

Automatic Kernel Based Models for Short Term Load Forecasting

Vitor Hugo Ferreira
Electrical Engineering Department
Fluminense Federal University (UFF)
Niterói, RJ, Brazil
vitor@vm.uff.br

Alexandre Pinto Alves da Silva
Electrical Engineering Graduate Program
Federal University of Rio de Janeiro, COPPE/UFRJ
Rio de Janeiro, RJ, Brazil
alex@coep.ufrj.br

Abstract— The application of support vector machines to forecasting problems is becoming popular, lately. Several comparisons between neural networks trained with error backpropagation and support vector machines have shown advantage for the latter in different domains of application. However, some difficulties still deteriorate the performance of the support vector machines. The main one is related to the setting of the hyperparameters involved in their training. Techniques based on meta-heuristics have been employed to determine appropriate values for those hyperparameters. However, because of the high nonconvexity of this estimation problem, which makes the search for a good solution very hard, an approach based on Bayesian inference, called relevance vector machine, has been proposed more recently. The present paper aims at investigating the suitability of this new approach to the short term load forecasting problem.

Keywords- Load forecasting, artificial neural networks, input selection, kernel based models, support vector machine, relevance vector machine.

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. 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.

Artificial Neural Networks (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 being more robust, the ANN input representation and complexity control should not be treated separately, as it is common practice when applied to load forecasting. The extent of nonlinearity required from an ANN is strongly dependent on the selected input variables.

Input selection is one of the most important tasks in load forecasting. Nonlinear feature extraction techniques such as the one presented in [1] have adopted a univariate approach, i.e., only load data have been pre-processed. Therefore, a more ANN oriented feature extraction approach is still needed. On the other hand, ANN complexity control aims at matching the data regularity with the extent of nonlinearity provided by the ANN, avoiding noise tracking and model generalization deterioration.

Originally proposed for dealing with classification problems, Support Vector Machines (SVMs) [2] have their training formulation oriented to maximizing generalization performance. Although some preliminary studies [3], [4] have shown the capability of such models, analytical methodologies are badly needed for estimating their hyperparameters. This is because cross-validation procedures should be avoided when dealing with time series to avoid loss of serial dependence information. The application of Bayesian inference to regression problems creates kernel based sparse models similar to SVMs. These models have been called Relevance Vector Machines (RVMs) [5], which have an embedded analytical mechanism for estimating the hyperparameters.

This paper explores automatic techniques for hyperparameter estimation in kernel based models applied to short term load forecasting. In particular, this work investigates the minimization of the upper bound of the generalization error for SVMs [6] and RVMs [5]. A probabilistic analysis of the hyperparameters is the basis of a new automatic technique for feature selection. The proposed methodology has been tested with three public domain databases, which are described in Section IV.

II. SUPPORT VECTOR MACHINES

In classification problems, 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 Eq. 2). 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:

$$y = \sum_{j=0}^m W_j \phi_j(\underline{x}) = \underline{W}' \underline{\phi}(\underline{x}) \quad (1)$$

where $\underline{\phi}(\underline{x}) = [1, \phi_1(\underline{x}), \dots, \phi_m(\underline{x})]'$ and $\underline{W} = [b, W_1, \dots, W_m]'$.

The following ε -insensitive cost function is adopted here:

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

SVMs that use Eq. (2) as the error function are called L2-SVMs, 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 cross-validation.

