Toward Estimating Autonomous Neural Network-Based Electric Load Forecasters

Vitor Hugo Ferreira and Alexandre P. Alves da Silva, *Senior Member, IEEE*

***Abstract*—Anticipation of load's future behavior is very important for decision making in power system operation and planning. During the last 40 years, many different load models have been proposed for short-term forecasting. After 1991, the literature on this subject has been dominated by neural network (NN) based proposals. This is mainly due to the NNs' capacity for capturing the nonlinear relationship between load and exogenous variables. However, one major risk in using neural models is the possibility of excessive training data approximation, i.e., overfitting, which usually increases the out-of-sample forecasting errors. The extent of nonlinearity provided by NN-based load forecasters, which depends on the input space representation, has been adjusted using heuristic procedures. Training early stopping based on cross validation, network pruning methods, and architecture selection based on trial and error are popular. The empirical nature of these procedures makes their application cumbersome and time consuming. This paper develops two nonparametric procedures for solving, in a coupled way, the problems of NN structure and input selection for short-term load forecasting.**

***Index Terms*—Bayes procedures, feedforward neural networks (NNs), input selection, load forecasting, model complexity, support vector machines (SVM).**

I. Introduction

OPERATIONAL decisions in power systems, such as unit commitment, economic dispatch, automatic generation control, security assessment, maintenance scheduling, and energy commercialization depend on the future behavior of loads. Therefore, several short-term load forecasting methods have been proposed during the last four decades. Such a long experience in dealing with the load forecasting problem has revealed some useful models such as the ones based on multilinear regression, Box–Jenkins method, artificial neural networks (ANNs) [1], fuzzy systems, and hybrid models. However, autonomous load forecasters, i.e., automatic input selection and model complexity control, are still needed to avoid expert intervention and to extend the application to the bus load level [2].

The relationship between electric load and its exogenous factors is complex and nonlinear, making it quite difficult to be modeled through conventional techniques such as linear time series and regression analyses. Classical methods are bias-prone, i.e., they are based on theoretical guesses about the underlying laws governing the system under study. On the other hand, after some years of practical experience, it has been recognized that ANNs can provide superior forecasting performance when dealing with nonlinear and multivariate problems involving large data sets, such as short-term load prediction. ANNs have more flexible functional forms in which there are few *a priori* assumptions about the relationships between input and output variables.

Although usually more robust than traditional load forecasting models, ANNs have overcome several problems in order to become commercially successful [3]. Since the first proposals of ANN-based load forecasters [4], five major drawbacks have been tackled: heavy training burden, lack of prediction interval estimation, inference opacity, input space representation, and model complexity control.

Fast training algorithms have been developed since the early nineties [5], which have allowed the tracking of load nonstationarities. On the other hand, sometime has passed until the recognition of the practical importance of prediction interval estimation [6]. Qualitative interpretations of the ANN's forecasts have been proposed in [7] and [8]. It seems that improvement on forecasting accuracy provided by ANNs cannot come without degrading model transparency. The ANN inference lack of interpretability can be mitigated using auxiliary tools such as the one described in [9]. However, it is hard to achieve a level of interpretability comparable to the one extractable from linear models.

The last two drawbacks are critical for short-term load forecasting, although they have not received much attention. The ANN input representation and complexity control should not be treated separately, as it is common practice in load forecasting. The extent of nonlinearity required from an ANN is strongly dependent on the selected input variables. One of the advantages of NN models is the universal approximation capability, i.e., unlimited precision for continuous mapping. However, this theoretical advantage can backfire if data overfitting is not avoided [10]. The main objective of model complexity control is to match data regularity with model structure, maximizing the generalization capacity.

A popular procedure for ANN complexity control is based on cross validation (CV) with training early stopping, i.e., the iterative updating of the connection weights until the error for the validation subset stops decreasing. This procedure is very heuristic, because it is not easy to detect the right iteration for interrupting the training process. Besides, although CV has been successfully applied to neural classifiers design, serial correlation information can be lost when it is used in time series forecasting. Shortcomings of CV and early stopping are fully analyzed in [11] and [12].

Manuscript received February 27, 2007; revised June 4, 2007. This paper was supported by the Brazilian Research Council (CNPq) and by the State of Rio de Janeiro Research Foundation (FAPERJ). Paper no. TPWRS-00414-2007.

The authors are with the COPPE/UFRJ, Electrical Engineering Graduate Program, Power Systems Laboratory, Rio de Janeiro, RJ, 21945-972, Brazil (e-mail: vitor@vishnu.coep.ufrj.br; alex@coep.ufrj.br).

Digital Object Identifier 10.1109/TPWRS.2007.908438

Input space representation is probably the most important subtask in load forecasting. It has been shown that input variable selection based on linear auto- and cross correlation analyses is not appropriate for nonlinear models such as ANNs. Feature extraction via multiresolution analysis, based on wavelets, has been proposed to overcome this problem [13]. However, a more ANN oriented input selection scheme is still needed to capture the important information about the linear and nonlinear interdependencies in the associated multivariate data.

This paper develops two methods based on some of the most suitable techniques for controlling ANN complexity, with simultaneous selection of appropriate explanatory input variables for short-term load forecasting. In order to automatically minimize the out-of-sample prediction error, Bayesian training [14], [15] and support vector machine (SVM) learning [16], [17] are investigated. These training methods include complexity control terms in their objective functions, which allow autonomous modeling and adaptation. An after-training complexity adjustment procedure, based on activation function gain scaling [18], is evaluated because of its simplicity.

Preliminary results on the applicability of Bayesian training and SVMs to short-term load forecasting have been reported in [19]. In this paper, several open questions are answered. State-of-the-art nonparametric regression tools are extended in this work to fulfill the requirements of the problem of interest. In Bayesian training, assumptions of different priors for load and weather related input variables are considered. Specific learning parameters for each input are also employed in [20]. However, their estimation is performed by genetic algorithms with CV-based fitness function. Here, CV is avoided in Bayesian and SVM training with the development of automatic analytical procedures for selecting among possible input variables and ANN structures.

The Bayesian approach has been fully exploited for the first time in load forecasting. A new procedure for determining useful inputs has been developed for avoiding the predetermination of significance thresholds. For the first time, support vector regression learning parameters are estimated along with the kernel parameters without CV, in contrast to recently proposed models [21]–[25].

Three databases have been used for testing. The first one corresponds to the load and temperature series, on an hourly basis, from a North American electric utility [4], [26], which has been used in load forecasting competitions. The second database is related to a daily peak load forecasting competition, with load and temperature data from the Eastern Slovakian Electricity Corporation [25]. The last dataset contains half-hourly loads, temperatures, and prices from the electricity market management company in Australia [27].

These internet-based datasets have been employed to allow reproduction of the results presented in this paper. Considering the intended reader, the paper is written to make the theoretical parts (Sections II–IV) as self contained as possible. Special emphasis is given to the important aspects in short-term load forecasting. Section V presents results and Section VI concludes with recommendations.

II. ANN Complexity Control

NN models commonly used in load forecasting have a feedforward structure with one hidden layer only [e.g., multilayer perceptrons (MLPs), radial basis functions (RBFs)]. In order to introduce the adopted nomenclature, this section describes the general structure of a feedforward ANN, with one hidden layer and one output neuron, under supervised learning.

