ELSEVIER

Electric Power Systems Research 63 (2002) 185–196

ELECTRIC POWER SYSTEMS RESEARCH

www.elsevier.com/locate/epsr

Short-term load forecasting based on artificial neural networks parallel implementation

K. Kalaitzakis *, G.S. Stavrakakis, E.M. Anagnostakis

Department of Electronics and Computer Engineering, Technical University of Crete, GR-73100, Chania, Greece

Received 5 February 2002; received in revised form 23 April 2002; accepted 6 May 2002

Abstract

This paper presents the development and application of advanced neural networks to face successfully the problem of the short-term electric load forecasting. Several approaches including Gaussian encoding backpropagation (BP), window random activation, radial basis function networks, real-time recurrent neural networks and their innovative variations are proposed, compared and discussed in this paper. The performance of each presented structure is evaluated by means of an extensive simulation study, using actual hourly load data from the power system of the island of Crete, in Greece. The forecasting error statistical results, corresponding to the minimum and maximum load time-series, indicate that the load forecasting models proposed here provide significantly more accurate forecasts, compared to conventional autoregressive and BP forecasting models. Finally, a parallel processing approach for 24 h ahead forecasting is proposed and applied. According to this procedure, the requested load for each specific hour is forecasted, not only using the load time-series for this specific hour from the previous days, but also using the forecasted load data of the closer previous time steps for the same day. Thus, acceptable accuracy load predictions are obtained without the need of weather data that increase the system complexity, storage requirement and cost. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Short-term load forecasting; Moving window regression training; Gaussian encoding neural networks; Radial basis networks; Real time recurrent neural networks

1. Introduction

Load forecasting is a very crucial issue for the operational planning of electric power systems, especially for the isolated ones. Short-term load forecasting (STLF) aims at predicting electric loads for a period of minutes, hours, days, or weeks. STLF plays an important role in the real-time control and the security functions of an energy management system. STLF applied to the system security assessment problem, especially in the case of increased renewable energy sources (RES) penetration in isolated power grids, can provide, in advance, valuable information on the detection of vulnerable situations. Long- and the medium-term forecasts are used to determine the capacity of generation, transmission, or distribution system additions, along with the type of facilities required in transmission expansion planning, annual hydro and thermal maintenance scheduling, etc. Short-term forecasts are needed not only for power system control and dispatching, but also as inputs to load-flow study or contingency analysis.

A short-term load forecast for a period of 1–24 h ahead is important for the daily operations of a power utility. It is used for unit commitment, energy transfer scheduling and load dispatch. With the emergence of load management strategies, the short-term forecast plays a broader role in utility operations, especially in the case of isolated power grids with increased RES penetration, as in the case of Crete Island. Development of an accurate, fast and robust STLF methodology is crucial to both, the electric utility and its customers.

Many techniques have been proposed during the last few decades regarding STLF [1]. Traditional techniques applied to STLF include Kalman filtering, the Box and Jenkins method, regression models, the autoregressive (AR) model and the spectral expansion techniques [1,2].

* Corresponding author

Time-series models employ extrapolation of historical data for the estimation of future hourly loads. A disadvantage of this type of models is that the load trend is considered as stationary and weather information or any other factors that contribute to the load behavior cannot be fully utilized. Regression models analyze the relationship among loads and other influential factors, such as weather conditions and consumers behavior. The main disadvantage of this kind of models is that complex modeling techniques and heavy computational efforts are required to produce reasonably accurate results [1–5].

An enormous upwelling of interest has grown in recent years in application of artificial intelligence techniques to industrial processes. Their advantage is that no complex mathematical formulation or quantitative correlation between inputs and outputs is required. Many years' data are also not necessary. The effective performance of artificial intelligence in the context of ill-defined processes has led to successful application in load forecasting procedures. As a consequence, pattern recognition [6], expert systems [7,8] and neural networks [9–15,17] have been proposed for electric load forecasting. Expert system based methods capture the expert knowledge into a comprehensive database, which is then used for predicting the future load. These models exploit knowledge of human experts for the development of rules for forecasting. However, transformation of an expert knowledge to a set of mathematical rules is often a very difficult task.

The artificial neural network (ANN) based models are the most popular ones for load forecasting applications. The advantage of ANN over statistical models lies in its ability to model a multivariate problem without making complex dependency assumptions among input variables [9,11,13,15,17]. Furthermore, the ANN extracts the implicit non-linear relationship among input variables by learning from training data.

An alternative technique for load forecasting using Recurrent High Order Neural Networks is considered in [11], [15]. This type of neural network is supposed, in theory, to approximate very accurately any non-linear function, with exponential error convergence to zero. Based on an interpolation procedure, the method gives superb results in the training phase, but in the testing phase the results are rather disappointing.

Most of the above neural based forecasting approaches can generally been classified into two categories in accordance with the employed techniques. The first category treats the load pattern as a time-series signal and predicts the future load using the already mentioned time-series analysis techniques. In the second category, the load pattern is considered to be heavily dependent both, on weather variables (temperature, humidity, etc.) and previous load patterns. Inaccuracy of weather forecasts, difficulties in weather-load relationship modeling and implementation problems limit the use of load forecasting models requiring weather data, thus several works have appeared recently omitting the use of weather data [9,11,17,23,24].

The ANN forecasting models, proposed in this paper, trace previous load patterns in an innovative way (see Section 8 below) and use recent load data to predict a load pattern with sufficient accuracy, thus eliminating the need for weather information. The ANNs structures implemented and illustrated in the present paper are: multi-layer perceptron (MLP), adaptive learning rate backpropagation (BP), Gaussian encoding (GE) BP, random activation weight networks (RAWNs), moving window regression trained RAWN (MWRAWN), radial basis function networks (RBFNs), real-time recurrent networks and AR recurrent networks.

The proposed neural network models are trained to identify the load model that reflects the stochastic behavior of the hourly load demand of the island of Crete in Greece. The results obtained are compared to those from the current standard utility practice (statistical method), providing useful comparative conclusions and observations for all described methods.

2. The autoregressive STLF method used by the utility

Although this work is primarily oriented to neural network approaches for STLF, the statistical method currently used in the utility of Crete Island is also modeled for comparison purposes. This method is the well-known stochastic AR approach and is applied to the same load data.

The original AR model is expressed by the equation:

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_p y_{t-p} + b_t \tag{1}$$

where y_t, y_{t-1},...,y_{t-p} are the present and previous values of the time series, a_1,a_2,...,a_p are the unknown weighting coefficients of these values, a_0 is a constant term and b_t is the random noise term.

