Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition Letters 25 (2004) 911–921

Pattern Recognition Letters

www.elsevier.com/locate/patrec

A Modal Symbolic Classifier for selecting time series models

Ricardo B.C. Prudêncio *, Teresa B. Ludermir [a], Francisco de A.T. de Carvalho [a]

[a] *Centro de Informática, Universidade Federal de Pernambuco, Caixa Postal 7851, CEP 50732-970, Recife (PE), Brazil*

Received 16 July 2003; received in revised form 5 February 2004
Available online 19 March 2004

Abstract

The selection of a good model for forecasting a time series is a task that involves experience and knowledge. Employing machine learning algorithms is a promising approach to acquiring knowledge in regards to this task. A supervised classification method originating from the symbolic data analysis field is proposed for the model selection problem. This method was applied in the task of selecting between two widespread models, and compared to other learning algorithms. To date, it has obtained the lowest classification errors among all the tested algorithms.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Time series forecasting; Model selection; Machine learning; Symbolic data analysis; Modal Symbolic Classifier

1. Introduction

Time series forecasting has been used in several real world problems to support the decision-making process (Montgomery et al., 1990). A number of different models can be used to forecast a time series. However, there is no single model considered to be the best in all cases. Selecting the most adequate model from a set of candidate models for a given time series can be a difficult task depending on the candidate models involved, as well as on the characteristics of the time series.

The development of expert systems is an approach that formalizes knowledge for model selection (Collopy and Armstrong, 1992). In this context, each expert rule associates time series features to the best model for forecasting. The main drawback of these systems is that the process of knowledge acquisition depends on experts which are often scarce and expensive. An alternative solution is the use of machine learning algorithms (Arinze, 1994), which treat the model selection as a classification problem where the class attribute is the best forecasting model and the predictors are time series features. In this solution, machine learning algorithms learn to relate the time series features and the best models for predicting the time series.

Symbolic data analysis (SDA) is a domain in the area of knowledge discovery and data management related to multivariate analysis, pattern recognition, databases and artificial intelligence. It aims to provide suitable methods (clustering, factorial techniques, decision trees, etc.) for managing

* Corresponding author. Tel.: +55-81-2126-8430; fax: +55-81-2126-8438.
E-mail addresses: rbcp@cin.ufpe.br (R.B.C. Prudêncio), tbl@cin.ufpe.br (T.B. Ludermir), fatc@cin.ufpe.br (F.de.A.T. de Carvalho).

0167-8655/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2004.02.004

aggregated data described through multi-valued variables where there are sets of categories, intervals, or weight (probability) distributions in the cells of the data table (for more details about SDA, see www.jsda.unina2.it). In this so-called symbolic data table, the rows are the symbolic objects and the columns are the symbolic variables (Bock and Diday, 2000). A symbolic variable is defined according to its type of domain. For example, for an object, an interval variable takes an interval of $\Re$ (the set of real numbers).

A Modal Symbolic (MS) Classifier, developed in the framework of the SDA field, is proposed for selecting between two time series models: simple Exponential Smoothing (Brown, 1963) and the Time-Delay Neural Network (Lang and Hinton, 1988). This symbolic classifier is able not only to suggest a single model but also to provide degrees of confidence for the candidate models. The MS Classifier was compared to traditional learning algorithms (decision trees and the *k*-nearest neighbours algorithm (*k*-NN)) for the same problem. The selection error rate obtained by the MS Classifier (34.34%) was lower than the error rate obtained by both the *k*-NN algorithm (45.45%) and by the decision trees (37.37%).

In Section 2, some approaches to model selection are presented. Section 3 brings a brief explanation of the use of MS Classifier for time series model selection, as well as a description of the algorithm. Section 4 brings the selection task being tackled by the algorithm, followed by Section 5, where the experiments and results are reported. Finally, Section 6 presents some conclusions and the proposed future projects.

2. Time series model selection

Over the years researchers have developed different techniques for forecasting time series. Among these techniques, the family of Exponential Smoothing models (Montgomery et al., 1990), the Box–Jenkins models (Box and Jenkins, 1970) and several Artificial Neural Networks models (Dorffner, 1996) have been used often with good results. Despite the diversity of models, empirical research has shown that there is no single model that performs better than the others in all series (Collopy and Armstrong, 1992). Selecting the most adequate models for describing individual time series can strongly improve the performance of the forecasting process.

Several approaches to dealing with the model selection problem can be identified. The most straightforward way is to perform a tournament where each candidate model is calibrated using a detached part of the time series data and then tested on the remaining data. The model which obtains the best results on the test data (based on a forecast-error criteria) is then chosen for forecasting the futures of the time series. Despite its simplicity, this solution can be very costly when there are a large number of candidate models or time series to forecast. Furthermore, some authors have reported a poor gain on the accuracy from applying tournament methods compared to unselectively applying the candidate models (Schnaars, 1986; Fildes, 1989; Tashman and Kruk, 1996). Tournaments are blind regarding the characteristics of the time series, and hence, the presence of certain factors, such as scanty available data, a lack of stability and outliers, can severely harm the performance.

In (Collopy and Armstrong, 1992), the authors emphasized the need for incorporating knowledge into the model selection process by associating time series characteristics to the model performance. One approach that can formalize knowledge in a reusable way is to use expert systems, such as the landmark Rule-Based Forecasting system (Collopy and Armstrong, 1992). In this work the authors implemented an expert system with 99 rules used to configure and combine four widespread time series models. The rule base was developed from the guidelines provided by five human experts through the analysis of real problems. The authors used time series features in the rules, such as level discontinuities, insignificant basic trend, and unusual last observation, among others.

