Available online at www.sciencedirect.com
SCIENCE DIRECT®
NEUROCOMPUTING
ELSEVIER
Neurocomputing 61 (2004) 121–137
www.elsevier.com/locate/neucom

Meta-learning approaches to selecting time series models

Ricardo B.C. Prudêncio*, Teresa B. Ludermir

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

Available online 9 June 2004

Abstract

We present here an original work that applies meta-learning approaches to select models for time-series forecasting. In our work, we investigated two meta-learning approaches, each one used in a different case study. Initially, we used a single machine learning algorithm to select among two models to forecast stationary time series (case study I). Following, we used the NOEMON approach, a more recent work in the meta-learning area, to rank three models used to forecast time series of the M3-Competition (case study II). The experiments performed in both case studies revealed encouraging results.

Keywords: Time-series forecasting; Model selection; Meta-learning

1. Introduction

Time-series forecasting has been used in several real world problems in order to eliminate losses resultant from uncertainty, as well as to support the decision-making process [15]. Several models can be used to forecast a time series. Selecting the most adequate model for a given time series, from a set of available models, may be a difficult task depending on the candidate models and the time series characteristics.

Several approaches to treating the problem of model selection can be identified. The most straightforward is trying out each model using the time series data, and selecting the one which obtained the best results. Despite its simplicity, this solution may be very costly when there is a large number of candidate models or series to forecast.

* Corresponding author. Tel.: +55-81-21268430; fax: +55-81-21268438.
E-mail address: rbcp@cin.ufpe.br (R.B.C. Prudêncio).

0925-2312/$ - see front matter
doi:10.1016/j.neucom.2004.03.008

A more efficient way to treat the problem is via the use of expert knowledge relating the models and the series to be forecasted [7]. Despite the good results of this solution, developing rules for model selection may be unfeasible, since good human experts are not always available and may be very expensive [3].

In order to minimize the above difficulty, machine-learning algorithms may be used to automatically acquire knowledge for model selection [3], leading to a reduced need for experts and a potential improvement of performance.

Building on that, we developed an original work which treats the model selection problem via meta-learning approaches [26]. This broaden solution is able not only to select the best model to forecast a given series, but also to provide more informative results, such as a ranking of the candidate models according to their performance in forecasting the given series.

The meta-learning approaches deployed in this work were originally proposed to select the best candidate algorithm for a machine learning problem (usually classification) based on the characteristics of the problem. In our work, we considered the time series forecasting as a machine learning problem and adapted meta-learning techniques to the problem of selecting time-series models.

We investigated two different meta-learning approaches for model selection, each one tested in a different case study. Initially, we used a single machine learning algorithm to select among two models (the simple exponential smoothing [15] and the time delay neural network [11]) to forecast stationary time series (case study I). Following, the NOEMON approach [9] was used to rank three models used to forecast time series of the M3-Competition [12] (case study II). The NOEMON approach is a more recent and sophisticated work in the meta-learning area which can provide a ranking of models by combining different machine learning algorithms. The experiments performed in both case studies revealed encouraging results in terms of both the accuracy in the selection task and the forecasting performance of the selected models.

In Section 2, we present some approaches to time-series model selection. Section 3 brings a brief explanation about meta-learning and some of its techniques. In Section 4, we present our proposal to select time series models using meta-learners, followed by two case studies in Sections 5 and 6. In Section 7, we comment some related work and finally, in Section 8, we present some conclusions and the future work.

2. Time-series model selection

According to [4], a time series is a set of observations generated sequentially in time. We can define the time-series forecasting problem as the estimation of the future values of a time series using the past values of that series. Time series forecasting can be used to support the decision-making in several application domains, such as finance, industry, among others [15].

Several techniques have been developed to forecast time series, such as the family of exponential smoothing models [15], the Box–Jenkins models [4] and several artificial neural networks models [8]. Despite the diversity of the existing models, there is no

single technique that performs better than the others in all cases [7]. The adequacy of a model depends on the features of the time series to be predicted and on the characteristics of the application, such as constraints of time and resources.

The force-brute solution to the model selection task is simply to try out all the candidate models to the problem at hand, comparing the performance obtained on the available data. Despite being very straightforward, this solution is costly and not feasible for applications with a large amount of data or with several candidate models. The use of (theoretical or empirical) knowledge could drastically reduce the effort involved in the task of model selection.

An approach that uses knowledge in model selection is based on the development of expert systems, such as the rule-based forecasting system [7]. In that work, the authors implemented an expert system with 99 rules used to weight four candidate models. The rule base was developed based on 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, last observation unusual, among others) to modify the weight associated to each model. Although expert systems can express knowledge in a practical and reusable way, the process of knowledge acquisition depends on human experts, who are often scarse and expensive [3].