The computation of the unknown coefficients of the AR model, a_0, a_1, a_2,...,a_p, is performed applying the recursive least squares method, in conjunction with the well-known U-D decomposition algorithm for numerical calculations and precision optimization. The order p of the AR model is approximated using the Akaike's information criterion [1,2,15]. The value of 4 for the order p is derived applying the Akaike's criterion on the test data (60% of the available data set) of this paper. Attempts to increase the model order, for the sake of improved accuracy, results in a severe deterioration of its performance.

The STLF results of the AR model for the minimum load case (2h00) and the maximum load case (14h00) of the Crete Island utility are shown in Fig. 1 and Fig. 2, respectively.

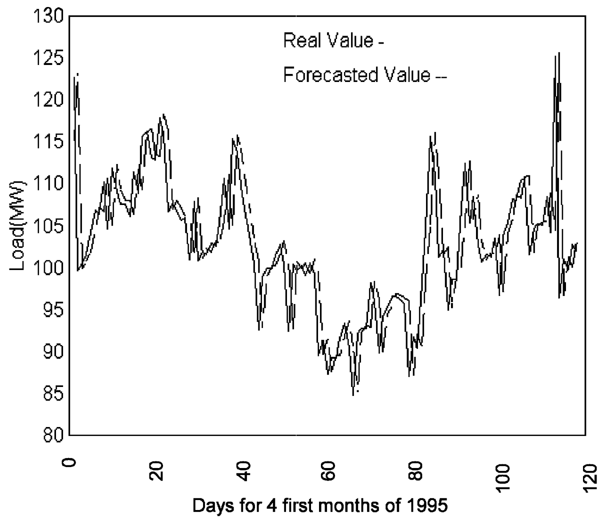

Fig. 1. Load forecasting of 2h00 using the AR model.

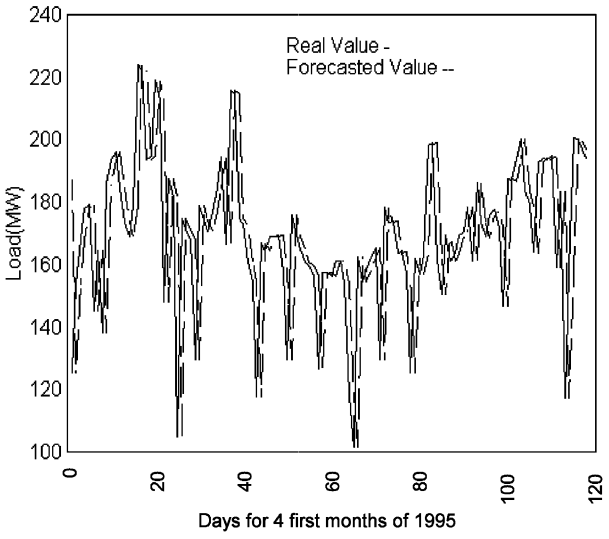

Fig. 2. Load forecasting of 14h00 using the AR model.

3. Advanced backpropagation algorithms

The MLP neural network, trained by the standard BP algorithm, is the most widely used approach for complex mappings forming between input and output and its mathematical properties for non-linear function approximation are well-documented [16,20]. The generalized delta rule is applied to adjust the weights of the feed forward networks thus minimizing a predetermined cost error function. The weights are adjusted using the following rule:

$$w_{ik}^{p}(t+1) = w_{ik}^{p}(t) + m\delta_k^p y_i^p + \alpha \Delta w_{ik}^{p}(t) \tag{2}$$

where m is the learning rate parameter, a is the momentum term ranging from 0 to 1 and d is the negative derivative of the total square error in respect to the neuron's output.

Training of MLPs using standard BP algorithms suffers from slow convergence. A simple heuristic strategy called the adaptive learning rate backpropagation neural network (ALRBP) has been adopted in this paper. The ALRBP relates the learning rate with the total error function E achieving acceleration of the convergence speed [18,20]. The algorithm uses the batch update mode, hence the update rule for the weights is:

$$w_{ij}^{p}(t+1) = w_{ij}^{p}(t) - \frac{q(E)\dfrac{\partial E}{\partial w_{ij}^{p}}}{\left\| \dfrac{\partial E}{\partial w_{ij}^{p}} \right\|^{2}} \tag{3}$$

where $q(E)$ is a function of the error E given by:

$$q(E) = \begin{cases} m \\ mE \\ m \tan \mathrm{h}\left(\dfrac{E}{E_{\mathrm{o}}}\right) \end{cases} \tag{4}$$

where m and E_{o} are constant, non-negative numbers, representing the learning rate and the error normalization factor, respectively. The main advantage of the above consideration is the dependence of the learning rate on the instantaneous value of the total squared error E. Therefore, a faster convergence of the algorithm is achieved, although there is a possibility for 'jumpy' behavior of the weights around a local minimum. In order to achieve satisfactory prediction, a structure with 2 hidden layers and 8 inputs is used. The ALRBP network topology applied to the Crete utility load data has 8 input nodes, 20 and 10 nodes for the two hidden layers and 1 output node.

Since the load forecasting procedure follows the dynamics of a non-linear system, a common approach is to configure and train a neural network to represent a non-linear AR model structure. It is a natural perception to reflect the dynamic nature of the problem by sequential information processing.

It has been found that a major factor affecting the neural model prediction accuracy is the data coding method. The conventional data conditioning method is re-scaling and representing them using a single node at the input or output network layers. An alternative representation, called Gaussian encoding (GE), a particular case of spread encoding, proved to ensure high degree of accuracy for the neural network [19].

According to the GE technique, each data value is represented as the mean value of a sliding Gaussian pattern of excitation over several nodes at the network input and output. The reverse procedure is applied at the network output to decode the values back into the original variable range. The encoding procedure has similarities with data fuzzification techniques, where the scalar dimensional space of each variable is fuzzified to a higher dimensions space. Also, decoding of the network output using the GE method is performed computing a weighted summation of the node excitations, which resembles the conventional center of gravity defuzzification technique. Thus, a network featuring GE can be considered as a fuzzy-neural-type network.

Analytically, the GE data conditioning method is formed by mapping each network variable, $x \in [x_{\min}, x_{\max}]$, *onto a sliding Gaussian activation pattern of N* network nodes, which includes additional nodes either side of the variable range. Hence, over-spilling resulting from a mapping function with wide support is included. Each node activation level is confined to be in the range [0.1, 0.9] similar to the conventional normalization technique. Each node is assigned a value, α_i, linearly spaced by a distance δ, to span the range of x. The center of the Gaussian excitation pattern corresponds to the value coded.