Although expert systems can express knowledge in a practical and reusable way, the process of knowledge acquisition depends on experts, who are often scarce and expensive (Turban, 1992). In these cases, machine learning algorithms can provide an interesting alternative solution for acquiring knowledge. These algorithms can be used to automatically acquire knowledge from data, leading to a reduced need for experts and a potential performance improvement (Arinze, 1994).

The use of learning algorithms for model selection was originally proposed by Arinze (1994), and adopted in other works (Chu and Widjaja, 1994; Venkatachalan and Sohl, 1999; Prudêncio, 2002, 2003). In general, these works treat the model selection as a classification problem where a learning algorithm is used as the classifier. Each training example consists of a description of a time series according to some of its intrinsic characteristics (e.g., descriptive statistics extracted from the time series) associated to a class attribute representing the best candidate model for forecasting this series. A set of such examples serves as a basis for the learning process. The main limitation of these previous works is that they used learners to return just the best model among the set of candidates. As it will be seen, the use of the MS Classifier can provide a more informative solution.

3. The proposed model selection approach

The work described here uses a symbolic classifier, the Modal Symbolic (MS) Classifier, applied to the problem of time series model selection. This classifier is conceived in the framework of an area in knowledge discovery, the Symbolic Data Analysis (Bock and Diday, 2000), which provides tools for managing complex, aggregate and high-level data described by multi-valued variables (symbolic variables), where the entries of a data table are sets of categories, intervals, or weight (probability) distributions (symbolic data).

Suitable versions of the MS Classifier were previously used for image processing (De Carvalho et al., 2001) and information filtering (Bezerra et al., 2002; De Carvalho and Bezerra, 2003), and the results were successful both in terms of accuracy and execution time.

Fig. 1 presents the general architecture of the system proposed for *time series* model selection.

This system has two different phases: training and use. In the training phase, the MS Classifier acquires knowledge from a set of training examples stored in the Database (DB). Each training example stores the features of a different time series and the candidate model that obtained the best forecasting results for the series. The time series features are pre-defined descriptive statistics automatically calculated from the time series data, such as the number of observations and the coefficient of variation, among others. Each training example actually contains the experience obtained in the past during the forecasting process of a specific time series. The MS Classifier generalizes the experience stored in the training examples by associating time series features to the most adequate models. The acquired knowledge may be refined as more examples are available in the DB (the system user, or the developer, provides a new time series and the best model for forecasting it).

In the use phase, the system receives a time series as input. This is a set of observations generated sequentially in time. Given the time series, the Feature Extractor (FE) module extracts the values of the time series features. According to the matching between the time series features and the modal symbolic description of the pre-defined

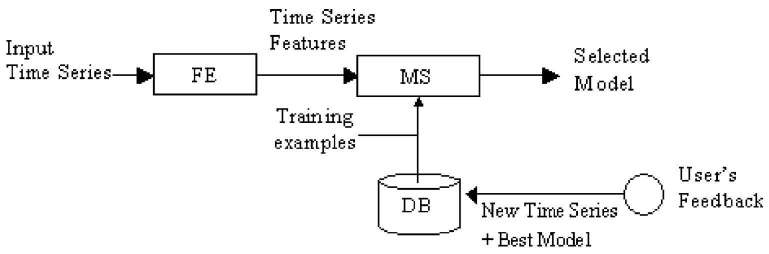

Fig. 1. System's architecture.

candidate models, the MS Classifier suggests a model chosen from the set of candidates for forecasting the given time series.

3.1. The Modal Symbolic Classifier

In this section the MS Classifier, which is based on modal symbolic descriptions, is described for selecting suitable time series models.

Let $E = \{e_1, \ldots, e_n\}$ be the set of examples, where each example stores the values of p features $X_1, \ldots, X_p$ for a particular time series and the value of an attribute C, which indicates the best candidate model, among K, for forecasting the series.

$D = \{c_1, \ldots, c_K\}$ is the domain of the categorical variable C where each class value $c_k \in D$ represents a candidate forecasting model. Each feature X_j $(j = 1, \ldots, p)$ has a domain D_j and represents a different descriptive statistic for time series. In this way each example $e_i \in E$ $(i = 1, \ldots, n)$ is represented as a vector of feature values $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^p, C(e_i))$ where $x_i^j \in D_j$ $(j = 1, \ldots, p)$ and $C(e_i) = c_k \in D$.

The selection of the time series models is accomplished according to the construction of the MS Classifier, involving the following main steps:

(1) *Learning step*. Construction of a modal symbolic description for each time series model:
   (a) *Pre-processing*. Generation of a new set of examples through the discretization of continuous numeric features.
   (b) *Generalization*. Using the pre-processed examples to compute the parameters of each modal symbolic description.
(2) *Allocation step*. Assignment of a new time series to the suitable time series model. This is done by measuring the matching between the time series features and the modal symbolic descriptions. The model that presents the greatest matching value is then selected.

3.1.1. Learning step

The aim of this step is to construct a modal symbolic description (a vector of modal symbolic data) for each candidate model that synthesizes the role information given by the examples associated to this model. The learning step has two sub-steps: pre-processing and generalization.

