Electrical Engineering (2018) 100:1985–1995
https://doi.org/10.1007/s00202-018-0678-8

ORIGINAL PAPER

Short-term electrical load forecasting using radial basis function neural networks considering weather factors

Surender Reddy Salkuti[1]

Received: 30 September 2017 / Accepted: 6 January 2018 / Published online: 17 January 2018

Abstract
In the recent years, the demand for electricity is growing rapidly and the accuracy of load demand forecast is crucial for providing the least cost and risk management plans. In the competitive power market, utilities tend to maintain their generation reserve close to the minimum required by the system operator. Load forecasting has many applications including energy purchasing and generation, load switching, contract evaluation, and infrastructure development. This creates the requirement for an accurate day-ahead instantaneous load forecast. Therefore, in this paper a novel methodology is proposed for solving the short-term load forecasting problem using the radial basis function neural networks (RBFNNs) considering the weather factors such as temperature and humidity. The RBFNN has the advantage of handling augment new training data without requiring the retraining. The hidden layer and linear output layer of RBFNNs has the ability of learning the connection weights efficiently without trapping in the local optimum. The simulation results are performed on Pennsylvania–New Jersey–Maryland (PJM) interconnection and the obtained results are promising and accurate. The simulation studies show that the forecast results are reliable, specifically when weather factors are included in the training data.

Keywords Load forecasting · Neural networks · Weather factors · Radial basis function · Load demand

1 Introduction

In the deregulated power market, it is essential for the power generating entities to supply high quality of power to the customers. To supply high quality of power to the customers with economical and secure manner, the utility companies faces several technical and economical problems in planning, operation and control of an electrical power system. In the area of power system monitoring and control, the computer-based energy management systems (EMSs) are now used widely in the energy control centers [1]. The STLF is a commonly addressed problem in the power systems literature. The recent developments in the computer technology has enhanced possibilities for these and other approaches working in a real-time environment. And also, there is an international movement toward the wider competition in the electrical markets. Load forecasting is very important for system planning and operational decision conducted by the electrical utility companies. The STLF can help to estimate the power flows and to make decisions that can prevent the congestion in the system. Load forecasting is a very difficult task, as the load series is very complex, and exhibits various levels of seasonality: the load demand at a given hour is dependent on the load demand at previous hour as well as the load demand at the same hour on previous day, and on the demand at same hour on the day with the same denomination in the last week. And also, there are several other important variables, especially the weather-related variables such as temperature and humidity [2,3], and economics.

During the last decade, many Artificial Intelligence and Expert Systems have been built for solving the problems in different areas within the field of power systems, such as system planning, analysis, operation, monitoring and operational planning. Nowadays, the advent of neural networks (NNs) provides neural network modules, which can be executed in an online environment [4]. These new techniques supplement conventional computing techniques and methods for solving the problems of power system planning, control and operation. A two-stage hybrid algorithm considering the artificial neural networks (ANNs) and support vector regression algorithms are aimed to solve the STLF problem is

✉ Surender Reddy Salkuti
salkuti.surenderreddy@gmail.com

[1] Department of Railroad and Electrical Engineering, Woosong University, Daejeon, Republic of Korea

proposed in Ref. [5]. Reference [6] proposes a hybrid STLF approach using Bayesian NN with preprocessing stage consisting of K-means clustering technique and a time series analysis. The performance of several machine-learning techniques applied to the electrical load datasets was described in Ref. [7] and it also performs an ANN analysis on STLF problem. In Ref. [8], the knowledge-based information is provided for the STLF problem using the fuzzy logic-based approach which is well known for handling the nonlinearity conditions among various forecasting methods. A new approach for daily STLF problem belonging to the class of "similar shape" techniques is proposed in [9]. Reference [10] proposes three cascaded modules (i.e., kernel-based clustering, decision tree and support vector regression) for the STLF problem. A comparative study of STLF using Artificial Intelligence (AI) and the conventional approach is proposed in Ref. [11].

Reference [12] presents the stepwise and multiple linear regressions and improves the performs of neural network (NN). A data-driven STLF considering the historic data to predict the expected load demand for next 24-h period is proposed in [13]. A new machine-learning technique combining the convolutional NN with K-means clustering technique for solving the STLF problem with improved scalability is proposed in Ref. [14]. Reference [15] exploits the applicability and performance of the feed-forward deep neural network (FF-DNN) and recurrent deep neural network (R-DNN) models on the basis of accuracy and computational performance in the context of time-wise short-term forecast of electrical load demand. A deep belief network is proposed in Ref. [16] for solving the STLF problem. A new approach to identify the factors in which the forecasting model for energy demand, and measure their effect on the mean absolute percentage error (MAPE) criterion and error performance is proposed in [17]. In Ref. [18], a ANN is trained through a back propagation in combination with genetic algorithm (GA) model is used for solving the STLF problem. Reference [19] introduces several factors such as due temperature, temperature, humidity, wind, and applied them to the ANN to determine its impact on load forecasting of Northern Cairo.

Reference [20] compares various NNs, conditional restricted Boltzmann machines and decision tree approaches for solving the STLF problem. The variation and periodicity of power system load demand data have been analyzed in Ref. [21] with removed bad data when the correlation process was conducted. A new STLF model was proposed in Ref. [22] by the combination of singular spectrum analysis, support vector machine and Cuckoo search algorithms. A new hybrid learning approach which combines the extreme learning machine with a new switching delayed particle swarm optimization (PSO) algorithm for solving the STLF problem is proposed in [23]. Reference [24] proposes the choosing mechanism of an energy company which gathers a library of electric load models and at every day chooses the best one for daily prediction. Recently, the extreme learning machine (ELM) algorithm has been applied successfully for STLF problem [25]. But, the random initialization of weight parameters in ELM may introduce inferior values, leading to unreliable result and also the ELM suffers from the overtraining problem [26].