An approach to acquiring empirical knowledge is to perform competitions among models on different sets of series, and to analyze the obtained results. This approach was used, for example, in the M3-Competition [12]. The drawback here is that the analysis is performed by human experts, who may not be efficient in analyzing the large number of hypothesis that can be tried out to explain the results. The insights obtained from these competitions, in general, have not been defined in precise terms, as the rules developed in the rule-based forecasting system. In the M3-Competition, for example, the authors suggested that simple methods performed just as well as more sophisticated procedures. However, this statement is too imprecise, and it does not say when a user should choose a more complex model instead of a simpler one.

As it will be seen in Section 4, our proposal represents a step forward in relation to both the competition and the expert system approaches, since we analyze empirical examples of time series deploying machine learning algorithms which returns reliable knowledge that can be used to support the selection task.

3. The meta-learning approaches

There exists several machine learning algorithms that can be applied to a learning problem. As there is no single best algorithm for all possible cases [2,9], an important question is how to choose the best algorithm to a given problem. Meta-learning emerges in this scenario as a promising approach to automatically acquiring empirical knowledge that can support non-expert users in the algorithm selection task.

According to [26], there are different interpretations of the term ‘meta-learning’. In our work, we understand meta-learning as the automatic process of generating

knowledge that relates the performance of machine learning algorithms to the characteristics of the problem (i.e., characteristics of its data sets). This process is performed by a meta-learner using empirical examples, where each example consists of a machine learning problem represented by a pre-defined set of characteristics (meta-features) associated to the performance previously obtained by the candidate algorithms (base-learners). The meta-learner may simply be a single machine learning algorithm that states conditions over the meta-features to indicate when one base-learner outperforms the others. This approach was used, for example, by Aha in [2]. As it will be seen below, more sophisticated approaches can be deployed to improve selection results.

One of the first works that clearly stated the ideas of meta-learning was [13] in the context of the STATLOG project. In this work, the authors applied 20 machine learning algorithms to 22 benchmark classification problems. For each algorithm, they generated one meta-learning problem consisting of a set of 22 training examples, each one associated either to the class 'applicable' or to the class 'inapplicable'. The class 'applicable' was assigned when the classification error obtained by the algorithm fell within a pre-defined confidence interval, and 'inapplicable' was assigned otherwise. Each problem was described by a set of 16 meta-features, from simple and statistical to information theory measures of the data sets. Finally, they applied in the meta-level the C4.5 algorithm to induce a decision tree. Although the authors used a very small set of problems to induce the meta-learners, this work is a landmark in the meta-learning field.

Another important work is the NOEMON approach [9], where each pair of base learners generates a particular meta-learning problem. Given the pair (X, Y) of base-learners, a training example was classified within one of the classes 'X', 'Y' or 'equal' according to the performance obtained by the algorithms on that problem. The difference of performance was measured using a statistical test. Given a new example as input, the NOEMON system could provide a ranking of algorithms by combining the outputs given by the meta-learners. Furthermore, the system provided a mechanism which enabled the definition of a different set of meta-features for each pair of algorithms. Due to these characteristics, this approach is more flexible than the STATLOG approach.

The Zoomed-Ranking approach [5,24] uses the k-nearest-neighbor (k-NN) algorithm to provide a ranking of base-learners for a given problem. It assumes that the performance of algorithms when applied to a specific problem will be similar to the performance of those algorithms when applied to a similar problem. An important feature of this work was that it used a multi-criterion measure to evaluate the base-learners, with the accuracy and execution time of the algorithms as parameters of a global performance measure.

The Landmarking approach [16] tries to relate the performance of the base-learners to the performance obtained by simpler and faster designed algorithms, called landmarkers. This approach claims that some widely used meta-features are very time consuming, even more than the base-learners. Hence, landmarking would be an economic approach to the data characterization step and could provide useful information for the meta-learning process.

4. Meta-learning for model selection

As seen in Section 2, although expert systems can be an interesting solution to model selection, the knowledge acquisition, depends on the availability and experience of human experts, in our case forecasting experts. In our work, we propose the use of meta-learning techniques to automatically acquire knowledge for the selection of time series models. By treating time series forecasting as a machine learning problem, we can tackle the model selection task using the same meta-learning techniques which have been commonly deployed to select algorithms for classification problems. Hence, the model selection is seen as the task of choosing the best learning algorithm, and the time series models correspond to the base-learners. The meta-features are attributes used to describe the time series being forecasted.

The work presented here was originated from a previous research in which we used a machine learning algorithm to select the architecture of neural networks for time series forecasting [18]. Although that work already deployed some implicit ideas of meta-learning, such as the use of features to associate time series to neural architectures, it was not clearly associated to the meta-learning area. The work presented here can be seen as an extension of this previous work, in which we use the ideas applied in neural network design to tackle the problem of time series model selection. According to the classification for intelligent hybrid systems proposed by [10], both works can be classified as a *function replacing* system, where a manual task of the forecasting process was replaced by an intelligent system.