The GE algorithm is formed creating a discrete map, which represents the mean value of a normalized Gaussian continuous probability distribution, $\phi(\alpha)$, within each class interval. This provides a simple mechanism for retrieving the original coded value as a sum of the node excitations activity, each weighted by the values at the centers of the class intervals α_i. For a particular value of x, each node excitation is defined by $\psi_i(x)$, which satisfies the requirement:

$$\sum_{i=1}^{N} \alpha_i \psi_i(x) = \int \alpha \phi(\alpha - x)\,\mathrm{d}\alpha = \bar{\alpha} = x \tag{5}$$

given that the distribution $\phi(\alpha)$ has unit area. The integral term in Eq. (5) is approximated using the first two terms of the trapezoidal approximation resulting in:

$$\psi_i(x) \approx \Phi(\alpha_i + \delta/2 - x) - \Phi(\alpha_i - \delta/2 - x) \tag{6}$$

which ensures sufficient accuracy in the case studies of this work.

The first step of the GE algorithm is the data scaling to a normalized range. Specifically, the original data range $r \in [r_{\min}, r_{\max}]$ is represented by $x \in [0, N-2N_0]$, where N is the total number of nodes, N_0 is the number of nodes on either side of the variable range and $\delta = 1$. The data coding is based on the following relation, resulting from a Taylor series expansion of the cumulative function around the interval center, α_i:

$$\psi_i(x) = \Phi(\alpha_i + 1/2 - x) - \Phi(\alpha_i - 1/2 - x), \quad i = 1, \dots, N \tag{7}$$

where:

$$\alpha_i = i - N_0 - c \tag{8}$$

and the cumulative Gaussian distribution function is approximated by a sigmoid function centered at x:

$$\Phi(\alpha - x) = \frac{1}{1 + \mathrm{e}^{-\beta(\alpha - x)}} \tag{9}$$

In Eq. (8), c is an offset term, which shifts the position of the nodes range limits. The width of the node excitations is inversely controlled by the parameter β in Eq. (9). Errors arise in decoding if Eq. (5) is straightforward applied, because the node excitations $\psi_i(x)$ are calculated by the approximation of Eq. (6). The accuracy of decoding is improved dividing the weighted sum by the node excitations sum. Thus, the network output is decoded back to the normalized range using:

$$x = \frac{\sum_{i=1}^{N} \alpha_i \psi_i(x)}{\sum_{i=1}^{N} \psi_i(x)} \tag{10}$$

which is similar to the conventional center of gravity defuzzification technique. The parameters used in the spread encoding algorithm implemented here are $N = 6$, $N_0 = 2$, $c = 0.5$ and $\beta = 2.3$, which were estimated by trial-and-error procedures to provide sufficiently accurate coding and decoding in the case studies of this paper.

The practical advantage of GE is better accuracy of derived models, due to the use of static feed-forward neural networks rather than representing normalized physical variables using single nodes. This is attributed to signal noise reduction related to spread encoding representations, because the coding function is appropriately matched with the interval width spanned for each node. Thus, the network's fault tolerance is enhanced.

Formal techniques for determination of optimum number of nodes in the hidden layers are still under research. Currently, this task is often accomplished by experimentation. The resulting MLP network topology with GE, applied to the case study load data, consists of 48 input nodes, 34 and 16 nodes for the two hidden layers and 6 output nodes. The statistical indices of the STLF results for the minimum load case (2h00) and the maximum load case (14h00) are illustrated in Table 1 and Table 2, respectively.

4. RAWN and MWRAWN

A general function approximation can be obtained by feed-forward neural networks consisting of just one hidden layer of non-linear neurons. The innovative idea behind RAWN networks is that training of the weights between the input and the hidden layer is not required. Initiating the activation weights as random numbers, the parameters estimation process can be considered of linear type, thus a linear least-squares estimator [21] can be used.

Table 1
Statistical indices for the 2h00 hour load forecasting (min load case)

Models	AR	BP	ALRBP
Relative error (%)	3.86297	4.22628	2.73460
Standard deviation error	5.90894	5.16192	3.91741
RMSE (%)	5.80199	5.18811	3.80634
Models	GE	RAWN	MWRAWN
Relative error (%)	1.51737	6.82535	2.75417
Standard deviation error	2.13450	9.24337	4.19438
RMSE (%)	2.03138	8.98179	4.06330
Models	ZORRBF	RTRL	ARNN
Relative error (%)	1.35119	2.98850	1.22330
Standard deviation error	1.96127	4.35816	1.83768
RMSE (%)	1.86722	4.08409	1.71771

Table 2
Statistical indices for the 14h00 hour load forecasting (max load case)

Models	AR	BP	ALRBP
Relative error (%)	11.18372	13.40970	11.9674
Standard deviation error	25.60678	27.20622	24.3568
RMSE (%)	17.20486	19.04599	17.6778
Models	GE	RAWN	MWRAWN
Relative error (%)	3.40010	11.81696	10.59024
Standard deviation error	7.71597	26.29459	21.74416
RMSE (%)	4.61941	17.54215	15.19347
Models	ZORRBF	RTRL	ARNN
Relative error (%)	2.79244	11.5430	2.73292
Standard deviation error	5.77855	24.2722	5.92847
RMSE (%)	3.60609	16.4966	3.68308

The RAWN network equations can be written easily in matrix form. The input vectors are grouped into a matrix, with one row for each element $X=(x(1)\dots x(k)\dots x(N_e))^T$. Similarly, the output is defined as $Y=(y(1)\dots y(k)\dots y(N_e))^T$. Then, the neural net can be written concisely as:

$$\begin{array}{ll} Z=XW^h & [N_e\times N_h] \\ U=f(Z)=\alpha\tan h(bZ) & [N_e\times N_h] \\ Y_n=UW^o & [N_e\times N_o] \end{array} \tag{11}$$

where W^h is the weight matrix between the input and hidden layer, U is the output matrix of hidden layer, W^o is the weight matrix between the hidden and output layer and α, b are the magnitude and phase of the activation function respectively. Let Y_d be the output data in the training set related to the input X. The matrix Y_d is generated by the process:

$$Y_d=UW^o+e \tag{12}$$

where e is the error term. Furthermore, the weights W^h are supposed to be fixed. Then, the training of the net is the determination of the weights W^o that minimize the difference between the neural net output $Y(=UW^h)$ and the target output Y_d. This is an ordinary least squares type problem. Provided that the matrix U is of full rank (i.e. has a rank of N_h), the least-squares solution is:

$$\bar{W}^o=(U^TU)^{-1}U^TY_d \tag{13}$$

The matrix U must have N_h linear independent columns. In an opposite case, the matrix U^TU is singular, thus no solution of Eq. (13) exists. If U would be formed from Z with a linear function transformation, then U would have the rank of Z. Since XW^h is an outer product, Z consists of linear-dependent columns. Consequently, U would not be of full rank. However, in the case of a non-linear activation function, U is of full rank, because the dependent columns of Z are transformed in a non-linear way, provided that no two columns of Z are exactly the same. The latter can be easily achieved giving random values to the activation weights W^h.