From the above literature review, it can be observed that the conventional load forecasting techniques (i.e., time series approach, multivariate regression approach and state estimation approach) have the advantage that load pattern can be forecasted using a simple prediction model. However, due to the nonlinear relationship between load pattern and the factors influencing it, it is difficult to find this nonlinear relationship using conventional methods [27]. The accuracy of load forecasting can be increased with the help of fuzzy logic and support vector machines techniques. However, despite their accuracy, these approaches have a number of inconveniences such as difficulty in parameter selection, nonobvious selection of variables and overfitting. ANNs can manage the nonlinear dependencies that exist among the load demands and the factors influencing it, forthrightly. Neural networks have the capability to sort out nonlinear curves suitably. ANN maps the input and output relations by the help of approximate linear or nonlinear mathematical functions. The back propagation (BP) learning does not ensure a global optimum solution to the neural network training. Particularly, the BP learning technique is stuck in a local optimum solution and does not optimize the NN weight bias values during the training process. This may affect the NN performance and lead to a higher forecast error. Therefore, there is a requirement to develop an efficient STLF method in the context of deregulated power system.

To overcome the abovementioned shortcomings, this paper develops a novel radial basis function neural network (RBFNN)-based load forecasting model. In RBFNNs algorithm, the load forecasting is obtained based on the logical relationships between weather load and the prevailing load patterns and has the advantage of dealing not only with the linear part of the load curve but also with special days, weekends and holidays. In this paper, a new methodology is proposed for solving the STLF problem using the RBFNNs considering the weather factors such as temperature and humidity. The RBFNN has the advantage of handling the augment new training data without requiring the retraining. The hidden layer and linear output layer of RBFNNs have the ability of learning the connection weights efficiently without trapping in the local optimum. The training of RBFNN requires less computation time, because only the second-layer weights have to be calculated using the error signal. The training of RBFNNs can be made even faster by applying the momentum and adaptive learning rates. Here, the Euclidean distance-based clustering technique has been employed to

select the number of hidden (RBF) units and unit centers. The normalized input and output data are used for training of the RBFNN. The optimal learning is achieved at the global minimum of testing error. This paper applies different weather variables to the STLF problem which reduces the error. Then, the results of actual load with the forecasted load are compared. The simulation results show that this technique leads to a reduction in forecasting error and fast convergence. Accurate models for the STLF problem are essential for planning and operation of an electrical utility company. Although the present paper focuses on STLF, the proposed method can be easily extended to electricity price and renewable power forecasting problems.

The remaining parts of the paper are organized as follows. Section 2 presents the factors influencing the accurate load forecasting. The description and implementation of radial basis function neural networks (RBFNN) for STLF problem is present in Sect. 3. Section 4 describes the simulation results and discussion. Section 5 presents the contributions with concluding remarks.

2 Factors influencing the accurate load forecasting

The important factors which effect the accuracy of load forecasting are weather factors, time factors, and the customer classes. In this paper, the STLF is performed considering the weather factors. Performing the STLF considering the time factors and customer classes is a scope for future research. Electric load has an obvious correlation to weather factors. Many utility companies have large components of weather sensitive load demands, like space heating loads, agricultural irrigation and air conditioning. For any given day, the deviation of weather variables from the normal value may cause significant load changes and requires major modifications to the unit commitment (UC) schedule. The time factors (i.e., the day of the week the hour of the day and holiday) affect the STLF. There are important differences in load between weekdays and weekends. The load on different weekdays also can behave differently. For example, Mondays and Fridays being adjacent to weekends, may have structurally different loads than Tuesday through Thursday. This is particularly true during the summer time. Holidays are more difficult to forecast than nonholidays because of their relative infrequent occurrence. The hour of the day, the load will have different value at different timings. Therefore, the day of the week whether weekday or weekend also the hour of the day will affect the load.

The important weather variables in terms of their effects on load demand are temperature, humidity, wind speed and cloud cover. Temperature has a close relationship with the load demand. Generally, the electrical load has a tendency to increase during the winter seasons because of the usage of heaters, and also during the summer seasons because of the usage of air conditioners. The temperature in a area which is located nearer to the equator will be higher. The past temperatures will also affect the load demand profile. The humidity also has a significant impact on the comfort or discomfort felt, since it affects the amount of heat that the human body rejects through the evaporation [3]. High levels of relative humidity impede the rejection of heat through the evaporation, whereas, the low levels accelerate it. Many people prefer relative humidity levels in the range between (40–60)%. During summer months (temperatures higher than 40 °C), high levels of relative humidity combined with high temperatures create the need for additional cooling loads and hence cause an increase in the load demand. The increase in the required cooling load is due to the relative humidity at a given temperature. It can be associated with an appropriate increase in this temperature assuming a level of relative humidity that cooling loads are insensitive to, for example 50%. On the other hand, low levels of relative humidity can reduce the required cooling load. On the similar lines, different types of customers such as, residential, commercial and industrial loads can affect the STLF.

The economic environment in which the utility operates has a clear effect on the electric demand consumption patterns. The economic trends have significant impact on the system load growth/decline trend. Typically, these economic factors operate with considerably longer time constants than one week and hence need not to be considered for self. However, these factors should be taken into consideration for long- and medium-term forecasting models. Also, the random disturbances include loads such as steel mills, synchrotrons, wind tunnels whose operation can cause large variations in electricity usage wide spread strikes, bands, special TV programmers whose effect on the load is not known a priori could cause sudden and unpredictable unload.

Other factors which have an impact on load demand are cloud cover and wind speed. There will be appropriate change in the electric load with the change in wind speed and cloud cover. Since, the weather variables are closely related to the electrical load demand, among which the temperature and humidity will affect the load. Therefore, in this paper, to improve the accuracy of load forecast the temperature and humidity factors are considered along with the load data as the inputs to the RBFNN model.

3 Description and implementation of radial basis function neural networks (RBFNNs) for STLF

As mentioned earlier, in this paper, we are performing the STLF using the RBFNN as it has the ability to augment new

training data without any requirement for the retraining. The linear output layer and hidden layer structures of RBFNNs provide the possibility of learning the connection weights efficiently without falling in the local minima. The training of RBFNN requires less computation time, because only the second-layer weights have to be calculated using the error signal. The training of RBFNNs can be made even faster by applying the momentum and adaptive learning rates [28,29]. Therefore, the problems of electrical load forecasting, price forecasting and load flows can be solved easily using the RBFNNs. The description and implementation of RBFNN for solving the STLF problem is presented next:

3.1 Operation of network

This network has two operating modes, i.e., training and testing. During the training mode, the adjustable parameters of network are set, to minimize the average error between the actual and the desired output over the vectors in a training set. In the testing mode, input vectors are applied and the output vectors are produced by the network [30]. The RBFNNs are benefited by clustering the training vectors when there is a large amount of training data. If the data form a cluster, an entire cluster of training vectors in the training set may connect to reduce the number of hidden nodes. This may substantially decrease the computation time in the reference mode.

