Learning and Nonlinear Models – Revista da Sociedade Brasileira de Redes Neurais, Vol. 3, No. 1, pp. 19-26,2005

REGULARIZATION OF SHORT TERM LOAD FORECASTING NEURAL MODELS

Vitor Hugo Ferreira, Alexandre P. Alves da Silva
Laboratório de Sistemas de Potência, Programa de Engenharia Elétrica, COPPE/UFRJ, 21945-970,
Rio de Janeiro, RJ, Brasil
e-mails: vitor@vishnu.coep.ufrj.br, alex@coep.ufrj.br

Abstract — The knowledge of loads' future behavior is very important for decision making in power system operation. During the last years, many load models have been proposed, and the neural ones have presented the best results. One of the disadvantages of the neural models for load forecasting is the possibility of excessive adjustment of the training data, named overfitting, which degrades the generalization capacity of the estimated models. This problem can be tackled by using regularization techniques. This paper shows the application of some of these techniques to short term load forecasting.

Index Terms — Short-term load forecasting, artificial neural networks, regularization techniques, Bayesian training, gain scaling, support vector machines.

1 Introduction

Operational decisions in power systems, such as unit commitment, economic dispatch, AGC, and maintenance scheduling, depend on the future behavior of the load [1]. Therefore, several load forecasting models have been proposed during the last forty years. Such a long experience in dealing with the load forecasting problem has shown some successful models such as multiple regression, Box-Jenkins, neural networks, including multilayer perceptron (MLP) and radial basis function (RBF) networks, fuzzy logic based and hybrid models.

One of the advantages of neural network models in short-term load forecasting is the property of being universal aproximators. However, this theoretical advantage can backfire if data overfitting is not avoided. Therefore, complexity control is crucial for nonlinear models. The main objective of model complexity control is to match the data regularity with the model nonlinearity, avoiding overfitting or underfitting, and maximizing the generalization capacity.

This paper compares different procedures for controlling the complexity of feedforward neural networks. In order to minimize the out-of-sample forecasting error, Bayesian training [8], [10], activation function gain scaling [14] and Support Vector Machines (SVM) [5]-[7] are compared. The database for testing corresponds to the load and temperature series, in hourly basis, from a North-American electric utility (http://www.ee.washington.edu/class/559/2002spr), which has been used in several load forecasting studies. One day ahead predictions have been simulated to compare the different proposals for regularizing the neural models.

2 Artificial Neural Networks

The neural network models commonly used in electric load forecasting have a feedforward structure with one hidden layer only. The following section presents the popular multilayer perceptron and the recently proposed Support Vector Machine, which has the MLP as a special case.

2.1 Multi-Layer Perceptrons

Let $\underline{x} \in \mathbb{R}^n$ be a vector representing input signals, $d \in \mathbb{R}$ the corresponding desired output, $\underline{w} \in \mathbb{R}^M$ the vector of connection weights of the neural network, with $M = mn + 2m + 1$, where m is the number of neurons in the hidden layer, and b and b_k, $k = 1, 2, ..., m$, the bias for the activation functions $\varphi(\cdot): \mathbb{R} \to \mathbb{R}$. The neural network output is represented by $y \in \mathbb{R}$, such that $y = f(\underline{x}, \underline{w})$. In the case of an MLP, the output is:

$$c_k = \varphi\left[\sum_{j=1}^{n}\left(w_{kj} x_j\right) + b_k\right], k = 1, 2, ..., m \qquad (1)$$

$$y = f(\underline{x}, \underline{w}) = \sum_{i=1}^{m} w_i c_i + b$$

Given a dataset D with N input/output pairs, $D = \{\underline{x}_i, d_i\}$, $i = 1, 2, ..., N$, the traditional training objective of an MLP is to estimate the weight vector $\underline{w}$ such that the empirical risk is minimized, i.e.:

$$\min_{\underline{w}} \left\{ E_s(\underline{w}, D) = \frac{1}{2} \sum_{j=1}^{N} \left[d_i - f(\underline{x}_i, \underline{w}) \right]^2 \right\} \quad (2)$$

There are several algorithms for minimizing (2). Independently of using the classical error backpropagation, or second order methods, such as the Levenberg-Marquardt [8], the main drawback of those training methods is the risk of overfitting.

