Wind Generation Forecast from the Perspective of a DSO

Miguel Duarte Saragaço Garcia Godinho – miguel.duarte.godinho@tecnico.ulisboa.pt
Prof. Dr. Rui Manuel Gameiro de Castro – rcastro@tecnico.ulisboa.pt
Instituto Superior Técnico – Universidade de Lisboa

Abstract — The constant investment growth in renewable energies by electric companies has raised a prominent issue about their integration in the electric grid. Particularly in wind power, since the wind has a high volatility and the respective energy produced cannot be stored in a large-scale due to its excessive costs, it is important to be able to forecast wind power generation accurately. This allows to know in advance an accurate estimate of the energy that will be produced for the near future creating a positive economic impact on the Distribution System Operator (DSO).

The aim of this work is to perform one hour ahead wind power forecasts by applying artificial intelligence methods, such as Artificial Neural Networks (ANN), Adaptive Neural Fuzzy Inference System (ANFIS) and Radial Basis Function Network (RBFN), and then find out which one gives the best performance. The persistence method is also employed since it is a simple method typically used for comparisons purposes.

The abovementioned methods, their architectures and the respective learning algorithms are presented. Then, the best architecture of each method is found out by optimizing it by season. Afterwards, the methods are trained and tested using real wind power data from 2010 to 2014. Additionally, to improve the forecast results and avoid the overfitting problem a cross-validation technique is applied. Finally, conclusions are drawn basing on the test results and comparisons of the forecasting performances among the methods and seasons are made.


I. INTRODUCTION

The constant investment growth by the electric companies in renewable energies has raised a critical issue regarding their integration in the electric grid. Moreover, that investment has originated an increasing dependency in renewable energies, which at first sight could mean a huge advantage since the use of fossil energies is being reduced. On the one hand, that is completely true but, on the other hand, there may be some disadvantages especially if the wind and/or solar energy have a significant level of penetration in electric grid. Since the wind and the sun are volatile, it is imperative to have future estimations of the power produced through those elements of nature.

Therefore, this work addresses one of the important processes that is nowadays a part of the daily activities in many electric companies – the wind power forecast process.

To predict wind power generation, there are many techniques available that have been developed over time. In [1] and [2], the authors give an overview of those available techniques, where they catalog them into some types of forecasting techniques: Persistence models, Numeric Weather Prediction (NWP) methods, Statistical and Artificial Neural Networks (ANN) methods and Hybrid methods or combined approaches.

In the NWP methods the input variables are usually physical variables of meteorological information such as the roughness of the terrain, temperature, pressure and wind speed. Although the accurate predictions of NWP methods, they are obtained with considerable execution times due to their complexity, which is not desirable for short-term predictions.

Therefore, for short-term predictions the statistical and soft computing methods are used instead. In the first one, historical data (wind power data) is used to tune the method parameters. Examples of statistical methods are the Auto-Regressive Moving Average (ARMA) and Auto-Regressive Integrated Moving Average (ARIMA). Despite their good performances, the statistical methods have become increasingly unappealing due to the arising and development of the soft computing methods.

The soft computing methods have the main advantage of dealing with nonlinear problems and data more effectively than the previously mentioned ones and hence achieving better performances and results. In this category of Artificial Intelligence (AI) methods, we can find methods based on ANN, data mining, fuzzy logic and evolutionary computation. Among them, the ANNs have been widely and successfully used in several forecasting applications. It has become the reference method within the AI ones and it hasn’t only the purpose of making accurate predictions, it is also used as a basis for comparison with other more advanced and complex AI methods.

Among those more advanced AI methods there are the Radial Basis Function Network (RBFN) and Adaptive Neuro Fuzzy Inference System (ANFIS) methods. The first one is a method created based on the ANNs having a similar architecture, however with a totally different way of training process whereas, the second one is a method that can be classified as a hybrid method, which consist of the combination of two or more methods. In the case of ANFIS method there is a combination between ANNs and fuzzy logic. The main goal is to benefit from the advantages of each individual method to then obtain a globally optimal forecasting performances [1]. Usually, in terms of performance, the hybrid methods present better results than if only a single method was used. For wind power forecasting problems, that combination can go from a
combination of physical models, like NWP methods, and statistical or AI methods to a combination of several AI methods.

Using one or other forecast techniques, what really matters is the achievement of reliable and accurate results. That is the perspective of energy companies whose core activity is the production, distribution and/or transmission of electric energy and that have an active role in the electricity market since the wind power forecast results have somehow an influence in their financial activities.

In the perspective of a Distribution System Operator (DSO), such as EDP Distribuição (Portuguese Energy Company), wind forecast activities are crucial since they allow to have a more efficient management of the electric grid. A DSO has the duty to guarantee the reliability of the electric grid ensuring the supply of electric energy to all the customers complying with the legal requirements and high standards of quality. Therefore, for an electric grid with high penetration of wind energy, a bad wind power forecast can result in a negative impact on its reliability.

However, it is not only the electric grid that is affected but also the electric energy market. In the market, the wind power producers have to make wind power predictions for a specific period of time. They must provide a generation schedule for a considered time horizon [2] where any deviation from this schedule could impose some undesirable penalties for them. To minimize those penalties, an accurate wind power forecast is required.

Therefore, the proposed goals for this thesis are the following:

- To exploit some AI methods full potential on wind power forecasting tasks, more specifically the ANN, ANFIS and RBFN methods.
- To implement the proposed methods, by first optimizing their architectures adapting them to the nature of the problem and then perform 1-hour-ahead wind power forecasts.
- To draw conclusions about their performances based on the results obtained.
- To identify the most accurate method, the one that produced the best forecast results and better suited to this kind of problems.

II. METHODS

In this section, all the proposed methods will be presented, more exactly, their architectures and learning algorithms. Starting with the simplest one, Persistence Method, followed by the ANN and ANFIS methods, and ending with the RBFN method.