4.1. General architecture

Fig. 1 presents the general architecture of systems used for model selection following our proposal. As it is usual in machine learning, the system has two phases: training and use.

In the training phase, the meta-learner (ML) acquires knowledge from the set of examples stored in the database (DB). This knowledge associates time series features to the performance of the candidate models. The acquired knowledge may be refined as more examples are available in the DB.

In the phase of use, given a new time series to be forecasted, the feature extractor (FE) extracts the values of the time series features. According to these values, the ML

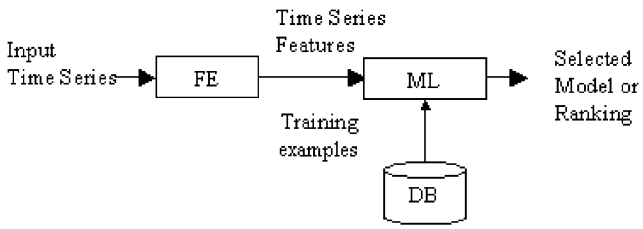

Fig. 1. System's architecture.

module suggests either one candidate model or a ranking of the available candidate models. For that, it uses the knowledge previously provided as a result of the training phase.

The FE module is responsible for extracting the features values of the input time series. This module is highly dependent on the time series category (Section 4.2), since each category of series is represented via different features (Sections 5.1 and 6.1).

The DB stores examples of times series used in the training phase. Each example associates a time series (represented by the chosen set of features) to the performance of the candidate models in the forecasting of that series (Sections 5.3 and 6.3). This set of examples is semi-automatically built: (1) the selection of series and models to be considered is a manual task (as usual); (2) the extraction of the series features is automatically performed by the FE module; and (3) the performance of the candidate models in the forecasting of a series is empirically obtained by directly applying each model to that series and evaluating the resulting forecasts.

The ML module implements the chosen meta-learning approach to acquiring knowledge (training phase) to be used in the selection or ranking of the candidate models (use phase)—see Sections 5.2 and 6.2. As seen in Section 3, the meta-learning approaches implement one or more machine learning algorithms to perform both tasks above mentioned.

4.2. Implementation issues

In order to implement a system according to the above architecture, one has to consider some important issues: (1) the category of the times series to be forecasted; (2) the candidate models to be considered; (3) the features used to describe the series; and (4) the meta-learning approach to be used.

The first issue to be addressed is the category of time series that are going to be forecasted, since it will influence all the other aspects in the system's implementation. This choice is at the user's hands, who may be interested in time series of a specific domain (such as sales or financial time series) or in more general categories. In [19], the authors present four general categories that have commonly been used by the time series analysis community: (1) stationary time series, which have no trend or seasonal behavior; (2) series presenting trend; (3) seasonal series; and (4) seasonal series with trend.

Following, it is necessary to determine what models will be considered to form the set of candidate models to forecast the time series under analysis. In [25], the authors provide some useful criteria to choose candidate models. First, the models should be well established in the literature and commonly used in practice to forecast the category of time series under analysis. Second, the models should require a minimal degree of user intervention in its design. Finally, the set of candidate models should contain representatives of different families of models. This way we have broader chances to select adequate models for the series under analysis.

The third issue to be considered is what features will be used by the FE module to describe the time series. This choice depends on the category of time series being analyzed. For example, while the feature 'period of seasonality' may be useful for

describing seasonal series, it is useless for stationary series. In the context of classification problems, we can find standard sets of meta-features that have been used in the meta-learning area. This is the case of the *data characterization tool*, developed within the METAL project.[1] Unfortunately, for time series forecasting, there is no such standard set of features, since the application of meta-learning to this problem is recent. However, we can follow some criteria to define the time series features. First, we should choose features that can be reliably identified, avoiding subjective analysis, such as visual inspection of plots. According to [1], subjective feature extraction is time consuming, requires expertise, and has a low degree of reliability. Second, we should use features that have already been used by other authors in the literature of time series analysis. Finally, we should use a manageable number of features in order to avoid a time consuming selection process.

The final issue to be addressed in our work is which meta-learning approach will be used in the ML module. This choice depends upon the user's requirements, since each meta-learning approach (Section 3) has its advantages and limitations. For example, the Aha's approach is simple and has a short execution time, since it uses a single machine learning algorithm to suggest a single best base-learner. On the other hand, the user may be interested in a more informative solution. The zoomed-ranking provides a ranking of base-learners, taking into account both the execution time and the accuracy of the base-learners. However, it is based on the application of the k-NN algorithm, which may present some drawbacks such as sensitiveness to irrelevant attributes [14]. The NOEMON approach, in turn, is more flexible regarding the choice of the machine learning algorithms used to generate the ranking, however, it may be more time consuming when compared to zoomed-ranking.