2.2 Support Vector Machines (SVMs)

For SVMs, the model output is given by:

$$y = \sum_{j=0}^{m} W_j \phi_j(\underline{x}) = \underline{W}^t \underline{\phi}(\underline{x}) \quad (3)$$

with $\underline{\phi}(\underline{x}) = [1, \phi_1(\underline{x}), \phi_2(\underline{x}), \ldots, \phi_m(\underline{x})]^t$ and $\underline{W} = [b, W_1, W_2, \ldots, W_m]^t$, where $\underline{\phi}(\underline{x})$ represents a set of nonlinear basis functions. Let the loss function be the ε-insensitive $L_\varepsilon(d, y)$, given by [5]:

$$L_\varepsilon(d, y) = \begin{cases} |d - y| - \varepsilon, & |d - y| \geq \varepsilon \\ 0, & |d - y| < \varepsilon \end{cases} \quad (4)$$

In equation (4), ε can be understood as the additive noise variance of the regression model [6]. In the following development, ε is assumed to be known, i.e., defined by the user. The training objective of an SVM model is the minimization of the empirical risk, i.e.:

$$\min_{\underline{W}} \left\{ E_s(\underline{W}, D) = \frac{1}{N} \sum_{i=1}^{N} L_\varepsilon(d_i, y_i) \right\} \quad (5)$$

subject to

$$\|\underline{W}\|^2 \leq c_0 \quad (6)$$

where c_0 is responsible for the model complexity control [6].

The nonlinear constraint in (6) can be incorporated by the objective function with the addition of new constraints, as follows:

$$\min \left\{ \Phi(\underline{W}, \underline{\xi}, \underline{\xi}') = C \left[\sum_{i=1}^{N} (\xi_i + \xi_i') \right] + \frac{1}{2} \underline{W}^t \underline{W} \right\} \quad (7)$$

where $\underline{\xi} = [\xi_1, \xi_2, ..., \xi_N]^t$ and $\underline{\xi}' = [\xi_1', \xi_2', ..., \xi_N']^t$

subject to

$$\begin{aligned} & d_i - \underline{W}^T \underline{\phi}(\underline{x}_i) \leq \varepsilon + \xi_i \\ & \underline{W}^T \underline{\phi}(\underline{x}_i) - d_i \leq \varepsilon + \xi_i' \\ & \xi_i \geq 0, \quad \xi_i' \geq 0, \quad i = 1, 2, ..., N \end{aligned} \quad (8)$$

In equation (7), C is a pre-specified parameter responsible for the balance between the training data fitting and the model complexity. In practice, this parameter is empirically determined using resampling techniques, such as cross-validation.

In order to solve the optimization problem formulated in (7) and (8), the following Lagrangean function can be defined:

$$J\left(\underline{W},\underline{\xi},\underline{\xi}',\underline{\alpha},\underline{\alpha}',\underline{\gamma},\underline{\gamma}'\right)=C\left[\sum_{i=1}^{N}\left(\xi_i+\xi_i'\right)\right]+\frac{1}{2}\underline{W}^t\underline{W} \tag{9}$$
$$-\sum_{i=1}^{N}\alpha_i\left[\underline{W}^t\underline{\phi}\left(\underline{x}_i\right)-d_i+\varepsilon+\xi_i\right]$$
$$-\sum_{i=1}^{N}\alpha_i'\left[d_i-\underline{W}^t\underline{\phi}\left(\underline{x}_i\right)+\varepsilon+\xi_i'\right]$$
$$-\sum_{i=1}^{N}\left(\gamma_i\xi_i+\gamma_i'\xi_i'\right)$$

$$\underline{\alpha}=\left[\alpha_1,\alpha_2,...,\alpha_N\right]^t,\ \underline{\alpha}'=\left[\alpha_1',\alpha_2',...,\alpha_N'\right]^t,$$
$$\underline{\gamma}=\left[\gamma_1,\gamma_2,...,\gamma_N\right]^t,\ \underline{\gamma}'=\left[\gamma_1',\gamma_2',...,\gamma_N'\right]^t$$

where $\underline{\alpha}$ and $\underline{\alpha}'$ are the Lagrange multipliers.

