Sensitivity Analysis for Time Lag Selection to Forecast Seasonal Time Series using Neural Networks and Support Vector Machines

Paulo Cortez

***Abstract*— Multi-step ahead forecasting is an important issue for organizations, often used to assist in tactical decisions. Such forecasting can be achieved by adopting time series forecasting methods, such as the classical Holt-Winters (HW) that is quite popular for seasonal series. An alternative forecasting approach comes from the use of more flexible learning algorithms, such as Neural Networks (NN) and Support Vector Machines (SVM). This paper presents a simultaneous variable (i.e. time lag) and model selection algorithm for multi-step ahead forecasting using NN and SVM. Variable selection is based on a backward algorithm that is guided by a sensitivity analysis procedure, while model selection is achieved using a grid-search. Several experiments were devised by considering eight seasonal series and the forecasts were analyzed using two error criteria (i.e. $SMAPE$ and MSE). Overall, competitive results were achieved when comparing the SVM and NN algorithms with HW.**

I. Introduction

The ability to forecast the future based on past data is an important tool to support individual and organizational decision making. Multi-step ahead predictions (e.g. issued several months in advance), are needed for tactical decisions, such as planning production resources [1]. The field of Time Series Forecasting (TSF) predicts the behavior of given phenomenon based solely on the past patterns of the same event [2]. Developments from the Operational Research has led to statistical TSF methods, such as the Holt-Winters (HW) [3]. The HW is from the family of exponential smoothing methods and it is quite used to predict series with trended and seasonal factors [4]. However, this method was developed decades ago, when higher computational restrictions prevailed (e.g. memory and computational power) and adopts a rather fixed model (i.e. with multiplicative or additive seasonality), that may not be suited when more complex components are present in the data.

Advances in information technologies have made it possible to collect and process massive datasets. Data mining (DM) techniques [5] aim at extracting useful knowledge from raw data. There are several DM algorithms, each one with its own advantages. For example, Neural Networks (NN) have become popular after the (re-)introduction of the backpropagation algorithm in 1986 [6]. More recently, Support Vector Machines (SVMs) have also been proposed [7][8]. It should be noted that SVMs present theoretical advantages over NNs, such as the absence of local minima in the learning phase. Both NNs and SVMs are more flexible (i.e. no *a priori* restriction is imposed) when compared with classical TSF models, presenting nonlinear learning capabilities. Thus, NNs and SVMs have been rapidly applied to TSF [9][10].

When applying NNs and SVMs to TSF, variable and model selection are critical issues. A sliding time window is often used to create a set of training examples from the series. A small time window will provide insufficient information, while using a high number of time lags will increase the probability of having irrelevant inputs. Thus, variable selection is useful to discard irrelevant time lags, leading to simpler models that are easier to interpret and that usually give better performances [11][12]. On the other hand, both NN and SVM have hyperparameters that need to be adjusted (e.g. number of NN hidden nodes or kernel parameter) [5]. Complex models may overfit the data, losing the capability to generalize, while a model that is too simple will present limited learning capabilities.

Sensitivity analysis is a simple procedure that is applied after the training phase and analyzes the model responses when the inputs are changed. This procedure is useful for measuring input importance and has also been proposed for variable selection, outperforming other input selection techniques (e.g. Forward Selection and Genetic Algorithms) in [13]. Originally proposed for NNs, this sensitivity method can also be applied to other algorithms, such as SVM. In [14], we presented a new computationally efficient procedure that performs a simultaneous variable and model selection for DM tasks. In this paper, we adapt such method for multi-step ahead forecasting. The sensitivity analysis is used to guide a backward selection search for the best time lags, while a grid-search is used for model selection. This approach is compared with the classical HW method over eight seasonal series from different domains.

The paper is organized as follows. Firstly, the time series data, evaluation and forecasting methods are presented in Section II. Next, the experiments and obtained results are presented and discussed in the Section III. Finally, closing conclusions are drawn (Section IV).

II. Materials and Methods

A. Time Series Data

A time series is a collection of time ordered observations $(y_1, y_2, \ldots, y_t)$, each one being recorded at a specific time t (period) [4]. A time series model $(\hat{y}_t)$ assumes that past patterns will occur in the future. Another relevant concept is the *horizon* or *lead time* (h), which is defined by the time in advance that a prediction is issued. Multi-step ahead forecasts

Paulo Cortez is with the Department of Information Systems/Algoritmi, University of Minho, 4800-058 Guimarães, PORTUGAL (phone: +351-253-510313; fax: +351-253-510300; email: pcortez@dsi.uminho.pt).

($h \in \{1, \ldots, N\}$) are often performed over monthly series and are used to support tactical decisions (e.g. leasing of plant and equipment) [1].

In this work, we address seasonal data, since we believe that time lag selection is particularly useful for these series, as seasonal lags should have a stronger affect in multi-step forecasts than other ones. This type of series is commonly present in several domains, such as Agriculture, Finance, Sales and Production [4]. We selected eight time series, with different characteristics and from different domains (Table I and Figure 1). The first seven datasets were selected from the Time Series Data Library (TSDL) repository [15], while the last chaotic series is described in [16].

