Professor Ion LUNGU, PhD
E-mail: ion.lungu@ie.ase.ro
Professor Adela BÂRA, PhD
E-mail: bara.adela@ie.ase.ro
The Bucharest Academy of Economic Studies
Associate Professor George CĂRUTASU, PhD
E-mail: georgecarutasu@yahoo.com
The Romanian-American University
Lecturer Alexandru PÎRJAN, PhD
E-mail: alex@pirjan.com
The Romanian-American University
Assistant Lecturer Simona-Vasilica OPREA, PhD
E-mail: simona.oprea@csie.ase.ro
The Bucharest Academy of Economic Studies

PREDICTION INTELLIGENT SYSTEM IN THE FIELD OF RENEWABLE ENERGIES THROUGH NEURAL NETWORKS

Abstract: *In this paper, we have developed a series of neural networks in order to design a decision support system for predicting, analysing and monitoring the performance indicators in the field of renewable energies in Romania. We have first analysed a series of comparative aspects regarding the algorithms used for developing the neural networks: the Levenberg-Marquardt, the Bayesian Regularization and the Scaled Conjugate Gradient algorithms. Then, we have developed, trained, validated and tested several neural networks based on the above-mentioned algorithms, using the Neural Network Toolbox from the development environment MatlabR2015a. Thus, we have obtained a solution that forecasts the total active energy export and the total active power, when knowing the solar irradiation, the ambient temperature, the humidity, the wind direction and the wind speed.*

Keywords: *Neural Networks, algorithms, renewable energy, solar power plant, Decision Support System.*

JEL Classification: C01, C53, C15, L86, O13, Q42, Q47

1. Introduction

Nowadays, in many European countries including Romania, it is necessary to encourage the usage of renewable energy resources as they offer a wide range of advantages [1]. Thus, the security of energy supply and the conservation of the traditional resources are ensured while the imports of primary energy resources are reduced significantly. Moreover, the usage of renewable energy stimulates the economic development at local, regional and global levels and creates new employment opportunities [2], [3]. A major advantage of using renewable energy is related to reducing the environmental pollution [4], [5].

Currently, at the national level, in the power plants that produce renewable energy, the renewable resource management is not sustained by a decision support system (DSS) that could enable the efficient monitoring and analysis of the energy produced from these sources. In many other European countries, there are decision support systems for the efficient management of electricity generation (e.g. Germany, Spain). Unfortunately, the developing costs of such systems are pretty high and their implementation in Romania must take into account the national specific.

The development of a DSS poses to the developers certain issues and risks. These are related to the fact that data is coming from heterogeneous sources, or that the predictions and notifications might be inaccurate as, unfortunately, at the national and international level the existing methods or systems provide a prediction error higher than 30% for Wind Power Plants (WPP) and 15% for Solar Power Plants (SPP) [2], [3]. A DSS for the prediction, analysis and monitoring the technological and business processes in the field of renewable energy in Romania can be developed using different approaches. Due to their undeniable advantages, in this paper we have chosen to develop a series of Neural Networks.

The Artificial Neural Networks (ANN) offer many advantages in the cases of statistical, stochastic and deterministic approaches. One of the most important advantages of the neural networks consists in the fact that the networks do not rely on a priori mathematical model, as in the case of other approaches; the results depend on the data and a model can be recognized without learning the definitions.

They are used in numerous applications, that include models for approximation, optimization, systems modelling and pattern recognition systems. The artificial neural networks are frequently used for engineering, economical modelling [6] or medicine [7].

We have first used stochastic methods, then we analysed a series of comparative aspects regarding the ANN algorithms: the Levenberg-Marquardt, the Bayesian Regularization and the Scaled Conjugate Gradient algorithms. We have developed, trained, validated and tested several neural networks based on the above-mentioned algorithms, using the Neural Network Toolbox from the development environment MatlabR2015a.

2. Prediction models for Solar Power Plant using stochastic methods

In order to develop the proposed solution, we have identified the input parameters (the solar irradiation, the ambient temperature, the humidity, the wind direction, the wind speed), along with the corresponding outputs (the total active energy export, the total active power). We have gathered the experimental data corresponding to the input and output parameters, resulting in a total number of 50,631 samples, through measurements conducted over a one-year period (from the

1st of January to the 31st of December 2014), from 10 to 10 minutes, in a solar power plant located in the Giurgiu County, in Romania.

The solar power plant where we have conducted the measurements comprises photovoltaic panels having two models of invertors: 600 kW and 760 kW PSV800, manufactured by the ABB Company. The ABB central inverters provide a high level of performance based on high efficiency, low auxiliary power consumption, together with a verified reliability and an experienced worldwide service organization. The inverters are ranging from 100 kW up to 1000 kW, being optimized for multi-megawatt photovoltaic power plants. Since the inverters implemented in the solar power plant where we have conducted the measurements are commonly used, the method that we have developed can be adapted easily to the cases of other solar power plants.

In the first attempts to develop the prediction solution, we have tested various other software products that made use of stochastic methods. We applied first-order and second-order autoregressive method (AR(1) and AR(2)), first-order and second-order moving average (MA(1) and MA(2)), mixed model ARMA and auto regressive integrated moving average (ARIMA).

The methods returned good results with acceptable errors, except the results from the months of January, February and December, cases in which the errors were high **(Figure 1)**.