Equation (10) is obtained from the optimality conditions of (9),

$$\underline{W}=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i'\right)\underline{\phi}\left(\underline{x}_i\right) \tag{10}$$
$$\gamma_i=C-\alpha_i,\ \ \gamma_i'=C-\alpha_i',\ \ i=1,2,...,N$$

The dual maximization problem corresponding to the primal minimization problem in (9) is formulated as:

$$\max\left\{Q\left(\underline{\alpha},\underline{\alpha}'\right)=\sum_{i=1}^{N}d_i\left(\alpha_i-\alpha_i'\right)-\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i'\right)\right. \tag{11}$$
$$\left.-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i-\alpha_i'\right)\left(\alpha_j-\alpha_j'\right)K\left(\underline{x}_i,\underline{x}_j\right)\right\}$$

subject to:

$$\sum_{i=1}^{N}\left(\alpha_i-\alpha_i'\right)=0 \tag{12}$$
$$0\le\alpha_i\le C,\ \ 0\le\alpha_i'\le C,\ \ i=1,2,...,N$$

In equation (11), $K\left(\underline{x}_i,\underline{x}_j\right)=\underline{\phi}^T\left(\underline{x}_i\right)\underline{\phi}\left(\underline{x}_j\right)$ is the inner product kernel defined according to Mercer's theorem [5]. Therefore, the output of an SVM is given by:

$$y=f\left(\underline{x},\underline{W}\right)=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i'\right)K\left(\underline{x},\underline{x}_i\right) \tag{13}$$

As can be seen from equation (13), the support vectors are the training patterns in which $\alpha_i\neq\alpha_i'$, i.e., the ones located outside the band defined by ε. An SVM model is, in fact, a feedforward neural network model with hidden layer units activation functions defined by the kernels $K\left(\underline{x},\underline{x}_i\right)$, as in Fig. 1.