Let $\underline{x} \in \Re^n$ be a vector representing input signals and $\underline{w} \in \Re^M$ the vector with the ANN connection weights, where $M = mn+2m+1$ and m is the number of neurons in the hidden layer. The biases of the hidden neurons sigmoidal activation functions are represented by $b_k, k = 1, 2, \ldots, m$, while b stands for the bias of the output neuron linear activation function. The final mapping is

$$y = f(\underline{x}, \underline{w}) = \sum_{k=1}^{m} (w_k c_k) + b \tag{1}$$

where

$$c_k = \varphi\left(\sum_{i=1}^{n} (w_{ik} x_i) + b_k\right).$$

Given a dataset U with N input/output pairs, $U = \{X, D\}$, for $X = (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N)$ and $D = (d_1, d_2, \ldots, d_N)$, where $d_j \in \Re$ represents the desired outputs, the ANN training objective usually is the estimation of the weight vector $\underline{w}$ such that the empirical risk (training error) is minimized, i.e.,

$$\min_{\underline{w}} \left\{ E_s(\underline{w}, U) = \frac{1}{2} \sum_{j=1}^{N} [d_j - f(\underline{x}_j, \underline{w})]^2 \right\}. \tag{2}$$

There are several algorithms for minimizing (2). Independently of using the classical error backpropagation, or second order methods, such as the Levenberg–Marquardt [13], or any other training method, the main drawback of this unconstrained training criterion is the absence of any concern regarding model complexity.

There are two basic approaches to control the ANN extent of nonlinearity. The first one is called structure stabilization, in which the objective is to determine the minimum number of neurons in the hidden layer. This approach can be implemented by comparing different structures using pruning or growing procedures [28], via CV or analytical estimation of model complexity (e.g., Vapnik Chervonenkis (VC) bounds [16] and NIC [29]).

Support vector machines (Section IV) belong to the structure stabilization approach. SVM learning is based on the minimization of the structural risk, i.e., the minimization of upper bounds on the generalization error (VC bounds), which hold with high confidence. Therefore, an SVM has its complexity implicitly controlled, with the model structure being a byproduct of training.

The second basic approach for controlling the NN complexity is based on regularization theory, in which analytical methods adjust the ANN extent of nonlinearity without necessarily changing the model structure. Section III presents two methods related to this approach: gain scaling [18] and Bayesian training [14].

III. Regularization Techniques

Regularization theory shows how generalization behaves as the number of examples tends to infinity. A balance between

training error and generalization capacity is obtained through the minimization of the total risk

$$\min_{\underline{w}}\{R(\underline{w}) = E_s(\underline{w}, U) + \lambda E_c(\underline{w})\}. \tag{3}$$

In (3), $E_s(\underline{w}, U)$ denotes the empirical risk, given by (2), while $E_c(\underline{w})$ estimates the model complexity. The factor λ is known as the regularization parameter, which weights the bias-variance tradeoff, i.e., training error versus model complexity. The setting of the regularization parameter λ is performed via resampling or by Bayesian estimation.

A. Gain Scaling Method

The activation function gain scaling method [18] is a post-training method equivalent to inserting noise in the training patterns (without doing that explicitly). The motivation for including corrupted versions of the original input patterns in the training set is to smooth the functional mapping, avoiding divergent outputs for similar inputs. Similar generalization capacity can be obtained with an ANN trained to minimize the empirical risk on the original dataset (i.e., without the corrupted patterns) if, after training, the gains (transition region slope) from the hidden neurons sigmoidal activation functions are multiplied by, a_k, i.e.,

$$c_k = \varphi\left[a_k\left(\sum_{i=1}^{n}(w_{ik}x_i) + b_k\right)\right] \tag{4}$$

where

$$a_k = \frac{1}{\sqrt{||\underline{w}_k||^2\sigma^2_{\text{noise}} + 1}}$$
$$\underline{w}_k = [w_{1k}, w_{2k}, \ldots, w_{nk}]^t, \quad k = 1, 2, \ldots, m.$$

The parameter $\sigma^2_{\text{noise}} \in \Re^+$, associated with the variance of the noise "added" to the training input patterns, is estimated via CV. For σ^2_{noise} different from zero, the ANN model provides less nonlinearity because the slopes of the activation functions are decreased (increasing their linear segments). In this work, the ANNs trained by backpropagation have served for the gain scaling procedure as a starting point.

B. Bayesian ANN Training

One way to define the functional form of $\lambda E_c(\underline{w})$, in (3), is through the application of Bayesian inference. Using Bayes' rule, the conditional probability density function (PDF) of, $\underline{w}$ given a dataset $U, p(\underline{w}|D, X)$, is estimated by

$$p(\underline{w}|D, X) = \frac{p(D|\underline{w}, X)p(\underline{w}|X)}{p(D|X)}. \tag{5}$$

Since X is conditioning all probabilities in (5), it will be omitted from this point on. Therefore, in (5), $p(D|\underline{w})$ is the likelihood of D given $\underline{w}, p(\underline{w})$ is $\underline{w}$'s *a priori* PDF, and $p(D) = \int p(D|\underline{w})p(\underline{w})d\underline{w}$ is enforcing $\int p(\underline{w}|D)d\underline{w} = 1$.

It is initially assumed that $\underline{w}$ presents a Gaussian distribution with zero mean and diagonal covariance matrix equal to $\alpha^{-1}\underline{\underline{I}}$, where $\underline{\underline{I}}$ is the $M \times M$ identity matrix, i.e.,

$$p(\underline{w}) = \frac{1}{Z_{\underline{w}}(\alpha)}e^{-\left(\frac{\alpha}{2}||\underline{w}||^2\right)} \text{ where } Z_{\underline{w}}(\alpha) = \left(\frac{2\pi}{\alpha}\right)^{\frac{M}{2}}. \tag{6}$$

The desired outputs can be represented by $d_j = f(\underline{x}_j, \underline{w}) + \zeta_j$, where ζ is Gaussian white noise with zero mean and variance equal to β^{-1}. The regularization factors α and β (learning parameters, also called hyperparameters), on the contrary of the other regularization techniques, are estimated along with the model parameters $\underline{w}$. Considering the previous hypotheses and assuming that the dataset patterns are independent, then

$$p(D|\underline{w}) = \frac{e^{\left\{-\frac{\beta}{2}\sum_{j=1}^{N}[d_j - f(\underline{x}_j, \underline{w})]^2\right\}}}{Z_Y(\beta)}$$

where

$$Z_Y(\beta) = \left(\frac{2\pi}{\beta}\right)^{\frac{N}{2}}. \tag{7}$$

Consequently, based on (5)

$$p(\underline{w}|D) = \frac{e^{[-S(\underline{w})]}}{\int e^{-S(\underline{w})}d\underline{w}} \tag{8}$$

where

$$S(\underline{w}) = \frac{\beta}{2}\sum_{j=1}^{N}[d_j - f(\underline{x}_j, \underline{w})]^2 + \frac{\alpha}{2}\sum_{l=1}^{M}w_l^2. \tag{9}$$

Therefore, the maximization of the *a posteriori* distribution of $\underline{w}, p(\underline{w}|D)$, is equivalent to the minimization of $S(\underline{w})$ [14]. Dividing $S(\underline{w})$ by β and making $\lambda = \alpha \div \beta$ in (3), the equivalence between $S(\underline{w})$ and $R(\underline{w})$ can be verified if

$$E_c(\underline{w}) = \frac{1}{2}||\underline{w}||^2. \tag{10}$$

The regularization term in (10), known as *weight decay*, favors neural models with small magnitudes for the connection weights. Small values for the connection weights tend to propagate the input signals through the almost linear segment of the sigmoidal activation functions. Notice that the requirement of prior information in Bayesian training is the primary instrument for controlling the ANN complexity.