TABLE I
TIME SERIES DATA

Series	Description (years)
cars	Monthly car sales in Quebec (1960-1968)
pigs	Monthly number of pigs slaughtered in Victoria (1980-1995)
pass	Monthly international airline passengers (1949-1960)
gas	Monthly gasoline demand in Ontario (1960-1975)
houses	Monthly sales of U.S. houses (1965-1975)
cradfq	Monthly critical radio freq. in Washington, D.C. (1934-1954)
suns	Annual Wolfer sunspot numbers (1770-1889)
MG	Mackey-Glass chaotic series

The autocorrelation coefficient is a useful tool to detect seasonal components or if a time series is random. It measures the correlation between a series and itself, lagged of k periods, and can be computed by [17]:

$$r_k = \frac{\sum_{t=1}^{P-k}(y_t - \overline{y_t})(y_{t+k} - \overline{y_t})}{\sum_{t=1}^{P}(y_t - \overline{y_t})} \quad (1)$$

where $y_1, y_2, ..., y_P$ stands for the time series and $\overline{y_t}$ for the series' average. Some examples of autocorrelations are plotted in Figure 2 (x-axis shows the time lags, while y-axis denotes the r_k values). In the plots, the seasonal period (K) is clearly visible. Also, several r_k values are above the 95% degree of confidence normal distribution bounds (horizontal dashed lines), showing that these series are not random and thus can be predicted. It should be noted that autocorrelations should not be used for time lag model selection (e.g. NN or SVM), since they only measure linear relationships. Still, we used the r_k values to detect/confirm the seasonal periods (K) presented in Table II.

B. Evaluation

The global performance of a forecasting model is evaluated by an accuracy measure, such as the *Mean Squared Error (MSE)*, *Relative Squared Error (RSE)* and *Symmetric Mean Absolute Percentage Error (SMAPE)*:

$$\begin{aligned} & e_t = y_t - \hat{y}_{t,t-h} \\ & MSE = \frac{1}{N}\sum_{i=P+1}^{P+N} e_i^2 \\ & RSE = \frac{\sum_{i=P+1}^{P+N} e_i^2}{\sum_{i=P+1}^{P+N}(y_t - \overline{y_t})^2} \times 100\% \\ & SMAPE = \frac{1}{N}\sum_{i=P+1}^{P+N} \frac{|e_i|}{(|y_i| + |\hat{y}_{t,t-h}|)/2} \times 100\% \end{aligned} \quad (2)$$

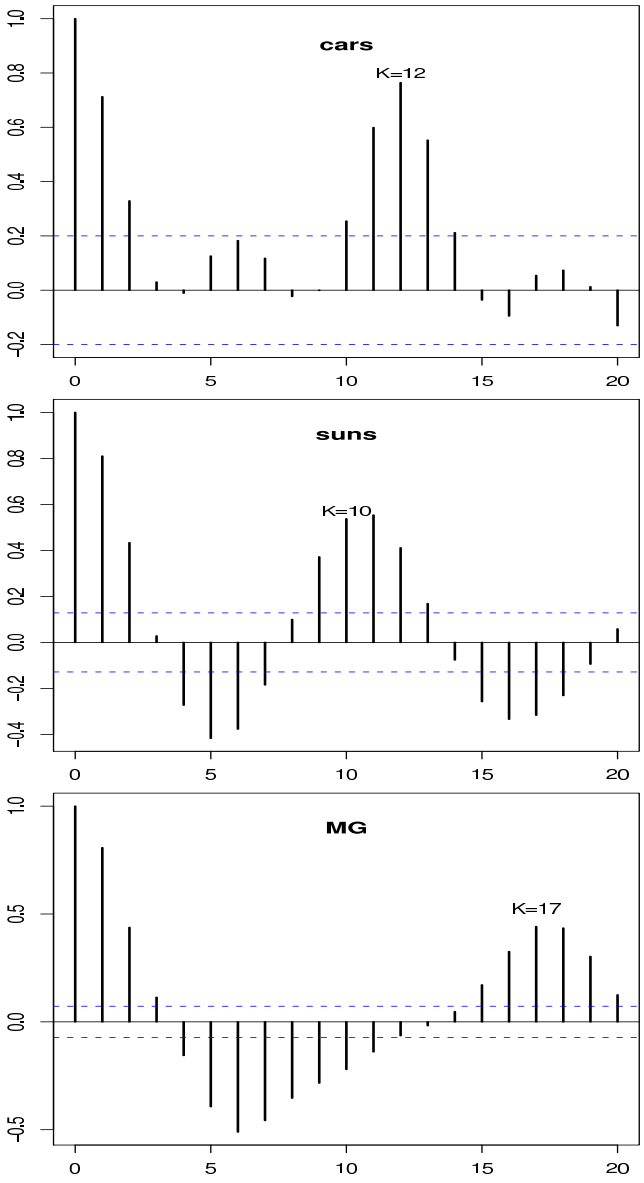

Fig. 2. Autocorrelations for the cars, suns and MG series (only training data was used)

TABLE II
TIME SERIES SEASONAL PERIOD AND TRAIN/TEST SIZES

Series	Seasonal Period (K)	Training Data Size	Test Set Size (N)	Total Size
cars	12	96	12	108
pigs	12	176	12	188
pass	12	125	19	144
gas	12	173	19	192
houses	12	120	12	132
cradfq	12	216	24	240
suns	10	264	25	289
MG	17	735	56	791