A. Persistence Method

Due to its great simplicity, the persistence method is rather used in time-series prediction problems usually as a means of comparison among the other approaches [3]. It is the easiest and fastest way of producing relatively accurate forecasts. Furthermore, it is proved that this method produces accurate predictions for short-term predictions. However, the accuracy of the method starts to decrease as the time scale of prediction increases [1]. This method assumes that the wind power forecast at time \( t + k \) is equal to the real wind power measured at time \( t \) (eq. 1).

\[
p_{\text{fore}}(t + k) = p_{\text{real}}(t),
\]

where \( k \) is the forecast horizon. Since one-hour-ahead forecasts are to be performed, the value of \( k \) is equal to 1 and \( t \) is expressed in hours.

B. Artificial Neural Network

The ANN is an artificial intelligence method which allows to simulate the human reasoning process through the creation of a network composed of several processing units, called neurons, bound together by connections with a weight associated. That network is then trained by a learning algorithm, where, in case of supervised learning, pairs of data (input, target) are presented to the network to update and optimize the weights through an iterative process that minimizes a predefined error function [4].

The ANNs, due to their easy implementation and assurance of reliable results with high accuracy rates, have become a very attractive tool to solve time-series prediction problems. In fact, the ANNs have been used to solve forecasting problems, such as wind power prediction [5], wind speed prediction [3], and also used as a means of comparison with other complex artificial intelligence methods [3], [6].

1) Architecture

An architecture example of an ANN is illustrated in Figure 1. This ANN is a Multilayer Perceptron (MLP), which is a feed-forward neural network where its learning procedure is performed by a supervised algorithm called backpropagation, and its structure is composed of one input layer, \( L \) hidden layers and one output layer linked by unidirectional weighted connections.

![Figure 1 - ANN's architecture: MLP](image)

In the input layer is where a vector of external data (inputs) \( \mathbf{i} = [i_1, \ldots, i_n] \) is presented to the network. The output layer is composed of \( p \) neurons and it is where the network outputs are obtained \( \mathbf{o} = [o, \ldots, o_p] \). Between the input and the output layers there are one or more hidden layers (from 1 to \( L \)) where each one of them is composed of several neurons (from 1 to \( H_L \)). The basic structure of a MLP neuron, represented by a circle in the Figure 1, is detailed in the Figure 2.
The MLP neuron output is given by
\[ y = f(s) = f \left( \sum_{i=1}^{n} w_i x_i + w_0 x_0 \right). \]

The neuron receives a weighted sum of inputs \((\sum_{i=0}^{n} w_i x_i)\) that passes through a transfer function \(f\), called activation function, to then produce an output \(y\). The activation function can be either a linear or a non-linear (log-sigmoid, tanh-sigmoid) differentiable function. The most common one is the log-sigmoid (eq. 3).
\[ f(s) = \frac{1}{1 + e^{-s}}. \]

2) **Levenberg-Marquardt Algorithm**

The Levenberg-Marquardt (LM) algorithm, which was first described by Marquardt in [7], is a supervised algorithm that is often chosen for training ANNs since it is one of the fastest backpropagation algorithms. Its fastness results from the approximation of the second derivatives by eq. 4, avoiding to compute the hessian matrix, \(H\).
\[ H \approx J J^T, \]
where \( J \) is the Jacobian matrix (eq. 5) containing the first derivatives of the error function \(SSE\) (eq. 6) considering the weights and biases, which is given by
\[ J(w) = \frac{\partial e_m}{\partial w_j}. \]

for \(1 < m < M\) and \(1 < j < R\). The error function is given by
\[ SSE = f(w) = \frac{1}{2} \| e(w) \|^2 = \frac{1}{2} \sum_{m=1}^{M} e_m^2(w) = \frac{1}{2} \sum_{m=1}^{M} \left( a_m(w) - d_m \right)^2, \]
where \(M\) is total number of patterns; \(a_m\) is the observed value for the \(m\)th pattern; \(d_m\) is the desired value for the \(m\)th pattern; \(w\) is a vector with all the weights and biases with dimension \((1 \times R)\).

Levenberg proposed an algorithm that combines the characteristics of the gradient descent and Gauss-Newton methods, where the weights and biases updating are performed as described in equation eq. 7.
\[ w^{k+1} = w^k - (H + \mu I)^{-1} J(w^k) e(w^k), \]
where \(I\) is the identity matrix and \(\mu\) the learning rate.

The LM algorithm could be described by the following pseudo-code:
1) Initialize randomly the weights and biases, \(w\).
2) Compute the initial value of the error function, \(SSE\) (eq. 6).
3) Compute the Jacobian, \(J\), and the Hessian, \(H\), matrixes (eq. 5 and eq. 4, respectively)
4) Update \(w\) (eq. 7).

5) Compute the error function again, \(SSE\) (eq. 6).
   a) If the \(SSE\) increased the weights and biases calculated in 4) are throwed out and the learning rate, \(\mu\), increases by a factor of \(\beta_{up}\). Then return to 4)
   b) If the \(SSE\) decreased the weights and biases calculated in 4) are stored and the learning rate, \(\mu\), decreases by a factor of \(\beta_{down}\). Then return to 3).
6) The algorithm ends if the stopping criteria is reached.

In this algorithm, the stopping criteria could be either if the value of the error function respecting to the validation set is higher than the minimum value of the error function achieved throughout the learning process, for the same set, for 6 consecutive epochs or if the value of the error function has been reduced to a threshold, called error goal.

C. **Adaptive Neural Fuzzy Inference System**

The ANFIS method was proposed by Jang [8], in 1993. He combined an ANN with a fuzzy inference system (FIS) highlighting that with that combination it is possible to create a set of fuzzy if-then rules with appropriate membership functions (MF) to achieve the desired input-output mapping [8].

