Selecting and Ranking Time Series Models Using the NOEMON Approach

Ricardo B. C. Prudêncio, Teresa B. Ludermir

Center of Informatics – Federal University of Pernambuco
P. O. Box 7851. Cidade Universitária, Recife – PE, Brazil. 50.732-970
{rbcp,tbl}@cin.ufpe.br

Abstract. In this work, we proposed to use the NOEMON approach to rank and select time series models. Given a time series, the NOEMON approach provides a ranking of the candidate models to forecast that series, by combining the outputs of different learners. The best ranked models are then returned as the selected ones. In order to evaluate the proposed solution, we implemented a prototype that used MLP neural networks as the learners. Our experiments using this prototype revealed encouraging results.

1 Introduction

Time series forecasting has been widely used to support planning and decision-making processes [1]. Several models have been developed to forecast time series, such as the Exponential Smoothing and the Box-Jenkins models, among others [1]. Given a set of candidate models to choose, a forecaster may either select *one* best model for all series, or he/she may select the best model for each time series. The former is called an *aggregate* selection rule and the latter an *individual* selection rule [2].

Although aggregate rules are simple to implement, empirical research has shown that there is no single model that performs better than the others in all series [3]. This fact motivated several authors to develop individual rules, commonly associating characteristics of the series to the best model. An approach that formalizes this knowledge in a reusable way is to use expert systems [3]. However, the process of acquiring knowledge from experts may be very expensive and time-consuming [4]. In this scenario, an interesting alternative is the use of Machine Learning (ML) techniques [5].

The ML algorithms were already used to model selection in different works [6][7][8][9]. In general, these works used a learner as a classifier that suggests just one model among the set of candidate ones. We believe that to provide a ranking of models is a more informative solution for model selection. In our work, we proposed to use the NOEMON approach [10] to select and rank time series models. This approach generates a ranking by combining the outputs of different learners. In our work, we implemented a prototype using MLP (Multi-Layer Perceptron) neural networks [11] as the learners, and performed experiments using a large set of time series. Our results revealed that the models suggested by our prototype were in average more accurate than the models suggested by different aggregate selection rules.

In section 2, we present the use of ML algorithms to model selection. Section 3 brings a brief explanation about the NOEMON approach and its application to model selection. In section 4, we present the description of the implemented prototype. Section 5 presents the experiments and results obtained by this prototype. Finally, we have the conclusion and future work in section 6.

2 Machine Learning to Model Selection

As we have previously said, a way to select time series models is using expert systems. The knowledge used in these systems is extracted from forecasting experts and it is expressed in the form of rules. In this context, we highlight the Rule-Based Forecasting system [3], which implemented an expert system with 99 rules, associating time series features (such as level discontinuities, basic trend…) to the available models. Although these systems can express knowledge in a practical and reusable way, the process of knowledge acquisition depends on human experts, which are often scarce and expensive [4]. In this scenario, ML techniques are an interesting alternative to acquire knowledge [5]. These techniques can be used to automatically learn from data, leading to potential improvement of performance, easy adaptability to new types of data, and a reduced need for experts [4].

The use of ML algorithms to model selection was originally proposed by [6], and adopted in other different works. In [6], the author used decision trees to select 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 [7], a neural network system was used to select among several exponential smoothing models, using the autocorrelations. In [8], the authors used a neural network to select a group of models and another neural network to select a single model of the group. In [9], the authors used a decision tree algorithm to select between the simple exponential smoothing model and a neural network model.

In general, these works treat the model selection as a classification problem where the class attribute represents the best candidate model to forecast the series, and then, they use a ML algorithm as the classifier. In this context, each training example consists of a time series described by a set of *time series features*, associated to the class attribute. The value of this attribute is commonly defined by experimenting all candidate models, and by choosing the one that obtained the best forecasting results for the series. A set of such examples is given as input of the ML algorithm, responsible for discovering knowledge associating the features to the candidate models.

3 NOEMON Approach to Model Selection

Previous works used ML techniques to suggest either one single model or a small group of models among the set of candidate ones. We believe that a more informative and flexible solution for model selection is to provide a ranking of the candidate models. First, if enough resources are available, more than one model may be used to forecasting a time series. Second, if the user has some preference for a specific subset of candidate models, he/she can select the model that get the best rank among the models of interest. In our work, we proposed the use of the NOEMON approach [10] to select and rank time series models. This approach has been recently used to select algorithms for classification problems and the results has been very promising [10]. In our work, we adapted the NOEMON approach to the model selection problem.

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 (n 2) classifiers (learners), each one associated to a specific pair of models. 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 ML 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 2) classifiers for the series. At following, NOEMON defines a score for each candidate model by counting how many times the model appears among the (n 2) collected outputs. The ranking is then generated by sorting the scores associated to the models. As an example, suppose that we have 3 available models (X, Y and Z). The NOEMON construct (3 2) = 3 classifiers C_1, C_2 and C_3, associated to the pairs (X, Y), (X, Z) and (Y, Z), respectively. Now, suppose that the outputs of the three classifiers for a new time series be 'Y', 'X' and 'Y', respectively. In this case, the scores associated to the models X, Y and Z are 1, 2 and 0, respectively. By sorting these values, the NOEMON generates the ranking [Y, X, Z] and consequently suggests the model Y as the best one for the input series. In fact, the model Y is supposed to be better than X according to the classifier C_1, and better than Z according to the classifier C_3.