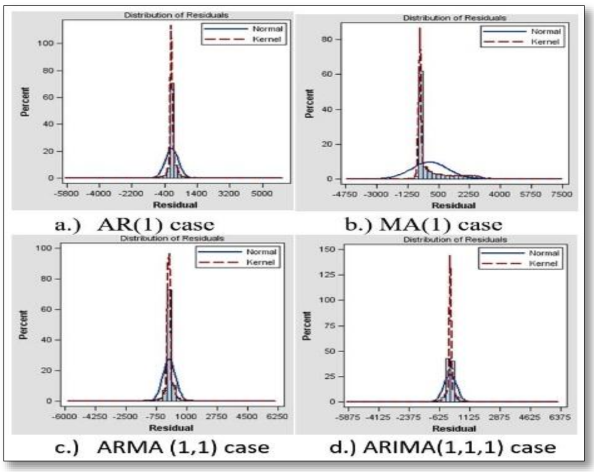


Figure 1. The error histograms obtained for the month of March using stochastic methods

The best results were registered for AR(1), ARMA(1,1) and ARIMA models, for both data sets: whole year and each month (**Table 1**).

Table 1. A comparison analysis of the results provided by the stochastic methods

Whole year	**MSE**	**R**	**March**	**MSE**	**R**
AR(1)	0.096505	0.989558	AR(1)	0.098276	0.989385
AR(2)	0.113464	0.978219	AR(2)	0.117567	0.977347
MA(1)	0.194253	0.948153	MA(1)	0.218928	0.948439
MA(2)	0.202734	0.923199	MA(2)	0.231024	0.916695
ARMA(1,1)	0.096478	0.989582	ARMA(1,1)	0.098237	0.989438
ARMA(2,2)	0.113297	0.978492	ARMA(2,2)	0.117486	0.977452
ARIMA(1,1,1)	0.096372	0.989939	ARIMA(1,1,1)	0.098257	0.989453
ARIMA(2,1,2)	0.096435	0.989787	ARIMA(2,1,2)	0.098294	0.989398

Although these methods provided satisfactory results, they had as a main drawback the fact that they did not allow the setting of more output parameters, but only one.

3. Developing a series of neural networks for predicting Solar Power Plant energy output in Romania

We have obtained the best results when developing our prediction solution through neural networks. First, we developed three neural networks for the annual data, one for each of the used algorithms: Levenberg-Marquardt (LM), Bayesian Regularization (BR) and Scaled Conjugate Gradient (SCG). In this case, we have obtained acceptable prediction results.

Taking into account the seasonal meteorological conditions, we improve the prediction accuracy by developing the neural networks for each month of the year, corresponding to each of the proposed algorithms. Thus, we have developed, trained and tested 39 neural networks for prediction and we have validated 26 of them (as for the networks trained using the Bayesian Regularization algorithm this step does not occur). We have noticed that through this method we have obtained improved prediction results than when using a single global network for the entire year.

After a series of tests, we have decided to develop the networks according to the following architecture that has proven to offer the best performance regarding the prediction accuracy: 5 neurons for the Input data, 10 in the Hidden layer, 2 in the Output layer and 2 for the Output data (**Figure 2**).

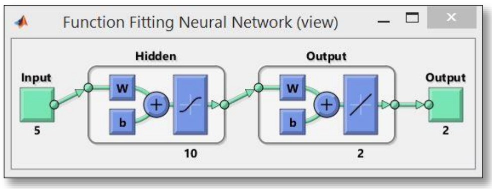
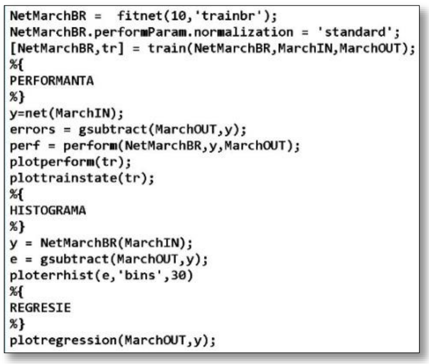


Figure 2. The Architecture of the Neural Networks

In order to train, validate and test the neural networks developed using the LM and the SCG algorithms, we have divided the data set in the following way: 70% of it for the training process, 15% for the validation process and the remaining 15% for the testing process. In order to train and test the neural networks developed using the BR algorithm, we have divided the data set in the following way: 70% of it for the training process, 15% for the testing process and the remaining 15% was not allocated (in order to obtain a relevant comparison of the final results achieved by the three algorithms). In all the cases, the samples have been randomly chosen as to cover the specified percentages. In order to train the neural networks, we have used the mean square error (MSE) as an objective function. When training a network with this function, if there are multiple outputs having different ranges of values, the accuracy is optimized for the output element that has a wider range of values and is less optimized relative to the output element with a smaller range of values. Thus, the network will learn to fit the first output element very well, while the second output element is not fit as accurate as the first. In order to solve this issue, we have normalized the errors, by setting the normalization performance parameter to its 'standard' value. By using this method, the errors have been computed as if both of the output elements had values ranging from -1 to 1 and consequently, the two output elements have been fitted very well (**Figure 3**).



```
NetMarchBR =  fitnet(10,'trainbr');
NetMarchBR.performParam.normalization = 'standard';
[NetMarchBR,tr] = train(NetMarchBR,MarchIN,MarchOUT);
%{
PERFORMANTA
%}
y=net(MarchIN);
errors = gsubtract(MarchOUT,y);
perf = perform(NetMarchBR,y,MarchOUT);
plotperform(tr);
plottrainstate(tr);
%{
HISTOGRAMA
%}
y = NetMarchBR(MarchIN);
e = gsubtract(MarchOUT,y);
ploterrhist(e,'bins',30)
%{
REGRESIE
%}
plotregression(MarchOUT,y);
```

Figure 3. The source code for developing one of the Neural networks and computing its performance parameters

In the following, we present some experimental results that we have obtained, along with the consequent performance analysis of the neural networks.