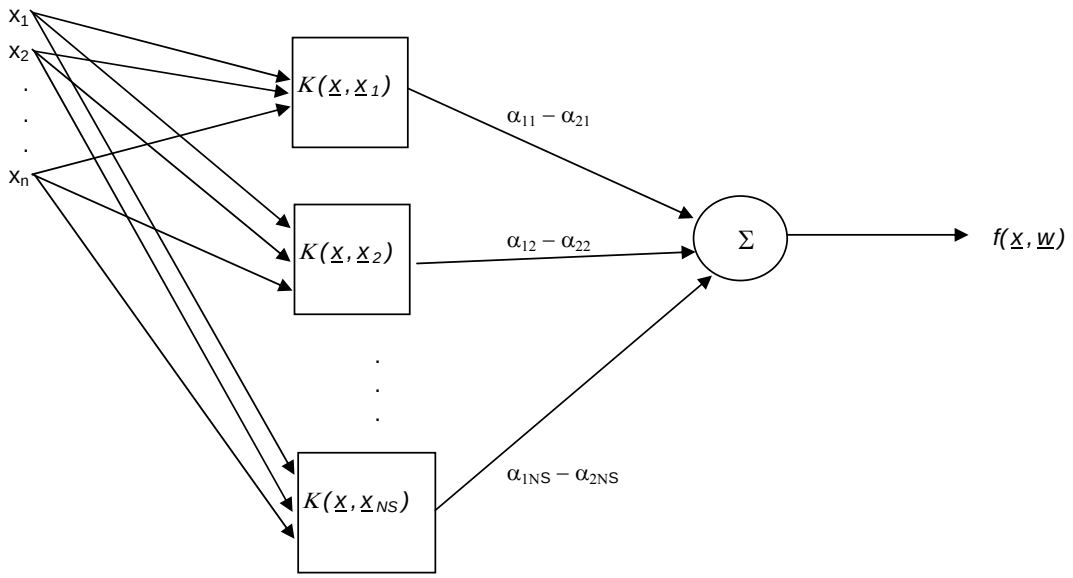

Fig. 1 – SVM architecture

2.3 Multilayer Perceptron versus Support Vector Machine

From the previous presentation, several important differences between MLPs and SVMs can be observed. First of all, training an MLP via error backpropagation presumes the pre-specification of the neural network architecture. With SVMs, the model structure is a by-product of training, depending on the parameters ε and C. Objective functions to be minimized in MLP training via error backpropagation generally focus on the empirical risk (training set error), only. On the other hand, SVM training is based on the minimization of the structural risk, i.e., the minimization of the upper bound of the generalization error. Therefore, SVM models have their complexity implicitly controlled. Another important difference is related to the nonlinearity of the objective functions. While for error backpropagation local minima can be troublesome, in SVM training the solution is unique, due to the quadratic nature of the optimization problem.

3 Regularization Techniques

There are two basic procedures to control an MLP extent of nonlinearity. The first one is called structure stabilization [8], where the aim is to determine the minimum number of neurons in the hidden layer, which can be implemented in three ways: comparison of different structures using resampling or analytical qualification (e.g., *NIC (Network Information Criterion)* [9], Bayesian model comparison [10], etc.); pruning [8]; and constructive networks [13].

The second basic procedure for controlling the neural network complexity is based on regularization theory. In this procedure, a balance between training error and generalization capacity can be obtained through the minimization of the total risk, i.e.:

$$\min\left\{R(\underline{w}) = E_s(\underline{w}, D) + \lambda E_c(\underline{w})\right\} \qquad (14)$$

In equation (14), $E_s(\underline{w}, D)$ represents the empirical risk, given by (2), while $E_c(\underline{w})$ denotes the model complexity [8], [10], [13]. The factor λ is known as the regularization parameter, which weights the balance between training error and model complexity. The setting of the regularization parameter, associated with the bias-variance trade-off, is performed by resampling techniques and/or analytical methods [10].

One way to define the functional form of $E_c(\underline{w})$ is through the application of Bayesian inference [8], [10]. Using Bayes' rule, the conditional probability density function of $\underline{w}$, given the dataset *D*, $p(\underline{w} \mid D)$, is estimated by:

$$p(\underline{w} \mid D) = \frac{p(D \mid \underline{w})\, p(\underline{w})}{p(D)} \qquad (15)$$

In equation (15), $p(D \mid \underline{w})$ is the likelihood of *D* given $\underline{w}$, $p(\underline{w})$ is the a priori probability density function of $\underline{w}$, and $p(D) = \int p(D \mid \underline{w})\, p(\underline{w})\, d\underline{w}$ is a normalization factor, which guarantees that $\int p(\underline{w} \mid D)\, d\underline{w} = 1$.

It is assumed that $\underline{w}$ presents a Gaussian distribution with zero mean and diagonal covariance matrix $\alpha^{-1}\underline{\underline{I}}$, and that the desired outputs are given by $d_i = f(\underline{x}_i, \underline{w}) + \zeta$, where ζ is Gaussian white noise with zero mean and variance β^{-1}. With the previous hypotheses, the maximization of the a posteriori distribution of $\underline{w}$, i.e., $p(\underline{w} \mid D)$, is equivalent to the minimization of the following expression [10]:

$$S(\underline{w}) = \frac{\beta}{2}\sum_{i=1}^{N}\left[f(x_i, \underline{w}) - d_i \right]^2 + \frac{\alpha}{2}\sum_{i=1}^{M} w_i^2 \qquad (16)$$

Dividing (16) by β, $S(\underline{w})$ becomes equal to $R(\underline{w})$, therefore,

$$E_c(\underline{w}) = \frac{1}{2}\|\underline{w}\|^2 \qquad (17)$$

This regularization term is known as “weight decay”, which favors neural models with small magnitudes for the connection weights.

One of the advantages of the Bayesian approach for training an MLP is the embedded iterative mechanism for estimating λ [10].