4 The Implemented Prototype

In order to verify the viability of our proposal, we implemented and tested a prototype. In our prototype, the NOEMON approach was used to rank and select the following models: *Random Walk (RW)*, *Holt's Linear Exponential Smoothing (HL)* and *Auto-Regressive model (AR)*. We choose these models based on criteria suggested in [8]. First, the models should be well established in the literature and commonly used in practice. The models should also require a minimal degree of user intervention and they should represent different forecasting procedures.

For these models, the NOEMON creates 3 different classification problems, each one associated to the pairs of models: (RW, HL), (RW, AR) and (HL, AR). For each problem, the NOEMON uses a ML algorithm as classifier. At following, we described the most important points in the construction of these classifiers.

4.1 Time Series Features

We followed some criteria to define the time series features. First, we tried to choose features that can be reliably identified, avoiding any subjective analysis, such as visual inspection of plots. According to [12], judgmental identification is time consuming, requires expertise, and has a low degree of reliability. Second, we tried to use features that had already been used by other authors. Finally, we tried to use a manageable number of features. Based on these criteria, we defined the following features:

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). A point X_t is a turning point of a series if $X_{t-1} < X_t > X_{t+1}$, or $X_{t-1} > X_t < X_{t+1}$. 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 point influences the value at the next point.
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'.

4.2 Definition of the Training Examples

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 4.1); 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 *T* observations of the series to estimate the models and the last *H* observations to test the models. We compared the Mean Absolute Error (MAE) obtained by each model on these *h* points, and assigned to the class attribute the model which obtained lower MAE.

4.3 Definition of the ML Technique

In our implementation of NOEMON, we decided to use MLP (Multi-Layer Perceptron) neural networks [11] as the classifiers. A reason for this choice is the good performance of neural networks when compared to other ML algorithms in several problems [13]. 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 it has to collect the outputs of (n 2) classifiers ($O(n^2)$) in order to generate a ranking. In the original implementation of NOEMON [10], the authors used the KNN algorithm, which requires less computation during training, but more computation to return an output [5]. Using an eager algorithm, such as the MLP neural network, the prototype can efficiently answer the new queries of the user.

In our prototype, we used a MLP with one hidden layer. The input layer represents the time series features (see Section 4.1). 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 represents the class of the input pattern, i.e. the model associated to the time series.

5 Experiments and Results

We describe here the experiments that evaluated the performance of our prototype. In our experiments, we used the 645 yearly time series of the M3-Competition [14]. Although, these series only represent certain economic and demographic domains, they represent a convenient sample for expository purposes [2]. The series of the M3-Competition has been commonly used as a benchmarking sample to evaluate model selection strategies [2][3][8].

For each series, we estimated the three candidate models using the first observations and we used the last H = 6 observations to evaluate the performance of the models. This number was defined following the definitions of the M3-Competition [14]. In table 1, we present the class distributions for each classification problem.

Table 1. Class distributions.

	(RW, HL)	(RW, AR)	(HL, AR)
RW	281	344	--
HL	364	--	379
AR	--	301	266

The set of 645 examples was equally divided into training, validation and test sets, each one composed of 215 examples. The number of hidden nodes was defined by a trial-and-error procedure. We trained networks using 2, 4 and 6 hidden nodes (five times for each configuration), and then saved the trained network which obtained the lowest Sum of Squared Errors (SSE) in the validation set. The training process was performed by the standard Backpropagation algorithm [11], using 0.002 as the value of the learning rate.

5.1 Classification Performance

Here, we present the classification performance obtained by the MLPs in the previously defined classification problems. The MLPs were compared to the *default classifier*, which always associates a new example to the most frequent class (for example the class 'HL' in the problem (RW, HL)). Table 2 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 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 forecast models. We observed that the best test result (around 18%) was obtained in the problem (RW, AR), where the classes are more equally distributed.

Table 2. 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 NN	78/215 (36.28%)	66/215 (30.70%)	72/215 (33.49%)
Obtained Gain	5.12%	18.6%	6.51%

5.2 Quality of the Suggested Rankings

The quality of a suggested ranking for a 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 [15] 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^2_{ij}). Then we calculate the sum of these squared differences for all models. Finally, the Spearman coefficient is defined by the equation:

$$SRC_i = 1 - (6. \sum_j D^2_{ij})/(n^3 - n) \qquad (\mathbf{1})$$

where *n* is the number of 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.

$$SRC = (1/215) \sum_{i \in \text{ test set}} SRC_i \qquad (\mathbf{2})$$

The NOEMON approach was compared to an aggregate 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 (see table 1). In our case, the aggregate ranking was [HL, RW, AR]. In table 3, we show the average Spearman coefficient for the rankings generated by NOEMON and for the aggregate ranking. As we can see, the rankings generated by the NOEMON method were in average more correlated to the ideal ranking.