One of the advantages of Bayesian training of an ANN is the embedded iterative mechanism for estimating λ, i.e., α and β, which avoids CV. For multivariate problems such as load forecasting, the use of one single hyperparameter α for dealing with all connection weights is not recommended. Load and weather related input variables, such as temperature, require different priors. Even among the same type of variables, different levels of interdependency are involved (e.g., $P(k)$ against $P(k+1)$ and *P(K-23)* against $P(k+1)$, for an hourly basis load).

In this work, each group of connection weights directly related to an input variable receives a different α_i. The same idea is applied to the groups of weights associated with the biases (one α_i for the connections with the hidden neurons and another for the output neuron connection). One last α_i is associated with all connection weights between the hidden and output layers. Therefore, for n-dimensional input vectors $\underline{x}$, the total number of α_is is $n + 3$.

1) Input Selection in Bayesian Training: For a given model structure, the magnitudes of the α_is can be compared to determine the relevance of the corresponding input variables (taken

from a predefined set). As $p(\underline{w}_i)$ is supposed to be normally distributed with zero mean and $\alpha_i^{-1}\underline{\underline{I}}$ covariance, then, the largest α_i s lead to the smallest $\underline{w}_i$ s. For estimating the *a posteriori* PDF of $\underline{w}$, Bayesian training combines the *a priori* PDF with the information provided by the training set (5). If an α_i is large, the prior information about $\underline{w}_i$ is almost certain, and the effect of the training data on the estimation of $\underline{w}_i$ is negligible. Another way to see the influence of α_i on $\underline{w}_i$ is through (9).

The impact on the output caused by input variables with very small $\underline{w}_i$s, i.e., very large α_is, is not significant. However, a reference level for defining a very large α_i has to be established. For short-term load forecasting, two different references of irrelevance are needed: one reference for continuous variables, such as loads and temperatures, and another for dummy variables, such as hours of the day and days of the week. Uniformly distributed input variables can be employed to define the references of irrelevance [30]. For continuous input variables, an uniform random variable with lower and upper limits equal to $-\sqrt{3}$ and $\sqrt{3}$, respectively, is used as reference of irrelevance, since continuous variables have been standardized (zero mean and unit variance). For dummy variables, the reference is a binary random variable with uniform distribution. These two reference variables are added to the predefined set of inputs.

After training the model with the predefined set of input variables, continuous, and dummy variables are separately ranked. For each rank, the variables with corresponding α_is larger than α_{ref} (irrelevance level) are disregarded. After input selection, the ANN is retrained with the selected variables.

2) Structure Selection in Bayesian Training: Bayesian inference can also be employed to determine the best structure among a predefined set of possibilities, e.g., $\mathrm{H} = \{H_1, H_2, \ldots, H_K\}$, for which the corresponding inputs have been previously selected, i.e.,

$$P(H_h|D) = \frac{p(D|H_h)P(H_h)}{p(D)}. \tag{11}$$

In (12), $p(H_h)$ represents the *a priori* probability of model H_h and $p(D|H_h)$ is given by

$$p(D|H_h) = \iint p(D|\alpha, \beta, H_h)p(\alpha, \beta|H_h)d\alpha d\beta. \tag{12}$$

Using Gaussian approximation around the estimated hyperparameters (from training), analytic integration of (12) is possible, leading to (13)

$$\begin{aligned}\ln p(D|H_h) &= -S(\underline{w}) - \frac{1}{2}\ln|\nabla\nabla S(\underline{w})| \\ &\quad + \frac{1}{2}\sum_{i=1}^{n+3} M_i\alpha_i + \frac{N}{2}\ln\beta + \ln(m!) \\ &\quad + 2\ln m + \frac{1}{2}\sum_{i=1}^{n+3}\ln\left(\frac{2}{\gamma_i}\right) + \frac{1}{2}\ln\left(\frac{2}{N-\gamma}\right)\end{aligned} \tag{13}$$

where m denotes the number of hidden neurons in the ANN model H_h. Since all models, *a priori*, are assumed equally probable, H_h is selected by maximizing $P(D|H_h)$, which is equivalent to maximizing $\ln p(D|H_h)$. Consequently, (13) can be used for ranking and selecting among MLPs with different numbers of neurons in the hidden layer.

3) Extended Bayesian Training Algorithm: The following steps describe the ANN structure and input selection via Bayesian inference.

Step 1) Set the minimum ($N_{\min}$) and maximum ($N_{\max}$) number of neurons in the hidden layer. In this work, $N_{\min} = 1$ and $N_{\max} = 10$.

Step 2) Make the number of neurons in the hidden layer $m = N_{\min}$.

Step 3) Add the reference of irrelevance variables to the user defined n-dimensional input vector. If dummy variables are used, the input set will contain $n = n + 2$ input variables. Otherwise, i.e., if only continuous inputs are preselected, $n = n + 1$.

Step 4) Set $l = 0$ and initialize

$$\underline{w}(l) = [\underline{w}_1(l), \ldots, \underline{w}_{n+3}(l)]^t, \underline{\alpha}(l) = [\alpha_1(l), \ldots, \alpha_{n+3}(l)]^t,$$

and $\beta(l)$.

Step 5) Minimize $S(\underline{w})$ on $\underline{w}(l)$ to obtain $\underline{w}(l+1)$.

Step 6) Calculate $\alpha_i(l+1), \beta(l+1)$, and $\gamma_i(l+1)$ using the following equations:

$$\begin{aligned}\nabla\nabla S(\underline{w})|_{\underline{w}=\underline{w}(l+1)} &= \beta(l)\,\nabla\nabla E_s(\underline{w}, U)|_{\underline{w}=\underline{w}(l+1)} \\ &\quad + \nabla\nabla E_c(\underline{w})|_{\underline{w}=\underline{w}(l+1)} \\ \underline{\underline{B}}_i(l+1) &= [\nabla\nabla S(\underline{w})|_{\underline{w}=\underline{w}(l+1)}]^{-1}\underline{\underline{I}}_i \\ \gamma_i(l+1) &= M_i - \mathrm{trace}\{\underline{\underline{B}}_i(l+1)\} \\ \alpha_i(l+1) &= \frac{\gamma_i(l+1)}{\|\underline{w}_i(l+1)\|^2} \\ \beta(l+1) &= \frac{N - \sum_{i=1}^{n+3}\gamma_i(l+1)}{\sum_{j=1}^{N}[d_j - f(\underline{x}_j, \underline{w}(l+1))]^2}.\end{aligned} \tag{14}$$

Step 7) Make $l = l + 1$ and return to Step 5) until convergence has been achieved. After convergence, go to the next step.