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:

$$\min_{\underline{W}} \left\{ E_s(\underline{W}, D) = \frac{1}{N} \sum_{i=1}^N L_\varepsilon(d_i, y_i) \right\} \quad (3)$$

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 Eq. (3) is transformed into its dual form, Eq. (4), to allow the incorporation of kernel functions, which avoid the requirement of knowing an appropriate $\underline{\phi}(\underline{x})$.

$$\max_{\underline{\alpha}, \underline{\alpha}'} \left\{ Q(\underline{\alpha}, \underline{\alpha}') = \sum_{i=1}^N d_i (\alpha_i - \alpha_i') - \varepsilon \sum_{i=1}^N (\alpha_i + \alpha_i') \right. \\ \left. - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i - \alpha_i') (\alpha_j - \alpha_j') [K(\underline{x}_i, \underline{x}_j) + \delta_{ij}/C] \right\} \quad (4)$$

subject to

$$\sum_{i=1}^N (\alpha_i - \alpha_i') = 0 \\ \alpha_i \geq 0, \alpha_i' \geq 0, i = 1, 2, \dots, N$$

In Eq. (4), $K(\underline{x}_i, \underline{x}_j) = \underline{\phi}'(\underline{x}_i) \underline{\phi}(\underline{x}_j)$ is the inner product kernel defined according to Mercer's theorem, δ_{ij} is the Kronecker delta function, and C is the regularization hyperparameter. Then, the output of an SVM is given by:

$$y = f(\underline{x}, \underline{W}) = \sum_{i=1}^N (\alpha_i - \alpha_i') K(\underline{x}, \underline{x}_i) \quad (5)$$

As indicated in Eq. (5), 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 [7] develops the concept of *span of support vectors*, from which a differentiable upper bound on the generalization error for regression can be derived [6]:

$$T_{SB}[f(\underline{x}, \underline{w})] = \sum_{i=1}^p (\alpha_i + \alpha_i') \tilde{S}_i^2 + N\varepsilon \quad (6)$$

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

$$\tilde{S}_i^2 = \min_{\underline{\mu}} \left\| \tilde{\phi}(\underline{x}_i) - \sum_{j=1, j \neq i}^p \mu_j \tilde{\phi}(\underline{x}_j) \right\|^2 + \eta \sum_{j=1, j \neq i}^p \frac{\mu_j^2}{(\alpha_i + \alpha_i')} \quad (7)$$

subject to

$$\sum_{j=1, j \neq i}^p \mu_j = 1, \text{ for } \mu_j \in \mathfrak{R}.$$

with η denoting a parameter responsible for promoting differentiability ($\eta=0$ turns the objective function in Eq. (7) to a nondifferentiable one) and $\tilde{\phi}(\underline{x}_j) = [\underline{\phi}(\underline{x}_j) \quad \underline{o}_j / \sqrt{C}]'$ representing an extended feature space mapping, where \underline{o}_j is an N dimensional vector with the j^{th} element equal to one and the other elements equal to zero.

The optimal solution for Eq. (7) is presented in [6], along with the partial derivatives of T_{SB} with respect to C , ε , and the kernel parameters. The minimization of Eq. (6), 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 reference [6], the present work estimates the individual contributions of each input to the Gaussian kernel as a way to select input variables.

Input weights, σ_i s, for measuring the significance of each pre-selected 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:

$$K(\underline{x}, \underline{y}) = e^{-\sum_{i=1}^n (\sigma_i x_i - \sigma_i y_i)^2} = e^{-\sum_{i=1}^n \sigma_i^2 (x_i - y_i)^2} \quad (8)$$

A small scaling factor means that the corresponding input is not relevant. Therefore, such an input should be disregarded. However, a reference for defining a ‘‘small’’ σ_i is needed. Therefore, two auxiliary input variables (a real-valued variable and a binary one for dealing with dummy variables) with uniform distribution are added to the input set in order to establish a reference for relevance. Afterwards, the input variables with σ_i 's less or equal to the auxiliary variables are disregarded, and a final model is estimated based on the significant variables only. 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, reference [8] derives useful expressions (Eq. 9) 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:

$$C_{est} = \max\left(\left|\bar{d} + 3s_d\right|, \left|\bar{d} - 3s_d\right|\right) \quad (9)$$

$$\varepsilon_{est} = 3s \sqrt{\frac{\ln N}{N}}$$

where $s = \sqrt{\frac{1}{N-n} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$. In Eq. (9), \bar{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 ARX (Auto Regressive Exogenous) 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.