Table 3. Average Spearman coefficient of the NOEMON and aggregate methods

Method	SRC
Aggregate	0.15
NOEMON	0.38

5.3 Forecasting Performance

We evaluated here the NOEMON approach as an individual selection rule. 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 aggregate selection rules. The first one 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 order to compare the quality of the selection rules for all series in the test set, we considered the *Percentage Better (PB)* measure [2]. This measure associated to a selection method *i* is defined as follows:

$$PB_i = 100 * (1/m) \sum_{j\in \text{ test set}} \sum_{t = T+1}^{T+6} \delta_{ijt} \qquad (3)$$

where,

$$\delta_{ijt} = \begin{cases} 1 & \text{se } | e_{Rjt} | < | e_{ijt} | \\ 0 & \text{otherwise.} \end{cases}$$

In the above definition, *R* represents a reference rule which serves as a basis for comparison. The e_{ijt} is the one-step-ahead error obtained by the method *i* in the series *j* at time *t*, and *m* is the number of times in which $| e_{Rjt} | \neq | e_{ijt} |$. 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 rule i, for all series and all points of test. Values greater than 50 for PB_i, indicates that the models selected by the rule *i* are in average, more accurate than the models suggested by the reference rule R.

In table 4, we show the PB estimates of the NOEMON method for the 215 series of test, using the three aggregate rules as the reference methods. As we can see, for all aggregate rules the PB measure was lower than 50%. Although the PB measure was not significantly low for the rule HL, it was nevertheless lower than 50%. These results indicate that the individual rule provided by the NOEMON method was in general more accurate than the aggregate rules.

Table 4. Comparative forecasting performance measured by PB

Aggregate Method	PB
RW	36.0
HL	47.9
AR	37.3

6 Conclusion

In this work, we proposed the use of the NOEMON approach to rank and select time series models. In order to evaluate the proposed solution, we implemented a prototype to select between three widespread models. In our prototype, we used MLP neural networks as the classifiers of NOEMON. In our experiments, the trained MLPs obtained a good classification performance for all the classification problems created by the prototype. We also observed that the rankings generated by NOEMON were well correlated to the ideal rankings, and the forecasting accuracy of the selected models was improved when the NOEMON approach was used. As future work, we intend to improve the performance of this prototype by augmenting the set of time series features and by performing feature selection for each pair of models.

References

1. Montgomery, D. C., Johnson, L. A. and Gardiner, J. S.: Forecasting & Time Series Analysis. Mc-Graw-Hill, New York (1990)
2. Shah, C.: Model Selection in Univariate Time Series Forecasting Using Discriminant Analysis. International Journal of Forecasting, 13 (1997) 489-500
3. Collopy, F. and Armstrong, J.S.: Rule-based Forecasting: Development and Validation of an Expert Systems Approach to Combining Time Series Extrapolations. Management Science, 38(10) (1992) 1394-1414
4. Arinze, B., Kim, S-L. and Anandarajan M.: Combining and Selecting Forecasting Models Using Rule Based Induction. Computers & Operations Research, 24(5) (1997) 423-433
5. Mitchel, T.: Machine Learning, MacGraw Hill, New York (1997)
6. Arinze, B.: Selecting Appropriate Forecasting Models Using Rule Induction. Omega-International Journal of Management Science, 22(6) (1994) 647-658
7. Chu C-H, Widjaja D: Neural Network System for Forecasting Method Selection. Decision Support Systems, 12(1) (1994) 13-24
8. Venkatachalan, A. R. and Sohl, J. E.: An Intelligent Model Selection and Forecasting System. Journal of Forecasting, 18 (1999) 167-180
9. Prudêncio, R. B. C. and Ludermir, T. B.: Selection of Models for Time Series Prediction via Meta-Learning. Proceedings of the 2th International Conference on Hybrid Intelligent Systems (HIS' 02), Santiago, Chile, IOS Press (2002) 74-83
10. Kalousis A. and Theoharis, T.: NOEMON: Design, Implementation and Performance Results of an Intelligent Assistant for Classifier Selection. Intelligent Data Analysis, 3(5) (1999) 319-337
11. Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning Representations by Backpropagation Errors. Nature, 323 (1986) 533-536
12. Adya M., Collopy. F., Armstrong, J.S. and Kennedy, M.: Automatic Identification of Time Series Features for Rule-Based Forecasting. Int. Jour. of Forecasting, 17(2) (2001) 143-157
13. Shavlik, J. W., Mooney, R.J. and Towell, G.G.: Symbolic and Neural Learning Algorithms: An Experimental Comparison. Machine Learning, 6(2) (1991) 111-143
14. Makridakis S., Hibon M.: The M3-Competition: Results, Conclusions and Implications. International Journal of Forecasting, 16(4) (2000) 451-476
15. Soares, C. and Brazdil, P.: Zoomed Ranking: Selection of Classification Algorithms Based on Relevant Performance Information. Principles of Data Mining and Knowledge Discovery: 4th European Conference (PKDD-2000), Springer-Verlag, (2000) 126-135