The K-means clustering algorithm requires a set of the initial conditions for the centers. These can be assigned at random; however, this is seldom optimal as centers may often fall in a region where there are no input vectors. It is better to locate the centers only where there is data nearby. Some clustering algorithms may erroneously define a cluster center where no cluster exists. This can make it impossible to train the output weights associated with the associated hidden layer neuron. If a cluster center is defined where there are no input vectors are nearby, the output of that hidden layer neuron will be essentially zero for all input vectors.

3.2 Computation of RBF unit centers (C_j)

The unit centers (C_j) are determined using the following algorithm:

Step 1 Initialize the center of each cluster to a randomly considered training pattern (i.e., C_{ji}). Where $i = 1, 2, 3, \ldots$, inputs (ipn); $j = 1, 2, 3, \ldots$, hidden nodes (hdn). Here, *ipn* is number of input nodes, *hdn* is number of output nodes. Let the training samples are represented by,

$$[x_{pi}] = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1i} \\ \vdots & & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pi} \end{bmatrix}_{(p\times i)} \tag{1}$$

where p is the number of training patterns and i is the number of input nodes. The general guidelines in RBFNN approach are: if there are p-patterns, the hidden nodes (60–70)% of total number of patterns. Then, assign randomly some of the training inputs ($X_{1i}, X_{2i}, \ldots, X_{ji}$) to each hidden node centers.

$$[C_{ji}] = [X_{ji}] \tag{2}$$

where j (number of hidden nodes) $<$ p (number of patterns).

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1i} \\ \vdots & & \ddots & \vdots \\ c_{j1} & c_{j2} & \cdots & c_{ji} \end{bmatrix}_{(j\times i)} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1i} \\ \vdots & & \ddots & \vdots \\ x_{j1} & x_{j2} & \cdots & x_{ji} \end{bmatrix}_{(j\times i)} \tag{3}$$

The general practice is that to assign first j-patterns of training patterns to $[C_{ji}]$ vector as shown above.

Step 2 Assign each training pattern to the nearest cluster. This can be achieved by calculating the Euclidean distance between the cluster centers and the training patterns [31,32]. The Euclidean distance between X and weight vectors (C_{ij}) of each neuron in Euclidean space is calculated using,

$$D_j = \sqrt{\sum_{i=1}^{ipn} \left(x_{pi} - C_{ji}\right)^2} \quad j = 1, 2, \ldots, hdn \tag{4}$$

where *hdn* is the number of nodes in the hidden layer. x_{pi} is the ith variable of pth input pattern, and C_{ji} is the center from input i to neuron j.

Step 3 Assign the pattern to the node which has the minimum Euclidean distance (D_j).

Step 4 When all the training patterns are assigned, determine the average position for each cluster center, and they will become the new cluster centers. Let's consider that for any hidden node j, the training patterns a, b, and c are assigned, then the new cluster center becomes the average of center C_{ji}, X_{ai}, X_{bi}, and X_{ci}. The initial cluster center $[C_{ji}] = [C_{j1}, C_{j2}, C_{j3,\ldots\ldots,}C_{ji}]$, then the new cluster center becomes,

$$[C_{ji}]^* = \left[\frac{\left(C_{j1} + X_{a1} + X_{b1} + X_{c1}\right)}{4}, \times \frac{\left(C_{j2} + X_{a2} + X_{b2} + X_{c2}\right)}{4}, \ldots, \frac{\left(C_{ji} + X_{ai} + X_{bi} + X_{ci}\right)}{4}\right] \tag{5}$$

Step 5 Repeat Steps 2–4, until the cluster centers do not change during the subsequent iterations.

3.3 Computation of unit widths (σ_j) of RBFNN

The widths of each RBF unit j can be calculated using,

$$\sigma_j = \sqrt{\frac{1}{hdn}\sum_{k=1}^{hdn}\sum_{i=1}^{ipn}\left(C_{ji} - C_{ki}\right)^2} \tag{6}$$

where C_{ji} and C_{ki} are the ith entries of centers of jth and kth hidden units.

3.4 Computation of activation (O_j)

After finding the centers and widths of each hidden unit, the activation level (output) of each hidden unit j is calculated using,

$$a_j\left(X_p\right) = e^{\sqrt{\frac{-\sum_{i=1}^{ipn}\left(x_{pi}-C_{ji}\right)^2}{2\sigma_j^2}}} \tag{7}$$

The connection between the output and hidden units is the weighted sums. The output value (O_{kp}) of the kth output node for pth incoming pattern is expressed using,

$$O_{kp} = \sum_{j=1}^{hdn} w_{kj} a_j\left(X_p\right) + w_{ko} \tag{8}$$

where w_{kj} is the weight between jth RBF unit and kth output node, w_{ko} is the biasing term at kth output node. In this paper, w_{ko} is taken as zero.

3.5 Supervised learning

After choosing the centers and widths, the output layer weight matrix can be optimized by the supervised learning. A training set must be available, composed of pairs of vectors. Each pair of vectors consists of an input vector designated x and a target t. The target vector indicates the output desired from the network when it's associated input vector. The training process is presented in the following steps [27]:

Step 1 Apply an input vector x from the training set.
Step 2 Calculate the outputs of hidden layer neurons, collectively referred as vector [a].
Step 3 Compute the network output vector [y]. Compare it with target vector t. Adjust each weight in [w] in a direction to reduce the difference. For this, delta rule is used and it is expressed as,

$$w_{kj}(n+1) = w_{kj}(n) + \eta\left(t_k - y_k\right) a_j. \tag{9}$$

where w_{kj} is the weight between the jth hidden layer neuron and the kth output layer neuron, y_k is the kth output of neuron in the output layer, t_k is the targeted or desired output for kth neuron in the output layer, and η is the learning rate coefficient.
Step 4 Repeat the Steps 1 and 3 for each vector in the training set.
Step 5 Repeat the Steps 1 and 4 until the error is acceptably small.