Step 2. Set $l=0$ and initialize $C(l)$ and $\varepsilon(l)$, using Eq. (9). Initialize the scaling factors $\underline{\sigma}(l) = [\sigma_1(l), \dots, \sigma_n(l)]^t$. In this work, all scaling factors are initially equal to 0.1.

Step 3. Solve Eq. (4) 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_i s associated with the continuous input variables and the σ_j s related to the dummy variables.

Step 7. For each list, select the inputs such that the corresponding $\sigma > \sigma_{ref}$, 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.

III. RELEVANCE VECTOR MACHINES

RVMs are kernel based probabilistic models that hold some of the advantages of SVMs, such as the sparse representation. Similarly to SVMs, only some of the training data contribute for the estimation of the regression surface. These training patterns are called relevant vectors.

Given a dataset $D = \{\underline{X}, \underline{Y}\}$, $\underline{X} \in \mathbb{R}^N \times \mathbb{R}^n$, $\underline{X} = [x_1 \ x_2 \ \dots \ x_N]^t$, $x_k \in \mathbb{R}^n$, $\underline{Y} \in \mathbb{R}^N$, $\underline{Y} = [d_1 \ d_2 \ \dots \ d_N]^t$, $d_k \in \mathbb{R}$, and assuming an additive noise $\zeta_k \in \mathbb{R}$ in the desired output, i.e., $d_k = F(x_k) + \zeta_k$, the generating function $F(x): \mathbb{R}^n \rightarrow \mathbb{R}$ can be mapped as follows.

Let the approximation function $f(\underline{x}, \underline{w}): \mathbb{R}^n \rightarrow \mathbb{R}$ be formed by a linear combination of basis functions $\Phi(\underline{x}, \underline{z}): \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ with the corresponding centers at each point of the dataset D . Then, the model output is given by:

$$f(\underline{x}, \underline{W}) = \sum_{i=1}^N w_i \Phi(\underline{x}, x_i) + b = [\Phi(\underline{x})]^t \underline{W} \quad (10)$$

where $\underline{w} \in \mathbb{R}^N$, $b \in \mathbb{R}$, $\underline{W} \in \mathbb{R}^{N+1}$, $\underline{W} = [b \ \underline{w}^t]^t$, and $\Phi(\underline{x}): \mathbb{R}^n \rightarrow \mathbb{R}^{N+1}$, which represents the basis functions $\Phi(\underline{x}, x_i) = \Phi_i(\underline{x})$ evaluated for \underline{x} and centered at each point of D , including the bias.

Bayes' rule can be utilized for estimating \underline{W} , i.e., the *a posteriori* $p(\underline{W}|Y, X)$ is:

$$p(\underline{W}|Y, X) = \frac{p(Y|\underline{W}, X)p(\underline{W}|X)}{p(Y|X)} \quad (11)$$

where $p(Y|X)$ is the normalization factor, $p(\underline{W}|X)$ stands for the *a priori* distribution of \underline{W} (given X) and $p(Y|\underline{W}, X)$ is the likelihood function (related to the noise distribution function of the desired output). For the sake of clarity, the input X is omitted from Eq. (11) on.

Assuming that noise ζ_k is independently generated from a Gaussian distribution with zero mean and variance $\sigma^2 \in \mathbb{R}$, the likelihood $p(Y|\underline{W})$ is given by:

$$p(Y|\underline{W}, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left(-\frac{\|\underline{Y} - \underline{\Phi}\underline{W}\|^2}{2\sigma^2}\right) \quad (12)$$

In reference [5], $\underline{\Phi} \in \mathbb{R}^N \times \mathbb{R}^{N+1}$ in Eq. (12) is the so called modeling matrix. The *a priori* distribution $p(\underline{W})$ can be estimated from the product of Gaussian distributions as follows:

$$p(\underline{W}|\underline{\alpha}) = \prod_{i=1}^{N+1} \frac{1}{\sqrt{2\pi\alpha_i^{-1}}} \exp\left(-\frac{1}{2\alpha_i^{-1}}W_i^2\right) \quad (13)$$

The hyperparameters σ^2 and $\underline{\alpha}$ also require *a priori* probabilities. Following [5], noninformative Gamma distributions are applied, reflecting the absence of prior knowledge regarding the hyperparameters' distributions.