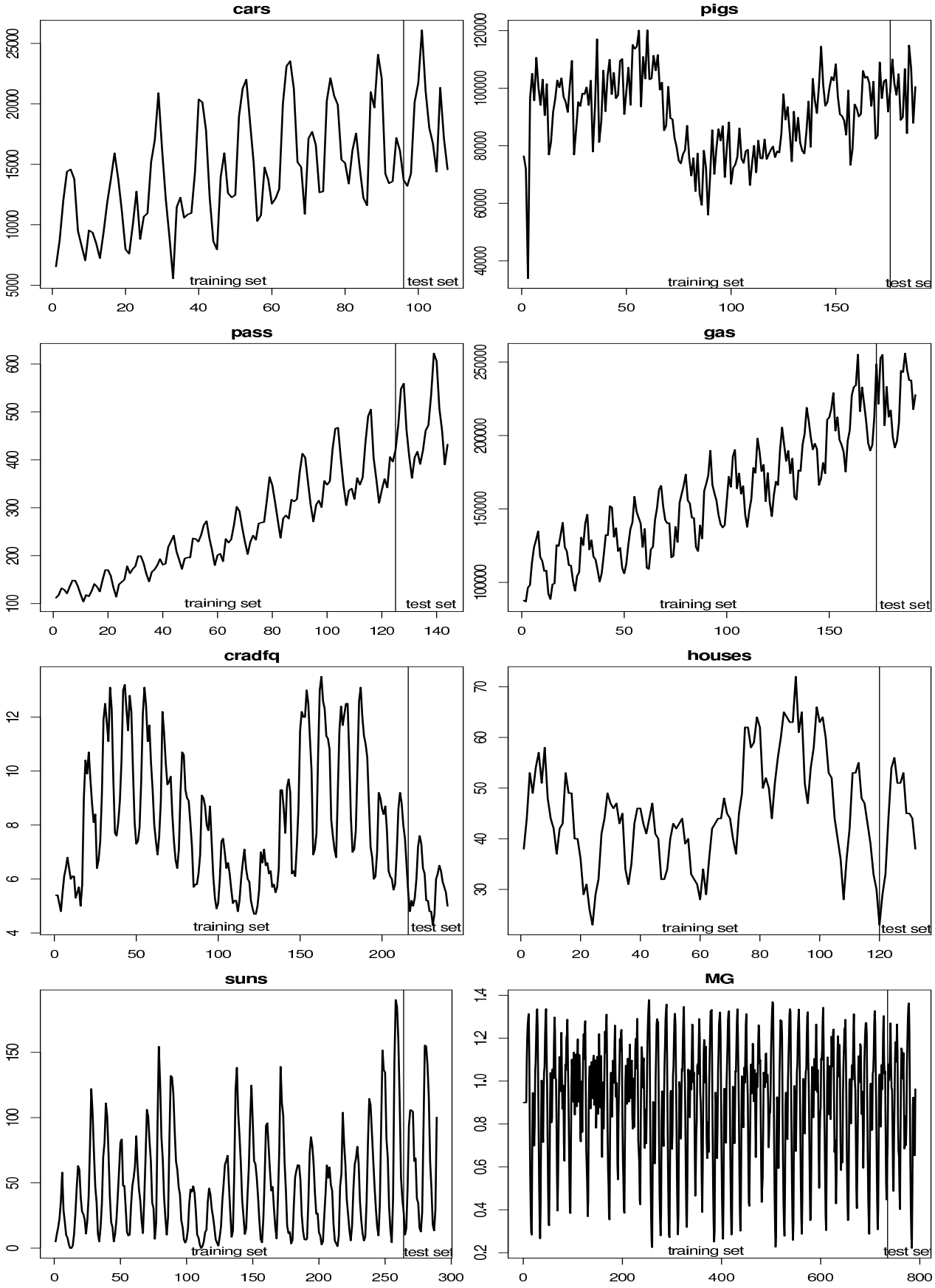

Fig. 1. Plots of the time series data (with training and test sets)

where e_t denotes the forecasting error at time t; y_t the desired value; $\hat{y}_{t,p}$ the predicted value for period t and computed at period p; P is the present time and N the number of forecasts. In all MSE, RSE and $SMAPE$ metrics, lower values indicate better forecasts. RSE and $SMAPE$ have the advantage of being scale independent. A value of RSE lower than 100% shows that the forecasting model is better than the naive average predictor, while the $SMAPE$ values range from 0% to 200%.

The holdout validation is commonly used to estimate the generalization capability of a model [18]. This method randomly partitions the data into training and test subsets. Due to the temporal nature of the forecasting domain, we adopt here a sequential (i.e. time ordered) holdout [19]. Also, when there is variance in the results (i.e. for NN), we apply 30 runs to the selected model and statistical confidence is given by the t-student test at the 95% confidence level [20].

C. Holt-Winters Method

The Holt-Winters (HW) is a popular forecasting technique from the family of Exponential Smoothing methods. The predictive model is based on underlying patterns, such as trends and seasonality, that are distinguished from random noise by averaging the historical values [3]. Its popularity is due to advantages such as reduced computational demand, simplicity of use and accuracy of the forecasts, specially with seasonal series. The multiplicative model is defined by [4]:

$$\begin{array}{ll} \text{Level} & S_t = \alpha \frac{y_t}{D_{t-K}} + (1-\alpha)(S_{t-1} + T_{t-1}) \\ \text{Trend} & T_t = \beta(S_t - S_{t-1}) + (1-\beta)T_{t-1} \\ \text{Seasonality} & D_t = \gamma \frac{y_t}{S_t} + (1-\gamma)D_{t-K} \\ & \hat{y}_{t+h,t} = (S_t + hT_t) \times D_{t-K+h} \end{array} \quad (3)$$

where S_t, T_t and D_t denote the level, trend and seasonal estimates and α, β and γ are the model parameters. When there is no seasonal component, the γ is discarded and the D_{t-K+h} factor in the last equation is replaced by the unity. To optimize the HW parameters, we adopt a grid search for the best training error (i.e. MSE), which is a common procedure within the forecasting field.