Because of the limited receptive field of each hidden layer neuron, most of the hidden layer outputs will be nearly zero for a given input vector. As it is seen from the weight updates equation, the weight change correspondingly small for all weights connected to such a neuron. Thus, these neurons may be ignored in the weight update process, thereby greatly reducing the training time. Also, there is only the linear output layer (no activation function is used at output layer), the network is unarmed to converge to a global minimum. For each input vector (x_i) in training set, the outputs from hidden layer made a row in matrix (H), target vectors (T_i) are placed in the corresponding row of target matrix (T). The training consists of solving the following matrix equation, and it is expressed as [33,34],

$$T = HW \quad \text{or} \quad W = H^{-1}T \tag{10}$$

where W is the weight array. H^{-1} is the inverse of matrix H. In general, the matrix (H) is not square, hence it is not invertible, only a pseudoinverse can be determined, if it exists. Finding the pseudoinverse involves inverting a matrix, which is frequently ill conditioned and cannot be accurately inverted by simple methods. It is possible to approximate this by angular value decomposition, but this may not be justified due to the resulting computational burden and limited accuracy of the results. For these reasons, an iterative technique is used.

3.6 Algorithm for RBFNN

The solution algorithm for STLF problem using the RBFNN [33,34] is presented next:

Step 1 Normalize the inputs and outputs with respect to their maximum values. Assume that for each training pair, there are 'N' inputs given by $[I_I]_{N\times 1}$ and 'NO' outputs $[O_O]_{NO\times 1}$ in a normalized form.
Step 2 Assume that the number of neurons in hidden layer (i.e., NH) is lies between $N < NH < NS$.
Step 3 Let [W] be the weights of synapses connecting the hidden and output neurons. Initialize the weights to a small random values ranging from − 1 to 1.

$$[W]^o = [\text{Random weights}];\ [\Delta W]^o = [0] \tag{11}$$

Step 4 By using the K-means clustering technique, assign some training samples to the centers. Determine the centers (C_j) for each hidden node.

Step 5 Determine the unit width of each hidden node by using,

$$\sigma_j = \sqrt{\frac{1}{NH}\sum_{k=1}^{NH}\sum_{i=1}^{N}\left(C_{ji} - C_{ki}\right)^2}. \tag{12}$$

Step 6 Set iteration count $k = 1$.

Step 7 The activation level (output) of each hidden unit is calculated using,

$$a_j\left(X_p\right) = e^{\sqrt{\frac{-\sum_{i=1}^{N}\left(x_{pi} - C_{ji}\right)^2}{2\sigma_j^2}}} \tag{13}$$

Step 8 Compute the output of output layer by multiplying the corresponding weights of synapses to the hidden layer activation levels. The network output is represented by,

$$\left[O_{kp}\right]_{(NO\times 1)} = [W]^T_{(NO\times NH)} \times \left[a_j\right]_{(NH\times 1)} \tag{14}$$

Step 9 Determine the error and difference between network output and desired output as for the kth training set using,

$$E^p = \frac{\sqrt{\sum_{j=1}^{n}\left(T_j - O_{kj}\right)^2}}{NO} \tag{15}$$

Step 10 Find [d] using,

$$[d]_{(NO\times 1)} = \begin{bmatrix} \cdots\cdots \\ \cdots\cdots \\ \left(T_j - O_{kj}\right) O_{kj}\left(1 - O_{kj}\right) \\ \cdots\cdots \\ \cdots\cdots \end{bmatrix}_{(NO\times 1)} \tag{16}$$

Step 11 Find $[Y]$ matrix using,

$$[Y]_{(NH\times NO)} = \left[a_j\right]_{(NH\times 1)} \times [d]_{(1\times NO)} \tag{17}$$

Step 12 Determine the change in weight and update the weights using,

$$[\Delta W]^{t+1}_{(NH\times NO)} = \alpha_z\,[\Delta W]^{t}_{(NH\times NO)} + \lambda_z\,[Y]_{(NH\times NO)} \tag{18}$$

$$[W]^{t+1}_{(NH\times NO)} = [W]^{t}_{(NH\times NO)} + [\Delta W]^{t+1}_{(NH\times NO)} \tag{19}$$

Step 13 Determine the error rate using,

$$\text{Error rate} = \frac{\sum E^p}{NS} \tag{20}$$

where E^p is the error and NS represents the number of samples in training data set.

Step 14 Repeat the Steps 7–13 until the convergence is met, i.e., the error rate is less than the specified tolerance value. The NN output needs descaling/denormalization to generate the desired forecasted load demands.

The simulation results and discussion are presented in the next section.

4 Results and discussion

In this section, the simulation results are performed on Pennsylvania–New Jersey–Maryland (PJM) interconnection system data to show the effectiveness of the RBFNN approach for solving the STLF problem. The data used for this load forecasting is obtained from the PJM website [35]. The typical structure of RBFNN for solving the STLF problem is a three layered NN with nonlinear radial basis function as the transfer function. The RBFNN model used in this paper has 38 neurons in the input layer and 1 neuron in the output layer. The number of hidden neurons selected as 600 with Gaussian density function. The Euclidean distance-based clustering algorithm has been used to select the number of hidden units and unit centers. During the training of RBFNN, care has been taken to avoid the over training [36]. All the simulation studies are coded in MATLAB R2016a on a Personal Computer with 2.9 GHz dual-core Intel Core i7 processor with 12 GB of RAM. In this paper, three case studies are performed and they are presented next:

- Case 1: STLF without considering weather factors.
- Case 2: STLF considering temperature.
- Case 3: STLF considering temperature and humidity.

4.1 Case 1: Short-term load forecasting (STLF) without considering weather factors

In this case study, the RBFNN is selected with the number of input, hidden and output nodes as 38, 600 and 1, respectively. Here, the number of training samples selected for the STLF problem is 2304 (i.e., 48 × 48) and the number of testing samples selected is 336 (i.e., 48 × 7). The learning rate, momentum rate and the maximum number of iterations selected are 0.02, 0.2 and 500, respectively.