Iterative methods updating the estimate and accelerating the calculations, whenever information is available, can be used. For fast and accurate calculations, a moving window regression method for the RAWN network training is implemented here, which is described and proposed for fist time in [22].

Lets define:

$$U=\begin{bmatrix} u_1(1) & \cdots & u_m(1) \\ \vdots & & \vdots \\ u_1(k) & \cdots & u_m(k) \end{bmatrix}=\begin{bmatrix} u^T(1) \\ \vdots \\ u^T(k) \end{bmatrix} \quad \text{and}$$

$$Y_d=[y_d(1)\dots y_d(k)]^T \tag{14}$$

For a moving window of length n_w, the following definitions are used [22]:

$$U_k=\begin{bmatrix} u^T(k-n_w+1) \\ \vdots \\ u^T(k) \end{bmatrix}=\begin{bmatrix} u^T(k-n_w+1) \\ \vdots \\ U(k,k-n_w+2) \end{bmatrix} \tag{15a}$$

$$U_{k+1}=\begin{bmatrix} u^T(k-n_w+2) \\ \vdots \\ u^T(k+1) \end{bmatrix}=\begin{bmatrix} U(k,k-n_w+2) \\ \vdots \\ u^T(k+1) \end{bmatrix} \tag{15b}$$

As shown in the appendix of Ref. [22], a moving window least-squares estimate can be derived for the weights by the following recursive equations:

$$\bar{W}^o(k+1)=\bar{W}^o(k)-P(k+1)\times[\Gamma(k+1)\bar{W}^o(k)-\delta(k+1)] \tag{16}$$

where:

$$P^{-1}(k+1)=(U_{k+1}^TU_{k+1})^{-1}=P^{-1}(k)+\Gamma(k+1) \tag{17}$$

is the covariance matrix of the estimate $W^o(k+1)$. The quantities δ and Γ are defined as:

$$\delta(k+1) = u(k+1)y(k+1) - u(k-n_w+1)y(k-n_w+1) \quad (18a)$$

$$\Gamma(k+1) = u(k+1)u^T(k+1) - u(k-n_w+1)u^T(k-n_w+1) \quad (18b)$$

The major advantage of this training method is that $\hat{W}^o(t)$ is estimated using information from the last n_w samples. As a result, there is a increase in speed, which is proportional to the window length, since the dimensions of P, Γ and δ are independent of the window size.

The main objective in this method, referred as MWRAWN in the following paragraphs, is right choice of the activation weights W^h. To avoid for the matrix $U^T U$ to come close to singularity, the sigmoid function should always operate in linear range. This is achieved with normalization of the input vector in the range $[-a,a]$, where a is a small number in the range $0 < a < 1$, and scaling the randomly chosen weights W^h in an efficient manner. Hence, the following equation is satisfied:

$$\max_k \operatorname{var}_j \left\{ \sum_{\ell=1}^{N_i+1} x_\ell(k) W^h_{\ell j} \right\} \leqslant a \quad (19)$$

The weight matrix W^h is randomly generated by a normal distribution with zero mean and standard deviation equal to 1. Then, the normalized weight matrix W^h is given by the following equation:

$$W^h = \sqrt{\frac{a}{\max_k \sum_{\ell=1}^{N_i+1} x^2_\ell(k)}} N(0,1) \quad (20)$$

where $N(0, 1)$ denotes a generator of random numbers with zero mean normal distribution and standard deviation equal to 1. For the case study under consideration, best results are obtained using an 8/20/1 RAWN and MWRAWN structure, while the statistical indices of the STLF results for the minimum load case (2h00) and the maximum load case (14h00) are illustrated in Table 1 and Table 2, respectively.

5. Radial basis functions networks

Another type of hybrid network, which features the architecture of the instar-outstar model and uses the hybrid unsupervised and supervised learning scheme, is the RBFN suggested by Moody and Darken [16,20]. The RBFN is designed to perform input-output mapping trained by examples (x_k, d_k), $k = 1, 2, \ldots, p$. The RBFN is based on the concept of the locally tuned and overlapping receptive field structure, studied in the cerebral cortex, the visual cortex, etc. Unlike the instar-outstar model, in which the hidden nodes are of linear winner-take-all-nodes type, the RBFN hidden nodes follow a normalized Gaussian activation function:

$$z_q = g_q(\mathbf{x}) \triangleq \frac{R_q(\mathbf{x})}{\sum_{k=1}^{\ell} R_k(\mathbf{x})} = \frac{\exp[-|\mathbf{x} - m_q|^2/2\sigma_q^2]}{\sum_{k=1}^{\ell} \exp[-|\mathbf{x} - m_k|^2/2\sigma_k^2]} \quad (21)$$

where $\mathbf{x}$ is the input vector. Thus, the hidden node q gives a maximum response to input vectors close to m_q. Each hidden node q is said to have its own receptive field $R_q(x)$ in the input space, which is a region centered on m_q with size proportional to σ_q, where m_q and $\sigma_q{}^2$ are the mean (an m-dimensional vector) and variance of the qth Gaussian function, respectively. Gaussian functions are a particular example of radial basis functions. The output of the RBFN is simply the hidden node output weighted sum:

$$y_i = \alpha_i \left(\sum_{q=1}^{\ell} w_{iq} z_q + \theta_i \right) \quad (22)$$

where $\alpha_i(\cdot)$ is the output activation function and θ_i is the threshold value. Generally, $\alpha_i(\cdot)$ is an identity function (i.e. the output node is a linear unit) and $\theta_i = 0$.

The present work adopts a systematic approach to the problem of center selection. Because a fixed center corresponds to a given regressor in a linear regression model, the selection of RBF centers can be regarded as a problem of subset selection. The orthogonal least squares method can be employed as a forward selection procedure, which constructs RBFN in a rational way. The algorithm chooses appropriate RBF centers one by one from training data points until a satisfactory network is obtained. Each selected center minimizes the increment to the desired output variance, thus ill conditioned problems, frequently occurring when random center selection is used, can automatically be avoided. In contrast to most learning algorithms, which can only work if a fixed network structure is first specified, the orthogonal least squares algorithm is a structural identification technique, where the centers and estimates of the corresponding weights can be simultaneously determined in a very efficient manner during learning. Orthogonal least squares learning procedure generally produces an RBF network smaller than a randomly selected RBF network [3,20]. Due to its linear computational procedure at the output layer, the RBFN is faster in training time compared to its BP counterpart.

A major drawback of this method is associated with the input space dimensionality. For large numbers of inputs units, the number of radial basis functions required can become excessive. If too many centers are used, the large number of parameters available in the regression procedure will cause the network to be over sensitive to the details of the particular training set and result in poor generalization performance (overfit). To avoid this problem, both the forward selection and zero-order regularization techniques are proposed and applied in [3] to construct parsimonious RBFN with improved generalization properties.