D. Neural Networks

Neural Networks (NNs) are innate candidates for forecasting due to their nonlinear and noise tolerance capabilities. The use of NNs for TSF began in the late eighties with encouraging results and the field has been consistently growing since [9][1][11][19].

Any regression algorithm (e.g. NN, SVM) can be applied to TSF by adopting a sliding time window, defined by the set of time lags $\{k_1, k_2, \ldots, k_I\}$ used to build a forecast. For a given time period t, the model inputs are $y_{t-k_I}, \ldots, y_{t-k_2}, y_{t-k_1}$ and the desired output is y_t. For example, let us consider the series $6_1, 10_2, 14_3, 18_4, 23_5$ (y_t values). If the $\{1, 3\}$ window is adopted, then two training examples can be created: $6, 14 \rightarrow 18$ and $10, 18 \rightarrow 23$. After training, the last known values are fed into the model and multi-step forecasts are built by iteratively using 1-ahead predictions as inputs [19].

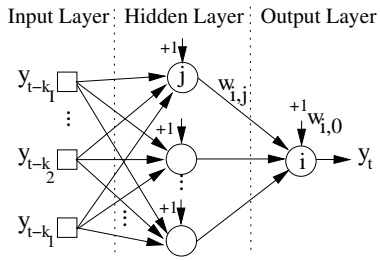

Fig. 3. Example of a multilayer perceptron

We use fully connected multilayer perceptrons (the most popular NN), with one hidden layer of H nodes and bias connections (Figure 3). To introduce nonlinearity, the logistic activation function is applied on the hidden nodes, while in the output node uses a linear function, to scale the range of the outputs [11]. The overall model is given in the form:

$$\hat{y}_{t,t-1} = w_{o,0} + \sum_{j=I+1}^{o-1} f(\sum_{i=1}^{I} y_{t-k_{I-i+1}} w_{j,i} + w_{j,0}) w_{o_j} \quad (4)$$

where $w_{i,j}$ denotes the weight of the connection from node j to i (if $j = 0$ then it is a bias connection), o denotes the output node and f the logistic function ($\frac{1}{1+e^{-x}}$).

Since NN training is not optimal, the final solution is dependent of the choice of starting weights. To solve this issue, the solution adopted is to use an NN ensemble, where R different networks are trained and the final prediction is given by the average of the individual predictions [5]. In general, ensembles are better than individual learners, provided that the errors made by the individual models are uncorrelated, a condition easily met with NNs, since the training algorithms are stochastic in nature [21].

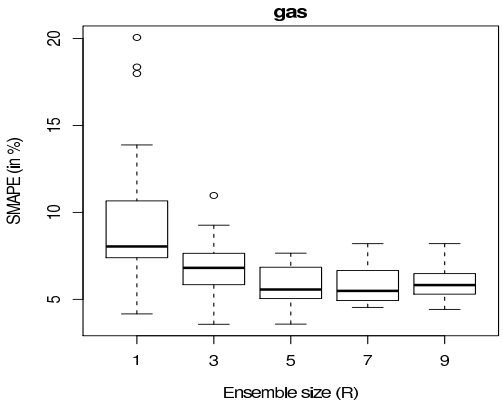

Fig. 4. Boxplots of the effect of increasing the NN ensemble size

To show the effect of increasing the ensemble size ($R \in \{1, 3, 5, 7, 9\}$, Figure 4), we selected series gas, the sliding time window $\{1, 2, \ldots, 13\}$, a NN with 13 inputs and 6 hidden nodes ($H = 6$). The training data (173 elements) was divided into training (with the first 154 values) and validation sets (last 19 values). We applied 30 runs for each ensemble and measured the forecasting error ($SMAPE$ values) on

the validation set. From the figure, it is clear that when R is increased, both the mean and standard deviation of the errors are reduced. We found that such effect was also consistent for other series. Since increasing R also increases the computational effort, in this work we fixed the ensemble size to a reasonable $R = 7$.

E. Support Vector Machines

Several works also proposed the use of Support Vector Machines (SVM) to the forecasting domain [10][12]. In SVM regression [8], the input $\mathbf{y} = (y_{t-k_I}, \ldots, y_{t-k_2}, y_{t-k_1})$ is transformed into a high m-dimensional feature space, by using a nonlinear mapping (ϕ) that does not need to be explicitly known but that depends of a kernel function. Then, the SVM algorithm finds the best linear separating hyperplane, tolerating a small error (ϵ) when fitting the data, in the feature space:

$$\hat{y}_{t,t-1} = w_0 + \sum_{i=1}^{m} w_i \phi_i(\mathbf{y}) \tag{5}$$

The ϵ-insensitive loss function sets an insensitive tube around the residuals and the tiny errors within the tube are discarded (Figure 5).

