

## Chapter 2

# Short Term Load Forecasting in Electric Power Systems with Artificial Neural Networks

G.J. Tsekouras, F.D. Kanellos, and N. Mastorakis

**Abstract** The demand in electric power should be predicted with the highest possible accuracy as it affects decisively many of power system's operations. Conventional methods for load forecasting were built on several assumptions, while they had to cope with relations between the data used that could not be described analytically. Artificial Neural Networks (ANNs) gave answers to many of the above problems and they became the predominant load forecasting technique. In this chapter the reader is first introduced to Artificial Neural Networks and their usage in forecasting the load demand of electric power systems. Several of the major training techniques are described with their pros and cons being discussed. Finally, feed- forward ANNs are used for the short-term forecasting of the Greek Power System load demand. Various ANNs with different inputs, outputs, numbers of hidden neurons etc. are examined, techniques for their optimization are proposed and the obtained results are discussed.

**Keywords** Artificial neural networks • ANN evaluation • Load Forecasting • Short term load forecasting • Training methods

## 2.1 Introduction

In a deregulated electricity market, the electric load has to be predicted with the highest possible accuracy for different time periods: very short-term (few minutes), short-term (few hours up to 1 week), midterm (few weeks up to few months) and long-term (few months up to years). Especially, the short-term load forecasting is very crucial as it affects decisively several power systems operations such as unit

---

G.J. Tsekouras (✉) • N. Mastorakis

Department of Electrical Engineering and Computer Science, Hellenic Naval Academy,  
Terma Hatzikiriaku, PIRAEUS, Athens 18539, Greece  
e-mail: tsekouras@snd.edu.gr; mastor@snd.edu.gr

F.D. Kanellos

School of Production Engineering and Management, Technical University of Crete,  
University Campus, Chania 73100, Greece  
e-mail: fkanellos@dpem.tuc.gr

commitment [1], spinning reserve scheduling [2], estimation of available transfer capability [3] and stability margins [3], load shedding decisions, etc. Consequently, the accurate load forecasting ensures higher reliability in power system operation while it facilitates the minimization of its operation cost by providing accurate input to day-ahead scheduling.

The efficiency of load forecasting is highly affected by the used input data. The most significant data used for short-term load forecasting are the hourly average values of the load for time periods extending from few past hours up to some weeks before the day the load is forecasted. The type of the day e.g. weekday, special day, weekend etc. plays also a key role in load forecasting accuracy. For example, load profile is different in weekdays and weekend, while load is more difficult to be forecasted in special days. Electric load can be also attributed with different behavior over the epochs of the year e.g. in Greece load demand is higher in summer due to the increased touristic activity and energy demand for air-conditioning. The above issues can be addressed with careful and methodical selection of the input data used for the load forecasting and the time period they are referred to.

Before applying a load forecasting model the load demand should be carefully decomposed in its components e.g. deterministic load, weather-dependent load component etc. Several load decomposition methods have been proposed so far in the literature. Namely, load decomposition:

- (a) in four components: Deterministic load, weather-independent load, weather-dependent load and noise component for the remaining load [4].
- (b) in three components: Yearly, seasonal and daily loads [5] that can be exploited in autoregressive models. Alternatively, the daily, weekly and cyclic components of the load can be used [6].
- (c) in two components: the basic and the weather-dependent load components [7].

Several methods have been used for short-term load forecasting with different levels of success, such as ARMAX models [8], regression [5], Artificial Neural Networks (ANNs) [9], fuzzy logic [10], expert systems etc. Among the variety of load forecasting methods ANNs have been proved the most effective [9]. In this chapter introduction to ANNs and their application to load forecasting is provided, special issues concerning the optimization of ANN training and validation are discussed, while results obtained from the exploitation of various types of ANNs for Greek power system load forecasting are presented and discussed.

## 2.2 Review of Short Term Load Forecasting Methods

A large number of modern load forecasting techniques are based on the exploitation of ANNs as there are no well-defined relations between the load and several factors affecting it, such as temperature, humidity, time, load values at previous hours etc. Despite the fact that ANNs were initially avoided [11, 12] as they did not help to understand the nature of the problem they finally dominated due to the same reason i.e. they allow to tackle extremely complex problems without fully understanding them.

Pioneers in this field were the members of the EPRI research team, Khotanzad et al., that developed various methodologies exploiting separate ANNs, for the prediction of time components of the load (weekly, daily, hourly) [13], for each prediction hours [14] and the load types (weather non-dependent and dependent load components) [15]. Papalexopoulos et al. introduced the concept of load seasonality through the use of sinusoidal functions of period equal to the number of days of the year [16].

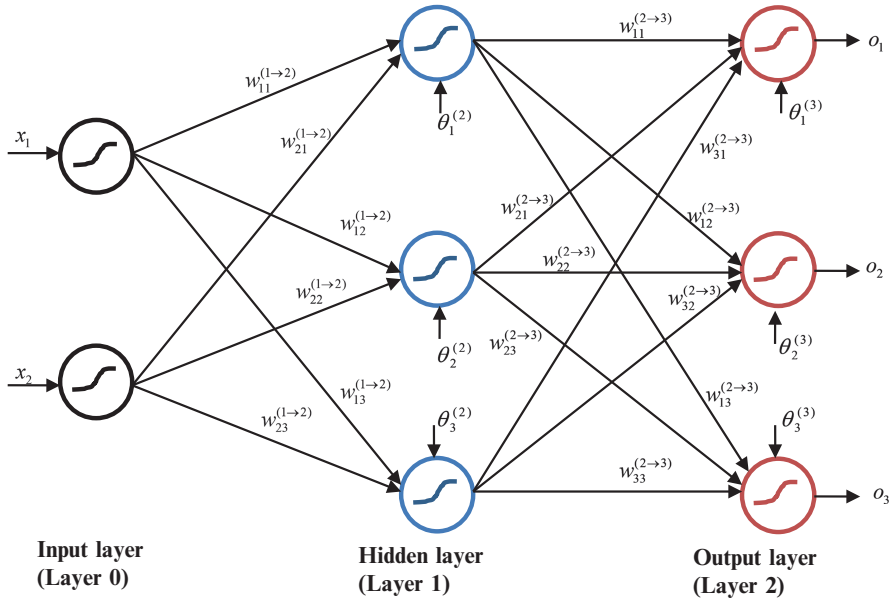
Rapid developments in load forecasting took place in the following period. More specifically, significant advances occurred in, the forecasting of load increase [17], learning techniques [18–21]. New network structures such as: radial basis function networks [22–24], recurrent networks [25], not fully connected networks [26], abductive networks [27], probabilistic networks [28, 29], and networks using similar days [30] were also explored. Moreover, several new techniques like fuzzy logic [31–39], wavelet decomposition [40], support vector machines [24, 41], genetic algorithms [42], harmony search [43] etc. were combined with ANNs to enhance their performance. Especially for Greece, pioneers in load forecasting were Bakirtzis et al., that proposed load forecasting techniques for interconnected power systems [44] as well as autonomous systems [45] followed by Kalaitzakis et al. [20] and Tsekouras et al. [46–48].

Hippert et al. [9], Choueiki et al. [49], and other researchers [50–54] have proved that the short-term load forecasting with classical multilayer artificial neural networks trained by error back propagation algorithm lead to mean absolute forecasting error ranging between 1.5 and 2.5 %. Usually, these techniques incorporate simple correction algorithms of data irregularities e.g. load behavior in holidays, measurement errors etc. However, the conclusions drawn can be hardly generalized as the influence of the inherent characteristics of the power systems is very strong and differs from system to system.

## 2.3 Multilayer Feed Forward Artificial Neural Networks

### 2.3.1 Introduction

A widely used type of multilayer artificial neural networks is the multilayer perceptron (MLP). The structure of a MLP is presented in Fig. 2.1. The neurons of the network are organized into three layers: the input, the hidden and the output layer. MLP is characterized as a feed forward network as the information flows only on the direction from the input to the output layer. According to Kolmogorov's theorem [55] an ANN can solve a problem by using one hidden layer provided that it comprises adequate number of neurons. Under these circumstances one hidden layer is used, but the number of its neurons should be properly selected. The number of the output layer neurons is equal to the number of the model output variables, while the input nodes correspond to the input variables of the model.



**Fig. 2.1** 2-3-3 (with 2 input, 3 hidden and 3 output neurons) feed forward artificial neural network

The increased computational capability of a MLP stems from the inherent nonlinear nature of its neurons, the complete interconnection between successive layers and its ability to learn after proper training. Research interest in MLPs first appeared by Rosenblatt in his work on the perceptron and also by Widrow who presented Madaline network in 1962. However, an efficient training algorithm for these networks was missing until 1985. Then, the error back-propagation algorithm was proposed for application to multilayer ANNs and it is still one of the most widely used ANN training methods. It should be noticed that error back propagation algorithm was first proposed by Werbos in his Ph.D. thesis in 1974. From 1985 until 1986 it was used by Rumelhart, Hinton, Williams, McClelland, Parker and LeCun, while it has been used since then in numerous applications [55, 56].

Next some basic information on ANN models is provided. The activation signal (input) of the  $k$ -th neuron of the  $\ell$ -th layer of an ANN is:

$$u_k^{(\ell)}(n) = \sum_{v=0}^{p_{\ell-1}} w_{kv}^{(\ell)}(n) y_v^{(\ell-1)}(n) \quad (2.1)$$

where  $w_{kv}^{(\ell)}$  is the interconnection weight between the  $k$ -th neuron of the  $\ell$ -th layer and the  $v$ -th neuron of the  $(\ell - 1)$ -th layer,  $p_{\ell-1}$  is the total number of the neurons of the  $(\ell - 1)$ -th layer and  $y_v^{(\ell-1)}$  is the output of the  $v$ -th neuron of the  $(\ell - 1)$ -th layer.

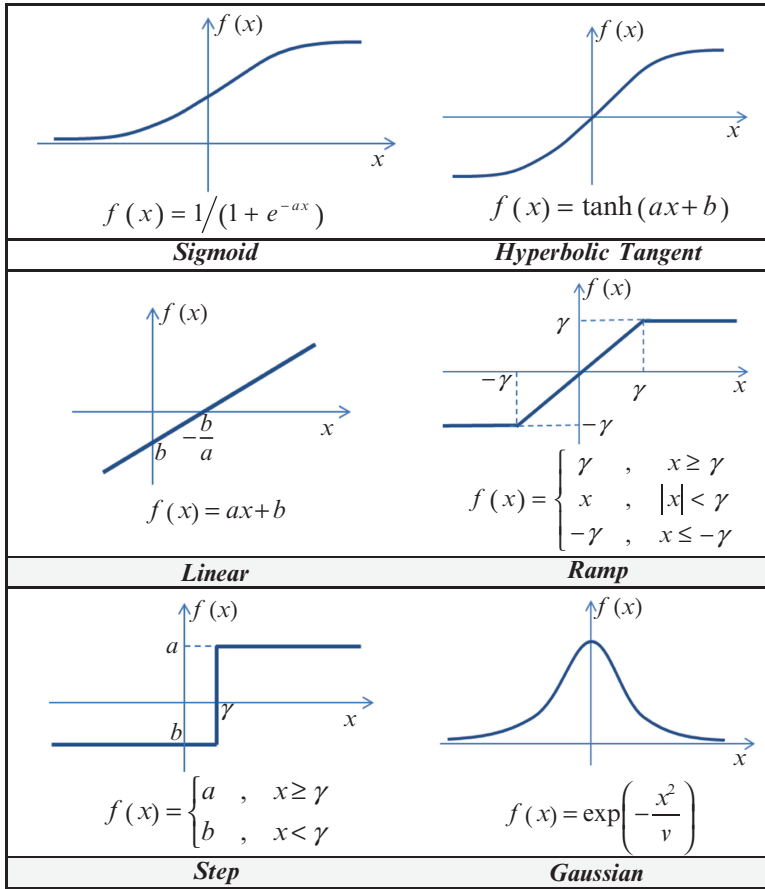


Fig. 2.2 Activation functions

The activation function of the neurons is denoted next with  $f$ . Hence, the output the  $k$ -th neuron of the  $\ell$ -th layer is:

$$y_k^{(\ell)}(n) = f\left(u_k^{(\ell)}(n)\right) \quad (2.2)$$

The most common activation functions are shown in Fig. 2.2.

Nonlinear activation functions are preferred in nonlinear problems. However, in that case saturation problems may occur. These problems can be attributed to the use of sigmoid activation functions which take values in a bounded area and are also highly nonlinear outside the region  $[-1, 1]$ . In order to avoid saturation problems the input and the output variables of the ANN are normalized as in the following relation:

$$\hat{x} = a + \frac{b - a}{x_{\max} - x_{\min}} (x - x_{\min}) \quad (2.3)$$

where  $\hat{x}$  is the normalized value of variable  $x$ ,  $x_{min}$  and  $x_{max}$  are the lower and the upper bounds of  $x$ , and  $a, b$  are the respective values of the normalized variable.

Alternatively, the input and output variables can be normalized by using their mean and standard deviation values as shown in Eq. (2.4).

$$\hat{x} = \frac{x - \mu}{\sigma} \quad (2.4)$$

However, the second method does not necessarily maintain the normalized variables outside the highly nonlinear region of the activation function.

### 2.3.2 Steepest Descent Error Back-Propagation Algorithm

Appropriate data sets are needed for the training, evaluation, validation and optimization of the network. Let us assume that  $m_1$  vectors are used for training,  $m_2$  for optimization-evaluation of ANN parameters and  $m_3$  for load forecasting purposes. The data set comprising the  $m_2$  vectors used for network evaluation and optimization may be a subset of the training data set.