3.1.1.1. Pre-processing. The attributes selected in this work for describing the time series are numeric (e.g., length of the time series) or binary (e.g., presence of significant autocorrelations). In order to have an homogeneous set of variables, we changed the scale of all numeric variables X_j in such a way that the domain $D_j \subseteq \Re$ (the set of real numbers) becomes $D'_j = \{0, 1\}$. This is accomplished according to the entropy-based method for discretization of continuous numeric variables, developed by Fayyad and Irani (1993). This has advantages over other methods since it takes classes into account during the process of discretization (Witten and Frank, 2000). The entropy-based discretization used in the pre-processing sub-step is described below.

For an example e_i, described by the feature vector $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^p, C(e_i))$, the continuous numeric value $X_j(e_i) = x_i^j$ becomes

$$X_j(e_i) = \tilde{x}_i^j = \begin{cases} 1 & \text{if } x_i^j \geqslant T_j \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

where $T_j \in D_j$ is a threshold. As it will be seen just below, the thresholds T_j are defined in order to reduce the entropy of the training examples taking into account the class attribute C.

Let the vectors of feature values be $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, describing the examples, where $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^p, C(e_i))$, $i = 1, \ldots, n$. For each numeric variable X_j, let $X_j(E) = \{x_i^j \in D_j \,|\, x_i^j = X_j(e_i), i = 1, \ldots, n\}$, $E_i = \{e_l \in E \,|\, X_j(e_l) \geqslant x_i^j\}$ and $\overline{E_i} = \{e_l \in E \,|\, X_j(e_l) < x_i^j\}$, $i = 1, \ldots, n$. Then, $E_i \cap \overline{E_i} = \emptyset$ and $E_i \cup \overline{E_i} = E$.

Also, let $F_k = \{e_i \in E \,|\, C(e_i) = c_k\}$, $k = 1, \ldots, K$. Of course, $F_k \cap F'_k = \emptyset$, if $k \neq k'$ and $\bigcup_{k=1}^{K} F_k = E$.

The $\mathrm{Gain}(X_j, x_i^j, E)$ being defined as

$$\mathrm{Gain}(X_j, x_i^j, E) = -\frac{|E_i|}{|E|} * \mathrm{Ent}(C \,|\, E_i, E) - \frac{|\overline{E_i}|}{|E|} * \mathrm{Ent}(C \,|\, \overline{E_i}, E) \qquad (2)$$

where

$$\mathrm{Ent}(C \,|\, E_i, E) = \sum_{k=1}^{K} -\frac{|F_k \cap E_i|}{|E|} \log_2 \frac{|F_k \cap E_i|}{|E|} \qquad (3)$$

the threshold T_j is defined as the value x_i^j which maximizes the *information gain*:

$$T_j = \underset{x_i^j \in X_j(E)}{\arg\max}\, \text{Gain}(X_j, x_i^j, E) \tag{4}$$

In this way, now each example e_i $(i = 1, \ldots, n)$ is described by a vector of binary values and the class attribute: $\tilde{\boldsymbol{x}}_i = (\tilde{x}_i^1, \ldots, \tilde{x}_i^p, C(e_i))$, $\tilde{x}_i^j = X_j(e_i) \in D'_j = \{0, 1\}$, $j = 1, \ldots, p$.

3.1.1.2. Generalization. This step seeks to construct a suitable modal symbolic description for each time series model that summarizes all information given by the examples that have this model as the best candidate.

A *modal variable Y* with domain D is defined in a set $\Omega = \{a, b, \ldots\}$ of objects. These objects may be, for instance, single individuals (examples) or groups (or classes) of individuals. A modal variable is a multi-state variable where for each object $a \in \Omega$, it gives a subset of its domain $S(a) \subseteq D$, and for each category m of this subset, it is given a weight $w(m)$ that indicates how relevant m is for a. Formally, $Y(a) = (S(a), q(a))$ where $q(a)$ is a weight distribution defined in $S(a) \subseteq D$ such that a weight $w(m)$ corresponds to each category $m \in S(a)$. $S(a)$ is the support of the measure $q(a)$ in the domain D.

A modal symbolic description (Bock, 2000) is a vector where there is a weight distribution in each component given by a modal symbolic variable.

Let u_k be a candidate time series model described by p modal symbolic variables $\{Y_1, \ldots, Y_p\}$ and let $\boldsymbol{y}_k = (y_k^1, \ldots, y_k^p)$ be its modal symbolic description, where $y_k^j = Y_j(u_k) = (S_j(u_k), q_j(u_k))$, $j = 1, \ldots, p$, with $S_j(u_k) = D'_j = \{0, 1\}$ being the support of the weight distribution $q_j(u_k) = \{w_{kj}(0), w_{kj}(1)\}$, and $w_{kj}(0)$ and $w_{kj}(1)$ being the weight associated, respectively, to the categories 0 and 1.

Let $G_j^0 = \{e_i \in E \,|\, \tilde{x}_i^j = X_j(e_i) = 0\}$ and $G_j^1 = \{e_i \in E \,|\, \tilde{x}_i^j = X_j(e_i) = 1\}$, $j = 1, \ldots, p$. G_j^0 and G_j^1 are the subsets of examples where the variable X_j after pre-processing takes as value, respectively, 0 and 1. Of course, $G_j^0 \cap G_j^1 = \emptyset$ and $G_j^0 \cup G_j^1 = E$.