3.1. Using Levenberg-Marquardt Algorithm

The Levenberg–Marquardt Algorithm (LM) is an artificial neural network (ANN) training algorithm, used in mathematics and computing. It is also known as the Damped Least-Squares (DLS) method and is useful in solving non-linear least squares problems. The LM algorithm is a curve fitting algorithm, whose purpose is to build a mathematical function or curve that matches a set of data that can be subjected to some constraints. LM is used in a wide range of applications related to curve-fitting problems, but like other fitting algorithms, it provides a local minimum and not a global one [16].

In the case of the curve-fitting problems, the least squares method consists in minimizing the sum of squared errors related to a parameterized function that attempts to match a set of data. If the function does not depend linearly on the parameters, one obtains the nonlinear least squares problems. In the nonlinear case, the method consists in the iterative improving of the parameters, in order to minimize the sum of squared errors. The Levenberg-Marquardt method is actually a combination of two minimization methods: the gradient descent method and the Gauss-Newton method [17].

Each of these methods is characterized by the manner in which the sum of the squared errors is reduced. In the gradient descent method, the reduction is achieved by updating the parameters as to obtain the greatest reduction of the least squares objective function. In the Gauss-Newton method, the reduction is achieved by assuming that the least squares function is locally quadratic and afterwards the minimum of the quadratic is computed. The Levenberg-Marquardt method combines the previous methods as follows: when the parameters are far from their optimal value, the LM algorithm behaves like a gradient-descent method; when the parameters are close to their optimal value, the LM algorithm behaves like the Gauss-Newton method [8].

The nonlinear least squares minimization problems' purpose is to minimize a function f of the form:

$$f(x) = \frac{1}{2}\|r(x)\|^2 \quad (1)$$

where $x = (x_1, x_2, \dots, x_n)$ is an n-dimensional real array, $r(x) = (r_1(x), r_2(x), \dots, r_m(x))$ is an array whose components are functions, defined on $\mathbb{R}^n$ and having real values, while $m \geq n$. The r_k functions are called residuals and the function f could also be written as:

$$f(x) = \frac{1}{2}[r_1^2 + r_2^2 + \dots + r_m^2] = \frac{1}{2}\sum_{k=1}^{m} r_k^2 \quad (2)$$

We denote by J the Jacobian matrix of r:

$$J(x) = \frac{\partial r_k}{\partial x_i}, 1 \leq k \leq m, 1 \leq i \leq n \quad (3)$$

In the linear case, the r_k functions are linear for each $1 \leq k \leq m$ and therefore each of these functions could be written in the following form:

$$r_k = \frac{\partial r_k}{\partial x_1}x_1 + \frac{\partial r_k}{\partial x_2}x_2 + \dots + \frac{\partial r_k}{\partial x_n}x_n + r_k(0), 1 \leq k \leq m \quad (4)$$

while the function f takes the form:

$$f(x) = \frac{1}{2}\|Jx + r(0)\|^2 \tag{5}$$

and

$$\nabla f(x) = J^T(Jx + r), \nabla^2 f(x) = J^T J \tag{6}$$

The necessary condition for the function $f(x)$ to have a local minimum at a point x_{min} is that $\nabla f(x) = 0$. From this condition we obtain

$$x_{min} = -(J^T J)^{-1} J^T r \tag{7}$$

In the non-linear case,

$$\nabla f(x) = \sum_{k=1}^{m} r_k(x) \nabla r_k(x) = J(x)^T r(x) \tag{8}$$

and

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{k=1}^{m} r_k(x) \nabla^2 r_k(x) \tag{9}$$

If the Jacobian matrix J is known and either r_k could be approximated by linear functions, or the residuals r_k are very small, then the Hessian $\nabla^2 f(x)$ could be written in the same form as in the linear case:

$$\nabla^2 f(x) = J^T J \tag{10}$$

If r_k are being approximated by linear functions then $\nabla^2 r_k(x)$ are small. As the assumption that the residuals are small was also used, it is important to remark that when dealing with large residual problems, the above mentioned quadratic approximation and the formula (10) are inappropriate and therefore the performance of some algorithms becomes poor.

The gradient-descent method is an intuitive technique for obtaining the minimum of a function. Thus, the minimum is obtained through an iteration in which, at each step, the gradient of the function, multiplied by a negative parameter is added:

$$x_{i+1} = x_i - \mu \nabla f \tag{11}$$

Taking into account that the method faces numerous convergence problems [8], different approaches can be used in order to improve its convergence. For example, the Newton's method proposes to solve the equation $\nabla f(x) = 0$, using the Taylor's formula for expanding the gradient in series around a certain state x_0, considered to be the current state:

$$\nabla f(x) = \nabla f(x_0) + (x - x_0)^T \nabla^2 f(x_0) + \text{terms of higher order} \tag{12}$$

If f is considered to be quadratic around x_0, the terms of higher order are negligible. Considering the necessary condition for a local minimum $\nabla f(x) = 0$, replacing x_0 by x_i and x by x_{i+1}, one obtains an improved version of the equation (11):

$$x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \nabla f(x_i) \tag{13}$$

As we have mentioned before, the Newton's method is based on the quadratic approximation of f and therefore, the Hessian matrix's exact computation can be avoided, being replaced by the approximation (10). The above depicted method has certain advantages regarding its rapid convergence, but it also has some disadvantages regarding the linearity around the starting location.

Taking into account the advantages and disadvantages of the gradient-descent method and of the Gauss-Newton iteration depicted above, Levenberg proposed a new method, with an update rule that was built as a blend of the two algorithms:

$$x_{i+1} = x_i - (H + \mu I)^{-1} \nabla f(x_i) \tag{14}$$

where $\mathrm{H} = \nabla^2 f(x_i)$ is the hessian matrix computed in x_i.

The update rule (14) is used taking into account the error. If, after an update, the error lowers, this means that the quadratic assumption is proper and in the next step, the parameter μ is adjusted by reducing it 10 times in order to reduce the influence of the gradient descent. Otherwise, if the error increases, this means that the influence of the gradient should increase, so the parameter μ is adjusted by increasing it 10 times [8].