The basic steps of the steepest descent error back-propagation training algorithm [55, 56] are as follows:

- (a) *Initialization*: Parameters like the number of the neurons in each layer, the training rate etc. are defined. Connection weights are initialized to small random values uniformly distributed in the interval  $[-0.1, 0.1]$ .
- (b) *Use of the training set*: In each training epoch all the training patterns are randomly used. For each input vector  $c$  and  $d$  steps are applied.
- (c) *Forward pass calculations*: The  $n$ -th training pattern is defined as  $\{\vec{x}_{in}(n), \vec{t}(n)\}$ , where  $\vec{x}_{in}(n)$  is the  $q_{in} \times 1$  input vector comprising the normalized input variables  $x_j$  and  $\vec{t}(n)$  the respective  $q_{out} \times 1$  normalized output vector. The activation signal of the  $k$ -th neuron of the  $\ell$ -th layer is:

$$u_k^{(\ell)}(n) = \sum_{v=0}^{p_{\ell-1}} w_{kv}^{(\ell)}(n) y_v^{(\ell-1)}(n) \quad (2.5)$$

where  $w_{kv}^{(\ell)}$  is the weight assigned to the connection of the  $k$ -th neuron of the  $\ell$ -th layer with the  $v$ -th neuron of the  $(\ell - 1)$ -th layer,  $p_{\ell-1}$  is the total number of neurons of the  $(\ell - 1)$ -th layer and  $y_v^{(\ell-1)}$  is the output of the  $v$ -th neuron of the  $(\ell - 1)$ -th layer. For  $v = 0$ , the bias is defined as  $\theta_k = w_{k0}$ , while  $y_0^{(\ell-1)} = -1$ . The activation function  $f$  at each layer can be the hyperbolic tangent, sigmoid or linear. The output of the neuron is:

$$y_k^{(\ell)}(n) = f \left( u_k^{(\ell)}(n) \right) \quad (2.6)$$

The output of the  $v$ -th neuron of the input layer is:

$$y_v^{(0)}(n) = x_v(n), \quad \forall v \quad (2.7)$$

where  $x_v(n)$  is the  $v$ -th element of network input vector  $\vec{x}_{in}(n)$ .

The output of the  $k$ -th neuron of the output layer (L) is:

$$y_k^{(L)}(n) = o_k(n), \quad \forall k \quad (2.8)$$

where  $o_k(n)$  is the  $k$ -th element of the output vector  $\vec{o}(n)$ , estimated by the ANN.

Hence, the error obtained at output of the  $k$ -th neuron of the output layer is:

$$e_k(n) = t_k(n) - o_k(n) \quad (2.9)$$

(d) *Reverse pass calculations*: The weights are updated by using the delta-rule:

$$w_{kv}^{(\ell)}(n+1) = w_{kv}^{(\ell)}(n) + \eta \cdot \delta_k^{(\ell)}(n) \cdot y_v^{(\ell-1)}(n) \quad (2.10)$$

where,  $\eta$  is the constant training rate and  $\delta_k^{(\ell)}(n)$  the local descent of the  $k$ -th neuron estimated for the output and hidden layers as following:

$$\delta_k^{(L)}(n) = e_k^{(L)}(n) \cdot f' \left( u_k^{(L)}(n) \right) \quad (2.11)$$

$$\delta_k^{(\ell)}(n) = f' \left( u_k^{(\ell)}(n) \right) \cdot \sum_i \delta_i^{(\ell+1)}(n) w_{ik}^{(\ell+1)}(n) \quad (2.12)$$

(e) *Stopping criteria*: Steps  $b$  up to  $d$  are repeatedly executed until the weights of neuron interconnections are stabilized or the output error function does not improve or the maximum number of epochs is exceeded.

Neurons' connection weights stabilization criterion is formulated as:

$$\left| w_{kv}^{(\ell)}(ep) - w_{kv}^{(\ell)}(ep-1) \right| < \text{limit}_1, \quad \forall k, v, \ell \quad (2.13)$$

where,  $\text{limit}_1$  is the upper limit of the absolute weight change and  $ep$  is the current epoch of training algorithm.

The output error function is the root mean square error  $RMSE_{va}$  estimated for the evaluation data set according to:

$$RMSE_{va} = \sqrt{\frac{1}{m_2 \cdot q_{out}} \sum_{m=1}^{m_2} \sum_{k=1}^{q_{out}} e_k^2(m)} \quad (2.14)$$

The respective stopping criterion is formulated as:

$$|RMSE_{va}(ep) - RMSE_{va}(ep - 1)| < \ellimit_2 \quad (2.15)$$

where  $\ellimit_2$  is the upper bound of the absolute change of RMSE in two successive epochs.

The maximum number of epochs exceedance criterion is formulated as:

$$ep \geq max\_epochs \quad (2.16)$$

If one of the above criteria becomes true, the main core of error back-propagation algorithm is ended. Otherwise the number of epochs is increased by one and the algorithm returns to step *b*. The above criterions are used on one hand to avoid data overfitting and on the other to enable the training algorithm to converge.

- (f) *Validation criteria:* The mean absolute percentage error (MAPE) is calculated for the evaluation data set as following:

$$MAPE_{va} = 100\% \cdot \sum_{i=1}^{m_2} |(t_k(i) - o_k(i)) / t_k(i)| / m_2 \quad (2.17)$$

In this training process each input vector is used randomly per epoch (stochastic training) to minimize the error function,  $G(n) = \frac{1}{2} \sum_{j=1}^{q_{out}} e_j^2(n)$ .

Alternatively, all input vectors can be used in a series during the forward process and afterwards the weights are updated minimizing the average output error function,  $G_{av} = \frac{1}{m_1} \sum_{n=1}^{m_1} G(n) = \frac{1}{2m_1} \sum_{n=1}^{m_1} \sum_{j=1}^{q_{out}} e_j^2(n)$ . This training process is called batch mode and the respective weights update term is calculated as:

$$\Delta \vec{w}(ep) = -\eta \cdot \nabla G(\vec{w}(ep)) \quad (2.18)$$

If a momentum term,  $a$ , is added then the respective equation becomes:

$$\Delta \vec{w}(ep) = -\eta \cdot \nabla G(\vec{w}(ep)) + a \cdot \Delta \vec{w}(ep - 1) \quad (2.19)$$

In steepest descent algorithm the parameters of learning rate and momentum are kept constant. Alternatively, decreasing exponential functions, as described in Eqs. (2.20) and (2.21) can be used:

$$\eta(ep) = \eta_0 \cdot \exp(-ep/T_\eta) \quad (2.20)$$

$$a(ep) = a_0 \cdot \exp(-ep/T_a) \quad (2.21)$$



where,  $T_n$ ,  $T_a$  are time parameters and  $n_0$ ,  $a_0$  the initial values of training rate and momentum term, respectively. Faster convergence can be achieved through the proper calibration of  $T_n$ ,  $n_0$ ,  $T_a$ ,  $a_0$  and it becomes even faster especially if the initial values  $n_0$ ,  $a_0$  are large.

### 2.3.3 Other Training Methods Based on Error Back-Propagation Algorithm

Variations of the error back-propagation algorithm are presented in the following paragraphs.

#### 2.3.3.1 Adaptive Error Back Propagation

In this method both the training rate and the momentum term are adaptively changed as described in Eqs. (2.22) and (2.23) in order to achieve rapid convergence.

$$\eta(ep) = \begin{cases} \eta(ep-1), & RMSE_{tr}(ep) > RMSE_{tr}(ep-1) \\ \eta(ep-1) \cdot \exp(-1/T_\eta), & RMSE_{tr}(ep) \leq RMSE_{tr}(ep-1) \end{cases} \quad (2.22)$$

$$a(ep) = \begin{cases} a(ep-1), & RMSE_{tr}(ep) \leq RMSE_{tr}(ep-1) \\ a(ep-1) \cdot \exp(-1/T_a), & RMSE_{tr}(ep) > RMSE_{tr}(ep-1) \end{cases} \quad (2.23)$$

where  $\eta_0 = \eta(0)$ ,  $a_0 = a(0)$  and  $RMSE_{tr}(ep)$  is the root mean square of the output error estimated for the training data set in training epoch,  $ep$ . If  $RMSE_{tr}(ep-1)$  is larger than  $RMSE_{tr}(ep)$ , which means that neurons' connection weights were updated in the correct direction, then training rate is decreased while the momentum term values remains the same in the next epoch. In this way, the previous successful update of the weights is rewarded. Otherwise, if  $RMSE_{tr}(ep) > RMSE_{tr}(ep-1)$  it is reasonable to reduce the momentum term and keep the learning rate constant as a penalty to the previous unsuccessful update of the weights. It should be noted that increasing the momentum term or the learning rate, as proposed in [57], may lead the training process to instability.

#### 2.3.3.2 Resilient Algorithm

In resilient algorithm only the sign of the derivative of the error function with respect  $w_{ij}$  is used for the estimation of the connection weight change direction. The weights are updated by using the following relations:

$$\Delta w_{ij}(ep) = \begin{cases} \delta_1 \cdot \Delta w_{ij}(ep-1), & \frac{\partial G_{av}}{\partial w_{ij}}(ep) \cdot \frac{\partial G_{av}}{\partial w_{ij}}(ep-1) > 0 \\ \Delta w_{ij}(ep-1), & \frac{\partial G_{av}}{\partial w_{ij}}(ep) \cdot \frac{\partial G_{av}}{\partial w_{ij}}(ep-1) = 0 \\ \frac{1}{\delta_2} \cdot \Delta w_{ij}(ep-1), & \frac{\partial G_{av}}{\partial w_{ij}}(ep) \cdot \frac{\partial G_{av}}{\partial w_{ij}}(ep-1) < 0 \end{cases} \quad (2.24)$$

where,  $\delta_1$ ,  $\delta_2$  are increasing and decreasing factors of the weight change over two successive epochs, respectively. If the derivative of the error function  $G$  with respect to the weight  $w_{ij}$  maintains the same sign during two sequential epochs, the weight change over epoch  $ep$  is the multiple of the respective change in epoch,  $ep-1$  (multiplied by  $\delta_1$ ). Respectively, if the sign of the derivative changes then the weight change is decreased else if the derivative is zero then the same weight change with the one of the previous epoch is applied.

### 2.3.3.3 Conjugate Gradient Algorithms

In conjugate gradient (CG) algorithm [21] a search is performed along conjugate directions leading generally to faster convergence than following the steepest descent direction. The basic steps of the CG algorithm (Fletcher-Reeves and Polak-Ribiere) are as follows:

- (a) In the first iteration the search direction  $\vec{p}_0$  is determined by the opposite of the output error function gradient:

$$\vec{p}_0 = -\nabla G(\vec{w})|_{\vec{w}=\vec{w}_0} \quad (2.25)$$

- (b) At  $k$ -th iteration the weights of the network are updated towards the search direction  $\vec{p}_k$  as following:

$$\Delta \vec{w}_k = a_k \cdot \vec{p}_k \quad (2.26)$$

The positive parameter  $a_k$  is calculated by numerical methods such as the golden section, bisection etc.

- (c) In the next iteration the search direction  $\vec{p}_{k+1}$  is calculated by:

$$\vec{p}_{k+1} = -\nabla G(\vec{w})|_{\vec{w}=\vec{w}_{k+1}} + \beta_{k+1} \cdot \vec{p}_k \quad (2.27)$$

$\beta_{k+1}$  is calculated either by the Fletcher-Reeves [58] or Polak-Ribiere equation [59] respectively:

$$\beta_{k+1} = \frac{\nabla G(\vec{w})|_{\vec{w}=\vec{w}_{k+1}}^T \cdot \nabla G(\vec{w})|_{\vec{w}=\vec{w}_{k+1}}}{\nabla G(\vec{w})|_{\vec{w}=\vec{w}_k}^T \cdot \nabla G(\vec{w})|_{\vec{w}=\vec{w}_k}} \quad (2.28)$$

$$\beta_{k+1} = \frac{\Delta \left( \nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_k}^T \cdot \nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_{k+1}} \right)}{\nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_k}^T \cdot \nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_k}} \quad (2.29)$$

- (d) If the algorithm has not converged then step (b) is repeated. It is mentioned that the  $k$ -th iteration usually coincides with the respective epoch, but this is not necessarily the case. The iterative process must be occasionally restarted in order to avoid a constant convergence rate. It is usual to restart it every  $N_w$  or  $(N_w + 1)$  iterations, where  $N_w$  is the number of the variables (weights and biases). Powell and Beale [60] proposed to restart the process of Polak-Ribiere algorithm occasionally or when the orthogonality between  $\vec{p}_k$  and  $\vec{p}_{k-1}$  is quite small:

$$\begin{aligned} & \left| \nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_k}^T \cdot \nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_{k+1}} \right| \\ & \geq \lim_{orthogonality} \cdot \left\| \nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_{k+1}} \right\|^2 \text{ with } k \geq 1 \end{aligned} \quad (2.30)$$

The limit,  $\lim_{orthogonality}$ , may take values within the interval (0.1, 0.9). Usually, it is set equal to 0.2.