4.1.1 Input variables selection

Different sets of lagged load demands have been selected as the input features for the STLF problem. Bearing in mind, the daily and weekly periodicity and trend of the load signal, the following set of 38 load demands are considered as the input variables and they are applied to the RBFNN.

$$InputVariables = \left\{ \begin{array}{c} L_{h\text{-}1}, L_{h\text{-}2}, L_{h\text{-}3}, L_{h\text{-}4}, L_{h\text{-}5}, L_{h\text{-}6}, L_{h\text{-}24}, L_{h\text{-}25}, L_{h\text{-}26}, L_{h\text{-}27}, \\ L_{h\text{-}48}, L_{h\text{-}49}, L_{h\text{-}50}, L_{h\text{-}51}, L_{h\text{-}72}, L_{h\text{-}73}, L_{h\text{-}74}, L_{h\text{-}75}, L_{h\text{-}96}, L_{h\text{-}97}, \\ L_{h\text{-}98}, L_{h\text{-}99}, L_{h\text{-}120}, L_{h\text{-}121}, L_{h\text{-}122}, L_{h\text{-}123}, L_{h\text{-}144}, L_{h\text{-}145}, L_{h\text{-}146}, L_{h\text{-}147}, \\ L_{h\text{-}168}, L_{h\text{-}169}, L_{h\text{-}170}, L_{h\text{-}171}, L_{h\text{-}192}, L_{h\text{-}193}, L_{h\text{-}194}, L_{h\text{-}195}. \end{array} \right\} \quad (21)$$

These 38 inputs are used at input layer, is considered as the input features for the present work. The input features of lagged loads are selected based on the experience of many researchers recommended in several publications [37–39]. L_h (i.e., load at hour h) is the single output feature of the training samples. When load at hour h is to be forecasted, then it is used as $L_{h\text{-}1}$ for the load prediction of the next hour, and this cycle is repeated until the load of whole forecast horizon (here, the next 24 h) is predicted.

4.1.2 Training period selection

In this paper, last 48-day load demand data are considered for training the RBFNN. Therefore, the total number of training samples will be 2304 (i.e., 48 (days) × 48 (intervals)) samples. The detailed explanation is presented next:

As mentioned earlier, in this paper, last 48 days have been presented as the training period of RBFNN for the loads prediction. Thus, the RBFNN is trained with training samples shown in Eq. (21). Then, it forecasts the loads of the next 24 h (one day ahead). For forecasting the today's 24-h load demands at half an hour interval, previous 48-day (each day 48 intervals) loads are used for training the NN. The training phase of RBFNN is executed for each day separately. It is to be noted that the selected input variables and training period (including 2304 training samples) have been considered for all cases studied in this paper. Table 1 presents the description of training samples used for the proposed STLF problem.

Where D is the index used for representing the day. To train the RBFNN with an architecture of 38 input neurons, 600 hidden neurons and one neuron at the output layer, the following approach is used.

For each load L_h of interval h, the data element value at that interval is considered as the single target output, and the data elements from previous 8 days (as shown in Table 2) are used as the input feature.

In Table 2, L_h is single target or load output at hth interval of selected day-D. $L_{h\text{-}1}$ to $L_{h\text{-}195}$ are the 38 inputs to the input layer, picked up as shown in Table 2, from the load data elements. Two-month (January and February, 2017) data are collected from the Pennsylvania–New Jersey–Maryland (PJM) interconnection [36]. Hence, total of 59 days data are made available. For this, at half an hour interval step, total 2832 (i.e., 59 (days) × 48 (intervals)) data elements are possible. As we use previous 8-day data for each load element value L_h for training the network, January 1, 2017 to January 8, 2017 data are set aside as buffer and actual training is initiated from the first half an hour of January 9, 2017. From January 9, 2007 to February 25, 2007, data (i.e., 48-day data) are used for training the selected network for predicting the load for 24 h at half an hourly interval of the remaining 3 days (26th–28th) of February 2017. Total 144 data elements are used as test data elements.

4.1.3 Normalization/scaling

Normalization/scaling is required so that all the inputs are at a comparable range. The normalized input and output data are used for training of RBFNN. The common normalization approach includes statistical normalization and Min-Max normalization. In this paper, min–max normalization is used. The input and output load demands are normalized with respect to their lower and upper values in each training pattern using,

$$x_{\text{norm}} = \frac{(x_i - x_{\min})}{(x_{\max} - x_{\min})} \times 0.8 + 0.1 \quad (22)$$

The radial basis function NN (RBFNN) algorithm is used for learning (training) the neural network. Using the trained neural network, the forecasting output is simulated using the test input patterns.

4.1.4 Testing period selection

The load prediction is carried out for 1 week (7 days) has 336 (i.e., 7 (days) × 48 (intervals)) samples. Using the trained NN, the forecasting output is simulated using the test input patterns.

4.1.5 Descaling/denormalization

Post-processing of data involves denormalizing or reversing the normalization. The NN output needs descaling to gener-

Table 1 Description of training samples used for STLF

(D-48)	(D-47)		(D-3)	(D-2)	(D-1)	(D)
(48, . . . , 3, 2, 1)	(48, . . . , 3, 2, 1)	…	(48, . . . , 3, 2, 1)	(48, . . . , 3, 2, 1)	(48, . . . , 3, 2, 1)	
Training period of previous 48 days and each day 48 intervals (i.e., 48 = 2304 training data elements)						Forecasting day (24 h)