The weights $w_{kj}(0)$ and $w_{kj}(1)$ $(k = 1, \ldots, K$ and $j = 1 \ldots, p)$ are the frequency of examples associated to the candidate time series model u_k in which the variable X_j takes as value, respectively, 0 and 1. They are evaluated as:

$$w_{kj}(0) = \frac{|G_j^0 \cap F_k|}{|F_k|} \quad \text{and} \quad w_{kj}(1) = \frac{|G_j^1 \cap F_k|}{|F_k|} \tag{5}$$

3.1.2. Allocation step

The comparison between a new time series and a candidate time series model is achieved by way of a matching function. This matching function accomplishes this comparison variable-wise first by taking content differences into account. This new time series is affected to a candidate time series model which furnishes the greatest value of comparison given by the matching function.

Let $\tilde{\boldsymbol{x}} = (\tilde{x}^1, \ldots, \tilde{x}^p)$, where $\tilde{x}^j \in \{0, 1\}$, $j = 1, \ldots, p$, be the description of a new time series z and $\boldsymbol{y}_k = (y_k^1, \ldots, y_k^p)$, where $y_k^j = Y_j(u_k) = (\{0, 1\}, \{w_{kj}(0), w_{kj}(1)\})$, $j = 1, \ldots, p$, be the modal symbolic description of the candidate time series model u_k. The comparison between z and u_k is achieved by the way of the following similarity matching function:

$$\text{Match}(\tilde{\boldsymbol{x}}, \boldsymbol{y}_k) = \frac{1}{p} \sum_{j=1}^{p} w_{kj}(\tilde{x}^j) \tag{6}$$

The affectation of the new series z to the candidate model u_{k^*}, $k^* \in \{1, \ldots, K\}$, is accomplished according to the following rule: allocate z to u_{k^*} such that

$$k^* = \underset{k=1,\ldots,K}{\arg\max}\, \text{Match}(\tilde{\boldsymbol{x}}, \boldsymbol{y}_k) \tag{7}$$

The MS classifier is a more informative solution when compared to previous work that used learners for model selection. The MS Classifier is able not only to select a single best model for a given series, but also to provide a degree of confidence to each candidate model. The more similar is a time series description $(\tilde{\boldsymbol{x}})$ to the modal symbolic description $(\boldsymbol{y}_k)$ of a specific model, the more confident the model is in forecasting the series. In the present system implementation, this information was used just to return the model that obtained the best matching value. However, one can also obtain a ranking of models by sorting the matching values.

If enough resources are available, more than one model may be used for forecasting a time series. In such a case, the user can select the most confident models according to the matching values.

3.1.3. Schema of the MS Classifier

In this section, a specific schema of the MS Classifier is provided.

(1) Initialization:
Let $E = \{e_1, \ldots, e_n\}$ be the set of examples and let $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^p, C(e_i))$, where $x_i^j = X_j(e_i)$ $(j = 1, \ldots, p)$ and $(i = 1, \ldots, n)$, be their corresponding descriptions.

(2) Learning step:

(a) Pre-processing

```
For $j = 1, \ldots, p$ do
  if $D_j \subseteq \Re$ then
    $T_j \leftarrow \arg\max_{x_i^j \in X_j(E)} \mathrm{Gain}(X_j, x_i^j, E)$
  end
end
For $i = 1, \ldots, n$ do
  For $j = 1, \ldots, p$ do
    if $D_j \subseteq \Re$ then
      if $x_i^j \geqslant T_j$ then
        $X_j(e_i) = \tilde{x}_i^j \leftarrow 1$
      else
        $X_j(e_i) = \tilde{x}_i^j \leftarrow 0$
      end
    end
  end
  $\tilde{\boldsymbol{x}}_i \leftarrow (\tilde{x}_i^1, \ldots, \tilde{x}_i^p, C(e_i))$
end
```

(b) Generalization:

```
For $k = 1, \ldots, K$ do
  For $j = 1, \ldots, p$ do
    $S_j(u_k) \leftarrow \{0, 1\}$
    $w_{kj}(0) \leftarrow |G_j^0 \cap F_k| / |F_k|$
    $w_{kj}(1) \leftarrow |G_j^1 \cap F_k| / |F_k|$
    $q_j(u_k) \leftarrow \{w_{kj}(0), w_{kj}(1)\}$
    $y_k^j \leftarrow (S_j(u_k), q_j(u_k))$
  end
  $\boldsymbol{y}_k \leftarrow (y_k^1, \ldots, y_k^p)$
end
```

(3) Allocation step:
Let z be a new example and let $\boldsymbol{x} = (x^1, \ldots, x^p)$ be its corresponding description.

```
For $j = 1, \ldots, p$ do
  if $D_j \subseteq \Re$ then
    if $x^j \geqslant T_j$ then
      $X_j(z) = \tilde{x}^j \leftarrow 1$
    else
      $X_j(z) = \tilde{x}^j \leftarrow 0$
    end
  end
end
$\tilde{\boldsymbol{x}} \leftarrow (\tilde{x}^1, \ldots, \tilde{x}^p)$
For $k = 1, \ldots, K$ do
  $\mathrm{Match}(\tilde{\boldsymbol{x}}, \boldsymbol{y}_k) \leftarrow 0$
  For $j = 1, \ldots, p$ do
    $\mathrm{Match}(\tilde{\boldsymbol{x}}, \boldsymbol{y}_k) \leftarrow \mathrm{Match}(\tilde{\boldsymbol{x}}, \boldsymbol{y}_k) + w_{kj}(\tilde{x}^j)$
  end
  $\mathrm{Match}(\tilde{\boldsymbol{x}}, \boldsymbol{y}_k) \leftarrow \mathrm{Match}(\tilde{\boldsymbol{x}}, \boldsymbol{y}_k) / p$
end
$C(z) = k^* \leftarrow \arg\max_{k=1,\ldots,K} \mathrm{Match}(\tilde{\boldsymbol{x}}, \boldsymbol{y}_k)$
```