1) **Architecture**

The ANFIS is a feedforward neural network and its architecture is composed of 5 layers. In Figure 3 is represented the equivalent ANFIS architecture with 2 inputs \(x\) and \(y\), and 2 MFs for each input.

![Figure 3 - Equivalent ANFIS Architecture [8]](image)

The characteristics of each layer will be detailed below. To represent the output of the neuron \(i\) in the layer \(l\) it is used \(O_{i}^{l}\) as notation.

a) **Layer 1**

This layer is composed of adaptive nodes. It is known as fuzzy layer. Here the nodes’ function is given by:
\[ O_{i}^{1} = \mu_{A_{i}}(x), \]
where \(x\) is the input of the node \(i\); \(\mu_{A_{i}}(x)\) is the MF of the fuzzy set \(A_{i}\).

The MF can take many different forms, such as triangular, gaussian and bell shape. Each one of them has a function associated which is defined by a set of parameters. For example, if the generalized bell function were used as a MF (eq. 9), \(bell(x,a,b,c)\), the parameters would be \(a\), \(b\) and \(c\). Regarding the notation in the Figure 3, we would have:
\[ \mu_{A_{i}}(x) = \frac{1}{1 + \left| \frac{x - c_{i}}{a_{i}} \right|^{2b_{i}}}, \]
where \((a_i, b_i, c_i)\) would be the parameters of each node \(i \ (1 < i < 2)\) in the first layer for the input \(x\). These parameters are called premise parameters.

b) **Layer 2**

This layer is known as the rule layer and it is composed of fixed nodes. It is where the firing strengths are computed for each rule, which are represented by the output \(O^2_i\). Here the product between both node inputs is performed (eq. 10). However, other \(T\)-norm operators that perform generalized AND can be used instead.

\[
O^2_i = w_i = \mu_{A_i}(x) \cdot \mu_{B_i}(x) \quad i = 1, 2 .
\] (10)

c) **Layer 3**

The third layer is called the normalized layer and it is composed of fixed nodes. It is where the firing strengths of each rule are normalized (eq. 11)

\[
O^3_i = \bar{w}_i = \frac{w_i}{w_1 + w_2} \quad i = 1, 2 .
\] (11)

d) **Layer 4**

The fourth layer is called the defuzzification layer and it is composed of adaptive nodes. Here, apart from the normalized firing strengths, the nodes have also as inputs the outputs of the first layer (\(x\) and \(y\)). The output \(O^4_i\) of this layer is given by

\[
O^4_i = \bar{w}_i \cdot f_i = \bar{w}_i \cdot (p_i x + q_i y + r_i) ,
\] (12)

where \((p_i, q_i, r_i)\) are the consequent parameters.

e) **Layer 5**

The output layer is composed of only one node, where the output is the summation of the previous layer’s outputs.

\[
O^5 = \sum_i O^4_i .
\] (13)

In the layers 2, 3 and 4, the number of nodes and the number of fuzzy if-then rules is given by \(M^n\), where \(M\) is the MFs number for each input and \(n\) the number of inputs [10].

2) **Hybrid Algorithm**

The ANFIS learning algorithm is a hybrid algorithm that combines the gradient descent method with the Least Squared Method (LSM) to estimate the network parameters. This algorithm can be split into 2 phases, the forward pass and the backward pass. In the first one, the consequent parameters are estimated using the LSM (the premise parameters are fixed), whereas in the second one the error rates are propagated from the output layer to the fuzzification layer allowing to compute and update the premise parameters using the gradient descent method in batch mode (the consequent parameters are fixed) [8], [9], [10], [11].

D. **Radial Basis Function Network**

The RBFN is quite similar to the traditional neural networks (ANNs), like Multilayer Perceptron (MLPs). However, there are some differences between them. Whereas the MLPs can have more than one hidden layer, the RBFN has only one whose neurons’ activation function is a radial basis function (RBF).

1) **Architecture**

The generalized architecture of RBFN is illustrated in Figure 4. This is a feed-forward neural network and its structure is composed of 3 layers (one input layer, one hidden layer and one output layer).

In the input layer the input vector \((x = (x_1, x_2, ..., x_n))\) goes straight to the hidden layer which means that the weights between those layers are equal to 1. This configures another difference between RBFN and ANNs.

In the hidden layer the neurons have a RBF as activation function where the most common one is the Gaussian function.

(eq. 14) In fact, the Gaussian function is a localized RBF with the property that \(\phi(a, \sigma^2) \rightarrow 0 \text{ as } a \rightarrow \infty\) which makes it the most selected function as RBF [13]. Considering the notation in Figure 4, \(h\) being the total number of neurons in the hidden layer and \(n\) the total number of inputs, the RBF is given by

\[
\phi_i(a_r, \sigma^2_r) = e^{\left(-\frac{x - c_r}{2\sigma^2_r}\right)},
\] (14)

where \(a_i\) is the Euclidean distance between the inputs \(x\) and the centers \((c_r = [c_1, c_2, ..., c_n] \text{ for } r = 1, 2, ..., n)\) (eq. 15), which are specific parameters of each neuron; \(\sigma^2\) is the width (variance) of that neuron; \(\phi\) is the neuron’s output.

So the Euclidean distance \(a\) in neuron \(i\) (\(i=1,2,..,h\)) is given by

\[
a_i = \|x - c_{ir}\| = \sqrt{(x_1 - c_{ir})^2 + (x_2 - c_{ir})^2 + (x_n - c_{ir})^2} .
\] (15)

As we can see in eq. 14, the Gaussian function gives the highest output when the input vector is practically in the same position of the neuron’s centers. In contrast, as the distance between the input vector and the centers starts to increase the output of the Gaussian function starts to decrease. Besides that, it is the width of the neuron that controls the smoothness of the function. So, if the width is large then decreases slowly, if it is small then decreases quickly.