The joint *a posteriori* probability $p(\underline{W}, \underline{\alpha}, \sigma^2 | Y)$ of \underline{W} , $\underline{\alpha}$ and σ^2 can be written as:

$$p(\underline{W}, \underline{\alpha}, \sigma^2 | Y) = p(\underline{W}|\underline{Y}, \underline{\alpha}, \sigma^2) p(\underline{\alpha}, \sigma^2 | Y) \quad (14)$$

The probability $p(\underline{W}|\underline{Y}, \underline{\alpha}, \sigma^2)$ is obtained from Bayes' rule:

$$p(\underline{W}|\underline{Y}, \underline{\alpha}, \sigma^2) = \frac{p(Y|\underline{W}, \sigma^2)p(\underline{W}|\underline{\alpha})}{p(Y|\underline{\alpha}, \sigma^2)} \quad (15)$$

As $p(Y|\underline{W}, \sigma^2)$ and $p(\underline{W}|\underline{\alpha})$ are Gaussian distributions, $p(Y|\underline{\alpha}, \sigma^2)$ can be estimated from the convolution of those distributions, i.e.:

$$p(\underline{Y}|\underline{\alpha}, \sigma^2) = \frac{1}{(2\pi)^{\frac{N}{2}} |\underline{C}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\underline{Y}'\underline{C}^{-1}\underline{Y}\right) \quad (16)$$

$$\underline{C} = \sigma^2 \underline{I} + \underline{\Phi}\underline{A}^{-1}\underline{\Phi}'$$

In Eq. (16), $\underline{I} \in \mathbb{R}^N \times \mathbb{R}^N$ is an identity matrix and $\underline{A} \in \mathbb{R}^{N+1} \times \mathbb{R}^{N+1}$ is a diagonal matrix with $a_{ii} = \alpha_i$. Therefore, the Gaussian distributed $p(\underline{W}|\underline{Y}, \underline{\alpha}, \sigma^2)$ is:

$$p(\underline{W}|\underline{Y}, \underline{\alpha}, \sigma^2) = \frac{\exp\left[-\frac{1}{2}(\underline{W} - \underline{\mu})' \underline{\Sigma}^{-1}(\underline{W} - \underline{\mu})\right]}{(2\pi)^{\frac{N+1}{2}} |\underline{\Sigma}|^{\frac{1}{2}}} \quad (17)$$

The covariance matrix $\underline{\Sigma} \in \mathbb{R}^{N+1} \times \mathbb{R}^{N+1}$ and the mean vector $\underline{\mu} \in \mathbb{R}^{N+1}$ are estimated as follows:

$$\underline{\Sigma} = (\sigma^2 \underline{\Phi}'\underline{\Phi} + \underline{A})^{-1} \quad (18)$$

$$\underline{\mu} = \sigma^{-2} \underline{\Sigma}\underline{\Phi}'\underline{Y}$$

After obtaining $p(\underline{W}|\underline{Y}, \underline{\alpha}, \sigma^2)$ aiming at calculating Eq. (14), the next step involves the probability $p(\underline{\alpha}, \sigma^2 | Y)$. From the principle of hyperparameters evidence maximization, the following expressions are derived:

$$\gamma_i(l) = 1 - \alpha_i(l) \Sigma_{ii}(l) \quad (19)$$

$$\alpha_i(l+1) = \frac{\gamma_i(l)}{\mu_i^2(l)} \quad (20)$$

$$\sigma^2(l+1) = \frac{\|\underline{Y} - \underline{\Phi}\underline{\mu}(l)\|^2}{N - \sum_{i=1}^{N+1} \gamma_i(l)} \quad (21)$$