4. Case study

The use of the MS Classifier was investigated in the task of selecting between two models: the Simple Exponential Smoothing (SES) (Brown, 1963) and the Time-Delay Neural Network (TDNN) (Lang and Hinton, 1988). Both models were used for short-term forecasting of time series with no trend or seasonality.

In order to generate a set of examples with which the MS Classifier is concerned, three tasks were performed: (1) collect a large set of time series; (2) define the most adequate model for each time series, i.e. define the class attribute, and (3) define the features that describe the time series. As mentioned in the previous section, each training example stored the values of the features for a particular time series, as well as the value of the class attribute.

In the first task, 99 time series, available in the Time Series Data Library (Hyndman and Akram, 2003), were collected. This repository of data contains time series data from several domains. Most of them were originally presented in books on time series analysis and used as benchmark problems in the forecasting field. Each collected series was used to generate a different training example.

In the second task, the class labels were assigned. For such, the following procedure was applied. Given a time series, its data was divided into two parts: the fit period and the test period. The test period consists on the last 30 points of the time series and the fit period consists on the remaining data. The fit data was used to calibrate the parameters of both models SES and TDNN. Both calibrated models were used to generate one-step-ahead forecasts for the test data. Finally, the class attribute was assigned as the model which obtained the lowest mean absolute forecasting error on the test data. Table 1 shows the class distribution obtained for the 99 time series. It is also shown the error rate obtained by the *default* classifier, which always associates the class with more training examples (in this case study, the class *tdnn*) to a new example.

In the third task, the features of each time series were extracted. For such, a set of 10 descriptive statistics was deployed. The set is composed by (1) 5 numeric features: X_1 (length of the time series), X_2 (mean of the 5 first serial autocorrelations), X_3 (coefficient of variation), X_4 (skewness coefficient) and X_5 (kurtosis coefficient); and (2) 5 boolean features: X_6 (presence of significant autocorrelations), X_7, X_8, X_9 (respectively, significance of the first, second and third autocorrelations), and X_{10} (test of turning points for randomness). According to Shah (1997), these features are classical measures for describing time series and can be quickly computed even for a large number of series. Different works in the literature used at least part of these features for model selection purposes (Arinze, 1994; Chu and Widjaja, 1994; Shah, 1997; Venkatachalan and Sohl, 1999; Prudêncio and Ludermir, 2003).

All the time series features were computed using just the fit period of the time series. Since the class label was defined considering the test period of the series, what it is actually supposed is that features describing the fit data can be used to predict what to expect in the test period. A learning algorithm should be able to associate the characteristics of the series up to the present and future performance of the models.

Table 1
Number of examples associated to each class value (*ses* and *tdnn*)

# Of *tdnn* examples	52
# Of *ses* examples	47
Default error	47.47%

The third row shows the error rate obtained by the default classifier.

The quality of the forecasts generated by the SES and TDNN models may be very different depending on the time series being forecasted. Hence, the forecasting performance can be severely harmed in the case of a wrong decision concerning the model selection. These statements can be better understood by regarding two time series where the considered models obtained contrasting results.

In Fig. 2(a), the time series is presented (among the 99 used to generate the training examples) for which the SES model obtained better results when compared to the TDNN model (see Fig. 2(b)). This series presents some notable characteristics: a small number of observations (only 48 time points) and no significant autocorrelations. These characteristics may explain, in part, the poor performance of the TDNN model. First, there is a reduced amount of data for calibrating a neural network model. Furthermore, the absence of significant autocorrelations may indicate that the series would have been generated by a purely random process, and hence, a simple model, like the SES model, would be more adequate.

Contrarily, Fig. 3(a) presents the series for which the TDNN obtained better comparative results. This series is very different from the one previously discussed. It presents a larger number of observations (160 time points) as well as significant autocorrelations. The forecasts generated by the TDNN model were, in turn, much more accurate than the forecasts generated for the first series (see Fig. 3(b)).