The basic drawback of CG algorithm is the complexity of the evolved calculations per iteration as a linear search is performed to determine the appropriate step size. In scaled conjugate gradient algorithm (SCGA) the search process is avoided by using the Levenberg-Marquardt approach. The basic steps of SCGA are as follows [61]:

- (a)  $\vec{p}_0$  is initialized by Eq. (2.25) and the vector of the weights and biases  $\vec{w}_0$  is properly chosen. The rest parameters of the algorithm ( $\sigma$ ,  $\lambda_0$ ,  $\bar{\lambda}_0$ ,  $flag$ ) are set as following:

$$0 < \sigma \leq 10^{-4} \quad 0 < \lambda_0 \leq 10^{-6} \quad \bar{\lambda}_0 = 0 \quad flag = 1$$

- (b) If  $flag$  is 1 then the following additional information is calculated:

$$\sigma_k = \sigma / \left\| \vec{p}_k \right\| \quad (2.31a)$$

$$\vec{s}_k = \left( \nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_k + \sigma_k \cdot \vec{p}_k} - \nabla G(\vec{w}) \Big|_{\vec{w}=\vec{w}_k} \right) / \sigma_k \quad (2.31b)$$

$$\delta_k = \vec{p}_k^T \cdot \vec{s}_k \quad (2.31c)$$

- (c) The parameter  $\delta_k$  is scaled according to:

$$\delta_k = \delta_k + \left( \lambda_k - \bar{\lambda}_k \right) \cdot \left\| \vec{p}_k \right\|^2 \quad (2.32)$$

(d) If  $\delta_k \leq 0$ , then the Hessian matrix is made positive by setting:

$$\bar{\lambda}_k = 2 \left( \lambda_k - \delta_k / \left\| \vec{p}_k \right\|^2 \right) \quad (2.33a)$$

$$\delta_k = -\delta_k + \lambda_k \cdot \left\| \vec{p}_k \right\|^2 \quad (2.33b)$$

$$\lambda_k = \bar{\lambda}_k \quad (2.33c)$$

(e) The step size is calculated as:

$$\mu_k = -\vec{p}_k^T \cdot \nabla G(\vec{w}) \big|_{\vec{w}=\vec{w}_k} \quad (2.34a)$$

$$a_k = \mu_k / \delta_k \quad (2.34b)$$

(f) The comparison parameter  $\Delta_k$  is calculated as:

$$\Delta_k = 2 \cdot \delta_k \cdot \left( G(\vec{w}) \big|_{\vec{w}=\vec{w}_k} - G(\vec{w}) \big|_{\vec{w}=\vec{w}_k + a_k \cdot \vec{p}_k} \right) / \mu_k^2 \quad (2.35)$$

(g) If  $\Delta_k \geq 0$  then a successful reduction in error can be achieved by applying the following equations:

$$\Delta \vec{w}_k = a_k \cdot \vec{p}_k \quad (2.36a)$$

$$\vec{r}_{k+1} = -\nabla G(\vec{w}) \big|_{\vec{w}=\vec{w}_{k+1}} \quad (2.36b)$$

$$\bar{\lambda}_k = 0 \quad (2.36c)$$

$$flag = 1 \quad (2.36d)$$

(h) If the number of iterations is multiple of the population of the weights and biases,  $N_w$ , then the algorithm is restarted:

$$\vec{p}_{k+1} = -\nabla G(\vec{w}) \big|_{\vec{w}=\vec{w}_{k+1}} \quad (2.37)$$

else:

$$\beta_{k+1} = \left( \left\| \nabla G(\vec{w}) \big|_{\vec{w}=\vec{w}_{k+1}} \right\|^2 - \nabla G(\vec{w}) \big|_{\vec{w}=\vec{w}_{k+1}}^T \cdot \nabla G(\vec{w}) \big|_{\vec{w}=\vec{w}_k} \right) / \mu_k \quad (2.38)$$

$$\vec{p}_{k+1} = -\nabla G(\vec{w}) \big|_{\vec{w}=\vec{w}_{k+1}} + \beta_{k+1} \cdot \vec{p}_k \quad (2.39)$$

If  $\Delta_k \geq 0.75$ , then  $\lambda_k = 0.25 \cdot \lambda_k$ , else  $\bar{\lambda}_k = \lambda_k$ ,  $flag = 0$ .

If  $\Delta_k < 0.25$ , then  $\lambda_k = \lambda_k + \delta_k (1 - \Delta_k) / \left\| \vec{p}_k \right\|^2$ .

- (i) If  $\nabla G(\vec{w})|_{\vec{w}=\vec{w}_{k+1}} \neq \vec{0}$ , then  $k = k + 1$  and the step (b) is repeated else the training process is completed.

The basic drawback of the SCGA algorithm is the increased complexity of the calculations within an iteration that is in the order of  $O(6N_w^2)$  instead of  $O(3N_w^2)$  of the basic steepest descent method. If the scale parameter  $\lambda_k$  is zero, then the SCGA coincides with the CGA. SCGA's basic advantage is that the error decreases monotonically as error increase is not allowed. If the error is constant for one or two iterations then the Hessian matrix has not been positive definite and  $\lambda_k$  has been increased. It is also recommended that the value of parameter  $\sigma$  should be as small as possible in order to constrain its effect on the performance of the algorithm.

### 2.3.3.4 Newton Algorithm

In *Newton* method the inverse of the Hessian matrix,  $\nabla^2 G(\vec{w})$ , is used to update the connection weights and biases of the network as follows:

$$\Delta \vec{w}_k = -\nabla^2 G(\vec{w})|_{\vec{w}=\vec{w}_k}^{-1} \cdot \nabla G(\vec{w})|_{\vec{w}=\vec{w}_k} \quad (2.40)$$

Usually, the convergence of this algorithm is more rapid than the aforementioned algorithms if the size of the problem is small. The calculation and the inversion of Hessian matrix are complex and computationally demanding processes. Hessian and Jacob matrices are estimated according to the following equations:

Hessian matrix:

$$\nabla^2 G(\vec{w}) = J(\vec{w})^T \cdot J(\vec{w}) + \sum_{j=1}^{m_1} e_j(\vec{w}) \cdot \nabla^2 e_j(\vec{w}) \quad (2.41)$$

Jacob matrix:

$$J(\vec{w}) = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \dots & \frac{\partial e_1}{\partial w_{N_w}} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \dots & \frac{\partial e_2}{\partial w_{N_w}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{m_1}}{\partial w_1} & \frac{\partial e_{m_1}}{\partial w_2} & \dots & \frac{\partial e_{m_1}}{\partial w_{N_w}} \end{bmatrix}_{m_1 \times N_w} \quad (2.42)$$

The *quasi-Newton* method belongs to this family of Newton's algorithms. In this method the second term of Hessian matrix is omitted as it usually takes small values that are not significant.

Alternatively, in the *one step secant* algorithm only the diagonal elements of Hessian matrix are used to simplify its inversion. It needs more iterations than the basic Newton method but the required computational effort is significantly compressed.

The *Levenberg-Marquardt* method [62, 63] is another commonly used variation of Newton algorithm. The weights and the biases of the network are updated as follows:

$$\begin{aligned}\Delta \vec{w}_k &= -(J^T \cdot J + \lambda \cdot \text{diag}[J^T \cdot J])^{-1} \cdot \nabla G(\vec{w})|_{\vec{w}=\vec{w}_k} \\ &= -(J^T \cdot J + \lambda \cdot \text{diag}[J^T \cdot J])^{-1} \cdot J^T \cdot \vec{e}(\vec{w}_k)\end{aligned}\quad (2.43)$$

Factor  $\lambda$  is updated with respect to the change of the output error function as following:

$$\lambda(k+1) = \begin{cases} \lambda(k) \cdot \beta, & G_{av}(k) > G_{av}(k-1) \\ \lambda(k), & G_{av}(k) = G_{av}(k-1) \\ \lambda(k)/\beta, & G_{av}(k) < G_{av}(k-1) \end{cases} \quad (2.44)$$

The parameter  $\beta$  usually takes values near 10. The *Levenberg-Marquardt* algorithm is not the optimal one but it performs extremely well if the number of the weights and biases is smaller than few thousands.

### 2.3.4 Bias-Variance Dilemma

One significant issue closely related with the efficiency of the ANNs is the selection of the number of the neurons at hidden layer. Using many neurons leads the network to memorize the training data and degradation of its generalization properties occurs i.e. the ANN does not perform satisfactorily on data not belonging to the training set. On the contrary, small number of neurons reduces the capability of the ANN to identify complex relations between the input data while generalization capability may be satisfactory.

Therefore, it is crucial to determine the optimal ANN structure that leads to accurate results and ensures at the same time satisfactory generalization properties. This is known as the bias-variance dilemma where the generalization error is expressed by the terms of bias and variance. Simple models are characterized by increased bias while complex models comprising large number of parameters lead to increased variance. According to bias-variance dilemma, if the bias increases then variance decreases and vice versa. Consequently, the best model is the one obtained by the optimal proportion of the two terms.

There are two major approaches to achieve balance between bias and variance. In the first, known as the “structural” approach, the number of the neurons of the

hidden layer is gradually increased until the optimal generalization is achieved. Generalization becomes optimal when MAPE of the test data set starts to increase and overtraining occurs.

The second approach is based on the idea of network parameters regularization. The simplest way to achieve regularization is the introduction of a penalty term at the output error function of the network. This term prevents the parameters (weights and biases) from taking large values. In this way, parameters of small significance are nullified and the number of the parameters of the network eventually decreases. The sum of the squared values of the network parameters is often used as the regularization term. Assuming that  $p_i$  are the parameters of the network and  $G$  the output error function of the network then the following function is minimized during the training process:

$$G_f = G + \lambda \cdot \sum_i p_i^2 \quad (2.45)$$

where,  $\lambda$  is the parameter that determines the significance of the two minimization goals that are the minimization of the output error function and the reduction of network parameters values.

### 2.3.4.1 Confidence Interval Estimation

In case of ANNs the confidence interval is not directly estimated unlike to the classical forecasting models. Three techniques have been proposed so far [64]:

- (a) *error output*: According to this technique the ANN model uses two outputs for each output variable, the first one is the forecasted mean value and the second one the respective absolute percentage error. After the training process a larger confidence interval is determined by multiplying the initial one by a proper factor in order to reach to the required confidence level.
- (b) *re-sampling*: The prediction and the respective error are calculated for each set and for all available  $m$  input vectors and are sorted in ascending order. The cumulative distribution function of the prediction errors can be estimated by:

$$S_m(z) = \begin{cases} 0, & z < z_1 \\ r/m, & z_r \leq z < z_{r+1} \\ 1, & z_m \leq z \end{cases} \quad (2.46)$$

When  $m$  is large enough,  $S_m(z)$  is a good approximation of the true cumulative probability distribution  $F(z)$ . The confidence interval is estimated by keeping the intermediate value  $z_r$  and discarding the extreme values, according to the desired confidence degree. The obtained intervals are equal in probability (not necessarily symmetric in  $z$ ). The number of cases to discard in each tail of the prediction error distribution is  $n \cdot p$ , where  $p$  is the probability in each tail. If  $n \cdot p$

is not an integer then  $\lfloor n \cdot p \rfloor$  cases are discarded in each tail. If the cumulative probability distribution  $F(Z_p)$  equals to  $p$ , then an error is less than or equal to  $Z_p$  with a  $p$  probability which indicates that  $Z_p$  is the lower confidence limit. Consequently,  $Z_{1-p}$  is the upper limit and there is a  $(1 - 2p)$  confidence interval for future errors.

- (c) *multi-linear regression model adapted to ANN*: This technique can be applied only if linear activation functions are used at the output layer. In this technique, a multi-linear regression model can be implemented for each neuron of the output layer. The inputs of the regression model are the outputs of the hidden neurons, while regression coefficients are their connection weights with the output neuron. The computation of the confidence interval is accomplished through the estimation of the prediction error variance:

$$\sigma^2 = \sum_{i=1}^{m_1} (t_i - o_i)^2 / (m_1 - p_c) \quad (2.47)$$

where,  $p_c$  is the number of the regression model coefficients. The confidence interval at a prediction point  $\tau$  can be computed using the prediction error variance estimated in Eq. (2.47), the ANN inputs and the desired confidence degree that follows  $t$ -Student's distribution with  $(m_1 - p_c)$  degrees of freedom [64].

Re-sampling technique is usually preferred to the other two techniques as the doubling of the ANN's outputs and the use of linear activation function at the output layer are voided. Finally, Silva et al. [65] propose the re-sampling technique as the most suitable for the estimation of the confidence interval with a high degree of confidence.

It is noted that the second method has been the most applied [65] with some empirical enhancements in some cases [66] while other similar probabilistic methods have been recently proposed [67, 68].

### 2.3.5 Compression of the Input Data

Usually, a large number of inputs is used in short-term load forecasting models. Compression techniques, such as the principal component analysis (PCA), can be applied [21, 69] to suppress the inputs of the model. Let us assume that  $X$  is a  $N \times p_{in}$  matrix whose rows contain the input vectors while the  $p_{in}$  columns of  $X$  represent the properly transformed input variables so as their means equal to zero. Also, vector  $\vec{a}$  is defined as the vector that maximizes variance of  $X$  projection on it ( $X \cdot \vec{a}$ ). The variance of  $\vec{a}$  is defined as:

$$\sigma_a^2 = \left( X \cdot \vec{a} \right)^T \cdot \left( X \cdot \vec{a} \right) = \vec{a}^T \cdot X^T \cdot X \cdot \vec{a} = \vec{a}^T \cdot V \cdot \vec{a}$$

where  $V = X^T \cdot X$  is the covariance matrix of  $X$ .