Step 8) Isolate in two lists the α_is associated with the continuous input variables and the α_js related to the dummy variables.

Step 9) For each list, select the inputs such that the corresponding $\alpha < \alpha_{\text{ref}}$, where α_{ref} stands for the hyperparameter associated with the added irrelevant input.

Step 10) Repeat Steps 4)–7) using the inputs selected in Step 9), with n equal to the number of selected variables, to obtain the trained model H_m.

Step 11) Evaluate the log evidence of the hypothesis (ANN structure) H_m using (13).

Step 12) If $m = N_{\max}$, then go to Step 13). Else, $m = m+1$ and return to Step 3).

Step 13) Select the H_k with the largest log evidence.

In (14), $\underline{\underline{I}}_i$ is an $M \times M$ diagonal matrix with ones at the positions corresponding to the ith group of weights and with zeros otherwise. M_i is the number of connection weights in each group. Details on how to calculate the Hessian $\nabla\nabla E_s(\underline{w}, U)$ can be found in [15].

IV. Support Vector Machines

In classification problems [31], maximum margin SVM classifiers are estimated to minimize the generalization error bounds. The training patterns that define the separation surface, based on which the maximum margin is obtained, are called support vectors. The other training patterns have no influence on the inference process.

In order to apply the same idea to regression problems, the concept of classification margin is adapted. A margin in regression means the amount by which the training and test accuracy can differ, i.e., different error functions are used for training and testing. During training, analogously to classification problems, an approximation error is not counted if it is inside a band of size $\pm\varepsilon$ [see (16)]. Any training point lying outside this band (support vectors) has its corresponding error taken into account.

As a linear machine on feature space, i.e., the space defined by a set of nonlinear basis functions $\underline{\phi}(\underline{x})$ that allows the model to produce nonlinear mappings on the original input space of $\underline{x}$, the SVM output is given by

$$y = \sum_{j=0}^{m} W_j \phi_j(\underline{x}) = \underline{W}^t \underline{\phi}(\underline{x}) \tag{15}$$

where

$$\underline{\phi}(\underline{x}) = [1, \phi_1(\underline{x}), \ldots, \phi_m(\underline{x})]^t$$

and

$$\underline{W} = [b, W_1, \ldots, W_m]^t.$$

The following ε-insensitive cost function is adopted here

$$L_\varepsilon(d, y) = \begin{cases} (|d-y| - \varepsilon)^2, & \text{for } |d-y| - \varepsilon \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

SVMs that use (16) as the error function are called L2-SVMs [32], in contrast with previously proposed SVM load forecasters (L1-SVMs), which use an ε -insensitive linear loss function. L2-SVMs have been employed in this work because they lead to differentiable analytical bounds for the generalization error. Such bounds cannot be derived for L1-SVMs. Then, the SVM hyperparameters can be directly estimated through mathematical programming techniques, avoiding CV.

In the following development, ε and c_0 are assumed to be known, i.e., defined by the user. This assumption will be removed later. The training objective of an SVM model is the following constrained minimization of the empirical risk:

$$\min_{\underline{W}} \left\{ E_s(\underline{W}, D) = \frac{1}{N} \sum_{i=1}^{N} L_\varepsilon(d_i, y_i) \right\} \tag{17}$$

subject to

$$\|\underline{W}\|^2 \leq c_0$$

where c_0 also affects the model complexity.

A. Support Vector Regression

The primal optimization problem formulated by (17) is transformed into its dual form, (18), to allow the incorporation of kernel functions, which avoid the requirement of knowing an appropriate $\underline{\phi}(\underline{x})$

$$\begin{aligned} \max_{\underline{\alpha}, \underline{\alpha}'} \Bigg\{ & Q(\underline{\alpha}, \underline{\alpha}') \\ & = \sum_{i=1}^{N} d_i(\alpha_i - \alpha_i') \\ & \quad - \varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i') - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_i') \\ & \quad \times (\alpha_j - \alpha_j') \left[K(\underline{x}_i, \underline{x}_j) + \frac{\delta_{ij}}{C} \right] \Bigg\} \end{aligned} \tag{18}$$

subject to

$$\begin{aligned} & \sum_{i=1}^{N} (\alpha_i - \alpha_i') = 0 \\ & \alpha_i \geq 0, \alpha_i' \geq 0, \quad i = 1, 2, \ldots, N. \end{aligned}$$

In (18), $K(\underline{x}_i, \underline{x}_j) = \underline{\phi}^t(\underline{x}_i)\underline{\phi}(\underline{x}_j)$ is the inner product kernel defined according to Mercer's theorem [16], δ_{ij} is the Kronecker delta function, and C is the regularization hyperparameter. Then, the output of an SVM is given by

$$y = f(\underline{x}, \underline{W}) = \sum_{i=1}^{N} (\alpha_i - \alpha_i') K(\underline{x}, \underline{x}_i). \tag{19}$$

As indicated in (19), the support vectors are the training patterns for which $\alpha_i \neq \alpha_i'$, i.e., the ones located outside the band defined by ε. In fact, an SVM model can be represented as a feedforward ANN model with hidden layer units activation functions defined by the kernel $K(\underline{x}, \underline{x}_i)$. Notice that an SVM model, depending on the adopted kernel function, has the MLP and the RBF as special cases, when the kernels are specified as sigmoid and Gaussian functions, respectively. However, an important difference compared with traditional training algorithms for MLPs and RBFs is related to the convexity of the corresponding objective functions. While for error backpropagation and clustering algorithms local minima can be troublesome, in SVM training the solution is unique due to the corresponding quadratic optimization problem.

B. SVM Input Selection

Reference [33] develops the concept of *span of support vectors*, from which a differentiable upper bound on the generalization error for regression is derived in [32]

$$T_{SB}[f(\underline{x}, \underline{w})] = \sum_{i=1}^{p} (\alpha_i + \alpha_i') \tilde{S}_i^2 + N\varepsilon \tag{20}$$

where α_i, α_i' are the Lagrange multipliers associated with the support vector $\underline{x}_i, p$ the number of support vectors, and

$$\tilde{S}_i^2 = \min_{\underline{\mu}} \|\underline{\tilde{\phi}}(\underline{x}_i) - \sum_{j=1, j\neq i}^{p} \mu_j \underline{\tilde{\phi}}(\underline{x}_j)\|^2 + \eta \sum_{j=1, j\neq i}^{p} \frac{\mu_j^2}{(\alpha_i + \alpha_i')} \quad (21)$$

subject to

$$\sum_{j=1, j\neq i}^{p} \mu_j = 1, \quad \text{for } \mu_j \in \Re$$

with η denoting a parameter responsible for promoting differentiability [$\eta = 0$ turns the objective function in (21) to a nondifferentiable one] and $\underline{\tilde{\phi}}(\underline{x}_j) = [\underline{\phi}(\underline{x}_j) \quad \underline{o}_j/\sqrt{C}]^t$ representing an extended feature space mapping, where $\underline{o}_j$ is an N-dimensional vector with the jth element equal to one and the other elements equal to zero.

