which hybridize weather forecasting models (global and mesoscale), and statistical techniques have been proposed in the literature [1]. The majority of these models use a global forecasting model and one or several mesoscale and local-scale models to obtain wind speed predictions at specific points of a wind park [2]. Moreover, in many cases, these systems use statistical down-scaling processes including auto-regressive models [3], artificial neural networks (ANNs) [4,5] or support vector machines [6] as a final step to improve the wind speed forecasting of the system. Two works [7, 8] have been presented in 2009, in which the authors propose the hybridization of a mesoscale model (MM5) [9] with neural networks to obtain a robust system for wind speed forecasting at wind parks in short-time horizons. Specifically, the authors use the predictions of a global forecasting model (Global Forecasting System from the National Center for Environmental Prediction, USA) [10], and some local data from atmospheric soundings as initial and boundary conditions for the MM5 model. The MM5 model performs an initial physical downscaling of the data from the global model, to obtain a prediction of the wind speed with better spatial resolution. The output of the MM5 model, together with other local variables, is processed by a neural network in order to obtain the wind speed prediction in each turbine of the park, by means of solving a regression problem.

Different types of ANNs are nowadays being used for regression purposes [11], including, among others: multilayer perceptron neural networks (MLPs) where the transfer functions are logistic or hyperbolic tangent functions, these last ones being referred in this paper as Hyperbolic Tangent Unit (HTU) basis functions; Radial Basis Function (RBF), General Regression

(*)This work has been partially supported by TIN 2008-06681-C06-03 project of the Spanish Ministerial Commission of Science and Technology (MICYT), FEDER funds, the P08-TIC-3745 project of the "Junta de Andalucía" (Spain) and CCG08-UAH/AMB-3993 project of the Universidad de Alcalá and Comunidad de Madrid (Spain).

Neural Networks (GRNN) proposed by Specht [12]; Product Unit Neural Networks (PUNNs) [13], etc. The performance of the complete forecasting system depend much on the specific network used, so different models have been tested in the literature [2,6,7]. An additional problem related to the application of ANN models is the selection of the most appropriate net architecture to be used. Classical neural network training algorithms assume a fixed architecture; however it is very difficult to establish beforehand the best structure of the network for a given problem. In the last few years, Evolutionary algorithms (EAs) [15], have demonstrated great accuracy in designing near optimal architectures, with different networks [14,16].

This paper investigates on the performance of hybrid evolutionary-based neural networks as final statistical down-scaling techniques in wind speed forecasting systems. Specifically, we present a hybrid evolutionary programming algorithm for automatically obtaining the structure and weights of a HTU neural network, and how this model can be inserted in the hybrid forecasting model described in [7], for predicting wind speed in several turbines of a wind park. The proposed method is compared to the previous method used in the system described in [7], in order to assess its performance. The paper is organized as follows: Section 2 introduces data regression by ANNs; Section 3 explains the proposed hybrid algorithms; Section 4 describes the experiments carried out; and Section 5 states the conclusions of the paper.

2. Data Regression by ANNs

Data regression is a mayor research topic in the area of function approximation. The regression problem involves determining the relationship between some response dependent variable y and a set of k independent variables $\mathbf{x} = (x_1, x_2, ..., x_k)$. The most common form of structural assumption is that the responses are assumed to be related to the predictor through some deterministic function f and some additive random error component ε, so that:

$$y = f(\mathbf{x}) + \varepsilon , \qquad (1)$$

where ε is a zero mean error distribution.

Our aim is to determine the f function so that we can uncover the true relationship between the response y at the predictor location **x**, given by $y = f(\mathbf{x})$. The true regression function f is unknown and we have no way of determining its analytic form exactly, even if one actually exists. We must content ourselves with finding approximations to it which are close to the truth. To do this we must make use of the observed training dataset, D, which consist of n observed responses at some known predictor locations so $D = \{\mathbf{x}_i, y_i\}$ for $i = 1, 2, ..., n$.