In order to test the viability of our proposal, we investigated two different meta-learning approaches for model selection, each one used in a different case study. Details of these case studies are presented in Sections 5 and 6.

5. Case study I

In this first case study, we focused on the problem of selecting models for forecasting stationary time series. The choice for this type of series was due the fact that even non-stationary time series can be reduced to stationary ones, as it can be seen in [4,8].

Following the criteria presented in Section 4.2, we used in this case study two candidate models, each one representing a different family: the simple exponential smoothing model (SES) [15], and the time-delay neural network (TDNN) [11]. Both models are well indicated for forecasting stationary time series, however they bear very distinct characteristics: while the former is a very simple and linear model, the latter is a more complex and non-linear model. As such, the quality of the forecasts obtained for a time series using these two models may be very different. A particular reason for considering a neural network model (the TDNN) is that, although this is a powerful and

[1] http://www.cs.bris.ac.uk/~cgc/METAL

promising forecasting approach, there is not much theoretical and empirical knowledge to guide its usage.

We implemented a prototype which followed the architecture presented in Fig. 1. In the following subsections, we present some relevant details about the three architecture's modules: the features extractor, the meta-learner and the database.

5.1. The feature extractor

In this case study, we used a set of 10 descriptive features:

(1) Length of the time series (LEN): number of observations of the series.
(2) Mean of the absolute values of the 5 first autocorrelations (MEAN-COR): high values of this feature suggests that the value of the series at a time point is very dependent of the values in recent past points.
(3) Test of significant autocorrelations (TAC): presence of at least one significant autocorrelation taking into account the first 5.
(4) Significance of the first, second and third autocorrelation (TAC1, TAC2 and TAC3): indicates significant dependences in more recent past points.
(5) Coefficient of variation (COEF-VAR): measures the degree of instability in the series.
(6) Absolute value of the skewness and kurtosis coefficient (KURT and SKEW): measure the degree of non-normality in the series.
(7) Test of Turning Points for randomness (TURN): A point Z_t is a turning point of a series if $Z_{t-1} < Z_t > Z_{t+1}$, or $Z_{t-1} > Z_t < Z_{t+1}$. The presence of a very large or a very low number of turning points in a series suggests that the series is not generated by a purely random process.

As this set is possibly not optimal, in future implementations we will consider new features.

5.2. The meta-learner

The meta-learner module implements a single machine learning algorithm that will select either the SES or TDNN model to forecast an input series. It receives as input the training examples stored in the DB (Section 5.3), and returns a learner associating features of the time series to the adequate model. We used here the J.48 algorithm, implemented in the WEKA Java package [27], to induce decision trees. This algorithm is a variant of the C4.5 algorithm proposed by Quinlan [20].

Although a great number of works in the meta-learning field implements the k-NN algorithm, we opted to use, in an initial stage, an algorithm that could learn symbolic knowledge, such as decision trees. This feature facilitated the interpretation of the acquired knowledge.

5.3. The database

The database stores training examples of time series forecasting problems. Each example has two parts: (1) the features describing the time series, which are those presented in Section 5.1; and (2) the class attribute, which has one of the values 'ses' or 'tdnn'. In order to assign the class attribute for a series, we applied both the SES and TDNN models on that series, and used its last 30 data points to test the models. We compared the Mean Absolute Error (MAE) obtained by each model on these 30 points, and assigned to the class attribute the model which obtained lower MAE.

5.4. Experiments and results

We describe here the experiments that evaluated the performance of our prototype. We collected 99 time series available in the *Time Series Data Library*.[2] Most of them were originally presented in books of time series analysis, and are used as benchmark problems in the forecasting field.

Some of the collected time series were originally non-stationary. In these cases, we applied difference operators to the series, as suggested by [4], in order to generate the corresponding stationary series. In the case of time series presenting trend, we applied the simple difference operator (see Eq. (1)):

$$Z^d(t) = Z(t) - Z(t-1), \tag{1}$$

where Z is the original series and Z^d is the transformed series. For seasonal series, we applied the seasonal difference operator (see Eq. (2)). After the application of these operators, the resulting training set consisted of 99 stationary time series.

$$Z^s(t) = Z(t) - Z(t-p), \tag{2}$$

where Z is the original series, p is the period of seasonality and Z^s is the transformed series.

In what follows, we evaluate the performance of the meta-learner in terms of classification performance (Section 5.4.1), and the quality of models selected by the meta-learner in terms of forecasting performance (Section 5.4.2).