In the output layer the weighted summation of each neuron’s output is performed for each RBFN’s output. This can be expressed as

\[
y_j = \sum_{i=1}^{h} w_{ij} \phi_i(a_r, \sigma^2_r) + w_0,
\] (16)

where \(y_j\) represents the \(j\) output of the network, for \(j=1,2,...,P\); \(w_{ij}\) the weights from neuron \(i\) to output \(j\); \(\phi_i(a_r, \sigma^2_r)\) is the
output of the neuron $i$; $w_0$ is the bias term; $h$ is total number of neurons.

2) Learning Algorithms

The RBFN can be trained in many different ways being one of them using the combination of unsupervised and supervised techniques. Therefore, three approaches of learning algorithms strategies are presented:

- Stochastic Gradient Descent (SGD) Algorithm;
- Hybrid Algorithm (K-means + LSM);
- Orthogonal Least Squares (OLS) Algorithm;

a) Stochastic Gradient Descent Algorithm

As in traditional ANNs the SGD can also be employed in RBFN. The only difference lies in the fact that the RBFN has different parameters to adapt throughout the learning process.

The main goal of SGD is to minimize the error function $E$ (SSE) described by $e_{i, k}$:

$$E = \sum_{k=1}^{K} \sum_{i=1}^{N} e_{i, k}^2$$

where $e_{i, k}$ is the error at the output of the neuron $i$, $K$ is the total number of training patterns, and $N$ is the total number of output nodes.

The only difference lies in the fact that the RBFN has parameters to adapt throughout the learning process.

Moreover, adaptive step size techniques are also used.

b) Hybrid Algorithm

In this learning strategy the learning procedure is performed in two phases. In the first one, the centers are computed through an unsupervised clustering algorithm (k-means). Afterwards, the widths are computed. And finally, in the second phase, the weights are obtained by minimizing the error function relative to the parameter set at iteration $k$.

The final expressions for the adaptation of the network parameters are the following, (for $i = 1, 2, \ldots, h$, $r = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$):

$$e_{i, k}^{(k+1)} = e_{i, k}^{(k)} + \eta \sum_{j=1}^{P} \left( w_{ij}^{(k)} \right) \frac{\phi_i(k) - \phi_i(k+1)}{\sigma_i^{2(k)}}$$

$$\sigma_i^{(k+1)} = \sigma_i^{(k)} + \eta \sum_{j=1}^{P} \left( w_{ij}^{(k)} \right) \frac{\phi_i(k) - \phi_i(k+1)}{\sigma_i^{2(k)}}$$

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} + \eta \phi_i^{(k)}$$

where $\eta$ is the learning rate; $e$ is the error $y - d$ where $y$ is the real output value and $d$ is the desired.

Moreover, to get better and faster results, a momentum and an adaptive step size techniques are also implemented.

3) Orthogonal Least Squares Algorithm

In this way of training, the network is built in a more efficient way than in the previous ones by choosing and adding centers one by one until the stopping criteria is reached. Throughout the centers’ choice procedure, all the training patterns are considered as candidates. After going through all the training patterns, the one that yielded the major reduction on the error function is chosen as a center. This process is repeated until the stopping criteria is reached.

For better understanding how this method works, it is fundamental to consider the RBFN as a special case of the linear regression model [15]:

$$d(C, x^k) = \sqrt{(c_{1, k} - x_{1, k})^2 + \cdots + (c_{l, k} - x_{l, k})^2}$$

where $c_l$ is the $l$th element of the vector $C$; $x_{l, k}$ is the $l$th element of the vector $x^k$; and $l$ is the total number of training patterns.
where \( d(t) \) is the desired output value; H the total number of candidates; \( w_i \) the parameters (weights); \( \phi_i(t) \) the regressors; 
\( e(t) \) the error signal.

The equation eq. 28 is represented in matrix form as follows:

\[
d = \Phi W + E,
\]

where

\[
d = [d(1), d(2), \ldots, d(M)]^T, \\
\Phi = [\phi_1, \phi_2, \ldots, \phi_H], \\
W = [w_1, w_2, \ldots, w_M]^T, \\
E = [e(1), e(2), \ldots, e(M)]^T,
\]

and M is the total number of training patterns.

The OLS method involves the transformation of the set of \( \phi_i \) into a set of orthogonal basis vector allowing to compute the individual contribution to the desired output value from each basis vector. In this way, \( \Phi \) is decomposed into

\[
\Phi = QA,
\]

where A is an upper triangular matrix with dimension (H x H); Q is a matrix (M x H) composed of orthogonal vectors \( q_i \). This means that \( q_i^T q_j = 0 \) for \( i \neq j \).

\[
A = \begin{bmatrix}
1 & a_{12} & a_{13} & \cdots & a_{1H} \\
0 & 1 & a_{23} & \cdots & a_{2H} \\
0 & 0 & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & a_{H-1, H} \\
0 & 0 & \cdots & 0 & 1
\end{bmatrix}, \\
Q = [q_1, q_2, \ldots, q_H].
\]

Since the space spanned by the set of orthogonal basis vector \( q_i \) is the same space spanned by the set of \( \phi_i \), the equation eq. 28 can be rewritten as follows

\[
d = Qg + E.
\]

Hence, the OLS solution \( \hat{g} \) is given by

\[
\hat{g} = A\hat{W} = K^{-1}Q^Td = (Q^TQ)^{-1}Q^Td,
\]

where \( K \) is a diagonal matrix given by

\[
K = Q^TQ = diag(k_1, k_2, \ldots, k_H) = \sum_{i=1}^{H} q_i(t) \ast q_i(t).
\]

Since Q is orthogonal, each component of \( \hat{g} \) can be computed independently.

\[
\hat{g}_i = \frac{q_i^Td}{q_i^Tq_i}.
\]

This is exactly why the OLS is a very efficient algorithm avoiding the computation of the pseudo-inverse matrix.