As already mentioned, there are other techniques for estimating neural models with good generalization capacity, in which complexity control is not attained by adding $\lambda E_c(\underline{w})$ in the objective function to be minimized. Among these techniques the early stopping procedure and the activation function gain scaling method [14] deserve more investigation.

In the early stopping procedure, cross validation is usually applied to partitioning the data into two subsets: one for training and the other for validation. The iterative updating of the weight $\underline{w}$ is interrupted during training as soon as the error for the validation subset stops decreasing. Although very popular in practice, [13], [15], this procedure is very heuristic. Moreover, it can deteriorate the extraction of information related to serial correlation in the load curve. Other drawbacks of early stopping can be seen in [11], [12].

The activation function gain scaling method is a post-training method equivalent to inserting noise in the training patterns (without doing that explicitly). The idea of including corrupted versions of the original input patterns in the training set is to smooth the mapping, avoiding divergent outputs for similar inputs [14].

Let’s consider neural networks with one hidden layer of sigmoidal activation functions and one linear output unit. According to [14], if the training data is generated so that the input space is uniformly covered and the additive noise is Gaussian distributed, i.e., $N(\underline{0}, \sigma^2_{noise}\underline{\underline{I}})$, then one network trained with noise to minimize the empirical risk will have a performance comparable to another neural network estimated with the original (non-corrupted) dataset, but with regularization. Similar generalization capacity can be obtained with a neural network trained on the original dataset and aiming at minimizing the empirical risk, i.e., without regularization, if the activation functions gains from the hidden units are multiplied by a factor a_k, such that:

$$a_k = \frac{1}{\sqrt{\|\underline{w}_k\|^2 \sigma^2_{noise} + 1}} \qquad (18)$$

$$\underline{w}_k = \left[w_{k1}, w_{k2}, ..., w_{kn} \right]^t,\ k = 1, 2, ..., m$$

This procedure produces analogous results to the weight decay regularization method [14].

4 Data Pre-processing

Data pre-processing is essential for the success of any system identification method. Besides estimating the missing data, the data have been transformed in order to improve the time series stationarity. Therefore, seasonalities and trends have been removed from the load and temperature series. Missing data have been estimated by linear interpolation. Seasonal behavior has been removed by seasonal differencing, while trends have been mitigated with first order differencing, i.e.:

$$Series1(k) = Series(k) - Series(k-1) \quad (19)$$
$$Series2(k) = Series1(k) - Series1(k-24)$$
$$Series3(k) = Series2(k) - Series2(k-168)$$

The load series "Series3" has been generated from October 30th. 1990 to December 31st 1991. This series has been standardized (zero mean and unit variance) also, producing the series $S(k)$. The sample autocorrelation functions, including the partial one, of $S(k)$ have been estimated in order to select the input variables for the neural NARX *(*Nonlinear Autoregressive Exogenous) model. From the correlograms, the most significant load variables have been *S(k-1), S(k-2), S(k-24)* and *S(k-168).*

After removing the daily seasonality and applying standardization to the temperature series, temperature lagged variables were added for every time instant for which a load variable was included, plus the forecasted temperature for the target hour, i.e., *T(k), T(k-1), T(k-2), T(k-24)* and *T(k-168).* Measured temperatures have been employed as temperature predictions for the next day.

Besides those input variables, two other variables for coding the hour of the day have been used [15], [16]:

$$HS(k) = \sin\left(\frac{2\pi k}{24}\right);\ HC(k) = \cos\left(\frac{2\pi k}{24}\right) \quad (20)$$
$$k = 1, 2, ..., 24$$

5 Load Forecasting Models

As the load dynamics varies from day to day, one model has been developed for each day of the week (holidays have not been considered). The training set for each model is formed by the corresponding data from the six weeks before the 24-hour forecasting horizon. For example, the forecasting model for Tuesdays uses a training set with 144 patterns. The forecasters are retrained at the end of each day to incorporate the most recent load information. Fig. 2 presents a diagram showing the procedure for building the training sets and the corresponding forecasting horizon.