As mentioned before, in the non-linear case, the Hessian matrix can be approximated as in the linear case (10), while the gradient is computed through (8). The $r(x)$ vector contains the network errors and the Jacobian matrix contains the derivatives of the network errors. Within the Levenberg–Marquardt algorithm, the Jacobian matrix is computed using a back-propagation technique, thus avoiding the computation of the Hessian matrix. The Newton update method (14) can also be written as:

$$x_{i+1} = x_i - (J^T J + \mu I)^{-1} J^T(x_i) r(x_i) \tag{15}$$

where $H = J^T J$ is the hessian matrix computed in x_i and $\nabla f(x_i) = J^T(x_i) r(x_i)$.

Through the values of the parameter μ, one can adjust the method. Thus, $\mu = 0$ corresponds to the Newton's method, while large values of μ correspond to the gradient descent method with a small step. By lowering or increasing the values of the parameter μ, the objective function is reduced at each step of the algorithm. Thus, the Levenberg–Marquardt Algorithm combines the performance of the gradient-descent method with the Gauss-Newton iteration, being more powerful than both of these methods.

Taking into account the undeniable advantages of the Levenberg–Marquardt Algorithm, we have decided to implement it in our research, by developing, training, validating and testing a neural network for each month of the year.

For the month of March, we have first analysed the validation performance and the forecasting accuracy, using the NetMarchLM Network, developed, trained, validated and tested using the Levenberg-Marquardt algorithm. We have obtained the best validation performance at the 90th epoch, having the MSE value of 0.061763. The solution is net superior to the one when a single neural network is trained for the whole year, using the LM algorithm (NetGlobalLM) that has the best performance of 0.20196 for the MSE, obtained at the 109th epoch. Thus, the solution for the month of March brings an improvement of 69% when compared to the global one. The graphic confirms a high degree of performance and accuracy, as the validation, the training and the test functions are very similar for the LM algorithm. A very important issue that we have taken into consideration when analysing the results was to study the test curve and compare it to the validation one. We have analysed the test curve and verified if it had increased significantly before the validation curve did, case in which the overfitting process might have occurred. In the analysed case, we have noticed that this phenomenon did not occur, which reflects the fact that we have divided appropriately the data sets and conducted efficiently the network training. Afterwards, we have represented the error histogram, when considering the forecasting for the month of March, using the NetMarchLM Network, developed, trained, validated and tested using the Levenberg-Marquardt algorithm (**Figure 4**).

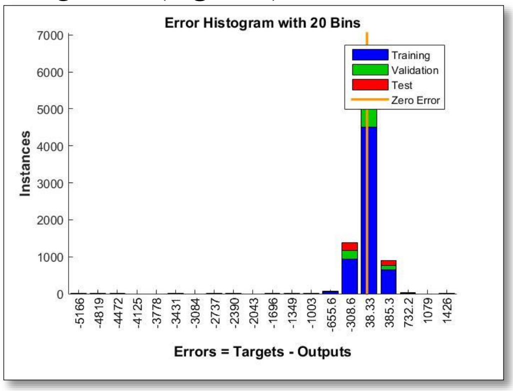


Figure 4. The error histogram when considering the forecasting for the month of March, using the NetMarchLM Network

We have noticed that while most of the errors fall between -308.6 and 385.3, there are a few training points with errors that fall outside this range, even if by using the above-depicted methods we have minimized the number of the outliers. When using a single neural network for the entire year, in the case of the LM algorithm, we have observed that in this case most of the errors fall between -3506 and 3983, but there also remain many training points with errors that fall outside this range. The range of errors for the entire year is much wider than the one for the month of March. The error histogram when considering the forecasting for the month of March, using the NetMarchLM Network developed, trained, validated and tested using the Levenberg-Marquardt algorithm, confirms the fact that the results are very good, taking into account that the forecasting must be performed for two output parameters that have different orders of magnitude, as follows: the total active power varies between -40.63 and 7466.74 kW and the total active energy export varies between 9300 and 18300 MWh. The obtained results are very good and confirm the usefulness and efficiency of the normalization process, that we have previously described. Afterwards, in order to validate the network, we have computed and represented the regressions between the network targets and network outputs, when considering the forecasting for the month of March, using the NetMarchLM Network, developed, trained, validated and tested using the Levenberg-Marquardt algorithm (**Figure 5**).

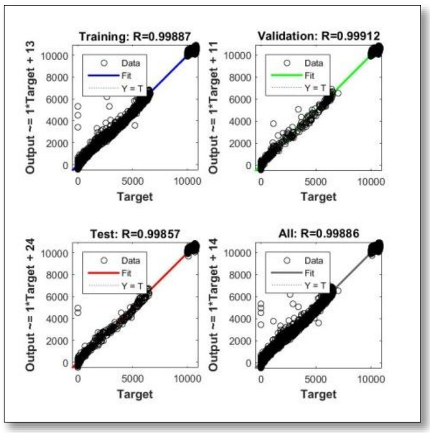


Figure 5. The regressions between the network targets and network outputs when considering the forecasting for the month of March

The values of the correlation coefficient R that we have obtained reflect a very good fit, all of them being greater than 0.998. When we have developed, trained, validated and tested a neural network for the entire year, using the NetGlobalLM Network, we have obtained lower values for the correlation coefficient R, even if all the values are greater than 0.957. Even if the results were also good in the yearly approach, the difference between the monthly results (highlighted through the month of March) and the yearly ones reflects the difference between a very good fit and a good fit of the prediction results.