To evaluate which center performs the highest reduction on the error function, it is used the error reduction ratio (ERR) given by

\[
[ERR]_i = \frac{\hat{g}_i^2 q_i^Tq_i}{d^Tq_i}.
\]

To solve the equation eq. 36 and estimate the parameters \( \hat{W} \) the classical Gram-Schmidt method is employed, where computes one column of \( A \) at a time and orthogonalizes \( \Phi \) in the following way: at the kth stage make the kth column orthogonal to each of the k-1 previously orthogonalized columns and repeat the operation for \( k = 2, \ldots, H \). The expressions in eq. 40 represent this procedure.

\[
q_1 = \phi_1 \\
\alpha_{ik} = \frac{q_i^T \phi_k}{q_i^Tq_i}, \quad 1 \leq i \leq k \\
q_k = \phi_k - \sum_{i=1}^{k-1} \alpha_{ik} q_i
\]

Now, the OLS algorithm can be finally summarized in the following steps:

1) Initialize the RBFN with H candidates and \( k = 0 \) centers. Here the width is defined a priori as well as the error goal (stopping criteria). Besides that, it is also possible to define an upper limit of neurons in the hidden layer, which is given by \( H_s \).

2) For the first center to be chosen (\( k=1 \)) do:
   a. For \( 1 \leq j \leq H \), compute:
      \[
      \begin{bmatrix}
      q_1^{(j)} \\
      \hat{g}_1^{(j)} \\
      ERR_1^{(j)} = \frac{\hat{g}_1^{(j)}q_1^{(j)} - \alpha_{1k}q_k^{(j)}}{d^Tq_i}
      \end{bmatrix}
      \]
   b. Find the pattern that yields the highest reduction on the error function:
      \[
      r_1 = \arg \max_j \left( ERR_1^{(j)} \right) \quad 1 \leq j \leq H.
      \]
   c. Select:
      \[
      q_1 = q_1^{(r_1)} = \phi_{r_1}.
      \]
   3) For \( k = 2, \ldots, H_s \) (At the kth step)
      a. For \( 1 \leq j \leq H \), \( j \neq r_1, \ldots, j \neq r_{k-1} \) compute:
         \[
         \begin{bmatrix}
         q_1^{(j)} \\
         \hat{g}_1^{(j)} \\
         ERR_1^{(j)} = \frac{\hat{g}_1^{(j)}q_1^{(j)} - \alpha_{1k}q_k^{(j)}}{d^Tq_i}
         \end{bmatrix}
         \]
      b. Find the pattern that yields the highest reduction on the error function:
         \[
         r_k = \arg \max_j \left( ERR_1^{(j)} \right) \quad 1 \leq j \leq H ; \quad j \neq r_1, \ldots, j \neq r_{k-1}.
         \]
      c. Select:
         \[
         q_k = q_k^{(r_k)} = \phi_{r_k} - \sum_{i=1}^{k-1} \alpha_{ik} q_i.
         \]
4) The procedure ends whether k reaches the upper limit of neurons in the hidden layer defined in the beginning or at a certain iteration \( k_{th} \) the following happens

\[
1 - \sum_{i=1}^{k_{th}} ERR_i < \rho,
\]
where $0 < \rho < 1$ is a chosen tolerance.

At the end of this process, a RBFN with $H_s$ or $k_s$ centers is obtained, depending on what was the stopping criteria.

### III. IMPLEMENTATION AND DATA

In this section, all the implementation processes, namely the data sets used, the data preprocessing and how the network parameters for the AI methods were determined, are detailed.

All methods presented in section II were implemented using the Neural Network and Fuzzy Logic toolboxes of MATLAB 2017a software.

The main goal is to perform one-hour-ahead wind power predictions only with wind power data as input. This can be described by

$$p_{t+1}^{\text{fore}} = f(P_t, P_{t-1}, P_{t-2}, \ldots, P_{t-(k-1)}),$$  

(48)

where $p_{t+1}^{\text{fore}}$ corresponds to the wind power forecast at instant $t+1$ (in hours); $P_t, P_{t-1}, P_{t-2}, \ldots, P_{t-(k-1)}$ are the input values from the wind power data at the instants before $t+1$; $k$ is the number of inputs, or lags ($k \geq 1$); $f$ corresponds to all the linear and nonlinear functions behind the applied method that produces the forecast values $p_{t+1}^{\text{fore}}$.

#### A. Wind Power Data

The wind power data used in this work was collected and discretized at intervals of 15 minutes by Redes Energéticas Nacionais (REN) from 2010 to 2014.

A daily and monthly average representation of that data considering the 5 years period is shown in Figure 5.

![Figure 5 - Daily/Monthly average of wind power from 2010 to 2014](image)

By analyzing Figure 5, it is possible to conclude the existence of seasonality in the data. Therefore, it makes sense to optimize, train and test the networks in this way, performing predictions for each season of the year (winter, spring, summer, autumn). For simplicity reasons the winter, spring, summer and autumn seasons are assumed to englobe the following set of months: December – January – February, March – April – May – June – July – August and September – October – November, respectively.

#### B. Data normalization

The data was normalized using the annual average values of installed wind power capacity ($P_{\text{av inst}}$) for each year. These values were computed by knowing the installed wind power capacity in the end of the years and then making the average for the corresponding years (Table 1).

<table>
<thead>
<tr>
<th>Year</th>
<th>$P_{\text{av inst}}$ [MW] (end of the year)</th>
<th>2009</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
<th>2014</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3357</td>
<td>3706</td>
<td>4080</td>
<td>4194</td>
<td>4364</td>
<td>4541</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>3352</td>
<td>3893</td>
<td>4137</td>
<td>4279</td>
<td>4453</td>
</tr>
</tbody>
</table>

As shown in Table 1, each year has a different value of $P_{\text{av inst}}$, therefore the data is split by year and then normalized according to eq. 49, to get values that range from 0 to 1.

$$p_n(\text{year}) = \frac{P(\text{year})}{P_{\text{av inst}}(\text{year})} [\text{p.u.}],$$  

(49)

where $P(\text{year})$ is the original wind power value and $p_n(\text{year})$ is the normalized one for a specific year.