Table 2 Selected load data elements from previous 8 days

(D-8)	(D-7)	…	(D-2)	(D-1)	(D)	
$L_{(h\text{-}192)}$	$L_{(h\text{-}168)}$	…	$L_{(h\text{-}48)}$	$L_{(h\text{-}24)}$	$L_{(h\text{-}1)}$	L_h (target output at interval h)
$L_{(h\text{-}193)}$	$L_{(h\text{-}169)}$	…	$L_{(h\text{-}49)}$	$L_{(h\text{-}25)}$	$L_{(h\text{-}2)}$	
$L_{(h\text{-}194)}$	$L_{(h\text{-}170)}$	…	$L_{(h\text{-}50)}$	$L_{(h\text{-}26)}$	$L_{(h\text{-}3)}$	
$L_{(h\text{-}195)}$	$L_{(h\text{-}171)}$		$L_{(h\text{-}51)}$	$L_{(h\text{-}27)}$	$L_{(h\text{-}4)}$	
					$L_{(h\text{-}5)}$	
					$L_{(h\text{-}6)}$	

ate the desired forecasted load demands using,

$$x_{\text{denorm}} = \frac{(x_{\text{norm}} - 0.1)}{0.8} \times (x_{\max} - x_{\min}) + x_{\min} \tag{23}$$

4.1.6 Error analysis

The daily and weekly mean errors are calculated for the forecasted output/load demand. The daily mean error (DME) is calculated using,

$$\text{DME} = \frac{1}{48} \sum_{i=1}^{48} \frac{|L_a(i) - L_f(i)|}{L_a(i)} \times 100 \tag{24}$$

The weekly mean error (WME) is calculated using,

$$\text{WME} = \frac{1}{336} \sum_{i=1}^{336} \frac{|L_a(i) - L_f(i)|}{L_a(i)} \times 100 \tag{25}$$

As mentioned earlier, in this paper, the STLF is performed using the RBFNN. Here, the STLF is performed without considering weather factors (i.e., Case 1). The obtained actual and forecasted load demands are Fig. 1. From this figure, it can be observed that the forecasted load is closely follows the actual load. The obtained daily mean error (DME) and the weekly mean error (WME) for this case are 3.54 and 13.83%, respectively.

The total time taken for Case 1 is 290.62 s, which includes the network scanning time of 22.84 s, training time of 264.73 s and the testing time of 3.05 s. Figure 2 depicts the plot between the error rate and number of iterations for Case 1.

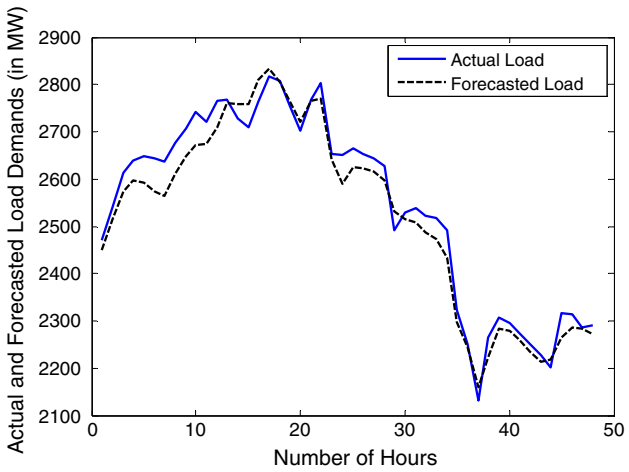

Fig. 1 Actual and forecasted load demands using RBFNN for Case 1

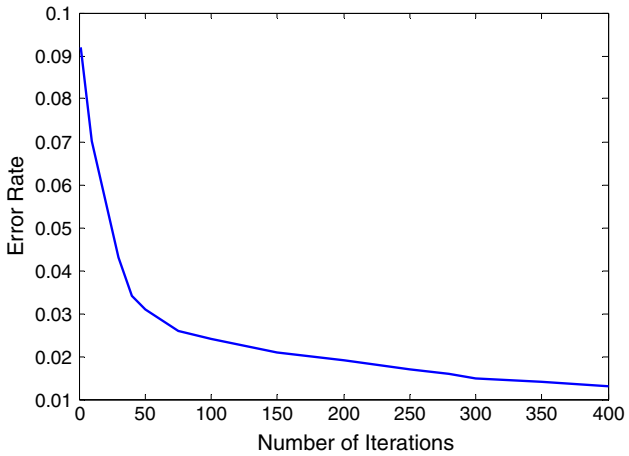

Fig. 2 Plot between the error rate and the number of iterations for Case 1 using RBFNN

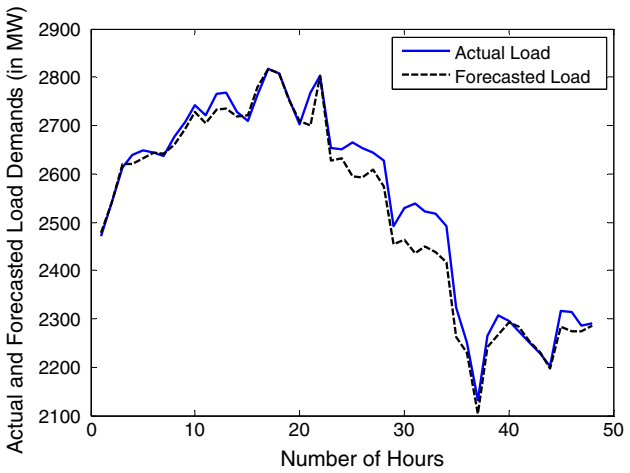

Fig. 3 Actual and forecasted load demands for Case 2 using the RBFNN

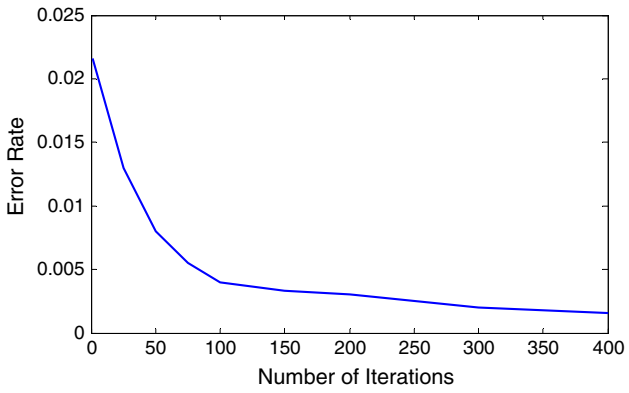

Fig. 4 Plot between the error rate and the number of iterations for Case 2 using HFNN