3.2. Using Regularization Algorithm

The Bayesian Regularization (BR) Algorithm targets the minimization of a function that contains a linear combination of squared weights and squared errors. The BR algorithm modifies this linear combination in such a way that when the network training ends, the obtained network has improved generalization qualities [9], [10]. The Bayesian Regularization is based on the Levenberg-Marquardt algorithm and also on the backward propagation of errors (back-propagation) that is used to compute the Jacobian J of the objective function with respect to the variables (weights and biases). Each variable is being adjusted according to the Levenberg-Marquardt algorithm and an adaptive value is used, being increased until a reduced value of the objective function is obtained. In that moment, the changes are applied to the network and the adaptive value is decreased.

The BR algorithm developed by David MacKay allows estimating the total number of parameters used by the model and thus, the number of network weights used for solving a certain problem. The BR algorithm introduces two Bayesian hyperparameters (α and β) whose purpose is to tell whether the learning process must seek in the direction of the minimal errors or in that of the minimal weights. Thus, the cost function can be written as:

$$C(i) = \alpha \cdot S_w + \beta \cdot S_e \qquad (16)$$

where S_w is the sum of all the squared weights and S_e is the sum of all the squared errors.

One of the main advantages of the Bayesian Regularization algorithm is that it avoids certain costs related to the validation procedures. For some problems, it is not possible to reserve a portion of data in order to achieve the validation. Through the Bayesian Regularization algorithm, these situations are avoided. Another advantage of this algorithm consists in the fact that one can reduce or even eliminate the need for testing various numbers of hidden neurons. By implementing a third variable, γ, one is able to control the influence of the weights that are used by the network and thus, it is possible to obtain information regarding the complexity of the network.

In many cases, the Bayesian Regularization implementations update the hyperparameters after each of the training cycles. However, in many cases, these updates produce weak iterations. Therefore, various methods to update the parameters have been developed, based on computing the inverse Hessian matrix.

Briefly, the Bayesian Regularization algorithm starts from computing the Jacobian J (by finite differences or using the chain rule) and then, the error gradient:

$$g = \nabla f(x) = J^T E \qquad (17)$$

In the next step, the Hessian matrix is approximated as

$$\mathrm{H} = J^T J \qquad (18)$$

The cost function is computed through the formula (16), then the equation

$$(H + \lambda I)\delta = g \qquad (19)$$

is solved in order to find δ. The values of δ are used in order to update the network's weights w and then, using the updated weights, the cost function is computed again. If the cost function has not decreased, the updated weights are discarded and λ is multiplied with an increasing factor v. If the cost function has decreased, λ is divided with a decreasing factor v. In the next step, the Bayesian hyperparameters α and β are updated, using one of the different existing approaches [8].

Throughout the time, the Bayesian Regularization algorithm has proved to be an extremely fast method for neural network learning when compared to other algorithms (like the standard back-propagation algorithm).

This is the reason why we have decided to research the possibility of developing, training and testing neural networks that implement it and analyse how well this networks can be used in forecasting critical parameters regarding the production of the solar renewable energy.

Afterwards, we have analysed the training performance when considering the forecasting for the month of March, using the NetMarchBR Network, developed, trained and tested using the Bayesian Regularization algorithm (**Fig. 6**).

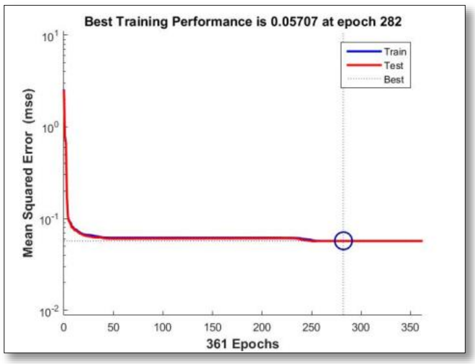


Figure 6. The training performance when considering the forecasting for the month of March, using the NetMarchBR Network

The best training performance that we have obtained was at the 282th epoch, when the MSE had the value of 0.05707. This solution is net superior to the one obtained by training a single neural network for the whole year, using the BR algorithm (NetGlobalBR) when the best performance was obtained at the 229th epoch and was $MSE = 0.19806$. As a consequence, the solution for the month of March brings an improvement of 71% when compared to the global one. The graphic confirms a high degree of performance and accuracy, as the training and the test functions are very similar for the BR algorithm. The overfitting process does not occur, as the test curves do not increase significantly before the training curves do.

Afterwards, we have represented the error histogram, when considering the forecasting for the month of March, using the NetMarchBR Network, developed, trained and tested using the Bayesian Regularization algorithm. Throughout the histogram, the red bars represent testing data and the blue bars represent training data. Thus, we have obtained information regarding the outliers. In the Bayesian Regularization case, there are a few training points with errors that fall outside the range -396.9 and 291 where most of the errors fall, even if we have reduced as much as possible the number of the outliers using the above-depicted methods. In the case of the BR algorithm, when training a single neural network for the entire year, we have noticed that most of the errors fall between -3440 and 3698, but there also remain many training points with errors ranging outside this interval.

The error histogram in this case, reflects very good results, in view of the fact that the forecasting has to be performed for two output parameters with different orders of magnitude: the total active power varies between - 40.63 and 7466.74 kW and the total active energy export varies between 9300 and 18300 MWh. The values of the correlation coefficient R are all greater than 0.998 reflecting a very good fit. The correlation coefficients R that we have obtained when we have developed, trained and tested a neural network for the entire year, using the NetGlobalBR Network, are all greater than 0.956. The results reflected a good fit of the prediction in the yearly approach and a very good fit in the case of the monthly approach (reflected through the month of March).