#### C. Inputs Number

The inputs number can be estimated by computing the Partial Autocorrelation Coefficients (PACs) of the time series [4]. These coefficients give the correlation between a variable at an instant ($t$) and the same variable at another instant ($t+a$) without taking into account other values of the variable at other instants ($a$ is the number of delays).

The first five coefficients for each year’s season are presented in Table 2.

<table>
<thead>
<tr>
<th>Year</th>
<th>Delays</th>
<th>Winter</th>
<th>Spring</th>
<th>Summer</th>
<th>Autumn</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>1</td>
<td>0.9850</td>
<td>-0.7219</td>
<td>0.1008</td>
<td>-0.0830</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-0.7450</td>
<td>0.1689</td>
<td>-0.0026</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-0.8359</td>
<td>0.2644</td>
<td>-0.0058</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-0.7192</td>
<td>0.1556</td>
<td>-0.0418</td>
<td></td>
</tr>
</tbody>
</table>

As we can see in Table 2, in general, the PACs are significant for lags (delays) ranging from 1 to 3. The major difference is in the summer season where the PACs for a delay of 2 and 3 are substantial higher than in the other seasons of the year.

In conclusion, the input number should range between 2 and 3.

#### D. Performance evaluation

The forecast accuracy measures to be used are the Mean Absolute Error (MAE) (eq. 50), Root Mean Squared Error (RMSE) (eq. 51) and Mean Absolute Percentage Error (MAPE) (eq. 52).

$$\text{MAE} = \frac{1}{N} \sum_{k=1}^{N} |p_{\text{targ}}(k) - p_{\text{fore}}(k)|,$$  

(50)

$$\text{RMSE} = \left(\frac{1}{N} \sum_{k=1}^{N} (p_{\text{targ}}(k) - p_{\text{fore}}(k))^2\right)^{1/2},$$  

(51)

$$\text{MAPE (\%)} = \frac{1}{N} \sum_{k=1}^{N} \left|\frac{p_{\text{targ}}(k) - p_{\text{fore}}(k)}{p_{\text{targ}}(k)}\right| \times 100.$$  

(52)

It is important to note that as $p_{\text{targ}}(k)$ gets closer to 0, the MAPE tends to infinity. Since the normalized data range between 0 and 1, this limitation could be visible on some results.
E. K-Fold Cross-Validation (CV) technique

K-Fold Cross-Validation (CV) is a very useful technique that allows to evaluate how well a method and its learning algorithm generalizes when in the presence of an independent data set.

In K-Fold CV, the initial data set, S, assigned for training purposes is divided into k none-overlapping sub-sets of equal sizes. Each pattern of S set is usually assigned randomly to each sub-set. However, if there is a strong autocorrelation in the time series data, which is this case, the sub-sets are created by dividing S into k blocks with consecutive patterns [16].

After the splitting process, one subset will be used as validation set and the remaining k−1 sub-sets as training set in the training phase. This process is repeated k times, but with each of k subset used exactly once as validation set. Once the K-Fold CV process is completed, the overall performance of the model is computed by averaging the validation RMSE achieved for each sub-set. Besides that, the model that produced the lowest validation RMSE is the chosen one to be used as model in the test phase. To better understand this technique, an example of a 4-Fold CV is presented in Figure 6.

IV. RESULTS AND DISCUSSION

The results presented here in this section stem from the simulations of the seasons of 2014. This means that the 2014 year’s seasons were used as testing sets (from December of 2013 to November of 2014), whereas, as training set, the other years’ season data were used (from January of 2010 to November of 2013) as shown in Figure 7. Therefore, each seasonal training set was split into 4 equal sized sub-sets and a 4-Fold Cross-validation technique was employed.

In Table 4, the performance results (Seasonal MAPE, MAE and RMSE) for each method are presented. The best method results appear in bold for each season.