In the present paper, the zero-order regularization RBF (ZORRBF) algorithm proposed in [3] is employed to model the hourly demand load. For the case study under consideration, best results are obtained using 10 inputs. Although an elegant approach to the selection of the regularization parameter λ is to adopt Bayesian techniques, in this work this parameter was set by trial and error to small positive values, which satisfy the optimal problem solution.

The λ parameter is set equal to 0.0002 and 0.0008 for the maximum and minimum load cases, respectively. As a result, the corresponding centers are found by the orthogonal least-squares procedure to be equal to 27 and 13, respectively. For comparison purposes, it should be noted that for the case of maximum load, the original orthogonal least-squares algorithm, without the use of λ parameter, gives a network with similar accuracy but with the computational cost of 69 centers. The STLF results for the minimum load case (2h00) and the maximum load case (14h00) are depicted in Fig. 3 and Fig. 4, respectively.

6. Real-time recurrent networks

One of the most popular training algorithms for recurrent networks based on the gradient descent is the real-time recurrent learning algorithm (RTRL) [19,20]. Consider a network consisting of total N neurons with N_i external inputs. Let $\mathbf{x}(t)$ be the $N_i \times 1$ external input vector applied to the network at time t and let $\mathbf{y}(t+1)$ denote the $N \times 1$ output vector produced

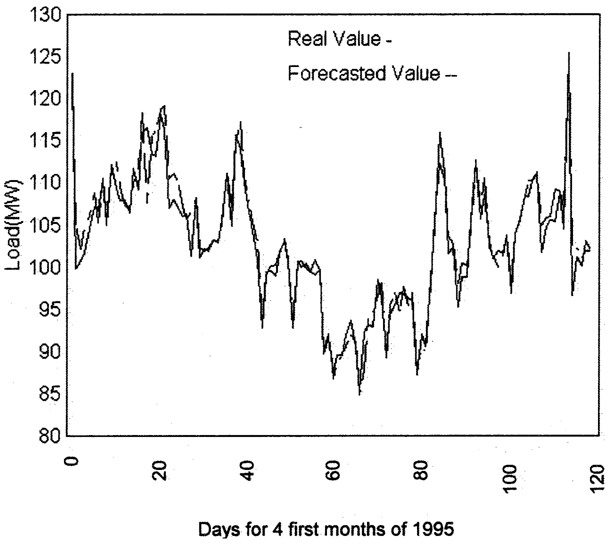

Fig. 3. Load forecasting of 2h00 using the ZORRBF model.

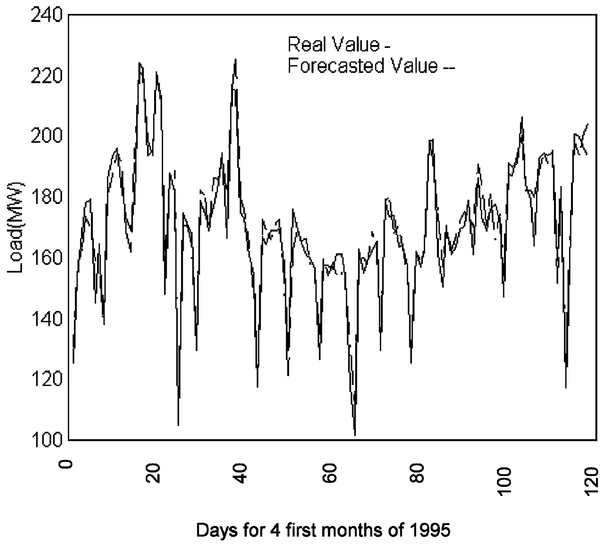

Fig. 4. Load forecasting of 14h00 using the ZORRBF model.

at time $t+1$. The input vector $\mathbf{x}(t)$ and the output vector $\mathbf{y}(t+1)$ are concatenated to form the $(N_i+N) \times 1$ vector $\mathbf{u}(t)$. The network has two layers, a concatenated input-output layer and a processing layer. The network is fully interconnected, thus there are N_iN forward connections and N^2 feedback connections; N of the feedback connections are self-feedback connections.

Let W be the $N(N_i+N)$ weight matrix of the network. The internal activity $v_j(t)$ of neuron j at time t is given by the equation:

$$v_j(t) = \sum_{i=1}^{N_i+N} w_{ji}(t)u_i(t) \tag{23}$$

At the next time step $t+1$, the output of each processing layer neuron is computed passing $v_j(t)$ through the non-linear activation function. This yields:

$$y_j(t+1) = F(v_j(t)) \tag{24}$$

It should be noted that the external input vector $\mathbf{x}(t)$ at time t does not influence the output of any neuron in the network until time $t+1$.

The desired output of neuron j is $d_j(t)$, thus the error $e_j(t)$ of the neuron is given by the equation:

$$e_j(t) = d_j(t) - y_j(t) \tag{25}$$

The objective is the minimization of the following cost function over time t,

$$E_{\text{total}} = \sum_{t=1}^{t_{\text{final}}} E(t) \tag{26}$$

where t_{final} is the final moment of the neural network run.

The steepest descent method is used for the minimization procedure, setting the gradient vector $\nabla_{\text{w}}E_{\text{total}}$ to zero:

$$\frac{\partial E_{\text{total}}}{\partial W} = 0 \Rightarrow \sum_{t=1}^{t_{\text{final}}} \frac{\partial E_{\text{total}}}{\partial W} = 0 \Rightarrow \sum_{t=1}^{t_{\text{final}}} \nabla_{\text{w}} E(t) = 0 \tag{27}$$

where $\nabla_{\text{w}} E(t)$ is the gradient of $E(t)$ with respect to the weight matrix W. In order to develop a learning algorithm that can be used to train the recurrent network in real time, an instantaneous estimate of the gradient $\nabla_{\text{w}} E(t)$ must be used, which leads to an approximation to the steepest descent method.

For the case of a particular weight $w_{k1}(t)$, the incremental change $\Delta W_{k1}(t)$ performed at time t is defined as follows:

$$\Delta w_{k\ell}(t) = -\eta \frac{\partial E(t)}{\partial w_{k\ell}(t)} = \eta \sum_{j=1}^{N} e_j(t) \frac{\partial y_j(t)}{\partial w_{k\ell}(t)} \tag{28}$$

where η is the learning rate parameter.