3.3. Using Scaled Conjugate Gradient algorithm

In 1993, Martin Moller introduced a supervised learning algorithm entitled the Scaled Conjugate Gradient (SCG), useful for developing and training neural networks. The SCG algorithm is based on the conjugate gradient methods [11]. The algorithm has certain indisputable advantages when compared to previous algorithms belonging to the same class, as the Rumelhart's standard back-propagation algorithm [12], the Johansson's conjugate gradient algorithm with line search [13] or the Battiti's one-step quasi-Newton algorithm [14].

The SCG algorithm is fully-automated, preventing the usage of user-dependent parameters that can become critical and also has the advantage of avoiding some of the time-consuming procedures used by the previous algorithms when determining the step size (the length of the weight update). In this stage, other algorithms require a line search for each iteration, which is a resource consuming process as for each search the network's response must be computed several times. The SCG algorithm avoids this line search by combining the conjugate gradient approach and the Levenberg-Marquardt's approach of model-trust region. As in the case of many other conjugate gradient methods, the SCG is based on the conjugate directions, but taking into account its performance and especially the fact that it does not perform a line search at each iteration, the Moller's implementation is faster than other conjugate gradient methods.

The backward propagation of errors or the back-propagation, represents a useful method for training artificial neural networks [15]. It is used in conjunction with optimization methods, for example with the gradient descent. By using this method, the gradient of the objective function with respect to the weights of the network is computed. The purpose is to minimize the objective function. The backward propagation of errors method requires a set of input values and their corresponding outputs, based on which the gradient of the objective function is computed. The weights of the network are adjusted in the direction that makes the objective function decrease most rapidly (the steepest descent directions) and this happens along the negative of the gradient. However, even if in this case the objective function decreases faster, this is not the fastest possible convergence. The conjugate gradient algorithms perform the search along the conjugate gradient directions of the previous steps, minimizing the objective function along these directions and improving the convergence of the general back-propagation method. In this way, a minimization performed in a certain step is not undone in the following one, as it happens in other cases.

Being second order techniques, the conjugate gradient methods aim to minimize functions of several variables by using their second derivatives, in contrast with the back-propagation methods that are first-order techniques and make use of the first derivatives of the objective functions. The methods of obtaining the local minimum of the objective functions, based on the second derivatives, impose increased computational costs but they do have certain advantages over the first order methods.

The Scaled Conjugate Gradient trains any network, with the condition that the involved elements (associated weight, net input, transfer functions) are derivable functions. As in the case of all the conjugate gradient methods, for the SCG the first iteration searches in the steepest descent direction $p_0 = -g_0$ [8], but then a line search is performed in order to determine the optimal distance that is required for moving along the current search direction:

$$x_{i+1} = x_i + \alpha_i g_i \tag{20}$$

The second search direction is conjugated to the previous one. Generally, for each step, the searching direction is obtained by combining the previous direction used for searching with the new steepest descent direction:

$$p_i = -g_i + \beta_i g_{i-1} \tag{21}$$

The way in which the β_i parameters are computed is different from one version of the conjugate gradient algorithm to another.

In our case, we have analysed the validation performance when considering the forecasting for the month of March, using the NetMarchSCG Network, developed, trained, validated and tested using the Scaled Conjugate Gradient algorithm. The best validation performance was obtained when the MSE was 0.11745 and we have obtained it at the 96th epoch. The solution of using a neural network for the month of March is net superior to the solution that trains a single neural network for the data set belonging to the whole year, using the SCG algorithm (NetGlobalSCG) that has the best performance of 0.23566 for the MSE, obtained at the 144th epoch. Thus, the solution for the month of March brings an improvement of 50% when compared to the global one. In this case, as in the previous ones, the over fitting process does not occur.

Analyzing the histogram, most of the errors fall between -275.5 and 402.4, but even if we have minimized the number of the outliers using the above-depicted methods, there are a few training points with errors that fall outside this range. We have observed that when using a single neural network for the entire year, in the case of the SCG algorithm, most of the errors fall between -4610 and 5083, but many training points having errors that fall outside this range also remain. The range of errors in the case of training a single network for the entire year has a much wider interval than the one obtained for the month of March.

In the case of the NetMarchSCG Network developed, trained, validated and tested using the Scaled Conjugate Gradient algorithm for the month of March, the error histogram highlights very good results, taking into account the different orders of magnitude of the two output parameters: the total active power that varies between - 40.63 and 7466.74 kW and the total active energy export that varies between 9300 and 18300 MWh.

For the SCG algorithm, we have obtained a very good fit as the values of the correlation coefficient R are all greater than 0.997 (**Figure 7**).

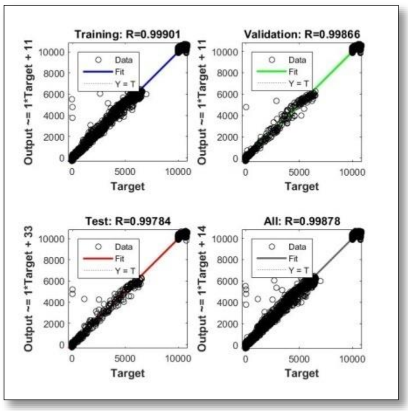


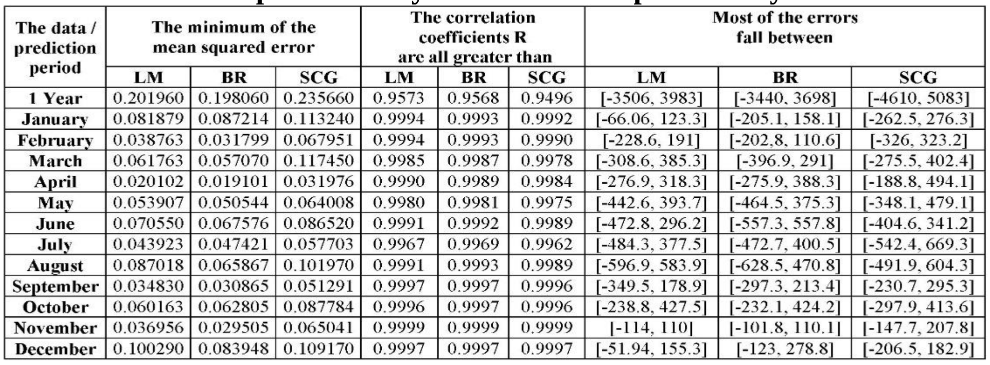
Figure 7. The regressions between the network targets and network outputs when considering the forecasting for the month of March, using the NetMarchSCG