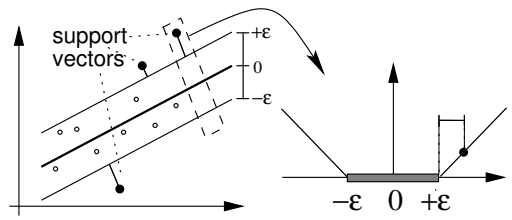

Fig. 5. Example of a linear SVM regression and the ϵ-insensitive loss function (adapted from [8])

We adopt the popular gaussian kernel, which presents less parameters than other kernels (e.g. polynomial) [22]: $exp(-\lambda||x-x'||^2), \lambda > 0$. The SVM performance is affected by three parameters: λ, ϵ and C (a trade-off between fitting the errors and the flatness of the mapping). To reduce the search space, the first two values are set using the heuristics [23]: $C = 3$ (for a standardized output) and $\epsilon = \hat{\sigma}/\sqrt{N}$, where $\hat{\sigma} = 1.5/N \times \sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ and $\hat{y}_i$ is the value predicted by a 3-nearest neighbor algorithm.

F. Variable and Model Selection

Given the setup adopted, the forecasting performance of the DM algorithms is affected by both time lag and model selection. A better generalization, due to the reduced input space, is achieved if only relevant time lags are fed into the models [12]. On the other hand, a NN with few hidden nodes has limited learning capabilities, while an excess value of H leads to overfitting. Regarding the SVM, the kernel parameter (λ) produces the highest impact in the SVM performance (when compared to C or ϵ), with values that are too large or too small leading to poor predictions.

Sensitivity analysis [13] is a procedure that is applied after the training phase and analyzes the model responses when the inputs are changed. Let $\hat{y}_{t-k}(j)$ denote the output obtained by holding all input variables at their average values except y_{t-k}, which varies through its entire range with $j \in \{1, \ldots, L\}$ levels. If a given input variable (y_{t-k}) is relevant then it should produce a high variance (V_k). Thus, its relative importance (R_k) can be given by:

$$\begin{array}{rcl} V_k & = & \sum_{j=1}^{L} (\hat{y}_{t-k}(j) - \overline{\hat{y}_{t-k}(j)})^2/(L-1) \\ R_k & = & V_k / \sum_{i=1}^{I} V_i \times 100\ (\%) \end{array} \tag{6}$$

This is a simple procedure that only measures single input variance and not interactions of inputs. Yet, even with this limitation, this computationally fast procedure has outperformed other more sophisticated algorithms (e.g. genetic algorithms) for variable selection [13].

To show how the sensitivity analysis works, Figure 6 plots the responses ($y-$axis) when the lags $k_1 = 1$, $k_5 = 5$ and $k_{13} = 13$ are varied through $L = 6$ levels for series cradfq (using only training data). In this example, we fitted a SVM with 13 inputs and $\lambda = 2^{-5}$. In the plot, it is clear that the first lag is most important one (leading to a higher V_k), while the lag $k_5 = 5$ hardly affects the SVM responses.

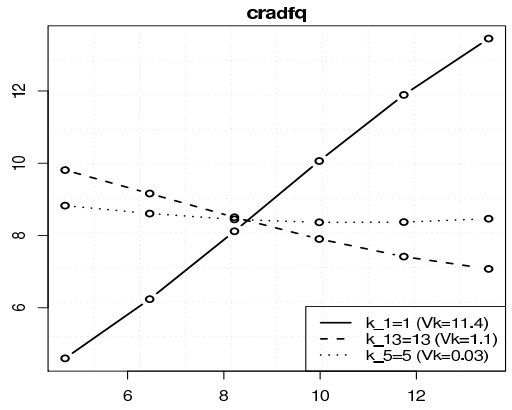

Fig. 6. Example of the sensitivity analysis procedure

We propose a simultaneous variable and model selection procedure for multi-step ahead forecasting (Algorithm 1). The method starts with all time lags and iteratively deletes one input until there are no time lags. The sensitivity analysis is used to select the least relevant lag to be deleted in each iteration, allowing a reduction of the computational effort by a factor of I when compared to the standard backward procedure. During a given iteration, a grid search is used to find the best model hyperparameter ($p \in \{p_1, \ldots, p_n\}$). The training data is divided into training and validation sets. The former, with $2/3$ of the training data, is used to train the DM model. The latter, with the remaining $1/3$, is used to estimate the model generalization capabilities. After the variable and model selection phase, the final model is retrained using all training data.

Algorithm 1 Variable and model selection for TSF

Input: K_{max} – the maximum time lag, S – the time series, M – the DM model (e.g. NN or SVM), $\{p_1, \ldots, p_n\}$ – the model parameter grid search set

1. $G_{bw} \leftarrow \inf$; $W \leftarrow \{1, 2, \ldots, K_{max}\}$ (time window)
2. **while** $\#W > 0$
3. **do** $D \leftarrow$ create training data using S and W
4. split D into training (T) and validation sets (V)
5. $G_{bp} \leftarrow \inf$
6. **for** $p \leftarrow \{p_1, \ldots, p_n\}$
7. **do** $M_{W,p} \leftarrow$ fit model M with T and p
8. $G_p \leftarrow$ error estimate of $M_{W,p}$ in V
9. **if** $G_p < G_{bp}$
10. **then** $G_{bp} \leftarrow G_p$; $p_b \leftarrow p$
11. $R_W \leftarrow$ compute relative importances of M_{W,p_b}
12. $R_{min} \leftarrow \min(R_W)$
13. **if** $G_{bp} < G_{bw}$
14. **then** $G_{bw} \leftarrow G_{bp}$; $W_b \leftarrow W$; $p_{bw} \leftarrow p_b$
15. $W \leftarrow$ delete R_{min} from W
16. $D \leftarrow$ create training data using S and W_b
17. $M_{W_b,p_{bw}} \leftarrow$ fit model of type M with D and p_{bw}
18. **return** $M_{W_b,p_{bw}}$