It is often the case that a number of competing theories or models exist to describe the process that generated y. In our case we can use different ANN models, these models may have different basis sets, as well as different non linear structures, but in this case we use a linear hyperbolic tangent basis structures in the form

$$y = f(\mathbf{x}; \boldsymbol{\beta}, \mathbf{w}) = \sum_{j=1}^{m} \beta_j B_j(\mathbf{x}, \mathbf{w_j}) \qquad (2)$$

where $\mathbf{x} \in D \subset R^k$ and $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_m)^T$ is the vector of components of the hidden-output layers weights, that coefficients are associated to the hyperbolic tangent basis functions $\mathbf{B} = \left(B_1(\mathbf{x}, \mathbf{w_1}), B_2(\mathbf{x}, \mathbf{w_2}), ..., B_m(\mathbf{x}, \mathbf{w_m})\right)^T$ and $\mathbf{w_j}$ is the vector of input-hidden layers weights for the jth hidden node. The network has k inputs that represent the independent variables of the model, m nodes in the hidden layer, and one node in the output layer. The activation of the jth node in the hidden layer is

$$B_j(\mathbf{x}, \mathbf{w_j}) = \frac{1 - \exp(-2\mathbf{w_j^T x})}{1 + \exp(-2\mathbf{w_j^T x})} \qquad (3)$$

For the determination of the wind speed, an approximation based on a one hidden layer neural network has been used, since a considerable number of papers have appeared in the literature [17-18] discussing that an ANN with two layers of weights and sigmoidal or HTU hidden units can approximate arbitrarily any functional (one-one or many-one) continuous mapping from one finite-dimensional space to another, provided that the number of hidden units m is sufficiently large. The fact that the neural networks can approximate any given function with the desired accuracy is a powerful basis for the application of neural networks to regression. This capacity of ANNs for approximating any continuous function and the quick development of the computational power has motivated other researchers to adopt ANNs as an alternative method to predict the values of the response variable y. Although most ANN models share a common goal of performing functional mapping, different networks architectures vary greatly in their ability to handle different types of problems. The MLP with the efficient back-propagation training algorithm is probably the most frequently used type of neural network model in practical applications. However, owing to its multilayer structure and the limitations of the slow back-propagation algorithm, the training process often settles in undesirable local minima of the error surface or converges too slowly.

3. Hybrid Evolutionary Programming Algorithms

In this work, different variants of hybrid EAs have been applied, as can be seen in [13], all of them based on an evolutionary programming (EP) algorithm.

3.1. Evolutionary Programming Algorithm

The EP algorithm is an EA similar to the proposed in [13,19]. The algorithm begins with the random generation of N_P individuals; then the evolution process starts and a population-update algorithm is applied, where the population is subjected to the replication and mutation operations, but crossover is not considered, as this operation is usually regarded as being less effective for ANNs evolution. It uses two types of mutations: structural and parametric mutations. The structural mutation implies a modification of the structure of the function performed by the network and allows an exploration of different regions of the search space. The parametric mutation modifies the coefficients of the model using a simulated annealing algorithm;The general structure of the applied EA is detailed next:

Evolutionary Programming (EP)

```
1. Generate a random population of size NP.
2. Repeat until the stopping criterion is fulfilled
   2.1. Apply parametric mutation to the best 10% of individuals
   2.2. Apply structural mutation to the remaining 90% of individuals.
   2.3. Calculate the fitness of every individual in the population.
   2.4. Add best fitness individual of the last generation (elitist algorithm).
   2.5. Rank the individuals with respect to their fitness.
   2.6. Best 10% of population individuals are replicated and substitute the worst 10% of individuals.
3. Select the best individual of the population in the last generation and return it as the final solution.
```

The algorithm begins with the random generation of a larger number of networks than the number used during the evolutionary process. *$10N_P$* networks are generated, from which the best N_P best individuals are considered to form the initial population to be trained during the evolutionary process. Two error measurements are used for determining the precision of the neural network model, the mean squared error (MSE):

$$MSE(f) = \frac{1}{n_T}\sum_{l=1}^{n_T}(y_l - f(\mathbf{x}_l))^2 , \qquad (4)$$

where y_l is l-th observed value, and $f(\mathbf{x}_l)$ is the predicted value for pattern $\mathbf{x}_l$, and the standard error of prediction (SEP):

$$SEP(f) = \frac{100}{|\bar{y}|}\sqrt{MSE(f)} , \qquad (5)$$

where $\bar{y}$ is the average output of all patterns in dataset.

The fitness function $A(f)$ is defined by means of a strictly decreasing transformation of the MSE:

$$A(f) = \frac{1}{1+MSE(f)}, \qquad 0 < A(f) \le 1 \qquad (6)$$

The adjustment of both weights and structure of the ANNs is performed by the complementary action of two mutation operators: parametric and structural mutation. Parametric mutation implies a modification in the coefficients (β_j and w_{ji}) of the model, using a self adaptive simulated annealing algorithm. Structural mutation modifies the topology of the neural nets, helping the algorithm to avoid local minima and increasing the diversity of the trained individuals. Five structural mutations are applied sequentially to each network: node deletion, connection deletion, node addition, connection addition and node fusion. When node deletion is applied, number of hidden nodes to be removed is obtained as a uniform value in a previously specified interval. Apart from this mutation, if connection deletion is applied, the number of connections to be deleted in the neural net is also obtained as a uniform value, but in this case, as the mutation is less disruptive, the selected interval is selected to be a wider one. More details about this EP algorithm can be found in [13].