The case study presented here has already been tackled in a previous work (Prudêncio, 2002), but using decision trees. The main drawback of the decision trees for model selection is that they contain numeric tests that generate hard surfaces of decision. For example, one of the decision trees induced in (Prudêncio, 2002) applied the test $X_2 \leqslant 0.18$ as a necessary condition for selecting a

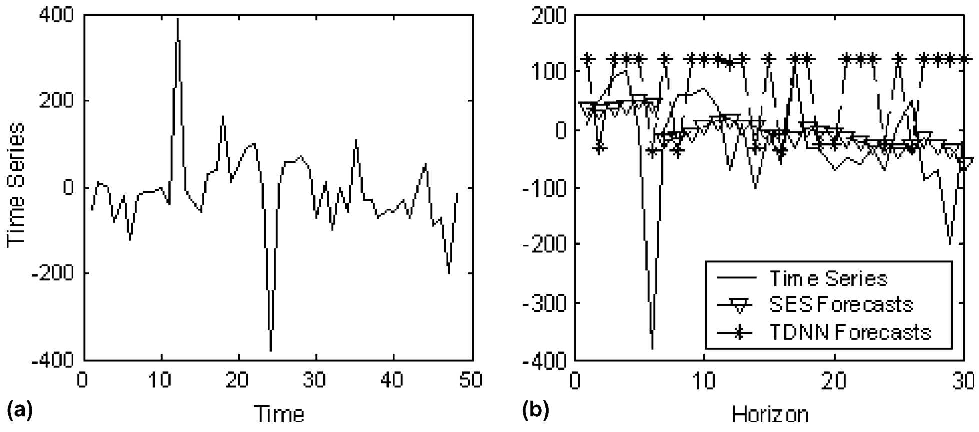

Fig. 2. Best *ses* comparative results.

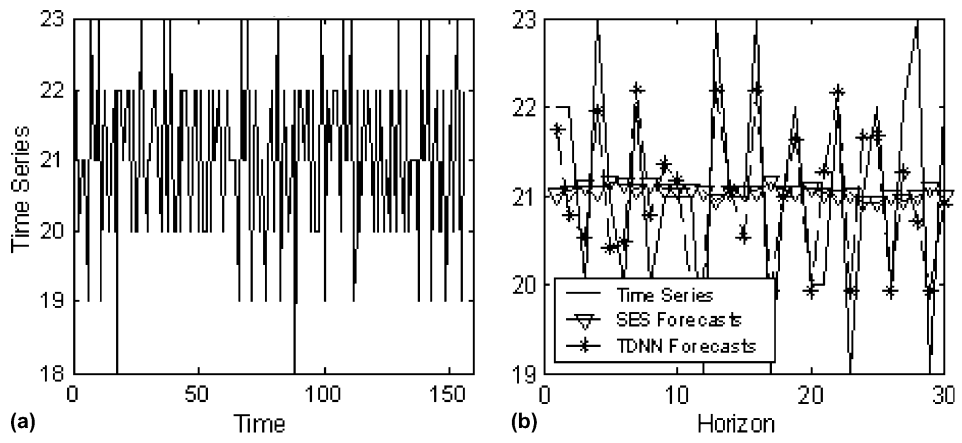

Fig. 3. Best *tdnn* comparative results.

model. However, it does not make sense in a real world application to consider a model as the best if X_2 is less than 0.18 and inapplicable when this attribute is just a little bit more. The MS Classifier in turn is a more flexible approach since it uses matching to support the model selection.

Table 2 shows an example of the matching process of a new instance z with the modal symbolic description for the classes *ses* and *tdnn*.

In the first and second columns, the modal symbolic descriptions $\boldsymbol{y}_{ses}$ and $\boldsymbol{y}_{tdnn}$ of the classes *ses* and *tdnn*, are shown, respectively. In the third column, the thresholds calculated for the first 5 numeric features are shown (the thresholds are calculated as described in the *pre-processing* step). In the fourth column, there is the description $\boldsymbol{x} = (x^1, \ldots, x^{10})$ of the new instance z to be classified. The fifth column shows the new description $\tilde{\boldsymbol{x}} = (\tilde{x}^1, \ldots, \tilde{x}^{10})$ of z with all numeric features being now binary attributes after the pre-processing step. In the next columns, the partial matching values of the new example with the modal symbolic descriptions are presented. The new instance was classified as *tdnn*, since its total matching (0.53) was greater than the matching to the class *ses* (0.35). Notice that the MS Classifier is such that a small change in the value of a single attribute has only a small effect in the matching values.

Table 2
Matching a new instance with the modal symbolic descriptions

$\mathbf{y}_{ses}$	$\mathbf{y}_{tdnn}$	T_j	$\mathbf{x}$	$\tilde{\mathbf{x}}$	Match $(\tilde{\mathbf{x}}, ses)$	Match $(\tilde{\mathbf{x}}, tdnn)$
1(0.2),0(0.8)	1(0.4),0(0.6)	$T_1 = 339$	$x^1 = 73$	$\tilde{x}^1 = 0$	0.8	0.6
1(0.4),0(0.6)	1(0.7),0(0.3)	$T_2 = 5.36$	$x^2 = 8.2$	$\tilde{x}^2 = 1$	0.4	0.7
1(0.3),0(0.3)	1(0.4),0(0.6)	$T_3 = 0.05$	$x^3 = 2.1$	$\tilde{x}^3 = 1$	0.3	0.4
1(0.8),0(0.2)	1(0.9),0(0.1)	$T_4 = 0.44$	$x^4 = 0.1$	$\tilde{x}^4 = 0$	0.2	0.1
1(0.5),0(0.5)	1(0.4),0(0.6)	$T_5 = 0.18$	$x^5 = 0.2$	$\tilde{x}^5 = 1$	0.5	0.4
1(0.2),0(0.8)	1(0.5),0(0.5)	–	$x^6 = 1$	$\tilde{x}^6 = 1$	0.2	0.5
1(0.9),0(0.1)	1(0.3),0(0.7)	–	$x^7 = 0$	$\tilde{x}^7 = 0$	0.1	0.7
1(0.9),0(0.1)	1(0.2),0(0.8)	–	$x^8 = 0$	$\tilde{x}^8 = 0$	0.1	0.8
1(0.3),0(0.7)	1(0.7),0(0.3)	–	$x^9 = 1$	$\tilde{x}^9 = 1$	0.3	0.7
1(0.4),0(0.6)	1(0.6),0(0.4)	–	$x^{10} = 0$	$\tilde{x}^{10} = 0$	0.6	0.4
Average					0.35	0.53

5. Experiments and results

In this section, the experiments and results obtained by the MS Classifier in the case study are presented.