$\sigma_a^2$  the variance is a function of both  $\vec{a}$  and  $X$  and it must be maximized. In order to avoid unconstrained increase of  $\vec{a}$  it is normalized according to the constraint,  $\vec{a}^T \cdot \vec{a} = 1$ . In this case the optimization problem is transformed to the maximization of the quantity  $f_{opt} = \vec{a}^T \cdot V \cdot \vec{a} - \lambda \cdot (\vec{a}^T \cdot \vec{a} - 1)$ , where  $\lambda$  is a Lagrange multiplier. By setting the derivative of  $f_{opt}$  with respect to  $\vec{a}$  equal to zero the maximization problem is finally reduced to the estimation of the eigenvalues of the covariance matrix  $V$  as follows:

$$\frac{\partial f_{opt}}{\partial \vec{a}} = 2 \cdot V \cdot \vec{a} - 2 \cdot \lambda \cdot \vec{a} = 0 \Rightarrow (V - \lambda \cdot I) \cdot \vec{a} = 0$$

If the eigenvalues of  $V$  are ranked in descending order, the first principal component  $\vec{a}_1$  is the eigenvector associated with the largest eigenvalue  $\lambda_1$  of the covariance matrix  $V$ , the second principal component  $\vec{a}_2$  is the eigenvector associated with the second largest eigenvalue  $\lambda_2$  etc. The obtained eigenvectors are orthogonal each other because the  $V$  matrix is real and symmetric.

A basic property of this method is that if the data are projected to the first  $k$  eigenvectors then the variance of the projected data can be expressed as the sum of the eigenvalues,  $\sum_{j=1}^k \lambda_j$ . Equivalently, the squared error in matrix  $X$  approximation

by using only  $k$  eigenvectors can be expressed as  $\sum_{j=k+1}^{p_{in}} \lambda_j / \sum_{j=1}^{p_{in}} \lambda_j$ . Increasing the number,  $k$ , of the principal components the respective mean square error  $J_k =$

$$\sum_{m=1}^N \left\| \sum_{j=1}^k \lambda_j \cdot \vec{a}_j - \vec{x}_m \right\|^2 \text{ decreases, where } \vec{x} \text{ are the column-vectors of } X^T.$$

In case of multidimensional data sets with strongly correlated elements, 5 up to 10 principal elements are necessary to achieve at least a 90 % accumulated percentage of explained variance.

## 2.4 Short Term Forecasting of the Greek Power System Load Demand by Using ANNs

In this paragraph feed forward multilayer ANNs are used for load forecasting purposes in Greek electric power system. Different configurations of ANNs and issues related with their optimization are examined.

### 2.4.1 Basic ANN Configuration

The basic ANN configuration used in the following analysis comprises 71 input variables [44, 46]. Assuming that the hourly average values of the load of the  $d$ -th day of the year are to be forecasted the input and output variables of the basic ANN model are grouped as in Table 2.1.

It is noted that according to Kolmogorov's theorem [56] ANNs comprising one hidden layer with adequate number of neurons can describe any nonlinear system. Hence, ANNs with one hidden layer are used and the optimal number of the neurons of the hidden layer should be found. The basic ANN configuration is shown in Fig. 2.3.

In the next paragraphs several case studies are presented concerning the use of different training methods, inputs/outputs of the ANN, confidence interval estimation etc. In each of the examined cases the general heuristic methodology shown in Fig. 2.4 is used to determine the optimal parameters of the ANN model.

A brief description of the basic steps of the methodology of Fig. 2.4 is provided next.

- (a) *Data selection*: The input and output variables of the forecasting model are selected. The inputs and the outputs of the basic ANN configuration are as described Table 2.1 while scenarios concerning the use of different inputs and outputs will be examined in the next paragraphs.
- (b) *Data pre-processing*: Data normality is examined and outliers are modified or extracted from the data (noise suppression). Next, input and output variables are normalized according to Eq. (2.3) in order to avoid saturation.
- (c) *Main procedure*: A large number of model parameters combinations are used in order to optimize the performance of the ANN model. The obtained forecasting models are tested and evaluated. Mean absolute percentage error (*MAPE*) is used for evaluation purposes.

*MAPE* is estimated by:

$$MAPE_{ev} = 100\% \cdot \frac{1}{m_{ev}} \cdot \sum_{d=1}^{m_{ev}} \sum_{i=1}^{24} \frac{\left| \widehat{L}(d, i) - L(d, i) \right|}{L(d, i)} \quad (2.48)$$

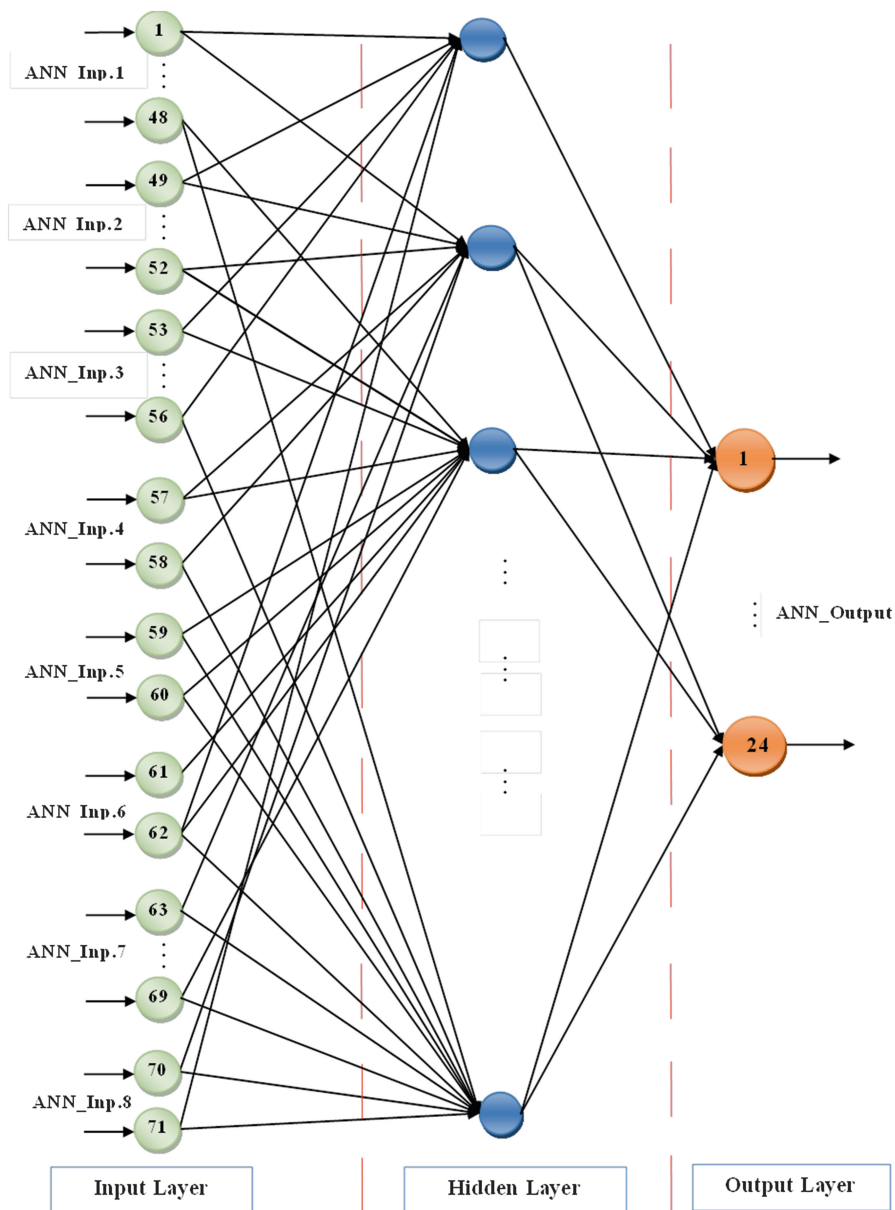
Where,  $L(d, i)$  is the measured value of load demand at the  $i$ -th hour of  $d$ -th day of the evaluation set,  $\widehat{L}(d, i)$  the respective forecasted load and  $m_{ev}$  is the number of the data vectors of the evaluation set.

- (d) *ANN test*: The load demand is finally estimated for the days of the test set by using the optimal values of the parameters that lead to the lowest *MAPE*.

Next, the above optimization method is applied to the basic ANN configuration of Fig. 2.3. Several parameters of the ANN model and the training method are optimized and they are listed next:

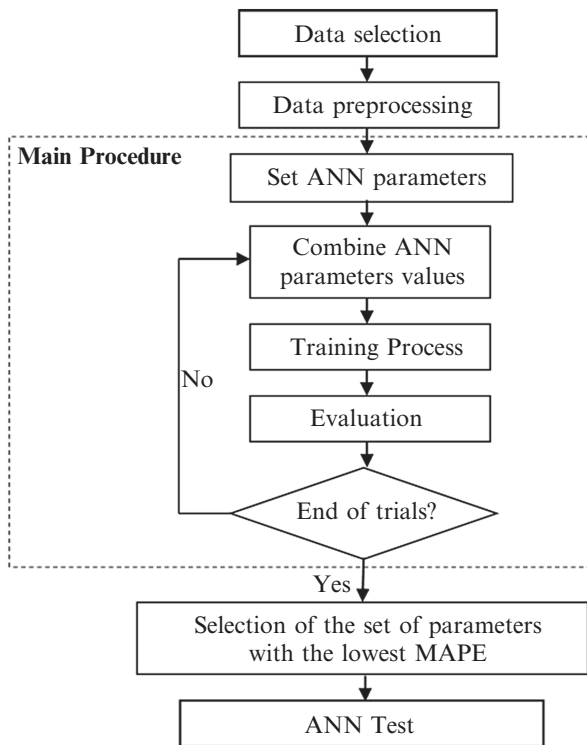
Table 2.1 Inputs and outputs of the basic ANN configuration

<b>ANN_Inp. 1</b>	The hourly load values of the two previous days: $L(d-1, 1), \dots, L(d-1, 24), L(d-2, 1), \dots, L(d-2, 24)$ (in MW),
<b>ANN_Inp. 2</b>	The maximum value of the 3-h average temperature measurements in Athens for the current day (d) and the previous one (d-1): $\max\_Temp_{Ath}(d), \min\_Temp_{Ath}(d), \max\_Temp_{Ath}(d-1), \min\_Temp_{Ath}(d-1)$ , respectively ( $^{\circ}\text{C}$ ),
<b>ANN_Inp. 3</b>	The maximum value of the 3-h average temperature measurements in Thessalonica for the current day and the previous one (d-1): $\max\_Temp_{Th}(d), \min\_Temp_{Th}(d), \max\_Temp_{Th}(d-1), \min\_Temp_{Th}(d-1)$ , respectively ( $^{\circ}\text{C}$ ),
<b>ANN_Inp. 4</b>	The difference between the maximum value of the 3-h average temperature measurements of the current day (d) and the previous one (d-1) for Athens and Thessalonica: $\Delta Temp_{Ath} = \max\_Temp_{Ath}(d) - \max\_Temp_{Ath}(d-1)$ $\Delta Temp_{Th} = \max\_Temp_{Th}(d) - \max\_Temp_{Th}(d-1)$
<b>ANN_Inp. 5</b>	The temperature dispersion from the temperature of the comfortable living conditions for Athens, for the current and the previous day $T_{Ath}^2(d), T_{Ath}^2(d-1)$ , respectively. Temperature dispersion is calculated as: $T_{dispersion}^2 = \begin{cases} (T_c - T)^2, & T < T_c \\ 0, & T_c < T < T_h \\ (T - T_h)^2, & T_h < T \end{cases}$ <p>Where, <math>T_c = 18^{\circ}\text{C}</math>, <math>T_h = 25^{\circ}\text{C}</math></p>
<b>ANN_Inp. 6</b>	The temperature dispersion from comfortable living conditions temperature for Thessalonica for the current and the previous day $T_{Th}^2(d), T_{Th}^2(d-1)$ , respectively,
<b>ANN_Inp. 7</b>	Seven digit binary numbers for the coding of the weekdays, e.g. Monday is coded with 1000000, Tuesday with 0100000, etc
<b>ANN_Inp. 8</b>	Two sinusoidal functions ( $\cos(2\pi d/T)$ , $\sin(2\pi d/T)$ ), which express the seasonal behavior of the current day. Where, $T$ is the total number of the days of the year
<b>ANN_Output</b>	The output variables are the 24 hourly average values of the load demand of the current day, $\widehat{L}(d, 1), \dots, \widehat{L}(d, 24)$



**Fig. 2.3** Basic ANN configuration

- the number of the neurons of the hidden layer ranging from 20 up to 70.
- the initial value of  $n$ ,  $n_0$ , and the time parameter  $T_n$  of the training rate, which get values from the sets  $\{0.1, 0.2, \dots, 0.9\}$  and  $\{1,000, 1,200, \dots, 2,000\}$ , respectively.



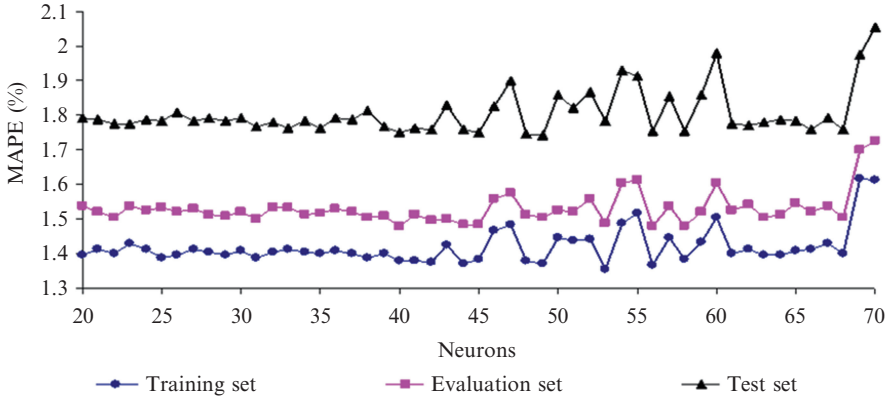
**Fig. 2.4** Flowchart of the ANN optimization and the evaluation process

- the initial value of parameter  $a$  and the time parameter  $T_a$  of the momentum term, which get values from the sets  $\{0.1, 0.2, \dots, 0.9\}$  and  $\{1,000, 1,200, \dots, 2,000\}$ , respectively,
- the type and the parameters of the activation functions used at the hidden and output layers. The type of the activation functions can be the *hyperbolic tangent*, *linear* or *logistic*, while their parameters  $a_1$ ,  $a_2$  get values from the set  $\{0.1, 0.2, \dots, 0.5\}$  and parameters  $b_1$ ,  $b_2$  from the set  $\{0.0, \pm 0.1, \pm 0.2\}$ .