3.2. Hybrid Evolutionary Programming Algorithms

We apply a hybrid evolutionary algorithm based on the use of a clustering algorithm for deciding which individuals are subject to local optimization. The basic aim of this methodology is the minimization of the number of times the local optimization algorithm is applied without reducing the performance of the algorithm. This is especially important when the algorithm involves a high computational cost. On the other hand, removing the local optimization procedure usually yields a worse performance, as we will show in the experimental section. Thus, this method offers a good trade-off between performance and efficiency, since we apply the optimization algorithm to a reduced number of individuals. Moreover, the clustering process allows us to select a subset of individuals with different features. In this way, the optimization algorithm is more efficient. The local optimization algorithm used in our work is the Levenberg-Marquardt (L-M) optimization method. This algorithm is designed specifically for minimizing a sum-of-squares error [20]. In any case, any other local optimization algorithm can be used in a particular problem. As we have mentioned, the hybrid evolutionary algorithms are based on the combination of an evolutionary algorithm, a clustering process and a local search procedure.

In the hybrid EP (HEP), the EP is run without the local optimization algorithm and then it is applied to the best solution obtained by the EP in the final generation. This allows the precise local optimum around the final solution to be found. Another version of hybrid EA is the HEP with the clustering algorithm (HEPC), which applies the clustering process over a large enough subset of the best individuals in the final population. The number of individuals in this subset and the number of clusters to be created are critical parameters of the clustering process. Once clusters have been determined, the best individual in each cluster is selected and then optimized using the local search algorithm. The main objective of these methodologies is to reduce the number of times it is necessary to apply the local optimization procedure, since local search algorithms commonly involve a high computational cost. The clustering process selects the most representative groups of the population, providing a subset of individuals with different features. The selected clustering method selected is *k*-means clustering, using a distance measure defined for the vectors of the different values obtained for each individual over the training dataset. Further information can be found in Martínez-Estudillo et al. [13]. The hybrid algorithms applied are detailed next:

Hybrid Evolutionary Programming (HEP)
1. Generate a random population of size NP.
2. Repeat EP algorithm until the stopping criterion is fulfilled
3. Apply L-M algorithm to best solution obtained in the EP algorithm.
4. Return the optimized individual as the final solution.

Hybrid Evolutionary Programming with Clustering (HEPC)
1. Generate a random population of size NP.
2. Repeat EP algorithm until the stopping criterion is fulfilled
3. Apply k-means process to best NC individuals of the population in the last generation and assign a cluster to each individual.
4. Apply L-M algorithm to best solution obtained in each cluster.
5. Select the best individual among optimized ones and return it as the final solution.

4. Experiments

As previously stated, we are facing a wind speed prediction problem in this paper. A first process of downscaling (physical downscaling) can be carried out from these initial and bounding data, using the fifth generation Mesoscale Model (MM5 model) [21]. The result of this physical downscaling is a forecast of the wind speed and direction in a more realistic orography than the one giving by the global forecasting model. The MM5 model interpolates the values of wind speed to obtain mean hourly predictions. The output of the MM5 model, integrated using the initial and bounding conditions specified by the global model and local conditions, will not properly cover the complete surface of a wind park. The input variables of the ANN must be selected with care. In our case, we have chosen as input of the neural network the following: the wind speed series at two grid points surrounding the park (the procedure for the selection of the two point of the park can be seen in [8]); the wind direction and temperature at one of the previous points and two temporal series to obtain a measure of the solar cycle, strongly related to atmospheric circulation. Note that all these data are collected from the MM5 results at a given height, approximately equal for all the turbines, considering the orography of the park. We use the following equations to express the solar cycle:

$$S_1 = \sin(H\frac{2\pi}{24}) , \qquad (7)$$

$$S_2 = \cos(H\frac{2\pi}{24}) \qquad (8)$$

where $H = [0, 23]$ is an integer vector.