The selection performance of the MS Classifier was measured using the leave-one-out cross-validation method (Mitchel, 1997). This method is a very common and reliable way to evaluate the performance of learning algorithms on a particular dataset. Suppose you have m training examples in a dataset (in our case $m = 99$). Each example, in turn, is left out and the learning algorithm is trained using the remaining $m - 1$ examples. The learning algorithm is then judged on the example left out, receiving either 1 or 0 for success or failure. This step is repeated once for each training example, resulting in m judgments. The number of correct judgments represents the success rate of the learning algorithm and the number of wrong judgments represents the error rate. In the experiments presented below, the leave-one-out method was used as the standard procedure for evaluation of learning algorithms.

Following the leave-one-out procedure, the MS Classifier obtained a number of 59 correctly classified examples from the set of 99 examples on the dataset, which represents an error of 40.40% (see Table 3, second column). Although the achieved error may be considered high, it is lower than the default error of 47.47% (see Table 3, fourth column).

After analyzing these initial results, we investigated how the performance of the MS Classifier could be improved in the case study. It was investigated the use of an automatic mechanism for feature selection. It is important to highlight that the MS Classifier uses a similarity measure as a basis for classifying new examples. If this measure is calculated using irrelevant attributes, classifier performance can be harmed. In fact, other similarity-based algorithms, such as the instance-based learning algorithms, are also sensitive to the presence of irrelevant attributes (Mitchel, 1997). Therefore, the next step in the work was to incorporate a mechanism of feature selection for the MS Classifier.

There are two general approaches to feature selection: filters and wrappers (Kohavi and John, 1997). In the former, the relevance of each attributes is measured in terms of the training data statistics, such as the entropy and correlation. The

Table 3
Results obtained by the learning algorithms following the leave-one-out procedure

	MS first run	MS second run	Default	J48	IBk
Correct judgments	59	65	52	62	54
Wrong judgments	40	34	47	37	45
Error rate (%)	40.40	34.34	47.47	37.37	45.45

The second and third columns, respectively, show MS Classifier performance using all attributes and using feature selection.

main drawback of this approach is that it does not take into account the algorithm being used. In the latter, a search is performed in the space of features. The learning algorithm is executed using different subsets of attributes, and the subset that obtained the best classification results in the training data is selected. The main disadvantage of this approach is the execution time, since the algorithm is executed several times before selecting a good subset of features. The wrapper approach was chosen since the MS classifier is not time consuming.

The Backward-Elimination algorithm, which is one of the most used wrappers (Kohavi and John, 1997), was applied to perform feature selection. In this algorithm, the initial search point is the whole set of features. At each step, new search points are generated by eliminating a different feature from the current search point. All these new subsets are then evaluated, and the one that obtains the best results is defined as the new current search point. This process iterates until all attributes are eliminated.

Table 3 (second and third columns) shows the leave-one-out results obtained by the MS Classifier using: (1) all attributes (MS first run) and (2) the mechanism for feature selection (MS second run). As it can be seen, the MS Classifier with feature selection obtained a classification error of 34.34%, which was significantly lower than the errors obtained in the first execution of the MS Classifier (40.40%).

The results obtained by the MS Classifier with feature selection were compared to the results obtained by other learning algorithms. For such, experiments were performed using the IBk algorithm, which is the *k*-nearest neighbors algorithm (*k*-NN) implemented in the WEKA Java package (Witten and Frank, 2000). The results of the MS Classifier were also compared to the results previously obtained in (Prudêncio, 2002) using the J.48 algorithm implemented in WEKA. This algorithm is a variant of the C4.5 algorithm proposed by Quinlan (1993) for decision tree induction.

As can be seen in Table 3 (fifth and sixth columns), the error rate obtained by the MS Classifier was lower than both the error obtained by the IBk algorithm (45.45%) and the error obtained by the J48 algorithm (37.37%). A statistical *t*-test (see Mitchel, 1997, Chapter 5) verified that the results obtained by the MS was significantly better than the IBk algorithm, considering a 95% degree of confidence. Although the performance of the MS Classifier was not statistically better than the J48 performance, the error obtained by the MS Classifier was nevertheless lower than the J48 error.

In terms of computational cost, the MS Classifier is not time consuming. In fact, the fast execution of the MS Classifier has been reported in (Bezerra et al., 2002, De Carvalho and Bezerra, 2003) when the algorithm was applied to an information filtering task. MS parameters can be computed in a single-pass through the training data and its complexity is linear in the number of training examples and attributes. The computational cost of the MS increases when feature selection is applied. However, as the generation of the modal variables is efficient, the impact of the feature selection mechanism is not drastic. In future work, experiments need to be reported in order to confirm the computational efficiency of the MS Classifier in the investigated case study.