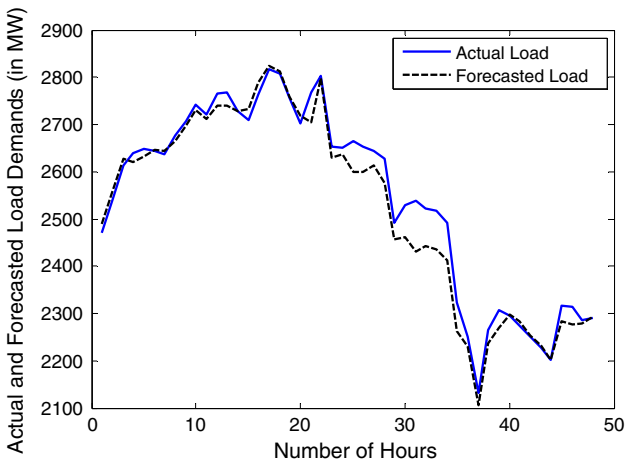

Fig. 5 Actual and forecasted load demands for Case 3 using the RBFNN

4.2 Case 2: STLF considering temperature

Many utility companies have large components of whether sensitive load demands, like air conditioning, space heating and agricultural irrigation. In many systems, temperature is an important weather variable in terms of its effects on load demand. The daily load is strongly influenced by recent temperature and average daily temperature. The average daily temperature is treated as part of the input set to reflect the daily load variations. Mainly residential load has a tendency to follow the temperature trend. In order to reflect this characteristic, the average daily temperatures, the maximum temperature and minimum temperature of the target day are considered in this paper. When temperature is considered then the three temperature inputs (i.e., maximum ($T_{\max}$), minimum ($T_{\min}$) and average (T_{avg}) temperatures) will be taken in addition to the existing load inputs.

In this case, the input neurons are of two different units, i.e., load demands are taken in MW and temperatures are taken in °C (in order of 20–40). To avoid the effect of temperature not to be overwritten by the load demands, the temperatures and load demands are normalized separately, and apply them to the input neurons. In this Case, 41 input nodes are considered which includes 38 input load variables same as in Case 1, and three temperature inputs (i.e., $T_{\max}$, $T_{\min}$ and T_{avg}).

Figure 3 depicts the actual and forecasted load demands for Case 2 using the RBFNN. The obtained DME and WME for this case are 2.63 and 13.59%, respectively. The total time taken for this Case 2 is 293.15 s, which includes the network scanning time of 23.96 s, training time of 265.18 s and the testing time of 4.01 s. The convergence characteristics, i.e., error rate vs number of iterations using HFNN for Case 2 is depicted in Fig. 4.

4.3 Case 3: STLF considering temperature and humidity

In this case, 38 input neurons from past load demand data are considered similar to Case 1, 3 input neurons are considered to accommodate the temperature effect, and 1 input neuron is considered to accommodate the humidity effect. Therefore, in this case, a total of 42 neurons will be taken as input neurons. The number of training samples, testing samples and all other parameters are same as in Cases 1 and 2. In this case, the humidity, temperatures and loads are normalized separately and then applied them to the input neurons.

Figure 5 depicts the obtained actual and forecasted load demands for Case 3 using the RBFNN. The obtained DME and WME for this Case are 2.62 and 16.51%, respectively. The total time taken for this Case 3 is 307.33 s, which includes the network scanning time of 24.01 s, training time of 278.93 s and the testing time of 4.39 s. The convergence characteristics, i.e., error rate vs number of iterations using RBFNN for Case 3 is depicted in Fig. 6.

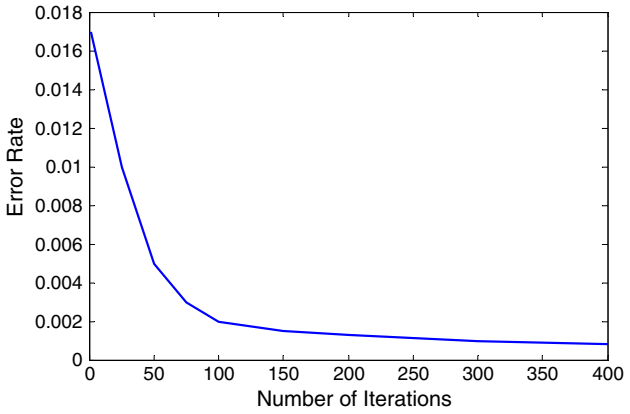

Fig. 6 Plot between the error rate and number of iterations for Case 3 using HFNN

From the above results, it can be observed that the daily mean error has decreased by considering the weather factors to the STLF. All the forecasting results obtained are promising and accurate.

5 Conclusions

In this paper, the short-term load forecasting (STLF) problem is solved using the radial basis function neural network (RBFNN). The RBFNN has the advantage of handling the augment new training data without requiring the retraining. The hidden layer and linear output layer of RBFNNs have the ability of learning the connection weights efficiently without trapping in the local optimum. The simulation results are performed on Pennsylvania–New Jersey–Maryland (PJM) interconnection, and the obtained results are promising and accurate. The detailed network parameters are presented along with the training and testing patterns. For each case under study, the actual and forecasted load demand values are presented which shows that the forecasted values are following the actual load values. From the simulation results, it can be observed that by considering both temperature and humidity along with the loads as inputs, there is a decrease in the daily mean error compared to the cases without considering the weather factors. The extension of proposed algorithm to different situations (e.g., seasonal, high volatility, peak load, low load), various customer classes and time factors is the scope for future research work.