The experimental design was conducted using a holdout cross-validation procedure with approximately $3n/4$ instances for the training dataset, 6284 patterns, and $n/4$ instances for the generalization dataset, 1572 patterns, where n is the size of the dataset. We tested the methodology for three turbines of the wind park "La Fuensanta", located in Albacete, Spain (Figure 1). All parameters of the hybrid evolutionary algorithms are common for the three turbines, for all methodologies: the coefficients are initialized in the [–5, 5] interval; the maximum number of hidden nodes is $m = 9$; the size of the population is $N_P = 1,000$. The number of nodes that can be added or removed in a structural mutation is within the [1, 3] interval, whereas the number of connections that can be added or removed in a structural mutation is within the [1, 7] interval. The only parameter of the L–M algorithm is the tolerance of the error to stop the algorithm; in our experiment, this parameter has the value 0.01. The *k*-means algorithm is applied to 250 best individuals in the population. The number of clusters is 4 for the HEPC algorithm. The input variables were scaled in the interval [0.1,0.9], and the output variable (the wind speed) in the interval [0.1,0.9]. Results obtained with the different algorithms were evaluated by using both MSE and SEP. Table 1 shows the statistical results, mean value and standard deviation (Mean±SD) of the generalization errors obtained by the models in the 30 runs for three turbines, using the EP and the other two hybrid algorithms (HEP and HEPC). The results also include the best MSE model obtained in the 30 runs, which is compared to the best model obtained by using the neural network in [8]. From a purely descriptive point of view, the best result is obtained using the HEPC methodology for the three turbines.

According to this study, the following final optimal network model for turbine 1 were those reported in Table 2; in which quantitative equation systems are presented for the direct determination of accurate wind speed predictions including: (a) the wind speed series at two grid points surrounding the park, the wind direction and temperature at one of the previous points, and only one measure of the solar cycle, S_2, because the x5 independent variable does not appear in the model; (b) the optimized network weights; and (c) the hyperbolic tangent transfer functions for the MLP models. According to the MSE values, this model can be readily used for the determination of the wind speed for turbine 1. It can be seen in Table 2, that HTUs enable a neural network to form function of inputs with increased information capacity and smaller network architecture.

5. Conclusions

This study demonstrated the potential of HTU neural networks models trained using a hybrid evolutionary algorithm for a problem of short-term wind speed prediction. We hybridized HTU neural networks models with global and mesoscale physical forecasting models, in such a way that the neural networks tackle the final statistical downscaling process. We have described the characteristics of the different hybrid algorithms used, and the complete system of forecasting including the evolutionary artificial neural network implemented. We have applied our models for three turbines in the short-term wind speed prediction in a wind park in Spain, with absolutely competitive results when compared to other neural network methodologies.

Table 1. Statistical results of generalization SEP and MSE errors for several turbines, for 30 executions of the EP, HEP, and HEPC algorithms, statistical results of the #links of the EP and HEPC algorithms and best MSE model generalization error of the EP and the MLP in [8] algorithms. The best result for the MSE and the #links has been represented in bold face.

	Mean± SD		
Method	Turbine 1	Turbine 15	Turbine 21
SEP_{EP}	41.50±0.75	41.83±0.64	41.57±0.47
SEP_{HEP}	40.54±0.71	40.79±1.09	40.69±0.73
SEP_{HEPC}	40.40±0.79	40.94±1.10	40.69±0.83
MSE_{EP}	6.07±0.22	5.77±0.18	5.55±0.13
MSE_{HEP}	5.79±0.20	5.49±0.29	5.38±0.19
MSE_{HEPC}	**5.75±0.23**	**5.53±0.30**	**5.32±0.22**
MSE_{Best} (HEPC)	**5.36** (3 nodes)	**5.05** (4 nodes)	**4.95** (6 nodes)
MSE_{Best} (MLP [8])	5.53 (14 nodes)	5.10 (15 nodes)	5.06 (12 nodes)
	# links		
EP	18.13±5.85	17.27±5.26	17.33± 5.67
HEPC	**17.93±6.15**	**16.93±5.64**	**17.26± 5.96**

6. References

[1] M. C. Alexiadis, P.S. Dokopoulos, H. Sahsamanoglou, I. M. Manousaridis, Short-term forecasting of wind speed and related electrical power, Solar Energy 63 (1) 61-68. 1998

[2] L. Landberg, Short-term prediction of local wind conditions, J. Wind Eng. Ind. Aerodyn. 89 235-245. 2001.