III. Experiments and Results

All experiments reported in this work were written in the **R** environment, a open-source tool for statistical and data analysis [24]. In particular, we adopted the rminer library [25], which facilitates the use of NN and SVM in R. We adopted the default R parameters for all models. The HW models are based on the implementation provided by the **stats R** package. Before fitting the NN/SVM models, the data was first standardized to a zero mean and one standard deviation [26]. In the training stage, the NN initial weights were randomly set within the range $[-0.7, 0.7]$ and then trained using 100 epochs of the BFGS algorithm (**nnet** package), while the SVM fit is based on the Sequential Minimal Optimization implementation provided by LIBSVM (**kernlab** package). After training, the NN/SVM outputs were post-processed with the inverse of the standardized transform.

The HW parameters were optimized using a 0.01 grid search and are presented in Table III. The seasonal coefficient (γ) is used by all series. The trended component (β) was detected in all cases except cars, suns and MG.

TABLE III
HOLT-WINTERS FORECASTING MODELS

Series	α	β	γ
cars	0.30	0.00	0.31
pigs	0.33	0.01	0.40
pass	0.33	0.03	1.00
gas	0.03	0.28	0.34
houses	0.79	0.03	0.89
cradfq	0.44	0.06	0.52
suns	0.92	0.00	1.00
MG	0.93	0.00	1.00

Regarding NN and SVM, we set $K_{max} = K + 1$ (e.g. $K_{max} = 13$ for the cars series). The intention is to include all up to the seasonal lag plus an additional one that may be relevant for trended series. The hyperparameters were searched within the ranges $H \in \{0, 1, \ldots, 9\}$ [5] and $\lambda \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$ [22] (in a total of 10 searches for each NN/SVM). We used the MSE metric during the model selection stage. The rationale is to reduce extreme errors that may highly affect multi-step ahead forecasts. The sensitivity analysis parameter was set to $L = 6$. The best forecasting models, as detected by the variable and model selection algorithm, are shown in Tables IV and V. Linear NN models were selected for series pigs, pass, gas and suns, while the highest number of $H = 9$ was selected for MG. The λ values range from 2^{-9} to 2^{-3}. Regarding the selected time lags, the seasonal period (K) was kept for all models, except for sun (SVM) and MG (NN and SVM) series This is not a surprising result, as the autocorrelation analysis is less valid for nonlinear series. In some cases, few or no lags were deleted (e.g. series cars), while a substantial reduction was performed for the last two series (e.g. 9 lags of the suns series were deleted for both NN and SVM). As an example, Figure 7 plots the best validation error (in terms of MSE/MSE_0, where MSE_0 is the error when all time lags are used, $y-$axis) during the search for the best set of time lags for series MG and SVM. In this case, after 7 deletions, an improvement of 64 pp is achieved.

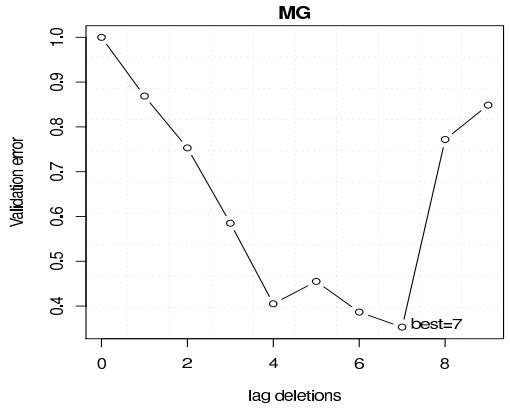

Fig. 7. Example of search for the best MG time lags

TABLE IV
BEST NN FORECASTING MODELS

Series	H	Window	#lag deletions
cars	1	{1,2,...,13}	0
pigs	0	{1,2,...,13}	0
pass	0	{1,2,...,13}	0
gas	0	{2,3,4,7,9,11,12}	6
houses	1	{1,2,3,4,5,7,8,10,11,12,13}	2
cradfq	4	{1,2,8,9,12,13}	7
suns	0	{1,2,3,9,10}	9
MG	9	{1,2,4,5,6,7,8,11,12}	9

TABLE V

BEST SVM FORECASTING MODELS

Series	λ	Window	#lag deletions
cars	2^{-7}	{1,2,...,12}	1
pigs	2^{-7}	{1,2,3,4,6,8,10,12}	5
pass	2^{-7}	{6,7,10,12,13}	8
gas	2^{-9}	{1,2,3,4,5,6,7,9,10,11,12}	2
houses	2^{-7}	{1,3,4,6,7,8,9,10,11,12,13}	2
cradfq	2^{-5}	{1,2,4,6,7,8,9,10,11,12,13}	2
suns	2^{-7}	{1,2,4,8,11}	9
MG	2^{-3}	{2,4,5,6,7,8,9,10,11,12,15}	7

After the model selection stage, the forecasts were performed for each method, using a lead time from h =1 to N (the N values are shown in Table II); i.e. no test set data was used to compute the predictions. In case of the NN, 30 runs were applied to the selected model and the results are presented in terms of the average and respective 95% confidence intervals.

Tables VI and VII show the forecasting errors (over the test data) for each method. In the tables, the best values are in **bold**. Paired t-tests were used to compare the NN results with the remaining methods for each series. In all cases, the differences are significative (i.e. p-value< 0.05). The last row denotes the average for each method, when the considering all eight series. For table VII, the average was computed using RSE (and not MSE) values, since RSE errors are scale independent.