References

1. Rao GM, Swamy IN, Kumar BS (2010) Deregulated power system load forecasting using artificial intelligence. In: IEEE international conference on computational intelligence and computing research, Coimbatore, pp 1–5
2. Wu Y (2012) Software engineering and knowledge engineering: theory and practice, advances in intelligent and soft computing. Springer, Berlin
3. Contaxi E, Delkis C, Kavatza S, Vournas C (2006) The effect of humidity in a weather-sensitive peak load forecasting model. In: IEEE PES power systems conference and exposition, Atlanta, GA, pp 1528–1534
4. Reddy SS, Momoh JA (2014) Short term electrical load forecasting using back propagation neural networks. North American power symposium, Pullman, WA, pp 1–6
5. Wu X, He J, Yip T, Lu J, Lu N (2016) A two-stage random forest method for short-term load forecasting. IEEE power and energy society general meeting, Boston, MA, pp 1–5
6. Ghofrani M, Carson D, Ghayekhloo M (2016) Hybrid clustering-time series-Bayesian neural network short-term load forecasting method. North American power symposium, Denver, CO, pp 1–5
7. Bećirović E, Ćosović M (2016) Machine learning techniques for short-term load forecasting. In: 4th international symposium on environmental friendly energies and applications, Belgrade, pp 1–4
8. Matthew S, Satyanarayana S (2016) An overview of short term load forecasting in electrical power system using fuzzy controller. In: 5th international conference on reliability, infocom technologies and optimization (trends and future directions), Noida, pp 296–300
9. Tucci M, Crisostomi E, Giunta G, Raugi M (2016) A multi-objective method for short-term load forecasting in European countries. IEEE Trans Power Syst 31(5):3537–3547
10. Wang Y, Yang J (2015) Kernel-based clustering for short-term load forecasting. In: 10th international conference on advances in power system control, operation & management, Hong Kong, pp 1–6
11. Luthuli QW, Folly KA (2016) Short term load forecasting using artificial intelligence. IEEE PES PowerAfrica, Livingstone, pp 129–133
12. Yang HP, Yan FF, Wang H, Zhang L (2016) Short-term load forecasting based on data mining, In: IEEE 20th international conference on computer supported cooperative work in design, Nanchang, pp 170–173
13. Høverstad BA, Tidemann A, Langseth H, Öztürk P (2015) Short-term load forecasting with seasonal decomposition using evolution for parameter tuning. IEEE Trans Smart Grid 6(4):1904–1913
14. Dong X, Qian L, Huang L (2017) Short-term load forecasting in smart grid: A combined CNN and K-means clustering approach. In: IEEE international conference on big data and smart computing, Jeju Island, South Korea, pp 119–125
15. Din GMU, Marnerides AK (2017) Short term power load forecasting using deep neural networks. In: International conference on computing, networking and communications, Silicon Valley, CA, USA, pp 594–598
16. Zhang X, Wang R, Zhang T, Zha Y (2016) Short-term load forecasting based on a improved deep belief network. In: International conference on smart grid and clean energy technologies, Chengdu, China, pp 339–342
17. Mares JJ, Mercado KD, Quintero CG (2017) A methodology for short-term load forecasting. IEEE Latin Am Trans 15(3):400–407
18. Ray P, Mishra DP, Lenka RK (2016) Short term load forecasting by artificial neural network. In: International conference on next generation intelligent systems, Kottayam, pp 1–6
19. Eljazzar MM, Hemayed EE (2016) Feature selection and optimization of artificial neural network for short term load forecasting. In: Eighteenth international middle east power systems conference, Cairo, pp 827–831
20. Warrior KP, Shrenik M, Soni N (2016) Short-term electrical load forecasting using predictive machine learning models. In: IEEE annual India conference, Bangalore, pp 1–6
21. Zhuang L, Liu H, Zhu J, Wang S, Song Y (2016) Comparison of forecasting methods for power system short-term load forecasting

based on neural networks. In: IEEE international conference on information and automation, Ningbo, pp 114–119
22. Zhang X, Wang J, Zhang K (2017) Short-term electric load forecasting based on singular spectrum analysis and support vector machine optimized by Cuckoo search algorithm. Electr Power Syst Res 146:270–285
23. Zeng N, Zhang H, Liu W, Liang J, Alsaadi FE (2017) A switching delayed PSO optimized extreme learning machine for short-term load forecasting. Neurocomputing 240:175–182
24. Duan Q, Liu J, Zhao D (2017) Short term electric load forecasting using an automated system of model choice. Int J Electr Power Energy Syst 91:92–100
25. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B (Cybern) 42(2):513–529
26. Li S, Wang P, Goel L (2015) Short-term load forecasting by wavelet transform and evolutionary extreme learning machine. Electr Power Syst Res 122:96–103
27. Khwaja AS, Naeem M, Anpalagan A, Venetsanopoulos A, Venkatesh B (2015) Improved short-term load forecasting using bagged neural networks. Electr Power Syst Res 125:109–115
28. Lalitha SVNL, Sydulu M (2011) Hybrid Neural Network models for determination of locational marginal price. In: 11th international conference on hybrid intelligent systems, Melacca, pp 424–429
29. Weron R (2014) Electricity price forecasting: a review of the state-of-the-art with a look into the future. Int J Forecast 30(4):1030–1081
30. Guo X, Xiao Y, Shi J (2008) An empirical research of forecasting model based on the generalized regression neural network. In: IEEE international conference on automation and logistics, Qingdao, pp 2950–2955
31. Khwaja AS, Zhang X, Anpalagan A, Venkatesh B (2017) Boosted neural networks for improved short-term electric load forecasting. Electr Power Syst Res 143:431–437
32. Dudek G (2016) Neural networks for pattern-based short-term load forecasting: a comparative study. Neurocomputing 205:64–74
33. Abdoos A, Hemmati M, Abdoos AA (2015) Short term load forecasting using a hybrid intelligent method. Knowl Based Syst 76:139–147
34. Hooshmand RA, Amooshahi H, Parastegari M (2013) A hybrid intelligent algorithm based short-term load forecasting approach. Int J Electr Power Energy Syst 45(1):313–324
35. Pennsylvania–New Jersey–Maryland (PJM) interconnection. http://www.pjm.com
36. Kumar DMV, Reddy GN, Venkaiah C (2006) Available transfer capability (ATC) determination using intelligent techniques. In: IEEE power India conference, New Delhi, p 6
37. Park C (1997) Electric load forecasting using an artificial neural network. IEEE Trans Power Syst 6(2):412–449
38. CContreras J, Esp'ınola R, Nogales FJ, Conejo AJ (2003) ARIMA models to predict next-day electricity prices. IEEE Trans. Power Syst 18(3):1014–1020
39. Shahidehpour M, Yamin H, Li Z (2002) Market operations in electric power systems, forecasting, scheduling and risk management. Wiley, New York