Table 4 - Wind Power forecast seasonal results

<table>
<thead>
<tr>
<th>Season</th>
<th>Method</th>
<th>MAPE\text{\text{season}} [%]</th>
<th>MAE\text{\text{season}} [pu]</th>
<th>RMSE\text{\text{season}} [pu]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winter</td>
<td>Persistence</td>
<td>9.73</td>
<td>0.0282</td>
<td>0.0369</td>
</tr>
<tr>
<td></td>
<td>ANN</td>
<td>11.46</td>
<td>0.0203</td>
<td>0.0269</td>
</tr>
<tr>
<td></td>
<td>ANFIS</td>
<td>9.38</td>
<td>0.0202</td>
<td>0.0260</td>
</tr>
<tr>
<td></td>
<td>RBFN-SDG</td>
<td>13.72</td>
<td>0.0210</td>
<td>0.0277</td>
</tr>
<tr>
<td></td>
<td>RBFN-Hyb.</td>
<td>18.77</td>
<td>0.0211</td>
<td>0.0279</td>
</tr>
<tr>
<td></td>
<td>RBFN-OLS</td>
<td>7.86</td>
<td>0.0201</td>
<td>0.0270</td>
</tr>
<tr>
<td></td>
<td>Persistence</td>
<td>12.97</td>
<td>0.0237</td>
<td>0.0322</td>
</tr>
<tr>
<td></td>
<td>ANN</td>
<td>9.36</td>
<td>0.0159</td>
<td>0.0216</td>
</tr>
<tr>
<td></td>
<td>ANFIS</td>
<td>9.07</td>
<td>0.0160</td>
<td>0.0216</td>
</tr>
<tr>
<td></td>
<td>RBFN-SDG</td>
<td>10.39</td>
<td>0.0179</td>
<td>0.0247</td>
</tr>
<tr>
<td></td>
<td>RBFN-Hyb.</td>
<td>10.99</td>
<td>0.0169</td>
<td>0.0225</td>
</tr>
<tr>
<td></td>
<td>RBFN-OLS</td>
<td>9.47</td>
<td>0.0160</td>
<td>0.0215</td>
</tr>
<tr>
<td>Summer</td>
<td>Persistence</td>
<td>14.56</td>
<td>0.0228</td>
<td>0.0306</td>
</tr>
<tr>
<td></td>
<td>ANN</td>
<td>9.18</td>
<td>0.0143</td>
<td>0.0192</td>
</tr>
<tr>
<td></td>
<td>ANFIS</td>
<td>8.90</td>
<td>0.0142</td>
<td>0.0192</td>
</tr>
<tr>
<td></td>
<td>RBFN-SDG</td>
<td>10.03</td>
<td>0.0147</td>
<td>0.0196</td>
</tr>
<tr>
<td></td>
<td>RBFN-Hyb.</td>
<td>10.31</td>
<td>0.0157</td>
<td>0.0210</td>
</tr>
<tr>
<td></td>
<td>RBFN-OLS</td>
<td>8.96</td>
<td>0.0142</td>
<td>0.0192</td>
</tr>
<tr>
<td></td>
<td>Persistence</td>
<td>13.97</td>
<td>0.0217</td>
<td>0.0300</td>
</tr>
<tr>
<td></td>
<td>ANN</td>
<td>11.32</td>
<td>0.0156</td>
<td>0.0218</td>
</tr>
<tr>
<td></td>
<td>ANFIS</td>
<td>9.94</td>
<td>0.0155</td>
<td>0.0217</td>
</tr>
<tr>
<td></td>
<td>RBFN-SDG</td>
<td>16.81</td>
<td>0.0172</td>
<td>0.0230</td>
</tr>
<tr>
<td></td>
<td>RBFN-Hyb.</td>
<td>13.20</td>
<td>0.0162</td>
<td>0.0227</td>
</tr>
<tr>
<td></td>
<td>RBFN-OLS</td>
<td>10.00</td>
<td>0.0155</td>
<td>0.0217</td>
</tr>
</tbody>
</table>

From the results in Table 4 some conclusion can be drawn considering individually either the seasons or methods. Starting by comparing each method result along the seasons, in general, the results suggest that the summer is the season where better wind power forecast accuracy is achieved for any kind of method.

Regarding the comparison between the methods for each of the 4 seasons, several conclusions are made. Starting with the winter, the results indicate that RBFN-OLS method was the best one in terms of MAPE and MAE, with 7.86% and 0.0201 pu, respectively. However, the ANFIS also obtained good MAPE and MAE performances with 9.38 % and 0.0202 pu, respectively, being in terms of RMSE, the method that produced less gross errors with 0.0260 pu.
In the spring season, the ANN, ANFIS and RBFN-OLS methods clearly obtained the best performances. The best MAE (0.0159 pu) and RMSE (0.0215 pu) were obtained for ANN and RBFN-OLS, respectively, being only 0.0001 lower for both cases than the following two best methods configuring an insignificant difference. In relative terms, the three methods also obtained the best MAPE results, being the difference between the best one (9.07% for ANFIS) and the third best one (9.47% for RBFN-OLS) about 0.4 %, which is also insignificant.

In the summer, the ANN, ANFIS and RBFN-OLS methods clearly obtained the best MAE and RMSE with 0.0142 pu and 0.0192 pu, respectively, being the MAE for ANN only 0.0001 higher which is practically irrelevant. In relative terms, the three methods also obtained the best MAPE results, being the difference between the best one (8.90% for ANFIS) and the third best one (9.18% for ANN) under 0.3 %, which is also insignificant.

Finally, in the autumn season the same conclusion drawn in the spring and summer season can be also drawn here. The ANN, ANFIS and RBFN-OLS methods were once again the best ones in all fronts. The difference between the best MAE (0.0155 pu for ANFIS and RBFN-OLS) and RMSE (0.00217 pu for ANFIS and RBFN-OLS) to the third best ones (ANN method) is still only 0.0001, which is irrelevant. However, this time in relative terms, the difference between the best one (9.94 % for ANFIS) and the third best one (11.32 % for ANN) is a bit higher than in the previous two seasons, around 1.4 %, which in a seasonal scale has some significance.

To have a different vision from the global results in Table 4, four days corresponding to the four seasons of the year were analyzed (27th January, 27th April, 27th July and 27th October). The daily performances results and graphs with the evolution of the hourly forecast values for all methods can be found in Table 5 and Figure 8, respectively.