In terms of the $SMAPE$ metric (Table VI), SVM is the best method in 5 of the 8 series, while HW and NN achieved the best results in 2 and 1 cases, respectively. When comparing NN and HW, the former outperforms the latter in 6 of the 8 series (in particular, HW performs poorly for series houses, suns and MG). Thus, the average $SMAPE$ for all eight series favors NN when compared to HW. It is interesting to notice that the MSE results (Table VII) show similar rankings for the forecasting methods. The few exceptions are series pigs, where SVM is better than HW, and cradfq, where HW outperforms NN. In some cases, the differences between the errors are higher when compared to Table VI. For instance, the SVM performance is much better than NN and HW for series cradfq. Overall, SVM presents the best results (best method in 5 series and lowest RSE average). Next comes NN, which outperforms HW in 5 series and presents the second lowest RSE average.

For demonstrative purposes, Figure 8 plots two examples of the NN/SVM forecasts (x-axis denotes the lead time, i.e. h). To improve clarity, only the last 20 predictions are presented for MG. Both graphs show a good fit by the SVM forecasts, in particular for the MG series. Another relevant issue is related with the computational complexity. The proposed approach does not require heavy computation. With an Intel Core 2 Duo 2.53 GHz processor, the variable and model selection algorithm processing times for the two examples were: gas – 14s for SVM and 54s for NN; MG – 84s for SVM and 178s for NN.

TABLE VI

COMPARISON OF THE FORECASTING ERRORS ($SMAPE$ VALUES, IN %)

Series	HW	NN	SVM
cars	**7.98**	9.25±0.01	9.72
pigs	6.82	**6.30**±0.00	7.20
pass	**3.60**	4.88±0.00	9.08
gas	7.89	7.67±0.00	**4.08**
houses	42.53	15.49±0.02	**10.68**
cradfq	24.64	22.21±1.36	**10.71**
suns	75.21	59.80±0.00	**42.33**
MG	32.45	3.72±0.75	**2.11**
Average	25.14	16.17	**11.99**

TABLE VII

COMPARISON OF THE FORECASTING ERRORS (MSE VALUES)

Series	HW	NN	SVM
cars	$\mathbf{3.46}\times10^{6}$	$3.78\pm0.00\times10^{6}$	4.05×10^{6}
pigs	7.47×10^{7}	$\mathbf{5.74}\pm0.00\times10^{7}$	6.85×10^{7}
pass	$\mathbf{3.39}\times10^{2}$	$5.60\pm0.00\times10^{2}$	28.93×10^{2}
gas	4.10×10^{8}	$4.05\pm0.00\times10^{8}$	$\mathbf{1.32}\times10^{8}$
houses	30.9×10^{1}	$5.70\pm0.01\times10^{1}$	$\mathbf{2.93}\times10^{1}$
cradfq	17.29×10^{-1}	$29.07\pm3.81\times10^{-1}$	$\mathbf{6.65}\times10^{-1}$
sun	2.64×10^{3}	$1.56\pm0.00\times10^{3}$	$\mathbf{1.03}\times10^{3}$
MG	103.55×10^{-3}	$1.65\pm0.63\times10^{-3}$	$\mathbf{0.46}\times10^{-3}$
Average[a]	147.9%	94.5%	**47.2**%

[a] – Average of RSE values.

IV. CONCLUSIONS

Multi-step ahead forecasting is an important tool to support tactical decisions, such as planning production resources. Such type of predictions can be achieved using Time Series Forecasting (TSF) methods, which predict the future based on past values of the same event. Several classical TSF methods have been proposed, such as the Holt-Winters (HW) model, which is popular for seasonal series. An interesting alternative comes from the use of more powerful and flexible algorithms, such as Neural Networks (NNs) and Support Vector Machines (SVMs) For TSF, the performance of these algorithms depends on a correct setting of inputs (i.e. time lags used to build the training cases) and hyperparameters (e.g. number of hidden nodes of the NN architecture or SVM kernel parameter).

In this work we proposed a computationally efficient variable and model selection algorithm for multi-step ahead forecasting, where time lag selection is based on a backward selection procedure that is guided by a sensitivity analysis and model selection is based on a grid-search. We applied such algorithm to both NN and SVM techniques over eight seasonal time series and the obtained multi-step forecasts were analyzed under two error criteria: $SMAPE$ and MSE. Competitive results were achieved, with SVM being the best method in 5 of the 8 series. The NN was ranked as the second best method, outperforming HW in 6 ($SMAPE$) and 5 (MSE) of the series, while presenting also a lower average (over all series) forecasting error. In particular, high quality results were achieved by SVM for the Mackey-Glass chaotic series.