Analysing the results obtained when we have developed, trained, validated and tested a neural network for the entire year, using the NetGlobalSCG Network, we have noticed that we have obtained lower values for the correlation coefficient R, which are all greater than 0.949. The results were good in the yearly approach, but were very good in the monthly one (as reflected by the month of March).

3.4. A comparison analysis of the results provided by ANN algorithms

We have obtained two solutions that forecast the total active energy export and the total active power, when knowing the solar irradiation, the ambient temperature, the humidity, the wind direction and the wind speed. We have developed, trained, validated and tested several neural networks based on three algorithms: the LM, the BR and the SCG. The first solution is based on developing a single neural network for the entire year for each of the proposed algorithms, thus obtaining 3 neural networks. The second solution is based on developing 12 different neural networks, one per month for each of the algorithms, thus resulting in 36 neural networks. In the following we analyse the forecasting performance provided by the 39 neural networks that we have developed (**Table 2)**.

Table 2. A comparison analysis of the results provided by the 39 NN



The data / prediction period	The minimum of the mean squared error			The correlation coefficients R are all greater than			Most of the errors fall between		
	LM	BR	SCG	LM	BR	SCG	LM	BR	SCG
1 Year	0.201960	0.198060	0.235660	0.9573	0.9568	0.9496	[-3506, 3983]	[-3440, 3698]	[-4610, 5083]
January	0.081879	0.087214	0.113240	0.9994	0.9993	0.9992	[-66.06, 123.3]	[-205.1, 158.1]	[-262.5, 276.3]
February	0.038763	0.031799	0.067951	0.9994	0.9993	0.9990	[-228.6, 191]	[-202,8, 110.6]	[-326, 323.2]
March	0.061763	0.057070	0.117450	0.9985	0.9987	0.9978	[-308.6, 385.3]	[-396.9, 291]	[-275.5, 402.4]
April	0.020102	0.019101	0.031976	0.9990	0.9989	0.9984	[-276.9, 318.3]	[-275.9, 388.3]	[-188.8, 494.1]
May	0.053907	0.050544	0.064008	0.9980	0.9981	0.9975	[-442.6, 393.7]	[-464.5, 375.3]	[-348.1, 479.1]
June	0.070550	0.067576	0.086520	0.9991	0.9992	0.9989	[-472.8, 296.2]	[-557.3, 557.8]	[-404.6, 341.2]
July	0.043923	0.047421	0.057703	0.9967	0.9969	0.9962	[-484.3, 377.5]	[-472.7, 400.5]	[-542.4, 669.3]
August	0.087018	0.065867	0.101970	0.9991	0.9993	0.9989	[-596.9, 583.9]	[-628.5, 470.8]	[-491.9, 604.3]
September	0.034830	0.030865	0.051291	0.9997	0.9997	0.9996	[-349.5, 178.9]	[-297.3, 213.4]	[-230.7, 295.3]
October	0.060163	0.062805	0.087784	0.9996	0.9997	0.9996	[-238.8, 427.5]	[-232.1, 424.2]	[-297.9, 413.6]
November	0.036956	0.029505	0.065041	0.9999	0.9999	0.9999	[-114, 110]	[-101.8, 110.1]	[-147.7, 207.8]
December	0.100290	0.083948	0.109170	0.9997	0.9997	0.9997	[-51.94, 155.3]	[-123, 278.8]	[-206.5, 182.9]

The first column of this table contains the periods for which the input data have been processed and the forecasted data have been obtained, through the corresponding neural network of the 39 that we have developed. The second column of the table contains the minimum of the mean squared error, computed for each neural network, trained with a specific algorithm (LM, BR or SCG). The third column contains the minimum values of the correlation coefficients R, for each neural network, trained using a specific algorithm. The last column of the table contains the intervals in which most of the errors fall, for each neural network, as resulting from the error histograms. The obtained results, the comparison between the monthly and yearly performance accuracy indicators, prove that the solution that consists in training separate neural networks based on the input data corresponding to each of the months is net superior to the solution of training a single neural network based on the whole input data set. In addition to this, comparing the obtained monthly results, we remark that the accuracy of the obtained forecasting for all the months is excellent, the results being similar.

Comparing the accuracy of the forecasting results obtained for each of the 3 algorithms (LM, BR, SCG) we notice that in most of the situations, the neural networks that make use of the Bayesian Regularization technique have the capability of producing a better generalization than the ones using the early stopping training method, implemented in the LM and SCG algorithms. In the event of adding new input data in the future, in order to improve the training and accuracy of the networks, the LM algorithm has the advantage of being faster than the BR algorithm in training the networks, but has the drawback of consuming a higher amount of memory. If, in the future, the training speed is a concern when adding new inputs and the available system's memory becomes a limitation, then the SCG algorithm proves to be a viable solution, being faster than both the LM and BR algorithms, having reduced memory requirements but a slightly lower degree of prediction accuracy.

After analysing the performance and the prediction accuracy for all the ANN algorithms, we have used a few methods in order to improve the obtained results. First, we have reinitialized each of the networks and its training. Each time when we have applied this procedure, the network parameters and the results have changed. In some cases, we have obtained improved results by using this method.