### Table 5 – Wind Power forecast daily results

<table>
<thead>
<tr>
<th>Day</th>
<th>Method</th>
<th>MAPE [%]</th>
<th>MAE [pu]</th>
<th>RMSE [pu]</th>
</tr>
</thead>
<tbody>
<tr>
<td>27th January (Winter)</td>
<td>Persistence</td>
<td>3.32</td>
<td>0.0246</td>
<td>0.0310</td>
</tr>
<tr>
<td></td>
<td>ANN</td>
<td>2.54</td>
<td>0.0176</td>
<td>0.0221</td>
</tr>
<tr>
<td></td>
<td>ANFIS</td>
<td>2.23</td>
<td>0.0165</td>
<td>0.0223</td>
</tr>
<tr>
<td></td>
<td>RBFN-SDG</td>
<td>2.50</td>
<td>0.0187</td>
<td>0.0237</td>
</tr>
<tr>
<td></td>
<td>RBFN-Hyb</td>
<td>2.82</td>
<td>0.0213</td>
<td>0.0253</td>
</tr>
<tr>
<td></td>
<td>RBFN-OLS</td>
<td>2.76</td>
<td>0.0167</td>
<td>0.0220</td>
</tr>
<tr>
<td>27th April (Spring)</td>
<td>Persistence</td>
<td>10.11</td>
<td>0.0276</td>
<td>0.0328</td>
</tr>
<tr>
<td></td>
<td>ANN</td>
<td>7.33</td>
<td>0.0209</td>
<td>0.0242</td>
</tr>
<tr>
<td></td>
<td>ANFIS</td>
<td>7.26</td>
<td>0.0206</td>
<td>0.0238</td>
</tr>
<tr>
<td></td>
<td>RBFN-SDG</td>
<td>7.91</td>
<td>0.0223</td>
<td>0.0254</td>
</tr>
<tr>
<td></td>
<td>RBFN-Hyb</td>
<td>7.81</td>
<td>0.0224</td>
<td>0.0257</td>
</tr>
<tr>
<td></td>
<td>RBFN-OLS</td>
<td>7.33</td>
<td>0.0208</td>
<td>0.0243</td>
</tr>
<tr>
<td>27th July (Summer)</td>
<td>Persistence</td>
<td>21.89</td>
<td>0.0159</td>
<td>0.0197</td>
</tr>
<tr>
<td></td>
<td>ANN</td>
<td>14.53</td>
<td>0.0110</td>
<td>0.0148</td>
</tr>
<tr>
<td></td>
<td>ANFIS</td>
<td>13.60</td>
<td>0.0108</td>
<td>0.0148</td>
</tr>
<tr>
<td></td>
<td>RBFN-SDG</td>
<td>16.35</td>
<td>0.0127</td>
<td>0.0160</td>
</tr>
<tr>
<td></td>
<td>RBFN-Hyb</td>
<td>16.55</td>
<td>0.0118</td>
<td>0.0161</td>
</tr>
<tr>
<td></td>
<td>RBFN-OLS</td>
<td>14.09</td>
<td>0.0108</td>
<td>0.0149</td>
</tr>
<tr>
<td>27th October (Autumn)</td>
<td>Persistence</td>
<td>23.39</td>
<td>0.0115</td>
<td>0.0173</td>
</tr>
<tr>
<td></td>
<td>ANN</td>
<td>38.61</td>
<td>0.0066</td>
<td>0.0085</td>
</tr>
<tr>
<td></td>
<td>ANFIS</td>
<td>21.85</td>
<td>0.0054</td>
<td>0.0076</td>
</tr>
<tr>
<td></td>
<td>RBFN-SDG</td>
<td>78.05</td>
<td>0.0079</td>
<td>0.0093</td>
</tr>
<tr>
<td></td>
<td>RBFN-Hyb</td>
<td>61.66</td>
<td>0.0076</td>
<td>0.0096</td>
</tr>
<tr>
<td></td>
<td>RBFN-OLS</td>
<td>25.75</td>
<td>0.0055</td>
<td>0.0076</td>
</tr>
</tbody>
</table>

Comparing the results in Table 4 with the ones in Table 5, the first thing that becomes clear is the major difference between the seasonal MAPE and the daily MAE.

Although the winter season has been the one with the highest seasonal MAPE value (RBFN-Hybrid: 18.77 %), on 27th January day for the same method the MAPE was only 2.82 % and not much far from the other methods. This fact is simply explained by the high wind power generation that occurred on this day, where the hourly wind power values ranged approximately between 0.65 and 0.85 pu (Figure 8). Since these values are relatively far from 0 the limitation of the MAPE enunciated in II - D. is not featured here.

On 27th July and 27th October the MAPEs for all methods were relatively high comparing to the seasonal MAPEs. This happens because, in contrast to what happened on 27th January, the 27th day of July was marked by the low wind power generation (the hourly wind power values fell below 0.025 pu in 11th, 12th and 13th hours), as well as on 27th October, where the values were very closer to 0 on the first 8 hours of the day. Since these values are relatively closer to 0 the limitation of the MAPE is somehow affecting the MAPE results.

Another important aspect in these results is the predominance of the ANFIS method as the best one among all performing wind power forecasts for those days. In fact, the ANFIS method obtained the best MAPE, MAE and RMSE on 27th April with 7.26 %, 0.0206 pu and 0.0238 pu, respectively, on 27th July, with 13.60 %, 0.0108 pu and 0.0148 pu, respectively and on 27th October, with 21.85 %, 0.0054 pu and 0.0076 pu, respectively. On 27th January, the ANFIS method was also the best one, but only in terms of MAPE and MAE, with 2.23% and 0.0165 pu, respectively. In terms of gross errors committed, the RBFN-OLS performed better with a RMSE of 0.0220 pu. The proof of the previous statement is reflected in the Figure 8 in the 7th, 8th and 11th hours. The RBFN-OLS curve had a much better behavior in rapid variation situations. The ANN and ANFIS curves also behaved well, though.

Besides that, in general, the RBFN-SDG and RBFN-Hybrid methods were the ones with more gross errors committed. That is confirmed by the RMSE results in the Table 5, and it is also visible in Figure 8, for example, on the first 8 hours of the 27th day of October, where the forecast values are relatively higher than the real wind power generated.

Another conclusion that can be drawn by looking at Figure 8 is that none of the AI methods managed to obtain a good wind power forecast when in the presence of peaks and valleys situations. One example of that particular situation can be spotted on the 21st hour of the 27th day of July, where the persistence method performed better than any other method.

### V. Conclusion

Being the ultimate goal of this work to identify among the implemented methods the one that produce the most accurate wind power generation predictions, a main conclusion can be drawn basing on the results. Although the ANN, ANFIS and RBFN-OLS methods have performed well, the ANFIS was clearly the best method among all performing one-hour-ahead wind power generation forecasts for all seasons. Its best performance was achieved in summer, which was also the best
season, with a MAPE, MAE and RMSE of 8.90 %, 0.0142 pu (37.72 % of improvement relative to the persistence method) and 0.0192 pu, respectively.

REFERENCES