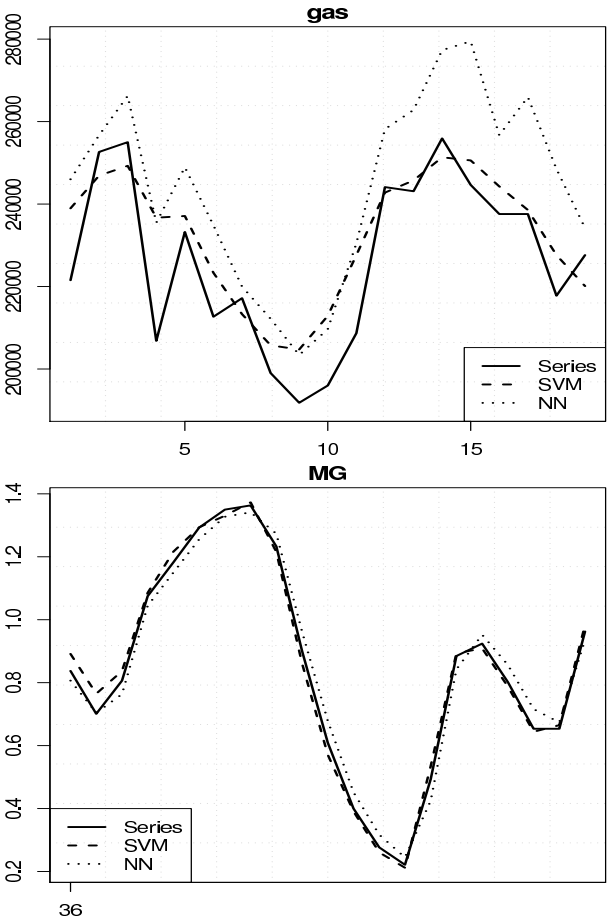

Fig. 8. Example of the multi-step ahead forecasts for series gas and MG

As future work, we intend to explore additional forecasting methods, such as Random Forests [27]. We also intend to apply the proposed forecasting techniques to real-world applications, such as Internet traffic and spam email arrival prediction.

ACKNOWLEDGMENT

This work is supported by FCT project PTDC/EIA/64541/2006.

REFERENCES

[1] X. Ding, S. Canu, and T. Denoeux, "Neural network based models for forecasting," in *Proceedings of ADT'95*, 1995.

[2] C. Chatfield, *The Analysis of Time Series - An Introduction*. UK: Chapman and Hall, 1989.

[3] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Management Science*, vol. 6, pp. 324–342, 1960.

[4] S. Makridakis, S. Weelwright, and R. Hyndman, *Forecasting: Methods and Applications*, 3rd ed. John Wiley & Sons, New York, USA, 1998.

[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. NY, USA: Springer-Verlag, 2008.

[6] D. Rumelhart, G. Hinton, and R. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, D. Rulmelhart and J. McClelland, Eds., vol. 1, MIT Press, Cambridge MA, 1986, pp. 318–362.

[7] B. Boser, I. Guyon, and V. Vapnik, "A training algoritm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. San Mateo, CA: Morgan Kaufmann, 1992, pp. 144–152.

[8] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[9] A. Lapedes and R. Farber, "Non-Linear Signal Processing Using Neural Networks: Prediction and System Modelling," Los Alamos National Laboratory, USA, Technical Report LA-UR-87-2662, 1987.

[10] K. Miiller, A. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, and V. Vapnik, "Predicting time series with support vector machines," in *Proceedings of the 7th International Conference on Artificial Neural Networks*. Springer, 1997, pp. 999–1004.

[11] P. Cortez, M. Rocha, and J. Neves, "Time Series Forecasting by Evolutionary Neural Networks," chapter III, *Artificial Neural Networks in Real-Life Applications*, Idea Group Publishing, USA, pages 47–70, 2006.

[12] W. He, Z. Wang, and H. Jiang, "Model optimizing and feature selecting for support vector regression in time series forecasting," *Neurocomputing*, vol. 72, no. 1-3, pp. 600–611, 2008.

[13] R. Kewley, M. Embrechts, and C. Breneman, "Data Strip Mining for the Virtual Design of Pharmaceuticals with Neural Networks," *IEEE Trans Neural Networks*, vol. 11, no. 3, pp. 668–679, May 2000.

[14] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[15] R. Hyndman, *Time Series Data Library*. http://robjhyndman.com/TSDL/, January, 2010.

[16] L. Glass and M. Mackey, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287–289, 1977.

[17] G. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden Day, San Francisco, USA, 1976.

[18] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Volume 2, Montreal, Quebec, Canada, Morgan Kaufmann, Aug. 1995.

[19] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet Traffic Forecasting using Neural Networks," in *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN 2006)*. Vancouver, Canada: IEEE, Jul. 2006, pp. 4942–4949.

[20] A. Flexer, "Statistical Evaluation of Neural Networks Experiments: Minimum Requirements and Current Practice," in *Proceedings of the 13th European Meeting on Cybernetics and Systems Research*, vol. 2, Vienna, Austria, 1996, pp. 1005–1008.

[21] T. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems, Lecture Notes in Computer Science 1857*, J. Kittler and F. Roli, Eds. Springer-Verlag, 2000, pp. 1–15.

[22] W. Wang, Z. Xu, W. Lu, and X. Zhang, "Determination of the spread parameter in the Gaussian kernel for classification and regression," *Neurocomputing*, vol. 55, no. 3, pp. 643–663, 2003.

[23] V. Cherkassy and Y. Ma, "Practical Selection of SVM Parameters and Noise Estimation for SVM Regression," *Neural Networks*, vol. 17, no. 1, pp. 113–126, 2004.

[24] R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-00-3, http://www.R-project.org, 2009.

[25] P. Cortez, "Data Mining with Neural Networks and Support Vector Machines using the R/rminer Tool," in *Advances in Data Mining, Proceedings of 10th Industrial Conference on Data Mining*, P. Perner, Ed. Berlin, Germany: Springer, Jul. 2010.

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. NY, USA: Springer-Verlag, 2001.

[27] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.