The optimal solution for (21) is presented in [32], along with the partial derivatives of T_{SB} with respect to C, ε, and the kernel parameters. The minimization of (20), via gradient descent, is applied here to select inputs and the L2-SVM structure, which is determined not only by C, ε, and the kernel parameters, but also by the selected input variables. Therefore, extending the proposal in [32], the present work estimates the individual contributions of each input to the Gaussian kernel as a way to select input variables.

Input weights, σ_is, for measuring the significance of each preselected input variable (i.e., input space is scaled by $\sigma_i x_i$) can be associated with the kernel parameters. This can be verified by writing the Gaussian kernels as follows:

$$K(\underline{x}, \underline{y}) = e^{-\sum_{i=1}^{n} (\sigma_i x_i - \sigma_i y_i)^2} = e^{-\sum_{i=1}^{n} \sigma_i^2 (x_i - y_i)^2}. \quad (22)$$

A small scaling factor means that the corresponding input is not relevant. Therefore, such an input is disregarded. Similarly to Bayesian training, a reference for a small σ_i is needed. Therefore, an analogous procedure is applied to rank the predefined inputs and disregard the less significant ones. Afterwards, the L2-SVM is retrained with the selected inputs. Notice that the standard SVM Gaussian kernel uses $\sigma_i = \sigma$ for all input variables.

Due to the nonconvex nature of T_{SB}, gradient descent depends on initialization, which is hard to set because the learning parameters optima values can be very different in magnitude. This is also troublesome for determining the gradients, because the sensitivity to parameters varying in small magnitude ranges is jeopardized. Logarithmic transformations can be used to overcome this problem. Regarding gradient descent initialization, [34] derives useful expressions (23) for estimating C and ε, which are employed here to start the search. The σ_i values have been initialized at 0.1. The initial values for C and ε are

$$C_{\text{est}} = \max(|\overline{d} + 3s_d|, |\overline{d} - 3s_d|)$$
$$\varepsilon_{\text{est}} = 3s\sqrt{\frac{\ln N}{N}} \quad (23)$$

where

$$s = \sqrt{(N-n)^{-1} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}.$$

In (23), $\overline{d}$ is the sample mean for the target values, s_d is the corresponding standard deviation, and s is the standard deviation of the regression model error. In this paper, s is estimated from the residues of an auto regressive exogenous (ARX) linear model.

C. Automatic L2-SVM Learning

The proposed L2-SVM learning algorithm can be summarized as follows.

- Step 1) Add the reference of irrelevance variables to the user defined set of inputs (as in the extended Bayesian training).
- Step 2) Set $l = 0$ and initialize $C(l)$ and $\varepsilon(l)$, using (23). Initialize the scaling factors $\underline{\sigma}(l) = [\sigma_1(l), \ldots, \sigma_n(l)]^t$. In this work, all scaling factors are initially equal to 0.1.
- Step 3) Solve (18) to obtain $\underline{\alpha}, \underline{\alpha}'$.
- Step 4) Minimize $T_{SB}[f(\underline{x}, \underline{w})]$ via gradient descent to get $C(l+1), \varepsilon(l+1)$, and $\underline{\sigma}(l+1)$.
- Step 5) Make $l = l + 1$ and return to Step 3) until convergence has been achieved. After convergence, go to the next step.
- Step 6) Isolate in two lists the σ_is associated with the continuous input variables and the σ_js related to the dummy variables.
- Step 7) For each list, select the inputs such that the corresponding $\sigma > \sigma_{\text{ref}}$, where σ_{ref} denotes the hyperparameter associated with the added irrelevant input.
- Step 8) Repeat Step 3) using the inputs selected in Step 7) and the previously optimized hyperparameters [Step 4)] to obtain the final model.

V. Test Results

The three datasets are standardized. The first one, with hourly load and temperature values, available at *ee.washington.edu/class/555/el-sharkawi/index_files/Page3404.html*, contains data from January 1, 1985–March 31, 1991. In this case, the task is to forecast the hourly load, from 16 up to 40 h (steps) ahead for weekdays, and from 16 up to 80 h ahead for weekends. The test (out-of-sample) period goes from November 1, 1990–March 31, 1991. With training data from the month to be forecasted and from two months earlier, along with the data corresponding to the same "window" in the previous year, seven models are estimated, one for each day of the week. Around 650 patterns are used for each model.

As the initial set of inputs, the following variables are tested: 24 dummy variables codifying the hour of the day; lags $S(k-1), S(k-2), \ldots, S(k-6), S(k-24), S(k-25), \ldots, S(k-29), S(k-168), S(k-169), \ldots, S(k-173)$ for load, temperature and temperature square series; the temperature forecast for hour k and its square value, i.e., $T(k)$ and $T^2(k)$, respectively; the daily maximum temperature forecast and its square value, $T_{\max}(d)$ and $T_{\max}^2(d)$; and the daily maximum temperature for the previous day and its square value, $T_{\max}(d-1)$ and $T_{\max}^2(d-1)$. Therefore, a total of 84 initial inputs (including dummies) have been presented to the models for selection. The

output is the predicted hourly load $L(k)$. As weather services can provide quite precise forecasts for the horizons of interest, the true temperatures have been employed as "perfect" predictions. The forecasts up to 80 h ahead are provided by recursion, i.e., load forecasts feed inputs. The number of preselected inputs has been deliberately made big. The idea is to verify the ability of the training algorithms in identifying the most significant variables. So far, the best results (benchmark) for this database are presented in [26].

For the second database, with daily peak load and temperature values from January 1, 1997–January 31, 1999, available at *http://neuron.tuke.sk/competition*, the out-of-sample period for 31-step ahead predictions of daily peak load goes from January 1, 1999 up to January 31, 1999. To avoid recursion, 31 models are estimated, one for each step ahead, using all data until January 1, 1999 ($\approx$ 720 patterns per ANN). For the jth model, the initial inputs are related to the seven most recent daily peak load values, plus $j + 7$ lagged temperature variables, and 19 dummy variables, seven for the days of the week and twelve for the months. Therefore, a total of $33 + j$ initial inputs (including dummies) have been presented to each model for selection. The lags for the load and temperature variables are $L(d-j), L(d-j-1), \ldots, L(d-(6+j))$ and $T(d), T(d-1), \ldots, T(d-(6+j))$, respectively. The model output is the daily peak load $L(d)$. As before, the true temperatures for the forecasting horizon are used as "predictions." The benchmark results for this database are presented in [25].

For the last database, at *www.nemmco.com.au*, with half-hourly load, price, and temperature values from December 4, 2001–December 31, 2003, the task is to forecast the hourly loads, from 1 up to 6 h ahead for several weeks in 2003. The series are transformed to an hourly basis by averaging two half hours. For any week to be forecasted, the corresponding training sets are built as for the first database (≈ 530 patterns per ANN). Six models are developed, one for each number of steps ahead, for each day of the week. The models for j steps ahead have the following predefined inputs: $19 - j$ lagged load, price and temperature variables, plus j temperature forecasts [i.e., $T(k), T(k-1), \ldots, T(k-j+1)$], and 24 dummy variables codifying the hour of the day, totalizing $81 - 2j$ inputs. The preselected lags are $S(k-j), S(k-j-1), \ldots, S(k-6), S(k-24), S(k-25), \ldots, S(k-29), S(k-168), S(k-169), \ldots$, and $S(k-173)$ for load, price and temperature. The output is the hourly load $L(k)$. The benchmark results for this database are presented in [27].