Stopping criteria are defined after a few trials as  $\max\_epochs = 10,000$ ,  $limit_1 = 10^{-5}$ ,  $limit_2 = 10^{-5}$ .

In this study the combinations resulting from the above assumptions account to 836,527,500 and they cannot be practically examined. To this end, calibration process through successive variations of the parameters values is applied in order to determine the optimal or nearly optimal values of the parameters.

First, the number of the hidden neurons is varied from 20 up to 70, while the remaining parameters are assigned with fixed values ( $a_0 = 0.4$ ,  $T_a = 1,800$ ,  $\eta_0 = 0.5$ ,  $T_\eta = 2,000$ , the type of activation functions at hidden and output layers is hyperbolic tangent with  $a_1 = a_2 = 0.25$ ,  $b_1 = b_2 = 0.0$ ). *MAPE* of the training, evaluation and



**Fig. 2.5** *MAPE (%)* of all sets versus the number of hidden neurons. ( $a_0 = 0.4$ ,  $T_a = 1,800$ ,  $\eta_0 = 0.5$ ,  $T_\eta = 2,000$ , activation functions in both layers: hyperbolic tangent,  $a_1 = a_2 = 0.25$ ,  $b_1 = b_2 = 0.0$ )

test sets versus the number of the hidden neurons are presented in Fig. 2.5. The *MAPE* of the evaluation and tests set keep step with the respective one of the training set and the following relationship is valid within the entire examined range of hidden neurons number:

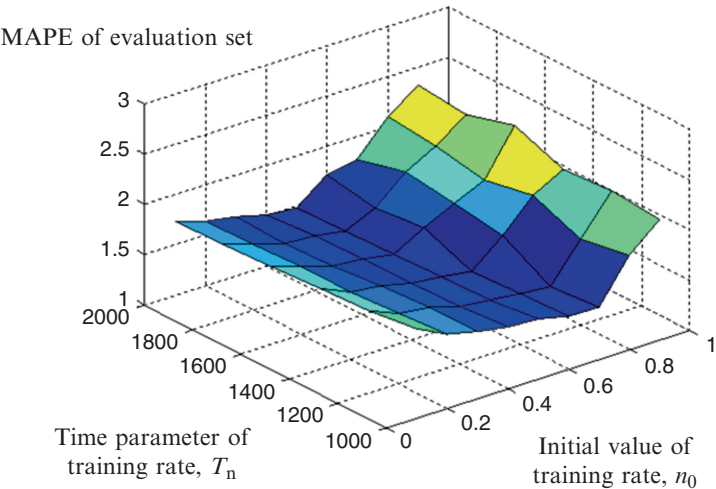
$$MAPE_{training} < MAPE_{evaluation} < MAPE_{test} \quad (2.49)$$

*MAPE* of the evaluation set is minimized for 45 neurons at hidden layer while it rapidly increases when the number of the hidden neurons increases.

Next, the initial value  $\eta_0$  and the time parameter  $T_\eta$  of the training rate are varied while the other parameters remain constant (hidden neurons = 45,  $a_0 = 0.4$ ,  $T_a = 1,800$ , hyperbolic tangent activation functions are used at hidden and output layers with  $a_1 = a_2 = 0.25$ ,  $b_1 = b_2 = 0.0$ ). It appears in Fig. 2.6 that the results obtained for the *MAPE* of the evaluation set are satisfactory for  $0.5 \leq \eta_0 \leq 0.8$  and  $1,000 \leq T_\eta \leq 1,400$ . The lowest *MAPE* is obtained for  $\eta_0 = 0.5$ ,  $T_\eta = 2,000$ . It is mentioned that *MAPE* increases dramatically for  $\eta_0 \geq 0.7$  and  $T_\eta \geq 1,600$ .

In the next step of the calibration process, the initial value  $a_0$  and the time parameter  $T_a$  of the momentum term are simultaneously varied, while the rest of the parameters are kept constant (hidden neurons = 45,  $\eta_0 = 0.5$ ,  $T_\eta = 2,000$ , activation functions at hidden and output layers: hyperbolic tangent with  $a_1 = a_2 = 0.25$ ,  $b_1 = b_2 = 0.0$ ). In this case, the obtained *MAPE* of the evaluation set is satisfactory for  $a_0 \geq 0.6$  and  $T_a \geq 1,600$ , while the lowest *MAPE* is obtained for  $a_0 = 0.9$ ,  $T_a = 2,000$ . It is mentioned that *MAPE* increases dramatically for  $a_0 \leq 0.5$ .

Similarly, it is found that the ANN gives better results if the hyperbolic tangent activation function with parameters  $a_1 = a_2 = 0.25$  and  $b_1 = b_2 = 0.0$ , is used at both layers. The results obtained by using different activation functions are registered in Table 2.2.



**Fig. 2.6** *MAPE* (%) of the evaluation set versus parameters  $T_n$  and  $n_0$ . ( $\eta_0 = \{0.1, 0.2, \dots, 0.9\}$ ,  $T_\eta = \{1,000, 1,200, \dots, 2,000\}$ , hidden neurons: 45,  $a_0 = 0.4$ ,  $T_a = 1,800$ , activation function used in both layers: hyperbolic tangent,  $a_1 = a_2 = 0.25$ ,  $b_1 = b_2 = 0.0$ )

**Table 2.2** *MAPE* (%) of (A) training set, (B) evaluation set, (C) test set for different activation functions

Activation function of output layer	Activation function of hidden layer								
	Hyperbolic sigmoid			Hyperbolic tangent			Linear		
	(A)	(B)	(C)	(A)	(B)	(C)	(A)	(B)	(C)
Hyperbolic sigmoid	2.030	2.048	2.625	1.510	1.621	1.817	1.788	1.850	2.091
Hyperbolic tangent	1.671	1.737	2.042	1.383	1.482	1.749	1.900	1.987	2.200
Linear	1.603	1.691	1.903	1.390	1.522	1.747	1.936	2.023	2.194

Number of hidden neurons: 45,  $a_0 = 0.4$ ,  $T_a = 1,800$ ,  $\eta_0 = 0.5$ ,  $T_\eta = 2,000$ ,  $a_1 = a_2 = 0.25$ ,  $b_1 = b_2 = 0.0$

The above described process finally leads to the following optimal intervals of the parameters values: 40 up to 50 neurons at the hidden layer,  $a_0 = 0.8 - 0.9$ ,  $T_a = 1,800-2,000-2,200$ ,  $\eta_0 = 0.5 - 0.6$ ,  $T_\eta = 1,000-1,200-1,400$ , activation functions of both layers: hyperbolic tangent with  $a_1 = a_2 = 0.20-0.25-0.30$ ,  $b_1 = b_2 = 0$ .

The minimum *MAPE* of the evaluation set is 1.48 % and it is obtained for an ANN with 45 neurons in the hidden layer,  $a_0 = 0.9$ ,  $T_a = 2,000$ ,  $\eta_0 = 0.5$ ,  $T_\eta = 2,000$ ,  $a_1 = a_2 = 0.25$  and  $b_1 = b_2 = 0$  with hyperbolic tangent activation functions in hidden and output layer.

### 2.4.2 Comparison of Different Training Algorithms

In this case study the basic ANN structure presented in paragraph 2.4.1 is used while parameters, such as the number of neurons of the hidden layer, activation functions, weighting factors, learning rate, momentum term, etc. are determined by a calibration methodology through an extensive search. The performance of each set of parameters is evaluated by the MAPE index of the evaluation data set. The ANN parameter calibration process is based on the philosophy of the heuristic process outlined in Fig. 2.4 and it is repeated for 14 different training algorithms. The main goals are:

- the determination of the optimal ANN structure (number of neurons in the hidden layer, learning rate initial value, etc.) for each of the examined training algorithms,
- the comparison of the training algorithms in terms of MAPE and computation time minimization
- the selection of the training algorithm with the best performance.

The examined training algorithms are registered in Table 2.3. Several parameters of the ANN model should be optimized for each of the case studies included in Table 2.3. The common parameters in all cases are:

- the number of the hidden neurons,  $N_n$ ,
- the type of the activation functions (hyperbolic tangent, logistic, linear),

**Table 2.3** Examined training algorithms

No.	Training algorithms
1	Stochastic training with learning rate and momentum term (decreasing exponential functions)
2	Stochastic training, use of adaptive rules for the learning rate and the momentum term
3	Stochastic training, constant learning rate
4	Batch mode, constant learning rate
5	Batch mode with learning rate and momentum term (decreasing exponential functions)
6	Batch mode, use of adaptive rules for the learning rate and the momentum term
7	Batch mode, conjugate gradient algorithm with Fletcher-Reeves equation
8	Batch mode, conjugate gradient algorithm with Fletcher-Reeves equation and Powell-Beale restart
9	Batch mode, conjugate gradient algorithm with Polak-Ribiere equation
10	Batch mode, conjugate gradient algorithm with Polak-Ribiere equation and Powell-Beale restart
11	Batch mode, scaled conjugate gradient algorithm
12	Batch mode, resilient algorithm
13	Batch mode, quasi-Newton algorithm
14	Batch mode, Levenberg-Marquardt algorithm



- the parameters of the activation functions,
- the maximum number of the training epochs.

The additional parameters used in the each of the examined training methods are listed next.

*Methods 1–6:*

- the time parameter,  $T_n$ , and the initial value of the learning rate,  $n_0$ ,
- the time parameter,  $T_\alpha$ , and the initial value of the momentum term,  $\alpha_0$  (not applied to methods 3, 4).

*Methods 7–10:*

- the initial value of  $s$ ,
- the number of iterations,  $T_{bn}$ , for the calculation of the basic vector,
- the number of iterations,  $T_{trix}$ , used for the trisection according to the golden section method for output error minimization.

*Method 11:*

- Parameters  $\sigma$  and  $\lambda_0$ ,

*Method 12:*

- the increasing and decreasing factors  $\delta_1$  and  $\delta_2$  (see Eq. (2.24))

*Method 14:*

- the initial value of  $\lambda$ ,  $\lambda(0)$ , and the multiplicative parameter  $\beta$ .

The number of all possible combinations of the values of the parameters that should be examined ranges in the order of some hundreds of millions. Hence, all possible combinations cannot be practically examined and a gradual calibration process is applied. The intervals of the parameters values used in this study are registered in Table 2.4 while the calibrated parameters obtained for the 14 examined scenarios of Table 2.3 are registered in Table 2.6.

The minimum MAPE of the evaluation data set is obtained by the stochastic training algorithm with adaptive rules for the learning rate and the momentum term and the scaled conjugate gradient algorithm. The MAPE of the test data set is also satisfying while the stochastic training algorithm with decreasing learning rate and momentum term leads to slightly better results (see Table 2.5).

The proportion of the computation times the first 11 training algorithms require, is:  $3.2 \div 3 \div 1.2 \div 4.0 \div 3.5 \div 1.4 \div 10 \div 12 \div 12 \div 12 \div 1$ .

Based on the above the scaled conjugate gradient algorithm is proposed.

An indicative daily load curve and the respective load curve forecasted by the proposed ANN are presented in Fig. 2.7. The MAPE of the specific daily load curve is 1.173 %.