5.4.1. Classification performance

We evaluated the classification performance of the meta-learner using the leave-one-out procedure [14]. At each step, 98 examples are used as the training set, and the remaining example is used to test the generated decision tree. This step is repeated 99 times, using at each time a different test example. The algorithm is then evaluated based on the obtained test errors.

In this experiment, we observed a number of 62 correctly classified examples from the set of 99 test examples, which means an error rate of 37.37% (see Table 1). Although the obtained error may be considered high in absolute terms, it is significantly lower than the default error of 47.47%, representing a gain of 10.10% in the selection

[2] http://www-personal.buseco.monash.edu.au/~hyndman/TSDL

Table 1
Classification performance of the J.48 algorithm

Test error default	47/99 (47.47%)
Test error J48	37/99 (37.37%)
95% Confidence interval	[27.84%, 46.90%]

Table 2
Comparative forecasting performance—case study I

Reference method	PB
Default SES	49.01
Default TDNN	47.36

task. The default error was obtained by the naive (or default) classifier, which always associates to a new example the class with more training examples (in our case, the class 'tdnn'). This improvement can also be statistically confirmed by analyzing the 95% confidence interval of the test errors obtained by the meta-learner. This interval has 46.90% as the upper bound, which is lower than 47.47%.

5.4.2. Forecasting performance

In order to evaluate the forecasting performance of the selected models, we considered the percentage better (PB) measure [22]. Given a selection method i, the PB measure is defined as follows:

$$\mathrm{PB}_i = 100\frac{1}{m}\sum_{j\in\mathrm{test}}\sum_{t=T+1}^{T+H}\delta_{ijt}, \tag{3}$$

where

$$\delta_{ijt} = 1 \quad \text{se } |e_{\mathrm{Rjt}}| < |e_{\mathrm{ijt}}|$$
$$0 \quad \text{otherwise.} \tag{4}$$

In the above definition, R represents a reference selection method which serves as a basis for comparison. The e_{ijt} is the one-step-ahead error obtained by the method i applied to the series j at time t, and m is the number of times where $|e_{Rjt}| \neq |e_{ijt}|$. Hence, PB_i indicates, in percentage terms, the number of times the error obtained by the reference method R was lower than the error obtained using the method i, for all test series and all H test points (in these case study $H = 30$). Values greater than 50 for PB_i indicate that the models selected by the method i are in percentage, more accurate than the models suggested by the reference method R.

In Table 2, we show the PB estimates (see Eq. (3)) of the meta-learner for the 99 series of test, using two default methods as reference. The first one is merely to use SES as the forecasting model for all series, and the second is to use TDNN. As it can be seen, for both reference methods the PB measure was lower than 50%. Although the

PB measure was not significantly low for the reference method SES, it was nevertheless lower than 50%. These results indicate that the models selected by the meta-learner were in general more accurate than SES and TDNN used as default models.

6. Case study II

The second case study investigated the selection of models to forecast the yearly time series of the M3-Competition [12]. These series are related to particular economic and demographic domains and represent a convenient sample for expository purposes [22]. In fact, the series of the M3-Competition has been commonly used as a benchmarking sample to evaluate model selection strategies, as it can be seen in [7,22,25]. In this case study, we used the following candidate models which have been commonly used to forecast the series of the M3-Competition [12]: random walk (RW), Holt's linear exponential smoothing (HL) and auto-regressive model (AR). A description of these models can be found in [15].

As seen in case study I, we used a learning technique to suggest one single time series model from the set of candidate ones. Although this is a valuable approach, a more informative and flexible solution for model selection is to provide a ranking of the candidate models to each series under analysis. First, if enough resources are available, more than one model may be used to forecast a time series. Second, if the user has some preference for a specific subset of candidate models, he/she can select the model that obtained the best rank among the models of interest.

In this case study, we adapted the NOEMON approach [9] to rank and select from the three above cited models. This approach has been recently used to select algorithms in classification problems and the results have been very promising [9]. The adapted NOEMON approach is described in more details in Section 6.2 below.

As in the first case study, we implemented here a prototype following the architecture presented in Fig. 1. We present below some important details of this implementation.

6.1. The feature extractor

In this module, we used the following features to describe the yearly series of the M3-Competition:

(1) Length of the time series (LEN): number of observations of the series.
(2) Basic trend (BT): slope of the linear regression model. Large values of this feature suggest the existence of a global trend in the series.
(3) Ratio of turning points (TP): percentage of turning points in the series (100* number of turning points divided by the length of the series). This feature attempts to measure the degree of oscillation in a series.
(4) First coefficient of autocorrelation (AC1): Large values of this feature suggest that the value of the series at a time influences the value at the following time.
(5) Type of the time series (TYPE): categorical variable that indicates the source of the data. It is represented by 6 categories: 'micro', 'macro', 'industry', 'finances', 'demographic' and 'others'.