The expected value \hat{d}_{N+1} and the variance $\hat{\sigma}^2$ of the desired output (d_{N+1}) associated with a test point \underline{x}_{N+1} are calculated as below:

$$\hat{d}_{N+1} = f(\underline{x}_{N+1}, \underline{\mu}^{MP}) = [\underline{\Phi}(\underline{x}_{N+1})]'\underline{\mu}^{MP} \quad (22)$$

$$\hat{\sigma}^2 = (\sigma^{MP})^2 + [\underline{\Phi}(\underline{x}_{N+1})]'\underline{\Sigma}^{MP}\underline{\Phi}(\underline{x}_{N+1})$$

In Eq. (22), $\underline{\mu}^{MP}$ and $\underline{\Sigma}^{MP}$ are calculated from the application of expressions (18), with $\underline{\alpha}^{MP}$ and σ^{MP} as the final values from the iterative process defined by Eqs. (19), (20), and (21).

As already mentioned in Section II.B, the adopted basis function allows the determination of the relevance of each input, i.e.:

$$\Phi(\underline{x}, \underline{z}) = e^{-\sum_{k=1}^n \eta_k^2 (x_k - z_k)^2} \quad (23)$$

Analogously to the Gaussian kernel used by the proposed SVMs, inputs associated with small η_k s have nonsignificant contributions for the output. Therefore, by using auxiliary inputs, which provide empirical relevance thresholds for the input variables, only significant variables are included in the final model. The hyperparameters η_k are estimated via gradient climb towards evidence maximization.

IV. TEST RESULTS

The three datasets are standardized. The first one, with hourly load and temperature values contains data from January 1st, 1985 to March 31st, 1991. In this case, the task is to forecast the hourly load, from 16 up to 40 hours (steps) ahead for weekdays, and from 16 up to 80 hours ahead for weekends. The test (out-of-sample) period goes from November 1st, 1990 to March 31st, 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 $S(k-1)$, $S(k-2)$, ..., $S(k-6)$, $S(k-24)$, $S(k-25)$, ..., $S(k-29)$, $S(k-168)$, $S(k-169)$, ..., $S(k-173)$ 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 hours ahead are provided by recursion, i.e., load forecasts feed inputs. The number of pre-selected 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 [9].

For the second database, with daily peak load and temperature values from January 1st, 1997 to January 31st, 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 1st, 1999 up to January 31st, 1999. To avoid recursion, 31 models are estimated, one for each step ahead, using all data until January 1st, 1999 (≈ 720 patterns per ANN). For the j^{th} 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 $L(d-j)$, $L(d-j-1)$, ..., $L(d-(6+j))$ and $T(d)$, $T(d-1)$, ..., $T(d-(6+j))$, 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 [3].

For the last database, at www.nemmco.com.au, with half-hourly load, price, and temperature values from December 4th, 2001 to December 31st, 2003, the task is to forecast the hourly loads, from 1 up to 6 hours 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 pre-defined inputs: $19-j$ lagged load, price and temperature variables, plus j temperature forecasts (i.e., $T(k)$, $T(k-1)$, ..., $T(k-j+1)$), and 24 dummy variables codifying the hour of the day, totaling $81-2j$ inputs. The pre-selected lags are $S(k-j)$, $S(k-j-1)$, ..., $S(k-6)$, $S(k-24)$, $S(k-25)$, ..., $S(k-29)$, $S(k-168)$, $S(k-169)$, ..., 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 [10].

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 proposed automatic RVM produces superior results in three out of 8 investigations. For case 3, one step ahead, the result from the proposed RVM is equivalent to the benchmark in statistical terms. Notice that the automatic RVM and the automatic SVM have achieved similar performance. However, the absence of any effort from the user regarding hyperparameter specification makes the proposed RVM a more convenient choice. Both automatic machines have outperformed the standard L2-SVM.