**Table 2.4** Intervals of the training parameters values

Training algorithm	Intervals of the training parameters values
1–2	$\alpha_0 = 0.1, 0.2, \dots, 0.9$ , $T_\alpha = 1,000, 1,200, \dots, 2,000$ , $\eta_0 = 0.1, 0.2, \dots, 0.9$ , $T_\eta = 1,000, 1,200, \dots, 2,000$
3	$\eta_0 = 0.01, 0.02, \dots, 0.1, 0.2, \dots, 3$
4	$\eta_0 = 0.1, 0.2, \dots, 3$
5–6	$\alpha_0 = 0.1, 0.2, \dots, 0.9$ , $T_\alpha = 1,200, 1,500, \dots, 6,000$ , $\eta_0 = 0.1, 0.2, \dots, 0.9$ , $T_\eta = 1,200, 1,500, \dots, 6,000$
7, 9	$s = 0.04, 0.1, 0.2$ , $T_{bv} = 20, 40$ , $T_{trix} = 50, 100$ , $e_{trix} = 10^{-6}$ , $10^{-5}$
8, 10	$s = 0.04, 0.1, 0.2$ , $T_{bv} = 20, 40$ , $T_{trix} = 50, 100$ , $e_{trix} = 10^{-6}$ , $10^{-5}$ , $\lim_{orthogonality} = 0.1, 0.5, 0.9$
11	$\sigma = 10^{-3}, 10^{-4}, 10^{-5}$ , $\lambda_0 = 10^{-6}, 10^{-7}, 5 \cdot 10^{-8}$
12	$\delta_1 = 0.1, 0.2, \dots, 0.5$ , $\delta_2 = 0.1, 0.2, \dots, 2$
13	–
14	$\lambda(0) = 0.1, 0.2, \dots, 1, 2, \dots, 5$ , $\beta = 2, 3, \dots, 9, 10, 20, \dots, 50$
Common parameters	$N_n = 20, 21, \dots, 70$ , activation function for hidden and output layer = hyperbolic tangent, linear, logistic, $a_1 = 0.1, 0.2, \dots$ , $0.5$ , $a_2 = 0.1, 0.2, \dots, 0.5$ , $b_1 = 0.0, \pm 0.1, \pm 0.2$ , $b_2 = 0.0$ , $\pm 0.1, \pm 0.2$

**Table 2.5** MAPE (%) of training, evaluation and test sets for all examined ANN training algorithms

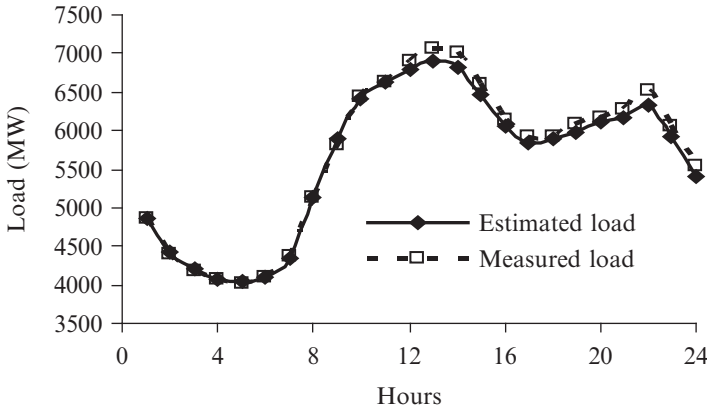
Training algorithm	MAPE (%)		
	Training set	Evaluation set	Test set
1	1.383	1.482	1.749
2	1.311	1.475	1.829
3	1.372	1.489	1.833
4	2.356	2.296	2.602
5	2.300	2.294	2.783
6	2.019	2.026	2.475
7	1.798	1.831	2.147
8	2.544	2.595	3.039
9	2.545	2.600	3.035
10	2.544	2.600	3.035
11	1.294	1.487	1.781
12–14	#	#	#

### 2.4.3 Study of ANN Inputs

In this paragraph, several scenarios regarding the input variables of the model are examined. A methodology based on the heuristic methodology introduced in paragraph 2.4.1 is used to obtain the optimal set of input variables and ANN parameters. The steps followed are briefly described next.

**Table 2.6** Number of hidden layer neurons, activation functions and training parameters for all examined ANN training algorithms

Training algorithm	Neurons	Activation functions	Rest parameters
1	45	$f_1 = \tanh(0.25x)$ , $f_0 = \tanh(0.25x)$	$\alpha_0 = 0.4$ , $T_\alpha = 1,800$ , $\eta_0 = 0.5$ , $T_\eta = 2,000$ , $e = 10^{-5}$ , max_epochs = 10,000
2	30	$f_1 = \tanh(0.40x)$ , $f_0 = 1/(1 + \exp(-0.25x))$	$\alpha_0 = 0.7$ , $T_\alpha = 1,800$ , $\eta_0 = 0.5$ , $T_\eta = 1,300$ , $e = 10^{-5}$ , max_epochs = 12,000
3	48	$f_1 = \tanh(0.50x)$ , $f_0 = \tanh(0.25x)$	$\eta_0 = 0.1$ , $e = 10^{-5}$ , max_epochs = 10,000
4	48	$f_1 = \tanh(0.40x)$ , $f_0 = 0.40x$	$\eta_0 = 2.0$ , $e = 10^{-5}$ , max_epochs = 12,000
5	25	$f_1 = \tanh(0.50x)$ , $f_0 = 0.25x$	$\alpha_0 = 0.9$ , $T_\alpha = 6,000$ , $\eta_0 = 0.9$ , $T_\eta = 6,000$ , $e = 10^{-5}$ , max_epochs = 12,000
6	22	$f_1 = \tanh(0.50x)$ , $f_0 = 0.25x$	$\alpha_0 = 0.9$ , $T_\alpha = 6,000$ , $\eta_0 = 0.9$ , $T_\eta = 6,000$ , $e = 10^{-5}$ , max_epochs = 12,000
7	43	$f_1 = \tanh(0.40x)$ , $f_0 = 0.20x$	$s = 0.04$ , $T_{bv} = 20$ , $T_{trix} = 50$ , $e = 10^{-5}$ , max_epochs = 5,000
8	43	$f_1 = \tanh(0.40x)$ , $f_0 = 0.20x$	$s = 0.04$ , $T_{bv} = 20$ , $T_{trix} = 50$ , $e = 10^{-5}$ , max_epochs = 5,000, $\lim_{orthogonality} = 0.9$
9	43	$f_1 = \tanh(0.40x)$ , $f_0 = 0.20x$	$s = 0.04$ , $T_{bv} = 20$ , $T_{trix} = 50$ , $e = 10^{-5}$ , max_epochs = 5,000
10	43	$f_1 = \tanh(0.40x)$ , $f_0 = 0.20x$	$s = 0.04$ , $T_{bv} = 20$ , $T_{trix} = 50$ , $e = 10^{-5}$ , max_epochs = 5,000, $\lim_{orthogonality} = 0.9$
11	52	$f_1 = \tanh(0.50x)$ , $f_0 = \tanh(0.25x)$	$\sigma = 10^{-5}$ , $\lambda_0 = 5 \times 10^{-8}$ , $e = 10^{-5}$ , max_epochs = 10,000
12-14	#	#	No convergence



**Fig. 2.7** Chronological load curve (measured and estimated load) of 1 day of the test set (Thursday, June 8, 2000)

(a) *Data selection*: In this step the input variables for the short term load forecasting model are defined.

- *1<sup>st</sup> scenario (basic)*: The input and output variables of the basic ANN configuration as described in paragraph 2.4.1 are used.
- *2<sup>nd</sup> scenario*: This scenario is same with the 1<sup>st</sup> except from the use of a seven-digit binary coding for the weekdays in replacement of the sinusoidal functions ( $\cos(2\pi d/7)$ ,  $\sin(2\pi d/7)$ ).
- *3<sup>rd</sup> scenario*: This scenario is same with the 2<sup>nd</sup> scenario except from the use of 3-h average temperature values of the  $d$ -th day and the previous one (for Athens and Thessalonica) in replacement of the group of inputs ANN\_Inp.2 up to ANN\_Inp.6. In this case the input vector of the ANN comprises 84 elements.
- *4<sup>th</sup> scenario*: This scenario is same with the 1<sup>st</sup> scenario except from the fact that only the hourly load measurements of the previous day ( $d - 1$ ) are used. In this scenario, the input vector comprises 47 elements.
- *5<sup>th</sup> scenario*: This scenario is same with the 1<sup>st</sup> scenario except from the fact that hourly load measurements for 3 days before the prediction day are used. In this case the input vector comprises 95 elements.
- *6<sup>th</sup> scenario*: In this scenario, PCA is applied to the input data used in the 2<sup>nd</sup> scenario eventually suppressing the input variables from 66 to 6 with a 99 % accumulated percentage of explained variance according to Kaiser's criterion [21].
- *7<sup>th</sup> scenario*: In this scenario, PCA is applied to the inputs used in 2<sup>nd</sup> scenario except from the ones dealing with the week days and the season of the year. The input variables are eventually suppressed from 66 to 19 with a 99 % accumulated percentage of explained variance according to Kaiser's criterion.

(b) *Data pre-processing*: As described in paragraph 2.4.1.

(c) *Main procedure*: The ANN is trained by using the scaled conjugate gradient algorithm (SCGA).

The optimal results obtained for each of the seven examined scenarios are summarized in Table 2.7. The basic scenario (no 1 of Table 2.7) leads to the minimum MAPE of the evaluation set. The second and the third scenario lead to similar results. It is worth mentioning that if MAPE of the test set is considered then the third scenario becomes the optimal while the first and the second scenarios perform similarly. MAPE increases in scenarios 4 and 5 while further increase occurs in scenarios 6 and 7.

The proportion of the computation times of the seven examined scenarios is:  $1 \div 0.98 \div 1.1 \div 0.8 \div 1.3 \div 0.4 \div 0.55$ . The use of compression techniques decreases the computational time significantly, but MAPE is considerably increased. According to the obtained results the first scenario is suggested for this case study.

#### 2.4.4 Study of ANN Outputs

The outputs of the ANN models examined so far are the 24 hourly load values that should be forecasted. This classical design imposes the use of ANNs with 24 output variables. An alternative solution comprises 24 different ANNs that are separately used for the prediction of each of the 24 hourly load demands. The performance of each of the 24 ANNs can be affected in a different way by the inputs it uses. Hence, various scenarios will be studied next regarding the inputs used. In all cases the scaled conjugate gradient training algorithm is used with its most crucial parameters being properly calibrated. Moreover, a large number of combinations of ANN parameters such as the number of neurons, the type of the activation functions, etc. are explored in order to finally obtain the optimal configuration of the short term load forecasting model and the optimal set of training parameters values. The performance of each of the examined forecasting models is evaluated by using the MAPE of the evaluation data set.

The scenarios examined are:

- 1<sup>st</sup> *scenario*: The ANN described in paragraph 2.4.1 trained with scaled conjugate gradient training algorithm is used.
- 2<sup>nd</sup> *scenario*: 24 ANNs are used; one for each of the predicted hourly average loads. The input variables of the ANNs are the same with those used in the 1<sup>st</sup> scenario except from the seven digit binary coding of the weekdays that is replaced by the two sinusoidal functions  $\cos(2\pi d/7)$  and  $\sin(2\pi d/7)$ .
- 3<sup>rd</sup> *scenario*: In this scenario, PCA is applied to the input variables used in 2<sup>nd</sup> scenario except from the ones dealing with the weekdays and the season of the year. The inputs are eventually suppressed from 66 to 19 with a 99 % accumulated percentage of explained variance according to Kaiser's criterion [21].

**Table 2.7** Results obtained from the examined scenarios and optimally calibrated ANN parameters

Scenario	MAPE (%)		Optimal number of neurons—range examined	Activation functions
	Training set	Evaluation set	Test set	
1	1.294	1.487	1.781	$f_1 = \tanh(0.50x)$ , $f_o = \tanh(0.25x)$
2	1.315	1.516	1.776	$f_1 = \tanh(0.50x)$ , $f_o = \tanh(0.25x)$
3	1.293	1.504	1.717	$f_1 = \tanh(0.50x)$ , $f_o = \tanh(0.25x)$
4	1.574	1.674	2.224	$f_1 = \tanh(0.30x)$ , $f_o = 1/(1 + \exp(-0.20x))$
5	1.524	1.804	1.808	$f_1 = \tanh(0.25x)$ , $f_o = 1/(1 + \exp(-0.25x))$
6	1.901	1.960	2.570	$f_1 = \tanh(0.30x)$ , $f_o = 1/(1 + \exp(-0.30x))$
7	1.458	1.583	1.900	$f_1 = \tanh(0.50x)$ , $f_o = \tanh(0.25x)$

- 4<sup>th</sup> *scenario*: The inputs of the  $i$ -th ANN (used for the forecasting of the average load demand of the  $i$ -th day hour) are:
  - The average load demands of the  $i$ -th,  $(i-1)$ -th and  $(i-2)$ -th hour of the previous 2 days of the day the load is forecasted.
  - The mean temperatures of Athens and Thessalonica for the current day and the previous one.
  - Two sinusoidal functions representing the week day.
  - Two sinusoidal functions representing the seasonal behaviour of the load.

In this scenario the input vector comprises 14 elements.

- 5<sup>th</sup> *scenario*: It is the same with the 4<sup>th</sup> scenario, but Principal Components Analysis is applied to all inputs except from those dealing with the weekly and seasonal behaviour of the load. The inputs are suppressed in a way that 99 % accumulated percentage of explained variance according to Kaiser's criterion. The number of the remaining inputs after PCA application is different for each ANN.

The *MAPE* estimated for the training, evaluation and test data sets and the output of the ANN used to forecast the load demand at the 12-th hour of the day of the year, are shown in Table 2.8. Moreover, the optimally calibrated parameters of the ANN are registered in the same Table. It is noted here, that the parameters of the training algorithm, the type and the parameters of the activation functions of the hidden and the output layers are the same with those used in the 1<sup>st</sup> scenario.

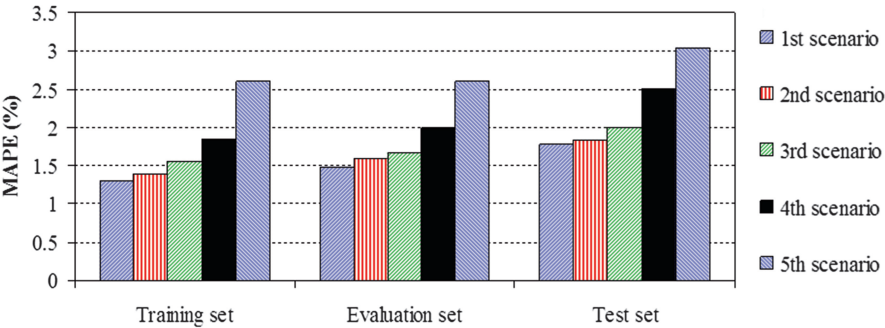
Respectively, the *MAPE* of training, evaluation and test sets and for all examined scenarios are shown in Fig. 2.8 and Table 2.9. For scenarios 2–5 the average of the *MAPE* of the 24 ANNs is calculated.