6.2. The meta-learner

The NOEMON generates a ranking of models for each time series given as input, and suggests to the user the models that get the top position in the ranking. To generate a ranking of n models, the NOEMON uses a number of classifiers (learners), given by Eq. (5), each one associated to a specific pair of models.

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}. \tag{5}$$

For constructing the classifier associated to a pair (X, Y), the NOEMON adopts the following procedure. First, it defines a set of learning examples where each example corresponds to a time series described by a set of features and associated to a class attribute (either 'X' or 'Y'). The class attribute is assigned according to the model (X or Y) which obtained the best forecasting results for that series. At following, the NOEMON applies a learning algorithm, which will be responsible for associating new time series either to the class 'X' or to the class 'Y'.

Given a new time series as input, NOEMON collects the outputs of the $n(n-1)/2$ classifiers for the series. At following, it defines a score for each candidate model by counting how many times the model appears among the $n(n-1)/2$ collected outputs. The ranking is then generated by sorting the scores associated to the models.

As said before, in this case study, we considered 3 available models (RW, HL and AR). As such, the NOEMON constructed $3(3-1)/2 = 3$ classifiers C_1, C_2 and C_3, associated to the pairs (RW, HL), (RW, AR) and (HL, AR), respectively. Now, suppose that the outputs of the three classifiers for a new time series are 'HL', 'RW' and 'HL', respectively. In this case, the scores associated to the models RW, HL and AR are 1, 2 and 0, respectively. By sorting these values, the NOEMON generates the ranking [HL, RW, AR] and consequently suggests the model HL as the best one for the input series. In this example, the model HL is supposed to be better than RW according to the classifier C_1, and better than AR according to the classifier C_3.

In our implementation of NOEMON, we used multi-layer perceptron neural networks (MLP) [21] as the classifiers. A reason for this choice is the good performance of neural networks when compared to other learning algorithms in several problems [23]. Another advantage of the MLP model is the reduced amount of time needed to generate an output to a given input pattern. This feature is crucial to NOEMON since, in order to generate a ranking, it has to collect the outputs of $n(n-1)/2$ classifiers ($O(n^2)$). In the original implementation of NOEMON [9], the authors used the k-NN algorithm, which requires less computation during training, but more computation to return an output [14]. By using an eager algorithm, such as the MLP neural network, the prototype can efficiently generate rankings for new series provided by the user.

We used MLPs with only one hidden layer. The input layer represents the time series features (see Section 6.1). The first four features were normalized and the categorical feature 'TYPE' was represented by 5 binary attributes, each one associated to one of the categories 'micro', 'macro', 'industry', 'finances', and 'demographic'. In the case of the category 'others', all 5 input received the value 0. The output layer repre-

Table 3
Classification performance of the MLPs

	(RW, HL)	(RW, AR)	(HL, AR)
Test error default	89/215 (41.40%)	106/215 (49.30%)	86/215 (40.00%)
Test error MLP	78/215 (36.28%)	66/215 (30.70%)	72/215 (33.49%)
Obtained gain	5.12%	18.6%	6.51%

sents the class of the input-pattern, i.e., the model associated to the time series. The training process was performed using the standard Backpropagation algorithm [21], and followed the benchmarking rules proposed in [17].

6.3. The database

For each pair of models (X, Y), the NOEMON stores a set of training examples where each example has two parts: (1) the features describing a time series (see Section 6.1); and (2) the class attribute, which has one of the values 'X' or 'Y'. In order to assign the class attribute, we observed the forecasting performance of the models on a sample which was not used to estimate them. We used the first observations of the series to estimate the models and the last 6 observations to test the models. This number of test points was defined following the rules of the M3-Competition [12]. We compared the mean absolute error (MAE) obtained by each model on these 6 points, and assigned to the class attribute the model which obtained lower MAE.

6.4. Experiments and results

We describe here the experiments that evaluated the performance of our prototype in this case study. In our experiments, we used the 645 yearly time series of the M3-Competition [12]. In the following sections, we present respectively the classification performance obtained by the neural networks, the evaluation of the rankings generated by NOEMON and the forecasting performance of the models selected by NOEMON.

6.4.1. Classification performance

Here, we present the classification performance obtained by the MLPs in the previously defined classification problems. In order to evaluate the MLP performance, we used the procedure suggested in Proben [17]. The set of 645 examples was equally divided into training, validation and test sets, each one composed by 215 examples. The first two sets are used in the training process and the test set is used to evaluate the trained networks.