The previous datasets specifications have been applied to all training methods. Test results have been generated for the following training methods: conventional error back-propagation (minimization of the empirical error only); backpropagation (BP) followed by gain scaling; BP with gain scaling and structure selection (SS) via CV; BP with early stopping for regularization of a invariable structure; BP with early stopping for regularization with SS; the extended Bayesian training; L2-SVM learning with parameters estimated via CV; and the proposed L2-SVM learning. Furthermore, correlation-based input selection (CIS) has been tested in combination with gain scaling, early stopping, and L2-SVM with CV. Correlation analysis is used not just for selecting significant linear dependencies between possible inputs and output. It is also employed to eliminate redundant input variables. All dummy variables are preserved when CIS is applied. Different from the proposed methods, CIS does not depend on the ANN model.

TABLE I
COMPARISON AMONG DIFFERENT MODELS (MAPE)

	Case 1	Case 2	Case 3					
			1 step ahead	2 step ahead	3 step ahead	4 step ahead	5 step ahead	6 step ahead
Backpropagation	10.43	5.05	0.97	1.33	1.49	1.57	1.80	1.72
Gain Scaling	14.18	4.87	1.53	1.60	1.86	1.97	2.09	2.44
Gain Scaling with SS	13.76	2.19	1.53	1.68	1.94	1.81	2.26	2.50
Gain Scaling with CIS and SS	17.80	2.77	2.41	3.58	3.51	3.03	3.24	3.38
Early Stopping	8.07	1.95	2.04	1.93	2.09	2.60	2.00	2.35
Early Stopping with SS	7.11	2.13	1.61	1.44	1.49	1.57	1.78	1.46
Early Stopping with CIS and SS	11.41	2.87	2.14	2.26	2.39	2.27	2.27	2.22
Extended Bayesian Training	4.89	1.75	0.49	0.72	0.82	0.94	0.99	1.07
L2-SVM with CV	4.88	3.52	0.81	0.93	1.06	1.15	1.20	1.36
L2-SVM with CIS and CV	10.54	2.87	1.57	2.15	2.15	2.15	2.24	2.24
L2-SVM Gradient Descent	8.72	2.07	0.88	0.84	1.01	1.20	1.56	1.20
Benchmark	4.73	1.98	0.56	0.83	1.00	1.15	1.20	1.30
Improvement (%)	-3.09	11.72	11.73	13.40	18.17	17.99	17.65	17.62

TABLE II
PROCESSING TIME (MINUTES)

	Case 1	Case 2	Case 3
Backpropagation	3.90	3.67	3.37
Gain Scaling	3.45	2.80	1.16
Gain Scaling with SS	23.99	20.94	16.29
Gain Scaling with CIS and SS	20.73	19.14	16.54
Early Stopping	0.03	0.02	0.01
Early Stopping with SS	24.04	9.65	0.04
Early Stopping with CIS and SS	7.00	4.54	0.03
Extended Bayesian Training	32.30	5.80	28.15
L2-SVM with CV	36.57	13.76	19.57
L2-SVM with CIS and CV	30.00	13.57	20.53
L2-SVM Gradient Descent	8.73	29.08	13.84

Table I presents the mean absolute percentage errors (MAPE) from the training methods. Its last line shows the performance improvements between the best models and the benchmarks. The extended Bayesian methodology produces superior results for all cases, except for case 1, in which it has been overcome by a small margin. Although exhibiting good results, L2-SVM with gradient descent has not been competitive for case 1. This method has defeated its counterpart based on CV for half of the test cases. However, execution time of L2-SVM with gradient descent is, in average, smaller than the one of L2-SVM with CV.

Table II indicates the computational burden on a 3-GHz/32-bit PC, using MATLAB interpreted code. The 32 min for the Extended Bayesian Training correspond to forecasting a full day load curve in an hourly basis, which is compatible with practical requirements. The source codes associated with the proposed training algorithms have been based on [35] and [36]. In L2-SVM with CV, the number of hyperparameters has been decreased by making $\sigma_i = \sigma$, otherwise CV is not viable. Therefore, L2-SVM with CV has used the full set of predefined inputs and the input set determined by CIS (L2-SVM with CIS and CV). Notice that early stopping variations do not compete in accuracy with the extended Bayesian training, in which the datasets have been fully exploited. Furthermore, on the contrary of the benchmarks, the two leading proposals have their input spaces automatically selected. The activation function gain scaling procedure has not exhibited good results.

Table III presents the maximum absolute errors. Maximum error units have been chosen according to the benchmarks. Again, the extended Bayesian training exhibits the best overall performance. In Table IV, the average numbers of inputs selected by each model are presented. This table indicates

TABLE III
COMPARISON AMONG DIFFERENT MODELS (MAXIMUM ERROR)

	Case 1 (%)	Case 2 (MW)	Case 3 (%)					
			1 step ahead	2 step ahead	3 step ahead	4 step ahead	5 step ahead	6 step ahead
Backpropagation	93.12	118.89	4.99	5.96	4.50	5.85	6.73	8.02
Gain Scaling	66.54	137.78	7.61	10.66	9.10	11.22	10.82	11.48
Gain Scaling with SS	87.50	55.95	6.89	9.16	14.87	7.46	11.88	10.21
Gain Scaling with CIS and SS	112.89	70.99	11.61	20.48	23.85	11.36	12.77	11.22
Early Stopping	43.98	40.28	7.84	10.38	7.91	15.13	7.02	12.95
Early Stopping with SS	46.07	50.90	5.97	6.79	5.56	6.18	7.16	5.73
Early Stopping with CIS and SS	54.03	71.26	7.32	9.43	8.66	8.34	8.52	8.72
Extended Bayesian Training	41.57	55.64	1.97	2.65	3.89	4.62	4.86	5.46
L2-SVM with CV	38.06	60.39	4.00	3.51	4.53	4.62	5.45	5.95
L2-SVM with CIS and CV	60.06	67.17	5.90	6.19	6.18	6.17	6.48	6.48
L2-SVM Gradient Descent	46.70	59.78	3.48	4.05	5.12	5.87	6.14	5.59
Benchmark	-	51.42	3.24	3.43	4.11	3.87	5.57	5.20
Improvement (%)	-	21.66	39.09	22.64	5.44	-19.26	12.69	-4.93