TABLE I
COMPARISON AMONG DIFFERENT MODELS (MAPE)

	Case 1	Case 2	Case 3					
			1 step	2 steps	3 steps	4 steps	5 passos	6 steps
L2-SVM	6.58	3.05	1.56	1.64	1.69	1.71	1.73	1.76
Auto L2-SVM	8.72	2.07	0.88	0.84	1.01	1.20	1.56	1.20
Auto RVM	8.46	2.76	0.60	1.23	0.99	1.40	1.11	1.18
Benchmark	4.73	1.98	0.56	0.83	1.00	1.15	1.20	1.30
Gain (%)	-39.11	-4.43	-7.77	-1.66	1.28	-4.67	7.29	9.12

Table II presents the maximum absolute errors. In Table III, the average numbers of inputs selected by each model are presented. This table indicates the capacity of the leading methodologies to reduce the input dimensionality, improving the models’ generalization ability.

TABLE II
COMPARISON AMONG DIFFERENT MODELS (MAXIMUM ERROR)

	Case 1 (%)	Case 2 (MW)	Case 3 (%)					
			1 step	2 steps	3 steps	4 steps	5 steps	6 steps
L2-SVM	32.83	58.86	5.21	5.50	5.61	5.88	5.97	6.02
Auto L2-SVM	46.70	59.78	3.48	4.05	5.12	5.87	6.14	5.59
Auto RVM	55.42	47.21	2.77	8.32	4.37	7.81	5.61	5.99
Benchmark	-	51.42	3.24	3.43	4.11	3.87	5.57	5.20

Table IV indicates the average numbers of support and relevant vectors for each test case. It is clear that the proposed automatic RVM is capable of estimating the desired mapping with a more sparse (parsimonious) representation. Figures 1 to 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). Figure 3 presents forecasts for six steps ahead in case 3.

TABLE III
AVERAGE NUMBER OF INPUTS

	Case 1	Case 2	Case 3					
			1 step	2 steps	3 steps	4 steps	5 steps	6 steps
L2-SVM	84	49	79	77	75	73	71	69
Auto L2-SVM	76	45	73	71	71	61	60	65
Auto RVM	84	49	79	77	75	73	71	69
Reduction (%)	9,19	7,18	7,23	8,16	5,71	16,83	15,29	6,21

TABLE IV
AVERAGE NUMBER OF SUPPORT OR RELEVANT VECTORS

	Case 1	Case 2	Case 3					
			1 step	2 steps	3 steps	4 passos	5 steps	6 steps
L2-SVM	642	669	523	519	515	514	509	509
Auto L2-SVM	642	707	518	515	509	513	510	505
Auto RVM	112	34	89	75	78	56	61	53

V. CONCLUSION

This paper has investigated the application of kernel based models to the short term load forecasting problem. With the motivation of providing more automatic tools to the system operators, the standard SVM and RVM models have been upgraded in this work to provide automatic input selection and setting of hyperparameters. Therefore, more load series can be treated in the on-line environment of a control center. Further developments are being envisioned for making the automatic setting of hyperparameters even more robust.

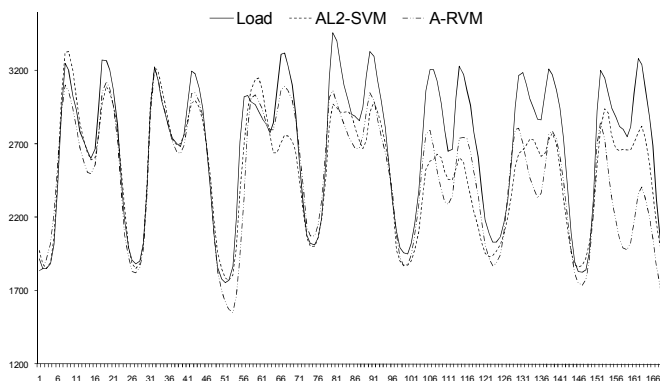


Figure 1: Case 1.