TRAINING DATASET						FORECASTING
5/17/90	5/24/90	5/31/90	6/7/90	6/14/90	6/21/90	6/28/90
24 PATTERNS	24 PATTERNS	24 PATTERNS	24 PATTERNS	24 PATTERNS	24 PATTERNS	24 PATTERNS

	TRAINING DATASET						FORECASTING
	5/24/90	5/31/90	6/7/90	6/14/90	6/21/90	6/28/90	7/5/90
	24 PATTERNS	24 PATTERNS	24 PATTERNS	24 PATTERNS	24 PATTERNS	24 PATTERNS	24 PATTERNS

Fig. 2 – Training and forecasting for each model

For the MLP, hyperbolic tangents have been used as activation functions of the hidden layer units. For the SVMs, Gaussian kernels have been employed, requiring the specification of σ_{kernel}.

Test results in the next section compare the following training methods:

conventional error backpropagation (minimization of the empirical risk;
backpropagation followed by gain scaling;
Bayesian training; and
SVM learning.

In order to establish a basic benchmark, ARX (Auto-Regressive Exogenous) linear models have been tested too.

6 Results

Test results for all days of the week, with the different models and methods, are shown in Tables I, II and III. In Table I, the models' structures and training parameters that have provided the best results are presented. The first three columns in Table I show the number of neurons in the hidden layer of the MLP for each day of the week, and the corresponding training method. The last column in this Table shows the average number of Support Vectors (ANSV) for each day of the week.

In Table II, a comparison among the mean absolute percentage errors is performed. Its last column shows the differences (in %) between the best and the worse results for each day of the week. The outstanding performance of the SVMs is remarkable, considering the lack of experience on their application to regression problems. It is curious to verify that for Thursday and

Fridays, when the Bayesian training has been the winner, the number of support vectors (ANSV) has been much less than in the other days of the week. (Table I). More research is needed to clarify this problem. Table III is analogous to Table II, but contains maximum percentage errors.

Table I - Model Structure and Training Parameters

	Without regularizer	*Bayesian Training*	Gain Scaling		SVM			
	Neurons	Neurons	Neurons	σ^2_{noise}	C	ε	σ_{kernel}	ANSV
Monday	2	2	2	0.12	0.1	0.100	4.24	57.3
Tuesday	2	2	2	0.16	1.0	0.001	4.24	140.6
Wednesday	3	2	3	0.07	1.0	0.100	2.72	58.7
Thursday	3	2	2	0.17	1.0	0.400	1.96	12.8
Friday	2	2	2	0.12	1.0	0.400	3.48	11.7
Saturday	4	2	4	0.05	0.1	0.001	5.00	143.4
Sunday	2	2	2	0.11	1.0	0.100	1.96	59.1

Table II – Comparison Among Different Models (Error Percentage)

	ARX	Without regularizer	*Bayesian training*	Gain Scaling	SVM	Performance Gain
Monday	8.23	8.76	6.43	7.00	5.23	40.3
Tuesday	8.16	7.04	7.47	6.52	4.97	39.1
Wednesday	8.15	6.94	7.08	6.18	5.00	38.7
Thursday	8.15	10.21	6.93	8.41	7.83	32.1
Friday	9.54	7.33	6.29	7.18	6.39	34.1
Saturday	7.38	9.57	7.38	8.17	5.24	45.3
Sunday	7.02	8.19	6.91	7.42	5.00	38.9

7 Conclusion

This paper has compared three techniques to control the complexity of neural network models for short-term load forecasting. All tested techniques have shown better results than the ones provided by the non-regularized neural networks (Table II).