TABLE IV
AVERAGE NUMBERS OF INPUTS

	Case 1	Case 2	Case 3					
			1 step ahead	2 step ahead	3 step ahead	4 step ahead	5 step ahead	6 step ahead
Backpropagation	84	49	79	77	75	73	71	69
Gain Scaling	84	49	79	77	75	73	71	69
Gain Scaling with SS	84	49	79	77	75	73	71	69
Gain Scaling with CIS and SS	26	20	27	26	26	26	26	26
Early Stopping	84	49	79	77	75	73	71	69
Early Stopping with SS	84	49	79	77	75	73	71	69
Early Stopping with CIS and SS	26	20	27	26	26	26	26	26
Extended Bayesian Training	70	40	66	67	63	51	60	56
L2-SVM with CV	84	49	79	77	75	73	71	69
L2-SVM with CIS and CV	26	20	27	26	26	26	26	26
L2-SVM Gradient Descent	76	45	73	71	71	61	60	65
Reduction (%)	68.55	58.99	66.37	66.79	65.90	64.97	63.78	62.73

TABLE V
AVERAGE NUMBERS OF NEURONS AND SUPPORT VECTORS

	Case 1	Case 2	Case 3					
			1 step ahead	2 step ahead	3 step ahead	4 step ahead	5 step ahead	6 step ahead
Backpropagation	10	10	10	10	10	10	10	10
Gain Scaling	10	10	10	10	10	10	10	10
Gain Scaling with SS	8	1	7	9	8	9	9	9
Gain Scaling with CIS and SS	6	2	4	6	6	5	6	6
Early Stopping	10	10	10	10	10	10	10	10
Early Stopping with SS	8	8	8	8	8	8	8	8
Early Stopping with CIS and SS	8	6	7	5	7	8	7	6
Extended Bayesian Training	8	7	7	8	7	5	5	3
L2-SVM with CV	428	464	344	338	328	330	333	347
L2-SVM with CIS and CV	425	464	347	348	346	344	342	341
L2-SVM Gradient Descent	642	707	518	515	509	513	510	505

the capacity of the leading methodologies to reduce the input dimensionality, improving the models' generalization ability. For example, in case 3, for six-step ahead forecasts, $T(k-168), P(k-6), P(k-26), P(k-29), P(k-168), P(k-171), D(4), D(6), D(10), D(11)$, and $D(16)$ have been disregarded by the Bayesian method, where $P(\cdot)$ and $D(\cdot)$ stand for price and dummy variables, respectively. For the same case, correlation-based input selection has saved $L(k-24)$ and $L(k-168)$, only. Input variables related to temperature and price have been disregarded by CIS due to their strong nonlinear relationship with load. In Table V, the average numbers of neurons in the hidden layer of the MLPs and the average numbers of support vectors are presented.

Figs. 1–3 show some forecasts for the three databases. Fig. 1 presents one example for the first database (case 1). Fig. 2 shows predictions for the second database (case 2). Fig. 3 presents forecasts for six steps ahead in case 3.

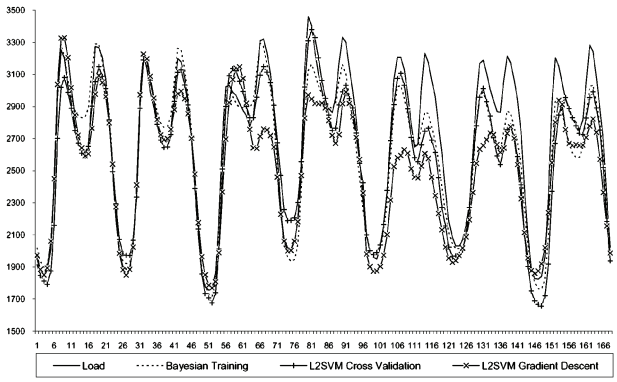

Fig. 1. Forecasts from 11/27/1990 to 12/03/1990, case 1.

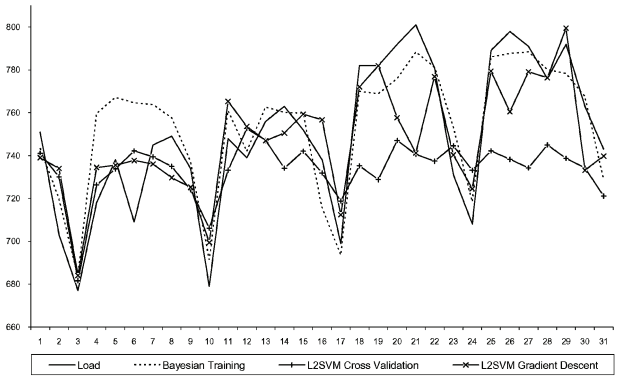

Fig. 2. Forecasts from 1/1/1999 to 1/31/1999, case 2.

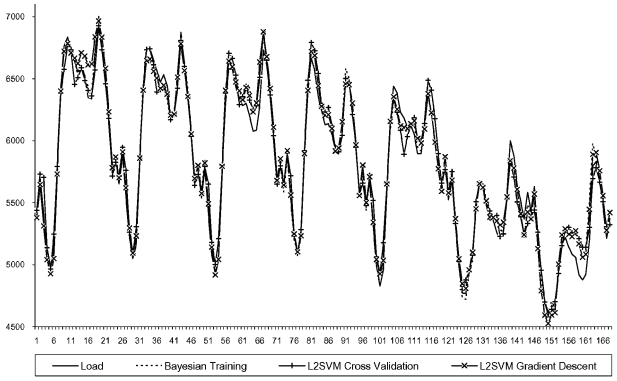

Fig. 3. Forecasts from 9/1/2003 to 9/07/2003, six steps ahead, case 3.

VI. CONCLUSION

This paper has extended Bayesian and SVM learning techniques to propose autonomous NN-based short-term load forecasters. The proposed methodologies are fully data-driven, providing accurate forecasts with very little information from the user. Although requesting a heavy computational burden, they seem to be the answer for dealing with the large-scale bus load forecasting problem, in which the particular dynamics of each load series does not allow manually tuned solutions.

Comparing the practical aspects of the hyperparameters estimation, without CV, in Bayesian and L2-SVM training, the following facts have been observed. First of all, hyperparameter estimation cannot be performed in L2-SVM learning without an auxiliary procedure. Furthermore, it is easier to get good estimations for the Bayesian training hyperparameters, based on Gaussian priors, than to optimize the L2-SVM learning parameters using gradient descent. Gaussian priors seem to be robust for different load series. On the other hand, the gradient descent algorithm usually requires many iterations (with one L2-SVM

training per iteration), and its convergence is strongly dependent on the stepsize control.

Bayesian inference has been applied for clustering load dynamics to feed different SVM load forecasting models [24]. However, the application of Bayesian inference to the estimation of SVM learning parameters looks more promising. There is already some research effort on this idea [37], and it is worthwhile to pursue this direction for the next generation of short-term load forecasting tools.

REFERENCES

[1] H. S. Hippert, R. C. Souza, and C. E. Pedreira, "Neural networks for load forecasting: A review and evaluation," *IEEE Trans. Power Syst.*, vol. 16, no. 1, pp. 44–55, Feb. 2001.

[2] N. Amjady, "Short-term bus load forecasting of power systems by a new hybrid method," *IEEE Trans. Power Syst.*, vol. 22, no. 1, pp. 333–341, Feb. 2007.

[3] A. Khotanzad, R. Afkhami-Rohani, and D. Maratukulam, "ANNSTLF-artificial neural network short-term load forecaster-generation three," *IEEE Trans. Power Syst.*, vol. 13, no. 4, pp. 1413–1422, Nov. 1998.