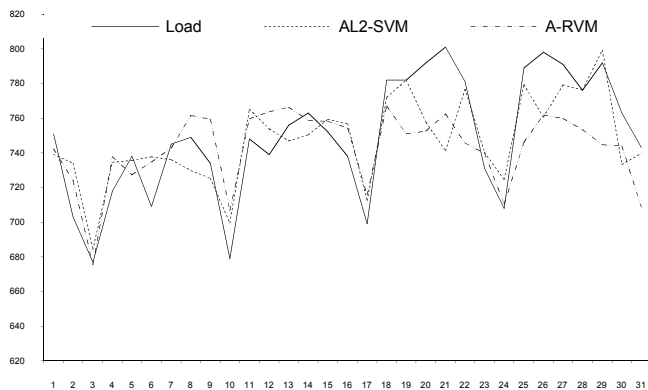


Figure 2: Case 2.

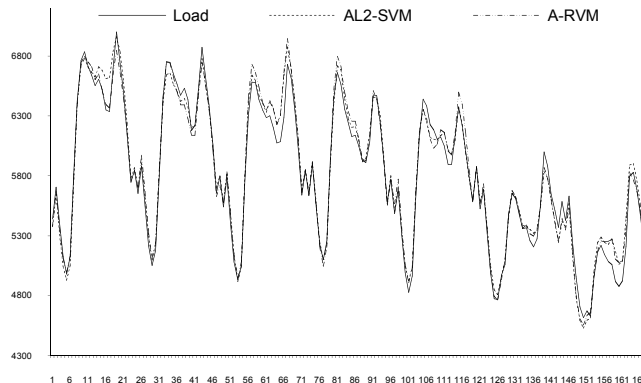


Figure 3: Case 3 – 6 steps ahead.

REFERENCES

- [1] Reis, A.J.R., Alves da Silva, A.P.: Feature Extraction Via Multi-Resolution Analysis for Short-Term Load Forecasting, *IEEE Transactions on Power Systems*, 20(1), pp. 189-198, 2005.
- [2] Vapnik, V.N.: *Statistical Learning Theory*, John Wiley & Sons, 1998.
- [3] Chen, B.-J., Chang, M.-W., Lin, C.-J.: Load Forecasting Using Support Vector Machines: A Study on EUNITE Competition 2001, *IEEE Transactions on Power Systems*, 19(4), pp. 1821-1830, 2004.
- [4] Ferreira, V.H., Alves da Silva, A.P.: Toward Estimating Autonomous Neural Network-based Electric Load Forecasters, *IEEE Transactions on Power Systems*, 22(4), pp. 1554-1562, 2007.
- [5] Tipping, M.: Sparse Bayesian Learning and the Relevance Vector Machine, *Journal of Machine Learning Research*, 1, pp. 211-244, 2001.
- [6] Chang, M.-W., Lin, C.-J.: Leave-One-Out Bounds for Support Vector Regression Model Selection, *Neural Computation*, 17(5), pp. 1188-1222, 2005.
- [7] Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing Multiple Parameters for Support Vector Machines, *Machine Learning*, 46, pp. 131-159, 2002.
- [8] Cherkassky, V., Ma, Y.: Practical Selection of SVM Parameters and Noise Estimation for SVM Regression, *Neural Networks*, 17(1), pp. 113-126, 2004.
- [9] Ramanathan, R., Engle, R., Granger, C.W.J., Vahid-Araghi, F., Brace, C.: Short-Run Fore-casts of Electricity Loads and Peaks, *International Journal of Forecasting*, 13(2), pp. 161-174, 1997.
- [10] Mandal, P., Senjyu, T., Uezato, K., Funabashi, T.: Several-Hours-Ahead Electricity Price and Load Forecasting Using Neural Networks, in *Proc. IEEE PES General Meeting, San Francisco, USA, 2005*.