Table III – Comparison Among Different Models (Maximum Error Percentage)

	ARX	Sem Regularizador	Bayesiano	Escalonamento	SVM	Ganho de Desempenho
Segunda	59.08	57.36	42.54	53.95	43.11	28.00
Terça	66.58	37.44	70.24	34.38	36.62	51.05
Quarta	87.13	164.04	171.62	53.74	30.97	81.95
Quinta	60.15	95.05	57.09	80.28	16.01	83.16
Sexta	131.26	89.71	101.08	64.13	15.31	88.34
Sábado	47.79	92.99	55.83	57.66	32.34	65.22
Domingo	68.65	287.78	101.94	94.44	19.39	93.26
Média	74.38	117.77	85.76	62.65	27.68	76.50

It is worth mentioning the superior performance of the Support Vector Machines, which have presented lower errors for all days of the week, except Thursdays and Fridays. Despites this superior performance, it is expected that the SVM forecasting errors would be even lower if a better search for the optimal parameters that define the SVM, i.e., the width σ_{kernel} of the Gaussian *kernel* $K(\underline{x},\underline{x}_i)$, C and ε, were conducted.

The activation function gain scaling procedure, although behind SVM and Bayesian training in the competition, is by far the simplest and most computationally efficient technique. It is the only one capable of improving a trained neural network without restarting from scratch. On the other hand, in the Bayesian training, the regularization parameter λ estimation is part of the method, allowing full use of the dataset and less intervention from the user. Although not exploited in this work, the Bayesian training is also capable of selecting significant input variables (possibly better than the ones selected by linear correlograms) and estimating confidence intervals.

References

A.S. Debs, *Modern Power Systems Control and Operation*, Kluwer Academic Publishers, 1988.

D.C. Park, M.A. El-Sharkawi, R.J. Marks II, "An Adaptively Trained Neural Network", *IEEE Transactions on Neural Networks*, v.2, n.3, pp. 334-345, May 1991.

A.Khotanzad, R. Afkhami-Rohani, D. Maratukulam, "ANNSTLF – Artificial Neural Network Short-Term Load Forecaster – Generation Three", *IEEE Transactions on Power Systems*, v.13, n.4, pp. 1413-1422, Nov. 1998.

S. Osowski, K. Siwek, "Regularization of Neural Networks for Improved Load Forecasting in the Power System", *IEE Proceedings on Generation, Transmission and Distribution*, v.149, n.3, pp. 340-344, May 2002.

V.N. Vapnik; *Statistical Learning Theory*, New York, John Wiley & Sons, 1998.

V. Cherkassky, F. Mulier, *Learning from Data - Concepts, Theory and Methods*, John Wiley & Sons, New York, USA, 1998.

B.Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond, Cambridge, Massachusetts, 2002.

C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

N. Murata, S. Yoshizawa, S.I. Amari, "Network Information Criterion – Determining the Number of Hidden Units for an Artificial Neural Network Model", *IEEE Transactions on Neural Networks*, v.5, n.6, 865-872, 1994.

D.J.C. Mackay, *Bayesian Methods for Adaptive Models*, Ph.D. dissertation, California Institute of Technology, Pasadena, California, USA, 1992.

S. Amari, N. Murata, K.R. Müller, M. Finke, H. Yang, "Statistical Theory of Overtraining – Is Cross-validation Asymptotically Effective?", *Advances in Neural Information Processing Systems*, v.8, pp. 176-182, 1996.

Z. Cataltepe, Y.S. Abu-Mostafa, M. Magdon-Ismail, "No Free Lunch for Early Stopping", *Neural Computation*, v.11, n.4, pp. 995-1009, May 1999.

N.K. Treadgold, T.D. Gedeon, "Exploring Constructive Cascade Networks", *IEEE Transactions on Neural Networks*, v.10, n.6, 1335-1350, 1999.

R. Reed., R.J. Marks II, S. Oh, "Similarities of Error Regularization, Sigmoid Gain Scaling, Target Smoothing and Training with Jitter", *IEEE Transactions on Neural Networks*, v.6, n.3, 529-538, 1995.

A.P. Alves da Silva, L.S. Moulin, "Confidence Intervals for Neural Network Based Short-Term Load Forecasting", *IEEE Transactions on Power Systems*, v.15, n.4, 1191-1196, 2000.

A.J.R. Reis, A.P. Alves da Silva, "Feature Extraction Via Multi-Resolution Analysis for Short-Term Load Forecasting", *IEEE Transactions on Power Systems*, v.20, issue 1, pp. 189-198, Feb. 2005.