[3] M. Khashei, M. Bijari, G. Raissi-Ardali, Improvement of auto-regressive integrated moving average models using fuzzy logic and artificial neural networks (ANNs), Neurocomputing, 2008,

[4] T. G. Barbounis, J. B. Theocharis, Locally recurrent neural networks for wind speed prediction using spatial correlation, Neurocomputing 70 1525-1542. 2007

[5] S. Li, C. D. Wunch, A. O'Hair, G. M. Giesselmann, Using neural networks to estimate wind turbine power generation, IEEE Trans. Energy Conversion 16 (3) 276-281. 2001

[6] M. A. Mohandes, T. O. Halawani, S. Rehman, A. A. Hussain, Support vector machines for wind speed prediction, Renewable Energy 29 939-947. 2004

[7] S. Salcedo-Sanz, Á. M. Pérez-Bellido, E. G. Ortiz-García, A. Portilla-Figueras L. Prieto, F. Correoso. Accurate short-term wind speed prediction by exploiting diversity in input data using banks of artificial neural networks Neurocomputing 72 1336-1341. 2009

[8] S. Salcedo-Sanz, Á. M. Pérez-Bellido, E. G. Ortiz-García, A. Portilla-Figueras, L. Prieto, D. Paredes. Hybridizing the fifth generation mesoscale model with artificial neural networks for short-term wind speed prediction. Renewable Energy 34 1451-1457. 2009

[9] J. A. Dudhia nonhydrostatic version of the Penn State-NCAR mesoscale model: validation, tests and simulation of an Atlantic cyclone and cold front. Monthly Weather Review. 121, 1493-513. 1993

[10] J.C. Alpert, K.A.Campana, P.M. Caplan, D.G. Deaven, M. Iredell, B. Katz. Recent changes implemented into the global forecast system at NMC. M. Kanamitsu. Weather and Forecasting. 6 (3), 425-35. 1991

[11] S. Haykin. Neural Networks and learning machines. Third Edition. Prentice Hall. 2009

[12] D. F. Specht. A general regression neural network. IEEE Transactionson Neural Networks, 2(6), 568-576. 1991.

[13] A. C. Martínez-Estudillo, F. J. Martínez-Estudillo, C. Hervás-Martínez, and N. García-Pedrajas, Evolutionary Product Unit based Neural Networks for Regression, Neural Networks 19 477-486. 2006

[14] X. Yao, Evolving artificial neural networks, Proceedings of the IEEE, 87(9) 1423-1447. 1999

[15] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York. 1994

[16] A. C. Martínez-Estudillo, C. Hervás-Martínez, F. J. Martínez-Estudillo, N. García-Pedrajas. Hybridization of evolutionary algorithms and local search by means of a clustering method. IEEE Trans. Syst. Man Cybernetics 36 (3), 534-545. 2006

[17] K.Funahashi, On the approximate realization of continuous mappings by neural networks. Neural Net. 2 (3), 183-192. 1989

[18] K. Hornick, M. Stinchcombe, and H. White. Multilayer feedforward network are universal approximators. Neural Net. 2 (5), 359-366. 1989

[19] P. J. Angeline, G. M. Saunders, J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans. Neural Networks 5 (1), 54-65. 1994.

[20] M. Bishop. Neural networks for pattern recognition. Oxford University Press, UK. 1995

[21] J. A. Dudhia nonhydrostatic version of the Penn State-NCAR mesoscale model: validation, tests and simulation of an Atlantic cyclone and cold front. Monthly Weather Review. 121, 1493-513. 1993

.

Table 2. Best model obtained with a feed forward neural network with hyperbolic tangent basis function trained by the HEPC algorithm.

Best Model Turbine 1 Hyperbolic Tangent Unit
WS(HTU)= 0.78-0.30*HTU1+3.17*HTU2+2.55*HTU3
HTU1= (exp[2(4.84*x4-5.92*x3-1.56*x1+0.75]-1) / (exp[2(4.84*x4-5.92*x3-1.56*x1+0.75]+1)
HTU2= (exp[2(0.05*x6+1.31*x4+0.31*x2-2.55]-1) / (exp[2(0.05*x6+1.31*x4+0.31*x2-2.55]+1)
HTU3= (exp[2 -2.97*x3+0.41*x1+3.67]-1) / (exp[2(-2.97*x3+0.41*x1+3.67]+1)
hidden nodes: 3; #parameters : 15; SEP_G= 39.02; MSE_G=5.36; HEPC. #gen=400

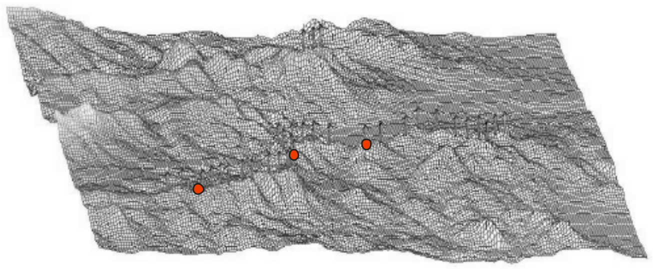

Figure 1. Wind park orography (La Fuensanta, Albacete, Spain) and location of the wind turbines in this study.