We have also analysed another approach, adjusting the number of hidden neurons. We have increased the number of neurons in the hidden layer gradually, experimenting with different values. However, an increased number of hidden neurons increase the number of parameters that optimized by the network and thus, leads to an increased flexibility of the network. In our cases, this method did not bring a significant increase in the prediction accuracy, a number of 10 neurons in the hidden layer being sufficient for obtaining an optimum level of prediction. Another improvement method that we have applied consists in using additional training data, modifying the way in which we divided the samples up to the moment when we have obtained the best results. Through this procedure, we have obtained networks with better generalization capabilities, confirming the results obtained in the first training steps.

4. Conclusions

Our proposed solution is of great help to the investors that need to assess the potential of a certain area in producing green energy from solar power plants. The costs of measuring the input data (solar irradiation, the ambient temperature, the humidity, the wind direction and the wind speed) for a few months (or even for years) are insignificant and fully justified for an investment of such a magnitude. The forecasting results are of paramount importance for the green energy producers in submitting the estimations of green energy production requested by the Romanian National Energy Regulatory Authority (ANRE) and in developing decision support systems for the efficient management of electricity generation from renewable sources.

ACKNOWLEDGEMENTS

This paper presents a series of results obtained within the SIPAMER research project ("Sistem Inteligent pentru Predicţia, Analiza si Monitorizarea Indicatorilor de Performanţă a Proceselor Tehnologice şi de Afaceri în Domeniul Energiilor Regenerabile"), PNII – "Parteneriate în domeniile prioritare", PCCA 2013, code 0996, no. 49/2014, financed by the National Authority for Scientific Research (NASR).

REFERENCES

[1] **Nassehzadeh, Sh.T., Behboodi, E., Aliyev, F.Q. (2012),** ***Towards Renewability by Applying Solar Energy Technologies For Improved Life Cycle***; International Journal on Technical and Physical Problems of Engineering (IJTPE), *Published by International Organization of IOTPE*, Issue 11, Volume 4 Number 2 Pages 7-12, ISSN 2077-3528;

[2] **Bara, A., Lungu, I., Oprea, S. V., Carutasu, G., Botezatu, C. P., Botezatu C. (2014),** ***Design Workflow for Cloud Service Information System for Integration and Knowledge Management Based in Renewable Energy***; Journal of Information Systems & Operations Management, *Universitară Publishing;* vol.8, no.2, ISSN 1843-4711;

[3] **Bara, A., Oprea, S.V., Velicanu, A., Botha, I. (2013), *Spatial Collaborative System for Wind Power Plants Using Service Oriented Architecture***; The 2013 International Conference of Computer Science and Engineering (ICCSE'13), World Congress on Engineering, London, UK, published in Lecture Notes in Engineering and Computer Science, pp 909-914, *Newswood Limited International Association of Engineers*, ISSN: 2078-0958 (print);
[4] **Luo, L. F., Hong, Y. (2012), *Renewable Energy Systems: Advanced Conversion Technologies and Applications***; *CRC Press*, ISBN 9781439891094;
[5] **Freris, L., Infield D. (2008), *Renewable Energy in Power Systems***; *Wiley;*
[6] **Radulescu M., Banica L., Polychronidou P. (2015), *Greek Banks Profitability Developments in Romania and the Banking Strategy of the Greek Banking Groups in the Eastern Europe: A Forecasting Approach*;** *Economic Computation and Economic Cybernetics Studies and Research*, 49 (2), 189-210**;**
[7] **Andrei, C.A., Oancea B., Nedelcu M., Sinescu R.D. (2015), *Predicting Cardiovascular Diseases Prevalence Using Neural Networks*;** *Economic Computation and Economic Cybernetics Studies and Research*, 49 (3), 73-84;
[8] **Kişi, Ö., Uncuoğlu, E. (2005), *Comparison of Three Back-propagation Training Algorithms for Two Case Studies***; Indian Journal of Engineering & Materials Sciences (IJEMS), 434–442;
[9] **MacKay**, **D.J.C. (1992), *Neural Computation*;** Vol. 4, No. 3, pp. 415–447;
[10] **Foresee, D., Hagan, M. (1997), *Gauss-Newton Approximation to Bayesian Learning;*** Proceedings of the 1997 International Joint Conference on Neural Networks, *Publisher: Institute of Electrical and Electronics Engineers Inc.*
[11] **Moller, M. (1993), *A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning***; Neural Networks, Vol. 6, No. 4, pp. 525-533;
[12] **Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986**), ***Learning Internal Representations by Error Propagation,*** *in D. E. Rumelhart, & J. L. McClelland* (Eds.), Parallel distributed processing. Exploration in the microstructure of cognition (pp. 318-362), Cambridge, MA: MIT Press;
[13] **Johansson, E. M., Dowla, E U., Goodman, D. M. (1991), *Backpropagation Learning for Multi-layer Feed-forward Neural Networks Using the Conjugate Gradient Method*;** International Journal of Neural Systems, 2(4), pp. 291-302;
[14] **Battiti, R. (1990), *Optimization Methods for Back-propagation: Automatic Parameter Tuning and Faster Convergence***; IJNNC-90-WASH.DC. 1, pp.593-596;
[15] **Rojas, R. (1996), *The Back Propagation Algorithm of Neural Networks - A Systematic Introduction***; *Springer-Verlag,* Berlin, New-York, ISBN 978-3540605058;
[16] **Levenberg, K. (1944), *A method for the solution of certain problems in least squares***, *Quart. Appl. Math.,* Vol. 2, pp. 164–168;
[17] **Marquardt, D. (1963), *An Algorithm for Least-squares Estimation of Nonlinear Parameters***; SIAM J. Appl. Math., Vol. 11, pp. 431–441.