The MLPs were compared to the naive (or default) classifier, which always associates a new example to the most frequent class (for example the class 'HL' in the problem (RW, HL)). Table 3 shows the classification test error obtained by the MLPs, the default test error and the gain obtained by using the MLPs. As we can see, for each

Table 4
Average Spearman coefficient

Method	SRC
Default	0.15
NOEMON	0.38

pair of models, we obtained a gain in the classification error when the MLPs were used. These results showed us that the networks were able to learn relationships associating the time series features to the forecasting models. We observed that the best test result (around 18%) was obtained in the pair (RW, AR), where the classes are more equally distributed.

6.4.2. Ranking performance

The quality of a suggested ranking for a given series was evaluated by measuring the similarity to the ideal ranking, which represents the correct ordering of the models according to the MAE error. In our work, we used the Spearman's rank correlation coefficient (see [5,24]) to measure the similarity between a suggested and the ideal rankings.

Given a series i, we calculate the squared difference between the suggested and the ideal ranks for each model j (D_{ij}^2). Following, we calculate the sum of these differences for all models:

$$D_i^2 = \sum_j D_{ij}^2. \tag{6}$$

Finally, the Spearman coefficient is calculated using the equation

$$\mathrm{SRC}_i = 1 - \frac{6 * D_i^2}{n^3 - n}, \tag{7}$$

where n is the number of candidate models. The larger is the value of SRC_i, the greater is the similarity between the suggested and the ideal rankings for the series i.

In order to evaluate the rankings generated for series in the test set, we calculated the average of the Spearman's correlation for all these series (Eq. (8)).

$$\mathrm{SRC} = \frac{1}{215} \sum_{i \in \mathrm{test}} \mathrm{SRC}_i. \tag{8}$$

The NOEMON approach was compared to a default ranking method, where the same ranking is suggested for all series. This ranking was defined by observing the number of series in which each candidate model obtained the best performance. In our case, the default ranking was [HL, RW, AR]. In Table 4, we show the average Spearman coefficient for the rankings generated by NOEMON and for the default ranking. As it can be seen, the rankings generated by the NOEMON method were, in average, more correlated to the ideal ranking.

Table 5
Comparative forecasting performance—case study II

Reference method	PB
Default RW	36.0
Default HL	47.9
Default AR	37.3

6.4.3. Forecasting performance

We also evaluated the forecasting performance of the models suggested by NOEMON. As we have previously said, the NOEMON selects the models that get the top position in the ranking. When more than one model is selected for a given series, the final forecasting is the simple average of the forecasts generated by the selected models. In our experiments, we compared the NOEMON method with three default methods. The first is merely to use RW as the forecast model for all series, the second is to use HL and the third is to use the AR model.

In Table 5, we show the PB estimates (see Section 5.4.2, Eq. (3)) of the NOEMON method for the 215 series of the test set, using the three default methods as the reference method. As it can be seen, for all reference methods, the PB measure was lower than 50%. These results indicate that the models selected by the NOEMON method was, in general, more accurate than the models suggested by the default methods.

7. Related work

The use of machine learning algorithms to time series model selection was originally proposed by Arinze [3], and later used in other works. In [3], the author used decision trees to select one among six available models. As training examples, he used a set of 67 time series described by six features: level of detail (quarterly or yearly), the number of turning points, autocorrelation coefficients, trend, coefficient of determination, and error of the linear regression model. In [6], a neural network system was used to select among several exponential smoothing models using the autocorrelations. In [23], the authors used a neural network to select groups of candidate models for each time series.

In general, these works treat model selection as a classification problem where the class attribute represents the best candidate model to forecast the series, and then they use a machine learning algorithm as the classifier. In fact, all these works can be viewed as particular cases of meta-learning, which are similar to case study I. However, these works were not developed within the context of meta-learning.

Our proposal is more general, since it includes the previous related works in a general framework and addresses new techniques for model selection which were not used yet.

8. Conclusion

In this work, we proposed a new approach to providing knowledge for the selection of time series forecasting models. We can point out contributions of this work to

two different fields: (1) in the meta-learning field, we applied meta-leaning concepts to a problem which had not yet been tackled: the time series forecasting. Although the concepts of meta-learning may be applied to different machine learning tasks (for example, clustering and regression problems), the previous efforts were mainly focused on the classification task; and (2) in the time series analysis field, we provided a new solution to acquiring empirical knowledge that can be used to optimize the forecasting performance and to provide a better understanding of the time series models' behavior.

In our work, we investigated two case studies, each one used a different meta-learning approach. In case study I, we used a single machine learning to select models to forecast stationary time series, and in case study II, we used the NOEMON approach to select models for series in the M3-Competition. The experiments performed in both case studies revealed satisfactory results, taking into account the quality in the selection and the forecasting performance of the selected models.