The use of the chain rule for differentiation, along with Eq. (23) and Eq. (24), yields:

$$\frac{\partial y_j(t+1)}{\partial w_{k\ell}(t)} = \frac{\partial y_j(t+1)}{\partial v_j(t)} \frac{\partial v_j(t)}{\partial w_{k\ell}(t)} = F'(v_j(t)) \frac{\partial v_j(t)}{\partial w_{k\ell}(t)} \tag{29}$$

The derivative $(\partial w_{ji}(t))/(\partial w_{k\ell}(t))$ equals to 1, if $j = k$ and $i = 1$; otherwise it is zero. Thus Eq. (29) becomes:

$$\frac{\partial v_j(t)}{\partial w_{k\ell}(t)} = \sum_{i=1}^{N_i+N} w_{ji}(t) \frac{\partial u_i(t)}{\partial w_{k\ell}(t)} + \delta_{kj} u_\ell(t) \tag{30}$$

where δ_{kj} is a Kronecker delta equal to 1 when $j = k$ and zero otherwise.

The following equations are assumed:

$$\frac{\partial u_i(t)}{\partial w_{k\ell}(t)} = \begin{cases} 0 & \text{if } i \text{ is an external neuron} \\ \dfrac{\partial y_i(t)}{\partial w_{k\ell}(t)} & \text{if } i \text{ is an output neuron} \end{cases}$$

$$\left.\text{At time } t = 0 \text{ it is assumed that } \frac{\partial y_i(0)}{\partial w_{k\ell}(0)} = 0 \right\} \mathbf{Z} \tag{31}$$

Finally, after the necessary calculations, the update rule for an external neuron weights is:

$$w_{k\ell}(t+1) = w_{k\ell}(t) + \eta \sum_{j=1}^{N} e_j(t)\, F'(v_j(t)) \times \sum_{i=1}^{N_i+N} [\delta_{kj} u_\ell(t)] \tag{32}$$

while the update rule for an output neuron weights is:

$$w_{k\ell}(t+1) = w_{k\ell}(t) + \eta \sum_{j=1}^{N} e_j(t)\, F'(v_j(t)) \sum_{i=1}^{N_i+N} \left[w_{ji}(t) \frac{\partial y_j(t)}{\partial w_{k\ell}(t)} + \delta_{kj} u_\ell(t) \right] \tag{33}$$

It should be noted that the initial values of the weights are random numbers obtained from a uniform distribution.

For the STLF case under consideration, the RTRL network is employed as one-step ahead predictor, similarly to the previous approaches. Best results are obtained using a 4/10/1 structure. Attempts to increase the input dimensionality, for the sake of improved accuracy, results in a severe deterioration of its performance. The statistical indices of the STLF results for the minimum load case (2h00) and the maximum load case (14h00) are illustrated in Table 1 and Table 2, respectively.

7. Autoregressive recurrent neural networks

The autoregressive recurrent neural network (ARNN) is a hybrid type feed-forward/feedback neural network, with feedback represented by recurrent connections appropriate for approximating a load time series.

There are two hidden layers, with sigmoid transfer functions, and a single linear output node. The ARNN topology allows recurrence only in the first hidden layer. For this layer, the memoryless BP model has been extended to include an AR memory, a form of self-feedback, where the output depends also on the weighted sum of previous outputs.

For the training of the ARNN, a modified BP algorithm was developed, which includes dynamic recursive functions of time [20]. The mathematical definition of the ARNN is analyzed below:

$$y(t) = O(t) = \sum_{\ell} W_\ell^O Q_\ell(t), \quad Q_\ell = f(S_\ell), \quad S_\ell = \sum_j W_{j\ell}^H Z_j(t) \tag{34}$$

and:

$$Z_j(t) = f(H_j(t)),$$

$$H_j(t) = \sum_{k=1}^{k=n} W_{jk}^D Z_j(t-k) + \sum_i W_{ij}^I I_i(t) \tag{35}$$

where $I_i(t)$ is the ith input to ARNN, $H_j(t)$ is the sum of inputs to the jth recurrent neuron in the first hidden layer, $Z_j(t)$ is the output of the jth recurrent neuron, $S_1(t)$ is the sum of inputs to the lth neuron in the second hidden layer, $Q_l(t)$ is the output of the lth neuron in the second hidden layer and $O(t)$ is the output of the ARNN. Here, $f(\cdot)$ denotes the sigmoid function and W^I, W^D, W^H, and W^O are the input, recurrent, hidden and output weights, respectively. The index n indicates the number of internal memories in the recurrent layer hidden nodes. For the case study under consideration, the index n was set by trial and error equal to five ($n =$ 5).

Let $d(t)$ and $y(t)$ be the desired and actual responses of the ARNN, respectively. An error function for a training cycle can be defined as

$$E = \frac{1}{2}[d(t) - y(t)]^2 \tag{36}$$

The weights can now be adjusted following the steepest descent method, thus the weights update rule becomes:

$$W(n+1) = W(n) + n\left(-\frac{\partial E}{\partial W}\right) + \alpha \Delta W(n) \tag{37}$$

where:

$$\frac{\partial E}{\partial W} = -e(t)\frac{\partial y(t)}{\partial W} = -e(t)\frac{\partial O(t)}{\partial W} \tag{38a}$$

$$\frac{\partial O(t)}{\partial W_\ell^O} = Q_\ell(t) \tag{38b}$$

$$\frac{\partial O(t)}{\partial W_{j\ell}^H} = f'(S_\ell)W_\ell^O Z_j(t) \tag{38c}$$

$$\frac{\partial O(t)}{\partial W_{jk}^D} = \delta_j(t)\frac{\partial Z_j(t)}{\partial W_{jk}^D} \tag{38d}$$

$$\frac{\partial O(t)}{\partial W_{ij}^I} = \delta_j(t)\frac{\partial Z_j(t)}{\partial W_{ij}^I} \quad \text{with} \quad \delta_j(t) = \sum_\ell f'(S_\ell)W_\ell^O W_{j\ell}^H \tag{38e}$$

and:

$$\frac{\partial Z_j(t)}{\partial W_{jk}^D} = f'(H_j)\left(Z_j(t-k) + \sum_{n=1}^{n=5} W_{jn}^D \frac{\partial Z_j(t-n)}{\partial W_{jk}^D}\right),$$

$$\frac{\partial Z_j(0)}{\partial W_{jk}^D} = 0 \tag{39a}$$

$$\frac{\partial Z_j(t)}{\partial W_{ij}^I} = f'(H_j)\left(I_i(t) + \sum_{n=1}^{n=5} W_{jn}^D \frac{\partial Z_j(t-n)}{\partial W_{ij}^I}\right), \quad \frac{\partial Z_j(0)}{\partial W_{ij}^I} = 0 \tag{39b}$$

The above-described training procedure is an innovation. For the STLF case under consideration, an ARNN with 8 inputs is employed, giving a structure of 8/20/14/1 nodes. The computer runs reveal that the proposed training procedure is faster compared with the standard MLP structures.