The proportion of the computation times of the examined scenarios is:  $1 \div 0.4 \div 0.05 \div 0.15 \div 0.03$ . This indicates that the computation time decreases significantly when the dimension of the input vector decreases. Even larger decrease could be achieved in scenarios 2–5 if 24 different computers were used (parallel processing) for the training of each ANN. However, the 1<sup>st</sup> scenario leads to the minimum *MAPE* of the evaluation data set while the second and the third scenario lead to slightly worse results. *MAPE* of test set remains almost the same in 1<sup>st</sup> and 2<sup>nd</sup> scenario. In the 4<sup>th</sup> scenario, leaving out some of the input data led to worse results. The application of PCA (3<sup>rd</sup> and 5<sup>th</sup> scenarios) decreases significantly the computation time but the performance of the forecasting model deteriorates. This becomes even more evident if *MAPE* of the test data set is considered. Based on the previous observations the data compression is not suggested while the 2<sup>nd</sup> scenario presents satisfactory behavior regarding *MAPE* and it could be preferred to the 1<sup>st</sup> scenario if minimization of the computation time is of high importance.

**Table 2.8** MAPE (%) of training, evaluation and test sets for the ANN forecasting the load at 12:00

Scenario	MAPE (%) of training set	MAPE (%) of evaluation set	MAPE (%) of test set	Neurons—Range of examined neurons	Activation functions
2	1.347	1.824	1.847	32 (10–60)	$f_1 = \tanh(0.50x)$ , $f_2 = \tanh(0.25x)$
3	1.492	1.876	1.924	17 (2–25)	$f_1 = \tanh(0.50x)$ , $f_2 = \tanh(0.25x)$
4	1.747	2.210	2.620	40 (10–60)	$f_1 = \tanh(0.25x)$ , $f_2 = \tanh(0.25x)$
5	2.120	2.404	2.953	14 (2–25)	$f_1 = \tanh(0.50x)$ , $f_o = \tanh(0.25x)$

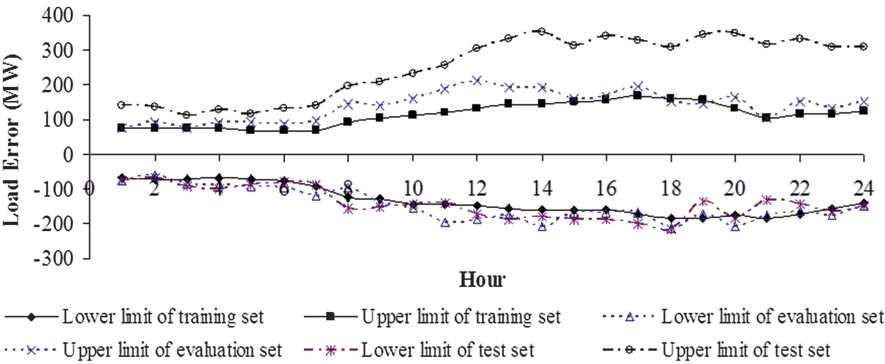




**Fig. 2.8** MAPE (%) of training, evaluation and test set for the interconnected Greek power system and the 5 different scenarios of output variables

**Table 2.9** MAPE (%) of training, evaluation and test sets for the 24-h load prediction

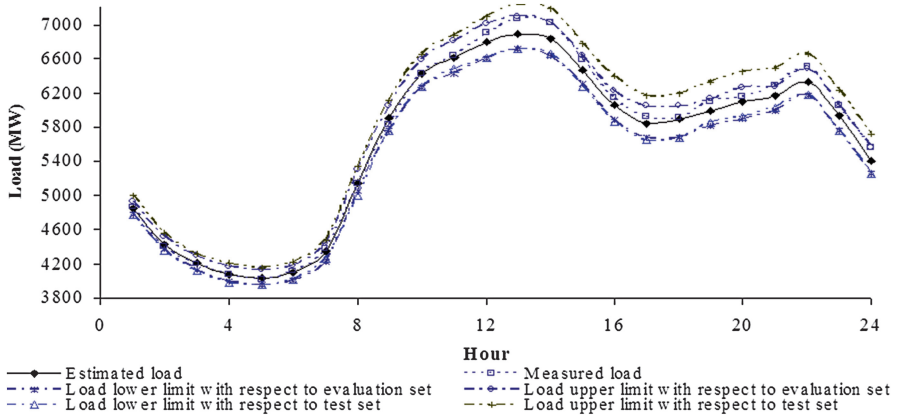
Scenario	MAPE (%)		
	Training set	Evaluation set	Test set
1	1.294	1.487	1.781
2	1.390	1.603	1.830
3	1.565	1.667	1.994
4	1.850	1.989	2.503
5	2.596	2.598	3.043



**Fig. 2.9** 90 % confidence interval limits with respect to the training, evaluation and test sets for the best ANN model for 8-6-2000 in Greek interconnected power system

2.4.5 Estimation of the Confidence Interval

Using the basic ANN configuration described in paragraph 2.4.1 the 90 % confidence interval is estimated using the re-sampling technique with the probability in each tail equal to 5 %. In Fig. 2.9 the prediction errors of a typical summer day for Greek interconnected power system of the year 2000 (Thursday 8-6-2000) are presented for the training, evaluation and test sets respectively, while in Fig. 2.10



**Fig. 2.10** Chronological active load curves of the measured load, the estimated load, the estimated load with the 5 % lower limit with respect to evaluation set, the estimated load with the 5 % upper limit with respect to evaluation set, the estimated load with the 5 % lower limit with respect to test set, the estimated load with the 5 % upper limit with respect to test set for the best ANN model for 8-6-2000 in Greek interconnected power system

the respective measured and estimated load values are presented together with the 90 % confidence intervals of the evaluation and the test data sets.

It is observed in Fig. 2.10 that the lower limits of the confidence intervals for the three data sets are quite similar. The ratio between the lower hourly errors of the test set to the respective one of the evaluation set varies between 0.71 and 1.60, while the mean value is equal to 1.00. However, the upper limit of the confidence intervals for the test set is almost the double of the respective one of the evaluation set. The ratio between the upper hourly errors of the test set to the respective one of the evaluation set varies between 1.22 and 3.02, while the mean value is equal to 1.78. It is observed in Fig. 2.10 that the confidence interval of the test set is broader than the respective one of the evaluation set. Similar behavior is presented for all days studied. In fact the limits of the test set are unknown in real applications, which means that they should be corrected, i.e. using the proper multiplying factor which is calculated by trial executions with historical data [66].

#### 2.4.6 Effect of the Special Days

Load curves of special days differ significantly from the respective ones of the regular days. One approach to dealing with this problem is to ignore the irregularity of the special days and include their data into the training and test sets. However, ANN performance is generally expected to deteriorate in this case.

In a different approach, proposed in [44], the load curve of a special day is decomposed in two components; one representing the load of a normal day and

another representing the special day effect. More specifically, the load is expressed as follows:

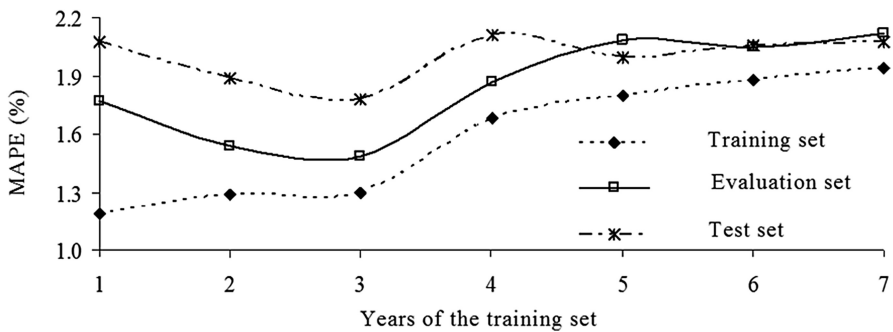
$$\bar{L}_{holiday} = \bar{L}_{normal} - \Delta\bar{L}_{holiday} \quad (2.50)$$

Hence, the vectors of the input data set corresponding to special days are increased by the special day corrective term,  $\Delta\bar{L}_{holiday}$ , before they are used in training process. Finally, the vectors used for the training are of the form  $\bar{L}_{normal} = N(\bar{X} + \Delta\bar{X}_{holiday})$ . Where,  $\bar{X}$  is the input vector comprising load data containing, temperature measurements and  $\Delta\bar{X}_{holiday}$  the correction term of the effect of the special days. The last term is calculated by reusing load data of respective special days of previous years.

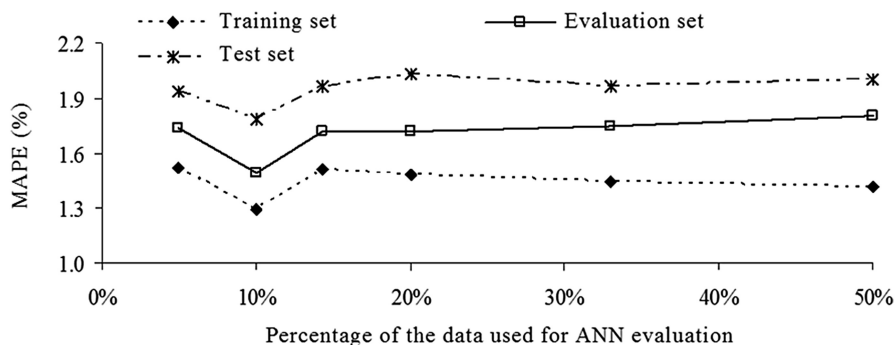
Another way to deal with the effect of the special days is to properly group the input data with emphasis placed on the special days [70].

#### 2.4.7 The Effect of the Time Period Length Used for the Training of the ANN

In order to study the effect of the duration of the training data time period seven different time periods were used. More specifically, these time periods extend on the last one, two, ..., and seven years before the reference year, respectively. The MAPE of the training, test and validation data sets obtained for the basic ANN configuration of paragraph 2.4.1 (trained with the scaled conjugate gradient algorithm) is shown in Fig. 2.11. It appears that in case of short term load forecasting in Greek power system using data from the last 3 years leads to the best results regarding MAPE minimization. The use of validation test set is necessary as it helps to achieve better generalization results and it seems to have similar behaviour with the test set rather than the training set.



**Fig. 2.11** MAPE of training, validation and test sets versus the training years. The scaled conjugate gradient training algorithm is use and the 10 % of the data is used for model evaluation



**Fig. 2.12** MAPE of training, validation and test sets versus the percentage of the data used for ANN evaluation. Scaled conjugate gradient training algorithm is used

Another important issue that should be carefully studied is the percentage of the data used for the evaluation of the model. Due to the abundance of the available data it was decided that the training and evaluation data sets should not overlap. For the specific case study, it is proved that the minimum MAPE is obtained if the 10 % of the data is used for evaluation purposes. The respective results are shown in Fig. 2.12.

## 2.5 Conclusions

Short term load forecasting in electric power systems is a very complex problem. ANNs have been proved very efficient in short term load forecasting and they constitute nowadays the predominant method in this field. The basics of ANNs and the major training techniques are described at the beginning of this chapter. Next, several variations concerning the structure of the ANN model, the training method and its parameters have been examined and applied to the Greek power system for short term load forecasting. The results obtained reveal that general conclusions cannot be easily extracted as the performance of each of the examined forecasting models depends highly on the characteristics of each case study. However, it seems that in most cases the basic ANN configuration described in 2.4.1 seems to be the most appropriate for application to Greek power system. Data compression techniques reduce greatly computation time but generally deteriorate performance. Hence, they could be avoided if computation time is not a constraining factor.

Further work on this topic could be the combination of ANNs with other techniques like fuzzy logic, GA algorithms etc. Moreover, short term load forecasting in emerging types of power systems incorporating extensive smart load demand management will be a topic of great importance in the forthcoming years.