One issue to investigate in future work is how to automatically define the set of time series features used in the meta-learning. This is particularly important in the second case study, which used neural networks in the meta-learner. Although we suggested some criteria to choose these features, we believe that an automatic process of feature selection could improve the results of the meta-learning.

Finally, we would like to highlight that several works can be developed from our proposal by implementing other meta-learners for different categories of time series, and by using other meta-learning approaches that were not used yet in the model selection problem.

References

[1] M. Adya, F. Collopy, J.S Armstrong, M. Kennedy, Automatic identification of time series features for rule-based forecasting, Int. J. Forecasting 17 (2) (2001) 143–157.
[2] D. Aha, Generalizing from case studies: a case study, Proceedings of the Ninth International Workshop on Machine Learning, Morgan Kaufmann, LosAltos, CA, 1992, pp. 1–10.
[3] B. Arinze, Selecting appropriate forecasting models using rule induction, Omega-Int. J. Manage. Sci. 22 (6) (1994) 647–658.
[4] G.E. Box, G.M. Jenkins, Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco, CA, 1970.
[5] P. Brazdil, C. Soares, J.P. da Costa, Ranking learning algorithms: using IBL and meta-learning on accuracy and time results, Mach. Learn. 50 (3) (2003) 251–277.
[6] C.-H. Chu, D. Widjaja, Neural network system for forecasting method selection, Decision Support Systems 12 (1) (1994) 13–24.
[7] F. Collopy, J.S. Armstrong, Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations, Manage. Sci. 38 (10) (1992) 1394–1414.
[8] G. Dorffner, Neural networks for time series processing, Neural Network World 6 (4) (1996) 447–468.
[9] A. Kalousis, T. Theoharis, NOEMON: design, implementation and performance results of an intelligent assistant for classifier selection, Intelligent Data Anal. 3 (5) (1999) 319–337.
[10] S. Khebbal, S. Goonatilake, Intelligent hybrid systems: issues, classifications and future directions, Intelligent Hybrid Systems, Wiley, London, 1995, pp. 1–20.
[11] K. Lang, G. Hinton, Time-delay neural network architecture for speech recognition, CMU Technical Report CS-88-152, Carnegie-Mellon University, Pittsburgh, PA, 1988.
[12] S. Makridakis, M. Hibon, The M3-competition: results, conclusions and implications, Int. J. Forecasting 16 (4) (2000) 451–476.

[13] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, New York, 1994.
[14] T. Mitchel, Machine Learning, MacGraw Hill, New York, 1997.
[15] D.C. Montgomery, L.A. Johnson, J.S. Gardiner, Forecasting and Time Series Analysis, McGraw-Hill, New York, 1990.
[16] B. Pfahringer, H. Bensusan, C. Giraud-Carrier, Meta-learning by landmarking various learning algorithms, Proceedings of the 17th International Conference on Machine Learning (ICML-2000), Stanford, CA, 2000, pp. 743–750.
[17] L. Prechelt, Proben 1: a set of neural network benchmark problems and benchmarking rules, Technical Report 21/94, Universitat Karlsruhe, Fakultat fur Informatik, 1994.
[18] R.B.C. Prudêncio, T.B. Ludermir, Design of neural networks for time series prediction using case-initialized genetic algorithms, Proceedings of the Eighth International Conference on Neural Information Processing (ICONIP' 01), Shanghai, China, 2001, pp. 990–995.
[19] R.B.C. Prudêncio, T.B. Ludermir, Selection of models for time series prediction via meta-learning, in: A. Abraham, J. Ruiz-del-Solar, M. Koppen (Eds.), Soft Computing Systems: Design, Management and Applications, IOS Press, Amsterdam, 2002, pp. 74–83.
[20] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
[21] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by backpropagation errors, Nature 323 (1986) 533–536.
[22] C. Shah, Model selection in univariate time series forecasting using discriminant analysis, Int. J. Forecasting 13 (1997) 489–500.
[23] J.W. Shavlik, R.J. Mooney, G.G. Towell, Symbolic and neural learning algorithms: an experimental comparison, Mach. Learn. 6 (2) (1991) 111–143.
[24] C. Soares, P. Brazdil, Zoomed ranking: selection of classification algorithms based on relevant performance information, Proceedings of the PKDD2000, Springer, Berlin, 2000, pp. 126–135.
[25] A.R. Venkatachalan, J.E. Sohl, An intelligent model selection and forecasting system, J. Forecasting 18 (1999) 167–180.
[26] R. Vilalta, Y. Drissi, A perspective view and survey of meta-learning, J. Artif. Intell. Rev. 18 (2) (2002) 77–95.
[27] I.H. Witten, E. Frank, WEKA: machine learning algorithms in Java, University of Waikato, Hamilton, New Zealand [www.cs.waikato.ac.nz], 2003.