[4] D. C. Park, M. A. El-Sharkawi, and R. J. Marks II, "An adaptively trained neural network," *IEEE Trans. Neural Netw.*, vol. 2, no. 3, pp. 334–345, May 1991.

[5] A. P. Alves da Silva, V. H. Quintana, and G. K. H. Pang, "Neural networks for topology determination of power systems," in *Proc. 1st Int. Forum Applications of Neural Networks to Power Systems*, Seattle, WA, Jul. 1991, pp. 297–301.

[6] A. P. Alves da Silva and L. S. Moulin, "Confidence intervals for neural network based short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 15, no. 4, pp. 1191–1196, Nov. 2000.

[7] A. G. Bakirtzis, J. B. Theocharis, S. J. Kiartzis, and K. J. Satsios, "Short term load forecasting using fuzzy neural networks," *IEEE Trans. Power Syst.*, vol. 10, no. 3, pp. 1518–1524, Aug. 1995.

[8] T. Matsui, T. Iizaka, and Y. Fukuyama, "A novel daily peak load forecasting method using analyzable structured neural network," in *Proc. IEEE PES Winter Meeting*, Jan. 2001, pp. 405–410.

[9] A. P. Alves da Silva, "Overcoming limitations of NNs for on-line DSA," presented at the IEEE PES General Meeting, San Francisco, CA, Jun 2005.

[10] H. S. Hippert, D. W. Bunn, and R. C. Souza, "Large neural networks for electricity load forecasting: Are they overfitted?," *Int. J. Forecast.*, vol. 21, no. 3, pp. 425–434, Jul. 2005.

[11] S. Amari, N. Murata, K. R. Müller, M. Finke, and H. Yang, *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1996, vol. 8, pp. 176–182.

[12] Z. Cataltepe, Y. S. Abu-Mostafa, and M. Magdon-Ismail, "No free lunch for early stopping," *Neural Comput.*, vol. 11, no. 4, pp. 995–1009, May 1999.

[13] A. J. R. Reis and A. P. Alves da Silva, "Feature extraction via multiresolution analysis for short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 20, no. 1, pp. 189–198, Feb. 2005.

[14] D. J. C. Mackay, "Bayesian methods for adaptive models," Ph.D. dissertation, California Inst. Technol., Pasadena, 1992.

[15] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[16] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[17] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press, 2002.

[18] R. Reed, R. J. Marks II, and S. Oh, "Similarities of error regularization, sigmoid gain scaling, target smoothing and training with jitter," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 529–538, May 1995.

[19] V. H. Ferreira and A. P. Alves da Silva, "Complexity control of neural models for load forecasting," in *Proc. Int. Conf. Intelligent System Application to Power Systems*, Washington, DC, Nov. 2005, pp. 100–104.

[20] Z. S. H. Chan, H. W. Ngan, A. B. Rad, A. K. David, and N. Kasabov, "Short-term ANN load forecasting from limited data using generalization learning strategies," *Neurocomputing*, vol. 70, no. 1–3, pp. 409–419, Dec. 2006.

[21] P. F. Pai and W. C. Hong, "Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms," *Elect. Power Syst. Res.*, vol. 74, no. 3, pp. 417–425, Jun. 2005.

[22] S. Fan and L. N. Chen, "Short-term load forecasting based on an adaptive hybrid method," *IEEE Trans. Power Syst.*, vol. 21, no. 1, pp. 392–401, Feb. 2006.

[23] J. F. Yang and J. Stenzel, "Short-term load forecasting with increment regression tree," *Elect. Power Syst. Res.*, vol. 76, no. 9–10, pp. 880–888, Jun. 2006.

[24] S. Fan, C. X. Mao, J. D. Zhang, and L. N. Chen, "Forecasting electricity demand by hybrid machine learning model," *Lecture Notes Comput. Sci.*, vol. 4233, pp. 952–963, Oct. 2006.

[25] B.-J. Chen, M.-W. Chang, and C.-J. Lin, "Load forecasting using support vector machines: A study on EUNITE competition 2001," *IEEE Trans. Power Syst.*, vol. 19, no. 4, pp. 1821–1830, Nov. 2004.

[26] R. Ramanathan, R. Engle, C. W. J. Granger, F. Vahid-Araghi, and C. Brace, "Short-run forecasts of electricity loads and peaks," *Int. J. Forecast.*, vol. 13, no. 2, pp. 161–174, Jun. 1997.

[27] P. Mandal, T. Senjyu, and T. Funabashi, "Neural networks approach to forecast several hour ahead electricity prices and loads in deregulated market," *Energy Conv. Management*, vol. 47, no. 15–16, pp. 2128–2142, Sep. 2006.

[28] N. K. Treadgold and T. D. Gedeon, "Exploring constructive cascade networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1335–1350, Nov. 1999.

[29] N. Murata, S. Yoshizawa, and S. I. Amari, "Network information criterion-determining the number of hidden units for an artificial neural network model," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 865–872, Nov. 1994.

[30] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar, "Ranking a random feature for variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1399–1414, Mar. 2003.

[31] L. S. Moulin, A. P. Alves da Silva, M. A. El-Sharkawi, and R. J. Marks II, "Support vector machines for transient stability analysis of large-scale power systems," *IEEE Trans. Power Syst.*, vol. 19, no. 2, pp. 818–825, May 2004.

[32] M.-W. Chang and C.-J. Lin, "Leave-one-out bounds for support vector regression model selection," *Neural Comput.*, vol. 17, no. 5, pp. 1188–1222, May 2005.

[33] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Mach. Learn.*, vol. 46, pp. 131–159, Jan. 2002.

[34] V. Cherkassky and Y. Ma, "Practical selection of svm parameters and noise estimation for SVM regression," *Neural Netw.*, vol. 17, no. 1, pp. 113–126, Jan. 2004.

[35] I. T. Nabney, *NETLAB: Algorithms for Pattern Recognition*. New York: Springer-Verlag, 2002.

[36] C.-C. Chang and C.-J. Lin, LIBSVM: A Library for Support Vector Machines, 2001. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm, available at

[37] C. Wei, S. S. Keerthi, and J. O. Chong, "Bayesian support vector regression using a unified loss function," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 29–44, Jan. 2004.

Vitor Hugo Ferreira received the B.Sc. and M.Sc. degrees in electrical engineering from the Federal University of Itajubá, Brazil, in 2002, and the Federal University of Rio de Janeiro, Brazil, in 2005, respectively. He is currently pursuing the Ph.D. degree at the Electrical Engineering Graduate Program, Federal University of Rio de Janeiro (COPPE/UFRJ).

His research interests include time series forecasting and neural networks.

Alexandre P. Alves da Silva (SM'00) received the B.Sc. and M.Sc. degrees from the Catholic University of Rio de Janeiro, Brazil, in 1984 and 1987, respectively, and the Ph.D. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1992, respectively.

During 1999, he was a Visiting Professor in the Department of Electrical Engineering, University of Washington, Seattle. Currently, he is a Professor of electrical engineering, Federal University of Rio de Janeiro (COPPE/UFRJ), Brazil. He has authored and coauthored 200 papers on intelligent systems application to power systems.