## References

1. Hobbs, B.F., Jitprapaikularn, S., Maratukulam, D.J.: Analysis of the value for unit commitment of improved load forecasts. *IEEE Trans. Power Syst.* **14**(4), 1342–48 (1999)
2. Gooi, H.B., Mendes, D.P., Bell, K.R.W., Kirschen, D.S.: Optimal scheduling of spinning reserve. *IEEE Trans. Power Syst.* **14**(4), 1485–1492 (1999)
3. Gravener, M.H., Nwankpa, C.: Available transfer capability and first order sensitivity. *IEEE Trans. Power Syst.* **14**(2), 512–518 (1999)
4. Fan, J.Y., McDonald, J.D.: A real-time implementation of short-term load forecasting for distribution power systems. *IEEE Trans. Power Syst.* **9**(2), 988–994 (1994)
5. Haidda, T., Muto, S.: Regression based peak load forecasting using a transformation technique. *IEEE Trans. Power Syst.* **9**(4), 1788–1794 (1994)
6. Rupanagunta, P., Baughman, M.L., Jones, J.W.: Scheduling of cool storage using non-linear programming techniques. *IEEE Trans. Power Syst.* **10**(3), 1279–1285 (1995)
7. Kassaei, H.R., Keyhani, A., Woung, T., Rahman, M.: A hybrid fuzzy neural network bus load modeling and prediction. *IEEE Trans. Power Syst.* **14**(2), 718–724 (1999)
8. Yang, H.T., Huang, C.M., Huang, C.L.: Identification of ARMAX model for short term load forecasting: an evolutionary programming approach. *IEEE Trans. Power Syst.* **11**(1), 403–408 (1996)
9. Hippert, H.S., Pedreira, C.E., Souza, R.C.: Neural networks for short-term load forecasting: a review and evaluation. *IEEE Trans. Power Syst.* **16**(1), 44–55 (2001)
10. Mastorocostas, P.A., Theocharis, J.B., Bakirtzis, A.G.: Fuzzy modeling for short-term load forecasting using the orthogonal least squares method. *IEEE Trans. Power Syst.* **14**(1), 29–36 (1999)
11. Infield, D.G., Hill, D.C.: Optimal smoothing for trend removal in short term electricity demand forecasting. *IEEE Trans. Power Syst.* **13**(3), 1115–1120 (1998)
12. Charytoniuk, W., Chen, M.S., Van Olinda, P.: Nonparametric regression based short-term load forecasting. *IEEE Trans. Power Syst.* **13**(3), 725–730 (1998)
13. Khotanzad, A., Hwang, R.C., Abaye, A., Maratukulam, D.: An adaptive modular artificial neural network hourly load forecaster and its implementation at electric utilities. *IEEE Trans. Power Syst.* **10**(3), 1716–1722 (1995)
14. Khotanzad, A., Afkhami-Rohani, R., Lu, T.L., Abaye, A., Davis, M., Maratukulam, D.: ANNSTLF- neural network-based electric load forecasting system. *IEEE Trans. Neural Netw.* **8**(4), 835–846 (1996)
15. Khotanzad, A., Afkhami-Rohani, R., Maratukulam, D.: Artificial neural network short-term load forecaster-generation three. *IEEE Trans. Power Syst.* **13**(4), 1413–1422 (1998)
16. Papalexopoulos, A.D., Hao, S., Peng, T.M.: An implementation of a neural network based load forecasting model for the EMS. *IEEE Trans. Power Syst.* **9**(4), 1956–62 (1994)
17. Mohammed, O., Park, D., Merchant, R., Dinh, T., Tong, C., Azeem, A.: Practical experiences with an adaptive neural network short-term load forecasting system. *IEEE Trans. Power Syst.* **10**(1), 254–265 (1995)
18. Wang, X., Shi, M.: Matlab for forecasting of electric power load based on BP neural network. *Proceedings ICICIS 2011, Part I, Series: Communications in Computer and Information Science* **134**: 636–642 (2011)
19. Barghinia, S., Ansarimehr, P., Habibi, H., Vafadar, N.: Short-term load forecasting of Iran national power system using artificial neural network, p. 5. *IEEE Porto Power Tech Conference, Porto, Portugal* (2001)
20. Kalaitzakis, K., Stavrakakis, G.S., Anagnostakis, E.M.: Short-term load forecasting based on artificial neural networks parallel implementation. *Electr. Power Syst. Res.* **63**, 185–196 (2002)
21. Saini, L.M., Soni, M.K.: Artificial neural network-based peak load forecasting using conjugate gradient methods. *IEEE Trans. Power Syst.* **17**(3), 907–912 (2002)
22. Gontar, Z., Hatziaargyriou, N.: Short term load forecasting with radial basis function network. *2001 IEEE Porto Power Tech Conference 10th-13th September, p. 4. Porto, Portugal* (2001)

23. Yun, Z., Quan, Z., Caixin, S., Shaolan, L., Yuming, L., Yang, S.: RBF neural network and ANFIS-based short-term load forecasting approach in real-time price environment. *IEEE Trans. Power Syst.* **23**(3), 853–858 (2008)
24. Ko, C.N., Lee, C.M.: Short-term load forecasting using SVR (support vector regression)-based radial basis function neural network with dual extended Kalman filter. *Energy* **49**, 413–422 (2013)
25. Vermaak, J., Botha, E.C.: Recurrent neural networks for short-term load forecasting. *IEEE Trans. Power Syst.* **13**(1), 126–132 (1998)
26. Chen, S.T., Yu, D.C., Moghaddamjo, A.R.: Weather sensitive short-term load forecasting using nonfully connected artificial neural network. *IEEE Trans. Power Syst.* **7**(3), 1098–1105 (1992)
27. Abdel-Aal, R.E.: Short-term hourly load forecasting using abductive networks. *IEEE Trans. Power Syst.* **19**(1), 164–173 (2004)
28. Ranaweera, D.K., Karady, G.G., Farmer, R.G.: Effect of probabilistic inputs on neural network-based electric load forecasting. *IEEE Trans. Neural Netw.* **7**(6), 1528–1532 (1996)
29. Taylor, J.W., Buizza, R.: Neural network load forecasting with weather ensemble predictions. *IEEE Trans. Power Syst.* **17**(3), 626–632 (2002)
30. Senjyu, T., Takara, H., Uezato, K., Funabashi, T.: One-hour-ahead load forecasting using neural network. *IEEE Trans. Power Syst.* **17**(1), 113–118 (2002)
31. Kim, K.H., Park, J.K., Hwang, K.J., Kim, S.H.: Implementation of hybrid short-term load forecasting system using artificial neural networks and fuzzy expert systems. *IEEE Trans. Power Syst.* **10**(3), 1534–1539 (1995)
32. Kim, K.H., Youn, H.S., Kang, Y.C.: Short-term load forecasting for special days anomalous load conditions using neural networks and fuzzy inference method. *IEEE Trans. Power Syst.* **15**(2), 559–565 (2000)
33. Srinivasan, D., Tan, S.S., Chang, C.S., Chan, E.K.: Parallel neural network-Fuzzy expert system strategy for short-term load forecasting: system implementation and performance evaluation. *IEEE Trans. Power Syst.* **14**(3), 1100–1106 (1999)
34. Daneshdoost, M., Lotfalian, M., Bumroongit, G., Ngoy, J.P.: Neural network with fuzzy set-based classification for short-term load forecasting. *IEEE Trans. Power Syst.* **13**(4), 1386–1391 (1998)
35. Banik, S., Anwer, M., Khodadad Khan, A.F.M.: Predictive Power of the Daily Bangladeshi Exchange Rate Series based on Markov Model, Neuro Fuzzy Model and Conditional Heteroskedastic Model. *Proceedings of 12th International Conference on Computer and Information Technology (ICCIT 2009)*, Dhaka, Bangladesh (2009)
36. Tanabe, T., Ueda, Y., Suzuki, S., Ito, T., Sasaki, N., Tanaka, T., Funabashi, T., Yokoyama, R.: Optimized operation and stabilization of microgrids with multiple energy resources. *7th International conference on Power Electronics, EXCO*, Daegu, Korea (2007)
37. Jaipraditham, C.: Next day load demand forecasting of future in electrical power generation on distribution networks using Adaptive Neuro-Fuzzy Inference. *1st International Power and Energy Conference PECon*, Putrajaya, Malaysia (2006)
38. Malkocecic, D., Konjic, T., Miranda, V.: Preliminary comparison of different neural-fuzzy mappers for load curve short term prediction. *8th Seminar on Neural Network Applications in Electrical Engineering, NEUREL-2006*, Faculty of Electrical Engineering, University of Belgrade, Serbia (2006)
39. Wang, X., Hatziaargyriou, N., Tsoukalas, L.H.: A new methodology for nodal load forecasting in deregulated power systems. *IEEE Power Eng. Rev.* **12**, 48–51 (2002)
40. Chen, Y., Luh, P.B., Guan, C., Zhao, Y., Michel, L.D., Coolbeth, M.A., Friedland, P.B., Rourke, S.J.: Short-term load forecasting: similar day-based wavelet neural networks. *IEEE Trans. Power Syst.* **25**(1), 322–330 (2010)
41. Ferreira, V.H., Alves da Silva, A.P.: Toward estimating autonomous neural network-based electric load forecasters. *IEEE Trans. Power Syst.* **22**(4), 1554–1562 (2007)
42. Cardenas, J.J., Romeral, L., Garcia, A., Andrade, F.: Load forecasting framework of electricity consumptions for an intelligent energy management system in the user-side. *Expert Syst. Appl.* **39**, 5557–5565 (2012)

43. Amjady, N., Keynia, F.: A new neural network approach to short term load forecasting of electrical power systems. *Energies* **4**, 488–503 (2011)
44. Bakirtzis, A.G., Petridis, V., Kiartzis, S.J., Alexiadis, M.C., Maissis, A.H.: A neural network short term load forecasting model for the Greek power system. *IEEE Trans. Power Syst.* **11**(2), 858–863 (1996)
45. Kiartzis, S.J., Zournas, C.E., Theocharis, J.M., Bakirtzis, A.G., Petridis, V.: Short term load forecasting in an autonomous power system using artificial neural networks. *IEEE Trans. Power Syst.* **12**(4), 1591–1596 (1997)
46. Tsekouras, G.J., Kanellos, F.D., Kontargyri, V.T., Tsirekis, C.D., Karanasiou, I.S., Elias, C.N., Salis, A.D., Mastorakis, N.E.: A comparison of artificial neural networks algorithms for short term load forecasting in Greek intercontinental power system. *WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing, Puerto De La Cruz, Canary Islands, Spain* (2008)
47. Tsekouras, G.J., Kanellos, F.D., Kontargyri, V.T., Tsirekis, C.D., Karanasiou, I.S., Elias, C.N., Salis, A.D., Kontaxis, P.A., Mastorakis, N.E.: Short term load forecasting in Greek Intercontinental Power System using ANNs: a Study for Input Variables. *10th WSEAS International Conference on Neural Networks Prague, Czech Republic* (2009)
48. Tsekouras, G.J., Kanellos, F.D., Elias, C.N., Kontargyri, V.T., Tsirekis, C.D., Karanasiou, I.S., Salis, A.D., Kontaxis, P.A., Gialketsi, A.A., Mastorakis, N.E.: Short term load forecasting in Greek interconnected power system using ANN: A study for output variables. *15th WSEAS International Conference on Systems Corfu, Greece* (2011)
49. Choueiki, M.H., Mount-Campbell, C.A., Ahalt, S.C.: Implementing a weighted least squares procedure in training a neural network to solve the short-term load forecasting problem. *IEEE Trans. Power Syst.* **12**, 1689–1694 (1997)
50. Metaxiotis, K., Kagiannas, A., Askounis, D., Psarras, J.: Artificial intelligence in short term electric load forecasting: a state-of-the-art survey for the researcher. *Energy Convers. Manag.* **44**, 1525–1534 (2003)
51. Kyriakides, E., Polycarpou, M.: Short term electric load forecasting: a tutorial (Chapter 16). In: Chen, K., Wang, L. (eds.), *Trends in neural computation, studies in computational intelligence*, 35 pp. 391–418. Springer (2007)
52. Hahn, H., Meyer-Nieberg, S., Pickl, S.: Electric load forecasting methods: tools for decision making. *Eur. J. Oper. Res.* **199**, 902–907 (2009)
53. Felice, M., Yao, X.: Short-term load forecasting with neural network ensembles: a comparative study. *IEEE Comput. Intell. Mag.* **6**(3), 47–56 (2011)
54. Rego, L.P., Santana, A.L., Conde, G., Silva, M.S., Francês, C.R.L., Rocha, C.A.: Comparative analyses of computational intelligence models for load forecasting: a case study in the Brazilian Amazon power suppliers. *Lecture Notes Comput. Sci.* **5553**, 1044–1053 (2009)
55. Haykin, S.: *Neural networks: a comprehensive foundation*. Prentice Hall (1994)
56. Likas, A.: *Computational Intelligence (in Greek)*, 1st edn. Ioannina, Greece (1999)
57. Ghosh, P.S., Chakravorti, S., Chatterjee, N.: Estimation of time-to-flashover characteristics of contaminated electrolytic surfaces using a neural network. *IEEE Trans Dielectr. Electr. Insul.* **2**(6), 1064–1074 (1995)
58. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. *Comput. J.* **7**, 149–154 (1964)
59. Polak, E.: *Computational methods in optimization: a unified approach*, 1st edn. Academic Publication, New York (1971)
60. Powell, M.J.: Restart procedures for the conjugate gradient method. *Math. Program.* **12**, 241–254 (1977)
61. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* **6**, 525–533 (1993)
62. Levenberg, K.: A method for the solution of certain problems in least squares. *Quart. Appl. Math.* **2**, 164–168 (1944)
63. Marquardt, D.: An algorithm for least squares estimation of nonlinear parameters. *SIAM J. Appl. Math.* **11**, 431–441 (1963)

64. Silva, A.P.A., Moulin, L.S.: Confidence intervals for neural network based short-term load forecasting. *IEEE Trans. Power Syst.* **15**(4), 1191–1196 (2000)
65. Mastorakis, N.E., Tsekouras, G.J.: Short term load forecasting in Greek intercontinental power system using ANN: the confidence interval. *Advanced aspects of theoretical electrical engineering*. Sozopol, Bulgaria (2010)
66. Tsekouras, G.J., Mastorakis, N.E., Kanellos, F.D., Kontargyri, V.T., Tsirekis, C.D., Karanasiou, I.S., Elias, C.N., Salis, A.D., Kontaxis, P.A., Gialketsi, A.A.: Short term load forecasting in Greek interconnected power system using ANN: Confidence Interval using a novel re-sampling technique with corrective Factor. *WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing*, Vouliagmeni, Athens, Greece (2010)
67. Khosravi, A., Nahavandi, S., Creighton, D.: Construction of optimal prediction intervals for load forecasting problems. *IEEE Trans. Power Syst.* **25**(3), 1496–1503 (2010)
68. Quan, H., Srinivasan, D., Khosravi, A.: Uncertainty handling using neural network-based prediction intervals for electrical load forecasting. *Energy* **73**, 916–925 (2014)
69. Hand, D., Manilla, H., Smyth, P.: *Principles of data mining*, 1st edn. The MIT Press, Cambridge (2001)
70. Fidalgo, J.N., Peças Lopes, J.A.: Load forecasting performance enhancement when facing anomalous events. *IEEE Trans. Power Syst.* **20**(1), 408–415 (2005)



Computational Problems in Science and Engineering

Mastorakis, N.; Bulucea, A.; Tsekouras, G. (Eds.)

2015, VII, 491 p., Hardcover

ISBN: 978-3-319-15764-1