The STLF results for the minimum load case (2h00) and the maximum load case (14h00) are shown in Fig. 5 and Fig. 6, respectively.

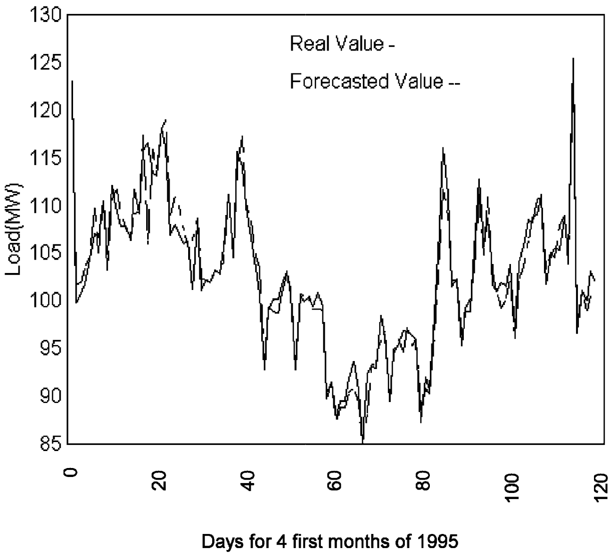

Fig. 5. Load forecasting of 2h00 using the ARNN model.

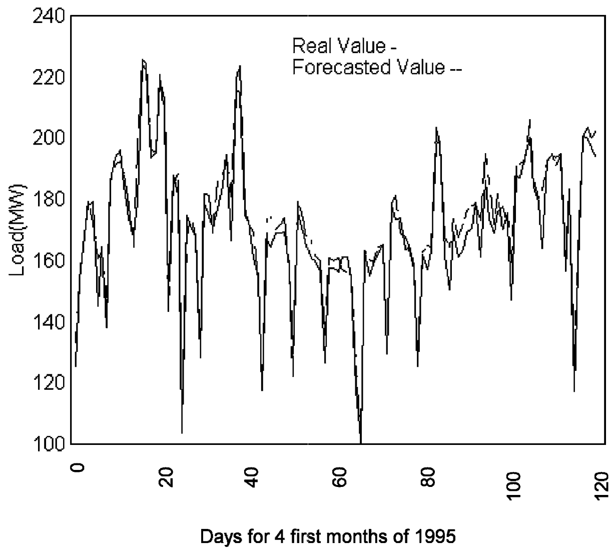

Fig. 6. Load forecasting of 14h00 using the ARNN model.

8. The 24 h ahead load prediction approach

The objective of STLF is to predict the n hourly loads ahead, where $n \leqslant 24$. A parallel processing forecasting procedure is proposed here, where n-neural blocks with a single output have been implemented and trained separately to provide the n hourly ahead load forecasts. Each neural block is fed by its precedent one. Hence, step-by-step, a n-hour ahead load prediction is obtained. According to this procedure, the requested load for each specific hour is forecasted, not only using the load time-series for this specific hour from the previous days, but also using the forecasted load data of the closer previous time steps for the same day. For the case studies of this work, the training data set consists of the hourly load data for the whole year 1994 (60% of the available data set). The testing set consists of the hourly load data of first 4 months of 1995 (40% of the available data set). The training and the testing sets are classified into 24 time-series, each one corresponding to an hour of the day.

The above-described parallel processing implementation improves significantly the results obtained from any forecasting model compared to those from the same forecasting model implemented without parallel processing. This is attributed to the proposed modular parallel architecture of 24 separate neural blocks, each one with a single output, featuring easier and faster training compared to traditional neural approaches, which treat the output as a unique 241 vector. Thus the accumulation and propagation of prior hours error is minimized. In addition, acceptable accuracy load predictions are obtained without the need of huge amounts of weather data that increase the system complexity, storage requirement and cost [2,4,7,9,12,17,23,24].

The proposed methodology features a very limited sensitivity to the peak variation from 1 day to the next, taking into account that, except the use of the previous days load history, the load value of just the previous hour is also used. This forces the model to follow the actual load variation trend.

9. Results and discussion

Forecast results and statistical properties obtained from application of the developed STLF ANN structures on the autonomous power system of the island of Crete (Greece) are presented and discussed in this section. Case studies for all proposed methods were carried out for a 24-h load forecasting. The STLF results for the utility of Crete Island, produced by all ANN structures presented here, are analyzed and compared on the basis of the following well-known statistical indices:

Standard deviation error:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[y_i - \bar{y}_i]^2} \tag{40}$$

Percent relative error:

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| - \frac{100}{y_i} \tag{41}$$

Root mean square error:

$$\varepsilon_1 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{y}_i}{y_i}\right]^2} - 100 \tag{42}$$

The STLF results for the hours with minimum and maximum load consumption are illustrated in the Table 1 and Table 2, respectively.

The AR method is the current standard utility practice used widely for STLF applications [4,5,7,9,12,17,18]. The AR approach along with the standard MLP combined with either a BP or a ALRBP learning algorithm, produce comparable results, thus they are considered as test bed cases in this work. On the other hand, the proposed GE structure provides a considerably higher degree of accuracy compared to the classic MLP structure. Although this performance improvement is generally at the expense of a larger network size, the use of the GE structure has significant advantages in applications requiring long-range predictions. The performance of a classical MLP acting as a recurrent model deteriorates severely, because errors of the predicted output tend to accumulate. This problem is remarkably limited with the proposed GE structure, where the error splits to several nodes providing extended network fault tolerance.

The RAWN algorithm is a least squares type method offering faster training. This algorithm uses a non-iterative training method for the computation of the set of weights and the corresponding results approximate those from a classic MLP using the BP learning rule. The proposed MWRAWN algorithm features the moving window regression method for network training, improving considerably the STLF accuracy.

An alternative STLF approach is a neural model employing radial basis functions (RBF). The presented ZORRBF algorithm improves both, training time and accuracy. An additional advantage of the specific algorithm is the elimination of the overfitting problem by the use of the regularized parameter λ. Because of this, the ZORRBF network has a very compact structure and achieves significant STLF accuracy improvement, compared to all other neural architectures presented here.

An innovation to the STLF problem is the use of recurrent neural networks. Here, the presented RTRL and/or the ARNN approaches use some kind of memory to encode past history, with objective the short training time. The resulting improved networks, compared to the standard MLP structures, reveal the advantages of memory neuron structures. Incorporation of five memories in the ARNN structure and related recurrence in the first hidden layer, enable the network to carry out five-steps ahead accurate predictions. Although the ARNN method depends on the number of 'memories' and therefore it is considered as a partially recurrent network, its use in the proposed modular parallel architecture allows expansion of the prediction horizon beyond the 24 h limit. The most significantly accurate STLF results are obtained applying the ARNN and the ZORRBF structures.

A general remark concerning the STLF accuracy is that he forecasting model is trained using load data from the isolated power system of the island of Crete. The load profile includes industrial, commercial and domestic loads. Small power systems experience wider load fluctuations reflecting higher RMSE values. It is expected that the relevant RMSE of large interconnected systems would be in the range of 1.0–1.5%.

All the above described models were developed using either the MATLAB 4.2c (adaptive learning rate BP, RAWN, MWRAWN, real-time recurrent networks) or the ANSI-C (MLP, GE BP, AR recurrent networks) environment. The computer was an Intel Pentium III at 500 MHz, similar to the one used by PPC in the Crete Island SCADA system, which is interfaced with the main workstation of the SCADA system for load forecasting purposes. The time for ANN learning takes 1–2 min for the structures ALRBP, GE, ZORRBF, RTRL and ARNN, while it takes 40–50 s for the structures AR, BP, RAWN and MWRAWN. The delay for prediction is about 10–25 s, depending on the structure complexity.

The weather influence on the load forecasting is captured by the adaptation mechanism inherent in both, the ANN models non-linear structure and the innovative implementation procedure presented in the Section 8 above. The weather influence can be better captured by retraining the models on a season or an even shorter period basis rather than on a whole year basis. This is due to the fact that the ANN models presented here, as well as the parallel implementation procedure of Section 8, can be applied to any kind of load time series (peak values, holidays values, specific period average values, etc), because they are of general structure and can be rather used as generic prediction tools. The models fine-tuning (weights retraining) is very easy and can be performed using the well known moving window techniques [15,22].

10. Conclusions

In this paper, a comparative analysis of ANN based STLF techniques is presented, using the load data of the Greek island of Crete. These methods are applied for 1-day-ahead prediction of the hourly electric load and employ a modular parallel architecture, based on a separate forecasting module for each hourly load. Several advanced neural architectures were tested including multilayer perceptrons, RAWNs, radial basis, recurrent neural networks and their innovative variations, presented in the present paper.

The obtained results reveal the advantages of the proposed neural approaches compared to the widely used AR and multilayer perceptron (with the standard BP algorithm) techniques. It should be noted that the proposed structures differ from usual time-series prediction models since they take into account load data of previous hours and previous days for hourly load forecasting.

The case study proves that the ZORRBF, the ARNN and the GE structures lead to the most accurate STLF results. The STLF accuracy achieved by the above structures, applied using the proposed modular parallel architecture, is significantly satisfactory, thus the need for additional load and/or weather information is not necessary.

References

[1] IEEE Committee report, Load forecasting bibliography Phase I, IEEE Trans. Power Appl. Syst. 99 (1980) 53–58.

[2] I. Moghram, S. Rahman, Analysis and evaluation of five short-term load forecasting techniques, IEEE Trans. Power Syst. 4 (4) (1989) 1484–1491.

[3] S. Chen, E. Chng, Regularized orthogonal least squares algorithm for constructing radial basis function networks, Int. J. Control 64 (5) (1996) 829–837.

[4] G. Irisarri, S. Widergren, On-Line load forecasting for energy control center applications, IEEE Trans. Power Appl. Syst. 101 (1) (1982) 71–78.

[5] K. Jabbour, J.F.V. Riveros, D. Landsbergen, ALFA: automated load forecasting assistant, IEEE Trans. PWRS 3 (1988) 908–914.

[6] J. Hertz, A. Krogh, R.G. Palmer, Introduction to Theory of Neural Computation, Addison-Wesley, 1991.

[7] K.L. Ho, et al., Short-term load forecasting of Taiwan power system using a knowledge based expert system, IEEE Trans. Power Syst. 5 (4) (1990) 1214–1219.

[8] S. Rahman, R. Bhatnagar, An expert system based algorithm for short-term load forecast, IEEE Trans. Power Syst. AS-101 (1982) 9.

[9] K. Lee, Y. Cha, C. Ku, A study on neural networks for short-term load forecasting, Proc. ANNPS '91, Seattle, July (1991) 26–30.

[10] C.N. Lu, H.T. Wu, S. Vemuri, Neural network based short-term load forecasting, IEEE Trans. Power Syst. 8 (1) (1993) 337–342.

[11] G.N. Kariniotakis, E. Kosmatopoulos, G. Stavrakakis, Load forecasting using dynamic high-order neural networks, Proc. IEEE-NTUA Joint Int. Power Conf., Athens, Greece, 1993, pp. 801–805.

[12] A. Bakirtzis, V. Petridis, S. Klartzis, A neural network short-term load forecasting model for the Greek power system, IEEE Trans. Power Syst. 11 (2) (1996) 858–863.

[13] A. Papalexopoulos, S. How, T.M. Peng, An implementation of a neural network based load forecasting model for the EMS, IEEE Trans. Power Syst. 9 (4) (1994) 1956–1962.

[14] Srinivasan, Dipti, Evolving artificial neural networks for short-term load forecasting, Neurocomputing 23 (1998) 265–276.

[15] G.N. Kariniotakis, G.S. Stavrakakis, E.F. Nogaret, Wind power forecasting using advanced neural networks models, IEEE Trans. Energy Convers. 11 (4) (1996) 762–767.

[16] M. El-Sharkawi, D. Niebur, Artificial neural networks with applications to power systems, IEEE Catalog No 96TP112-0, IEEE Service Center, NJ, 1996.

[17] I. Drezga, S. Rahman, Short-term load forecasting with local ANN predictors, IEEE Trans. Power Syst. 14 (1999) 3.

[18] A.G. Parlos, B. Fernandez, A. Atiya, J. Muthusami, W.K. Tsai, An accelerated learning algorithm for multilayer perceptron networks, IEEE Trans. Neural Networks 5 (3) (1994) 220–238.

[19] K. Swingler, Applying Neural Networks—a Practical Guide, Academic Press, London, 1996.
[20] C.-T. Lin, G.S.G. Lee, Neural Fuzzy Systems, Prentice Hall, NJ, 1996.
[21] H.A.B. Te Braake, G. Van Straten, Random activation weight neural net (RAWN) for fast non-iterative training, Eng. Appl. Artif. Intell. 8 (1) (1995) 71–80.
[22] G.S. Stavrakakis, A. Pouliezos, Fatigue life prediction using a new moving window regression method, Mech. Syst. Signal Processing 5 (4) (1991) 327–340.
[23] M.S. Kandil, S.M. El-Debeiky, N.E. Hasanien, Overview and comparison of long-term forecasting techniques for a fast developing utility: part I, Electric Power Syst. Res. 58 (2001) 11–17.
[24] B.-L. Zhang, Z.-Y. Dong, An adaptive neural-wavelet model for short term load forecasting, Electric Power Syst. Res. 59 (2001) 121–129.