6. Conclusion

A modal symbolic classifier was applied to a problem of time series model selection. Contributions of this work can be regarded in two different fields: (1) in the *Symbolic Data Analysis* field, since some of its concepts were applied to a problem which had not yet been tackled; and (2) in the *Time Series Forecasting* field, since we provided a new approach for acquiring knowledge that can be used to select time series models.

In the experiments, it was verified that the performance of the MS Classifier was satisfactory when compared to other traditional learning algorithms. It was also observed that the algorithm was sensitive to the features used in the classification process. In fact, the classification performance of the MS Classifier was improved when a mechanism of feature selection was applied.

As future work, some modifications in the current implementation of the algorithm will be performed, such as evaluating different matching functions. Furthermore, other time series features may be included in the experiments, along with other time series models. We also intend to use and evaluate the MS Classifier for providing a ranking of models, instead of simply suggesting a single candidate model.

Acknowledgements

This paper is supported by grants from CNPq, CAPES and FINEP, Brazil.

References

Arinze, B., 1994. Selecting appropriate forecasting models using rule induction. Omega-Internat. J. Manage. Sci. 22 (6), 647–658.

Bezerra, B.L.D., De Carvalho, F.A.T., Ramalho, G.L., Zucker, J.-D., 2002. Speeding up recommender systems with meta-prototypes. In: Lecture Notes in Artificial Intelligence. Springer, Berlin, pp. 227–236.

Bock, H.H., 2000. Symbolic data. In: Bock, H.H., Diday, E. (Eds.), Analysis of Symbolic Data. Springer, Berlin, pp. 39–53.

Bock, H.H., Diday, E., 2000. Analysis of Symbolic Data. Springer, Heidelberg.

Box, G.E., Jenkins, G.M., 1970. Time Series Analysis: Forecasting and Control. Holden-Day, San Francisco.

Brown, R.G., 1963. Smoothing, Forecasting and Prediction. Prentice-Hall, Englewood Cliffs.

Chu, C.-H., Widjaja, D., 1994. Neural network system for forecasting method selection. Decision Support Systems 12 (1), 13–24.

Collopy, F., Armstrong, J.S., 1992. Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations. Management Science 38 (10), 1394–1414.

De Carvalho, F.A.T., Bezerra, B., 2003. Information filtering based on modal symbolic objects. In: Schader, et al. (Eds.), Between Data Science and Applied Data Analysis. In: Proc. 26th Annual Conf. of Germany Classification Soc. (GfKl'2002). Springer, Berlin, pp. 394–404.

De Carvalho, F.A.T., Cananea, I.C., Verde, R., 2001. Symbolic classifier based on modal symbolic descriptions. In: Book of Short Papers of the Meeting of the Classification and Data Analysis Group of the Italian Statistical Society (CLADAG'2001). Library of the Faculty of Economics of Palermo Editions, Palermo, pp. 197–200.

Dorffner, G., 1996. Neural networks for time series processing. Neural Network World 6 (4), 447–468.

Fayyad, U.M., Irani, K.B., 1993. Multi-interval discretization of continuous attributes as preprocessing for classification learning. In: Bajcsy, R. (Ed.), Proc. 13th Internat. Joint Conf. on Artificial Intelligence. Morgan Kauffmann, Los Altos, CA, pp. 1022–1027.

Fildes, R., 1989. Evaluation of aggregate and individual forecast method selection rules. Manage. Sci. 15 (9), 1056–1065.

Hyndman, R., Akram, M., 2003. Time Series Data Library. Available from <http://www-personal.buseco.monash.edu.au/~hyndman/TSDL>.

Kohavi, R., John, G., 1997. Wrappers for feature subset selection. Artificial Intelligence 97 (1–2), 273–324.

Lang, K., Hinton, G., 1988. Time-delay neural network architecture for speech recognition. CMU Technical Report CS-88-152, Carnegie-Mellon University, Pittsburgh.

Mitchel, T., 1997. Machine Learning. McGraw-Hill, New York.

Montgomery, D.C., Johnson, L.A., Gardiner, J.S., 1990. Forecasting and Time Series Analysis. McGraw-Hill, New York.

Prudêncio, R.B.C., Ludermir, T.B., 2002. Selection of models for time series prediction via meta-learning. In: Abraham, A., Ruiz-del-Solar, J., Koppen, M. (Eds.), Soft Computing Systems Design Management and Applications. IOS Press, Amsterdam, pp. 74–83.

Prudêncio, R.B.C., Ludermir, T.B., 2003. Selecting and ranking time series models using the NOEMON approach. In: Lecture Notes in Computer Science 2714, pp. 654–661.

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo.

Schnaars, S.P., 1986. An evaluation of rules for selecting an extrapolation model on yearly sales forecasts. Interfaces 16 (6), 100–107.

Shah, C., 1997. Model selection in univariate time series forecasting using discriminant analysis. Internat. J. Forecast. 13, 489–500.

Tashman, L.J., Kruk, J.M., 1996. The use of protocols to select exponential smoothing procedures: A reconsideration of forecasting competitions. Internat. J. Forecast. 12 (2), 235–253.

Turban, E., 1992. Expert Systems and Applied Artificial Intelligence. Macmillan Publishing Company, New York.

Venkatachalan, A.R., Sohl, J.E., 1999. An intelligent model selection and forecasting system. J. Forecast. 18, 167–180.

Witten, I.H., Frank, E., 2000. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco.