European Symposium on
Time Series Prediction

ESTSP 2008

17-18-19 September 2008
Porvoo, Finland

Proceedings

European Symposium on
Time Series Prediction

ESTSP'08

17-18-19 September 2008
Porvoo, Finland

Proceedings

Acknowledgements

The Second European Symposium on Time Series Prediction is organized in collaboration with TKK (Helsinki University of Technology, Espoo), with financial support from:

- PASCAL network of excellence
- International Neural Networks Society
- European Neural Networks Society

Technical co-sponsors:

- International Neural Networks Society
- IEEE Computational Intelligence Society
- IEEE Region 8
- Pattern Recognition Society of Finland

The steering committee of ESTSP'08 would like to thank them for their appreciated contribution to the success of the conference.

Published by:
Multiprint Oy / Otamedia
Miestentie 3 a, FI-02150 Espoo, Finland
Phone: (09) 4393 200
Fax: (09) 4393 2020

Editor:
Amaury Lendasse

ISBN 978-951-22-9544-9

Foreword

Time series forecasting is a challenge in many fields. In finance, experts forecast stock exchange courses or stock market indices; data processing specialists forecast the flow of information on their networks; producers of electricity forecast the load of the following day.

The common point to their problems is the following: how can one analyze and use the past to predict the future?

The Second European Symposium on Time Series Prediction (ESTSP'08) is an event in the fields of neural networks, statistics and econometrics. It is held on 17-19 September 2008 in Porvoo, Finland. ESTSP'08 is a unique opportunity for researcher from statistics, neural networks, machine learning, control and econometrics to share their knowledge in the field of Time Series Prediction.

Forty-six papers have been submitted to ESTSP'08 and reviewed. The best twenty-eight papers have been accepted. All the presentations will be oral. The selection was difficult due to the high scientific quality of the submitted papers.

We would like to thank all members of the scientific committee of ESTSP'08 for their appreciated work in the reviewing process; they were helped by many colleagues, most of them remaining anonymous, and we associate them in our grateful thanks.

For the steering and local committee,
Amaury Lendasse

Steering and local committee

Francesco Corona	Helsinki Univ. of Technology (FI)
Marie Cottrell	Univ. Paris I (F)
Amaury Lendasse	Helsinki Univ. of Technology (FI)
Elia Liitiainen	Helsinki Univ. of Technology (FI)
Yoan Miche	Helsinki Univ. of Technology (FI)
Antti Sorjamaa	Helsinki Univ. of Technology (FI)
Michel Verleysen	UCL Louvain-la-Neuve (B)

Scientific committee

Eric de Bodt	Univ. Lille II (F) & UCL Louvain-la-Neuve (B)
Manfred Deistler	Vienna University of Technology (AUT)
Mark J. Embrechts	Rensselaer Polytechnic Institute (USA)
Manuel Grana	UPV San Sebastian (E)
Tom Heskes	Univ. Nijmegen (NL)
Jaakko Hollmen	Helsinki Univ. of Technology (FI)
Christian Jutten	INPG Grenoble (F)
Juha Karhunen	Helsinki Univ. of Technology (FI)
Samuel Kaski	Helsinki Univ. of Technology (FI)
Erkki Oja	Helsinki Univ. of Technology (FI)
Alberto Prieto	Universitad de Granada (E)
Jose Principe	University of Florida (USA)
Joseph Rynkiewicz	Univ. Paris I (F)
Francisco Sandoval	Univ.Malaga (E)
Eric Severin	Universite de Lille1 (F)
Olli Simula	Helsinki Univ. of Technology (FI)
Jochen Steil	Univ. Bielefeld (D)
Johan Suykens	KUL Leuven (B)
Marc Van Hulle	KUL Leuven (B)
Thomas Villmann	Univ. Leipzig (D)
Vincent Wertz	UCL Louvain-la-Neuve (B)
Donald Wunsch	University of Missouri (USA)

ESTSP'08 Program

Wednesday, September 17, 2008

08:40 *Conference Opening*
Olli Simula, Amaury Lendasse and Timo Honkela

Invited Talk

09:00 **Data Analysis using Self-Organizing Maps**
Marie Cottrell (Univ. Paris 1)

10:00 *Coffee Break*

10:20 *ESTSP Opening (Amaury Lendasse)*

Session 1 Chairman: Antti Sorjamaa

10:25 **On the Benefit of using Time Series Features for Choosing a Forecasting Method**
C. Lemke and B. Gabrys

10:50 **Kernel Based Imputation of Coded Data Sets**
T. Pitkäranta

11:15 **Reliability of ARMA and GARCH Models of Electricity Spot Market Prices**
P. Ptak, M. Jablonska, D. Habimana and T. Kauranne

11:40 *Lunch Break*

Session Tools Chairman: Elia Liitiäinen

12:40 **A New Interface for MPI in MATLAB and its Application Over a Genetic Algorithm**
A. Guillén, I. Rojas, G. Rubio, H. Pomares, L. J. Herrera and J. Gonzáles

13:05 **Projection of time series with periodicity on a sphere**
V. Onclinx, V. Wertz and M. Verleysen

13:30 **A Variable Selection Approach Based on the Delta Test for Extreme Learning Machine Models**
F. Mateo and A. Lendasse

13:55 **Instance or Prototype Selection for Function Approximation Using Mutual Information**
A. Guillén, L. J. Herrera, G. Rubio, A. Lendasse, H. Pomares and I. Rojas

14:20 *Coffee Break*

Invited Talk

14:40 **From Raw Data to Abstract Concepts**
Harri Valpola (Helsinki Univ. of Technology)

Session 2 Chairman: Jaakko Hollmén

15:30 **Automatic Detection of Onset and Cessation of Tree Stem Radius Increase using Dendrometer Data and CUSUM Charts**
M. Sulkava, H. Mäkinen, P. Nöjd and J. Hollmén

15:55 **Multistep-ahead Prediction of Rainfall Precipitation using the NARX Network**
J. Menezes-Júnior and G. Barreto

16:15 *AKRR and ESTSP Participant Map Presentation*

Session 3 Chairman: Yoan Miche

16:45 **Multivariate Scenario Generation using Latent Spaces Application to Credit Default Swap Spreads Prediction**
F. Vrins, E. Oja and A. Lendasse

17:10 **Hybrid Criteria for Nearest Neighbor Selection with Avoidance of Biasing for Long Term Time Series Prediction**
R. Abbas and M. Arif

18:00 *Sauna and Spa*

21:00 *Dinner (expected ending at 23:00)*

Thursday, September 18, 2008

Invited Talk

09:00 **Information Theoretic Learning and Kernel Methods**
José Príncipe (Univ. Florida)

10:00 *Coffee Break*

Session Competition 1 Chairman: Guilherme Barreto

10:20 **Playout Delay Prediction in VoIP Applications: Linear Versus Nonlinear Time Series Models**
J. Aragão and G. Barreto

10:45 **Automatic Modelling of Neural Networks for Time Series Prediction in Search of a Uniform Methodology Across Varying Time Frequencies**
N. Kourentzes and S. F. Crone

11:10 **Long Term Time Series Prediction with Multi-Input Multi-Output Local Learning**
G. Bontempi

11:40 *Lunch Break*

Session Competition 2 Chairman: Sven Crone

12:40 **Revisiting Linear and Non-linear Methodologies for Time Series Prediction - Application to ESTSP'08 Competition Data**
M. Olteanu

13:05 **Using Reservoir Computing in a Decomposition Approach for Time Series Prediction**
F. Wyffels, B. Schrauwen and D. Stroobandt

13:30 **Time Series Prediction using LS-SVMs**
M. Espinoza and T. Falck

13:55 **Exogenous Data and Ensembles of MLPs for Solving the ESTSP Forecast Competition Tasks**
P. Adeodato, A. Arnaud, G. Vasconcelos, R. Cunha and D. Monteiro

14:20 *Coffee Break*

Session Competition 3 Chairman: Amaury Lendasse

14:40	**Use of Specific-to-problem Kernel Functions for Time Series Modeling** G. Rubio, A. Guillén, L. J. Herrera, H. Pomares and I. Rojas
15:05	**Tabu Search with Delta Test for Time Series Prediction using OP-KNN** D. Sovilj, A. Sorjamaa and Y. Miche
15:30	**Long-term Prediction of Nonlinear Time Series with Recurrent Least Squares Support Vector Machines** I. Jaganjac
15:55	**Regressive Fuzzy Inference Models with Clustering Identification: Application to the ESTSP'08 Competition** F. Montesino Pouzols and A. Barriga Barros
16:20	**Results of the ESTSP'08 Competition** A. Lendasse
17:00	*Guided Tour in the town of Porvoo*
21:00	*Conference Dinner (expected ending at 23:00)*

Friday, September 19, 2008

Invited Talk

09:00 **Evidence based forecasting - Neural Networks for Time Series Prediction in Industry and Finance**
Sven F. Crone (Lancaster University Management School)

10:00 *Coffee Break*

Session 4 Chairman: José Príncipe

10:20 **Multiple Local ARX Modeling for System Identification using the Self-organizing Map**
L. Souza and G. Barreto

10:45 **Time Series Opportunities in the Petroleum Industry**
R. Nybø

11:10 **Homicide Flash-up Prediction Algorithm Studying**
D. Serebryakov and I. Kuznetsov

11:40 *Lunch Break*

Session Finance Chairman: Eric Séverin

12:40 **Neural Networks and their Application in the Fields of Corporate Finance**
E. Séverin

13:05 **Evolution of Interest Rate Curve: Empirical Analysis of Patterns using Nonlinear Clustering Tools**
M. Kanevski, M. Maignan, V. Timonin and A. Pozdnoukhov

13:30 **Bankruptcy Prediction and Neural Networks: the Contribution of Variable Selection Methods**
P. Du Jardin

13:55 **A Methodology for Time Series Prediction in Finance**
Q. Yu, A Sorjamaa and Y. Miche

14:10 *Coffee Break*

CONTENTS

Acknowledgements ii
Foreword iii
Committees iv
Program v

Session 1 1

On the Benefit of using Time Series Features for Choosing a Forecasting Method 1
C. Lemke and B. Gabrys

Kernel Based Imputation of Coded Data Sets 11
T. Pitkäranta

Reliability of ARMA and GARCH Models of Electricity Spot Market Prices 21
P. Ptak, M. Jablonska, D. Habimana and T. Kauranne

Session Tools 37

A New Interface for MPI in MATLAB and its Application Over a Genetic Algorithm 37
A. Guillén, I. Rojas, G. Rubio, H. Pomares, L. J. Herrera and J. Gonzáles

Projection of time series with periodicity on a sphere ... 47
V. Onclinx, V. Wertz and M. Verleysen

A Variable Selection Approach Based on the Delta Test for Extreme Learning Machine Models 57
F. Mateo and A. Lendasse

Instance or Prototype Selection for Function Approximation Using Mutual Information 67
A. Guillén, L. J. Herrera, G. Rubio, A. Lendasse, H. Pomares and I. Rojas

Session 2 77

Automatic Detection of Onset and Cessation of Tree Stem Radius Increase using Dendrometer Data and CUSUM Charts 77
M. Sulkava, H. Mäkinen, P. Nöjd and J. Hollmén

Multistep-ahead Prediction of Rainfall Precipitation using the NARX Network 87
J. Menezes-Júnior and G. Barreto

Session 3 97

Hybrid Criteria for Nearest Neighbor Selection with Avoidance of Biasing for Long Term Time Series Prediction ... 97
R. Abbas and M. Arif

Session Competition 1 107

Playout Delay Prediction in VoIP Applications: Linear Versus Nonlinear Time Series Models 107

J. Aragão and G. Barreto

Automatic Modelling of Neural Networks for Time Series Prediction in Search of a Uniform Methodology Across Varying Time Frequencies 117
N. Kourentzes and S. F. Crone

Long Term Time Series Prediction with Multi-Input Multi-Output Local Learning 129
G. Bontempi

Session Competition 2 139

Revisiting Linear and Non-linear Methodologies for Time Series Prediction - Application to ESTSP'08 Competition Data 139
M. Olteanu

Using Reservoir Computing in a Decomposition Approach for Time Series Prediction 149
F. Wyffels, B. Schrauwen and D. Stroobandt

Time Series Prediction using LS-SVMs 159
M. Espinoza and T. Falck

Exogenous Data and Ensembles of MLPs for Solving the ESTSP Forecast Competition Tasks 169
P. Adeodato, A. Arnaud, G. Vasconcelos, R. Cunha and D. Monteiro

Session Competition 3 177

Use of Specific-to-problem Kernel Functions for Time Series Modeling 177
G. Rubio, A. Guilln, L. J. Herrera, H. Pomares and I. Rojas

Tabu Search with Delta Test for Time Series Prediction using OP-KNN 187
D. Sovilj, A. Sorjamaa and Y. Miche

Long-term Prediction of Nonlinear Time Series with Recurrent Least Squares Support Vector Machines 197
I. Jaganjac

Regressive Fuzzy Inference Models with Clustering Identification: Application to the ESTSP'08 Competition 205
F. Montesino Pouzols and A. Barriga Barros

Session 4 215

Multiple Local ARX Modeling for System Identification using the Self-organizing Map 215
L. Souza and G. Barreto

Time Series Opportunities in the Petroleum Industry .. 225
R. Nybø

Homicide Flash-up Prediction Algorithm Studying 235
D. Serebryakov and I. Kuznetsov

Session Finance.....243

Neural Networks and their Application in the Fields of Corporate Finance 43
E. Séverin

Evolution of Interest Rate Curve: Empirical Analysis of Patterns using Nonlinear Clustering Tools 261
M. Kanevski, M. Maignan, V. Timonin and A. Pozdnoukhov

Bankruptcy Prediction and Neural Networks: the Contribution of Variable Selection Methods 271
P. Du Jardin

A Methodology for Time Series Prediction in Finance ..285
Q. Yu, A Sorjamaa and Y. Miche

On the benefit of using time series features for choosing a forecasting method

Christiane Lemke and Bogdan Gabrys

Bournemouth University - School of Design, Engineering and Computing
Poole House, Talbot Campus, Poole, BH12 5BB - United Kingdom

Abstract. In research of time series forecasting, a lot of uncertainty is still related to the question of which forecasting method to use in which situation. One thing is obvious: There is no single method that performs best on all time series. This work examines whether features extracted from time series can be exploited for a better understanding of different behaviour of forecasting algorithms. An extensive pool of automatically computable features is identified, which is submitted to feature selection algorithms. Finally, a possible relationship between these features and the performance of forecasting and forecast combination methods for the particular series is investigated.

1 Introduction

Extensive empirical studies of the performance of forecasting and forecast combination algorithms, for example conducted by Makridakis and Hibon [1] and Stock and Watson [2], revealed that there is no clear cut winner among the pool of methods investigated which works well for all time series. In a response to the results of the M3 competition [1], Robert J. Hyndman [3] put the future challenges for time series forecasting research into the following words: *"Now it is time to identify why some methods work well and others do not"*.

It is generally acknowledged that different types of time series require different treatment. This brings up the question if characteristics of time series can be used to draw conclusions about which method will work best for forecasting their future values. This work investigates an automatic approach to this problem, since the thorough analysis by experts is often not feasible in practical applications that process a large number of time series in very limited time.

A classic and straightforward classification for time series has been given by Pegels [4]. Time series can thus have patterns that show different seasonal effects and trends, both of which can be additive, multiplicative or non-existent. Gardner [5] extended this classification by including damped trends. Time series do however have many more features that can be taken into account for a potential selection of a method that works best.

Time series analysis in order to find an appropriate ARIMA model has been discussed since the seminal paper of Box and Jenkins [6]. Guidelines are summarised in [7] and rely heavily on examining autocorrelation and partial autocorrelation values of a series. Some publications focus on automatically detecting time series characteristics for model selection: Adya et al. [8] identify 28

possible features of time series that are used for a rule-based forecasting system presented in [9]. This system weights and selects between the forecasting techniques random walk, linear regression, Holt's exponential smoothing and Brown's exponential smoothing. Parameters of the smoothing methods are also determined via rules. This method was submitted to the M3 competition ([1]) but did not provide convincing results.

Vokurka et al. [10] present another rule-based expert forecasting system, which performs automatic preprocessing of the series and automatically determines features of the time series to choose between a simple exponential smoothing, a dampened trend exponential smoothing and a decomposition approach as well as a simple-average combination of these three. This approach was able to improve upon a random walk model and the simple average combination.

The work presented here significantly extends the feature pool that was used in the publications introduced in the previous paragraph. Another focus lies on the functional diversity of the pool of forecasting and combination algorithms. The paper is organised as follows: Section two introduces the methodology of the underlying empirical experiments and justifies the choice of the forecasting and forecast combination algorithms. Section three describes the feature pool and feature selection processes. A relationship between the features and the performance of forecasting approaches is sought in section four. Section five concludes.

2 Underlying empirical experiments

A data set consisting of 111 monthly empirical business time series with 52 to 126 observations has been obtained from a Forecasting Competition conducted in 2006/2007 [11]. The task was to predict 18 future values. In previous work published in [12], experiments on this data set are summarised, using the last 18 observations of the provided time series for an out-of-sample error estimation. The forecast pool consisted of eight forecasting and seven forecast combination algorithms for single-step-ahead prediction as well as twelve forecasting and seven forecast combination algorithms for multi-step-ahead prediction. Where applicable, two approaches for parameter estimation have been considered, namely grid searching for a value that performs best in-sample (*tuned methods*) and setting the parameter to the middle of the parameter range (*untuned methods*).

As a number of the implemented forecasting and forecast combination methods shared the same functional approach, it was considered beneficial to choose just one from every group to reduce the number of class labels and gain clearer insights into which method works best for which time series. In an attempt to obtain a functionally diverse and well-performing method pool, the following methods have been selected:

One-step-ahead forecasting *Taylor's exponential smoothing (Taylor):* A modified dampened trend exponential smoothing was introduced in [13]. A growth rate and the level of the time series are estimated by exponential smoo-

thing and then combined with a multiplicative approach. All parameters are determined by a grid search or set to 0.5.
ARIMA: Autoregressive integrated moving average models (ARIMA) according to Box and Jenkins [6] are models with an autoregressive and a moving average part, fitted to differenced data. The original series as well as its first and second order differences are submitted to the automatic ARMA selection process of a MATLAB toolbox [14], choosing the prediction with the lowest in-sample error. The same process is implemented with undifferenced series only.
Neural network (NN): A feedforward neural network with one hidden layer containing 12 neurons, trained by a backpropagation algorithm with momentum has been implemented. Input variables are 12 lagged values of the time series. These characteristics have been selected based on findings of an extensive review of work using artificial neural networks for forecasting purposes by Zhang et al. [15]. Ten neural networks have been trained and their predictions averaged.
Variance-based combination model (VBW): Weights for a linear combination of forecasts are determined using past forecasting performance ([16]).
Variance-based pooling, three clusters (VBP): Past performance is used to group forecasts into two or three clusters by a k-means algorithm as suggested by Aiolfi and Timmermann [17]. Forecasts of the historically better performing cluster are then averaged to obtain a final forecast.
Regression combination (Regr): In regressing realisations of the target variable on forecasts over past periods, combination weights are estimated by a least squares approach with weights being restricted to be non-negative.

Multi-step-ahead forecasting *Taylor's exponential smoothing (Taylor):* This method is implemented as described for the one-step-ahead problem, but following a direct approach for the multi-step prediction, where n different models are trained directly on the multi-step problem.
ARIMA: An ARIMA model can natively provide multi-step-ahead forecasts, so the single-step method remains unchanged.
Neural network (NN): This was also implemented as described above, obtaining multi-step-ahead predictions by feeding the last forecast back to the model.
Simple average with trimming (SAT): This algorithm averages individual forecasts, only taking the best performing 80% of the models into account.
Variance-based pooling, two clusters (VBP): This is implemented as in the multi-step problem, only using two clusters instead of three.

3 Time series features and their selection

Based on the previous section, a classification task can be formulated as follows: Given a set of time series features, can we predict a) the best performing forecasting method, b) the best performing forecast combination method or c) whether or not combinations work better than individual methods? Each of the three problems can be investigated for single- and multi-step-ahead forecasting.

Table 1 summarises the resulting six problems, for each of which tuned and untuned individual methods as explained in section 2 can be used.

One-step-ahead	
best forecasting method:	3 classes: Taylor, ARIMA, NN
best combination method:	3 classes: VBW, VBP
best general approach:	2 classes: individual method or combination
Multi-step-ahead	
best forecasting method:	3 classes: Taylor, ARIMA, NN
best combination method:	2 classes: SAT, VBP
best general approach:	2 classes: individual method or combination

Table 1: Classification tasks, abbreviations referring to methods introduced in section 2.

Based on the publications cited above and a book by Makridakis et al. [7], a number of features listed in table 2 have been identified.

descriptive statistics	
abbreviation	description
slope	trend (absolute value of the slope of linear regression line)
std	standard deviation of de-trended series
stdrate	ratio between the standard deviation of the first and second half of the de-trended series
skew	skewness of series
kurt	kurtosis of series
sign	sign change measure (counting sign changes of de-trended series divided by its length)
length	length of series
pred	predictability measure according to [18]
nonlin	nonlinearity measure also according to [18]
frequency domain	
abbreviation	description
ff[1-3]	frequencies at which the three maximal values of the power spectrum occur
ff[4]	maximum value of the power spectrum of the fourier transform of the series
ff[5]	number of peaks not lower than 60% of the maximum peak
autocorrelations	
abbreviation	description
acf[1-12]	autocorrelations at lags 1-12
pacf[1-12]	partial autocorrelations at lags 1-12

Table 2: Feature pool

Including irrelevant features in a machine learning algorithm can cause degrading performance of the resulting model [19]. The use of redundant attributes may have the same effect. This is why one automatic and one judgemental feature selection algorithm have been used on the complete feature pool in order to generate a suitable subset of features. Judgementally, the following six features have been selected:

• The intuitive sign change measure, to capture volatility.

• The length of the series, as the number of observations available for training might influence the performance of methods.

• The nonlinearity measure, to quantify predictability of a series.

• The maximum value of the power spectrum of the fourier transform of the series, to identify a strong higher- or lower frequent component

• Partial autocorrelations at lag one and twelve, to capture nonstationarity and yearly seasonality if present.

The automatic method called "Subset Selection" was proposed in [20] and is implemented in the Weka collection of machine learning algorithms [19]. It belongs to the so-called filter methods, which are known for fast and efficient selection of features in a preprocessing step, independent of a learning algorithm. The quality of a feature subset is measured by two components: the individual predictive power given by correlation values and the level of intercorrelation among them. Searching the feature space is done using a Best First algorithm with an empty feature set as a starting point. All possible expansions are then evaluated and the best one is picked to be expanded again.

Using a ten-fold cross validation and selecting features that have been chosen in at least five of the ten calculations, tables 3 and 4 list the features selected for each of them.

Tuned methods	
class label	selected features
best individual forecast	slope, skew, nonlin, acf[1-11], pacf[1,3-4, 6-8, 10-11]
best combination	slope, std, acf[1,9-11], pacf[1,3,8,10-11]
individual vs combination	acf[11], pacf[6,10-11]
Untuned methods	
class label	selected features
best individual forecast	acf[4], pacf[3,8]
best combination	pacf[5,11]
individual vs combination	pacf[2-3,10]

Table 3: Features automatically selected for one-step-ahead forecasting

The tables show that the automatic approach generally chooses completely different features for each of the twelve sub-problems identified. Consequently it can be concluded, that there is no obvious feature that helps to decide for a suitable algorithm in every case. Appearing in seven cases, partial autocorrela-

Tuned methods	
class label	selected features
best individual forecast	skew, acf[7], pacf[5-6]
best combination	std, pacf[2,4,8]
individual vs combination	skew, acf[6], pacf[6]
Untuned methods	
class label	selected features
best individual forecast	ff[1-2,4], pacf[4-7,10]
best combination	std, pacf[2,5-9]
individual vs combination	acf[6], pacf[6]

Table 4: Features automatically selected for multi-step-ahead forecasting

tion at lag six is the feature that gets selected most, indicating that seasonality might be a factor that is important to many of the decisions.

4 Results

Decision trees have been selected as a simple machine learning method giving easily interpretable results. They are built in Matlab, choosing the minimum-cost-tree after a ten-fold crossvalidation. In the figures, the leaf to the left of a node represents the data that fulfils its condition, the leaf to the right hand side represents data that does not. The numbers following the methods in the leafs denote the number of times this particular method performed best on the data subset.

4.1 One-step-ahead

For one-step-ahead tuned forecasting methods, the trees in Figure 1 are created, both having a misclassification cost of 48.6%. Both essentially say the same thing: neural networks work better with yearly seasonality. This is not too surprising since yearly differences have not been taken for the ARIMA model, which works best with purely stationary data. No tree was built for combination methods, both feature sets suggest the regression method with a misclassification cost of 54.9%. The same occurs for the question of whether to use individual methods or combinations, combinations are suggested at a cost of 35.1%.

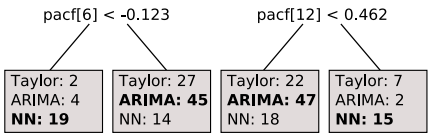


Fig. 1: Tree for tuned forecasting methods, left: automatically selected features, right: judgemental feature selection

Figure 2 shows minimum cost trees for untuned individual methods. The subset selection feature set suggests a neural network if the autocorrelation at lag four is below a certain number and an ARIMA model if it is above (cost 38.7%). Like for the tuned individual methods, the judgementally selected feature set suggests a neural network for series with stronger seasonality and an ARIMA model otherwise (cost 37.8%).

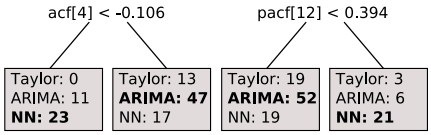


Fig. 2: Tree for untuned forecasting methods, left: automatically selected features, right: judgemental feature selection

For combinations, the subset method suggests the regression method (cost 52.2%), while the judgemental method produces a tree with two nodes (cost: 33.3%) shown in figure 3, which can be read as follows: For longer series, a regression approach seems to work best, while variance-based pooling works better for shorter series with a stronger negative partial autocorrelation at lag one. Variance-based weights are the best option for short series with a positive or small negative partial autocorrelation at lag one. It can be suspected that the regression approach that takes all individual methods into account might need more stable individual forecasts than the others, which cannot be provided by series with a smaller training set. The strong dynamic trimming carried out by variance-based pooling works best for more stationary series, while non-stationarity might be handled better with weights calculated based on past variance.

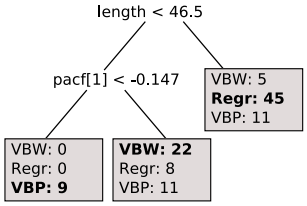


Fig. 3: Tree for untuned combination methods, judgemental feature selection

Comparing individual (fc) and forecast combination (fcc) methods, two trees are shown in figure 4, producing costs of 36.9% and 38.7%, respectively. The tree generated with features based on subset selection is not intuitively readable, having seemingly random partial autocorrelation values as conditions in the nodes. The other tree suggests individual forecasting methods for series with a stronger negative autocorrelation at lag one and combinations otherwise.

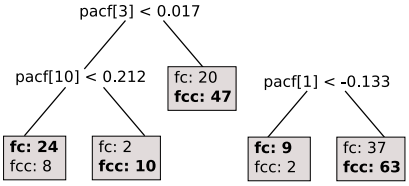


Fig. 4: Tree comparing untuned methods, left: automatically selected features, right: judgemental feature selection

4.2 Multi-step-ahead

Trees for tuned multi-step-combination models with quite high misclassification costs (59.0% and 57.4%) are shown in figure 5. The tree generated by subset selected features suggests methods depending on the partial autocorrelation at lag 4. Judgementally selected features produce a tree that suggests an ARIMA method for low-frequent zigzag and a neural network for a higher-frequent one.

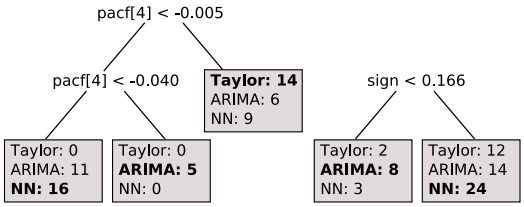


Fig. 5: Tree for tuned forecasting methods, left: automatically selected features, right: judgemental feature selection

For combination of tuned methods, the subset feature selection proposes simple average with trimming (cost 45.9%). Judgemental selection produces the tree shown in figure 6 with a cost of 40.9%, suggesting simple average with trimming for series with weaker seasonality and variance-based pooling otherwise. This might be explained by some methods not being capable of handling seasonality, which are hopefully dynamically removed from the combination in the variance-based pooling approach. Comparing individual to combination methods, both feature selection algorithms suggest individual methods (cost 29.5%).

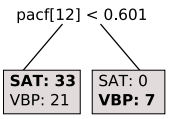


Fig. 6: Tree for tuned combination methods, judgemental feature selection

For untuned multi-step-ahead methods, a one-leaf tree is generated for most cases, suggesting a neural network as an individual method (cost: 55.7%), variance-based pooling as a combination (cost: 44.2%) and individual forecasts over combinations (cost 29.5%). Only judgemental feature selection for individual methods produces an actual tree (cost 47.5%) which is shown in figure 7. It suggests using neural networks for series with lesser nonstationarity indicated by the autocorrelation at lag one. On the other side of the tree, a neural network is again suggested for seasonal series, while series with lower seasonality and a higher nonlinearity measure are better predicted with Taylor's or the ARIMA method.

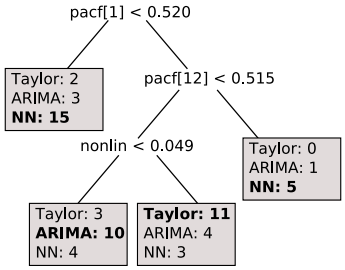


Fig. 7: Tree for untuned forecasting methods, judgemental feature selection

5 Conclusions

This paper investigates an automatic approach to use time series features for choosing a method that will work well for their forecasting. It extends the feature pool of previous work as well as the diversity of methods used as class labels. Both a judgemental and an automatic approach to feature selection have been employed. As a first interesting result, the automatic feature selection approach selected different features for every sub-problem, indicating that there is no obvious feature that always affects the performance of forecasting methods.

Summarising the results presented in section four, it can be seen that characteristics of time series can in some cases give an indication about which method might work best for forecasting its future values. Looking at features in the nodes of the trees, the partial autocorrelation at the lags one and twelve are often present, indicating that nonstationarity and seasonality of a series are important factors for choosing a prediction method. However, the seasonality issue also shows the importance of data preprocessing, because some of the differences in performances of the methods might not occur if quarterly, yearly or any other seasonality had been removed from the series in a preprocessing step.

However, not every sub-problem produced a decision tree that could easily be interpreted. This suggests that it could be beneficial to further extend the feature pool and selection of methods in future work, or that there must be

other mechanisms than just the characteristics of the time series that decide about success or failure of a forecasting method.

References

[1] S. Makridakis and M. Hibon. The M3-Competition: Results, Conclusions and Implications. *International Journal of Forecasting*, 16(4):451–476, 2000.

[2] J.H. Stock and M.W. Watson. A Comparison of Linear and Nonlinear Univariate models for Forecasting Macroeconomic Time Series. In R.F. Engle and H. White, editors, *Cointegration, causality and forecasting. A festschrift in honour of Clive W.J. Granger*, pages 1–44. Oxford University Press, 1999.

[3] Commentaries on the M3-Competition, October-December 2001.

[4] C.C. Pegels. Exponential Forecasting: Some New Variations. *Management Science*, 15(5):311–315, 1969.

[5] E. S. Gardner. Exponential Smoothing: The State of the Art. *Journal of Forecasting*, 4(1):1–28, January-March 1985.

[6] G.E.P. Box and G.M. Jenkins. *Time Series Analysis*. Holden-Day San Francisco, 1970.

[7] S.G. Makridakis, S.C. Wheelwright, and R.J. Hyndman. *Forecasting: Methods and Applications*. John Wiley, New York, 3rd edition, 1998.

[8] M. Adya, F. Collopy, J.S. Armstrong, and M. Kennedy. Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting*, 17(2):143–157, 2001.

[9] M. Adya, J.S. Armstrong, F. Collopy, and M. Kennedy. An application of rule-based forecasting to a situation lacking domain knowledge. *International Journal of Forecasting*, 16:477–484, 2000.

[10] R.J. Vokurka, B.E. Flores, and S.L. Pearce. Automatic feature identification and graphical support in rule-based forecasting: a comparison. *International Journal of Forecasting*, 12(4):495–512, 1996.

[11] NN3 Forecasting Competition [Online], 2006/2007. Available online: http://www.neural-forecasting-competition.com/ [13/06/2007].

[12] C. Lemke and B. Gabrys. Do we need experts for time series forecasting? In *Proceedings of the 16th European Symposium on Artificial Neural Networks, Bruges*, pages 253–258, 2008.

[13] J. W. Taylor. Exponential Smoothing with a Damped Multiplicative Trend. *International Journal of Forecasting*, 19(4):715–725, October-December 2003.

[14] Delft Center for Systems and Control - Software [Online], 2007. Available online: http://www.dcsc.tudelft.nl/Research/Software [13/06/2007].

[15] G. Zhang, B.E. Patuwo, and M.Y. Hu. Forecasting with Artificial Neural Networks: The State of the Art. *International Journal of Forecasting*, 14:35–62, 1998.

[16] P. Newbold and C.W.J. Granger. Experience with Forecasting Univariate Time Series and the Combination of Forecasts. *Journal of the Royal Statistical Society. Series A (General)*, 137(2):131–165, 1974.

[17] M. Aiolfi and A. Timmermann. Persistence in Forecasting Performance and Conditional Combination Strategies. *Journal of Econometrics*, 127(1-2):31–53, 2006.

[18] T. Gautama, D.P. Mandic, and M.M. Van Hulle. A novel method for determining the nature of time series. *IEEE Transactions on Biomedical Engineering*, 51, 2004.

[19] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[20] M. A. Hall. *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand, 1998.

Kernel Based Imputation of Coded Data Sets

Tapio Pitkaranta[1]

1- Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Information Processing Science

Abstract. Coded data sets can be used as compact representations of primary business processes. Data values that are missing from these data sets are a quality issue especially for secondary purposes that rely on such data. This study proposes a registry based machine learning algorithm for imputation of coded data sets. The proposed technique utilizes a kernel based data mining algorithm for efficient nearest neighbour queries. Preliminary results show that the algorithm could be used for routine and standard healthcare secondary processes.

1 Introduction

Coded data sets can be used as compact representations of primary business processes. In these data sets the data values belong to a classification, i.e. to a discrete finite set of possible values. For instance, in healthcare so-called minimum data sets are used as compact representations of patient care processes. Minimum data sets typically consist of different types of diagnosis and procedure codes together with basic information about the patient. Therefore the minimum data set contains the primary classification of patient care.

Secondary business processes used for monitoring and controlling the primary processes tend to rely on the data that is produced by the primary processes. Secondary processes apply different sorts of data aggregations and secondary classifications to capture relevant aspects of the primary processes. In healthcare systems there are several secondary purposes such as activity planning and monitoring, benchmarking, cost modelling, reimbursement and funding, service monitoring and clinical pathway development [4]. For these purposes a secondary classification is sometimes superimposed on the minimum data sets. Internationally a common secondary classification mechanism for such healthcare secondary purposes is the Diagnosis Related Grouping (DRG).

2 Problem Statement

The introduction of secondary classification systems into Finnish hospitals has raised debate about the underlying minimum data sets. Previous studies have reported various problems in the minimum data sets that are collected in Finland [1, 5, 6]. These minimum data sets can be manually recoded, however that requires reviewing the complete patient care documentation. However, this is labour intensive, requires expertise and information can be missing even from the complete documentation.

This study discusses a particular issue related to the minimum data sets, i.e. missing data. Sometimes diagnoses and procedures are missing from the minimum data sets. In these cases the minimum data sets do not give a complete description of patient care.

Calculation complexity criteria must be considered before developing data imputation mechanisms. In routine and standard healthcare secondary processes the calculation complexity criteria for data pre-processing (including the data imputation algorithms) can be strict. In practice this means that pre-processing cannot take days or weeks even with large amounts of data.

2.1 Data Space

The number of all possible minimum data sets is the number of all possible combinations of diagnosis and procedure codes. In this study the number of possible values in the diagnosis and procedure coding classifications was around 12000. Therefore, the number of different minimum data set combinations C (with 40 diagnosis and procedure codes) is:

$$C = \binom{ICD}{DG_{codes}} \cdot \binom{NCSP}{PR_{codes}} = \binom{12000}{40} \cdot \binom{12000}{40} = 2.8 \cdot 10^{230} \qquad (1)$$

The minimum data sets constitute a high dimensional data space. The data space is also sparse since there are a relatively small number of minimum data sets, i.e. patient cases, compared to the theoretical number of different minimum data set combinations. If the data is transformed into a binary matrix for the purpose of feeding it to a neural network, the number of dimensions will be a major problem. For the given 40 diagnosis and 40 procedure codes and 12000 alternatives for each code, there are around one million dimensions in the data. Even in this case the minimum data sets are simplified because same code values can occur multiple times, the order of the codes is partly significant and there is other information besides the diagnosis and procedure codes. Although dimension reduction and random projection methods can be used to scale down the dimensions of the data space, this study makes the assumption that neural networks such as SOMs are improper for this type of data.

2.2 Related Work

Previous studies have approached the problem by using additional patient documentation that is used to automatically create clinical codes. The term *computer aided coding* (CAC) is used to denote technology that automatically assigns codes from clinical documentation for a human to review, analyze, and use. There are a variety of methodologies employed by developers of CAC software to read text and assign codes. The software can use structured input or natural language processing. Even within the natural language processing range of products, there are a variety of approaches with varying levels of sophistication.

These include usage of external knowledge databases and feature extraction using self organizing maps (SOM) [12]. The methodology used has a tremendous impact on data transmission and the output reviewed by the coders [8]. Some studies have shown CAC software performing strongly in comparison with human coding [7]. Other studies have concluded that no productivity increase was achieved [8].

3 Multiple Imputation of Coded Data Sets

This section proposes a kernel-based data mining algorithm for multiple data imputation of minimum data sets. The proposed algorithm relies on machine learning principles and is based on the minimum data sets. The process of proposing missing data can be divided into two phases: first the incomplete minimum data sets must be discovered (data editing) and then a corrective piece of information must be inserted to complete the data set (data imputation).

3.1 Data Editing

There are various ways of doing data editing in the context of minimum data sets. One common way used in previous studies is to apply the secondary classifier for data editing purposes [6, 9, 10]. The secondary classification logic contains heuristics on whether the minimum data set contains inconsistencies.

3.2 Data Imputation

This study proposes using kernel-based data mining algorithms for the imputation of classified data sets. The applied kernel-based term vector algorithm locates k-nearest neighbours for the query vector, creates association rules for possible values for imputation from the k-nearest neighbours. The improper values for imputation are filtered using additional logic.

Kernel-based algorithms such as term vector analysis are used in high dimensional data spaces for calculating distances of two data sets. Frequently used techniques for locating nearest neighbours are distance measurements such as the Cosine Angle Distance (CAD) and Euclidean distance (EUD). These distance measurements have been reported to perform similarly in high dimensional data spaces for nearest neighbour queries [3].

Because of the calculation complexity requirements and high dimensional domain data space, this study adopts the term vector approach for locating the nearest neighbours from the existing knowledge bases. The term vector query is formed from part of the minimum data set that is supposed to be incomplete. This technique is flexible for weighting different parts of the term vector which can be assumed to be appropriate in the domain: some codes are more important than the others.

Figure 1 depicts a sample network how a collection of minimum data sets are interlinked by the diagnosis and procedure codes. The similarity between the query vector $Q = (q_1, q_2, ..., q_t)$ depicted in Figure 1, and document representing

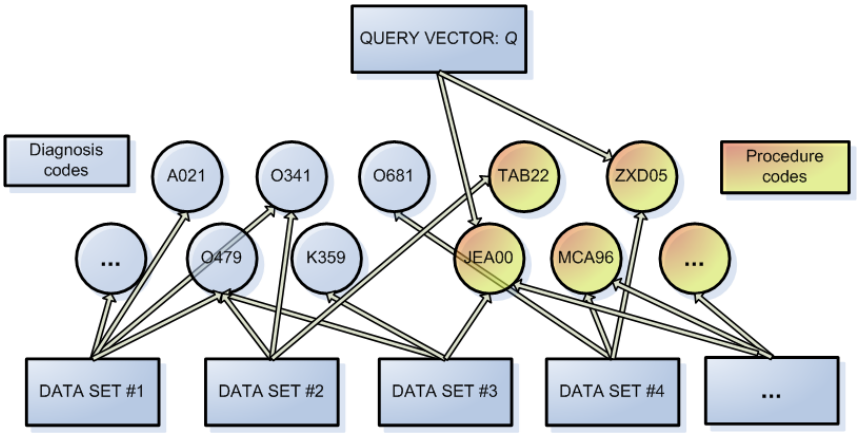


Fig. 1: A sample minimum data set network with diagnosis and procedure codes

the minimum data set $D_i = (d_{i1}, d_{i2}, ..., d_t)$ using corresponding query weights q_j for each term, is described by Equation 2:

$$s(q, d_i) = \frac{\sum_{j=1}^{t} (q_j d_{i,j})}{\sqrt{\sum_{j=1}^{t} q_j^2} \sqrt{\sum_{j=1}^{t} d_{i,j}^2}} \tag{2}$$

A common way for defining the document weights is described in Equation 3 and query weights in Equation 4 [13]. In these equations the $log(N/f_j)$ is the so-called *inverse document frequency* where N is the number of documents in the database and f_j is the number of documents that contain term t_j. Furthermore, the tf_{ij} is the *within document frequency* indicating the number of occurrences of term t_j in document i.

$$d_{ij} = tf_{ij} \cdot \log(\frac{N}{f_j}) \tag{3}$$

$$q_j = \begin{cases} \log \frac{N}{f_j} & \text{if term } t_j \text{ appears in the query;} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

These equations can be used to flexibly modify the weight of a particular term in the query vector. For instance in case of minimum data sets, a particular procedure or diagnosis code can be given a greater weight than patient basic

information such as sex and age. Furthermore, some data values can be left out from the query vector depending on the type of data that is being searched for.

From the result set sorted with the ranking mechanism, k-nearest neighbours are collected. From this set, an association rule table is formed to describe values and probabilities for imputation. Depending on the use case, different sorts of probability distributions can be utilized in the creation of the association rule table. A simple way is to search all terms from the k-nearest neighbours that do not exist in the original query vector Q and sort these terms based on their frequency within the neighbours. The resulting values in the association rule table can be further filtered using case specific logic, such as the secondary classifier, to achieve proper values for imputation.

3.3 Weights and Principal Components

The proposed algorithm can be used to impute various types of data not limited to minimum data sets. However, for the accurate nearest neighbour query, the query vector must be carefully selected. The query vector accuracy would benefit if the principal component could be identified from the data set. The principal component analysis as a general information processing topic can be time consuming especially in extremely high dimensional data spaces. Therefore it is convenient to provide the algorithm with additional information about the semantics of the data.

Since the values in the classified data sets correspond to a well-defined coding scheme, this sort of additional knowledge can be provided for the query mechanism. For instance in the case of minimum data sets surgical procedures can be given a greater weight than the codes representing blood samples. Furthermore, external weights and price lists can be utilized to further empower the query vector.

4 Empirical Study

This section presents results from an empirical test of the algorithm for multiple imputation of minimum data sets. As discussed, the proposed algorithm can be used for different purposes to impute different types of data. In this study, the algorithm was tested for imputing primary diagnoses for minimum data sets containing a surgical procedure. Previous studies have noted that surgical procedures are coded more accurately than the diagnosis codes [2, 11]. However, since the currently used secondary classification is heavily based on the diagnosis coding, it is possible that patient episodes that contain expensive surgical operations end up in an inappropriate patient group if the primary diagnosis is incorrect.

4.1 Methods

In this study, the algorithm was tested with material that was extracted from an existing data warehouse of one Finnish hospital district. The selected material

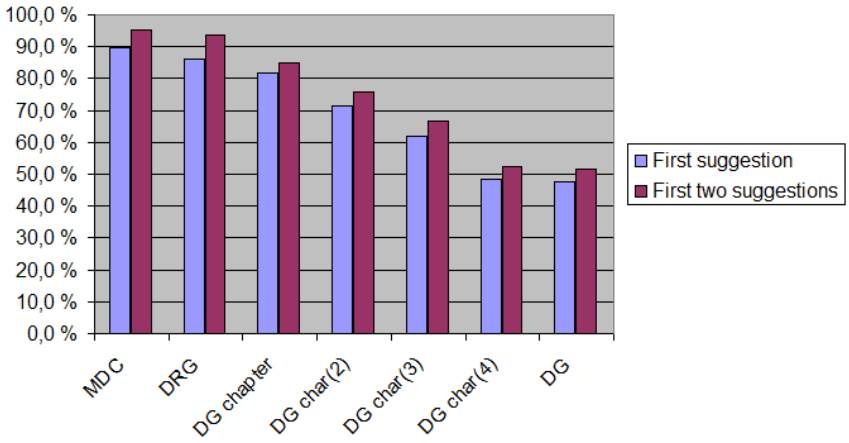


Fig. 2: Imputation accuracy in different granularity levels

represents inpatient data for one year. Since the proposed data imputation algorithm utilizes machine learning techniques, the algorithm needs to be trained with a knowledge base before it can be used. In this study the algorithm was trained using several different materials from Finnish hospital districts. These materials represent inpatient material of varying time periods. Together the training set contained several hundreds of thousands of minimum data sets.

First, a subset was created from the material with the selection criteria that the minimum data sets contained a surgical procedure. This sub-material was corrupted by removing all primary diagnoses. The algorithm was then applied for primary diagnosis imputation. The original material was used as a golden standard to evaluate the accuracy of the results. The imputation accuracy was evaluated using different classification granularity levels.

4.2 Results

The imputation accuracy results are depicted in Figure 2. As discussed, the diagnosis classification is very fine grained containing over 12 000 codes. Therefore the imputation accuracy is measured in Figure 2 using various classification granularity levels. These granularity levels include the fine grained diagnosis code, denoted by *DG* in Figure 2, and coarse grained diagnosis codes with four, three and two character precision. The imputation accuracy is measured also in the main chapter level of the diagnosis classification, denoted by *DG chapter* in

Figure 2, and Main Diagnostic Category (MDC) and Diagnosis Related Group (DRG) levels.

As can be seen from Figure 2, the imputation accuracy depends on the classification granularity level. This result is probably due to the sparse data space: the granularity of the classification instruments are overwhelming compared to the number of real data sets. For the most fine granular diagnosis classification the imputation accuracy is 47.6%. The accuracy rises for the coarse grained diagnosis classifications: for instance for the two character level diagnosis codes the accuracy is 71.4%.

From Figure 2 it can be noted that for the MDC and DRG, that are used for secondary purposes, the accuracy is more precise than for primary classification. For MDC the primary diagnosis imputation accuracy is 89.8% for the first proposed value and 95.3% when one of two first proposed alternatives are used. For the DRG the corresponding accuracies are 86.2% and 93.6%.

4.3 Evaluation

Minimum data set imputation is a challenging topic in many ways. The data space is high dimensional and there are several aspects that were not addressed in this study. One of these aspects is time. The occurrence of some diseases varies depending on the time of year. Another issue is data editing, i.e. discovering cases in which information might be missing.

The proposed imputation algorithm relies on machine learning principles. For accurate imputation the algorithm should be trained using a golden standard, i.e. with data that is correct. It is clear that several hundreds of thousands of minimum data sets do not cover the data space thoroughly. It can also be assumed that the imputation results would be more accurate if the training database would contain more data. Furthermore, in the context of minimum data sets the golden standard should be created using several independent healthcare professionals who would evaluate each data set separately. However, with such a method it is impossible to create a golden standard that would cover such a high dimensional data space. Therefore, in this study the golden standard is normal data extracted from hospitals without extra evaluation about the quality of the data. Previous studies have reported various types of problems in the minimum data sets that are collected in Finland and similar problems can be expected to

Table 1: Incorrect primary diagnosis

Minimum Data Set	**Value**	**Label**
Primary Diagnosis	O21.0	Mild hyperemesis gravidarum
Procedure	MBA00	Vacuum aspiration from uterus after delivery or abortion
Age	35	Patient age: 35 years
Sex	Female	
Length of stay	6	

be present in the data that is used in this study [1, 5, 6]. Previous studies have reported that the accuracy of the primary diagnosis varies between 60%-95% depending on the audit criteria. Therefore it is clear that there are minimum data sets in which the primary diagnosis is incorrect in the golden standard used in this study.

An example case illustrating this methodological issue is listed in Table 1, which lists an example minimum dataset in which the primary diagnosis is incorrect. In this minimum data set, the primary diagnosis is probably the physician's first assumption about why the patient sought medical care. However, the performed procedure code *MBA00* indicates that during care a normal delivery or an abortion has been performed on the patient. In either case, a new primary diagnosis should be given for the patient, since the first diagnosis is not the reason for the patient care. However, for some reason a new diagnosis has not been assigned and the case listed in Table 1 cannot be corrected without adding information to the record.

When the primary diagnosis is removed from the patient case listed in Table 1 and imputed with implemented algorithm, the first imputation is the diagnosis code *O02.1: Missed abortion*. The imputed minimum data set is listed in Table 2. It is clinically clear that this code is more correct as primary diagnosis than the original primary diagnosis *O21.0: Mild hyperemesis gravidarum*. However, as the original material is used as the golden standard, this case is marked as incorrectly imputed.

5 Conclusion

Secondary business processes are used to support, monitor and control primary business processes. Secondary processes tend to rely on the data that is produced by the primary processes. Missing data values distort secondary business processes since the data gathered does not accurately reflect relevant aspects of the primary processes.

This study proposed a registry based machine learning algorithm for coded data imputation. The algorithm was tested with minimum data sets from healthcare. The preliminary results show that the imputation accuracy may be sufficient for secondary processes that apply different sorts of data aggregations and

Table 2: Imputed primary diagnosis

Minimum Data Set	**Value**	**Label**
Primary Diagnosis	O02.1	Missed abortion
Procedure	MBA00	Vacuum aspiration from uterus after delivery or abortion
Age	35	Patient age: 35 years
Sex	Female	
Length of stay	6	

secondary classifications. With minimum data sets, the secondary classification accuracy was 86.2% for the first proposed value and 93.6% when one of two proposed values was used.

6 Future Work

Future work includes enhancing the imputation accuracy for different use cases and includes developing the selection of query terms and weights depending on the type of information that can be assumed to be missing. This study applied external knowledge about the semantics of the procedure classification to identify principal components. It can be anticipated that there are many cases, in which instead of a single principal component such as an expensive surgical procedure, there are multiple minor observations that are relevant for the imputation.

References

[1] S. Aro, R. Koskinen, and I. Keskimaki. Sairaalastapoistorekisterin diagnoosi-, toimenpide- ja tapaturmatietojen luotettavuus. *Duodecim 106*, pages 1443–1450, 1990.

[2] C. Colin, R. Ecochard, F. Delahaye, G. Landrivon, P. Messy, E. Morgon, and Y. Matillon. Data quality in a drg-based information system. *International Journal for Quality in Health Care*, (6):275–280, 1994.

[3] Q Gang, S. Sural, Y. Gu, and S. Pramanik. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1232–1237. Michigan State University, ACM, 2004. ISBN: 1-58113-812-1.

[4] Juvonen I. Hakkinen U. Jarvelin J Pekurinen M. Junnila M., Linna M. *Sairaaloiden tuottavuuden kehitys 2001-2005.* National Research and Development Centre for Welfare and Health (STAKES), May 2007. ISSN: 1459-2355.

[5] R. Lahti. *From Findings to Statistics: An Assessment of Finnish Medical Cause-of-death Information in Relation to Underlying-cause Coding.* PhD thesis, University of Helsinki, 2005.

[6] A. Rauhala and M. Linna. Coding of diagnoses in finnish specialised health care - do the statistics reflect medical or coding practices? *Finnish Medical Journal*, 32(62):2785–2790, 2007. Article is in Finnish.

[7] P. Resnik, M. Niv, M. Nossal, G. Schnitzer, J. Stoner, A. Kapit, and R. Toren. Using intrinsic and extrinsic metrics to evaluate accuracy and facilitation in computer-assisted coding. In *Perspectives in Health Information Management, Computer Assisted Coding Conference*. American Health Information Management Association, 2006.

[8] C. Servais. Computer/assisted coding for inpatients / a case study. In *Perspectives in Health Information Management, Computer Assisted Coding Conference*. American Health Information Management Association, 2006.

[9] Socialstyrelsen. *Kodningskvalitet i patientregistret, Slutenvård 2005*. Number 2007-125-13. 2007. ISBN 978-91-7164-281-3, Available (in Swedish) at URL: http://www.socialstyrelsen.se/.

[10] Socialstyrelsen. *Kvalitet och innehåll i patientregistret, Utskrivningar från slutenvård 1964-2006 och besok i oppenvård (exklusive primarvårdsbesok) 1997-2006*. Number 2008-125-1. 2008. ISBN 978-91-7164-281-3, Available (in Swedish) at URL: http://www.socialstyrelsen.se/.

[11] J. Stausberg, P.-D. Med, D. Koch, J. Ingenerf, R. Nat, and Betzler. Comparing paper-based with electronic patient records: Lessons learned during a study on diagnosis and procedure codes. *Journal of the American Medical Informatics Association*, (10):470–477, 2003.

[12] D. J. Tufts-Conrad, A. N. Zincir-Heywood, and D. Zitner. Som - feature extraction from patient discharge summaries. In *Symposium on Applied Computing*, pages 263–267. ACM, 2003. ISBN: 1-58113-624-2.

[13] R. Wilkinson and P. Hingston. Using the cosine measure in a neural network for document retrieval. *ACM*, pages 202–210, 1991.

Reliability of ARMA and GARCH models of electricity spot market prices

Piotr Ptak, Matylda Jabłońska, Dominique Habimana and Tuomo Kauranne *

Lappeenranta University of Technology - Dept of Mathematics
P.O.Box 20, 53851 Lappeenranta - Finland

Abstract. Electricity spot market price is notoriously difficult to predict because of the high variability of its volatility that results in prominent price spikes, interlaced with more Gaussian behavior. Such varying volatility has prompted researchers to use GARCH modeling to forecast spot prices. In this article, we study the reliability of an optimally chosen GARCH and its accompanying ARMA model of two electricity spot market price time series using a Markov Chain Monte Carlo (MCMC) method. The MCMC method is used to estimate the parameters of the ARMA-GARCH model. It appears based on this analysis that even an optimally chosen ARMA-GARCH model is not sufficient to explain the behavior of electricity spot market price.

1 Introduction to electricity spot markets

Nordic power suppliers generated around 397,6 TWh last year, 40% of which came from Sweden, 35% from Norway, 16% from Finland and remaining 9% from Denmark. Most energy producers try to keep flexibility between different energy sources, mostly to diversify raw materials price risk. Table 1 presents repartition of electric energy origins among the Scandinavian countries.

Table 1: Different types of energy sources in Scandinavia.

Country	Hydropower	Nuclear Power	Other thermal sources (coal, gas)	Other renewable sources (wind)
Norway	99%		1%	
Finland	20%	33%	47%	
Denmark			81%	19%
Sweden	46%	42%	12%	

Electricity spot markets have been studied widely over the last twenty years due to the complex structure of electricity price time series [1]. Electricity prices on real-time markets are both highly volatile and difficult to predict. However, ongoing analyses of spot markets are conducted in order to make markets as close to perfect as possible. The main obstacle is that techniques of calculating electricity prices differ significantly in different countries. Nevertheless, the aim is to set the prices based on day-ahead and hour-ahead orders, so that the balance between supply and demand is met.

*This work has been supported by the Tekes MASI programme and by Fortum Inc.

In spite of being highly volatile, electricity prices have some visible statistical features. Firstly, they are highly correlated with temperature and hydrological conditions - the higher is the precipitation, the cheaper is electricity. Secondly, the prices are extremely dependent on demand. When power generation is below the adequate level, prices rise. This forces buyers to consume less and suppliers to increase production. When supply is sufficient, prices drop, resulting in lower power generation and ordinary consumption levels.

Spot markets are exchange markets where the exchange of takes place within up to two working days after striking a deal. This characterizes equally share, bond, currency and commodity exchanges. Electricity trading is one of the most significant spot markets. However, there is one main feature which distinguishes electricity from other types of exchangeable stock. Usually differences between demand and supply can be managed by storage capacity. Unfortunately, electricity is something that cannot be kept in a warehouse. In this manner, spot trading provides a possibility of almost permanent balance between supply and demand.

2 Spot trading on NORDPool/NEPool

In 1996 the first international electric power exchange was set up. The main goal was to create a common Nordic market with a guarantee of strong competition between suppliers in the area. That was possible due to a wide diversity of Scandinavian energy sources: hydropower (Norway, Sweden, Finland), nuclear power (Sweden, Finland), thermal power (Sweden, Finland, Denmark) and significantly increasing wind power (Denmark). Nowadays, Nordic Power Exchange (NORDPool) is owned by two Scandinavian grid companies: Norwegian Statnett SF and Swedish Affärsverket Svenska Kraftnät, 50 per cent of shares for each.

The part of NORDPool's activity, that we are interested in, is Elspot - the spot floor, collecting next day's demands for electric power for each of the 24 hours of the following day and set the system prices for that day. A strict daily schedule is obligatory for all market participants. It covers receiving buy/sell offers from participants, system prices' calculation, data verification and discussion on probable participants' concerns and, finally, next day's prices publication by the exchange.

The New England Power Pool (NEPool) was formed in 1971 as a six-state region electricity coordinator. Though it is a corporation (not a stock exchange) its most important role is to provide spot market trading, which will match electric power supply and demand. Similarly to NORDPool, hour-ahead and day-ahead orders are used in estimating the system prices, which should be a compromise between buyers' and sellers' expectations. Moreover, the Pools are of a not-for-profit character. Their goal is to work out electricity prices in order to match demand and supply. In addition they have strict policies forbidding any professional connections between employees and companies trading in the Pools.

Both Pools provide an interesting data set for mathematical modelling. Their unique features emerge from the impossibility of storing electric energy.

3 Time series forecasting

Classical Box-Jenkins time series methods have been extended by many new features in the hope of making them apply to time series with more complex behavior. Typical Box-Jenkins methods [2], such as ARMA and ARIMA forecasting, are based on an underlying assumption of ergodicity over some time scale, and on linear dynamics.

In practice, these assumptions are hard to verify and one often resorts to empirical trial and error in finding a suitable model and hoping that the residuals it leaves do not display any significant structure.

More recently, it has become computationally possible to study the validity of such assumptions by Monte Carlo simulation. A particularly appropriate variant is the Markov Chain Monte Carlo method that can be used to study the covariance of model parameters as well as the robustness of its forecasts by treating ARMA and GARCH model parameters as samples from some distribution.

3.1 GARCH models built upon ARMA models

In a classical time series approach, one of the biggest challenges is to provide a mathematical explanation of changing volatility in the data. Since returns of electricity price data shows heteroscedasticity, i.e. volatility that varies in time, we use (Generalized) Autoregressive Conditionally Heteroscedastic (G)ARCH fitting [2, 3]. These types of models are widely used for time series that have variance varying with time. Financial data sets are often characterized by so-called variance clustering [2, 4], which means noticeable periods of higher and lower disturbances in the series.

An ARCH or GARCH model is used to complement an underlying ARMA model. An ARMA model is just a GARCH model that assumes homoschedasticity, i.e. a constant variance. A GARCH model is therefore applied to the residual left by the ARMA model.

An autoregressive conditional heteroscedasticity model represents the variance of a current error term as a function of variances of error terms at previous time periods. ARCH simply describes the error variance by the square of error at a previous period.

In general, an ARCH(Q) model is represented as follows:

$$u_t = C + \sigma_t v_t$$

$$\sigma_t^2 = K + \alpha_1 u_{t-1}^2 + \ldots + \alpha_Q u_{t-Q}^2$$

where:

- C is a constant in error term
- $v_t \sim N(0,1)$

- u_t are the return residuals (differences between the base ARMA model and original returns)
- σ_t^2 is the variance of residuals in time step t
- u_{t-i}^2 is the squared error term from i-th lag

A model is called generalized autoregressive and conditionally heteroscedastic (GARCH), if a second autoregressive moving average model (ARMA model) is used to represent error variance. A GARCH(P,Q) model is given by:

$$u_t = C + \sigma_t v_t$$

$$\sigma_t^2 = K + \alpha_1 u_{t-1}^2 + \ldots \alpha_Q u_{t-Q}^2 + \beta_1 \sigma_{t-1}^2 + \ldots + \beta_P \sigma_{t-P}^2$$

where:

- σ_{t-i}^2 is the variance from i-th lag

Moreover, except the conditional variances estimated in the model for every time step t, there is an unconditional variance of the series which can be expressed by the following formulae:

$$\sigma^2 = \frac{K}{1 - \sum_{i=1}^{Q} \alpha_i - \sum_{i=1}^{P} \beta_i}$$

The conditional standard deviation forecast changes from period to period and approaches the unconditional standard deviation. In the case of stationary ARCH/GARCH forecasting, predicted magnitudes for conditional variances always converge to the unconditional ones. Moreover, for estimation in heteroscedastic models a maximum likelihood method (unlike to ARMA methods) needs to be employed instead of ordinary least squares.

4 MCMC for time series

Markov Chain Monte Carlo (MCMC) techniques are numerical computation methods that can be used to estimate unknown parameters of ARMA(P,Q)-GARCH(P,Q) models which will be constructed for both NORDPool and NEPool spot markets. These techniques can be extended up to several estimates in any given model. MCMC techniques are also used to construct the distributions of unknown parameters based on random variables generated from specific well known distributions, as described in a Bayesian formulation of any problem [5]. MC methods are used to sample random numbers from different probability distributions.

When one wants to study a particular problem, an MCMC method is constructed in such way that it generates a random sample from given distributions. In general, the prior distribution contains the prior knowledge about the

unknown parameters given any model. A good selection of the prior distribution results in the best parameters known to be more probable than others. In Markov Chain Monte Carlo methods, the main idea is to create a Markov Chain using random sampling so that the created chain has the posterior distribution as its unique stationary distribution, i.e. the MCMC methods create ergodic Markov Chains meaning that the process will end up in having the same stationary distribution independent of the initial distribution.

4.1 Random Walk Metropolis Algorithm

It has been shown that with too wide a proposal distribution many of the candidate points are rejected and the chain stagnates for long periods and the target distribution is reached slowly. On the other hand, when the proposal distribution is too narrow, the acceptance ratio is high but a representative sample of the target distribution is achieved slowly. A very practical way for solving this issue takes the previously simulated value into account when the proposal is constructed.

Step 1: Initialization

- Choose θ_0, then set $\theta_{old} = \theta_0$
- Choose the covariance matrix C
- Choose the length of the chain M, and set $i = 1$

Step 2: Acceptance step (Metropolis step)

- Choose sample θ_{old} from $\mathrm{N}(\theta_{old}, C)$ and u from $\mathrm{U}[0,1]$
- Calculate $\mathrm{SS}_{\theta old}$ and $\mathrm{SS}_{\theta new}$
- If $\mathrm{SS}_{\theta new} < \mathrm{SS}_{\theta old}$ or $u < e^{-\frac{1}{2\sigma^2}(\mathrm{SS}_{\theta new} - \mathrm{SS}_{\theta old})}$, set $\theta_i = \theta_{new}$. Else set $\theta_i = \theta_{old}$
- if $i < M$, set $i = i + 1$ and go to step 1. Else, stop the algorithm [5].

Where

- θ_0 is a vector of initial parameter values of the model;
- θ_{old} is a vector of the previous sampled parameter values;
- θ_{new} is a vector of new sampled parameter values;
- M is the length of the chain;
- i is the number of iterations;
- u is the random value;

- $SS_{\theta_{old}}$ is the total sum of squares of previous sampled parameter values;
- $SS_{\theta_{new}}$ is the total sum of squares of new sampled parameter values.

In the algorithm the proposal width is the covariance matrix C of the Gaussian proposal distribution, or variance in one dimensional case. The problem of how to choose a proposal distribution is now transformed into the problem of choosing the covariance matrix C so that the sampling is efficient. In general, this is done by choosing a fixed covariance matrix by hand, by using some heuristic or "trial and error" strategy. But recently, some new techniques based on modifications of the Metropolis algorithm have been introduced in order to update the covariance matrix, like adaptive proposal (AP) and adaptive Metropolis (AM) [6].

4.2 Initialization of MCMC

When the Random Walk Metropolis algorithm with a Gaussian proposal distribution is used, the covariance matrix should be defined. It is important to choose the starting point θ_0 for the convergence rate. In a nonlinear model the starting point for an MCMC implementation is

$$\theta_0 = \min_{\theta} \sum_{i=1}^{n} \left(y_i - f\left(x_i, 0\right)\right)^2$$

where

- i is measurement index;
- θ is a vector of unknown parameter values;
- y_i represents the measurement vector;
- x_i represents the control variable.

The covariance matrix of the Gaussian proposal can be chosen by trial and error. However, it is useful to use the covariance approximation obtained from linearization. This means that the model is linearized and then the formula from linear theory

$$\widehat{C} = \sigma^2 \left(X^T X\right)^{-1}$$

is used. Where X is a vector of all control variables in the model.

In the case of NORDPool and NEPool time series we use MCMC techniques to sample the parameter values of ARMA(P,Q)-GARCH(P,Q) models based on the estimated parameter values of constructed models as inputs of MCMC methodology. Finally, we compare the standard errors associated to the estimated parameters with MCMC errors and test the reliability of forecasts by comparing MCMC simulated predictions to original data.

5 Estimating NORDPool/NEPool return series

Estimation and Forecasting procedures are based on two sets of data. First set comprises a total of 289 weekly points of historical spot prices for NORDPool. NEPool data set of daily prices lasts over 2551 days, so nearly 7 years. Use of GARCH technique requires returns as an input data. Both original time series and returns for NORDPool and NEPool are shown in Figure 1.

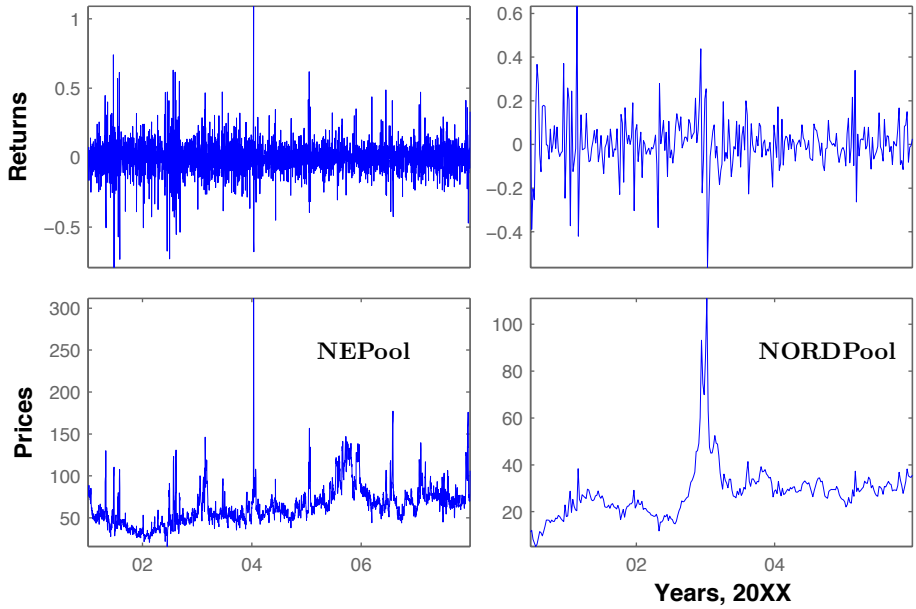


Fig. 1: Original series and its returns.

We can see that both sets are build of clusters with different variation of amplitude. Peaks are common components of energy spot prices. Due to their appearances, such signals are difficult to estimate by basic mathematical tools.

Peaks are undesired because of their non-differentiable nature. Use of Stochastic Differential Equations is impossible and one has to address this problem with methods of discrete type.

5.1 Identifying GARCH coefficients

First step is to examine autocorrelation and partial autocorrelation functions of the given data sets. These functions are depicted in Figure 2. As we can see both correlation and partial correlation at different lags are not very high and reach -0.17 for NEPool data set.

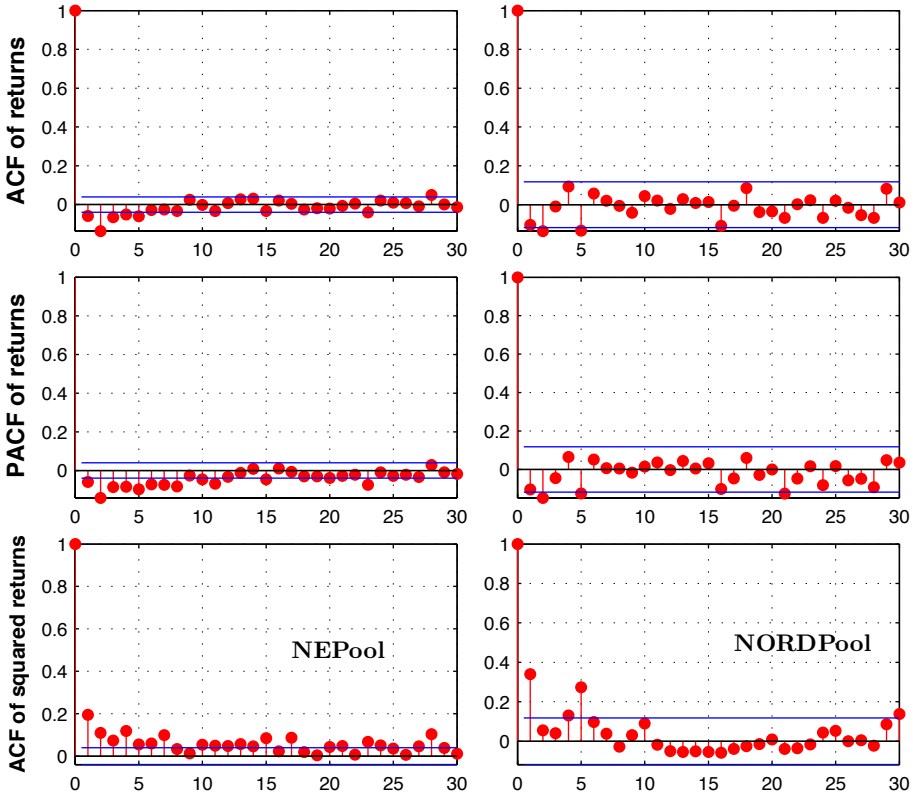


Fig. 2: Significantly higher correlation for squared returns.

Decision on type of model adequate for data comes partially from two tests: Engle's hypothesis test [7] for presence of ARCH/GARCH effects and Ljung-Box Q-statistic lack-of-fit hypothesis test [8]. The former examines a signal for a presence of GARCH components. The later checks if a signal includes ARMA effects.

Ljung-Box test verifies if there is a significant serial correlation in the raw returns for NORDpool and NEPool tested for 1 to 20 lags of the ACF at the 5% level of significance. The same test for squared returns indicates that both NORDPool and NEPool contain significant serial correlation.

Engel's test for the raw returns of NORDPool and NEPool rejects hypothesis that both series do not contain ARCH effect at the 5% level of significance. Squared returns of NORDPool do not include ARCH effect whereas squared returns of NEPool indicate presence of this effect.

Therefore, the presence of heteroscedasticity for NEPool indicates that GARCH modeling is appropriate.

5.2 Model fitting

This section describes a way to find a good GARCH model for the NEPool data. It also describes a criteria function build on Schwarz's Bayesian information criteria (SBIC), see [2]. Engel's and Ljung-Box tests give an output in a binary form, 1 or 0. Here, zero indicates lack of GARCH/ARMA effect in the series, while one indicates its presence. The SBIC is formulated as follows:

$$\text{SBIC} = \log\left(\sigma_{res}^2\right) + \frac{k}{T} \cdot \log\left(T\right)$$

where:

σ_{res}^2	variance of residuals between returns and its fitted model
k	number of parameters of GARCH model
T	length of tested time series

We suggest a new information criteria function, called SLEIC:

$$\text{SLEIC} = \left[\text{SBIC} \cdot \left(1 + \frac{\alpha}{2L} \sum_{i=1}^{L} \left(H_{1,i} + H_{2,i}\right)\right)\right]^{-1}$$

where:

SLEIC	information criteria function based on Schwartz-Bayesian information criteria, Ljung-Box test and Engel's test
$H_{1,i}$	vector of logical outputs for Ljung-Box test, $i = 1, 2, ..., 2L$
$H_{2,i}$	vector of logical outputs for Engel's test, $i = 1, 2, ..., 2L$
α	importance coefficient of Ljung-Box and Engel's tests
L	number of lags analyzed by Engel's/Ljung-Box tests

To find an appropriate model for both Pools, we maximize SLEIC function while varying orders P, Q, R and M of GARCH(P,Q) and ARMA(R,M) models.

$$\max_{P,Q} \text{SLEIC}(res, k, H_1, H_2)$$

Figure 3 depicts the information criteria level (SLEIC) with respect to model complexity. Level of information criteria for NEPool returns is higher that for the NORDPool ones. It is due to lack of ARCH effect within squared returns of NORDPool series, i.e. no heteroscedasticity. Chosen models for NEPool and NORDPool are ARMA(1,1) GARCH(2,1) and GARCH(2,1), respectively.

The difference in the shapes of the SLEIC values for NordPool and NEPool is likely a result of the different length of the time series. Our NordPool series only contains 250 values, whereas the NEPool comprises 2500 values. NEPool data can therefore be modelled reliably by many more GARCH models than the sparse NordPool data set we have. This fact is reflected also in the higher values of the SLEIC function for NEPool

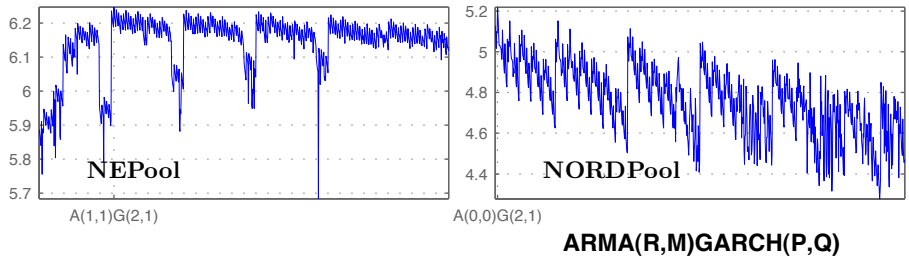


Fig. 3: Information criteria SLEIC with subject to realizations of different GARCH models.

SLEIC level analysis was performed for all possible ARMA and GARCH models up to ARMA(5,5) and GARCH(5,5), which results in 850 realizations. Explicit formulas for optimal models are:

NEPool
$y_t = -1.206 \cdot 10^{-4} + 0.6844 \cdot y_{t-1} - 0.9096 \cdot \varepsilon_{t-1} + \varepsilon_t$
$\sigma_t^2 = 9.7011 \cdot 10^{-4} + 0.2758 \cdot \sigma_{t-1}^2 + 0.4713 \cdot \sigma_{t-2}^2 + 0.1943 \cdot \varepsilon_{t-1}^2$

NORDPool
$y_t = 8.345 \cdot 10^{-3} + \varepsilon_t$
$\sigma_t^2 = 2.623 \cdot 10^{-3} + 0.373 \cdot \sigma_{t-2}^2 + 0.516 \cdot \varepsilon_{t-1}^2$

5.3 Post-estimation analysis

To examine chosen models both tests from Section 5.1 should be applied to residuals resulting from difference between returns and series of fitted model.

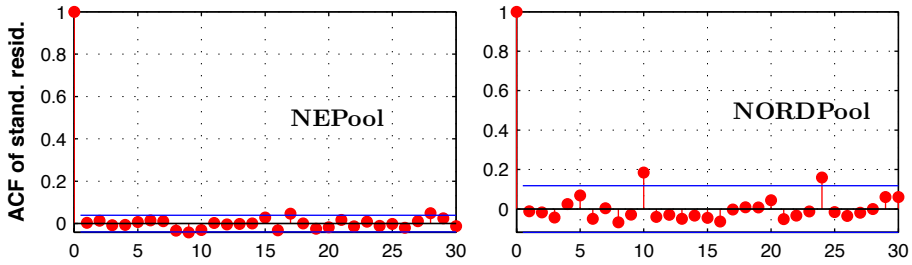


Fig. 4: Autocorrelation of standardized residuals.

Here by standardized residuals we mean the innovations divided by their conditional standard deviation. Tests for presence of GARCH/ARMA effects show that neither of standardized residuals of Pool series contains these effects.

6 Results: statistics and reliability of forecasts

6.1 Scatter plots and histograms of the sampled parameters

Since we found the most appropriate models and estimated their parameters, it is advisable to perform verification of the estimates reliability. Employing the MCMC methodology, we state the initial parameter values $\theta_{0,ne}$ as a vector of the estimated coefficients from ARMA(1,1)-GARCH(2,1) model for NEPool expressed as

$$\theta_{0,ne} = [\psi_{0,ne}\, \phi_{0,ne}\, C_{ne}\, K_{ne}\, \alpha_{1,ne}\, \beta_{1,ne}\, \beta_{2,ne}]^T$$

where

- $y_t = \lambda_{0,ne} + \lambda_{1,ne} y_{n-1} + \varepsilon_t$
- $\epsilon_t = C_{ne} + \sigma_t v_t E$
- $\sigma_t^2 = K_{ne} + \alpha_{1,ne}\epsilon_{t-1}^2 + \beta_{1,ne}\sigma_{t-1}^2 + \beta_{2,ne}\sigma_{t-2}^2$

and from GARCH(2,1) model for NORDPool as

$$\theta_{0,no} = [C_{no}\, K_{no}\, \alpha_{1,no}\, \beta_{1,no}\, \beta_{2,no}]^T$$

where

- $y_t = \lambda_{0,no} + \varepsilon_t$
- $\varepsilon = C_{no} + \sigma_t v_t$
- $\sigma_t^2 = K_{no} + \alpha_{1,no}\varepsilon_{t-1}^2 + \beta_{1,no}\sigma_{t-1}^2 + \beta_{2,no}\sigma_{t-2}^2$

Since the prior distribution for the unknown parameters θ is assumed to be Gaussian, it is treated as an extra sum of squares, then,

$$\mathrm{SS}_{new} = \sum_{i=1}^{p} \left(\frac{\theta_i - \mu_i}{v_i} \right)^2$$

where

- μ_i is the average value of the sampled parameter values at iteration i;
- v_i is standard deviation of the sampled parameter values at iteration i;
- $\theta_i \sim \mathrm{N}(\mu_i, v_i^2)$, that is, independent prior specification for θ.

After generating parameter chains with a length of 5000, we study their pair wise joint distributions, to reveal possible correlation between estimated parameters. We find that correlation coefficients for NEPool model vary from -0.9 to 0.47. This fact shows a significant level of correlation. Similarly, we analyze

NORDPool estimates and obtain coefficients with a range from approximately -0.72 to 0.44. Given results violate MCMC assumptions that require model parameters to be uncorrelated. This violation is a sign of non-ergodicity present in the residuals of GARCH models.

A next step is to study the parameters' histograms to verify the character of their distributions. These do not follow MCMC theory either – distributions of some parameters appear to be non-Gaussian for both models. This can be easily seen from Figures 5 and 6 presenting histograms and pair wise scatter plots for NEPool and NORDPool respectively.

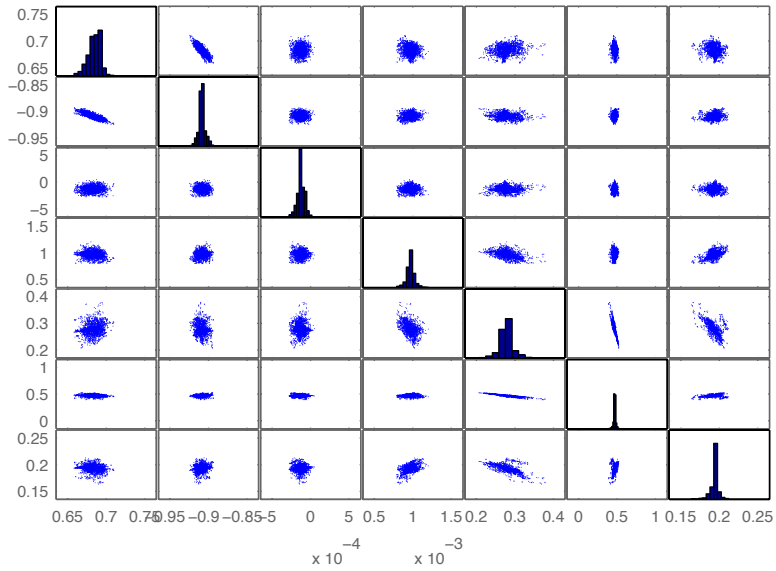


Fig. 5: Pair wise scatter plots for NEPool model parameters.

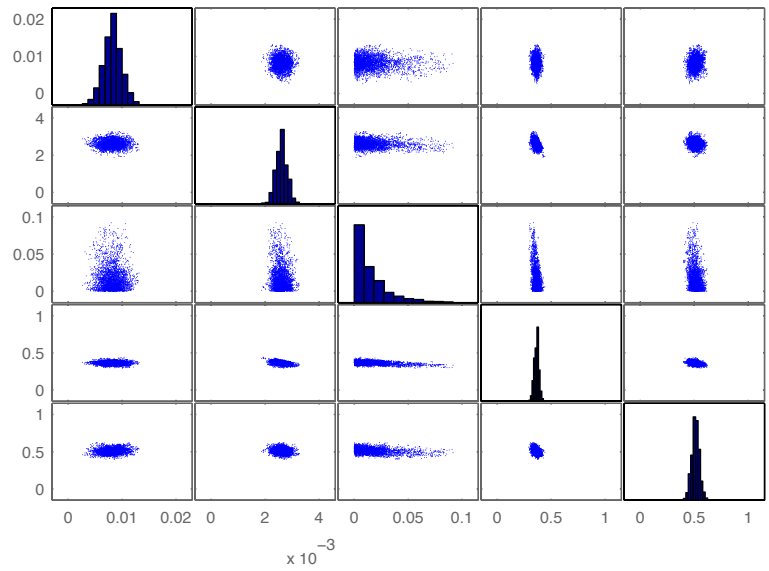


Fig. 6: Pair wise scatter plots for NORDPool model parameters.

One reason to the non-Gaussian distribution reflected by parameter covariance is the constraint of non-negativity imposed upon most parameters, which were bounding ranges of prior distributions.

6.2 Predictive distributions of sampled price returns

MCMC methods are based on random sampling and result in empirical distributions for unknown parameters. Moreover, it is possible to sample values for model prediction at different points and construct a distribution also for the response curves of the model, called 'predictive distributions', which give the information related to uncertainties in unknown parameters.

In case of NEPool spot market, a predictive distribution was constructed based on the sampled values for model prediction in terms of price returns, where 22 values were predicted as shown in Figure 7.

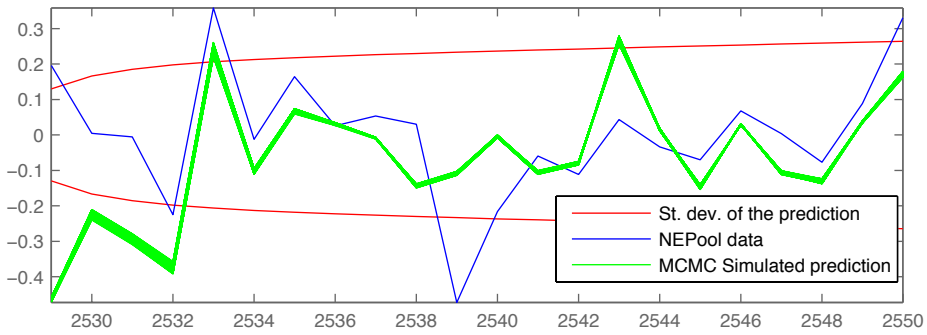


Fig. 7: Predictive distribution of price returns for NEPool series.

Figure 7 shows that the predictive distribution for the price returns will most likely lie inside the calculated bounds. However, we can see that the longer the forecasting horizon is, the more uncertainty predicted values have. On the other hand, the posterior distribution of the forecast is concentrated around the initial prediction. Figure 7 indicates that ARMA(1,1) GARCH(2,1) model for NEPool can be used for forecasting returns, but only in a short-term horizon ahead. This conclusion stems from comparison of random variations of the predictive distribution of returns and the original return series.

In case of NORDPool spot market, 10 values were predicted from the sampled returns. Analogically, comparison of predictive distribution for portfolio returns and original returns indicates that a GARCH(2,1) model for NORDPool can also be used for forecasting the returns for a short-term horizon, as shown in Figure 8.

On the other hand, the fact that the true time series does not lie within the posterior distribution of GARCH forecasts means that there must be some essential feature in electricity spot price time series not captured by the GARCH paradigm, and by implication not by any ARMA model either.

Fig. 8: Predictive distribution of price returns for NEPool series.

In summary, even though some MCMC assumptions were violated, shapes of predictive distributions for model coefficients confirmed the initial prediction of their values. They also indicated that both estimated models may work reasonably in short-term forecasting.

7 Conclusions

We have identified ARMA and corresponding GARCH forecast models for two time series of electricity spot market prices, the Nordic NordPool and the U. S. NEPool. Models for both series are statistically optimal within a wide spectrum of ARMA and GARCH orders. Both the size of the data sets, and the behavior of the two time series are quite different, even if both series display prominent spikes.

GARCH models assume that a time series can be modeled by a linear model

with the sole assumption that its variance may depend on past variance history. We have tested the validity of this assumption by carrying out a Markov Chain Monte Carlo (MCMC) analysis on the parameters of such optimally identified GARCH models.

The results of the MCMC analysis indicate that although the models are able to forecast the future behavior of spot market prices with some skill, the models are not well identifiable. This is shown in the non-Gaussian structure of model parameter covariance, and also in the escape shown by the true spot price from the confidence envelope provide by MCMC sampling of model parameters.

Such results indicate that the behavior of electricity spot price is not captured by just adding the assumption of heteroschedasticity - there must be something deeper at play. In fact, other research groups have come to the same conclusion by different means, such as Bottazzi, Sapio and Secchi [9]. They study the Subbotin family of distributions and similarly identify that NordPool time series needs at least two different distributions to capture its dynamics.

Indeed, it appears as if the price time series would obey two different dynamics. The first of these is a relatively regular "elastic" behavior, when the market is efficient with supply and demand that balance each other. The second one occurs when some event pushes the market to a "seller's market" that allows spot prices to surge because non-elastic demand temporarily exceeds potential supply. Such a dual market nature would call for at least two different models to be used simultaneously. The reliability of such a dual model setup can, on the other hand, be analyzed using an appropriate modification of the MCMC paradigm, the so-called Reversible Jump MCMC (RJ-MCMC), as proposed by Laine et al [6].

References

[1] Aleksander Weron and Rafał Weron. *Power Exchange: Risk management strategies*. CIRE, Wrocław, Poland, 2000.

[2] Chris Brooks. *Introductory econometrics for finance*. Cambridge University Press, United Kingdom, 2002.

[3] Alan Pankratz. *Forecasting with Univariate Box-Jenkins Models: Concepts and Cases*. John Wiley and Sons, United States, 1983.

[4] Joon Y. Park. Nonstationary nonlinear heteroskedasticity. *Journal of Econometrics*, 110:383–415, October 2002.

[5] Antti Solonen. Monte carlo methods in parameter estimation of nonlinear models. Master's thesis, Lappeenranta University of Technology, Lappeenranta, Finland, January 2006.

[6] Marko Laine. *Adaptative MCMC Methods with Applications in Environmental and Geophysics Models*. PhD thesis, Lappeenranta University of Technology, Lappeenranta, Finland, 2008.

[7] Robert F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50:987–1007, 1982.

[8] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time series analysis: forecasting and control*. Prentice-Hall, Englewood Cliffs, 3 edition, 1994.

[9] Giulio Bottazzi, Sandro Sapio, and Angelo Secchi. Some statistical investigations on the nature and dynamics of electricity prices. *Physica A*, 355:54–61, 2005.

A New Interface for MPI in MATLAB and its Application over a Genetic Algorithm

A. Guillen[1] and I. Rojas[2] and G. Rubio[2] and H. Pomares[2] and L.J. Herrera[2] and J. González[2] *

1- University of Jaen - Dept of Informatics
Jaen - Spain

2- University of Granada - Dept of Computer Technology and Architecture
Granada - Spain

Abstract.
The work consists in the development of a new interface that allows MATLAB standalone applications to call MPI standard routines. The interface allows programmers and researchers to design parallel algorithms with the MATLAB application using all its advantages. The new interface is compared with other approaches showing smaller latency times in communications and an application to an algorithm to design RBFNN for function approximation is shown in the experimental results.

1 Introduction

MATLAB has binary files to be executed in all the most common platforms: UNIX, Linux, Mac. This program is used by a significant number of researchers and engineers to develop their applications and test models, however, when parallel programming is tackled, MATLAB does not provide a mechanism to exploit explicit parallelism although a recently developed ToolBox to parallelize certain parts of the code is available (*http://www.mathworks.com/products/distriben/*). More concretely, the Message Passing Interface standard, which is one of the most used libraries in parallel programming, is not supported. In [1] an interface to call MPI [2] functions was developed, however, it is only possible to use it when using Linux Operating System (OS) in a x86 architecture and for the concrete implementation LAM/MPI not considering others like Sun MPI, OpenMPI, etc.

This paper presents a new interface for MATLAB so its applications can invoke MPI functions following the standard and ensuring the possibility of being run in any platform where MATLAB has binaries to be executed on. The applications must be deployed using the MATLAB Compiler so no instances of MATLAB are required to be running at the same time, this is specially adequate for clusters. Thus, the rest of the paper is organized as follows: Section 2 will introduce briefly the MPI standard, then Section 3 will comment the MATLAB Compiler, in Section 4 the new interface will be exposed and in Section 5 a comparison between the new interface and a previous one will be shown as well

*This work has been partially supported by the Spanish CICYT Project TIN2007-60587 and the Junta Andalucia Project P07-TIC-02768.

as an example of a distributed genetic algorithm that was coded in MATLAB using the new interface.

2 Message Passing Interface: MPI

As it is defined in *http://www-unix.mcs.anl.gov/mpi/*, MPI is:

> a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementers, and users.

Among the advantages of, MPI that have made this library well known, are:

- The MPI standard is freely available.
- MPI was designed for high performance on both massively parallel machines and on workstation clusters.
- MPI is widely available, with both free available and vendor-supplied implementations.
- MPI was developed by a broadly based committee of vendors, implementers, and users.
- Information for implementers of MPI is available.
- Test Suites for MPI implementations are available.

The Message Passing Interface was designed in order to provide a programming library for inter-process communication in computer networks, which could be formed by heterogeneous computers. The library is available in many languages such us C, C++, Java, .NET, python, Ocaml, etc.

MPI is the most used library for inter-communication in High-performance computing (HPC) application. There are several vendors and public implementations availables OpenMPI [1], LAM-MPI [2] and MPICH [3], for instance.

3 MATLAB Compiler

MATLAB software has available a tool called *Compiler* which allows MATLAB to generate executable applications (stand-alones) that can be run independently of MATLAB, this is, there is no need of having MATLAB installed in the computer to run the application. The stand-alone requires a set of libraries which can be distributed after being generated with MATLAB, these libraries start the *Component Runtime* (MCR) that interprets the .m files as MATLAB would do.

[1] http://www.open-mpi.org/
[2] http://www.lam-mpi.org/
[3] http://www-unix.mcs.anl.gov/mpi/mpich1/

A *Component Technology File* (CTF) is generated during the compilation process. This file contains all the .m files that form the deployed application after being compressed and encrypted so there is no way to access the original source code. When the application is run for the first time, the content of this file is decompressed and a new directory is generated.

The process that MATLAB follows to generate a stand-alone application is made automatically and totally transparent to the user so he only has to specify the .m files that compose the application to be deployed and MATLAB will perform the following operations:

- Dependence analysis between the .m files
- Code generation: the C or C++ code interface is generated in this step.
- File creation: once the dependencies are solved, the .m files are encrypted and compressed in the CTF.
- Compilation: the source code of the interface files is compiled.
- Link: the object code is linked with the required MATLAB libraries.

This whole process is depicted in Figure 1.

As listed above, there is a code generation step where an interface for the MCR is created. These wrapper files allow a concrete architecture to run the compiled MATLAB code.

3.1 MPIMEX: A new MPI interface for MATLAB

In [1] the *Message Passing Interface ToolBox* (ToolBoxMPI) was presented. This toolbox has become quite popular, showing the increasing interest of the fusion between MATLAB and the emerging parallel applications. The main problem that this toolbox has is that it is implemented only for x86 machines running Linux and with the LAM/MPI implementation of MPI. As cited in the Introduction, there is a large variety of implementations of the MPI standard so there is the need of allowing MATLAB use MPI programming in other types of architectures and other MPI implementations. This is the main reason why the new interface proposed in this paper was developed.

MATLAB provides a method to run C/C++ and FORTRAN code within a .m file so the command interpreter can call another function as if it was another .m file. The file that has the .c source code must be written using a special library of functions called mex-functions generating what is know as mex-files [3]. Once the code is written using these special functions, it has to be compiled using the MATLAB *mex* compiler that generates an specific .mexXXX where XXX stands for the concrete architecture MATLAB is running on:

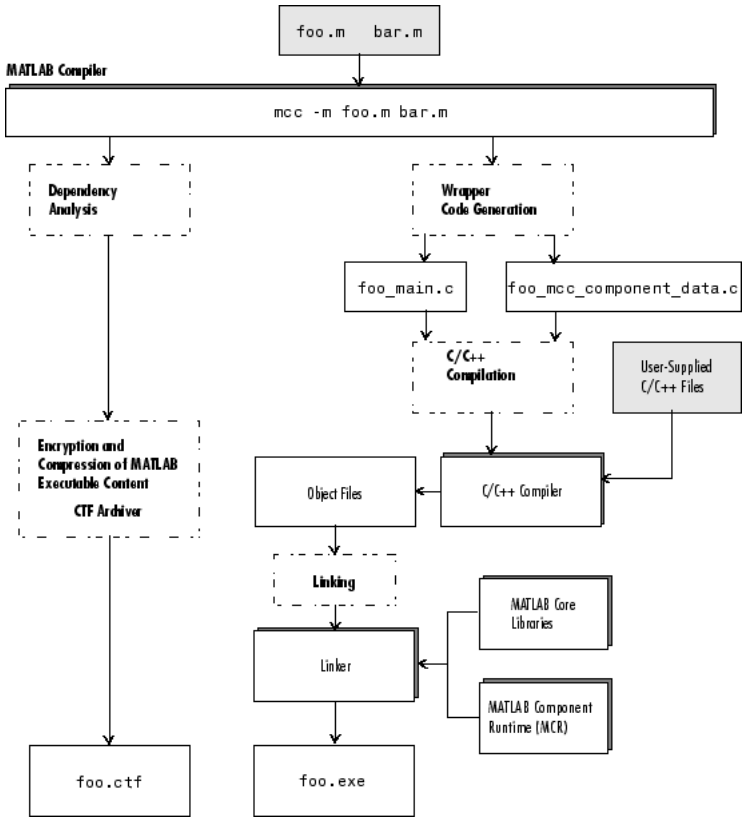
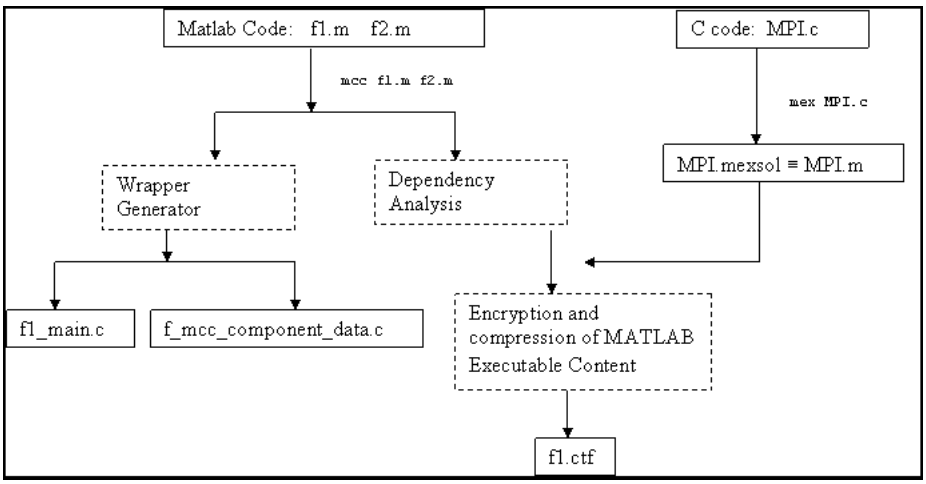


Fig. 1: Deployment process of an application using the MATLAB *Compiler*.

Fig. 2: Deploying application process for a MATLAB program calling the MPI routines



Platform	**MEX extension**
Linux (32-bits) / 64 bits	mexglx / mexa64
Macintosh (PPC)	mexmac
Macintosh (Intel)	mexmaci
32-bit Solaris SPARC / 64 bits	mexsol / mexs64
Windows (32-bits) / 64-bits	mexw32 / mexw64

The new interface developed uses this feature so it is possible to invoke the .c standard functions of MPI within the MATLAB source code so when the MATLAB Compiler is executed to deploy the application, those functions are treated as regular .m files. The result is that the deployed application can start the MPI environment and call all the routines defined by the standard. The process to generate a stand-alone application that uses MPI is shown in Figure 2.

3.1.1 Coding in MATLAB

The new interface keeps the sintaxis of the MPI standard in order to make easier the use of it by people that have already some experience coding with MPI in C. However, it is still easier to use than in C because it uses some of the advantages of MATLAB. For example, the initialization of the environment has been simplified comprising three functions of MPI such us MPI_Init, MPI_Comm_size and MPI_Comm_rank so all these parameters can be initialized with a single line of code, as an example will show below.

As the interface has been coded in a single file (MPI.mexXXX), a unique function has to be invoked from the MATLAB code. This function has a parameter that indicates the interface which MPI function will be called, so the

header of the MPI function is: MPI(MPI_function,...) where MPI_function is a string that has to include the exact name of the C functions excluding the prefix 'MPI_'. For example, to invoke the MPI_Send function which has the following header:

```
Name: MPI_Send - Performs a standard-mode blocking send.
C Syntax:
int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest,
             int tag, MPI_Comm comm)
Input Parameters:
             buf Initial address of send buffer (choice).
             count Number of elements send (nonnegative integer).
             datatype Datatype of each send buffer element (handle).
             dest Rank of destination (integer).
             tag Message tag (integer).
             comm Communicator (handle).
```

The corresponding MPIMEX call within the MATLAB code would be:

```
MPImex('Send',array, numel(array), 'MPI_DOUBLE', destination,
       tag, 'MPI_COMM_WORLD');
```

where all the parameters correspond to the ones defined in the MPI standard although, thanks to MATLAB, there is no need to worry about the type of data of the parameters array, destination and tag.

As commented before, the MPI_Init function has been coded in a slightly different way with the idea of simplifying the coder's task. When MPI('Init') is invoked, it returns the values of the parameters rank and size provided by the functions MPI_Comm_size and MPI_Comm_rank. Therefore, the line of code to initialize MPI in MATLAB using MPIMEX is:

```
[rank,size]=MPImex('Init');
```

so the output values are obtained in the same way as MATLAB standar functions. Although the MPI_Comm_size and MPI_Comm_rank are included in this called, they can be invoked separately using other communicators as MPI allows to define different communicators, assigning different ranks to the same process. Due to the lack of space, please visit https://atc.ugr.es/ aguillen/ for further details on how to use it or contact the authors by e-mail.

4 Experiments

This section shows a comparison between the new interface developed and an existing one. Afterwards, it also shows a real application where MPIMEX has been used.

4.1 Efficiency gain

The portability among the different platforms is not the only advantage over previous toolboxes for message passing in MATLAB but the new interface adds less overhead time when calling MPI routines. To show this efficiency gain, the new interface was compared to a previous one. The two classical message passing routines for the communication between two processes are MPI_Send and MPI_Recv whose header defined by the standard is (the MPI_Send has been described previously):

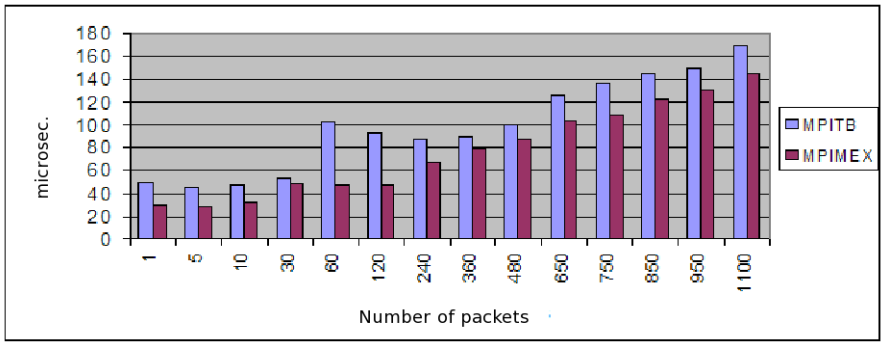


Fig. 3: Comparison between MATLAB MPI interfaces for MPI_Send

Name: MPI_Recv - Performs a standard-mode blocking receive.
C Syntax:
int MPI_Recv(void *buf, int count, MPI_Datatype datatype,int source, int tag,
MPI_Comm comm, MPI_Status *status)
Input Parameters:
count Maximum number of elements to receive (integer).
datatype Datatype of each receive buffer entry (handle).
source Rank of source (integer).
tag Message tag (integer).
comm Communicator (handle).
Output Parameters
buf Initial address of receive buffer (choice).
status Status object (status).
IERROR Fortran only: Error status (integer).

A simple program that performs a Send and Recv between two processes running in two processors was implemented. The program was executed 10000 times and on each run, the time elapsed during the MPI function calls was measured. Results are shown in Table 1 for MPI_Send and in Table 2 for MPI_Recv, these data have been graphically represented in Figures 3 y 4 respectively.

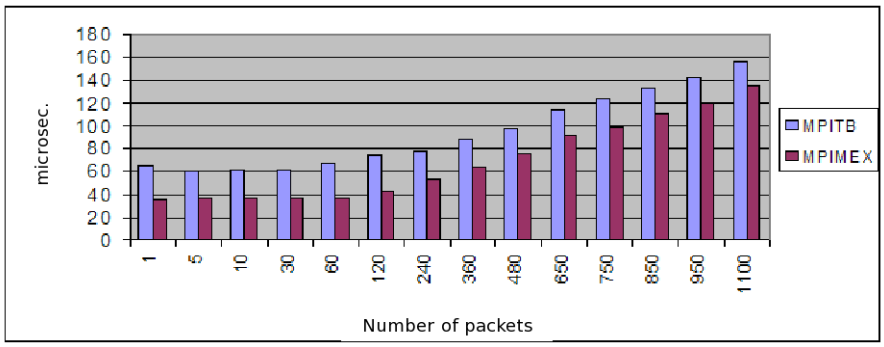


Fig. 4: Comparison between MATLAB MPI interfaces for MPI_Recv

Function MPI_Send		
# packets	ToolBoxMPI	MPIMEX
1	49.85 (168.7)	30.01 (8.2)
5	44.41 (9.8)	28.81 (8.1)
10	47.66 (9.5)	32.33 (10.76)
30	52.69 (7.8)	49.09 (552.1)
60	102.45 (859.7)	46.69 (402.6)
120	93.11 (577.5)	47.52 (281.1)
240	87.92 (409.6)	66.96 (563.1)
360	89.32 (210.3)	79.58 (575.3)
480	100.91 (255.7)	87.32 (417.1)
650	125.09 (461.4)	103.25 (434.2)
750	136.41 (480.1)	109.30 (431.4)
850	145.37 (482.0)	122.20 (444.6)
950	149.08 (326.3)	130.22 (447.8)
1100	169.11 (495.4)	145.61 (449.5)

Table 1: Mean of the time measures in μs. and standard deviation (in brackets) when calling the MPI_Send function using different number of elements.

Function MPI_Recv		
# packets	ToolBoxMPI	MPIMEX
1	65.63 (178.1)	35.91 (17.4)
5	59.92 (20.2)	36.63 (33.8)
10	62 (154.4)	36.28 (33.4)
30	60.74 (9.1)	37.31 (29.5)
60	66.72 (6.6)	36.64 (7.0)
120	73.93 (57.8)	42.56 (10.2)
240	77.61 (11.4)	52.75 (15.1)
360	88.01 (15.7)	63.98 (20.9)
480	97.71 (19.3)	76.12 (24.4)
650	113.97 (24.3)	92.52 (38.2)
750	124.24 (27.9)	99.35 (38.4)
850	133.23 (35.8)	110.89 (45.9)
950	142.53 (44.3)	120.05 (51.3)
1100	156.00 (50.6)	135.43 (65)

Table 2: Mean of the time measures in μs. and standard deviation (in brackets) when calling the MPI_Recv function using different number of elements.

As the results show, there is a larger overhead time when using the other MPI interface than when using the new one. This is the consequence of performing an unique call to a mexfile as explained in the subsection above. As the size of the packet increases, the overhead time becomes imperceptible, however, for fine grained applications where there exists many communications steps, this overhead time can become crucial for the application to be fasted.

4.2 Application over a distributed heterogeneous genetic algorithm

This section shows how this new interface becomes quite useful for parallel models development. The algorithm presented in [4] was implemented using the new interface so it was possible to be executed in a Sun Fire E15K. This machine can reach the number of 106 processors UltraSPARC III Cu 1.2 GHz with a memory of 1/2 TeraByte. The bandwith of the Sun Fire can reach 172.7 Gigabytes per second.

The algorithm consists in a distributed heterogeneous genetic algorithm that has the task of design Radial Basis Function Neural Networks (RBFNNs) to approximate functions [5]. The

parallelism that can be extracted from this application has two perspectives: data parallelism and functional parallelism. The functional parallelism makes reference to the one that can be obtained when distributing the different tasks in several processes so, as was demonstrated in [6, 7], this kind of parallelism increases the efficiency and improves the results. The data parallelism can be applied to genetic algorithms from two perspectives: the data could be the individuals or the input for the problem. In this case, the first one was considered so an initial population of 300 individuals was processed by a initial set of three specialized island. The algorithm was executed using a synthetic function to be approximated and the execution times are shown in Table 3 and in Figure 5. The speedup obtained thanks to the parallelism is represented in Figure 6.

	Execution time
3 Proc.	2620(117.21)
6 Proc.	809.5(5.29)
9 Proc.	504.3(21.59)
12 Proc.	346.3(20.55)
15 Proc.	281.6(25.73)
30 Proc.	165.2(21.07)

Table 3: Execution times in seconds and standard deviation (in brackets).

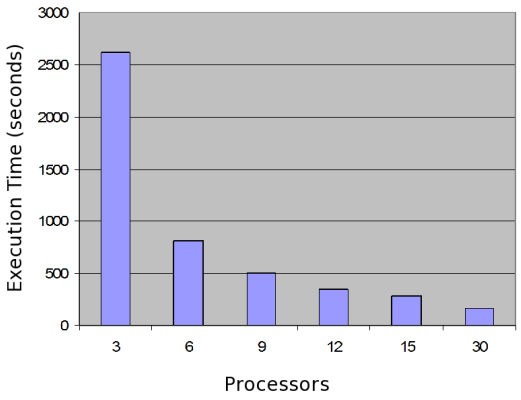


Fig. 5: Execution times in seconds.

5 Conclusions

This paper has presented a new interface that allows MATLAB users to take advantage of the message passing paradigm so they can design parallel applications using the MPI standard. The benefits of this new interface in comparison with previous ones is that it is possible to use it independently of the platform the application will be run on, the implementation of the MPI standard and it also has smaller overhead times. All these is translated in a better perfomance when building models such as RBFNN for time series prediction, regression or function approximations.

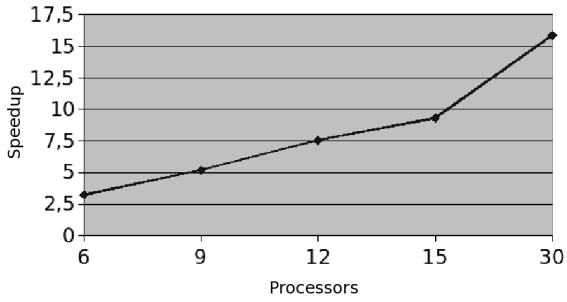


Fig. 6: Speedup obtained in the computation time when increasing the number of processors.

References

[1] J. Fernández, M. Anguita, E. Ros, and J.L. Bernier. SCE Toolboxes for the development of high-level parallel applications. *Lecture Notes in Computer Science*, 3992:518–525, 2006.

[2] http://www-unix.mcs.anl.gov/mpi/, 2005.

[3] http://www.mathworks.com/support/tech-notes/1600/1605.html#intro.

[4] A. Guillén, H. Pomares, J. González, I. Rojas, L.J. Herrera, and A. Prieto. Parallel Multi-objective Memetic RBFNNs Design and Feature Selection for Function Approximation Problems. *Lecture Notes in Artificial Intelligence*, 4507:341–349, 2007.

[5] J. Park and I. Sandberg. Approximation and Radial Basis Function Networks. *Neural Computation*, 5:305–316, 1993.

[6] A. Guillén, I. Rojas, J. Gonz ález, H. Pomares, L.J. Herrera, and B. Paechter. Improving the Performance of Multi-objective Genetic Algorithm for Function Approximation Through Parallel Islands Specialisation. *Lecture Notes in Artificial Intelligence*, 4304:1127–1132, 2006.

[7] A. Guillén, I. Rojas, J. Gonz ález, H. Pomares, L.J. Herrera, and B. Paechter. Boosting the performance of a multiobjective algorithm to design RBFNNs through parallelization. *Lecture Notes in Artificial Intelligence*, 2007.

Projection of time series with periodicity on a sphere

Victor Onclinx[1,2], Michel Verleysen[1] and Vincent Wertz [1,2] *

1- Université catholique de Louvain - Machine Learning Group
Place du Levant, 3, 1348 Louvain-la-Neuve - Belgium

2- Université catholique de Louvain - Department of Applied Mathematics
Avenue Georges Lemaître, 4, 1348 Louvain-la-Neuve - Belgium

Abstract. Predicting time series necessitates choosing adequate regressors. For this purpose, prior knowledge of the data is required. By projecting the series on a low-dimensional space, the visualization of the regressors helps to extract relevant information. However, when the series includes some periodicity, the structure of the time series is better projected on a sphere than on an Euclidean space. This paper shows how to project time series regressors on a sphere. A user-defined parameter is introduced in a pairwise distance criterion to control the trade-off between trustworthiness and continuity. Moreover, the theory of optimization on manifolds is used to minimize this criterion on a sphere.

1 Introduction

Time series forecasting is an important topic in many application domains. Conceptually, traditional methods [1, 2, 3] use the past values of a time series to predict future ones; these methods fit a linear or a nonlinear model between the vectors that gather the past values of the series, the regressors, and the values that have to be predicted. Note that exogenous variables and prediction errors may be used as inputs to the model too.

A first difficulty encountered by these methods is the choice of a suitable regressor size. Indeed, the regressors have to contain the useful information to allow a good prediction [4]. If the regressor size is too small, the information contained in the vector yields a poor prediction. Conversely, with oversized regressors, there can be redundancies such that the methods will overfit and predict the noise of the series.

For this reason and many other ones, including the choice of the model itself, it is useful to visualize the data (here the regressors) for a preliminary understanding before using them for prediction. This can be achieved by data projection methods [5, 6, 7, 8] which are aimed at representing high-dimensional data in a lower dimensional space. The projection of the regressors makes, for example, easier the visualization of some peculiarity in the time series.

*V. Onclinx is funded by a grant from the Belgium F.R.I.A. Part of this work presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its author(s). The authors thank Prof. Pierre-Antoine Absil for his suggestions on the theory of optimization on manifolds.

Moreover, assuming that data projection methods minimize the loss of information between the initial regressors and the projected ones, the forecasting of a time series can be achieved by using the projected regressors instead of the original ones, expecting that the smoothing resulting from the projection will help increasing the prediction performance.

In a first step, oversized regressors are projected to remove their potential redundancies and to reduce the noise. Most distance-based projection methods define the loss of information by the preservation of the pairwise distances. However, projection methods have to deal with a trade-off between trustworthiness and continuity [9], respectively the risk of flattening and tearing the projection. To control these types of behaviour, a user-defined parameter is introduced in the criterion [10] that implements the trade-off and that allows its control.

Furthermore, when time series have a periodic behaviour, it is difficult to embed them in an Euclidean space because of their complex structure [11]. Indeed, let us assume that the oversized regressors are lying close to an unknown manifold embedded in a high-dimensional space. Since the series is periodic, the manifold probably intercepts itself. In this context, the choice of a suitable projection manifold is motivated by its ability to keep the loops observed in the original space; the quality of the projection relies on its ability to preserve the global topology underlying the data distribution. The constraint of preserving loops is widely used in the context of topology-based projection methods, as the Self-organizing maps, where spheres [12, 13] and tori [14] are often used as projection manifolds; this paper presents a distance-based projection method on a sphere, a manifold that allows loops in the projection space.

The projection is achieved by the minimization of the pairwise distance criterion presented in Section 2. Since the projection space is non-Euclidean, Section 3 presents an adequate optimization procedure. Next to a brief introduction of the theory of optimization on manifolds [15], the theory is adapted to project data on a sphere. The projection of a sea temperature series on a sphere is presented in Section 4.1.

In order to take into consideration the advantages of the projection on manifolds, the forecasting methods should be adapted such that the prediction of time series can be based on the projected regressors. Section 4.2 is dedicated to the prediction of time series. By projecting the regressors on a sphere, a new *projected* time series is defined on the sphere; this series can easily be predicted using the Optimal-Pruned Learning Machine method [16]. Following these first results, the original time series is predicted with the projected regressors; the results of the forecasting are compared with the prediction of the series based on a 52-dimensional oversized regressors.

2 Projection criterion

This section aims at defining a projection criterion. As previously mentioned, data projection methods have to deal with a trade-off between trustworthiness and continuity. Two illustrative examples of the projection of a cylinder on $\mathbb{R}^2$

comment the trustworthiness and the continuity of a projection. Having in mind the compromise to reach these two objectives, a pairwise criterion can then be defined without restriction on the structure of the manifold.

Assuming that data close to a cylinder must be projected on the two-dimensional Euclidean space, a first option is to cut the cylinder along a generating line and to unfold it on the $\mathbb{R}^2$ Euclidean space. The resulting projection is trustworthy since two data that are close in the projected space ($\mathbb{R}^2$) are also close in the original space (the cylinder). However, because the cylinder has been torn, the projection cannot be continuous.

A second option is to flatten the cylinder to preserve the continuity. Actually, two data that are close in the original space, the cylinder, remain close in the projected one; the projection is thus continuous. Nevertheless, this projection is no more trustworthy since data coming from opposite part of the cylinder may be projected close from each other.

By counting the points that are close in one space but not in the other space, the trustworthiness and the continuity quality measures [9] are intuitively defined. Nevertheless, these measures are discrete and the optimization of these criteria is therefore difficult. To bypass this problem, distance-based projection methods minimize some weighted mean square errors between the original distance D_{ij} and the distance δ_{ij} on the projection manifold; the distances D_{ij} and δ_{ij} are defined between points i and j in their corresponding space with $1 \leq i, j \leq N$, N being the number of data.

The minimization of the unweighted cost function

$$f \equiv \sum_{i=1}^{N-1} \sum_{j>i}^{N} (D_{ij} - \delta_{ij})^2$$

cannot yield good results since large distances increase the cost function. In the projection context, this situation is against the intuition; one prefers to preserve the pairwise distances between close data rather than minimizing f.

By dividing each term of the cost function by the original distance D_{ij}, the minimization of the tearing error favours the continuity of the projection. Indeed, if it happens that two original data are close despite they are faraway in the projected space, they will dominate. Therefore, the minimization of the following cost function tends to make these data closer in the projected space:

$$\textit{Tearing error} \equiv \sum_{i=1}^{N-1} \sum_{j>i}^{N} \frac{(D_{ij} - \delta_{ij})^2}{D_{ij}}.$$

Conversely, by weighting each term with the corresponding distance δ_{ij} in the projected space, the trustworthiness of the projection is favoured:

$$\textit{Flattening error} \equiv \sum_{i=1}^{N-1} \sum_{j>i}^{N} \frac{(D_{ij} - \delta_{ij})^2}{\delta_{ij}}.$$

The flattening error expresses that points that are close in the projected space while they are not in the original space (small δ_{ij} and large D_{ij}) have to move faraway from each other during the optimization procedure.

Finally, to implement a trade-off between the trustworthiness and the continuity, a user-defined parameter $\lambda \in [0, 1]$ is introduced:

$$f \equiv \sum_{i=1}^{N-1} \sum_{j>i}^{N} \left(\lambda \frac{(D_{ij} - \delta_{ij})^2}{D_{ij}} + (1 - \lambda) \frac{(D_{ij} - \delta_{ij})^2}{\delta_{ij}} \right). \quad (1)$$

3 Optimization on manifolds

This section shows how to minimize the pairwise distance criterion (1). Because the projected points have to lie on a manifold, traditional optimization procedures cannot be used; the theory of optimization on manifolds proposes a powerful alternative. After an introduction to the topics from the theory of optimization on manifolds, adaptations to project data on a sphere are presented.

One could argue that to perform an optimization while keeping the projected points on a sphere, it is possible to perform a standard optimization in the spherical coordinate space. Unfortunately, this is not true since there are singularities in the two poles of the sphere. Actually, these two points are represented by two segments in the spherical coordinate space. Moreover, because the search space is limited to $\{(\phi, \theta) \in [0, 2\pi[\times[\frac{-\pi}{2}, \frac{\pi}{2}]\}$ and because it is not an Euclidean space anymore, traditional optimization methods cannot be used.

To circumvent these difficulties, the theory of optimization on manifolds proposes to consider the problem as an unconstrained minimization problem but by taking in mind that each point has to stay on the manifold all along the optimization procedure [15].

Working on a manifold does not allow movements through straight lines, as it is the case in the steepest descent gradient method; the curves of the manifold can however replace these straight directions since they include the curvature of the manifold and its global topology.

Searching for a minimum of a cost function f can be achieved by adapted line-search algorithm. Let us assume that the algorithm has successfully performed the k first iterations and that it has found the vector $\mathbf{y}(k) = (\mathbf{y}_1(k), ..., \mathbf{y}_N(k))$ where $\mathbf{y}_i(k)$ is the location of data i on the projection manifold after iteration k. Moreover, let us denote the vector $\nu(k)$ that gathers the parameters of the manifold; since the optimal projection manifold cannot be determined *a priori*, this vector has to be optimized too. For example, in the case of the sphere, $\nu(k)$ will denote the radius of the sphere (which is unknown *a priori*).

First the gradient $-\nabla f(\mathbf{y}_1(k), ..., \mathbf{y}_N(k), \nu(k))$ is evaluated. Nevertheless, this direction may point faraway from the manifold. To take into consideration the manifold constraint and its curvature, the gradient $-\nabla f$ is projected on the tangent space $\mathcal{T}_{\mathbf{y}}\mathcal{M}$. In this way, the new direction $-\nabla' f(\mathbf{y}_1(k), ..., \mathbf{y}_N(k), \nu(k))$ is tangent to some curve $\gamma : \mathbb{R} \mapsto \mathcal{M} : t \mapsto \gamma(t)$ and therefore close to the manifold.

By searching in this direction with a step size α, a new location $\mathbf{y}'(k)$ can be found on the tangent space $\mathcal{T}_{\mathbf{y}}\mathcal{M}$. However, this location is not *on* the manifold; it has then to be retracted on the latter. The retraction, which is a kind of deterministic projection from the tangent space to the manifold, has to be chosen such that the new candidate location $\mathbf{y}(k+1)$ belongs to the curve γ determined by the direction $-\nabla' f$. The step size α is chosen under the Armijo condition [15] that ensures a sufficient decrease of the cost function. This means that the decrease of the cost function must be larger than the expected decrease of the first order approximation of the cost function f with a smaller step size $\sigma\alpha$ where $\sigma \in [0, 1]$. In other words, if the Armijo condition

$$f(\mathbf{y}(k)) - f(\mathbf{y}(k+1)) \quad \geq \quad \sigma\alpha||\nabla' f||^2 \tag{2}$$

is satisfied, the cost function has sufficiently decreased.

For details of the propose line-search algorithm see [15]. Fig. 1 shows the different steps of a single iteration.

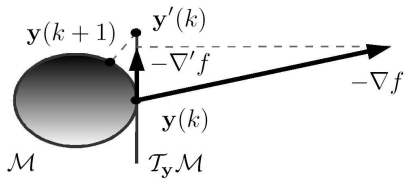


Fig. 1: Optimization iteration

After this brief introduction to the theory of optimization on manifolds, the latter is adapted to the problem of minimizing criterion (1) on a sphere. First, one has to define the manifold $\mathcal{M}$ and the tangent space $\mathcal{T}_{\mathbf{y}}\mathcal{M}$. In addition to the spherical form of the manifold, one has also to add its radius R. The value of the radius is a scaling factor; this means that the radius R is considered as a parameter of the manifold because the adequate sphere is not known *a priori*. As each vector on the sphere has to have the same norm, the definition of the manifold can be expressed by:

$$\mathcal{M} \quad \equiv \quad \{(\mathbf{y}_1, ..., \mathbf{y}_N, R) \in S_R^3 \times ... \times S_R^3 \times \mathbb{R}^+ | \mathbf{y}_i^T \mathbf{y}_i - R^2 = 0, 1 \leq i \leq N\}.$$

By differentiating the set of constraints, the tangent space $\mathcal{T}_{\mathbf{y}}\mathcal{M}$ is defined by:

$$\mathcal{T}_{\mathbf{y}}\mathcal{M} \quad \equiv \quad \{(\mathbf{u}_1, ..., \mathbf{u}_N, u_R) \in \mathbb{R}^3 \times ... \times \mathbb{R}^3 \times \mathbb{R} | \mathbf{y}_i^T \mathbf{u}_i - R u_R = 0, 1 \leq i \leq N\}.$$

Finally, if the angle between the vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ is known, the product between the radius and this angle defines the distance between $\mathbf{y}_i$ and $\mathbf{y}_j$. In order to evaluate this angle, the geodesic distance between $\mathbf{y}_i$ and $\mathbf{y}_j$ on the sphere is defined by the expression $\delta_{ij} \equiv R \arccos \frac{\mathbf{y}_i' \mathbf{y}_j}{\|\mathbf{y}_i\| \|\mathbf{y}_j\|}$. Concerning the distance in the high-dimensional space, the geodesic distance is approximated

by the construction of a graph through the data where the edges are weighted by the Euclidean distances. The distance D_{ij} is evaluated by a shortest path algorithm [17, 18] such as Dijkstra's one. At the end, the evaluation of the gradient $-\nabla f$ is defined by the partial derivatives with respect to the locations $\mathbf{y}_i$ and the radius R.

4 Experiments

In this section, the data projection method is illustrated on the ESTSP2007 competition dataset of the weekly evolution of the sea temperature. The series is represented in Fig. 2 where the colour varies with the temperature. The series contains 875 temperature measures; a yearly periodicity can easily be observed.

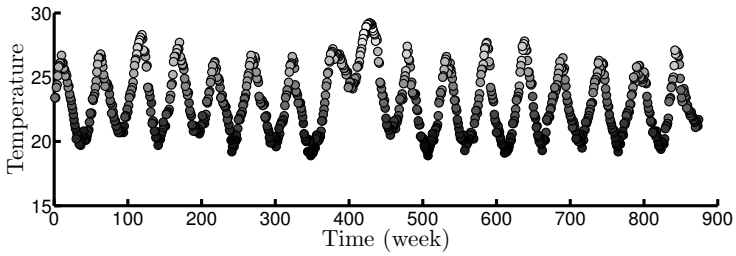


Fig. 2: Weekly evolution of the sea temperature

The methodology to forecast a periodic time series, as proposed in this paper, begins by building oversized regressors. The size of the regressors is chosen experimentally with respect to the length of a single period: 52-dimensional oversized regressors are built. Even if they probably contain all useful information for the prediction, these regressors are noisy and they certainly contain redundancies. The regressors are thus projected on a sphere according to the above methodology. The forecasting of the time series is, at the end, based on the projected regressors.

Section 4.1 shows the results of the projection; hence, the projected regressors define a curve on the optimal sphere. Section 4.2 first studies the forecasting of this new time series on the sphere to show the accuracy of the projection and of the methodology. Finally, the prediction of the original time series is performed and evaluated. Both the prediction of the projected time series on the sphere, and the prediction of the original time series based on the projected regressors, use the OPELM method [16].

4.1 Projection of the sea temperature series

The intrinsic dimension of the 52-dimensional oversized regressors is much lower than the embedding Euclidean space. For example, by projecting the data with Principal Component Analysis [19] in order to reduce the dimensionality to

the 10 principal components, the residual variance is less than 1 percent; this motivates the idea of projecting the regressors on a low-dimensional manifold.

The geodesic distance in the high-dimensional space D_{ij} is approximated by the shortest path in the graph built through the 50 closest neighbours [17, 18].

Fig. 3: 52-regressors projected on the sphere with $\lambda = 0.9$

The result of the projection on the sphere is shown in Fig. 3 where the colour varies smoothly with respect to the value of $\mathbf{y}(t)$. The colours used are the same as in Fig. 2; it can be easily seen that similar values of the original time series, thus similar colours, are close on the sphere. The additional curve in Fig. 3 joins points that are consecutive in time to illustrate the path of the projected time series on the manifold. The projected time series turns around the sphere such that the sphere keeps the periodicity of the time series. Furthermore, the isolated part of the projected data in the upper left region of the sphere in Fig. 3 corresponds to the irregularities of the time series observed between times $t = 380$ and $t = 420$ in Fig. 2.

In Fig. 4, the corresponding result in the spherical coordinate space is represented in order to visualize all the data; the *glyph* in the center of the figure corresponds to the above-mentioned irregularities. According to both Fig. 3 and 4, the projection of the times series makes it possible to isolate its irregularities in a visual way.

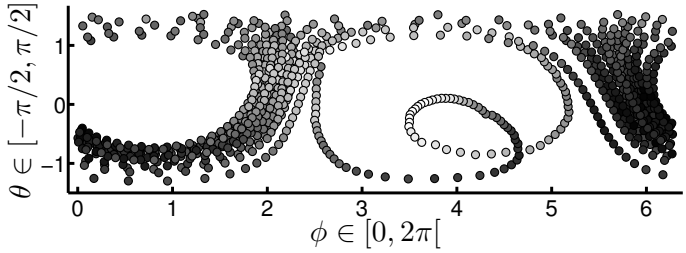


Fig. 4: 52-regressors projected on the sphere, in the spherical coordinate space

4.2 Prediction of the sea temperature series using the projected regressors

Besides the visualization applications, the projection of the time series defines new regressors where redundancies are removed and noise is probably reduced. This subsection shows how the projected regressors can be used.

Let us consider the projected time series defined by the locations $\mathbf{y}(t)$ on the sphere, with t between 1 and N. To test the quality of the projected time series, a model $\hat{\mathbf{y}}(t+1) = f(\mathbf{y}(t), \mathbf{y}(t-1), \theta)$ is built with the Optimal-Pruned Learning Machine method [16]. OPELM is a two-layer regression model, where the first layer is chosen randomly among a set of possible activation functions and kernels, and the second layer is optimized with linear tools. The speed of optimizing such models makes it possible to test a large number of them, among which the best according to some validation criterion is selected. θ represents the parameters of the method, more specifically the number and the types of kernels or functions; both Gaussian and sigmoidal functions are used. The learning and validation errors are estimated according to the following definitions:

$$\begin{aligned} \textit{Learning error} &\equiv \frac{\sum_{t=1}^{N_1} ||\hat{\mathbf{y}}(t) - \mathbf{y}(t)||^2}{N_1} \\ \textit{Validation error} &\equiv \frac{\sum_{t=1}^{N_2} ||\hat{\mathbf{y}}(t) - \mathbf{y}(t)||^2}{N_2}, \end{aligned}$$

where N_1 and N_2 represent respectively the size of the learning and of the validation sets. The learning set is randomly built with 66 percent of the initial set; 10000 simulations are performed in order to estimate the learning and the validation errors as average over all the 5000 experiments. The results are shown in Fig. 5 with respect to the number of kernels/functions used in the OPELM tool.

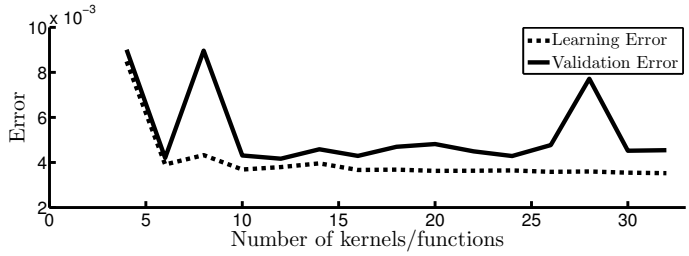


Fig. 5: Learning and validation errors of the normalized projected time series versus the number of kernels/functions used

Fig. 5 shows that the projected time series on the sphere can easily be predicted. However, this result does not mean that the original series can be easily predicted too. As a first attempt in this direction, we propose to build another prediction model based on the projected regressors. Assuming that the locations

$\mathbf{y}(t)$ on the sphere are known, they define reduced regressors such that it can be used to forecast the original time series $x(t)$. In [20], the authors define new regressors by concatenating the projected regressors with the corresponding value $x(t)$. Here, we use an alternative idea, which consists in predicting the variations in the time series using the projected regressors. The model is thus defined by:

$$x(t+1) \quad = \quad x(t) + \tilde{f}(\mathbf{y}(t), \theta). \tag{3}$$

The quality of the prediction is close to the forecasting with the 52-dimensional regressors as shown in Fig. 6. In this figure the learning error of the prediction based on the projected regressors is higher than the learning error based on the 52-dimensional initial regressors, but the validation error is lower when using the projection. This is likely to be due to overfitting of the model based on the 52-dimensional regressors.

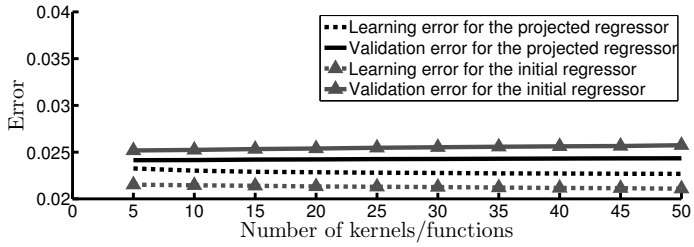


Fig. 6: Learning and validation errors for the prediction of the normalized time series with the initial regressors and the projection on the sphere

5 Conclusion

This paper presents a nonlinear method aimed at projecting the regressors of a time series on a sphere such that redundancies are removed and noise is reduced. The method minimizes a pairwise distance cost function where the trade-off between trustworthiness and continuity is controlled by a user-defined parameter. The projection on a sphere is aimed at embedding the periodicity of time series using a dedicated optimization method. The quality of the projection is assessed through the trustworthiness and the continuity quality measures and is compared to the same measures obtained after projecting on Euclidean spaces.

The projected regressors can be used to forecast the original time series. First results are shown using the OPELM algorithm. Nevertheless, the OPELM prediction method is not specifically adapted to spherical data for which the manifold contains another part of useful information. This will be studied in future work.

References

[1] G.E.P. Box and G. Jenkins. *Time Series Analysis : Forecasting and Control*. Holden-Day, Incorporated, 1990.

[2] L. Ljung. *System Identification, Theory for the user*. Prentice Hall Information and System Sciences Series, 1987.

[3] C. Chatfield and A.S. Weigend. Time series prediction: Forecasting the future and understanding the past. *International Journal of Forecasting*, 10(1):161–163, June 1994.

[4] F. Takens. On the numerical determination of the dimension of an attractor. In *Dynamical Systems and Bifurcations*. Groningen, 1984.

[5] J.A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer Science+Business Media, LLC, 2007.

[6] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.

[7] A. Brun, C.-F. Westin, M. Herberthson, and H. Knutsson. Fast manifold learning based on riemannian normal coordinates. In Heikki Kälviäinen, Jussi Parkkinen, and Arto Kaarna, editors, *SCIA*, volume 3540 of *Lecture Notes in Computer Science*, pages 920–929. Springer, 2005.

[8] J.A. Lee and M. Verleysen. Nonlinear projection with the isotop method. In J. R. Dorronsoro ed., editor, *Artificial Neural Networks*, Lecture Notes in Computer Science 2415, pages 933–938, London, UK, Augustus 2002. ICANN, Springer-Verlag.

[9] J. Venna and S. Kaski. Neighborhood preservation in nonlinear projection methods: An experimental study. In *ICANN '01: Proceedings of the International Conference on Artificial Neural Networks*, pages 485–491, London, UK, August 21-25 2001. Springer-Verlag.

[10] J. Venna and S. Kaski. Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity. In *Proceedings of WSOM'05, 5th workshop on self-organizing maps*, pages 695–702. WSOM, September 5-8 2005.

[11] V. Onclinx, V. Wertz, and M. Verleysen. Nonlinear data projection on a sphere with a controlled trade-off between trustworthiness and continuity. In *ESANN 2008, European Symposium on Artificial Neural Networks*, pages 43–48, Bruges (Belgium), April 23-25 2008. ESANN, d-side publi.

[12] H. Ritter. Self-organizing maps on non-euclidean spaces. In S. Oja and E. Kaski, editors, *Kohonen Maps*, pages 97–108. Elsevier, Amsterdam, 1999.

[13] H. Nishio, Md. Altaf-Ul-Amin, K. Kurokawa, K. Minato, and S. Kanaya. Spherical som with arbitrary number of neurons and measure of suitability. In *Proceedings of WSOM'05, 5th workshop on self-organizing maps*, pages 323–330, September 5-8 2005.

[14] J.X. Li. Visualization of high-dimensional data with relational perspective map. *Information Visualization*, 3(1):49–59, 2004.

[15] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, January 2008.

[16] Y. Miche, P. Bas, C. Jutten, O. Simula, and A. Lendasse. A methodology for building regression models using extreme learning machine: Op-elm. In *European Symposium on Artificial Neural Networks (ESANN)*. d-side publi., April 23-25 2008.

[17] J.A. Lee, A. Lendasse, and M. Verleysen. Curvilinear distance analysis versus isomap. In *ESANN 2002, European Symposium on Artificial Neural Networks*, pages 185–192, Bruges (Belgium), April 22-24 2002. ESANN, d-side publi.

[18] J.A. Lee, A. Lendasse, N. Donckers, and M. Verleysen. A robust nonlinear projection method. In *ESANN 2000, European Symposium on Artificial Neural Networks*, pages 13–20, Bruges (Belgium), April 28-28 2000. ESANN, D-Facto public.

[19] K. Pearson. Analysis of a complex statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[20] A. Lendasse, J. Lee, V. Wertz, and M. Verleysen. Forecasting electricity consumption using nonlinear projection and self-organizing maps. *Neurocomputing*, 48, 2002.

A variable selection approach based on the Delta Test for Extreme Learning Machine models

Fernando Mateo[1] and Amaury Lendasse[2]

1- Universidad Politécnica de Valencia - Dept. Ingeniería Electrónica
Camino de Vera s/n, 46022 Valencia - Spain

2- Helsinki University of Technology - Adaptive Informatics Research Centre
Konemiehentie 2, 02150 Espoo - Finland

Abstract. Extreme Learning Machine, ELM, is a newly available learning algorithm for single layer feedforward neural networks (SLFNs), and it has proved to show the best compromise between learning speed and accuracy of the estimations. In this paper, a methodology based on Optimal-Pruned ELM (OP-ELM) for function approximation enhanced with variable selection using the Delta Test is introduced. The least angle regression (LARS) algorithm is used after variable selection to rank the input variables, and scaling is also introduced as a way to estimate the influence of each input in the output value. The performance is assessed on a dataset related to anthropometric measurements for children weight prediction. The accurate results show that this combination of techniques is very promising to solve real world problems and represents a good alternative to classic backpropagation methods.

1 Introduction

In many real-life problems it is convenient to reduce the number of involved features (variables) in order to reduce the complexity, especially when the number of features is large compared to the number of observations. There are several criteria to tackle this variable reduction problem. Three of the most common are: maximization of the mutual information (MI) between the inputs and the outputs, minimization of the k-nearest neighbors (k-NN) leave-one-out generalization error estimate and minimization of a nonparametric noise estimator (NNE).

Extreme Learning Machine (ELM)[1] is a new learning technique to train single layer feedforward neural networks (SLFN) which chooses the input weights randomly and determines the output weights analytically. This algorithm is designed to build models that provide the best possible generalization in the shortest time. Given its success, it has already been applied to several fields of machine learning such as text classification [2] or time series prediction [3].

This work intends to make use of the methodology described in [4] which proposes a combination of Extreme Learning Machine with optimal pruning (OP-ELM) and variable selection. In this case, we focus on the use of a NNE as a selection criterion, concretely by using the Delta Test (DT) as estimator. The applicability of the method is assessed on a dataset of children anthropometric measurements.

This paper is structured as follows: Section 2 explains the variable selection methodology using the Delta Test as a criterion, and how it is integrated in the forward-backward search (FBS) algorithm. In Section 3 there is a description of the LARS methodology for input ranking and Section 4 gives a mathematical perspective on the Extreme Learning Machine method. In Section 5, the experiments are described and some relevant results are presented, while Section 6 summarizes the conclusions of this work.

2 Variable selection

2.1 The Delta Test

The Delta Test, firstly introduced by Pi and Peterson for time series [5], is a technique to estimate the variance of the noise, or the mean squared error (MSE), that can be achieved without overfitting. Given N input-output pairs $(x_i, y_i) \in \mathbb{R}^M \times \mathbb{R}$, the relationship between x_i and y_i can be expressed as

$$y_i = f(x_i) + r_i, \qquad i = 1, ..., N$$

where f is the unknown function and r is the noise. The DT estimates the variance of the noise r.

The DT is useful for evaluating the nonlinear correlation between two random variables, namely, input and output pairs. The DT can be also applied to input selection: the set of inputs that minimizes the DT is the one that is selected. Indeed, according to the DT, the selected set of inputs is the one that represents the relationship between inputs and output in the most deterministic way. DT is based on hypotheses coming from the continuity of the regression function. If two points x and x' are close in the input space, the continuity of regression function implies the outputs $f(x)$ and $f(x')$ will be close enough in the output space. Alternatively, if the corresponding output values are not close in the output space, this is due to the influence of the noise.

The DT can be interpreted as a particularization of the Gamma Test [6] considering only the first nearest neighbor. Let us denote the first nearest neighbor of a point x_i in the $\mathbb{R}^M$ space as $x_{NN(i)}$. The nearest neighbor formulation of the DT estimates $\mathrm{Var}[r]$ by

$$\mathrm{Var}[r] \approx \delta = \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_{NN(i)})^2, \text{ with } \mathrm{Var}[\delta] \to 0 \text{ for } N \to \infty$$

where $y_{NN(i)}$ is the output of $x_{NN(i)}$.

2.2 Forward-backward search methodology

To overcome the difficulties and the high computational time that an exhaustive search would entail (i.e. $2^N - 1$ input combinations, being N the number of variables), there are several other search strategies. These strategies are suboptimal because they do not test every input combination, and they are clearly affected

by local minima, but they are preferred to an exhaustive search if the number of variables is large.

Among the typical search strategies, there are three that share similarities:

- Forward search
- Backward search (or pruning)
- Forward-backward search (or forward stagewise regression)

The difference between the first two is that the forward search starts from an empty set of selected variables and adds variables to it according to the optimization of a search criterion, while the backward search starts from a set containing all the variables and removes those for which the elimination optimizes the search criterion.

Both forward and backward search suffer from incomplete search. The forward-backward search (FBS) is a combination of them. It is more flexible in the sense that a variable is able to return to the selected set once it has left it, and vice versa, a previously selected variable can be discarded later. This method can start from any initial input set: empty set, full set, custom set or randomly initialized set. If we consider a set of N input-output pairs $(x_i, y_i) \in \mathbb{R}^M \times \mathbb{R}$, the forward-backward search algorithm can be described as follows

1. Initialization:

   Let S be the selected input set, which can contain any input variables, and F the unselected input set, which contains the variables not present in S. Compute Var[r] using Delta Test (see section 2.1) on the set S.

2. Forward-Backward selection step:

   Find

   $$x^S = \arg\, min_{x_i,x_j}\{(\text{Var}[r]|S \cup x^j) \cup (\text{Var}[r]|S \setminus x^i)\}, \;\; x^i \in S, x^j \in F$$

3. If the old value of Var[r] on the set S is lower than the new result, stop; otherwise, update set S and save the new Var[r]. Repeat step 2 until S is equal to any former selected S.

4. The selected input set is S

3 The LARS algorithm

Least angle regression (LARS)[7] is a stylized version of the stagewise procedure that uses a simple mathematical formula to accelerate the computations. Only m steps are required for the full set of solutions, where m is the number of covariates.

The LARS procedure works roughly as follows. Supposing that the initial estimate is μ_0, the algorithm takes a first step in the direction of the most correlated predictor, say x_1. When another predictor, say x_2, becomes sufficiently correlated to become one of the chosen variables, the next step is taken in the bisecting angle between x_1 and x_2. This happens again when a third predictor, x_3, gains the sufficient importance to contribute to the model. The process continues until all the covariates have entered the model. This method is illustrated in Fig. 1

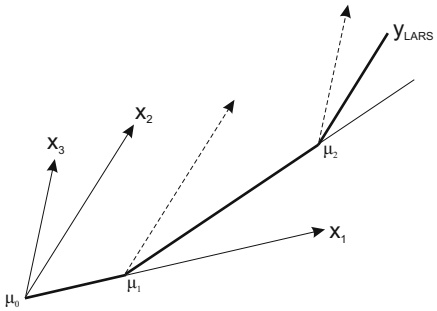


Fig. 1: LARS algorithm evolution for m = 3 covariates.

The entire sequence of steps in the LARS algorithm with $m < n$ variables, where n is the number of observations, requires $\mathrm{O}(m^3 + nm^2)$ computations (the cost of a least squares fit on m variables). The LARS algorithm works gracefully for the case where there are many more variables than observations ($m >> n$).

It is important to take into account that the procedure requires that the variables and outputs are previously scaled to have zero mean and variance equal to 1.

4 Extreme Learning Machine

Let us consider a set of N points $(\boldsymbol{x}_i, \boldsymbol{t}_i) \in \mathbb{R}^n \times \mathbb{R}^m$, where $i = 1, ..., N$. A standard single layer feedforward neural network (SLFN) with L hidden neurons and activation function $g(x)$ can be mathematically modeled as

$$\sum_{i=1}^{L} \beta_i g(\boldsymbol{w}_i \boldsymbol{x}_j + b_i) = \boldsymbol{d}_j \qquad j = 1, ..., N$$

where $\boldsymbol{w}_i$ is the weight vector connecting inputs and the ith hidden neuron, β_i is the weight vector connecting the ith hidden neuron and output neurons, b_i is the threshold of the ith hidden neuron, and $\boldsymbol{d}_j$ is the output given by the ELM for data point j. The standard SLFN with n hidden neurons and activation function $g(x)$ can approximate these N samples with zero error in the ideal case, meaning that $\sum_{j=1}^{L} \|\boldsymbol{d}_j - \boldsymbol{t}_j\| = 0$, and thus, there exist β_i, $\boldsymbol{w}_i$ and b_i such that

$$\sum_{i=1}^{L} \beta_i g(\boldsymbol{w}_i \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j \qquad j = 1, ..., N$$

The above equations can be written compactly as:

$$\mathbf{H}\beta = \mathbf{T}$$

where

$$\mathbf{H} = \begin{pmatrix} g(\boldsymbol{w}_1 \boldsymbol{x}_1 + b_1) & \dots & g(\boldsymbol{w}_L \boldsymbol{x}_1 + b_L) \\ \vdots & \dots & \vdots \\ g(\boldsymbol{w}_1 \boldsymbol{x}_N + b_1) & \dots & g(\boldsymbol{w}_L \boldsymbol{x}_N + b_L) \end{pmatrix}_{N \times L}$$

$$\beta = \begin{pmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{pmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{pmatrix}_{N \times m}$$

The matrix $\mathbf{H}$ is called the hidden layer output matrix of the neural network. When the number of neurons in the hidden layer is equal to the number of samples, $\mathbf{H}$ is square and invertible. Otherwise, the system of equations needs to be solved by numerical methods, concretely by solving

$$\min_\beta \|\mathbf{H}\beta - \mathbf{T}\|$$

The solution that minimizes the norm of this least squares equation is

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$$

where $\mathbf{H}^\dagger$ is called Moore-Penrose generalized inverse [1]. The most important properties of this solution are:

- Minimum training error.
- Smallest norm of weights and best generalization performance.
- The minimum norm least-square solution of $\mathbf{H}\beta = \mathbf{T}$ is unique, and is $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$.

Hence, the ELM algorithm for SLFNs can be summarized in these steps:

Given a training set $(\boldsymbol{x}_i, \boldsymbol{t}_i) \in \mathbb{R}^n \times \mathbb{R}^m$, $i = 1, ..., N$ activation function $g(x)$ and L hidden neurons:

1. Assign arbitrary input weights w_i and bias b_i, $i = 1, ..., L$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weights β:

   $$\beta = \mathbf{H}^\dagger \mathbf{T}$$

   where $\mathbf{H}$, β and $\mathbf{T}$ are as defined before.

5 Experiments and results

5.1 The "anthrokids" dataset

The dataset used for testing the described methodology was a collection of anthropometric data that represents the results of a three-year study on 3900 infants and children representative of the U.S. population of year 1977, ranging in age from newborn to 12 years of age. The dataset comprises 121 variables and the target variable to predict is children's weight. The data repository can be found in http://ovrt.nist.gov/projects/anthrokids/.

This dataset is characterized by the presence of many missing values. Therefore, a first sample and variable discrimination had to be done to build a robust and reliable dataset. The approach to do this was to keep a minimum amount of 1000 samples out of the possible 3900. The number of variables was chosen by means of an iterative routine which attacked the data set reduction both in terms of number of samples and number of variables. The variables were removed one by one (every time the one with the highest amount of missing values) while the number of samples removed per iteration could be tuned. The best compromise, with 43 samples removed per iteration, was a set of 54 variables which led to a set of 1019 samples, free of missing values. Figures 2(a) and 2(b) describe this process. One extra variable was removed because it had a constant value for all samples, yielding a final set of 53 variables.

The resulting dataset was subdivided into training and test sets, with 70% and 30% of the samples respectively. The variables were normalized to have zero mean and standard deviation 1 before being processed.

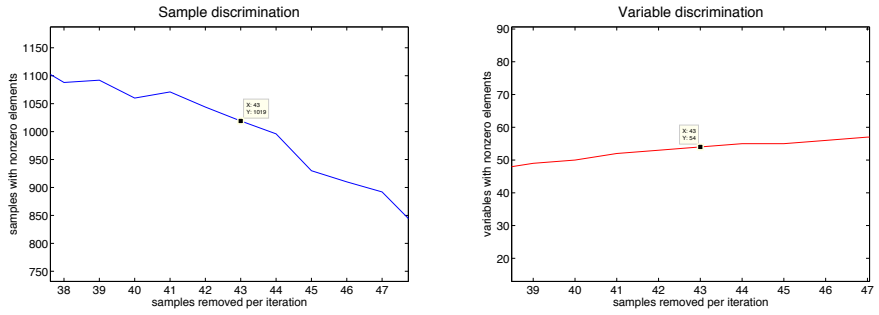
5.2 Forward-backward search

Forward-backward search with Delta Test as the performance criterion and empty initial search set was applied to the training set. The FBS algorithm was applied to several training set combinations to find the best (lowest) DT value. The method selected the 12 variables listed in Table 1 and the value of Var$[r]$ versus the number of variables is shown in Fig. 3. The lowest DT value achieved was 0.0070 and the algorithm converged in 281.63 seconds.

5.3 OP-ELM model construction

After this initial selection, an OP-ELM model was built using these 12 variables and the same percentages of training and test samples. The algorithm was initialized with a high number of kernels (100) so that the pruning did not affect the accuracy of the result.

The criterion to optimize during training was the estimation of the leave-one-out (LOO) validation error. Usually, LOO estimation is too time consuming, especially when the number of samples is large. For that reason, the estimation was done using PRESS (PREdiction Sum of Squares) statistics, which gives an exact formula for LOO calculation when the problem is linear. This exact formula defines the LOO error ϵ as



(a) (b)

Fig. 2: Composition of the dataset. The first step was the determination of the number of samples to remove per iteration to achieve the minimum number of samples (>1000) with nonzero elements (a) and the second was to find the number of variables with nonzero elements determined by this amount of samples removed between iterations (b).

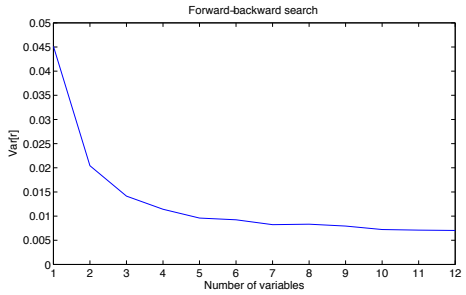


Fig. 3: Forward-backward search algorithm evolution. The algorithm reaches the minimum value of Var[r] for 12 variables.

$$\epsilon = \frac{\boldsymbol{t}_i - \mathbf{h}_i}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T}$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T\mathbf{H})^{-1}$, $\mathbf{H}$ is the hidden layer output matrix, $\mathbf{h}_i$ are the column vectors of matrix $\mathbf{H}$, $\boldsymbol{t}_i$ are the target values and β are the output weights (see section 4 for details). Table 2 lists the results of several OP-ELM models, using different activation functions for the hidden layer. A linear component is always maintained because it generally helps fitting the data if there is any linearity between the inputs and the outputs of the model.

The results show that variable selection allows building a model with the same performance as with the full set of variables (or even improving it when linear and sigmoid activation functions are used) in a much shorter time. The reduction in the number of required neurons is also noticeable.

Variable number	Variable name	LARS ranking
1	Stature	12
8	Erect sitting height	7
13	Buttock-knee length	10
34	Shoulder breadth	4
38	Upper arm circumference	5
40	Elbow-hand length	8
62	Chest circumference	1
65	Waist circumference	6
70	Hip circumference	3
76	Upper thigh circumference	11
80	Calf circumference	2
85	Foot length	9

Table 1: Variables selected by FBS (DT = 0.0070) and ranking given by LARS algorithm. The variable numbers correspond to their position in the original dataset

Number of variables	Activation function	Comput. time (s)	LOO error	Test error	Number of neurons
53 (full set)	L + S	7.79	0.0221	0.0354	88
	L + G	8.12	0.0060	0.0167	118
	L + S + G	7.98	0.0069	0.0169	143
12 (FBS)	L + S	3.15	0.0110	0.0214	47
	L + G	2.71	0.0060	0.0170	17
	L + S + G	3.24	0.0062	0.0173	22

Table 2: Performance achieved by several OP-ELM models using different combinations of activation functions. L: linear, S: sigmoid, G: gaussian.

5.4 LARS ranking

The LARS algorithm is guaranteed to find the best ranking among all possible inputs if a problem is linear. The OP-ELM model makes use of LARS to rank the neurons of the hidden layer, since this layer can be considered an input layer to the last (linear) stage of the neural network. We also used the LARS algorithm to provide the best ranking of the variables selected. Despite the non-linearities of the problem, LARS managed to find a coherent order for the set of selected variables. The resulting ranking of variables appears in Table 1.

5.5 Scaling

The next step to take in order to optimize the variable selection is to scale the selected variables according to their influence on the output value. Let us consider f as the unknown function that determines the relationship between

the inputs and outputs of a regression problem, $y = f(x) + r$, with $x \in \mathbb{R}^M, y \in \mathbb{R}, r \in \mathbb{R}$. Let d be the number of selected variables that minimize the Delta Test without any scaling. Thus, the estimate of the output, $\hat{y} \in \mathbb{R}$, can be expressed as $\hat{y} = g(x') + r$, with $x' \in \mathbb{R}^d$ and g is the model that best approximates the function f. The objective is to find a scaling vector $\alpha \in \mathbb{R}^d$ such that

$$\hat{y} = g(\alpha_1 x'_1, \alpha_2 x'_2, \ldots, \alpha_d x'_d) + r$$

minimizes the variance of the noise (DT) of the problem.

Intuitively, the scaling will assign high values of α to the variables that are most correlated with the output, and low values to those less correlated. It can happen that some variable that initially was not selected, now enters the set of selected variables with a low scaling factor.

The method of FBS with scaling was applied to the anthropometrics example, providing the ranking shown in Table 3. The result includes all the previously selected variables with scaling factor 1, and adds 6 more with lower scaling factors ranging from 0.1 to 0.5. The computational time employed was 1495.95 seconds and the DT with scaling was reduced to 0.0064. Finally, Table 4 shows a comparison of several OP-ELM models built using the scaled variables.

Scaling factor	Variable number	Variable name
1.0	1	Stature
1.0	8	Erect sitting height
1.0	13	Buttock-knee length
1.0	34	Shoulder breadth
1.0	38	Upper arm circumference
1.0	40	Elbow-hand length
1.0	62	Chest circumference
1.0	65	Waist circumference
1.0	70	Hip circumference
1.0	76	Upper thigh circumference
1.0	80	Calf circumference
1.0	85	Foot length
0.5	42	Forearm circumference
0.4	36	Shoulder-elbow length
0.2	22	Lower face height
0.1	47	Hand breadth
0.1	57	Maximum fist circumference
0.1	112	Birth order

Table 3: Variables selected by FBS with scaling factors (DT = 0.0064).

6 Conclusion

This work has presented the use of variable selection using the Delta Test as a selection criterion, based on the minimization of the variance of the noise

Number of variables	Activation function	Comput. time (s)	LOO error	Test error	Number of neurons
18 (scaled)	L + S	3.61	0.0106	0.0210	68
	L + G	3.71	0.0058	0.0168	17
	L + S + G	3.72	0.0058	0.0166	17

Table 4: Study of the performance of several OP-ELM models using scaled variables. The different activation functions are: L: linear, S: sigmoid, G: gaussian.

in a regression problem. This selection of variables has been combined with optimally pruned ELM models that effectively accelerate the learning process of single layer feedforward neural network.

The OP-ELM models by themselves produced short training times (of the order of 8 seconds for a problem involving 53 variables and around 1000 samples) and relatively accurate estimations in terms of validation and test error. In the example studied, the combination with variable selection using DT reduces the computational time to less than half of the achieved with OP-ELM, maintaining, or improving in some cases, the calculated errors. It also proved to reduce substantially the necessary number of nodes in the network. The best performing models for this application were those which included gaussian kernels, either with or without sigmoid components.

The use of scaling factors to weight the variables according to their importance in the model slightly increases the accuracy but on the other hand it increases the computational time too. Therefore, the convenience of using scaling or not will depend on each specific application.

In particular, we consider that this methodology could be effectively used for time series prediction as it is done in [3] but automatizing the choice of hidden neurons and saving computational time by reducing the number of variables.

References

[1] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, Extreme learning machine: A new learning scheme of feedforward networks, *Neurocomputing*, 70:489–501, 2006.

[2] Y. Liu, H.-T. Loh and S.-B. Tor, Comparison of extreme learning machine with support vector machine for text classification, LNAI 3533:390–399, Springer-Verlag Berlin Heidelberg, 2005.

[3] R. Singh and S. Balasundaram, Application of extreme learning machine method for time series analysis, *Int. Jour. Int. Tech.*, 2(4):256–262, 2007.

[4] Y. Miche, P. Bas, Ch. Jutten, O. Simula and A. Lendasse, A methodology for building regression models using extreme learning machine: OP-ELM, In *Proceedings of ESANN 2008, European Symposium on Artificial Neural Networks*, pp. 247–252, 2008.

[5] H. Pi and C. Peterson, Finding the embedding dimension and variable dependencies in time series, *Neural Computation*, 6(3):509–520, 1994.

[6] A.J. Jones, New tools in non-linear modelling and prediction, *Comput. Manage. Sci.*, 1:109–149, 2004.

[7] B. Efron, T. Hastie, I. Johnstone and R. Tibshirani, Least angle regression, In *Annals of Statistics*, 32:407–499, 2004.

Instance or Prototype Selection for Function Approximation using Mutual Information

A. Guillen[1], L. J. Herrera[2], G. Rubio[2], A. Lendasse[3], H. Pomares[2], and I. Rojas[2] *

1- University of Jaen - Dept. of Informatics
Jaen - Spain

2- University of Granada - Dept. of Computer Technology and Architecture
Granada - Spain

3- Helsinki University of Technology - Lab. of Computer and Information Science
Granada - Spain

Abstract.
The problem of selecting the patterns to be learned by any model is usually not considered by the time of designing the concrete model but as a preprocessing step. Information theory provides a robust theoretical framework for performing input variable selection thanks to the concept of mutual information. The computation of the mutual information for regression tasks has been recently proposed providing good results in feature selection. This paper presents a new application of the concept of mutual information not to select the variables but to decide which prototypes should belong to the training data set in regression problems. The proposed methodology consists in deciding if a prototype should belong or not to the training set using as criteria the estimation of the mutual information between the variables. The novelty of the approach is to focus in prototype selection for regression problems instead of classification as the majority of the literature deals only with the last one. Other element that distinguish this work from others is that it is not proposed as an outlier identificator but as algorithm that determine the best subset of input vectors by the time of building a model to approximate it. As the experiment section shows, this new method is able to identify a high percentage of the real data set when it is applied to a highly distorted data sets.

1 Introduction

The task of selecting the correct subset of input vectors that are included in a training set when classifying, approximating or predicting an output is a relevant task that, if accomplished correctly, can provide storage and computational savings and improve the accuracy of the results.

Three main approaches have been used in order to optimize the set of inputs that the training algorithm will use: *incremental*, *decremental* and *batch*. The incremental approach starts from an empty set of input vectors and defines the training set including input vectors [1, 2]. The opposite perspective is taken in

*This work has been partially supported by the Spanish CICYT Project TIN2007-60587 and Junta Andalucia Project P07-TIC-02768.

the *decremental* approach that starts considering all the input vectors available and, following a prefixed criteria, proceeds to remove the non desired instances [3, 4]. The batch method iterates several times before deleting the instance from the training set, setting a flag on the instances that could be removed in next iterations [5]. Recently, many other approaches have been proposed such as evolutive algorithms [6, 7, 8], boosting-based algorithms [9, 10], and pruning techniques [11].

The work developed in this paper is framed within the decremental approach since it considers the whole data set at the beginning. The criteria to remove the input vectors has been taken from the method used to perform feature selection. The problem of finding the adequate set of variables is quite important by the time of designing models to predict, approximate or classify input data. If the set of input data has redundant or irrelevant data, the training can result in overfitted model with poor generalization capabilities [12, 13, 14]. Furthermore, if the dimensionality is not reduced, some local approximator models suffer the curse of dimensionality, making it impossible to design accurate models.

To tackle the feature selection problem. two main streams have been followed: $filter$ and $wrapper$ methods. The filter approach consists in a preprocessing of the input data so the model is built after. The wrapper approach attempts to design the model at the same time that performs the variable selection. The concepts of entropy and mutual information make the Information theory an interesting framework for filtering approaches.

The majority of the research in prototype selection has been focused in classification problems [8], although few works aimed at solving problems for continuous output. For example [15], presents a method to select the input vectors when calculating the output using the k-Nearest Neighbors algorithm (k-NN), however, this methodology does not permit the selection of the input vectors before designing more complex models such as neural networks. In [6], a genetic algorithm is used to select both the feature and the input subsets, however, it is only suitable for linear regression models. The main problem of genetic algorithms that use binary encoding to determine if an input vector should be included considered, is that the higher the number of instances is, the longer becomes the individual, making difficult to find a good solution.

As commented before, in regression problems, the input and the output values are real and continuous values so additional techniques have to be used to estimate the probability distribution [16].Although there exists a variety of algorithms to calculate the Mutual Information (MI) between variables, this paper uses the approach presented in [17] which is adapted for continuous variables thanks to the use of the MI estimator based on the k-NN algorithm [18].

2 Prototype Selection Based on the Mutual Information

This section firstly describes the mutual information and the methodology to calculate it, then, the algorithm to perform the prototype selection is introduced.

2.1 Mutual Information

Given a single-output multiple input (MISO) function approximation or classification problem, with input variables $X = [x_1, x_2, \ldots, x_n]$ and output variable $Y = y$, the main goal of a modelling problem is to reduce the uncertainty on the dependent variable Y. According to the formulation of Shannon, and in the continuous case, the uncertainty on Y is given by its entropy defined as

$$H(Y) = -\int \mu_Y(y) \log \mu_Y(y) dy, \tag{1}$$

considering that the marginal density function $\mu_Y(y)$ can be defined using the joint PDF $\mu_{X,Y}$ of X and Y as

$$\mu_Y(y) = \int \mu_{X,Y}(x, y) dx. \tag{2}$$

Given that we know X, the resulting uncertainty of Y conditioned to known X is given by the conditional entropy, defined by

$$H(Y|X) = -\int \mu_X(x) \int \mu_Y(y|X = x) \log \mu_Y(y|X = x) dy dx. \tag{3}$$

The joint uncertainty on the $[X, Y]$ pair is given by the joint entropy, defined by

$$H(X, Y) = -\int \mu_{X,Y}(x, y) \log \mu_{X,Y}(x, y) dx dy. \tag{4}$$

The mutual information (also called cross-entropy) between X and Y can be defined as the amount of information that the group of variables X provide about Y, and can be expressed as

$$I(X, Y) = H(Y) - H(Y|X). \tag{5}$$

In other words, the mutual information $I(X, Y)$ is the decrease of the uncertainty on Y once we know X. Due to the mutual information and entropy properties, the mutual information can also be defined as

$$I(X, Y) = H(X) + H(Y) - H(X|Y), \tag{6}$$

leading to

$$I(X, Y) = \int \mu_{X,Y}(x, y) \log \frac{\mu_{X,Y}(x, y)}{\mu_X(x)\mu_Y(y)} dx dy. \tag{7}$$

Thus, only the estimate of the joint PDF between X and Y is needed to estimate the mutual information between two groups of variables.

2.2 Prototype Selection using Mutual Information

The idea that motivates this paper is: since the MI is able to let us know how much information from the output can be retrieved using the different variables starting from a set of input vectors (prototypes), it would be possible that if a significant prototype is removed from the set of input vectors, the amount of MI that could be retrieved would be decreased. On the other hand, if an insignificant prototype is deleted from the original set, the amount of MI should not be decreased. These two sentences are correct, however, there are situations where they might not be completely true. For example, if there are outliers, they will probably provide a significant amount of MI but they should not be considered. On the other hand, if the output of the system remains constant, the amount of information will not fluctuate if similar prototypes are removed.

Thus, in order to make an objective evaluation of how relevant an input vector is, it is necessary to consider the lost of MI relatively to its neighbors in such a way that, if the lost of MI is similar to the prototypes near $\vec{x}_i$, this input vector must be included in the filtered data set. The algorithm proposed to calculate the reduced set of prototypes is described below:

---

calculate the k nearest neighbors in the input space of $\vec{x}_i$ for $i = 1...n$
estimate the mutual information MI between X and the output Y
$diff$=0;
for i=1...n
  calculate the mutual information MIf_i when removing $\vec{x}_i$ from X
end
normalize MIf_i in [0,1]
for i=1...n
  for $cont$=1...α_2
    $diff$= $|MIf_i| - |MIf_{cont}|$
    if $diff > \alpha_1$
      $Cdiff$=$Cdiff$+1
  end
  if $Cdiff < \alpha_2$
    discard prototype
  else
    select prototype
  end
end

---

where α_1 is a predefined threshold that determines how different the MI should be respect the neighbors and α_2 is the number of neighbors to be considered in the comparisons.

When calculating how much of MI was lost, two approaches could be taken: 1) to calculate the MI between the complete set of variables and the output or 2) to compute the MI between each variable and the output. With the MI

RBF centers	radii	weights
0.5463	0.1698	0.4829
0.6366	0.1787	0.2096
0.5709	0.2435	-0.3246
0.9271	0.7518	0.3583
0.8638	0.1991	0.4094

Table 1: Parameters for the function f_1

estimator used in the experiments, no difference between those two approaches could be seen, however, other implementations should be analyzed.

3 Experiments

This section presents the results of applying the new algorithm to highly distorted data sets. These data sets were generated syntheticlly son it was possible to know excatly which elements were the originals and which the noisy ones.

The first experiment was performed using a one dimensional function (Figure 1 a)) that was generated using a gaussian Radial Basis Function Neural Network (RBFNN) ($e^{-\frac{||\vec{x}_k - \vec{c}_i||^2}{r_i^2}}$) using the randomly chosen parameters in Table 1.

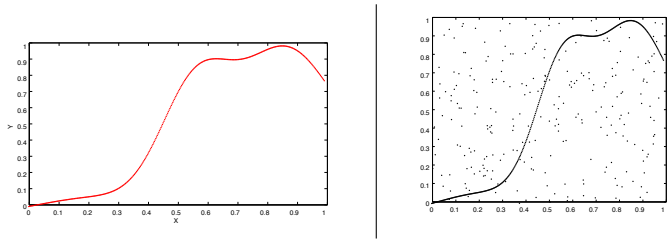


Fig. 1: a) Original target function and b) distorted data set

The original data set consisted in 400 prototypes with their corresponding output. This data set was modified adding a set of 250x2 random values in [0,1] from an uniform distribution, obtaining a new data set of 650 prototypes of dimension 1 with one output. This data set is represented in Figure 1 b).

The proposed method was applied using the value 0.01 for the threshold α_1 and 1 for α_2, obtaining a filtered data set of size 400. From the 250 elements removed, 208 were added prototypes and 42 were original prototypes, this is, the algorithm discriminated the 83.2% of the incorrect prototypes and identified the 89.5% of the original prototypes. Figure 3 depicts the results, where the circles represent the original prototypes and the stars represent the prototypes selected from the distorted data set. If a star is included in a circle, it means that the original prototype was chosen correctly.

To evaluate the utility and effectiveness of the proposed approach, several RBFNNs were designed using the three different data sets: original, distorted

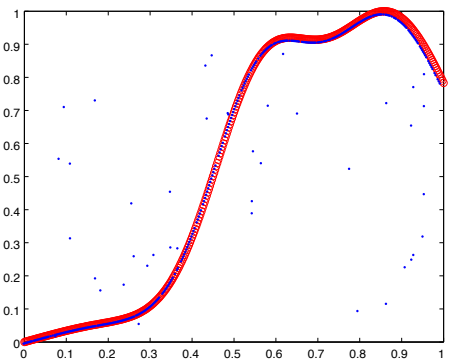


Fig. 2: Filtered data (stars) and original data (circles)

Data set	error	error over original data set
original	0.2124	0.0024
distorted	0.7425	0.4020
selected	0.3265	0.1068

Table 2: Approximation errors (Normalized Root Mean Squared Error) obtained when training the networks using the different data sets.

and filtered. The methodology to design the RBFNN was: first, initialize the centers with the ICFA algorithm [19], then apply k-NN to get a first value for the radii and then, apply a local search to make a fine tuning of these parameters. As it was expected, thanks to the prototype selection, the approximation errors (Table 2) that can be obtained are much smaller than if no prototype selection was made. Figure 3 shows the approximations of the original function by the RBFNNs generated using the distorted data a) and using the data after the prototype selection b).

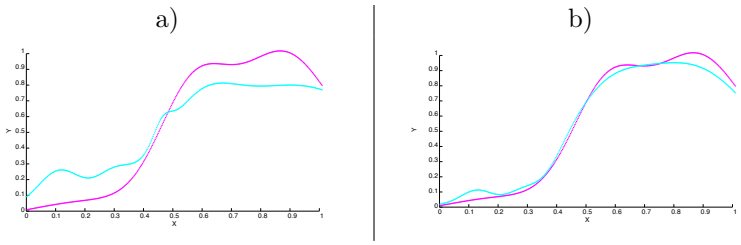


Fig. 3: Approximation made by the RBFNN trained with the data before de prototype selection (a) and after the selection (b)

Secondly, a two dimensional synthetic function f_4 (Figure 4 a)), defined in

Equation 8, was used. First, 400 input vectors were generated, then, 200 input vectors were generated using random values in [0,1] from an uniform distribution, remaining the complete data set as it is depicted in Figure 4 b).

$$f_4(x_1, x_2) = 1.9[1.35 + e^{x_1} sin(13(x_1 - 0.6)^2) e^{-x_2} sin(7x_2)] \; x_1, x_2 \in [0, 1] \quad (8)$$

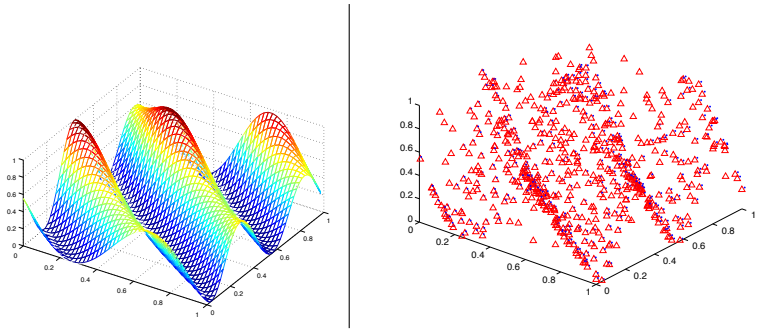


Fig. 4: a) Original target function and b) distorted data set (diamonds) with the original training data (dots)

The algorithm was applied in the same way than in the previous case using $\alpha = 0.015$ and $\alpha_2 = 2$, Figure 5 shows the results. In this occasion, the algorithm identified the 82.75% of the real input vector and the 50% of the noise ones, demonstrating again the good behavior of the proposed approach.

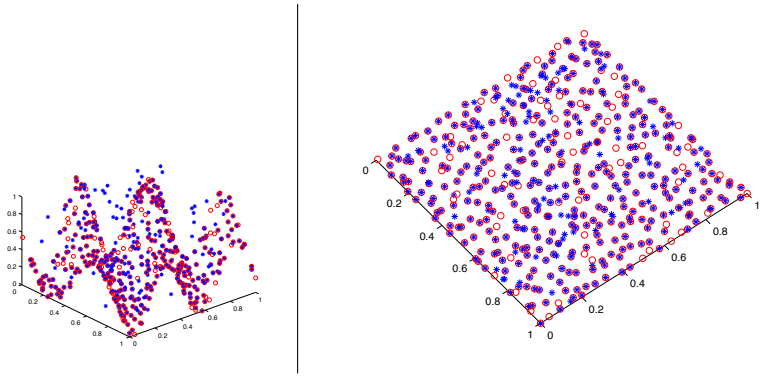


Fig. 5: Filtered data (stars) and original data (circles)

4 Conclusions and Further Work

This paper has presented a possible approach to solve the problem of selecting adequate inputs before using any model to approximate a function. This new

method is based on the concept of MI which was used before for feature selection. The main difference between the already existing approaches and the proposed one is that is oriented to data sets with a continuous output value instead of a predefined set of labels and with the global perspective that the MI provides of the complete data set. As the experiments have shown, the method seems quite effective selecting the correct prototypes with a high accuracy. Further work could be done regarding the influence of the two parameters the algorithm has, how to estimate their values building models to evaluate the quality of the selection, and also a comparison among the different ways of calculating the mutual information.

References

[1] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Mach. Learn.*, 6(1):37–66, 1991.

[2] David W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *Int. J. Man-Mach. Stud.*, 36(2):267–287, 1992.

[3] D. R. Wilso and T. Martinez. Reduction techniques for instance based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[4] D. R. Wilso and T. Martinez. Instance prunning techniques. In *Proceedings of the 14th International Conference on Machine Learning*, pages 404–411. Morgan Kaufmann Publishers, 1997.

[5] I. Tomek. An experiment with edited nearest neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6:448–452, 1976.

[6] J. Tolvi. Genetic algorithms for outlier detection and variable selection in linear regression models. *Soft Computing*, 8(8):527–533, 2004.

[7] R. Baragona, F. Battaglia, and C. Calzini. Genetic algorithms for the identification of additive and innovation outliers in time series. *Computational Statistics & Data Analysis*, 37(1):1–12, July 2001.

[8] H. Ishibuchi, T. Nakashima, and M. Nii. Learning of neural networks with ga-based instance selection. *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, 4:2102–2107 vol.4, 25-28 July 2001.

[9] Richard Nock and Marc Sebban. Advances in adaptive prototype weighting and selection. *International Journal on Artificial Intelligence Tools*, 10(1-2):137–155, 2001.

[10] M. Sebban, R. Nock, and S. Lallich. Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problems. *Journal of Machine Learning Research*, 3:863–865, 2002.

[11] V. B. Zubek and T. G. Dietterich. Pruning improves heuristic search for cost-sensitive learning. pages 27–34, 2002.

[12] S. Haykin. *Neural Networks*. Prentice Hall, New Jersey, 1998.

[13] E. Liitiäinen, F. Corona, and A. Lendasse. Non-parametric residual variance estimation in supervised learning. In *IWANN 2007, International Work-Conference on Artificial Neural Networks, San SebastiÃ¡n (Spain)*, Lecture Notes in Computer Science. Springer-Verlag, June 20-22 2007.

[14] E. Eirola, E. Liitiäinen, A. Lendasse, F. Corona, and M. Verleysen. Using the delta test for variable selection. In *European Symposium on Artificial Neural Networks, Bruges (Belgium)*, April 2008.

[15] J. Zhang, Y. Yim, and J. Yang. Intelligent selection of instances for prediction functions in lazy learning algorithms. *Artificial Intelligence Review*, 11:175–191, 1997.

[16] B.V. Bonnlander and A.S. Weigend. Selecting input variables using mutual information and nonparametric density estimation. In *Proc. of the ISANN*, Taiwan, 2004.

[17] L.J. Herrera, H. Pomares, I. Rojas, M. Verleysen, and A. Guillen. Effective Input Variable Selection for Function Approximation. *Lecture Notes in Computer Science*, 4131:41–50, 2006.

[18] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physics Review*, June 2004.

[19] A. Guillén, J. González, I. Rojas, H. Pomares, L.J. Herrera, O. Valenzuela, and A. Prieto. Improving Clustering Technique for Functional Approximation Problem Using Fuzzy Logic: ICFA algorithm. *Neurocomputing*, DOI:10.1016/j.neucom.2006.06.017, June 2007.

Automatic detection of onset and cessation of tree stem radius increase using dendrometer data and CUSUM charts

Mika Sulkava[1], Harri Mäkinen[2], Pekka Nöjd[2], and Jaakko Hollmén[1] *

1- Helsinki University of Technology
Department of Information and Computer Science
P.O. Box 5400, FI-02015 TKK - Finland

2- Finnish Forest Research Institute - Vantaa Research Unit
P.O. Box 18, FI-01301 Vantaa - Finland

Abstract. Dendrometers are devices, which measure continuously the stem radius of a tree. In this work, we studied the use of cumulative sum (CUSUM) charts for automatically and, thus, objectively determining the onset and cessation dates of radial increase based on dendrometer data. We used data measured in two forest stands in southern Finland to demonstrate the idea and to test the performance of the CUSUM chart. In order to produce reliable results, one has to choose suitable parameter values for the chart. Once configured properly, the method produced results similar to those determined by an expert.

1 Introduction

Formation of wood depends on various endogenic and exogenic factors and it is restricted to a certain period in the year, e.g. [1]. Wood formation is regulated by several factors including genotype, site, silviculture, and climatic variation. In spite of the basic nature of the underlying process, our present knowledge concerning the timing of the various phases and the rate of increment during a growing season is still far from complete. This lack of knowledge is largely due to difficulties in measuring wood formation at short intervals. Dendrometers have traditionally been used for measuring the intra-annual wood formation of trees with high precision, e.g. [2, 3].

Changes in stem dimensions are not solely a result of wood formation; they are often caused by other processes, especially changes in stem hydration, e.g. [4]. Because of the large and frequent changes in stem radius associated with fluctuations in stem water potential, it is difficult to use dendrometer measurements to determine the onset, cessation, and rate of wood formation, i.e., radial increment due to formation of new cells, e.g. [5, 6, 7].

In recent years, few studies have been published describing the timing of radial increase or weather-growth relationship based on dendrometer data, e.g. [7, 8, 9]. However, the definitions and approaches for identifying the onset and cessation of radial increase have been different between the studies. Thus, criteria

*M. S. and J. H. were supported by Academy of Finland, research project: analysis of dependencies in environmental time-series data (AD/ED) (grant number 116853).

and methods for determining onset and cessation dates are not yet generally accepted.

The problem in this study is to objectively and automatically detect the onset and cessation of radial increase period based on dendrometer data. We tested the suitability of the cumulative sum (CUSUM) charts [10, 11] for performing this determination automatically solely based on the data, cf. [12].

2 Dendrometer data

Dendrometer data was collected from two sites located 300 m from each other in southern Finland (Ruotsinkylä, 60° 21' N, 25° 00' E). In both stands, sample trees without visible damage were selected from the dominant tree layer. In the first stand (altitude 45 m a.s.l.), the Norway spruce trees were growing in a pure spruce stand on fertile mineral soil (H_{100} = 30 m, dominant height at age 100 years) classified as *Oxalis-Myrtillus* forest type [13]. Mean stem diameter of the sample trees at breast height was 27 cm and relative crown length was 68%. The sample trees were monitored during the growing seasons of 2001–2005. In the second stand (altitude 60 m a.s.l.), four Norway spruce and four Scots pine trees were monitored during the growing seasons of 2002–2003, and another four spruce and four pine trees during the growing seasons of 2004–2005. They were growing in a mixed spruce-pine stand on a relatively fertile mineral soil classified as *Myrtillus* forest type (H_{100} = 27 m) [13]. The total number of observations (year × tree combinations) was, thus, 57. Mean daily temperatures and precipitation sums were obtained from a meteorological station of the Finnish Meteorological Institute located about 5 km from the study stands.

Stainless-steel band-dendrometers were installed on each tree at a height of about 2 m. Before installing the band, the outer bark under the band was lightly brushed to ensure smooth contact with the trunk. The girth band consists of three basic elements: 1) a stainless-steel band encircling a tree, 2) a sensor (rotating potentiometer) reacting to movements of the stainless-steel band and 3) an aluminium-body and fastening mechanism (Fig. 1). The fastening mechanism consists of three parts: 1) a constant force spring, 2) a fastening arm, and 3) an adjustable foot (Fig. 2). The spring stretches the steel band around the tree with a force of about 3 N, making the band capable of reacting to small variations in girth but without damaging the tree.

Changes in tree girth were measured at a resolution of 0.1 mm, corresponding to diameter change of about 0.03 mm. The output of the dendrometers was stored as hourly averages. Examples of dendrometer data showing the change in the stem radius during one year are shown in Fig. 3. From these measurements, the daily values of stem circumference were calculated as the mean of hourly values and the circumference changes were converted to radial changes assuming a circular stem cross-section. The dendrometer has been described in more detail in [3].

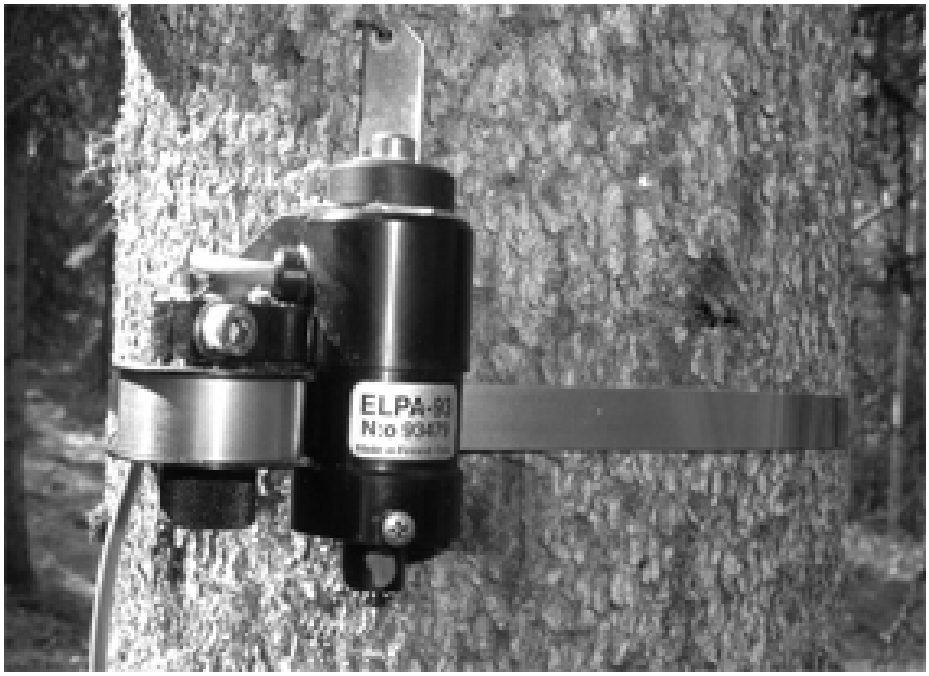


Fig. 1: The girth band mounted on a tree.

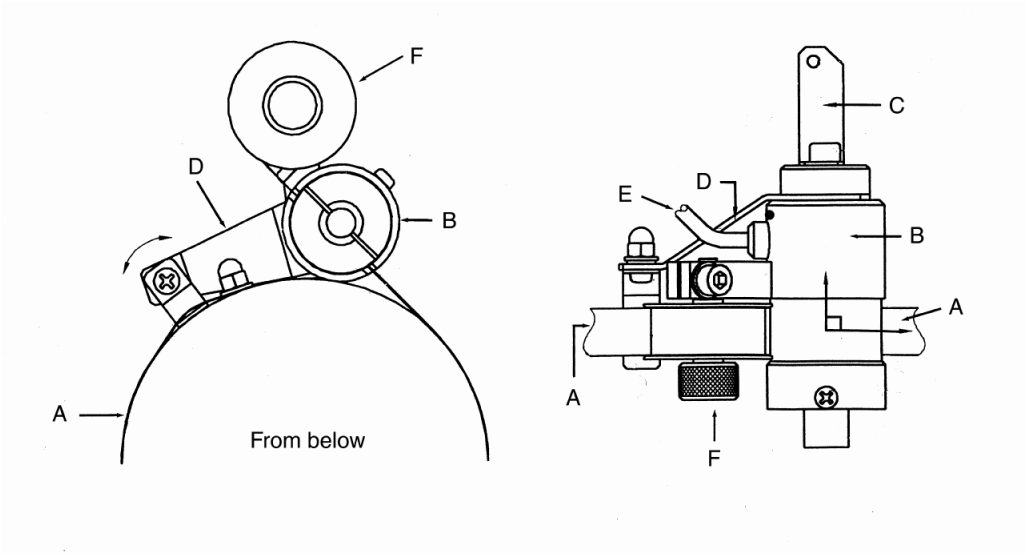


Fig. 2: Structure of the girth band: A = stainless-steel band; B = rotating potentiometer; C = fastening arm; D = adjustable foot; E = cable; F = spring.

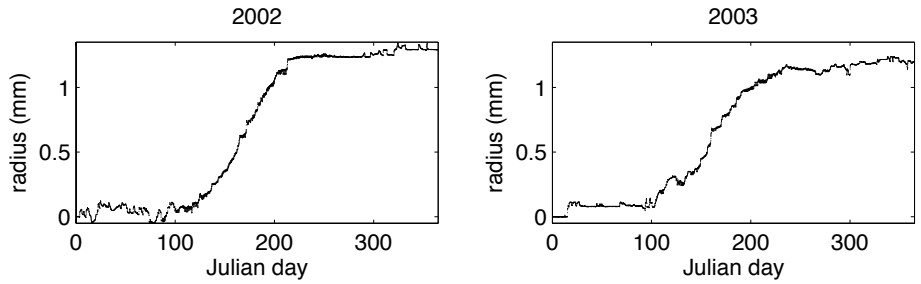


Fig. 3: Examples of the dendrometer data; a Norway spruce tree in Ruotsinkylä measured in 2002 and 2003. The zero level is the first observation of the year. The resolution the measurements is one hour.

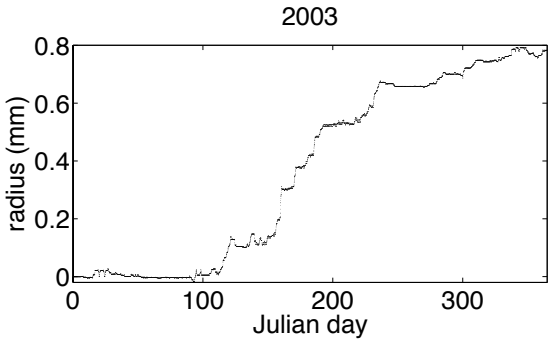


Fig. 4: Example of a difficult growth pattern. It is difficult to distinguish when the actual radial increase starts: on around day 115 or around day 150. Also, only based on the curve it is difficult to tell whether the increase ceases on around day 190, 235, or 340.

We visually evaluated the methods used in the previous studies (cited in the introduction) for determining the onset and cessation dates of radial increase from dendrometer data. All the methods had difficulties in identifying the onset and cessation dates because of stem hydration changes, i.e., they were not sufficient for identification of the crucial dates for all trees and years. It may happen, for example, that during spring stem radius starts to increase at a certain time point, but then stays constant for a rather long time, before the increasing trend reappears (Fig. 4). It is difficult to say, whether new xylem has actually been formed or whether the stem has just swollen.

We ended up, therefore, determining the onset and cessation dates visually from the dendrometer data to obtain labeling for training purposes. It should be emphasized that manual determination of the crucial dates is laborious and is likely to result in observer-related subjective differences and human errors. The dates identified visually are called below as the 'expert choice'.

The tests can be done with all the data. Alternatively, difficult cases (Fig. 4) can be left outside the data in order to get more reliable results. We labeled an onset or cessation date of radial increase as a difficult case if there was a clear jump-like increase of at least about 0.07 mm in stem radius before the onset date or after the cessation date or if the minimum achievable difference between the predicted date and the 'expert choice' was at least 10 days. Altogether, there were 22 and 16 difficult cases for onset and cessation, respectively.

3 Cumulative sum chart

The cumulative sum (CUSUM) chart is a statistical process control tool, which detects small changes in the mean μ of a signal. It monitors the cumulative sum

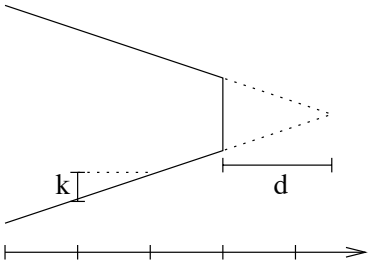


Fig. 5: Parameters d and k of the CUSUM chart define the shape of the V-mask. The shape affects how soon and how large changes are detected.

of previous observations x.

$$X \sim N(\mu, \sigma^2) \tag{1}$$

$$S_m = \sum_{i=1}^{m} (x_i - \mu) \tag{2}$$

The cumulative sum S_m at time step m is the sum of the differences between the previous observations and μ. The CUSUM chart detects up or down drifts and abrupt changes in μ, which is assumed to be known. The chart has 2 parameters: d and k, which depend on standard deviation σ, size of detectable change ΔX, and probabilities of type I (α) and type II (β) errors. The parameters define the shape of a so-called V-mask (Fig. 5), which is used to detect the changes in the mean. In case a previous value of S_m is outside the mask, it is concluded that the mean has changed (see Fig. 6).

We studied how to set the parameters of the chart to produce results as similar as possible to the 'expert choice'. One has to choose suitable values for ΔX, σ, α, and β. In addition the onset or cessation levels μ for radial increase have to determined. Cessation of radial increase can be detected by running the chart in reverse time.

The correct values of suitable detectable size of change and magnitude of noise were not known in advance. Therefore, parameters d and k were varied in a wide range: d between 1 and 50 and k between 0.01 and 0.1. The accuracy of the predicted dates was verified by 10-fold cross-validation [14]. In 10-fold cross-validation, the data set is divided into 10 disjoint sets of equal size. The parameters of the chart yielding the minimum error are selected 10 times and each time one of the sets is held out as a validation data set. Parameter values, which yielded the minimum mean absolute deviation e in days (Equation 3) for a training data set, were chosen. The performance of the model is estimated as the mean of the 10 errors obtained using the validation data sets.

$$e = \frac{1}{n} \sum_{i=1}^{n} |a_e - a_c| \tag{3}$$

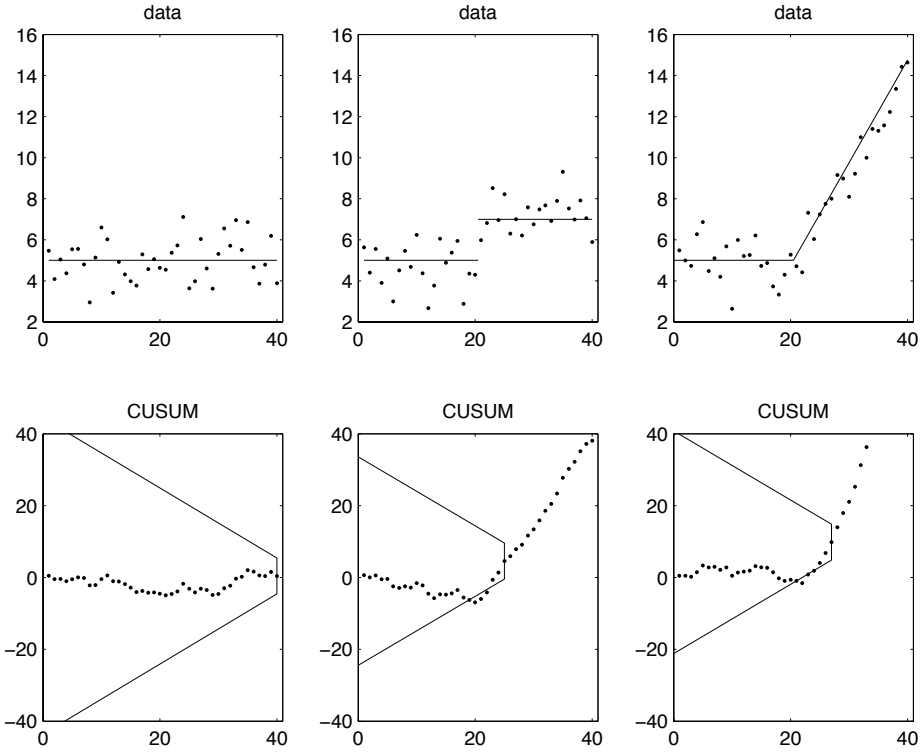


Fig. 6: Artificial data with constant mean (left column) and abrupt (middle column) and gradual change in mean (right column) and the respective CUSUM charts (bottom row). When μ stays the same, the values of cumulative sum S_m stay inside the V-mask. A change in μ is detected, when a previous value of cumulative sum is outside the V-mask.

Above, n is the number of onset or cessation dates in the training or validation set, a_e is the 'expert choice' and a_c is the date predicted by the CUSUM chart.

Performance of the CUSUM chart can be improved by focusing on only upward (onset) or downward (cessation) shifts in mean, i.e., using only one half of the V-mask for detecting the changes. This reduces the number of false alarms. This procedure was used to produce the presented results.

4 Results

It was found that the average stem radius during spring or fall is a better value for μ than the average of whole winter due to, e.g., swelling and shrinkage of the stem. The spring was defined as the two week period before the first day with average temperature > 3 °C. Similarly, fall was defined as the two week period after the first day with average temperature < 3 °C after summer.

Lowest average errors obtained with separate parameter values k and d for each tree and year were 1.6 days for onset and 2.9 days for cessation, respectively. The onset and cessation dates of other trees cannot, however, be detected with the same accuracy using these parameter values, because the optimal parameters differed between the trees and years. Same parameter values were, therefore, used for all the trees and years. The average training errors were 8.6 and 9.8 days for onset and cessation, respectively. The average errors in the validation data set were 8.9 and 11.0 days for onset and cessation, respectively.

The relatively high errors were mainly caused by some difficult cases. The difficulty was caused by increasing stem radius related to stem hydration before the actual radial increase begins and during fall after the radial increase has ceased (Fig. 4). These kinds of curves were also difficult for the CUSUM chart, i.e., it detected the onset date too early and cessation date too late. When the difficult cases were left out, the average errors clearly decreased. The average errors in the training data set were 2.5 days and 7.3 days for the onset and cessation, respectively. The average errors in the validation data set were 3.1 days and 8.2 days for the onset and cessation, respectively.

5 Discussion

The development of stem radius or circumference can be monitored with a high time resolution and summed up to long-term results using modern automatic dendrometers. Durability, automated measurement, and low price make dendrometers suitable to be used in a growth monitoring network covering large regions.

Much of the variation in stem radius is, however, independent of xylem formation. In late winter and early spring, rising temperature may increase evapotranspiration, whereby this water loss cannot be replaced by water uptake due to soil frost and it consequently causes a reduced stem radius, e.g. [7]. Later in spring, water uptake will result in an increase of stem radius not related to the

formation of new tracheids, e.g. [15, 16, 8, 17]. Onset of wood formation may, thus, be masked by the rehydration of the stem, cf. [18].

Likewise, increases in stem radius in late summer may be caused by stem swelling rather than wood formation, making it difficult to determine the cessation of wood formation from dendrometer measurements. Especially in the slow-growing boreal forests, the daily swelling and shrinkage of the stem is relatively large compared to the radial stem increase caused by cell division and enlargement. In cold regions, such as Finland, wintertime ice formation in stems does further complicate the interpretation of girth band data [19].

In the experiments, the CUSUM chart proved to be a good starting point for detecting automatically the dates when the radial increase of trees begins and ceases. Suitable parameters values for the CUSUM chart are needed to get accurate results. Once configured properly, the CUSUM chart produced results similar to the 'expert choice'. In most cases, the results were accurate for the onset of radial growth. To detect the cessation date is more difficult, because during fall the amount of stem radius increase is small compared to the reversible changes in stem radius, which cause false alarms.

A relatively high number of cases (39% and 28% for the onset and cessation, respectively) were labeled as difficult. The changes in stem radius caused by wood formation and other factors can not always be distinguished using the CUSUM chart. It proved necessary to compare the dendrometer measurement with direct measurements on tracheid formation on the stems. Thus, the results achieved by using the CUSUM chart should always be checked by an expert before using them in further analysis. The amount of manual work needed for identifying the crucial dates can anyway be reduced by the CUSUM chart.

The CUSUM chart is also a useful method for on-line detection of onset of radial increase. In contrast, the cessation of radial increase can not be detected on-line using the CUSUM chart. In retrospective analysis of onset and cessation of radial increase it is probably advantageous to use the observations on both sides of the change point to improve the performance of the detection method.

6 Conclusions

In this study, using the CUSUM chart was studied for automatic detection of onset and cessation dates of radial increase of stems based on automated dendrometer data. At their best, the results agreed rather well with dates determined by an expert. Detecting the cessation of radial increase was a more challenging task compared to the onset date.

In the experiments, trees from two stands were analyzed. The method and the estimated parameters can be used to assign preliminary onset and cessation labels to trees in other stands.

Other change detection methods may also be suitable for the problem. In a future study, the CUSUM chart will be compared with, e.g., segmentation and regression methods and trend tests such as the Mann-Kendall test and F-

test. We will also analyse the relationship between environmental factors and the onset and cessation dates of radial increase.

References

[1] R. A. Savidge. Xylogenesis, genetic and environmental regulation (review). *IAWA Journal*, 17:269–310, 1996.

[2] K. Yoda, M. Suzuki, and H. Suzuki. Development and evaluation of a new type of optoelectronic dendrometer. *IAWA Journal*, 21:425–434, 2000.

[3] Erkki Pesonen, Kari Mielikäinen, and Harri Mäkinen. A new girth band for measuring stem diameter changes. *Forestry*, 77:431–439, 2004.

[4] H. Abe and T. Nakai. Effect of the water status within a tree on tracheid morphogenesis in *Cryptomeria japonica* D. Don. *Trees*, 14:124–129, 1999.

[5] Harri Mäkinen, Pekka Nöjd, and Pekka Saranpää. Seasonal changes in stem radius and production of new tracheids in Norway spruce. *Tree Physiology*, 23:959–968, 2003.

[6] Harri Mäkinen, Jeong-Wook Seo, Pekka Nöjd, Uwe Schmitt, and Risto Jalkanen. Seasonal dynamics of wood formation: a comparison between pinning, microcoring and dendrometer measurements. *European Journal of Forest Research*, 127:235–245, 2008.

[7] J. Tardif, M. Flannigan, and Y. Bergeron. An analysis of the daily radial activity of 7 boreal tree species, northwestern Québec. *Environmental Monitoring and Assessment*, 67:141–160, 2001.

[8] G. Downes, C. Beadle, and D. Worledge. Daily stem growth patterns in irrigated *Eucalyptus globules* and *E. nitens* in relation to climate. *Trees*, 14:102–111, 1999.

[9] Annie Deslauriers, Sergio Rossi, and Tommaso Anfodillo. Dendrometer and inta-annual tree growth: What kind of information can be inferred. *Dendrochronologia*, 25:113–124, 2007.

[10] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, June 1954.

[11] Amitava Mitra. *Fundamentals of quality control and improvement*. Prentice-Hall, 2nd edition, 1998.

[12] Mika Sulkava, Harri Mäkinen, Pekka Nöjd, and Jaakko Hollmén. CUSUM charts for detecting onset and cessation of xylem formation based on automated dendrometer data. In Ivana Horová and Jiří Hřebíček, editors, *TIES 2007 – 18th annual meeting of the International Environmetrics Society, Book of Abstracts*, page 111, Mikulov, Czech Republic, August 2007. The International Environmetrics Society, Masaryk University.

[13] A. Cajander. Forest types and their significance. *Acta Forestalia Fennica*, 56:1–71, 1949.

[14] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B (Methodological)*, 36(2):111–147, 1974.

[15] T. T. Kozlowski and C. H. Winget. Diurnal and seasonal variations in radii of tree stems. *Ecology*, 45:149–155, 1964.

[16] K. M. Herzog, R. Häsler, and R. Thum. Diurnal changes in the radius of a subalpine Norway spruce stem: their relation to sap flow and their use to estimate transpiration. *Trees*, 10:94–101, 1995.

[17] R. Zweifel, H. Item, and R. Häsler. Link between diurnal stem radius changes and tree water relations. *Tree Physiology*, 21:869–877, 2001.

[18] K. Kuroda and Y. Kiyono. Seasonal rhythms of xylem growth measured by the wounding method and with a band-dendrometer: an instance of *Chamaecyparis obtusa*. *IAWA Journal*, 18:291–299, 1997.

[19] R. Zweifel and R. Häsler. Frost-induced reversible shrinkage of bark of mature subalpine conifers. *Agricultural and Forest Meteorology*, 102:213–222, 2000.

Multistep-Ahead Prediction of Rainfall Precipitation Using the NARX Network

José Maria P. Júnior[1] and Guilherme A. Barreto[2] *

Federal University of Ceará, Department of Teleinformatics Engineering
Av. Mister Hull, S/N, CP 6005, CEP 60455-760, Fortaleza, Ceará, Brazil.
josemenezesjr@gmail.com, guilherme@deti.ufc.br

Abstract. In this paper, a recently proposed methodology for univariate time series prediction through the NARX network is applied to the problem of forecasting monthly rainfall precipitation in the city of Fortaleza, located at Brazil's northeast coast. The proposed approach is compared with the classic Box-Jenkins (AR) model and other standard neural networks methods (FTDNN and Elman networks). Among these models, the obtained results indicates a better performance for the NARX network.

1 Introduction

Artificial neural networks (ANNs) have been successfully applied to a number of time series prediction and modeling tasks (see [1], for a recent survey of techniques and applications). In particular, when the time series is noisy, the underlying dynamical system is nonlinear and temporal dependencies span long time intervals (also called long memory process), ANN models frequently outperform standard linear techniques, such as the well-known Box-Jenkins models. In such cases, the inherent nonlinearity of ANN models and a higher robustness to noise seem to partially explain their better prediction performance.

In one-step-ahead prediction tasks, the predictive models are required to estimate the next sample value of a time series, without feeding back it to the model's input regressor. If the user is interested in a longer prediction horizon, a procedure known as multistep-ahead or long-term prediction, the model's output should be fed back to the input regressor for a fixed but finite number of time steps [2]. Multistep-ahead prediction is much more complex to deal with than one-step-ahead prediction, and it is believed that these are complex tasks in which ANN models play an important role, in particular recurrent ones.

Simple recurrent networks (SRNs) comprise a class of recurrent neural models that are essentially feedforward in the signal-flow structure, but also contain a small number of local and/or global feedback loops. Even though feedforward networks can be easily adapted to process time series through an input tapped delay line, giving rise to the well-known Focused Time Delay Neural Network (FTDNN), they can also be easily converted to SRNs by feeding back the neuronal outputs of the hidden or output layers, giving rise to Elman and Jordan networks, respectively.

*The authors thank CAPES, FUNCAP and FINEP (PSICO project) for their financial support and FUNCEME for providing the rainfall data.

SRNs are usually trained by means of classic backpropagation algorithm or temporal variants of it (e.g. BPTT and RTRL). However, learning to predict time series with long memory properties can be quite difficult using gradient-based learning algorithms. Lin *et al* [3] report that learning such long-term temporal dependencies with gradient-descent techniques is more effective in a class of SRN model called *Nonlinear Autoregressive with eXogenous input* (NARX) than in Elman/Jordan models. Despite this important property of the NARX network, its feasibility as a nonlinear tool for time series modeling and prediction has not been fully explored yet. Usually, the NARX network is indeed reduced to the FTDNN model in order to be applied to univariate time series prediction [4]. Or, prediction in time of a certain variable is carried out with the help of the time series of another (exogenous) variable [5].

Recently, we evaluated the applicability of the NARX network to *univariate* time series prediction [6, 7]. By univariate we mean that there is no another exogenous variable, i.e. the prediction task is performed using the time series of a single variable only. But even so, the NARX topology can still fully explore its computational power by creating a virtual exogenous variable. According to this approach there are two input regressors to the NARX network, one providing the feedback from the most recent network's outputs and the other taking past values of the variable of interest spaced in time according to Takens' theorem [8]. The first regressor plays the role of an autoregressive model, while the second one plays the role of an (implicit) exogenous variable time series.

In this paper, we investigate the application of the NARX network to the problem of forecasting monthly rainfall precipitation in the city of Fortaleza, located at Brazil's northeast coast. Climate prediction in this tropical region is a particularly hard task due to complexity of the dynamical phenomena involved (e.g. El Niño). Several studies have investigated the predictability of rainfall precipitation in Brazil's northeast cost [9, 10, 11], most of them using complex numerical models of the climate dynamics. Very few papers have investigated the prediction performance of empirical (e.g. statistical and neural) models (e.g. [12]). The main goal of this paper is to help to fill this gap by comparing the NARX approach with standard Box-Jenkins models and other neural network methods (FTDNN and Elman networks).

The remainder of the paper is organized as follows. In Section 2, we describe the NARX network model and its application to univariate time series prediction. Simulations and discussion of results are presented in Section 3. The paper is concluded in Section 4

2 Univariate Time Series Prediction with NARX Network

The NARX model describes an important class of discrete-time nonlinear systems that can be mathematically represented as

$$y(n+1) = f\left[y(n), \ldots, y(n-d_y+1); u(n), u(n-1), \ldots, u(n-d_u+1)\right], \quad (1)$$

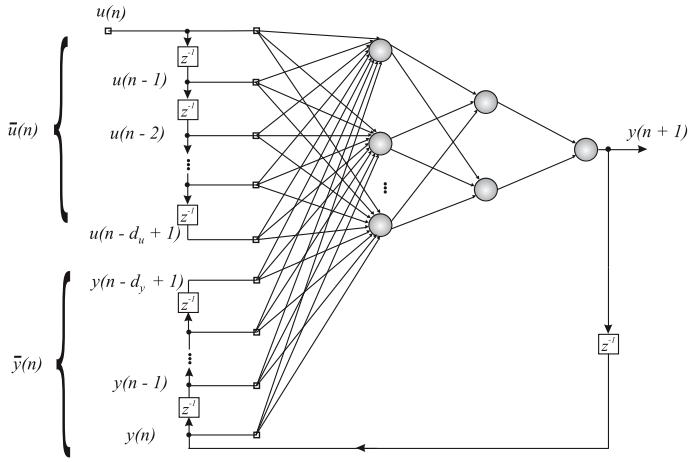


Fig. 1: NARX network with d_u delayed inputs and d_y delayed outputs.

where $u(n) \in \mathbb{R}$ and $y(n) \in \mathbb{R}$ denote, respectively, the input and output of the model at time step n, while $d_u \geq 1$ and $d_y \geq 1$, $d_u \leq d_y$, are the input-memory and output-memory orders, respectively. In vector form it can be written as

$$y(n+1) = f[\mathbf{y}(n); \mathbf{u}(n)], \tag{2}$$

where $\mathbf{y}(n)$ and $\mathbf{u}(n)$ denote the output and input regressors, respectively. The nonlinear mapping $f(\cdot)$ is generally unknown and can be approximated, for example, by an MLP network. The resulting connectionist architecture is then called a *NARX network* [13] (see Figure 1).

As mentioned in the introduction, the particular topic of this paper is the issue of nonlinear univariate time series prediction with the NARX network. In this type of application, the output-memory order is usually set $d_y = 0$, thus reducing the NARX network to the TDNN architecture [4], i.e.

$$\begin{aligned} y(n+1) &= f[\mathbf{u}(n)], \\ &= f[u(n), u(n-1), \ldots, u(n-d_u+1)], \end{aligned} \tag{3}$$

where $\mathbf{u}(n) \in \mathbb{R}^{d_u}$ is the input regressor. This simplified formulation of the NARX network eliminates a considerable portion of its representational capabilities as a dynamic network; that is, all the dynamic information that could be learned from the past memories of the output (feedback) path is discarded.

Takens [8] has shown that, under very general conditions, the state of a deterministic dynamic system can be accurately reconstructed by a time window of finite length sliding over the observed time series as follows:

$$\mathbf{x}_1(n) \triangleq [x(n), x(n-\tau), \ldots, x(n-(d_E-1)\tau)] \tag{4}$$

where $x(n)$ is the sample value of the time series at time n, d_E is the embedding dimension and τ is the embedding delay. If we set $\mathbf{u}(n) = \mathbf{x}_1(n)$ and $y(n+1) =$

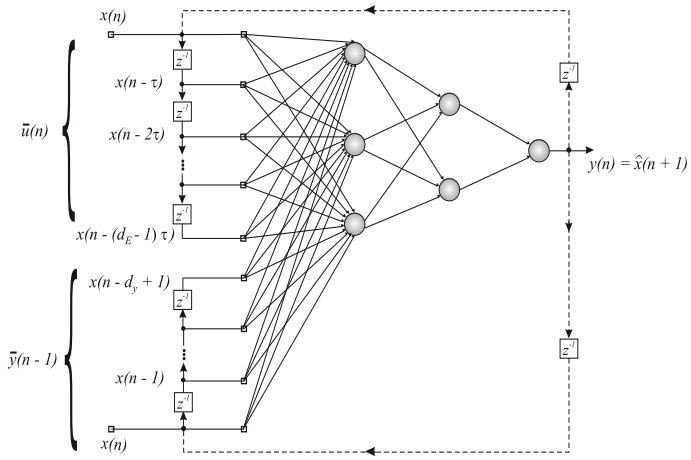


Fig. 2: Architecture of the NARX network during training and testing. The feedback loops (dotted lines) are required only during testing.

$x(n+1)$ in Equation (3), then it leads to an intuitive interpretation of the nonlinear state-space reconstruction procedure as equivalent to the time series prediction problem whose the goal is to compute an estimate of $x(n+1)$. Thus, the only thing we have to do is to train a TDNN model. Once training is completed, the TDNN is used for predicting the next samples of the time series.

Despite the correctness of the TDNN approach, recall that it is derived from a simplified version of the NARX network by eliminating the output memory. In order to use the full computational abilities of the NARX network for nonlinear time series prediction, we proposed novel definitions for its input and output regressors ([6, 7]).

Firstly, the input signal regressor, denoted by $\mathbf{u}(n)$, is defined by the delay embedding coordinates of Equation (4):

$$\mathbf{u}(n) = \mathbf{x}_1(n) = [x(n), x(n-\tau), \ldots, x(n-(d_E-1)\tau)], \tag{5}$$

where we set $d_u = d_E$. In words, the input signal regressor $\mathbf{u}(n)$ is composed of d_E values of the observed time series, separated from each other of τ time steps. Secondly, the output regressor $\mathbf{y}(n)$ can be written as

$$\mathbf{y}(n) = [x(n), \ldots, x(n-d_y+1)]. \tag{6}$$

Note that this regressor contains d_y past values of the observed time series. Henceforth, NARX network is trained using the regression pairs $\{\mathbf{y}(n), \mathbf{x}_1(n)\}$. In a sum, the NARX network implement the following predictive mapping (see Figure 2):

$$\hat{x}(n+1) = \hat{f}[\mathbf{y}(n), \mathbf{u}(n)] = \hat{f}[\mathbf{y}(n), \mathbf{x}_1(n)], \tag{7}$$

where the nonlinear function $\hat{f}(\cdot)$ be readily implemented through a MLP trained with plain backpropagation algorithm.

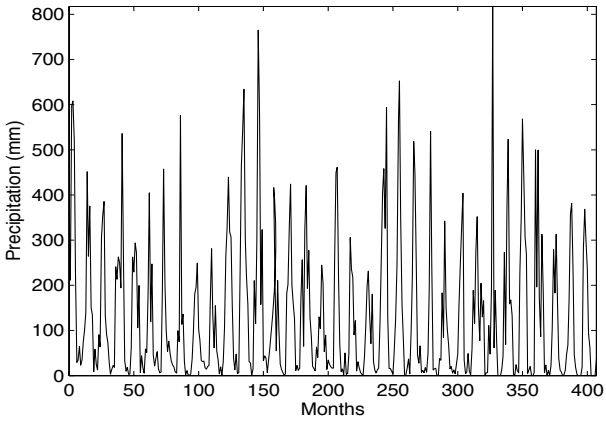


Fig. 3: Monthly rainfall precipitation in Fortaleza, Brazil (Jan/1974-Dec/2007).

Note that Figure 2 illustrates the way the NARX network is trained and tested. During the training the feedback loops (dotted lines) are not used. During the testing phase, however, since multistep-ahead predictions are required, the predicted values should be fed back to both, the input regressor $\mathbf{u}(n)$ and the output regressor $\mathbf{y}(n)$, simultaneously. Thus, the resulting predictive model has two feedback loops, one for the input regressor and another for the output regressor.

Thus, unlike the TDNN-based approach for the nonlinear time series prediction problem, the proposed approach makes full use of the output feedback loop. Equations (5) and (6) are valid only for one-step-ahead prediction tasks. Again, if one is interested in multistep-ahead , the estimates $\hat{x}$ should also be inserted into both regressors in a recursive fashion.

3 Simulations and Discussion

In this paper, our aim is to evaluate the ability of the NARX network on the prediction of monthly rainfall precipitation in the city of Fortaleza, located at Brazil's northeast coast. For the sake of completeness, a performance comparison with the TDNN and Elman networks as well as with the Box-Jenkins AR model is carried out. The time series consists of monthly accumulated rainfall precipitation (in millimeters) observed at the city of Fortaleza, the capital of the brazilian state of Ceará. It was provided by the *Ceará State Institute of Meteorology and Water Resources* (FUNCEME, acronym in portuguese). The data were collected from January of 1974 to December of 2007, resulting in 408 observations (Figure 3).

For training the neural models the time series is rescaled to the range $[-1, 1]$. The series is neither detrended nor deseasonalized. The rescaled time series is

further split into two sets for the purpose of performing holdout validation, so that the first 396 samples were used for training and the remaining 12 samples (one-year-ahead prediction) for testing.

For the AR model, the time series is firstly transformed by the Box-Cox method to reduce data variation and to make it more gaussian-like distributed:

$$Z_t^+ = \begin{cases} (Z_t^\lambda - 1)/\lambda, & \text{if } \lambda \neq 0 \\ \log Z_t, & \text{if } \lambda = 0 \end{cases} \tag{8}$$

where Z_t is the original observation, Z_t^+ is the transformed observation and λ is the transformation exponent. Using Hinkley's method [14] we estimated an optimal value of $\lambda = 0.25$. Secondly, the following variance-stabilizing transformation is applied to the time series $\{Z_t^+\}$ to deseasonalize it:

$$Z_t^{++} = \frac{Z^+_{t(r,m)} - \mu_m}{\sigma_m}, \tag{9}$$

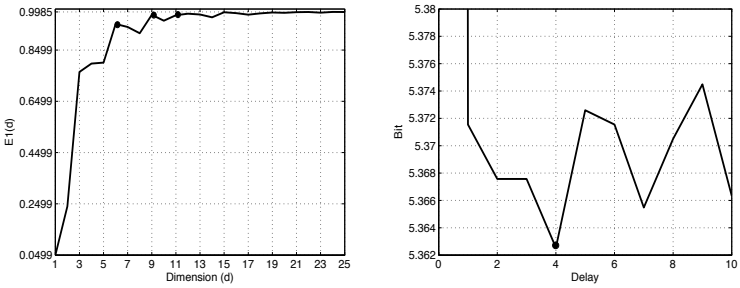
where m is the month index ($m = 1, \ldots, 12$), r is the year index ($r = 1, \ldots, 34$), μ_m and σ_m are, respectively, the mean and standard deviation of the rainfall precipitation computed for the m-th month over all years, and $Z^+_{t(r,m)}$ is the observation corresponding to the m-th month of the r-th year. Finally, an AR(1) is fitted to the time series $\{Z_t^{++}\}$, whose order was found via the AIC method.

All the ANN models have two hidden layers (with 10 and 5 neurons, respectively) and one output neuron. All neurons use hyperbolic tangent activation functions. The number of neurons in each hidden layer was determined after some experimentation with the data. The standard backpropagation algorithm is used to train the networks as one-step-ahead predictors. For the Elman network, only the outputs of the neurons in the first hidden layer are fed back to the input layer. The learning rate was set to 0.01 for the Elman and NARX networks and 0.05 for the FTDNN network.

The embedding dimension (d_E) is estimated by Cao's method [15], which is a variant of the well-known false nearest neighbors method. The curve generated by Cao's method is shown in Figure 4(a). The values to be chosen are the maxima around the "knee" of the curve (i.e. $d_E = \{6, 9, 11\}$). The embedding delay is estimated as $\tau = 4$, obtained from the mutual information method proposed by [16]. This method states that the first minimum in the mutual information curve (see Figure 4(b)) is to be taken as a good estimate for τ.

Once a given network has been trained, it is required to provide estimates of the future sample values of a given time series for a certain prediction horizon h. The predictions are executed in a recursive fashion until desired prediction horizon is reached, i.e., during h time steps the predicted values are fed back to the model's inputs.

For the sake of statistical accuracy, each training/testing run for a given model is repeated $N = 100$ times. For each run, the weights and biases are randomly initialized within the range $[-0.5, +0.5]$. Quantitatively, for the l-th training/testing run, the models are evaluated in terms of the *Normalized Mean*



(a) (b)

Fig. 4: (a) Cao's method curve for embedding dimension estimation, (b) Mutual information curve for embedding delay estimation).

Squared Errors (NMSE) obtained after h time steps:

$$NMSE(h,l) = \frac{1}{h \cdot \sigma_x^2} \sum_{k=1}^{h} \Big(x(n+k) - \hat{x}^{(l)}(n+k) \Big)^2, \tag{10}$$

where $x(n+k)$ is the observed value of the time series at time step $n+k$, $\hat{x}^{(l)}(n+k)$ is the predicted value at time step $n+k$ for the l-th training/testing run, and $\hat{\sigma}_x^2$ is the sample variance of the observed time series.

All tables in the paper report values of the median of NMSE values obtained throughout all the training/testinhg runs for a given prediction horizon h, i.e.

$$\text{median}[NMSE(h,1), NMSE(h,2), \ldots, NMSE(h,N)]. \tag{11}$$

The use of the median instead of the mean value of $NMSE(h,l)$, for a given h, is preferred since it is more robust to outliers [17].

Since the Cao's method indicated 3 possible values for d_E, we decided to test all the combinations within the ranges $d_E \in [5..14]$ and $\tau \in [2..9]$ and choose the pair (d_E, τ) that gives the lowest $NMSE$ value. The 12-step-ahead prediction results of a FTDNN network trained for 1500 epochs and validated by the holdout method are shown in Table 1. The best pair found is $(d_E, \tau) = (11, 4)$, thus confirming the values suggested in Figure 4. Adopting the same methodology, we have obtained $(d_E, \tau) = (13, 4)$ for the Elman network and $(d_E, \tau) = (14, 4)$ for the NARX network. In what concern the NARX network, we have used $d_y = 5$, a value that turned out to be suitable for the current data.

Having selected and trained all models, the next test aims at evaluating their long-term predictive performance. Box-plots[1] for $NMSE(h,l)$ samples (one sample of size 100 for each predictive model), obtained for $h = 12$ and $l = 1, \ldots, 100$, are shown in Figure 5. The AR model presents no variation of

[1]A boxplot is a graphical way of depicting numerical data through five quantities: the smallest observation, lower quartile, median, upper quartile, and largest observation.

Table 1: NMSE as a function of (d_E, τ) ($h = 12$, number of epochs=1500).

	τ=2	τ=3	τ=**4**	τ=5	τ=6	τ=7	τ=8	τ=9
d_E =5	0.1630	0.3423	0.1838	0.2270	0.8843	0.3358	0.3987	0.1233
d_E =6	0.1521	0.3762	0.2085	0.4910	0.9207	0.3853	0.2432	0.1342
d_E =7	0.2151	0.2887	0.1951	0.4507	0.6039	0.3029	0.0771	0.1182
d_E =8	0.3323	0.3134	0.1967	0.4021	0.3339	0.1775	0.1259	0.1438
d_E =9	0.3376	0.2973	0.3450	0.2833	0.3255	0.1773	0.1418	0.1505
d_E =10	0.2555	0.3143	0.0750	0.2264	0.3062	0.1805	0.0917	0.2292
d_E =**11**	0.2565	0.3662	**0.0615**	0.1667	0.3575	0.2059	0.2917	0.2653
d_E =12	0.2235	0.2037	0.0751	0.1807	0.3660	0.3380	0.2379	0.2358
d_E =13	0.2649	0.1083	0.1085	0.2165	0.3644	0.3928	0.2908	0.2307
d_E =14	0.3233	0.1408	0.0876	0.1662	0.4230	0.3527	0.2077	0.1928

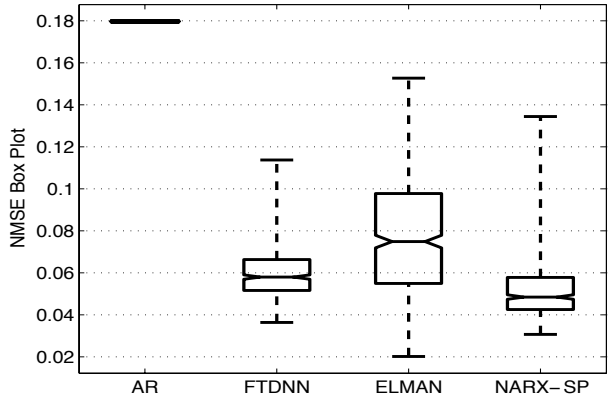


Fig. 5: NMSE values provided by the several evaluated models ($h = 12$).

the results since its parameters are computed through the (batch) least-squares method.

This figure reveals that the NARX network has the best performance, followed closely by the FTDNN model. An interesting issue to highlight is the poor performance of the Elman network (worst performance among the neural models). One possible explanation for this poor performance is that the type of feedback path used by the Elman network amplifies the accumulated prediction error (noise) much more than the one used by the NARX network. The presence of a direct feedback path from the output gives additional predictive power to the NARX network, while the absence of it in the FTDNN network puts it in an intermediate situation in terms of performance.

Finally, Figure 6 shows the 12-month-ahead prediction results for the NARX network corresponding to the year 2007. The solid line indicates the actually observed values of rainfall precipitation, the vertical rectangles denote the values

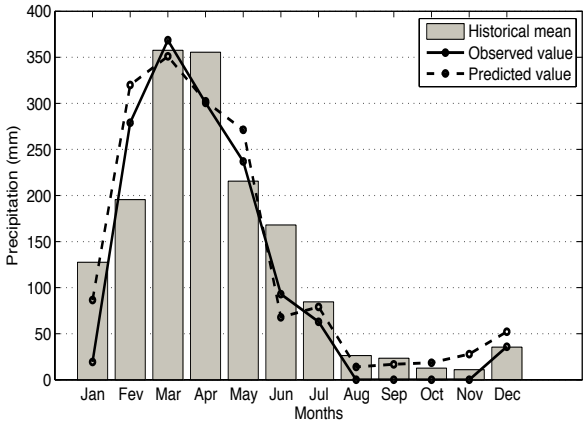


Fig. 6: Predicted values for the NARX network ($h = 12$) averaged over 100 runs.

of the historical mean of the corresponding month computed from 1974 to 2007, and the dotted line shows the predicted values averaged over 100 runs. The results are considered very good by the meteorologists in FUNCEME, being comparable to the results provided by the complex phenomenological (numerical) model of the climate dynamics they really use to deliver rainfall predictions to the society. The main advantage of the ANN (empirical) approach is its velocity in providing the estimates (few minutes, including training and testing), while the numerical model takes several hours.

4 Conclusions and Further Work

In this paper, we evaluate the predictive ability of the NARX network in providing multistep-ahead estimates of rainfall precipitation in the city of Fortaleza, located at Brazil's northeast coast. A performance comparison was carried out among several models (AR(1), FTDNN, Elman and NARX), with the best performance being provided by the NARX model. Currently, we are extending the approach to the prediction of rainfall precipitation in other locations within the state of Ceará. An in-depth performance comparison between the NARX network and the numerical model used by FUNCEME for climate prediction is being also carried out.

References

[1] A. K. Palit and D. Popovic. *Computational Intelligence in Time Series Forecasting*. Springer Verlag, 1st edition, 2005.

[2] A. Sorjamaa, J. H. N. Reyhani, Y. Ji, and A. Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16–18):2861–2869, 2007.

[3] T. Lin, B. G. Horne, and C. L. Giles. How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. *Neural Networks*, 11(5):861–868, 1998.

[4] T. Lin, B. G. Horne, P. Tino, and C. L. Giles. A delay damage model selection algorithm for NARX neural networks. *IEEE Transactions on Signal Processing*, 45(11):2719–2730, 1997.

[5] L. Pape, B.G. Ruessink, M. A. Wiering, and I. L. Turner. Recurrent neural network modeling of nearshore sandbar behavior. *Neural Networks*, 20(4):509–518, 2007.

[6] J. M. Menezes-Júnior and G. A. Barreto. A new look at nonlinear time series prediction with NARX recurrent neural network. In *Proceedings of the 9th Brazilian Symposium on Neural Networks (SBRN'2006)*, pages 28–33. IEEE Computer Society, 2006.

[7] J. M. Menezes-Júnior and G. A. Barreto. On recurrent neural networks for auto-similar traffic prediction: a performance evaluation. In *Proceedings of the Proceedings of the 2006 IEEE/SBrT International Telecommunications Symposium (ITS'2006)*, pages 495–500, 2006.

[8] F. Takens. Detecting strange attractors in turbulence. In D. A. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer, 1981.

[9] L. Greischar and S. Hastenrath. The rainy seasons of the 1990s in northeast Brazil: real-time forecasts and verification. *Journal of Climate*, 13(21):3821–3826, 2000.

[10] C. K. Folland, A. W. Colman, D. P. Rowell, and M. K. Davey. Predictability of northeast Brazil rainfall and real-time forecast skill, 1987-98. *Journal of Climate*, 14(9):1937–1958, 2001.

[11] A. C. Harvey and R. C. Souza. Assessing and modeling the cyclical behavior of rainfall in northeast Brazil. *Journal of Applied Meteorology*, 26(10):1339–1344, 1987.

[12] A. D. Moura and S. Hastenrath. Climate prediction for Brazil's nordeste: Performance of empirical and numerical modeling methods. *Journal of Climate*, 17(13):2667–2672, 2004.

[13] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[14] D. Hinkley. Miscellanea: On quick choice of power transformation. *Applied Statistics*, 26(1):67–69, 1977.

[15] L. Cao. Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D*, 110(1–2):43–50, 1997.

[16] A. M. Fraser and H. L. Swinney. Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33:1134–40, 1986.

[17] J. G. De Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 22(3):443–473, 2006.

Hybrid Criteria for Nearest Neighbor Selection with Avoidance of Biasing for Long Term Time Series Prediction

Syed Rahat Abbas and Muhammad Arif

Pakistan Institute of Engineering and Applied Sciences
Department of Computer and Information Sciences
P. O. Nilore, Islamabad, Pakistan

Abstract. Nearest neighbor is pattern matching method for time series prediction in which most recent values of the time series are compared with previous available values and forecasting is achieved by finding the best match pattern (nearest neighbor). Usually Euclidean distance is used to check the similarity of pattern. In this paper two hybrid criteria of pattern matching are being proposed and evaluated for multistep-ahead time series prediction. The first selection criterion is hybrid of "Maximum distance and Cross-Correlation" and second is hybrid of 'Manhattan distance and Cross-correlation". Better forecasting has been achieved using these algorithms.

1 Introduction

Time series prediction plays an important role in management of many systems. A huge pyramid of prediction methods are available based upon simple regression to very complex machine learning methods. Each method has pros and cons.

Nearest neighbor method is a pattern matching method in which the most recent pattern of the time series (reference pattern) is matched with all the available past patterns (candidate patterns). The prediction is carried out by the next value of the best matched pattern.

Nearest neighbor method was initially proposed by Cover and Hart [1]. In different forms it has been used for classification and prediction problems. Modifications in the nearest neighbor method were carried out time to time. Time series prediction using delay coordinate embedding was proposed in [2]; the mixture of direct and iterated method for prediction using four nearest neighbors with interpolation was carried out and method was applied for Santa Fe time series prediction competition in 1992. The nearest neighbor method with upsampling and cross-correlation was carried out [3]. The comparison of nearest neighbor method with other method for prediction of foreign exchange shows that results are data dependent [4]. Simultaneous nearest neighbor method performed marginally better than ARIMA and random walk methods as reported in [5]. The prediction of chaotic behavior of market response is carried out using multivariate nearest neighbor method for precise prediction [6]. Divide and conquer approach to develop pair-wise class nearest neighbor method was proposed in [7]. Locally adaptive metric nearest-neighbor classification method was also proposed in [8]; they have used updating of weighted distance for getting optimal nearest neighbors. It was proposed that advanced data structures significantly reduce the execution time of nearest neighbor

regression [9]. Subset features space was used by to improve nearest neighbor classification [10]. Subspace of candidate was used by [11] for fast search of nearest neighbors. Discriminate adaptive nearest neighbor classification was suggested by [12]. They used local linear discriminant analysis to estimate an effective metric for computing neighborhoods. The nearest neighbor method in economics is also used recently [13]. The hybrid of Euclidean distance and normalized cross-correlation method [14] is proposed by us; which provided better forecasting than classical nearest neighbor method.

In this paper the hybrid criterion of maximum distance with normalized cross-correlation and Manhattan distance with normalized cross-correlation is being proposed.

2 Proposed algorithm for time series prediction

In nearest neighbor method last few values of the available time series are taken which are considered as referenced pattern. The number of values of the time series used for matching are called window size (*'w'*). The reference pattern is compared with all available patterns (candidate patterns) of same length. The forecasting is achieved as the next value of best matched pattern. The schematic of nearest neighbor algorithm is illustrated as Fig 1.

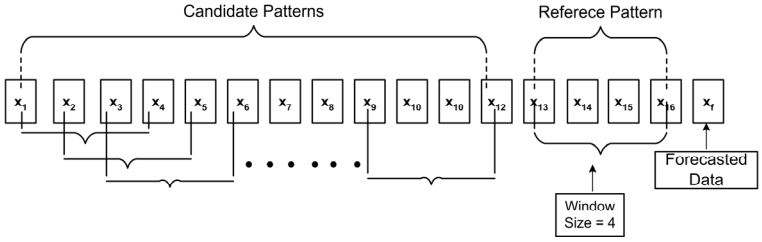


Fig 1: Schematic For Nearest Neighbor Search

The main steps are (a) window size selection (b) pattern matching (3) prediction procedure. Following is the detail of these steps.

2.1 Search for optimal window size

For nearest neighbor algorithm the first maximum after lag=0 of Auto-Correlation Function (ACF) plot gives the useful window size [3], [15]. The window sizes (*'w'*) of six series studied in this paper are approximated by ACF plot and shown in Table 1. The description of time series and their sources are presented in section 3.

Table 1: Window Size for Time Series

Series	Window Size
Sunspot	10
IOWA Electricity Series	12
River Series	12
ESTSP08 First Series	12
ESTSP08 Second Series	7
ESTSP08 Third Series	24

The ACF plot for ESTSP Competition Series (2nd) is shown in Fig 2.

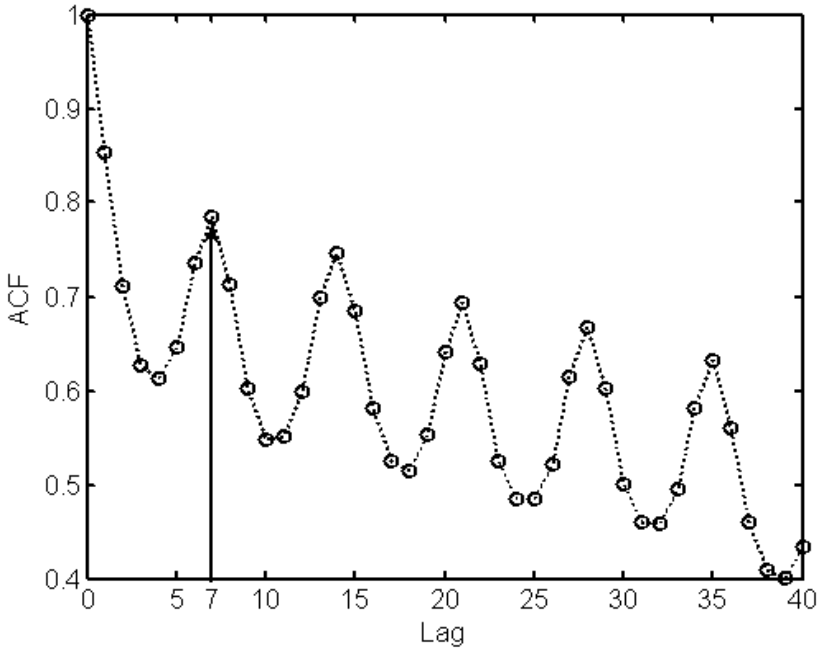


Fig 2: ACF plot of ESTSP08 Competition Series (2nd)

2.2 Usual Pattern Matching Criterion

Usually in nearest neighbor algorithm the best match pattern is selected which has the least Euclidean distance from the reference pattern. The Euclidean distance 'Ed' between two vectors 'X' and 'Y' is given by **Error! Reference source not found.**.

$$E_d = \sqrt{\sum_i (X(i) - Y(i))^2}$$

2.3 Proposed Matching Criterion

Euclidean distance based search in the standard nearest neighbor gives similarity in terms of the distance between the two patterns without considering the shape of two patterns. Two other distances i.e. Maximum distance and Manhattan distance are also used for pattern matching. The maximum and Manhattan distance between two vector X and Y are given by

$$r_{\max} = \max \left| X(i) - Y(i) \right|$$

$$r_{man} = \sum_i \left| X(i) - Y(i) \right|$$

The distances are amplitude dependent for example if Sin(x) and 5Sin(x) is considered they will give high value of distance between them. We can also use zero order cross correlation to find the best nearest neighbor in terms of shape. Zero order cross correlation of two vectors can be described as follows, If ' X' and 'Y' are two vectors, the normalized cross-correlation $Xcorr_{norm}$ with delay 'TD' is defined as

$$Xcorr_{norm}(TD) = \frac{\sum_i X(i)\ Y(i - TD)}{\sqrt{\sum_i X^2(i) * Y^2(i)}}$$

For zeroth order cross-correlation $TD = 0$. The normalized cross-correlation is amplitude independent. It will give value '1' (perfect match) for Sin(x) and 5Sin(x).

An example of calculation is being presented to highlight the effect of these distances and cross correlation. Let us consider the following set of patterns sampled with time step 0.1.

$$x = \sin t$$

$$y = 2.5 \sin t$$

$$z = \cos(t + 0.3) \qquad t \in [0, 2.5\pi]$$

$$v = 2 \sin(t + 0.4)$$

$$p = 0.3 \cos t$$

Let 'x' is our reference pattern and other four are candidate patterns. Normalized zero order cross-correlation, Maximum and Manhattan distances of the reference pattern to the candidate patterns are given in Table 2. Maximum value of cross-correlation is considered as the measure of closest pattern and minimum value of distance is considered as the criterion for the closest pattern. If we consider only cross-correlation as selection criterion then the closest match of pattern 'x' is 'y' and

if we consider the maximum or Manhattan distance only, the closed pattern of 'x' is 'p'. But Error column shows that closest pattern is 'v'.

Cross-correlation based search can give the best nearest neighbor having similar shape but it is possible that the amplitudes of the two patterns may differ a lot. we have proposed hybrid selection criteria using normalized cross correlation and maximum distance and cross correlation with Manhattan distance. The normalized cross-correlation is invariant to the amplitude of the patterns but depends on the shape of the two patterns. Combining the two selection criteria can give us better pattern selection considering both shape and amplitude.

Table 2: Distance and Cross-Correlation of Sin(t) with other series

Candidate Series	Series Name	Normalized Cross-correlation	Manhattan Distance	Maximum Distance	Error with Actual Value
2.5 Sin(t)	y	1.000	74.9352	1.4999	1.492
Cos(t+0.3)	z	-0.179	74.5138	1.6096	1.384
2Sin(t+0.4)	v	0.928	62.4379	1.1470	0.672
0.3 Cos(t)	p	0.126	49.7579	1.0440	1.025

Following is the algorithm of the hybrid selection criteria,

Step 1: Take the zeroth order normalized cross-correlation of the reference pattern with the candidate patterns and arrange them in descending order according to their cross-correlation values.

Step 2: Pick only those candidate vectors whose cross-correlation value with the reference pattern is greater than θ. We have tried different value of θ and found that it can be taken as 0.8. If no such candidate vector exists then only maximum/Manhattan distance will be use for pattern matching.

Step 3: Calculate the maximum/Manhattan distance of all the patterns selected in step 2 with the reference vector and consider the best nearest neighbor having minimum maximum/Manhattan distance with the reference pattern.

Considering the above example, we will select vectors 'y' and 'v' only based on their cross correlation values with the reference pattern (Step 2). Considering the maximum or Manhattan distances of 'y' and 'v' from reference pattern 'x', pattern 'v' will be selected as the nearest neighbor. From the Table 2, it can be seen that minimum error in the forecasting of 'x' is achieved by using the pattern 'v'.

In long-term forecasting, forecasted values are iterated to get multi-step ahead prediction. Hence any forecasting error occurred at a certain time will be propagated

and increased in the later forecasting values. This makes the accuracy of the forecast value and the comparison strategies to find the nearest neighbor a critical issue for long range forecasting.

2.3.1 Avoidance of Biasing

Let for some *'ith'* step ahead, the query vector is $[x_i \quad x_{i+1} \quad . \quad . \quad . \quad x_{i+w-1}]$ and the selected vector from the database is the *'rth'* vector $=[x_r \quad x_{r+1} \quad . \quad . \quad . \quad x_{r+w-1}]$. For *'(i+1)th'* step, the query vector will become $[x_{i+1} \quad x_{i+2} \quad \ldots \quad x_{r+w}]$. The *'(r+1)th'* vector in the database is $[x_{r+1} \quad x_{r+2} \quad \ldots \quad x_{r+w}]$. As the last value of both the query vector and *'(r+1)th'* vector are exactly same so the search in *'(i+1)th'* step will be biased towards this *'(r+1)th'* vector in the database. To remove this biasing effect, it is proposed that the last value of the query vector will not participate in calculating the maximum or Manhattan distance.

2.3.2 Prediction on the base of best match pattern

The best match pattern is one which has the maximum correlation value, after finding it the prediction of one value is achieved as the next value in the time series of the best matched pattern.

For the multistep-ahead prediction, the reference vector is updated by dropping the oldest value in it and padding the forecasted value at the end so that the length of the reference pattern remains intact. The new reference vector is again matched with candidate patterns and this process is iterated for required prediction steps.

3 Results and Discussion

To evaluate the proposed algorithm, three time series from the forecasting literature are studied. First series is famous Wolfer Sunspot number time series which is chaotic and a benchmark for time series prediction. 200 values were used to forecast next 50 values. The second series is monthly electricity consumption in IOWA city US. 70 data values were used to forecast next 30 values. The third series is river flow at fair oaks, California for the period October 1906 to September 1960. First 540 values were taken to forecast next 60 values. These series are taken from time series data library by R. J. Hyndman (web:http://wwwpersonal.buseco.monash.edu.au/ hyndman/tsdl/) .

In Table 3 the comparison of forecasting error by using standard Euclidean distance based nearest neighbors (SNN) algorithm and proposed hybrid algorithms are presented. In case of sunspot series the normalized mean squared error (NMSE) was reduced from 2.581 to 0.8143 in case of hybrid of maximum distance. The hybrid of Manhattan distance did not improved results in this case. For electricity consumption time series the NSME with classical nearest neighbor method was 0.4915 which reduced to 0.2641 in case of Manhattan and further decreased to 0.1943 in case of maximum distance. In case of River flow series the NMSE reduced from

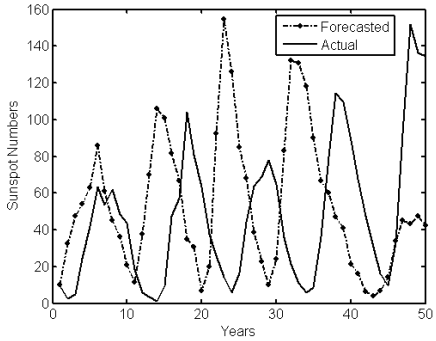
0.9563 to 0.9158 in case of hybrid of maximum distance. Manhattan distance in this case degrades the results.

Both of the proposed algorithms also used to forecast ESTSP'08 Competition time series. For dataset 1 only third column is used and exogenous inputs are ignored. Dataset 3 is very long; its subset is taken using visual guess. Three subsection of dataset 3 are taken for nearest neighbor search i.e. (13001:14500), (21501:23500), (30001:31614).

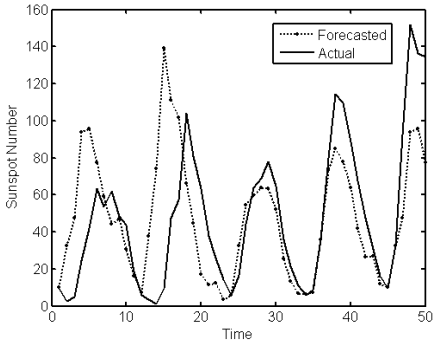
Table 3: Forecasting Results using Proposed Algorithm

Sr. No	Time Series	NMSE		
		SNN	Max Distance + Xcorr	Manhatt+Xcorr
1	Sunspot	2.581	0.8143	2.5811
2	IOWA Elec	0.4915	0.1943	0.2641
3	River	0.9563	0.9158	1.4828

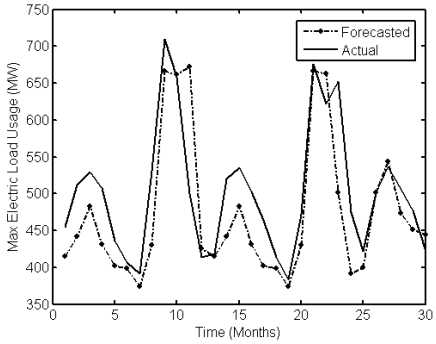
In case of hybrid algorithm of Euclidean distance and cross-correlation [14] the NMSE for Sunspot time series was 0.747, for IOWA elec. time series was 0.4930 and for River series it was 0.8956. So for different time series different algorithm performed well and there is not general conclusion.



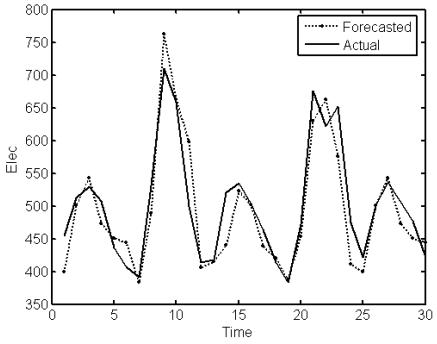
Using classical NN for Sunspot Series



Using proposed algorithm for Sunspot series



Using classical NN for IOWA Electric Series



Using proposed algorithm for IOWA Electric Series

Fig 3: Comparison of classical nearest neighbor method and proposed algorithm

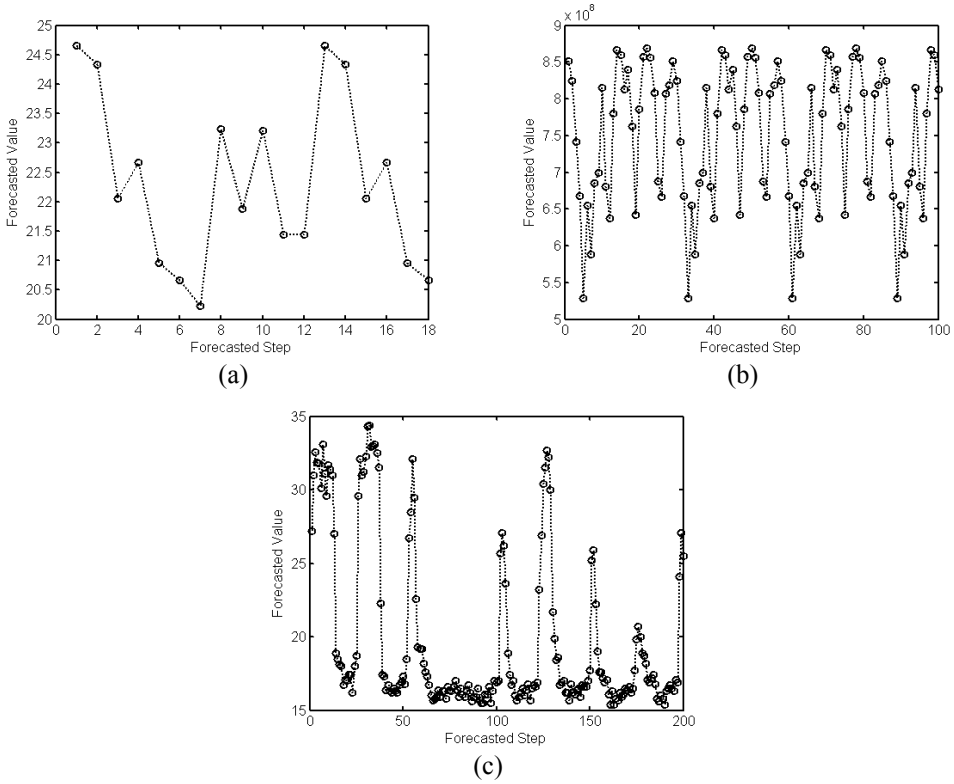


Fig 4: Forecasting Plots of ESTSP'08 Competition (a) Dataset 1 (b) Dataset 2 (c) Dataset 3

It has been found that hybrid criterion of nearest neighbor selection based on maximum distance and cross-correlation performed better than that of Manhattan distance (Table 3, in table 3 Xcorr is abbreviation of cross-correlation). In future hybrid of other distances with cross-correlation can be studied.

4 Conclusion

The hybrid criteria based on Maximum/Manhattan distances and zeroth order normalized cross-correlation are proposed. It is found that forecasting results for Sunspot series, IOWA electricity consumption series and River Flow series has been improved especially when maximum distance and cross-correlation is used. The forecasting results for ESTSP'08 competition series have been submitted.

References

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21-27, 1967.

[2] T. Sauer, "Time Series Prediction by Using Delay Coordinate Embedding," in *Time series prediction: Forecasting the future understanding the past*, A. S. Weigend and N. A. Gershenfeld, Eds.: Addison Wesley, 1993, pp. 175-193.
[3] S. R. Abbas and M. Arif, "Long Range Time Series Forecasting by Upsampling and using Cross-Correlation Based Selection of Nearest Neighbor," *International Journal of Pattern Recognition and artificial intelligence (IJPRAI)*, vol. 20, pp. 1261-1278, 2006.
[4] N. Meade, "A comparison of the accuracy of short term foreign exchange forecasting methods," *International Journal of Forecasting*, vol. 18, pp. 67-83, 2002.
[5] F. Fernandez-Rodriguez, S. Sosvilla-Rivero, and J. Andrada-Felix, "Exchange-rate forecasts with simultaneous nearest-neighbour methods: evidence from the EMS," *International Journal of Forecasting*, vol. 15, pp. 383-392, 1999.
[6] F. J. Mulhern and R. J. Caprara, "A nearest neighbor model for forecasting market response," *International Journal of Forecasting*, vol. 10, pp. 191-207, 1994.
[7] T. Raicharoen and C. Lursinsap, "A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm," *Pattern Recognition Letters*, vol. 26, pp. 1554-1567, 2005.
[8] C. Domeniconi, J. Peng, and D. Gunopulos, "Locally Adaptive Metric Nearest-Neighbor Classification," *IEEE transaction on pattern analysis and machine intelligence*, vol. 24, 2002.
[9] B. L. Smith, "Effect of Parameter Selection on Forecast Accuracy and Execution Time in Nonparametric Regression," presented at IEEE intelligent Transportation system Conference, USA, 2000.
[10] S. B. Day, "Combining Nearest Neighbor Classifiers Through Multiple Feature Subsets," presented at 5th International Conference on Machine Learning, Madison WI, 1998.
[11] A. Djouadi and E. Bouktache, "A fast Algorithm for Nearest-Neighbor Classifier," *IEEE transaction on pattern analysis and machine intelligence*, vol. 19, 1997.
[12] T. Hastie and R. Tibshirani, "Discriminant Adaptive Nearest Neighbor Classification," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 18, 1996.
[13] B. Sloboda, "Nonparametric econometrics: Theory and practice," *International Journal of Forecasting*, vol. 23, pp. 717-719, 2007.
[14] S. R. Abbas and M. Arif, "Modified Nearest Neighbor Method For Multistep Ahead Time Series Forecasting," *International Journal of Pattern Recognition and artificial intelligence (IJPRAI)*, vol. 21, pp. 463-481, 2007.
[15] R. Abbas, W. Aziz, and M. Arif, "Modified Nearest Neighbour Algorithm for time series forecasting," *Sci. Int. (Lahore)*, vol. 17, pp. 191-194, 2005.

Playout Delay Prediction in VoIP Applications: Linear versus Nonlinear Time Series Models

José B. Aragão Jr.[1] and Guilherme A. Barreto[2] *

1, 2 - Federal University of Ceará - Department of Teleinformatics Engineering
Av. Mister Hull, S/N - Campus of Pici - Center of Technology
CP 6007, CEP 60455-970, Fortaleza, Ceará, Brazil

Abstract. Voice over IP (VoIP) applications requires a buffer at the receiver to minimize the packet loss due to late arrival. Several algorithms are available in the literature whose goal is to predict an optimal playout buffer delay. Classic algorithms differentiate themselves from the novel ones basically due to the lack of learning mechanisms. This paper proposes two new formulations of learning algorithms, the first one is based on the linear autoregressive model, while the second one is based on the MLP network. The obtained results indicate that the proposed algorithms present better overall performance than the classic ones.

1 Introduction

Voice over IP (VoIP) technology is becoming an important paradigm in today's portfolio of multimedia applications over the internet [1]. This is happening in such a very fast pace that it has drawn wide interest among both research and commercial communities alike. However, the Internet was not originally designed to replace the circuit switched networks that traditionally carry voice traffic over the public switched telephone network. To move through the Internet, user's continuous speech must be converted to IP packets. As a consequence, the statistical nature of data traffic and the dynamic routing techniques employed in packet-switched networks results in a varying network delay (jitter) experienced by IP packets, which can considerably degrade the quality of the service.

Technically speaking, jitter is the measure of the variability over time of the latency across a network. A widely used solution to alleviate the effects of jitter is to buffer the received audio packets before playing them out in the correct temporal order they were generated [2]. The playout of received audio packets from this buffer is postponed by a certain amount of time, to allow subsequent longer delayed packets to arrive at the receiver ahead of their scheduled playout times. The packets that still do not arrive within their delayed playout schedules are considered lost and are discarded.

The playout delay (or, more accurately, end-to-end application-to-application delay) is defined to be the difference between the playout time at the receiver and the generation time at the sender. If the playout delay in jitter buffer is increased then less packet are lost due to late arrival, but more delay is added to the voice call. A reduction in the playout delay turns out in less delay but more packet loss. The playout delay of the voice packets needs to be continuously

*The authors thanks CAPES for the financial support.

adapted in order to maintain an acceptable compromise between late packet loss and tolerable additional delay over the entire duration of the voice call.

The most commonly implemented solution for playout delay adaptation is suited for use in silence suppressed speech transmission scenarios, where the playout delay is set for individual talkspurts. Using an estimate of the network delay of upcoming voice packets, the playout delay is varied only at the beginning of a new talkspurt resulting in either compression or expansion of silent periods while the temporal structure of packets within a talkspurt is maintained intact.

Standard algorithms for playout delay prediction are based on simple descriptive statistics of the studied phenomenon, such as mean and standard deviation of the end-to-end delay during previous talkspurts. In this paper, we propose a novel formulation for the prediction of the playout delay for individual talkspurts and evaluate it using two time series models. The first one is based on the linear autoregressive (AR) model, while the second one is a nonlinear AR model implemented via an MLP network. The performances of the proposed models are compared with standard playout delay prediction algorithms.

Adapted from [3], the following notation will be used throughout this paper to describe the packet-audio stream. Figure 1 helps understanding the timing information of audio packets in a talkspurt.

- M: number of talkspurts in a given trace[1].
- t_k^i: sender timestamp of the i-th packet in the k-th talkspurt.
- a_k^i: receiver timestamp of the i-th packet in the k-th talkspurt.
- n_k: number of packets in the k-th talkspurt. Here, we only consider those packets actually received at the receiver.
- $\hat{d}_k^i$: delay between the generation of the i-th packet of the k-th talkspurt at the sender and its reception at the receiver, namely $\hat{d}_k^i = a_k^i - t_k^i$.
- $\hat{d}$: smallest network delay in a trace, i.e. $\hat{d} = \min_{1 \leq k \leq M, 1 \leq i \leq n_k}(\hat{d}_k^i)$.
- d_k^i: normalized delay of the i-th packet of the k-th talkspurt, i.e. $d_k^i = \hat{d}_k^i - \hat{d}$. This normalization is required to compensate for the asynchrony between the sender and receiver clocks.
- $d_k^{(i)}$: i-th smallest normalized delay in the k-th talkspurt.
- v_k^i: delay variation from the 1st to the i-th packet of the k-th talkspurt.
- $\hat{e}_k$: estimated excess delay for the k-th talkspurt. It is the amount of delay imposed by the buffer to the k-th talkspurt.
- $\hat{p}_k$: predicted playout delay for the k-th talkspurt. It is the total elapsed time between the emission of the k-th talkspurt and its execution at the receiver.
- p_k^i: time when the receiver plays out the i-th of the k-th talkspurt.
- alp: average lost packet rate in a trace.

The remainder of the paper is organized as follows. From Section 2 to 4 we present three classic algorithms for the prediction of the playout delay. In Section 5 we introduce the proposed approach based on AR time series models.

[1]Roughly speaking, a trace is a time series containing the actual network delays experienced by each packet.

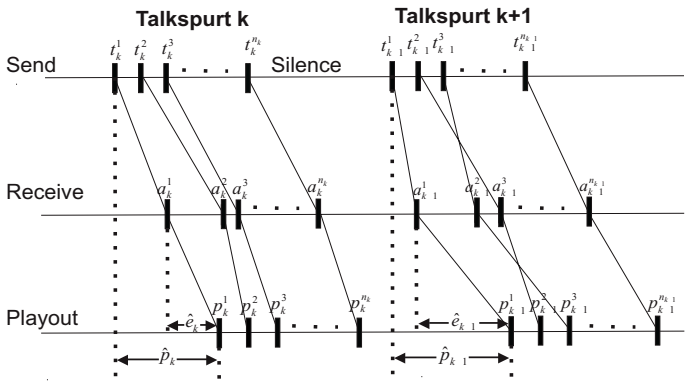


Fig. 1: Timings associated with the i-th packet in the k-th talkspurt.

The results of the performance evaluation of the classic and proposed algorithms is carried out in Section 6. The paper is concluded in Section 7.

2 Algorithm 1: Optimal Algorithm for a Single Talkspurt

This algorithm is actually a non-causal method to compute the optimal playout delay. It is non-causal because the playout delay is computed *after* the arrival of all packets of the k-th talkspurt. Thus, it is useful only as a reference for comparing the performances of other prediction algorithms.

Algorithm 1 allows the user to assess a posteriori which would have been the playout delay to be applied to the k-th talkspurt in order to ensure the loss of $n_k - i$ packets. So, the optimal playout delay is computed as

$$\hat{p}_k = \hat{d}_k^{(i)}, \quad \text{where} \quad i = n_k(1 - alp). \tag{1}$$

Once the value of $\hat{p}_k$ is computed, it is used to estimate the value of $\hat{e}_k$ as follows

$$\hat{e}_k = \hat{p}_k - d_k^f, \tag{2}$$

where d_k^f is the network delay of the 1st packet of the k-th talkspurt to arrive at the receiver. Then, the $\hat{e}_k$ is used as the excess delay for the (k+1)-th talkspurt.

3 Algorithm 2: Temporal Smoothing of Network Delays

This algorithm, proposed by [4], predicts the playout delay of the k-th talkspurt (i.e. $\hat{p}_k$) through the linear combination of the estimated values of the network delays of all packets in a trace and their corresponding estimated standard deviations. First, the network delays are estimated by the following recursive equations:

$$\hat{d}_k^i = \alpha \hat{d}_k^{i-1} + (1 - \alpha) d_k^i, \quad 2 \leq i \leq n_k, 1 \leq k \leq M, \tag{3}$$

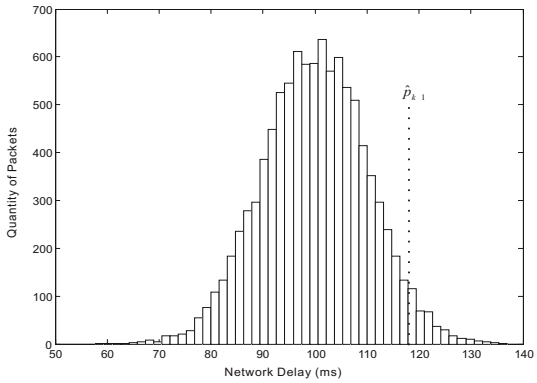


Fig. 2: Packet delay histogram for w packets.

and

$$\hat{d}_k^1 = \alpha \hat{d}_{k-1}^{n_{k-1}} + (1-\alpha) d_k^1, \quad 1 \leq k \leq M, \tag{4}$$

where $\hat{d}_k^i$ is the estimated network delay of the i-th packet of the k-th talkspurt, $\hat{d}_k^{n_k}$ is the estimated delay for the last packet of the k-th talkspurt and $\alpha = 0.998002$ is a constant weight responsible for the exponentially decaying memory of the algorithm. Second, the estimated delay variation for the i-th packet of the k-th talkspurt is computed as

$$\hat{v}_k^i = \alpha \hat{v}_k^{i-1} + (1-\alpha)|\hat{d}_k^i - d_k^i|, \quad 1 \leq i \leq n_k, 1 \leq k \leq M. \tag{5}$$

Finally, the predicted playout delay for the k-th talkspurt is given by

$$\hat{p}_k = \hat{d}_{k-1}^{n_{k-1}} + \beta \hat{v}_{k-1}^{n_{k-1}}, \quad 1 \leq k \leq M, \tag{6}$$

where the constant β is usually set to 4. Eq. (2) is also used to compute the excess delay $\hat{e}_k$.

4 Algorithm 3: Histogram-based Method

This algorithm was proposed by [5]. It functions by storing the network delays of w packets and building a histogram from them (Figure 2). In this paper we use $w = 150$. Once defined an acceptable alp (e.g. 0.05) for the problem, $\hat{p}_k$ is computed as the $100(1-alp)$-th percentile[2] of the packet delay distribution. Equation (2) is also used to compute the excess delay $\hat{e}_k$.

[2]The percentile of a distribution is a number z such that a percentage p of the population values are less than or equal to z. For example, the 75th percentile is a value (z) such that 75% of the values of the variable fall below that value.

5 Prediction via Times Series Models

A common feature of the algorithms to be described in this section is the use of learning or adaptive strategies for predicting the playout delay. By learning we roughly mean the ability of an algorithm to change its parameters according to the network conditions so that a better estimation of the playout delay is expected to be provided.

5.1 Algorithm 4: Proposed Model 1

It is commonly assumed that the $\hat{p}_k$ can be decomposed as

$$\hat{p}_k = J\mu(d_k) + L\sigma(d_k), \tag{7}$$

where J and L are, respectively, the weights associated with the mean ($\mu(d_k)$) and standard deviation ($\sigma(d_k)$) of the network delays for the k-th talkspurt. The values of $\mu(d_k)$ and $\sigma(d_k)$ are computed over fixed-length blocks of packets.

Let us further assume that the dynamics of $\hat{p}_k$ can be modeled by a linear autoregressive (AR) model of order n. Thus, we have

$$\hat{p}_k = \theta_1\hat{p}_{k-1} + \theta_2\hat{p}_{k-2} + \cdots + \theta_n\hat{p}_{k-n}, \tag{8}$$

where n denotes the length of the sliding window that includes the n most recent talkspurts previous to the current one.

If we substitute the definition in Equation (7) into Equation (8) we can write

$$\begin{aligned}\hat{p}_k \quad = \quad & \theta_1 J_1\mu(d_{k-1}) + \theta_1 L_1\sigma(d_{k-1}) + \theta_2 J_2\mu(d_{k-2}) + \theta_2 L_2\sigma(d_{k-2}) + \cdots \\ & + \theta_n J_n\mu(d_{k-n}) + \theta_n L_n\sigma(d_{k-n}).\end{aligned} \tag{9}$$

Since $\theta_i J_i$ and $\theta_i L_i$, $1 \leq i \leq n$, are also constants, we can rewrite Equation (9) as

$$\hat{p}_k = \theta_1^{\mu}\mu(d_{k-1}) + \theta_1^{\sigma}\sigma(d_{k-1}) + \theta_2^{\mu}\mu(d_{k-2}) + \theta_2^{\sigma}\sigma(d_{k-2}) + \cdots + \theta_n^{\mu}\mu(d_{k-n}) + \theta_n^{\sigma}\sigma(d_{k-n}). \tag{10}$$

We use the standard least-squares (LS) method for estimating the parameters of the model in Eq. (10). For this, considering a trace with M talkspurts and a sliding window of length n, we can write down Eq. (10) in the matrix form as follows

$$\mathbf{p} = \mathbf{X}\boldsymbol{\theta}, \tag{11}$$

where

$$\mathbf{X} = \begin{bmatrix} \mu(d_n) & \sigma(d_n) & \cdots & \mu(d_1) & \sigma(d_1) \\ \mu(d_{n+1}) & \sigma(d_{n+1}) & \cdots & \mu(d_2) & \sigma(d_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu(d_{M-2}) & \sigma(d_{M-2}) & \cdots & \mu(d_{M-n-1}) & \sigma(d_{M-n-1}) \\ \mu(d_{M-1}) & \sigma(d_{M-1}) & \cdots & \mu(d_{M-n}) & \sigma(d_{M-n}) \end{bmatrix}, \tag{12}$$

and

$$\boldsymbol{\theta} = [\theta_1^{\mu} \;\; \theta_1^{\sigma} \;\; \cdots \;\; \theta_n^{\mu} \;\; \theta_n^{\sigma}]^T \quad \text{and} \quad \mathbf{p} = [p_{n+1} \;\; p_{n+2} \;\; \cdots \;\; p_{M-1} \;\; p_M]^T, \tag{13}$$

where the superscript T denotes the transpose vector. The LS estimate of $\boldsymbol{\theta}$ is then given by

$$\hat{\boldsymbol{\theta}} = [\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{p}. \tag{14}$$

For predicting $\hat{p}_k$ we insert the estimated parameter vector $\hat{\boldsymbol{\theta}}$ into Eq. (10) and run this equation. As before, Equation (2) is used to compute the excess delay $\hat{e}_k$. In this paper we set $n = 2$ for this algorithm.

5.2 Algorithm 5: Proposed Model 2

In this algorithm we assume that dynamics of the playout delay is modeled by a nonlinear AR (NAR) model of order n. Thus, we have

$$\hat{p}_k = F(\hat{p}_{k-1}, \hat{p}_{k-2}, \ldots, \hat{p}_{k-n}), \tag{15}$$

where $F : \mathbb{R}^n \rightarrow \mathbb{R}$ is an unknown mapping. In this paper we use the multilayer Perceptron (MLP) network to approximate the mapping $F(\cdot)$. Thus, using the MLP and substituting Eq. (7) into the right-hand side of Eq. (15), it is straightforward to show that this equation can be re-written as

$$\hat{p}_k = G(\mu(d_{k-1}), \sigma(d_{k-1}), \mu(d_{k-2}), \sigma(d_{k-2}), \ldots, \mu(d_{k-n}), \sigma(d_{k-n})), \tag{16}$$

where $G : \mathbb{R}^{2n} \rightarrow \mathbb{R}$ is also an unknown nonlinear mapping. We approximate the nonlinear mapping $G(\cdot)$ through an MLP network with $2n+1$ inputs (including bias), one hidden layer with Q neurons, and one output neuron (see Figure 3). The hidden and output neurons use hyperbolic tangent activation functions. Weights and biases are randomly initialized in the range $[-0.5, 0.5]$ and adjusted through the standard gradient descent backpropagation algorithm with learning rate set to 0.05. For this algorithm, the memory order is set to $n = 5$. As shown in Figure 3, the output of the MLP provides an estimate of the playout delay for the k-th talkspurt, i.e. $O^{mlp}(k) = \hat{p}_k$. As always, Eq. (2) is used to compute the excess delay $\hat{e}_k$.

It is clear that compared with Algorithm 4, which is linear, Algorithm 5 is nonlinear. Besides this major difference between them, there are more two important ones. First, Algorithm 4 is trained in batch mode, while Algorithm 5 is trained in a pattern-by-pattern mode. Second, there is no training phase for Algorithm 5 in the usual neural network sense. This algorithm is used as an adaptive filter, i.e. there is no freezing of the weights and biases after a training period. In other words, once the MLP network is initialized, its output starts to predict the playout delay and its weights (and biases) are allowed to change in response to every input vector.

The proposed MLP-based NAR model for predicting the playout delay is similar to the neural network algorithm proposed in [6], differing from it basically in the definition of the neural network output. While the former requires only

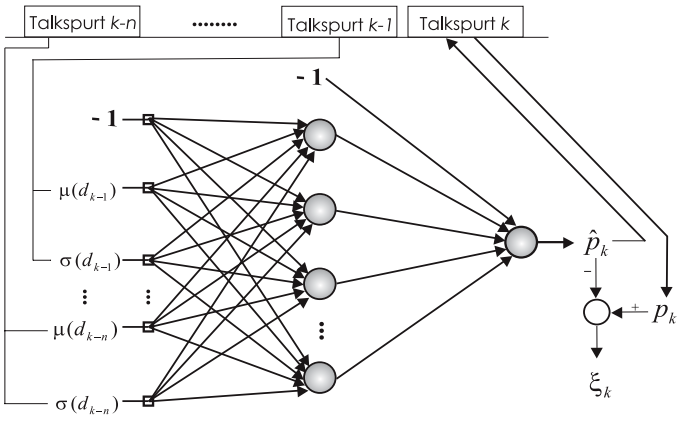


Fig. 3: MLP network topology used for predicting the playout delay.

one output since it predicts $\hat{p}_k$ directly, the latter requires two outputs, one for predicting the network delay ($\mu(d_k)$) for the k-th talkspurt and other for predicting its standard deviation ($\sigma(d_k)$). These outputs are then combined as in Eq.(7) to predict $\hat{p}_k$.

5.3 Algorithms 6 and 7: Elman and Jordan Recurrent Networks

Algorithms 6 and 7 are similar to Algorithm 5. The main differences among them are concerned with the input vector that each one processes. More specifically, for the k-th talkspurt, the input vector of the Algorithm 5 is defined as

$$\begin{aligned} \mathbf{x}(k) &= [x_0(k)\ \ x_1(k)\ \ x_2(k)\ \cdots\ x_{2n-1}(k)\ \ x_{2n}(k)]^T \\ &= [-1\ \ \mu(d_{k-1})\ \ \sigma(d_{k-1})\ \cdots\ \mu(d_{k-n})\ \ \sigma(d_{k-n})]^T. \end{aligned} \tag{17}$$

Hence, the corresponding input vectors for the Elman and Jordan recurrent network are defined, respectively, as follows

$$\mathbf{x}^e(k) = [\mathbf{x}(k) \mid y_1(k-1)\ \ y_2(k-1)\ \ \cdots\ y_Q(k-1)]^T \tag{18}$$

$$\mathbf{x}^j(k) = [\mathbf{x}(k) \mid x^c(k)]^T \tag{19}$$

where $y_i(k-1)$, $i = 1, 2, \ldots, Q$, are the previous outputs of the hidden neurons of the Elman network, and $x^c(k) = \gamma x^c(k-1) + O^{jordan}(k-1)$ is the context unit of the Jordan network, with $0 < \gamma < 1$ as the memory parameter and $O^{jordan}(k-1)$ as the previous output of the Jordan network. The Elman and Jordan recurrent networks are implemented with the same topology and training parameters as the MLP network of Algorithm 5.

6 Simulation Results

The performance evaluation of the seven algorithms previously described is carried out using the same six traces used by [3]. Their summary statistics, shown

Table 1: Summary statistics of the studied traces.

Trace	**Network delay** (ms)			
	Min	**Med**	**Max**	**Std**
1	0	210.03	2464	239.96
2	0	505.87	3200	357.27
3	0	181.88	844	75.98
4	0	48.92	1080	65.72
5	0	30.72	360	16.71
6	0	820.02	1540	123.65

in Table 1, indicate the presence of delay spikes in the time series, making the prediction task very challeging. Two metrics are used to quantify the performance of the algorithms, namely: (i) the average percentage of lost packets (alp) and (ii) the average total end-to-end delay (ted). They are chosen because the computation of the optimal playout delay is basically a trade-off between them. A low ted for a voice audio stream is desirable to the end user. However, a lower ted typically results in more lost packets due to late arrival. A decrease in the ted, therefore, typically causes an increase in the number of average lost packets. By the same token, a low alp is desirable to the end user since the conference quality of service may be affected if the vocoder cannot compensate accordingly. To obtain a lower alp, however, there is usually an increase in the ted.

In the computation of the alp we considered only the packets lost due to late arrival. We ignore packets dropped by the network due to congestion at routers and assumes that they are compensated for by the vocoder at the receiver. As pointed out previously, the Algorithm 1 is used only as a base line for performance comparison since it can not be used in practice (real-time prediction). The results for this algorithm were obtained for a pre-specified alp of 5% (i.e. alp=0.05). For all the evaluated neural network models, the number of hidden neurons is set to $Q = 5$. No normalization of the input data is carried out, since it is an on-line application. The memory orders for the Algorithms 4 and 5 are set to $n = 2$ and $n = 5$, respectively. Algorithms 6 and 7 (recurrent netowrks) use the same memory order of the Algorithm 5 (MLP).

The results for all the algorithms and all the traces are shown in Table 2. Some interesting conclusions can be drawn from this table. First, the best performance in average for the six traces is achieved by the Algorithm 4. Second, despite the fact that the neural network models (Algorithms 5, 6 and 7) did not achieved very good alp values when compared to the other algorithms, if we consider the compromise between a low alp and a low ted the neural network results are very good ones. Finally, considering only the Algorithms 5, 6 and 7, their results are very similar in terms of alp and ted, so there is no special advantage in using recurrent networks for this problem.

To emphasize the importance of jointly minimizing the alp and the ted, we build a table with two different performance rankings (see Table 3), averaged for the six traces. The first ranking considers only the performance evaluation

Table 2: Results for the six traces in terms of the TED and ALP metrics.

Algorithm	**Trace 1**		**Trace 2**	
	ted	**alp (%)**	**ted**	**alp (%)**
1	333.01	5.00	719.30	5.00
2	519.46	2.06	1251.30	1.51
3	406.04	6.61	819.50	13.75
4	328.03	1.81	549.51	6.14
5	362.52	2.64	763.90	13.71
6	362.13	2.74	764.35	13.75
7	363.72	2.60	764.74	13.75
Algorithm	**Trace 3**		**Trace 4**	
	ted	**alp (%)**	**ted**	**alp (%)**
1	259.76	5.00	89.60	5.00
2	361.23	1.61	143.22	3.01
3	278.43	8.07	121.87	4.62
4	254.14	2.86	149.43	1.24
5	294.33	4.42	137.00	1.64
6	294.51	4.46	136.60	1.64
7	295.05	4.45	140.23	1.77
Algorithm	**Trace 5**		**Trace 6**	
	ted	**alp (%)**	**ted**	**alp (%)**
1	48.63	5.00	881.00	5.00
2	80.17	1.47	1073.10	2.26
3	57.14	3.37	915.78	11.18
4	77.53	0.92	935.83	13.25
5	78.35	0.99	922.58	12.39
6	78.32	0.99	922.75	12.32
7	78.93	0.94	922.86	12.36

in terms of alp values. In this case, the Algorithm 2 is the best one. The second ranking considers the performance evaluation in terms of both alp and ted values. In this case, the best performance is achieved by the Algorithm 4, followed closely by the Algorithms 5, 6 and 7.

The good performance of the Algorithm 4 can be explained by the fact that it is trained in batch mode, while all the neural network models are trained as adaptive (online) filters. We also trained the MLP in batch mode, but the obtained results were inferior to those produced by the online-trained MLP. The Algorithm 4 can also be trained on-line through the LMS learning rule. We tested this alternative, but the results were very poor, confirming the results

Table 3: Ranking of the performance of the studied algorithms.

Criterion	**Performance**					
alp	**4**	**2**	**5**	**6**	**7**	**3**
alp+ted	**4**	**5**	**6**	**7**	**3**	**2**

of previous studies (e.g. see [7]). The online training mode seems to work for the MLP due to the presence of the derivative of the sigmoidal activation function in the generalized Delta rule used to update the weights. This derivative makes the adaptive filter less sensitive to sudden changes in the input signals as commonly occurs in VoIP applications in the form of delay spikes. Finally, it is worth mentioning a potential drawback of the Algorithm 4. The matrix inversion required by Eq. (14) can lead to numerical problems. In practice this can be solved through the use of Tikhonov regularization [8, ch.9]. This approach requires a regularization parameter which can be determined by cross-validation.

7 Conclusions

We introduced two novel approaches for the prediction of the playout delay for individual talkspurts. The first one was based on the linear AR model, while the second one was a nonlinear AR model implemented via an MLP network. The obtained results indicated that the proposed algorithms present better overall performance than the classic nonadaptive ones. We are currently trying to improve the performance of the Algorithm 5 (MLP) when trained in batch-mode. Other learning algorithms, such as Hidden Markov Models [9] and Support-Vector Machine [10], are also being evaluated.

References

[1] J. Davidson, J. Peters, M. Bhatia, S. Kalidindi, and S. Mukherjee. *Voice over IP Fundamentals*. Cisco Press, 2nd edition, 2006.

[2] W. A. Montgomery. Techniques for packet voice synchronization. *IEEE Journal on Selected Areas in Communications*, 1(6):1022–1028, 1983.

[3] S. B. Moon, J. Kurose, and D. Towsley. Packet audio playout delay adjustment: performance bounds and algorithms. *Multimedia Systems*, 6(1):17–28, 1998.

[4] R. Ramjee, J. Kurose, and D. Towsley. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of the 13th IEEE Annual Conference on Computer Communications (INFOCOM'94*, volume 2, pages 680–688, 1994.

[5] C. J. Sreenan, J. C. Chen, P. Agrawal, and B. Narendran. Delay reduction techniques for playout buffering. *IEEE Transactions on Multimidia*, 2(2):100–112, 2000.

[6] M. K. Ranganathan and L. Kilmartin. Neural and fuzzy computacional techniques for playout delay adaptation in VoIP networks. *IEEE Transactions on Neural Networks*, 16(5):1174–1194, 2005.

[7] A. Shallwani. An adaptive playout algorithm with delay spike detection for real-time voip. Master's thesis, McGill University, Department of Electrical & Computer Engineering, Montreal, Canada, 2003.

[8] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford university Press, 1996.

[9] T. Yensen, J. P. Lariviere, I. Lambadaris, and R. A. Goubran. HMM delay prediction technique for VoIP. *IEEE Transactions on Multimedia*, 5(3):444–457, 2003.

[10] K.-R. Müller, A. J. Smola, G. Rütsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Using support vector machines for time series prediction. In B. Schölkopf, C. J. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 243–253. MIT Press, 1999.

Automatic modelling of neural networks for time series prediction – in search of a uniform methodology across varying time frequencies

Nikolaos Kourentzes and Sven F. Crone

Lancaster University Management School, Department of Management Science
Bailrigg campus, Lancaster, LA1 4YX, United Kingdom

Abstract. In time series prediction, modelling neural networks poses multiple challenges in specifying suitable input vectors, network architectures, and training parameters depending on the underlying structure of the time series data. The data properties are often determined by the frequency in which the time series is measured, such as low frequency data of yearly, quarterly or monthly observations, or high frequency data of weekly, daily, hourly or even shorter time intervals. As different time frequencies require distinct modelling heuristics, employing neural networks to predict a set of time series of unknown domain, which may exhibit different characteristics and time frequencies, remains particularly challenging and limits the development of fully automated forecasting methodologies for neural networks. We propose a methodology that unifies proven statistical modelling approaches based upon filters and best practices from previous forecasting competitions into one framework, providing automatic forecasting without manual intervention by inferring all information from the data itself to model a diverse set of time series of varying time frequency, like the ESTSP'08 dataset.

1 Introduction

Artificial neural networks (NN) have found increasing consideration in forecasting research and practice, leading to successful applications in time series prediction and explanatory forecasting [1]. However, despite their theoretical capabilities of non-parametric, data driven approximation of any linear or nonlinear function directly from the dataset, NN have not been able to confirm their potential in forecasting competitions against established statistical methods, such as ARIMA or Exponential Smoothing [2]. As NN offer many degrees of freedom in the modelling process, from the selection of activation functions, adequate network topologies of input, hidden and output nodes, to learning algorithms and parameters and data pre-processing in interaction with the data, their valid and reliable use is often considered as much an art as science. Previous research indicates that the parsimonious identification of input variables to forecast an unknown data generating process without domain knowledge poses one of the key problems in model specification of NN [3, 4]. While literature provides guidance in selecting the number of hidden layers of a NN using wrapper approaches [5, 6], selecting the correct contemporaneous or lagged realisation of the dependent variable, and / or multiple explanatory variables, remains a challenge [7].

The issue of input variable and lag selection becomes particularly important, as the input vector needs to capture all characteristics of complex time series, including

the components of deterministic or stochastic trends, cycles and seasonality, interacting in a linear or nonlinear model with pulses, level shifts, structural breaks and different distributions of noise. While some components may be addressed in a univariate model using only lagged realisations of the dependent variable, others may require the integration of explanatory dummy-variables with adequate time-delays. Although a number of methodologies have been developed to support the valid and reliable identification of the input vector for NNs, they do not perform well consistently [8], there have been no comparative evaluations between them [4] and consequently there is currently no consensus on what methodology should be applied under which circumstances and time series frequency. Furthermore, it is argued [9, 10] that these methodologies to specify the input vector do not apply to high frequency data of weekly or higher frequency, like those datasets provided for the 2008 ESTSP competition. In addition to identifying a methodology to specify the input vector for a given time series frequency, this raises a more substantial challenge associated with the variety of modelling methodologies: the challenge of developing a valid and reliable methodology for a set of time series of different frequency, which ultimately prohibits the generation of a fully automated NN forecasting system.

To address this challenge, this paper suggests a methodology founded on established best practices from previous time series forecasting competitions for NN and proven statistical methods. The resulting approach can be applied automatically, without the need of manual intervention from a human expert, producing forecasts for sets of time series of unknown domain and different frequencies. Finally, through the necessary research that led to the development of this modelling methodology, a set of problems associated with modelling NN on high frequency data were encountered and explored. These are discussed in contrast to the challenges of modelling on low frequency time series, revealing the increasing complexity of high frequency data and pointing to potential for future research. The paper is organized as follows. First, we briefly introduce NN in the context of time series forecasting. Methodologies for selecting the input vector and the number of hidden nodes are also discussed. Section 3 presents the experimental design and the results obtained. A discussion of the problem arising from the transition from low to high frequency time series is done in section 4. Finally, we provide conclusions and future work in section 5.

2 Methods

2.1 Forecasting with multilayer perceptrons

Forecasting with NNs provides many degrees of freedom in determining the model form and input variables to predict a dependent variable $\hat{y}$. Due to the large degrees of freedom in modelling NN for forecasting, we present a brief introduction to specifying feedforward NN for time series modelling; a general discussion is given in [11, 12]. Through specification of the input vector of n lagged realisations of only the dependent variable y a feedforward NN can be configured for time series forecasting as $\hat{y}_{t+1} = f(y_t, y_{t-1}, \dots, y_{t-n+1})$, or by including i explanatory variables x_i of metric or nominal scale for causal forecasting, estimating a functional relationship of the form $\hat{y} = f(x_1, x_2, \dots, x_z)$. By extending the model form through lagged realisations of the

independent variables $x_{i,t-n}$ and dependent variable y_{t-n} more general dynamic regression and autoregressive (AR) transfer function models may be estimated. To extend the autoregressive model forms of feed-forward architectures to other stochastic processes, recurrent architectures incorporate moving average components (MA) of past model errors into the model, in analogy to the ARIMA-Methodology of Box and Jenkins [13]. Forecasting time series with NN is conventionally based on modelling a feed-forward topology in analogy to an non-linear autoregressive AR(p) model using a Multilayer Perceptron (MLP) [1, 14]. The architecture of a MLP of arbitrary topology is displayed in figure 1.

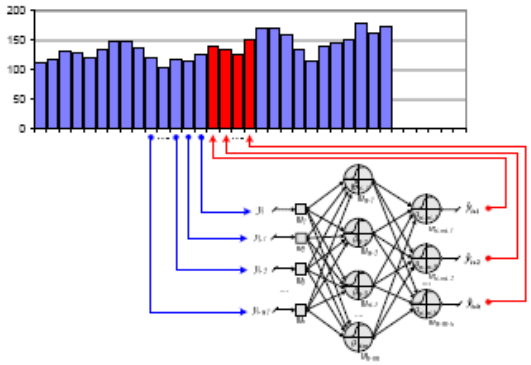


Fig. 1: Autoregressive MLP for time series forecasting

In time series prediction, at a point in time t a one-step ahead forecast $\hat{y}_{t+1}$ is computed using $p=n$ observations y_t, y_{t-1},…, y_{t-n+1} from n preceding points in time *t, t-1, t-2, …, t-n+1*, with n denoting the number of input units of the NN. Data is presented to the MLP as an overlapping set of input vectors formed as a sliding window over the time series observations. The task of the NN is to model the underlying generator of the data during training, so that a valid forecast is made when the trained NN is subsequently presented with a new input vector value [15]. The network paradigm of MLP offers extensive degrees of freedom in modelling for prediction tasks. Structuring the degrees of freedom, each expert must decide upon the selection and sampling of datasets, the degrees of data pre-processing, the static architectural properties, the signal processing within nodes and the learning algorithm in order to achieve the design goal, characterized through the objective function or error function. For a detailed discussion of these issues and the ability of NN to forecast univariate time series, the reader is referred to [1]. The specification of the input vector has been identified as being particularly crucial to achieving valid and reliable results followed by the specification of the number of hidden nodes [16, 17]. Both will be examined in the next section.

2.2 Input variable selection for multilayer perceptrons

The identification of relevant input variables and variable lags aims at capturing the relevant components of the data generating process in a parsimonious form. In time series modelling, it is closely related to identifying the underlying time series components of trend and seasonality and capturing their deterministic behaviour in

lags of the dependent variable. A simple visual analysis of the time series components frequently fails to reveal the complex interactions of autoregressive and moving average components, multiple overlying and interacting seasonality of different cycle lengths and nonlinear patterns. Several methodologies have been suggested for input variables selection of the significant lags in forecasting, most originating from linear statistics and engineering. However, there exists no uniformly accepted approach to identify linear or nonlinear input variables [4]. After reviewing the alternative methodologies suggested in literature for specifying the input vector of MLPs, the most widespread approach was found to be a form of stepwise regression [18-20]. The approach employs a conventional stepwise regression to identify the significant lags of the dependent variable and uses them as inputs for the MLP, with straightforward extensions of this approach for multivariate modelling [19]. Conventionally, the parametric approach of linear stepwise regression assumes a stationary time series, which must not be satisfied for trended or seasonal time series patterns. However, no consensus exists on whether a time series with identified trend should be detrended, and whether a seasonal time series should be deseasonalised first to enhance the accuracy of NN predictions [3, 21, 22]. Alternatively seasonality or trend be incorporated in the NN structure using additional model terms and explanatory variables [23-25]. As removing trending and / or seasonality prior to identifying significant lags may impact on the structure of the identified input vector, we evaluate three candidates of stepwise regression using (a) the original time series, (b) the detrended time series, and (c) deseasonalised versions of it. The resulting input vectors were different in structure and length, and were used as competing candidates to specify the input vector for the original, undifferenced time series.

A problem largely neglected but directly related to identifying significant lags from the time series is setting a maximum number of lags into the past the input vector should be explored for significance. The common practice involves the use of an arbitrary heuristic, e.g. using lags up to three seasons and hence 36 lags, or through an iterative trial and error process during modelling similar to the ARIMA-methodology. While both approaches may be feasible for low frequency data, they fail for high frequency time series where the large sample size for each lag induces low significance bounds. As a result most lags in the past become statistically significant and should be included in the model, although a particular seasonality may be best captured by including only the relevant lag of the true season. As the significance of lags further in the past does not fade away as with low frequency data, all lags up to an arbitrary maximum would be included, creating very large input vectors. Despite its universal relevance for NN, Regression and ARIMA-modelling, this issue has not been explored in literature, with the exception of one paper noting the issue in the context of forecasting low frequency time series with MLPs [26]. As a solution to determine the maximum lag number that is required for high-frequency time series, we propose a method based on the Euclidean distance of a seasonal year-on-year-plot. Assuming no prior seasonal information, the time series of length n is split into n / s 'seasons' of different length, with $s = \{2, 3, 4, \ldots, n / 2\}$, and the Euclidean distance between all observations across seasonal sub- series is calculated. The seasonal length s^* that minimises the Euclidean distance indicates the minimum possible deviation (in squared error terms) of the seasons in a seasonal plot, thus

providing an indication of seasonality and an upper limit of using 3*s as a maximum lag length. The global minimum identifies the strongest single seasonality, while local minima found in this sense reflect the minimum distances as seasonality increases, indicating seasonality or multiples of seasonality. The identified seasonality then provides relevant modelling information of single or multiple seasonality to be incorporated into the input vector [25], using the Euclidean distance. (For example, assuming a daily time series, which exhibits both day of the week and day of the year seasonality, the Euclidean distance will reveal both seasonalities, with the weekly being 7 observations and the annual being 365 observations.)

Regarding the selection of the number of hidden layers, theory regarding universal approximation [5, 6] suggests that one hidden layer is sufficient to invoke the universal approximation properties of the NN. Therefore, the question on specifying the network topology may be simplified to specifying the number of hidden nodes to include in the single layer. For time series of varying frequency, prior research indicates that a different number of hidden nodes maybe required depending on the pre-specified input vector of different length [27]. To reflect this, we employ a wrapper with a constant grid size to select the correct number of hidden units, which reflects the most popular approach to specifying the NN architecture [4].

3 Experimental design

3.1 Exploratory data analysis and input vector specification

The ESTSP 08 competition provided three time series without any information on the domain of origin nor on the time series frequency, which are displayed in fig.2. As each time series may contain different characteristics, they are explored using the Seasonal Euclidian distance and the Augmented Dickey-Fuller (ADF) test for trend.

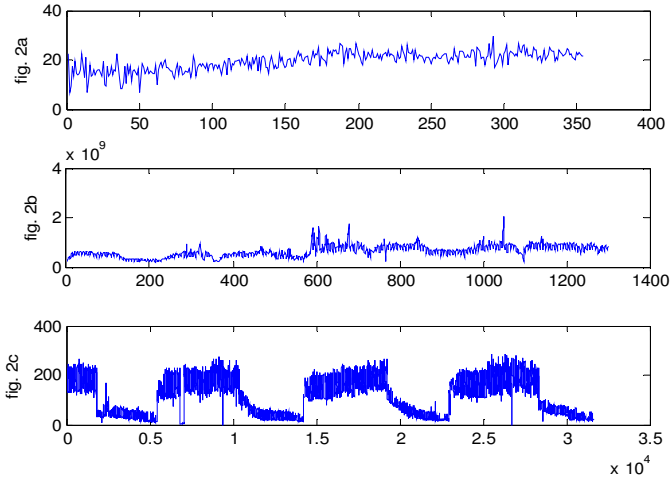


Fig. 2: The three ESTSP'08 time series

The first time series of the competition, plotted in fig. 2a, is comprised of 354 observations plus two explanatory time series to aid with the modelling of the time series. No domain knowledge on the time series is provided. The objective is to forecast the next 18 values. Applying the Euclidean distance approach a seasonality of 12-observations is identified; hence the time series is treated as monthly data containing 29.5 years of data. The abundance of data allows the use of three full seasons to identify the input vector. The ADF-test indicates the absence of trend, leaving only two options to model the input vector regarding pre-processing of the time series: both the original time series (1.a) and the time series after taking a 12^{th} order difference to remove the seasonality (1.b) is used. Note that the seasonal differenced time series is used only for the stationary identification of the input vector - the NNs are modelled only on the original time series.

The second time series of the competition contains 1,300 observations without any explanatory time series. The objective is to forecast the next 100 values. Using the ADF-test an instationary time series with trend is identified. However, careful visual inspection of the time series and the seasonal series plot indicates a structural break in the form of a single level shift, visible in fig. 2b, rather than a continuous trend. As a consequence no 1^{st} order differencing is required. Using the Euclidean distance approach seasonality of 7 and 365 observations are identified and we may infer that the time series contains daily observations. The significant lags are identified on the original time series (2.a), applying a 7^{th} order differencing (2.b), a 365^{th} order differencing (2.c) and both differences (2.d) in order to identify possible input vectors candidates.

The third time series of the competition, plotted in fig. 2c, contains 31,614 observations; the objective is to predict the 200 next values. The ADF-tests identifies no significant trend. Using the Euclidian distance approach we identify three potentially overlaying seasonalities of 24, 168 and 8,760 observations, indicating an hourly time series with hour of the day, day of the week, and day in the year seasonality. This provides several alternative input vectors by applying different levels of seasonal differencing, including the original series (3.a), the 24^{th} (3.b), the 168^{th} (3.c), and the $8{,}760^{th}$ differenced series (3.d), plus four combinations or the differences. All identified candidate input vectors will be constructed and evaluated in a set of NN candidates, which are specified in the next section.

3.2 **Artificial Neural Network models**

We construct a set of conventional MLPs using a consistent methodology, where all modelling parameters are identical but the choice of the input vector and the number of hidden nodes, as specified above. In addition to the lagged inputs of the dependent variable and possible explanatory time series identified through the stepwise regression analysis, where applicable, a set of additional inputs was as a set of candidates for all time series. A single integer variable was used to code a deterministic seasonality, in contrast to conventional s-1 binary dummies that substantially increase the size of the input vector. This was done in order to capture additional aspects of the seasonality in addition to the AR(p) terms modelled through

time lagged realisations, as suggested from previous studies [24]. Also, binary dummies were introduced to code level shifts for time series 2.

All MLPs apply a single output node with the identity activation function for a on-step-ahead prediction of $t+1$. Due to the possible interaction of the input vector size with the number of hidden nodes in a single hidden layer we evaluate different NN models for every input vector candidate using a stepwise grid-search with 2, 4, 6, 8, 10 and 12 hidden nodes to be considered for model selection. All hidden nodes for time series 1 and 2 apply a hyperbolic tangent as the activation function, while time series 3 uses the logistic activation function. This choice was made due to problems discovered during training of the 3rd time series, most probably due to the length of the time series resulting in a large number of training examples and the high degrees of freedom of the relevant neural network candidates. Each MLP is trained using simple back-propagation with momentum for 1,000 epochs or until an early stopping criterion is satisfied. For the early stopping criterion the mean squared error (MSE) is evaluated every epoch, and training is halted if no improvement was made for hundred epochs. The initial learning rate is set to $\eta=0.5$, applying a cooling factor $\Delta\eta$ to reduce the learning rate by 0.01 per epoch; the momentum term is kept constant at $\varphi=0.4$. All data is pre-processed using linear scaling into the interval of [-0.6, 0.6] and presented to the MLP using random sampling without replacement. Each MLP candidate is initialised 40 times with random starting weights in the interval of [-0.6, 0.6] in order to avoid local minima during the training and to provide an adequate error distribution using sufficient results.

3.3 **Model selection**

Given the large number of different alternative candidate models created, applying a different number of hidden nodes, input vectors and across the 40 initialisations used in training, model selection of the MLP candidate which promises the best out-of-sample performance on unseen data can be very challenging. The limited prior performance of NN, and, in particular, their low consistency and robustness of performance across homogeneous datasets in time series prediction [8] can in part be contributed to suboptimal model selection using a simple 1-fold cross validation. In contrast to selecting the best performing MLP candidate, we consider an ensemble of diverse candidates to generate average predictions. In addition to substantial evidence in classification that ensembles of simple methods perform well, this has long been confirmed for time series prediction, e.g. at the M competition, where a simple average of all competing methods performed better than each of the competing methods itself[2]. Based on this finding we rank all the MLP candidates for each time series, select the 10 best models and average their forecasts for each future horizon. These ensemble forecasts circumvent aspects of the challenges in model selection, however pose additional problems in evaluating different ensemble schemes. The ESTSP'08 competition assesses the accuracy of the models using a normalised mean squared error (NMSE) for each time series averaged over all three series. In order to align the performance metric for parameterisation and model selection with the final metric, a MSE proportional to the final metric was used during model development.

4 Experimental results

The composite ensemble forecasts for time series 1, 2 and 3 are given in fig. 3.

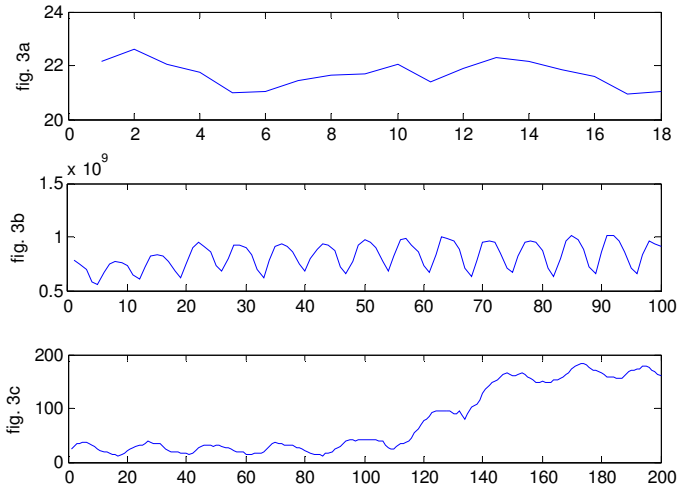


Fig. 3: Forecasts for the ESTSP'08 time series

Due to the large number of different MLP candidates evaluated and the space constraints it is infeasible to provide a comprehensive overview of the experimental results and architectures of the individual candidate models. Therefore we will restrict our discussion to some generalised findings: 80% of the top 10 candidate models (which were used to create the composite forecasts) uses an integer dummy variable to code each deterministic seasonality of different length, implying that this strategy aids the model to capture the complex overlying seasonal forms. For the candidate approaches for which the input vector was identified both on the original and the differenced time series, both model forms were always selected to be within the top 10 across all time series, implying that these approaches are complementary. An unexpected finding was that the univariate models for time series 1 outperformed all multivariate models using the two additional explanatory time series. This reduced the complexity of creating the final forecasts, as no predictions for the explanatory time series were required and no accumulation of the errors due to inaccurate forecasts of the explanatory variables could be introduced into the final forecasts of time series 1.

5 Challenges in modelling high frequency data

One elementary characteristic of high frequency time series data is the increase in length of the time series given a constant time interval, and the resulting increase in training vectors. For the frequencies employed in the ESTSP'08 competition the hourly time series would be 24 and 720 times longer than the daily and the monthly time series, had an identical time period been used. This difference in the sample size creates several challenges in the three frequency domains even though an identical

same modelling procedure is followed. The most important implications for this set of experiment, handling the degrees of freedom, input vector length in model identification and computational time, are outlined in Table 1 and discussed below.

Time Series	Frequency	Average no. of inputs	% Difference in inputs	Maximum time lag	% Difference in lag
1	Monthly	7	-	t-36	-
2	Daily	30	328.57%	t-392	988.89%
3	Hourly	354	4957.14%	t-9072	25100.00%

Table 1: Average number of inputs and maximum time lag per time series.

5.1 Degrees of freedom

This analysis employed several ways to identify the input vector for each time series. These derived from different options on performing seasonal differencing in the presence of a single or multiple seasonalities, or not. Across all number of input vectors determined for each candidate we compute the average number of inputs for each time series. The findings listed in table 1 exemplify the magnitude of the increase in both the size of the number of inputs used for time series of increasing frequency, and of the resulting increase of the degrees of freedom purely from the number of input nodes. This illustrates the increased complexity of training a MLP as the data frequency increases. Taking into consideration the number of hidden nodes, a candidate model developed for the hourly time series would use 2,478 parameters on average, in comparison to only 49 for the monthly time series. The implications this has for the training are apparent, as well as the difficulty of solving such a complex optimisation problem. Further interactions seem to exist also with alternative modelling choices: for time series 3 the architectures using a hyperbolic tangent activation function in the hidden layer could not be trained using backpropagation, as the optimiser could not cope with the degrees of freedom. This suggests the need for future research regarding NN topology, not only with regard to predictive accuracy but also with regard to robustness and consistency of the architecture.

5.2 Model identification

In addition to the increase in input vector size, our experiments identified a positive correlation between the frequency of the time series and the size of the search space required to find suitable input lags. This is again illustrated in table 1, where the maximum lag that was evaluated for each time series is provided. Not only does the input vector for time series of higher frequency increase in size, the maximum time lag to be considered also moves further into the past. Most methodologies to identify the input vector based upon wrappers, grid search, exhaustive random search, genetic algorithms and other meta-heuristics based on computational force are bound to encounter constraints in providing results in a reasonable time frame. In contrast, the filter approach based upon an iterative stepwise regression equally requires long computation times to identify the appropriate lags to use, proportional to the increase in the search space. On the other hand, filter approaches utilising the autocorrelation and the partial autocorrelation information of the time series are limited in their accuracy to provide useful information for model identification due to the increased

number of significant lags resulting from a growing sample size and tight confidence intervals. Consequently, modelling time series of higher frequencies requires the careful consideration of the trade-of between compute power, filter and wrapper based approaches.

5.3 Computational time

The regression approach employed here appears to be adequate for the time series frequencies in question, providing solid identification of the relevant time lags for forecasting in an acceptable time. However, computational time varied substantially, ranging from virtually instantaneous for the time series 1 and 2 to several hours for time series 3. Experiments for the first two time series were computed on a 2.2 GHz INTEL dual core processor with 3 GB of RAM, running 2-3 hours. For the third time series initial computations identified resource problems. As a consequence the experiments were computed on a high performance cluster with two dual core processors at 2.4 GHz with 10GB of RAM dedicated for this task, which required several days. It appears that experiments on high frequency data require additional computational power beyond the scope of normal personal computers, in particular for multiple architectures and model ensembles. Alternatively, these may provide the requirements for developing alternative training methods to perform well for large datasets under the current computational resources constraints.

6 Conclusions

This paper proposes an initial methodology for automatic modelling of time series with arbitrary time frequencies, seasonalities and trends, using the true ex ante predictions of the ESTSP'08 competition. The principle of the model is to compute competing candidate models of MLPs with different input vectors utilising varying temporal information on trends, stochastic and deterministic seasonality through autoregressive (AR) and / or integer dummy variables respectively. In order to omit the need of manual intervention we employ a composite ensemble forecast from the 10 best models on the in sample performance of each time series. Ways to avoid arbitrary modelling decisions are described, concerning the selection of the input vector, number of hidden layers and the hidden nodes. The proposed methodology, which is based on established tools and methods, manages to surpass the problems that trouble most neural network methodologies in literature when facing sets of time series of varying time granularity and frequency.

The analysis finishes with identifying some of the main problems encountered in the extension of the methodology towards high frequency data. Given the computational resources, high frequency data remain to be extremely demanding and limit the amount of ad-hoc experimentation. Unique problems arise that are beyond the scope of this paper, requiring further research. There is an apparent need to explore the possibility of training the MLPs in a way that the sheer amount of data will not require unreasonably long time and can cope with the increased degrees of freedom of the neural network models.

References

1. G. Zhang, B.E. Patuwo, and M.Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, 1998, pp. 35-62.
2. S. Makridakis, and M. Hibon, "The M3-Competition: results, conclusions and implications," *International Journal of Forecasting*, vol. 16, no. 4, 2000, pp. 451-476.
3. T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," *Management Science*, vol. 42, no. 7, 1996, pp. 1082-1092.
4. G.Q. Zhang, B.E. Patuwo, and M.Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, 1998, pp. 35-62.
5. K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators " *Neural Networks*, vol. 2, no. 5, 1989, pp. 359-366.
6. K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, 1991, pp. 251-257.
7. B. Curry, and P.H. Morgan, "Model selection in neural networks: Some difficulties," *European Journal of Operational Research*, vol. 170, no. 2, 2006, pp. 567-577.
8. J.S. Armstrong, "Findings from evidence-based forecasting: Methods for reducing forecast error," *International Journal of Forecasting*, vol. 22, no. 3, 2006, pp. 583-598.
9. C.W.J. Granger, "Extracting information from mega-panels and high-frequency data," *Statistica Neerlandica*, vol. 52, no. 3, 1998, pp. 258-272.
10. J.W. Taylor, L.M. de Menezes, and P.E. McSharry, "A comparison of univariate methods for forecasting electricity demand up to a day ahead," *International Journal of Forecasting*, vol. 22, no. 1, 2006, pp. 1-16.
11. C.M. Bishop, *Neural networks for pattern recognition*, Oxford University Press, 1995.
12. S.S. Haykin, *Neural networks : a comprehensive foundation*, Prentice Hall, 1999.
13. G.E.P. Box, and G.M. Jenkins, *Time series analysis: forecasting and control*, Holden-Day, 1970.
14. A. Lapedes, and R. Farber, "How neural nets work," *Neural Information Processing Systems*, D.Z. Anderson ed., American Institute of Physics, 1988, pp. 442-456.
15. G. Lachtermacher, and J.D. Fuller, "Backpropagation in Time-Series Forecasting," *Journal of Forecasting*, vol. 14, no. 4, 1995, pp. 381-393.
16. G.P. Zhang, B.E. Patuwo, and M.Y. Hu, "A simulation study of artificial neural networks for nonlinear time-series forecasting," *Computers & Operations Research*, vol. 28, no. 4, 2001, pp. 381-396.
17. G.P. Zhang, "An investigation of neural networks for linear time-series forecasting," *Computers & Operations Research*, vol. 28, no. 12, 2001, pp. 1183-1202.
18. N.R. Swanson, and H. White, "Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models," *International Journal of Forecasting*, vol. 13, no. 4, 1997, pp. 439-461.
19. M. Qi, and G.S. Maddala, "Economic factors and the stock market: A new perspective," *Journal of Forecasting*, vol. 18, no. 3, 1999, pp. 151-166.
20. C.M. Dahl, and S. Hylleberg, "Flexible regression models and relative forecast performance," *International Journal of Forecasting*, vol. 20, no. 2, 2004, pp. 201-217.
21. M. Nelson, T. Hill, W. Remus, and M. O'Connor, "Time series forecasting using neural networks: Should the data be deseasonalized first?," *Journal of Forecasting*, vol. 18, no. 5, 1999, pp. 359-367.
22. G.P. Zhang, and M. Qi, "Neural network forecasting for seasonal and trend time series," *European Journal of Operational Research*, vol. 160, no. 2, 2005, pp. 501-514.
23. S.F. Crone, "A new perspective on forecasting seasonal time series with artificial neural networks," *25th International Symposium on Forecasting*.
24. S.F. Crone, and N. Kourentzes, "Input variable selection for time series prediction with neural networks-an evaluation of visual, autocorrelation and spectral analysis for varying seasonality," *European Symposium on Time Series Prediction*, pp. 195-205.
25. B. Curry, "Neural networks and seasonality: Some technical considerations," *European Journal of Operational Research*, vol. 179, no. 1, 2007, pp. 267-274.
26. S.D. Balkin, and J.K. Ord, "Automatic neural network modeling for univariate time series," *International Journal of Forecasting*, vol. 16, no. 4, 2000, pp. 509-515.
27. K.P. Liao, and R. Fildes, "The accuracy of a procedural approach to specifying feedforward neural networks for forecasting," *Computers & Operations Research*, vol. 32, no. 8, 2005, pp. 2151-2169.

Long Term Time Series Prediction with Multi-Input Multi-Output Local Learning

Gianluca Bontempi
Machine Learning Group, Département d'Informatique
Faculté des Sciences, ULB, Université Libre de Bruxelles
1050 Bruxelles - Belgium
e-mail: gbonte@ulb.ac.be

Abstract. Existing approaches to long term time series forecasting are based either on iterated one-step-ahead predictors or direct predictors. In both cases the modeling techniques which are used to implement these predictors are multi-input single-output techniques. This paper discusses the limits of single-output approaches when the predictor is expected to return a long series of future values and presents a multi-output approach to long term prediction. The motivation for this work is the fact that, when predicting multiple steps ahead of a time series, it could be interesting to exploit the information that a future series value could have on another future value. We propose a multi-output extension of our previous work on Lazy Learning, called LL-MIMO, and we introduce an averaging strategy of several long term predictors to improve the final accuracy. In order to show the effectiveness of the method, we present the results obtained on the three training time series of the ESTSP'08 competition.

1 Introduction

A regular time series is a sequence of measurements φ^t of an observable φ at equal time intervals. Both a deterministic and a stochastic interpretation of the forecasting problem on the basis of historical dataset exist. The deterministic interpretation is supported by the well-known Takens theorem [13] which implies that for a wide class of deterministic systems, there exists a *diffeomorphism* (one-to-one differential mapping) between a finite window of the time series $\{\varphi^{t-1}, \varphi^{t-2}, \ldots, \varphi^{t-m}\}$ (*lag vector*) and the state of the dynamic system underlying the series. This means that in theory it exists a multi-input single-output mapping (*delay coordinate embedding*) $f : R^m \to R$ so that:

$$\varphi^{t+1} = f(\varphi^{t-d}, \varphi^{t-d-1}, \ldots, \varphi^{t-d-m+1}) \tag{1}$$

where m (*dimension*) is the number of past values taken into consideration and d is the lag time. This formulation returns a state space description, where in the m dimensional space the time series evolution is a trajectory, and each point represents a temporal pattern of length m.

The representation (1) does not take into account any noise component, since it assumes that a deterministic process f can accurately describe the time series. Note, however, that this is only a possible way of representing the time series phenomenon and that alternative representations should not be discarded a priori. In fact, once we assume that we have not access to an accurate model of the function f, it is reasonable to extend the deterministic formulation (1) to a statistical Nonlinear Auto Regressive (NAR) formulation [8]

$$\varphi^{t+1} = f\left(\varphi^{t-d}, \varphi^{t-d-1}, \ldots, \varphi^{t-d-m+1}\right) + w(t) \tag{2}$$

where the missing information is lumped into a noise term w. In the rest of the paper, we will then refer to the formulation (2) as a general representation of the time series which includes as particular instance the case (1).

The success of a reconstruction approach starting from a set of observed data depends on the choice of the hypothesis that approximates f, the choice of the order m and the lag time d. In this paper we will address only the problem of the modeling of f, assuming that the values of m and d are available a priori or selected by conventional model selection techniques. Good references on the order selection are given in [7, 16].

A model of the mapping (2) can be used for two objectives: *one-step* prediction and *iterated* prediction. In the first case, the m previous values of the series are assumed to be available and the problem is equivalent to a problem of function estimation. In the case of iterated prediction, the predicted output is fed back as an input to the following prediction. Hence, the inputs consist of predicted values as opposed to actual observations of the original time series. A prediction iterated for H times returns a *H-step-ahead* forecasting. Examples of iterated approaches are recurrent neural networks [17] or local learning iterated techniques [9, 12].

Another way to perform *H-step-ahead* forecasting is to have a model which returns a direct forecast at time $t+h$, $h = 1, \ldots, H$:

$$\varphi^{t+h} = f^h(\varphi^{t-d}, \varphi^{t-d-1}, \ldots, \varphi^{t-d-m+1})$$

Direct methods often require high functional complexity in order to emulate the system. In some cases the direct prediction method yields better results than the iterated one [16]. An example of combination of local techniques of integrated and direct type is provided by Sauer [15].

Iterated and direct techniques for multi-step-ahead prediction share a common feature: they model from historical data a multi-input single-output mapping where the output is the variable φ^{t+1} in the iterated case and the variable φ^{t+h} in the direct case, respectively. This paper advocates that when a very long term prediction is at stake and a stochastic setting is assumed, the modeling of a single-output mapping neglects the existence of stochastic dependencies between future values, (e.g. φ^{t+h} and φ^{t+h+1}) and consequently biases the prediction accuracy. A possible way to remedy to this shortcoming is to move from the modeling of single-output mapping to the modeling of multi-output dependencies. This requires the adoption of a multi-output technique where the predicted value is no more a scalar quantity but a vector of future values of the time series. If there are multiple outputs it is common, apart from some exceptions [11], to treat the prediction problem as a set of independent problems, one per output. Unfortunately this is not effective if the output noises are correlated as it is the case in a time series. The contribution of the paper is to present a simple extension of the Lazy Learning paradigm to the multi-output setting[5, 2]. Lazy Learning (LL) is a local modeling technique which is *query-based* in the sense that the whole learning procedure (i.e. structural and parametric identification) is deferred until a prediction is required. In previous works we presented an original *Lazy Learning* algorithm [5, 2] that selects automatically on a query-by-query basis the optimal number of neighbors. Iterated versions of Lazy Learning were successfully applied to multi-step-ahead time series prediction [4, 6]. This

paper presents instead a multi-output version of LL for the prediction of multiple and dependent outputs in the context of long term prediction.

2 Multi-step-ahead and multi-output models

Let us consider a stochastic time-series of dimension m described by the stochastic dependency

$$\varphi^{t+1} = f\left(\varphi^{t-d}, \varphi^{t-d-1}, \ldots, \varphi^{t-d-m+1}\right) + w(t) = f(X) + w(t) \tag{3}$$

where w is a zero-mean noise term and X denotes the lag vector

$$X = \{\varphi^{t-d}, \varphi^{t-d-1}, \ldots, \varphi^{t-d-m+1}\}$$

Suppose we have measured the series up to time t and that we intend to forecast the next H, $H \geq 1$, values. The problem of predicting the next H values boils down to the estimation of the distribution of the H dimensional random vector

$$Y = \{\varphi^{t+1}, \ldots, \varphi^{t+H}\}$$

conditional on the value of X. In other terms, the stochastic dependency (2) between a future value φ^t of the time series and the past observed values X induces the existence of a multivariate conditional probability $p(Y|X)$ where $Y \in \mathbb{R}^H$ and $X \in \mathbb{R}^m$. This distribution can be highly complex in the case of a large dimensionality m of the series and a long term prediction horizon H. An easy way to visualize and reason about this complex conditional distribution is to use a probabilistic graphical model approach. Probabilistic graphical models [10] are graphs in which nodes represent random variables, and the lack of arcs represent conditional independence assumptions. For instance the probabilistic dependencies which characterize a multi-step-ahead prediction problem for a time series of dimension $m = 2$, lag time $d = 0$ and horizon $H = 3$ can be represented by the graphical model in Figure 1. Note that in this figure, $X = \{\varphi^t, \varphi^{t-1}\}$ and $Y = \{\varphi^{t+1}, \varphi^{t+2}, \varphi^{t+3}\}$. This graph shows that φ^{t-1} has a direct influence on φ^{t+1} but only an indirect influence on φ^{t+2}. At the same time φ^{t+1} and φ^{t+3} are not conditionally independent given the vector $X = \{\varphi^t, \varphi^{t-1}\}$.

Any forecasting method which aims to perform multi-step ahead prediction implements (often in an implicit manner) an estimator of the highly multivariate conditional distribution $p(Y|X)$. The graphical model representation can help us in visualizing the differences between the two most common multi-step-ahead approaches, the iterated and the direct one.

The iterated prediction approach replaces the unknown random variables $\{\varphi^{t+1}, \ldots, \varphi^{t+H-1}\}$ with their estimations $\{\hat{\varphi}^{t+1}, \ldots, \hat{\varphi}^{t+H-1}\}$. In graphical terms this method models an approximation (Figure 2) of the real conditional distribution where the topology of conditional dependencies is preserved though non observable variables are replaced by their noisy estimators. The direct prediction approach transforms the problem of modeling the multivariate distribution $p(Y|X)$ into H distinct and parallel problems where the target conditional distribution is $p(\varphi^{t+h}|X)$, $h = 1, \ldots, H$. The topology of the dependencies of the original condition distribution is then altered as shown in Figure 3. Note

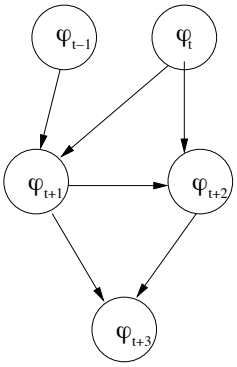


Fig. 1: Graphical modeling representation of the conditional distribution $p(Y|X)$ for $H = 3$, $m = 2$, $d = 0$

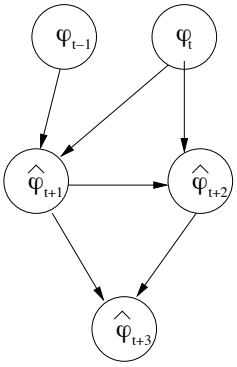


Fig. 2: Graphical modeling representation of the distribution modeled by the iterated approach in the $H = 3$, $m = 2$, $d = 0$ prediction problem.

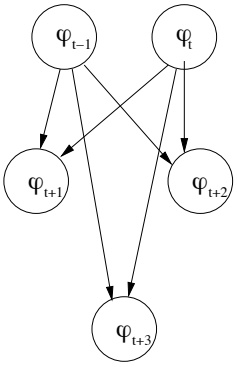


Fig. 3: Graphical modeling representation of the distribution modeled by the direct approach in the $H = 3$, $m = 2$, $d = 0$ prediction problem.

that in graphical model terminology this is equivalent to make a *conditional independence assumption*

$$p(Y|X) = p(\{\varphi^{t+1}, \ldots, \varphi^{t+H}\}|X) = \prod_{h=1}^{H} p(\varphi^{t+h}|X)$$

Such assumption is well known in the machine learning literature since it is exploited by the Naive Bayes classifier to simplify multivariate classification problems. Figures 2 and 3 visualize the disadvantages associated to the adoption of the iterated and the direct method, respectively. Iterated methods may suffer of low performance in long horizon tasks. This is due to the fact that they are essentially models tuned with a one-step-ahead criterion and therefore they are not able to take temporal behavior into account. In terms of bias/variance decomposition we can say that the iterated approach returns a non biased estimator of the conditional distribution $p(Y|X)$ since it preserves the dependencies between the components of the vector Y though it suffers of high variance because of the propagation and amplification of the prediction error.

On the other side, direct methods, by making an assumption of conditional independence, neglect complex dependency patterns existing between the variables in Y and consequently return a biased estimator of the multivariate distribution $p(Y|X)$.

In order to overcome these shortcomings, this paper proposes a multi-input multi-output approach where the modeling procedure does not target any more single-output mappings (like $\varphi^{t+1} = f(X) + w$ or $\varphi^{t+k} = f^k(X) + w$) but the multi-output mapping

$$Y = F(X) + W$$

where $F : \mathbb{R}^m \rightarrow \mathbb{R}^H$ and the covariance of the noise vector W is not necessarily diagonal or symmetrical [11]. The multi-output model is expected to return a multivariate estimation of the joint distribution $p(Y|X)$ and, by taking into account the dependencies between the components of Y, to reduce the bias of the direct estimator. However, it is worth noting that, in case of a large forecasting

horizons H, the dimensionality of Y is large too, and the multivariate estimation could be vulnerable to large variance. A possible countermeasure to such a side effect is the adoption of combination strategies, which are well reputed to reduce variance in case of low bias estimators. The idea of combining predictors is well known in the time series literature [15]. What is original here is that a multi-output approach allows the availability of a large number of estimators once the prediction horizon H is long. Think for example to the case where $H = 20$ and we want to estimate the value φ^{t+10}. A simple way to make such estimate more robust and accurate is to compute and combine several long term estimators which have an horizon larger than 10 (e.g. all the predictors with horizon between 10 and 20).

For multi-output prediction problems the availability of learning algorithms is much more reduced than in the single output case [11]. Most of existing approaches propose what is actually done by the direct approach, that is to decompose the problem into several multi-output single-output problems by making the assumption of conditional independence. What we propose here is to remove this assumption by using a multivariate estimation of the conditional distribution. For this purpose we adopt a nearest neighbor estimation approach where the problem of adjusting the size of the neighborhood (bandwidth) is solved by a strategy successfully adopted in our previous work on the Lazy Learning algorithm [5, 2].

3 A locally constant method for multi-output regression

We discuss here a locally constant multi-output regression method to implement a multi-step-ahead predictor. The idea is to return, instead of a scalar, a vector which smoothes the continuation of the trajectories which at time t resemble the most to the trajectory X. This method is a multi-output extension of the Lazy Learning algorithm [5, 2] and is referred to as LL-MIMO.

The adoption of a local approach to solve a prediction task requires the definition of a set of model parameters (e.g. the number of neighbors, the kernel function, the parametric family, the distance metric). In local learning literature different methods exist to automatically select the adequate configuration [1, 2] by adopting tools and techniques from the field of linear statistical analysis. One of these tools is the PRESS statistic which is a simple, well-founded and economical way to perform *leave-one-out* (l-o-o) cross-validation and to assess the performance in generalization of local linear models. By assessing the performance of each local model, alternative configurations can be tested and compared in order to select the best one in terms of expected prediction. This is known as the *winner-takes-all* approach in model selection. An alternative to the winner-takes-all approach was proposed in [5, 2] and consists in combining several local models by using the PRESS leave-one-out error to weigh the contribution of each term. This appears to particularly effective in large variance settings [3] as it is presumably the case of a stochastic multi-step-ahead task.

LL-MIMO extends the bandwidth combination strategy to the multi-output case where H denotes both the horizon of the long term prediction and the number of outputs. What we propose is a combination of local approximators with different bandwidths where the weighting criterion depends on the multiple

step leave-one-out errors e_h, $h = 1, \ldots, H$, computed over the horizon H.

In order to apply local learning to time series forecasting, the time series is embedded into a dataset D_N made of N pairs (X_i, Y_i), where X_i is a temporal pattern of length m, and the vector Y_i is the consecutive temporal pattern of length H.

Suppose the series is measured up to time t and assume for simplicity that the lag $d = 0$. Let us denote

$$\bar{X} = \{\varphi^t, \ldots, \varphi^{t-m+1}\}$$

the lag embedding vector at time t. Given a metric on the space $\mathbb{R}^m$ let us order increasingly the set of vectors X_i with respect to the distance to $\bar{X}$ and denote by $[j]$ the index of the jth closest neighbor of $\bar{X}$. For a given number k of neighbors the H step prediction is a vector whose hth component is the average

$$\hat{Y}_h^k = \frac{1}{k} \sum_{j=1}^{k} Y_h^{[j]}$$

where $Y^{[j]}$ is the output vector of the jth closest neighbor of $\bar{X}$ in the training set D_N. We can associate to the estimation $\hat{Y}_h^k$ a multi-step leave-one-error

$$E^k = \frac{1}{H} \sum_{h=1}^{H} e_h^2$$

where e_h is the leave-one-out error of a constant model used to approximate the output at the h step. In case of constant model the l-o-o term is easy to derive [3]

$$e_h = k \frac{Y_h^{[j]} - \hat{Y}_h^k}{k-1}$$

Though the optimal number of neighbors k is not known a priori, in [5, 2] we showed that an effective strategy consists in (i) allowing k to vary in a set $k_1, \ldots, k_b$ and (ii) returning a prediction which is the combination of the predictions $\hat{Y}_h^{k_i}$ for each bandwidth k_i, $i = 1, \ldots, b$. If we adopt as combination strategy the *generalized ensemble method* proposed in [14], we obtain that the outcome of the LL-MIMO algorithm is a vector of size H whose hth term is

$$\hat{\varphi}^{t+h} = \hat{Y}_h = \frac{\sum_{i=1}^{b} \zeta_i \hat{Y}_h^{k_i}}{\sum_{i=1}^{b} \zeta_i}, \qquad h = 1, \ldots, H \tag{4}$$

and the weights are the inverse of the multiple-step l-o-o mean square errors: $\zeta_i = 1/E^{k_i}$.

4 Experiments and final considerations

The LL-MIMO approach has been tested by applying it to the prediction of the three time series from the *ESTSP08 Competition*. The first time series (ESTSP1) has a training set of 354 three-dimensional vectors and the task is to predict the continuation of the third variable for $H = 18$ steps. The second time

series (ESTSP2) has a training set of 1300 values and the task is to predict the continuation for $H = 100$ steps. The third time series (ESTSP3) has a training set of 31614 values and the task is to predict the continuation for $H = 200$ steps.

The experimental session aims to compare the following set of methods on a long term prediction task (i) a conventional iterated approach (ii) a direct approach (iii) a multi-output LL-MIMO approach (iv) a combination of several LL-MIMO predictors (denoted by LL-MIMO-COMB) (v) a combination of the LL-MIMO and the iterated approach (denoted by LL-MIMO-IT).

In the strategy LL-MIMO-COMB the prediction at time $t + h$ is

$$\hat{\varphi}^{t+h} = \frac{\sum_{j=h}^{H} \hat{Y}_h^{(H_j)}}{H - h + 1},$$

where $\hat{Y}_h^{(H_j)}$ is the prediction of a multi-output LL-MIMO for an horizon $H_j \geq h$. In the strategy LL-MIMO-IT the prediction

$$\hat{\varphi}^{t+h} = \frac{\hat{Y}_h^{(H)} + \hat{Y}_h^{it}}{2}$$

where $\hat{Y}_h^{it}$ is the prediction returned by an iterated scheme. The rationale behind this two averaging methods is the reduction of the variance as discussed at the end of Section 2.

Note that in all the considered techniques the learner is implemented by the same local learning technique which combines a set of constant models whose number of neighbors range in the same interval $[5, k_b]$ with k_b parameter of the algorithm. In order to perform a correct comparison all the techniques are tested under the same conditions in terms of test intervals, embedding order m, values of k_b and lag time d. In detail

- the series ESTSP1 is used to assess the five techniques on the last portion of the training set of size $H = 18$, for values of m ranging from 5 to 20, for values of d ranging from 0 to 1 and and for k_b ranging from 10 to 25,
- the series ESTSP2 is used to assess the five techniques on the last portion of the training set of size $H = 100$, for values of m ranging from 5 to 35, for values of d ranging from 0 to 1 and for k_b ranging from 10 to 25,
- the series ESTSP3 is used to assess the five techniques on the last portion of the training set of size $H = 200$ for $m \in \{20, 50, 80, \ldots, 200\}$, for values of d ranging from 0 to 2 and for k_b ranging from 10 to 15.

Table 1 compares the average NMSE (Normalized[1] Mean Squared Error) prediction errors of the five techniques for the three datasets. The bold notation designs the technique which is significantly better than all the others (with 0.05 significativity level of the permutation test). Table 2 compares the minimum of the NMSE prediction errors attained by the five techniques over all the different configurations in terms of dimension m, lag time d and number k_b.

The experimental results show that for long term prediction tasks the LL-MIMO-COMB and LL-MIMO-IT strategies, i.e. the averaging formulations

[1]The normalization is done with respect to the variance of the entire series

Table 1: Average NMSE of the predictions for the three time series. The bold notation stands for significantly better than all the others at 0.05 significativity level of the paired permutation test.

Test data	LL-IT	LL-DIR	LL-MIMO	LL-MIMO-COMB	LL-MIMO-IT
ESTSP1	1.016	0.239	0.240	**0.219**	0.453
ESTSP2	0.426	0.335	0.335	0.326	**0.189**
ESTSP3	1.63e-2	1.05e-2	1.04e-2	**1.02e-2**	1.12e-2

Table 2: Minimum NMSE of the predictions for time series.

Test data	LL-IT	LL-DIR	LL-MIMO	LL-MIMO-COMB	LL-MIMO-IT
ESTSP1	0.228	0.171	0.172	0.1678	0.190
ESTSP2	0.188	0.130	0.125	0.115	0.104
ESTSP3	1.00e-2	0.96e-2	0.95e-2	0.88e-2	0.93e-2

of the LL-MIMO algorithm, can outperform conventional direct and iterated methods. LL-MIMO alone does not emerge as a competitive algorithm probably because of the excessive variance induced by the large dimensionality. The low biased nature of LL-MIMO however makes of this approach a good candidate for averaging approaches, as demonstrated by the good performance of LL-MIMO-COMB and LL-MIMO-IT. On the basis of these experiences we decided to submit to the Competition the LL-MIMO-IT prediction of the continuation of ESTP2, and the LL-MIMO-COMB prediction of the continuation of ESTP1 and ESTP3. A plot of the LL-MIMO-COMB prediction on the last portion of ESTP3 is illustrated in Figure 4.

We hope that the final validation provided by the Competition continuation series will confirm the importance of multi-output strategies in long term time series forecasting.

References

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5):11–73, 1997.

[2] M. Birattari, G. Bontempi, and H. Bersini. Lazy learning meets the recursive least-squares algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *NIPS 11*, pages 375–381, Cambridge, 1999. MIT Press.

[3] G. Bontempi. *Local Learning Techniques for Modeling, Prediction and Control*. PhD thesis, IRIDIA- Université Libre de Bruxelles, 1999.

[4] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for iterated time series prediction. In J. A. K. Suykens and J. Vandewalle, editors, *Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling*, pages 62–68. Katholieke Universiteit Leuven, Belgium, 1998.

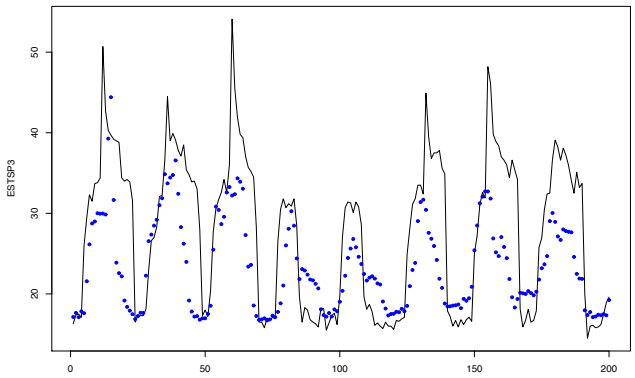


Fig. 4: ESTSP3: time series (line) vs. LL-MIMO-COMB prediction (dots).

[5] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for modeling and control design. *International Journal of Control*, 72(7/8):643–658, 1999.

[6] G. Bontempi, M. Birattari, and H. Bersini. Local learning for iterated time-series prediction. In I. Bratko and S. Dzeroski, editors, *Machine Learning: Proceedings of the Sixteenth International Conference*, pages 32–38, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[7] M. Casdagli, S. Eubank, J. D. Farmer, and J. Gibson. State space reconstruction in the presence of noise. *Physica D*, 51:52–98, 1991.

[8] J. Fan and Q. Yao. *Nonlinear Time Series*. Springer, 2005.

[9] J. D. Farmer and J. J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 8(59):845–848, 1987.

[10] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[11] J. M. Matias. Multi-output nonparametric regression. In *Progress in Artificial Intelligence*, pages 288–292, 2005.

[12] J. McNames, J. Suykens, and J. Vandewalle. Winning contribution of the k.u. leuven time-series prediction competition. *International Journal of Bifurcation and Chaos*, 1999. to appear.

[13] N. H. Packard, J. P. Crutchfeld, J. D. Farmer, and R. S. Shaw. Geometry from a time series. *Physical Review Letters*, 45(9):712–716, 1980.

[14] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman and Hall, 1993.

[15] T. Sauer. Time series prediction by using delay coordinate embedding. In A. S. Weigend and N. A. Gershenfeld, editors, *Time Series Prediction: forecasting the future and understanding the past*, pages 175–193. Addison Wesley, Harlow, UK, 1994.

[16] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, and A. Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 2007.

[17] R. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

Revisiting linear and non-linear methodologies for time series prediction - application to ESTSP'08 competition data

Madalina Olteanu

Universite Paris 1 - SAMOS CES
90 Rue de Tolbiac, 75013 Paris - France

Abstract. The goal of this paper is to remind some good sense principles that should be considered when modelling and forecasting time series. The use of more and more powerful computers lead to the developement of more and more prediction methods. Most of them are "black-box" algorithms that take the time-series in the entry and display a predicted value, without making any fundamental assumption such as stationarity. We shall use the data sets proposed for the competition to remind principles such as parcimony, preprocessing, significance or uncorrelated residuals.

1 Introduction

No doubt that time series prediction is a challenging topic in many fields, whether we speak of finance, energy consumption, wheather or internet. For most of these phenomena, there is a time-dependence structure which makes appealing an autoregressive approach : one uses the past to explain the present and predict the future.
Modelling and forecasting a time series supposes, in most of the cases, making some hypothesis on its behaviour. Since a time series contains noise, the main hypothesis is to assume that "there is random, but not too much". We shall translate this mathematically by "stationarity", which implies some regularities of the process and offers the frame for establishing asymptotic properties.
The pioneer autoregressive model (AR hereafter) of Yule [1] supposes that the dependence on the past is finite and linear and that the noise is Gaussian. Although largely used for quite a long time, this model could not provide a good fit in all cases. Most of the data exhibit nonlinearities such as volatility clustering, periodicity, ruptures or asymmetries which are not handled by AR models. A wide variety of more complex models, designed to overcome these drawbacks, have been proposed starting with the 80's. Let us recall some of the most popular : heteroscedastic conditional volatility models (ARCH [2], GARCH [3]), threshold or piecewise linear models [4], autoregressive regime switching models [5] or neural networks such as multilayer perceptron models [6].
Nowadays, more and more complex models became available in the neural networks community. In most of the cases, the statistical properties of these models are not established (most of the time due to their complexity), while modelling and prediction are performed by a "black-box" algorithm, with no assumption of

stationarity and no preprocessing. However, stationarity is an important issue and should be taken care of. On one hand, stationary processes are relatively easy to predict (the statistical properties will be the same in the future as they have been in the past) and, on the other hand, they provide meaningfull sample statistics.
The main goal of this paper is to use the opportunity of a forecasting competition to revisit the "classics" (AR and seasonal ARIMA models, multilayer perceptrons, hidden Markov models) and emphasize some points such as preprocessing, stationarity, parcimony, significant estimates and residuals independence. This paper doesn't seek for the best forecast on the three samples. It simply compares simple and complex models and shows that good use of simple models may provide better results than blind application of complex models.

2 Multivariate versus univariate models for Data1

The first data set contains three variables and 354 observations. The goal is to predict the next 18 values of the third component. Since the length of the series is not important, a parcimonious model will be needed. First, let us consider the autocorrelation function of the three dimensional process plotted in the left part of Fig 1. The first and the second components behave quite similarly. The autocorrelation functions are slowly decaying for all components and the process is subject to seasonality.
After having removed the seasonality in all components, the autocorelation function of the new process shows that the correlations between the third variable and the first two are not significant. This remark will be very useful in the sequel, since it means that a model based on the third variable alone may be considered, without loosing much information. If the selected model contained the first or the second variable, forecasting would be a more difficult task with an important propagation error (besides forecasting the third variable, one should need to predict the first two also).

The differenced series $Y_t = \nabla_{12} X_t$ was split into a training sample (318 inputs) and a test sample (24 inputs). Before estimation, the training sample was normalized. On the training sample, several classes of models were tested : an AR model for the third variable only, an ARX model considering all variables, a MLP model for the third variable and a MLP for the three variables. For every class of models, the "best" model was chosen with the BIC criterion.
Let us now make a comment concerning the AR models and the MLP models that will be important hereafter for model selection. MLP models are recognized as universal approximators and give very good short-term predictions. However, things change if we speak of long-term prediction. If the time-series to be modelled is stationary, a long-term prediction with an AR model will converge to the mean value of the process, but this property no longer holds for MLPs. The nonlinear function in the perceptron has attraction and/or repulsion points and the predictions will either converge to the attraction point, or oscillate between the repulsion points. This explains why in most of the present examples, the

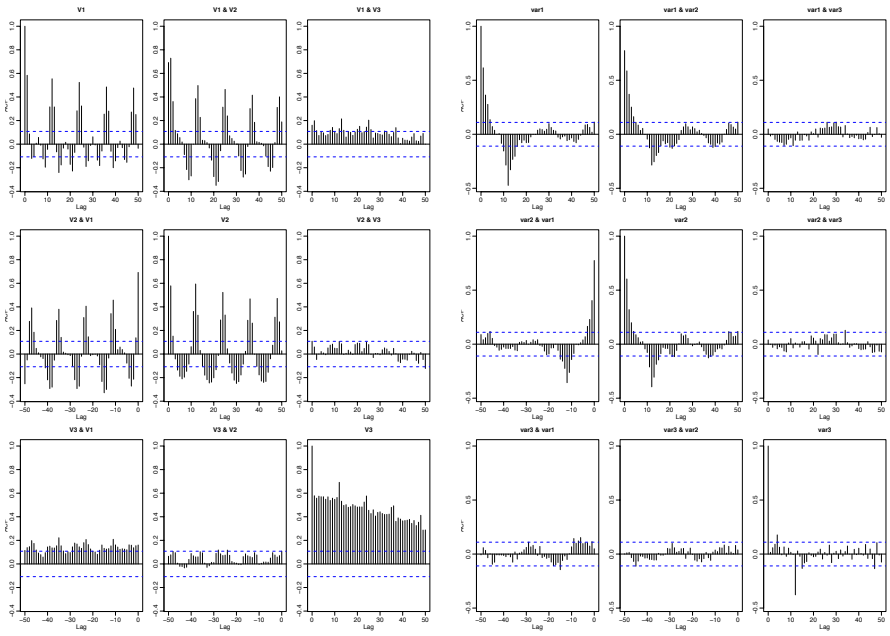


Fig. 1: ACF of Data 1 before and after preprocessing

AR models outperform the MLPs. For the first data set, the best two models in terms of normalized mean squared error (NMSE) measured on the test set are an AR(12) model for the third variable (NSME=0.26301) and a MLP for all variables (NSME=0.25624). The MLP has one hidden unit and takes as entries all variables are until lag 12. The predictions on the test set are represented in Figure 2.

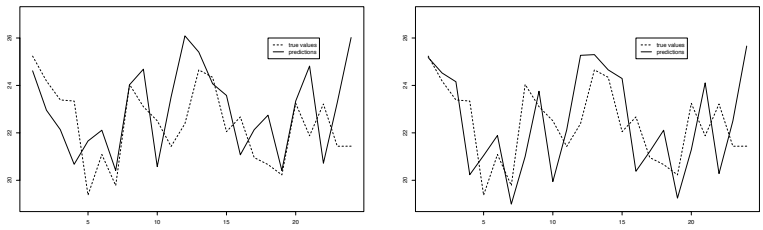


Fig. 2: Predictions on the test set - univariate AR model and multivariate MLP

Although the perceptron performs better as a predictor, it has two important drawbacks : the number of parameters (43 parameters against 12 for the AR model) and the residuals, which fail the independence test (Figure 3). Thus, although the forecasting results on the test set are quite similar, we prefer the

AR model for making the 18 predictions (Figure 4).

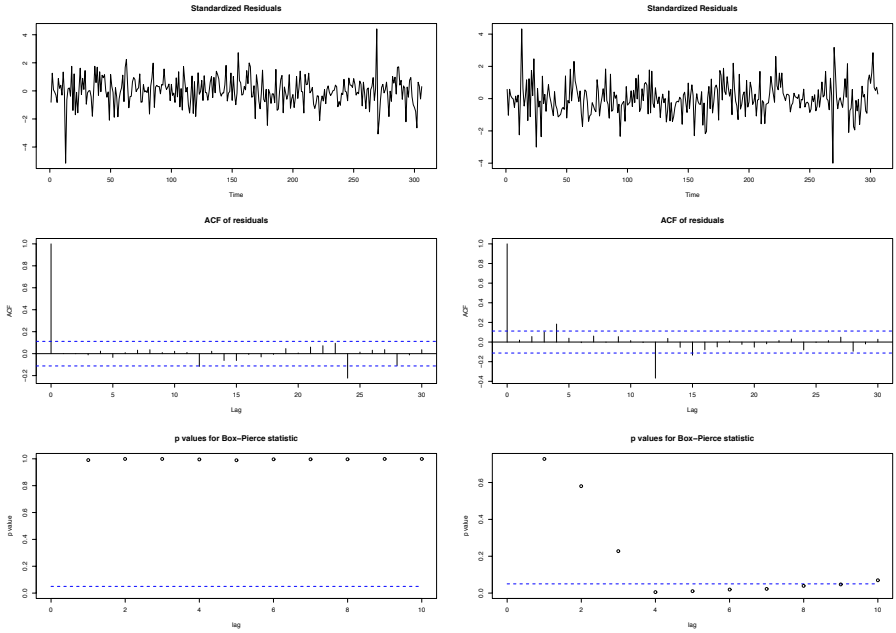


Fig. 3: Residuals - univariate AR model and multivariate MLP

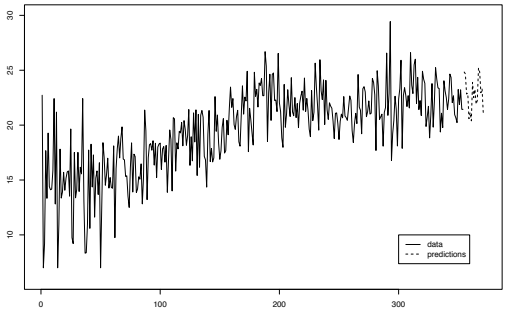


Fig. 4: Predictions - univariate AR model

3 Seasonal ARIMA models versus multilayer perceptrons (MLP) for Data2

The challenge for the second data set is to predict the next 100 values for a series of length 1300 (Figure 5). First of all, let us remark that the data are highly nonstationary and nonnegative with values of order 10^8. The autocorrelation function is slowly decaying and shows a periodicity of order 7.

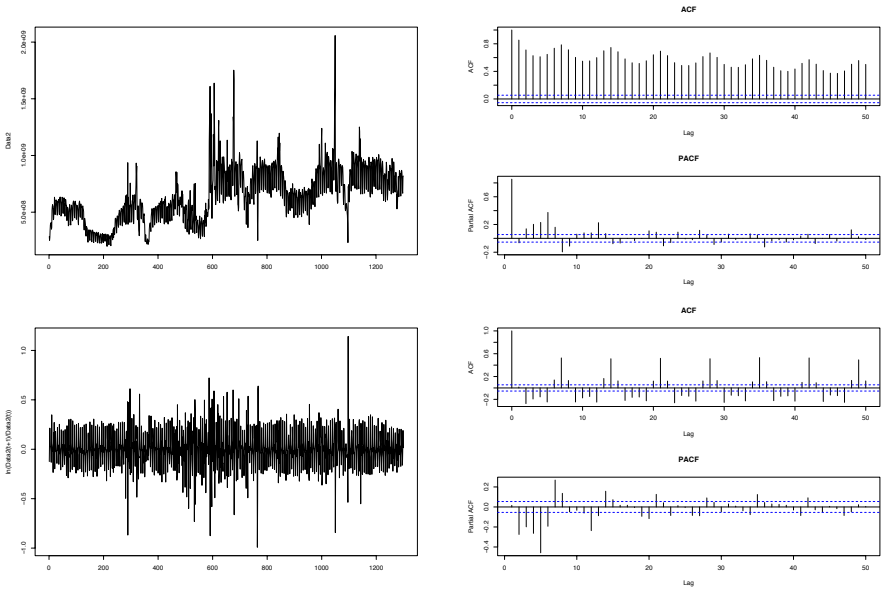


Fig. 5: Time series 2 before and after preprocessing

Modelling this data without preprocessing may lead to computation problems (saturation of the sigmoid functions in multilayer perceptrons, for example) or models that are not robust (nonstationarity). A transform of the data seemed then necessary : the natural logarithm and a first-order difference were considered. If X_t, $t \in \{1, ..., 1300\}$ are the initial data, we decided to study $Y_t = \ln\left(\frac{X_t}{X_{t-1}}\right)$. The transformed series is split into a training sample (1199 inputs) and a test sample (100 inputs). The data in the training sample was normalized before estimation. Two classes of models were competing in this example, seasonal ARIMA models and MLPs. For every class, the "best" model was selected with a BIC criterion. The final SARIMA model is the following :

$$
\begin{aligned}
\phi(B)\left(1-B^7\right)Y_t &= \theta(B)\,\Theta\left(B^7\right)\epsilon_t \\
\phi(z) &= 1+1.128z+0.655z^2+0.027z^3-0.501z^4-0.504z^5 \\
\theta(z) &= 1+0.833z+0.182z^2-0.43z^3-0.741z^4 \\
&\quad -0.483z^5+0.219z^6+0.03z^7 \\
\Theta(z) &= 1-0.885z-0.102z^2\ , \qquad (1)
\end{aligned}
$$

where B is the first-order difference operator $BY_t = Y_{t-1}$. The residuals of the SARIMA model are plotted in Figure 6. The autocorrelation function and the Llung-Box statistics validate the white-noise hypothesis. The forecasts on the test set are also represented in Figure 6. The top right graph contains the predicted values of the transformed series $\widehat{Y}_{t+h}$, while the bottom right graph gives the predictions for the initial data $\widehat{X}_{t+h}$ computed as follows :

$$
\widehat{X}_{t+h} = X_t \exp\left(\sum_{i=1}^{h} \widehat{Y}_{t+i}\right) \qquad (2)
$$

Predictions are quite accurate for the first 20 values, while for the rest, the true values are overestimated. On the log-returns series, the prediction error is NSME=0.02131, while on the initial series, NSME=0.30534.

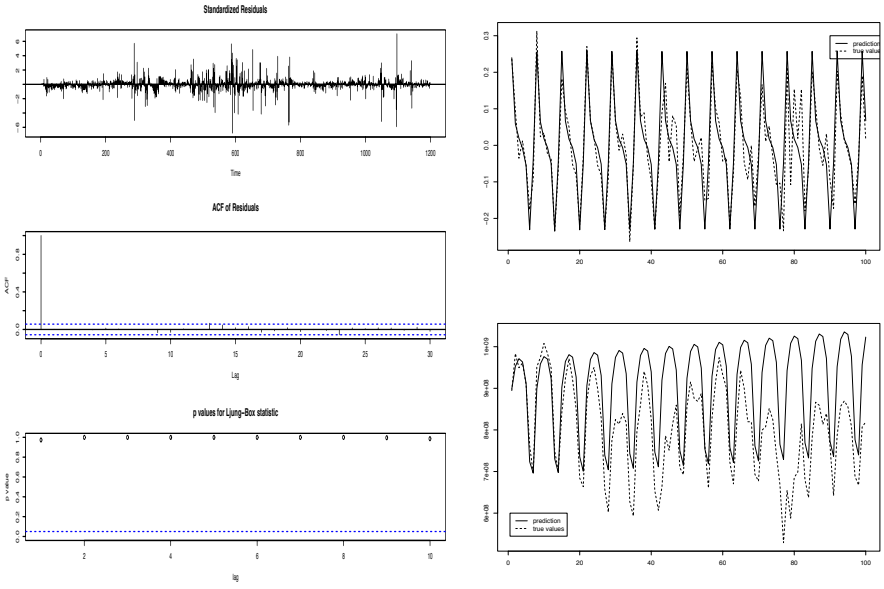


Fig. 6: Residuals and predictions on the test set - SARIMA model

Now, let us compare the SARIMA results with the MLP. As shown in Figure 7, the residuals of the MLP model fail the independence test, while the predicting

values are largely overestimating the true values. For illustration, the top right corner contains the predictions made by the SARIMA model, while the bottom right graph represents the MLP predictions. In terms of mean squared error for the MLP, NSME=2.92917. Since the SARIMA model outperforms the MLP in terms of prediction and robustness, we used it for predicting the 100 awaited values of the series (Figure 8).

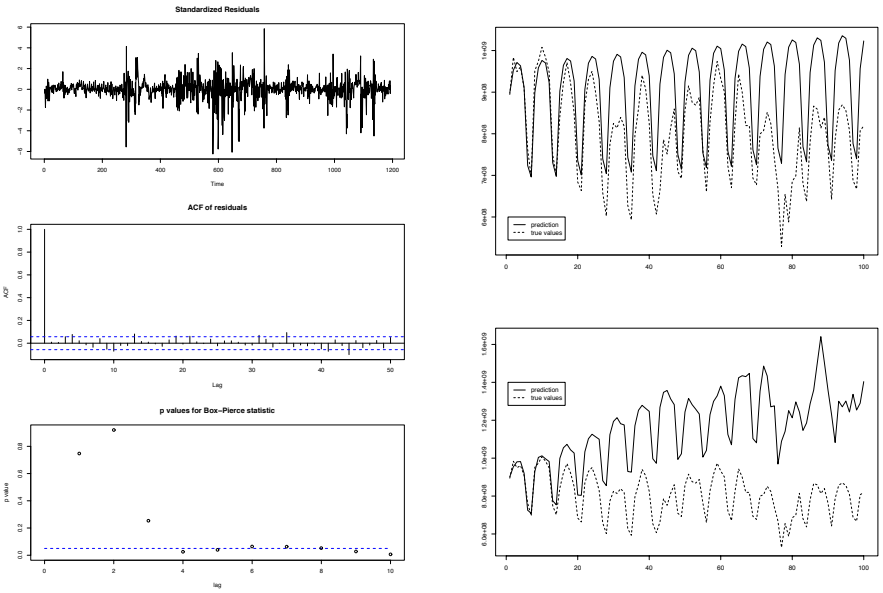


Fig. 7: Residuals - MLP model

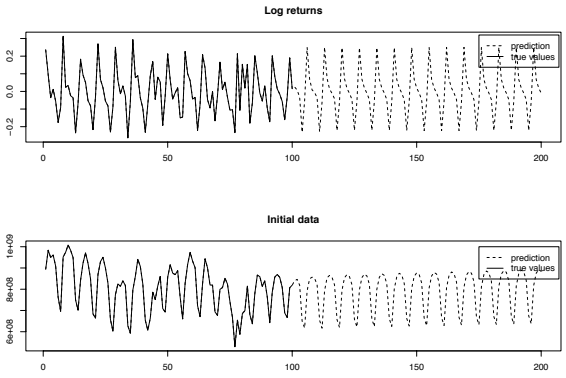


Fig. 8: 100 predicted values

4 Autoregressive Markov-switching models versus AR models for Data3

The third data set is highly nonstationary and contains over 31000 inputs. The goal is to forecast the next 200 values. The series plotted in Figure 9 suggests the possible existence of two regimes with quite opposite behaviours (increasing and decreasing trends). Also, the very slow decay of the autocorrelation function confirms the nonlinearity and the nonstationarity of the series.
A closer look at the autocorrelation function shows a double periodicity, of orders 24 and 168. This may lead to think that the data are hourly recordings of some phenomenon, showing daily and weekly seasonality. If we make the hypothesis of some hourly recorded data, the plot of the series suggests also the existence of yearly periodicity. We did not consider it in our study, but it should be interesting to see whether taking the differenced series of order 8760 improves the results.

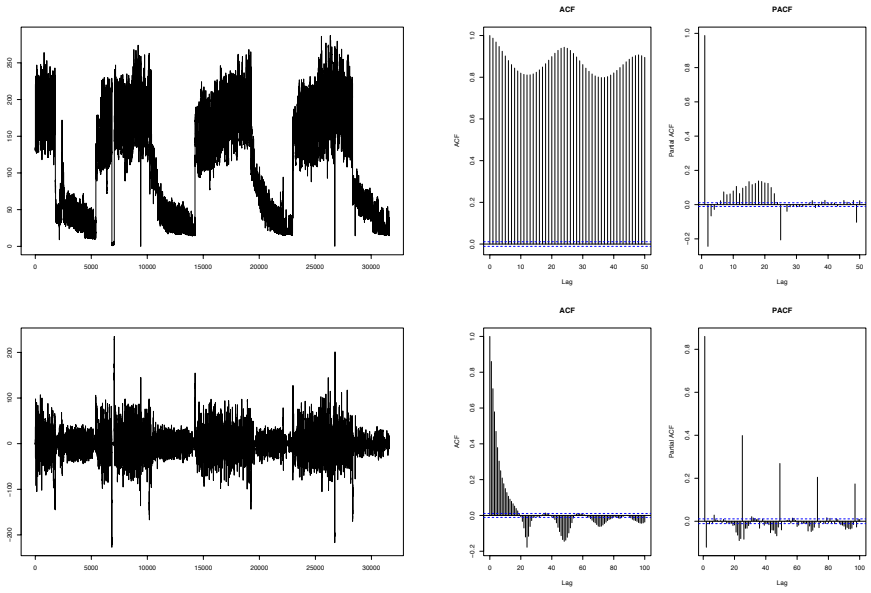


Fig. 9: Time series 3 before and after preprocessing

The periodicity of order 24 was removed from the initial series and the resulting data was split into a training sample (31000 inputs) and a test sample (200 inputs). Before estimation, the training sample was normalized. Since we suspected the existence of two regimes, two classes of models were tested : autoregressive Markov-switching models (hereafter, HMM) and AR models. The BIC criterion selected 168 lagged variables for both models. On the test set, the NSME is 0.01242 for the HMM model and 0.01249 for the AR model (Figure 10). Once again, as in the previous example, although the more complex model

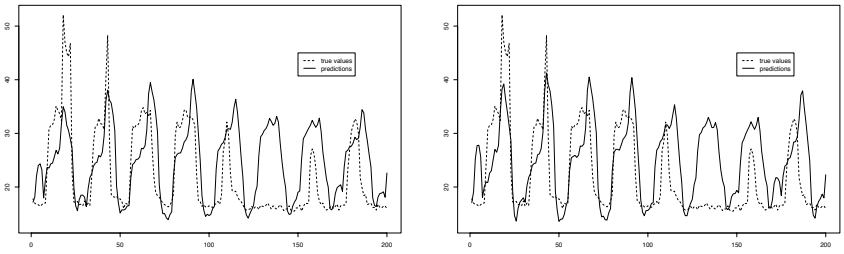


Fig. 10: Predictions on the test set - HMM and AR model

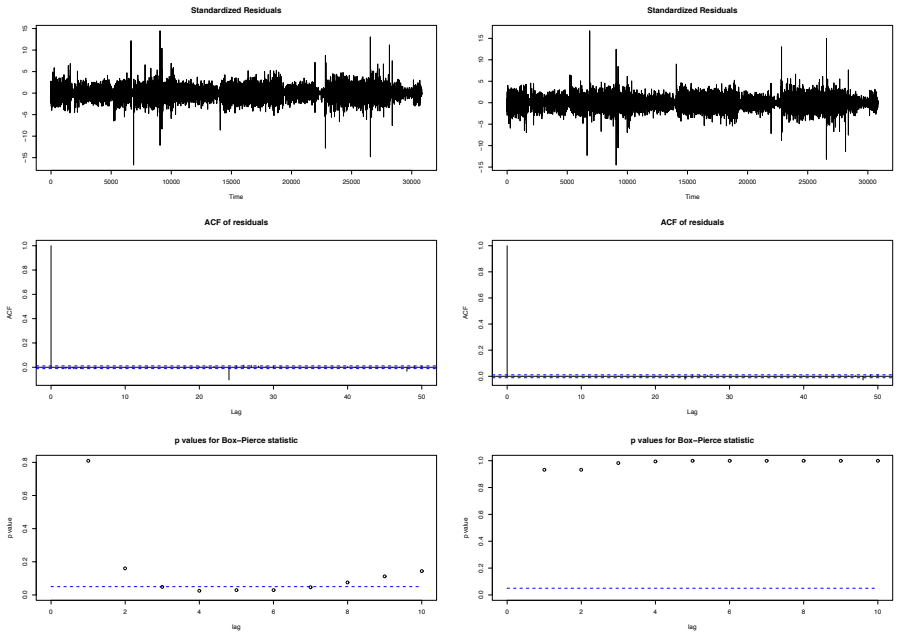


Fig. 11: Residuals - HMM and AR model

is slightly better in forecasting, the residuals fail the white noise tests and the AR model seems better (Figure 11). The demanded 200 forecasts are plotted in Figure 12.

5 Conclusion

Three classes of models (AR and seasonal ARIMA, MLP, HMM) were compared on the three data sets of the competition. The comparison was made in terms of parcimony, quality of predictions and residuals. All proposed time series are

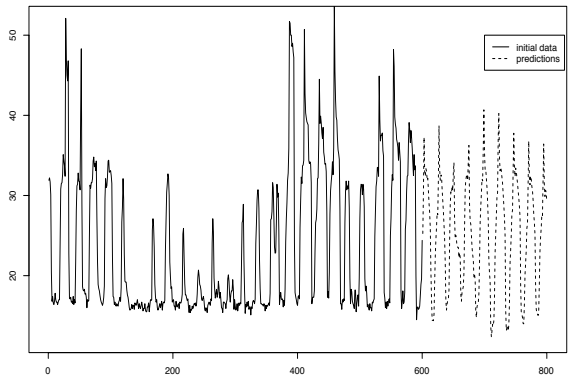


Fig. 12: Predictions - AR model

nonstationary with important seasonal components. Seasonal ARIMA models generally outperformed multilayer perceptrons and hidden Markov models in terms of long-term prediction. This is mainly explained by the property of AR models of converging to the mean value of the series, while the other models "get stucked" in some attraction point and lead to the spread of the prediction error. For this reason, long-term prediction remains a difficult problem which complex models do not always solve, but its difficulty may be lessened by preprocessing.

References

[1] U. Yule, On a method of investigating periodicities in disturbed series with special reference to Wolfers's sunspot numbers, *Philos. Trans. Royal. Soc. London, Series A*, 226:267-298, 1927.

[2] R.F. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of U.K. inflation, *Econometrica*, 50:987-1008, 1982.

[3] T. Bollerslev, Generalized autoregressive conditional heteroscedasticity, *J. Econ.*, 31:307-327, 1986.

[4] H. Tong *Threshold models in nonlinear time series analysis*, Lecture Notes in Statistics 21, Springer, Heidelberg, 1983.

[5] J.D. Hamilton, A new approach to the economic analysis of nonstationary time series and the business cycle, *Econometrica*, 57:357-384, 1989

[6] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Networks*, 4:359-366, 1989

Using reservoir computing in a decomposition approach for time series prediction.

Francis wyffels, Benjamin Schrauwen and Dirk Stroobandt *

Ghent University - Electronics and Information Systems Department
Sint-Pietersnieuwstraat 41, 9000 Gent - Belgium

Abstract. In this paper we combine wavelet decomposition and recurrent neural networks to provide fast and accurate time series predictions. The original time series is decomposed by means of wavelet decomposition into a hierarchy of time series which are easier to predict. The prediction core of our solution is given by reservoir computing, which is a recently developed technique for the very fast training of recurrent neural networks. The three time series of the ESTSP 2008 competition will be used as an illustration for our method.

1 Introduction

Forecasting is a domain with a broad range of useful applications. Therefore, researchers working on time series prediction come from a wide variety of fields and are using many methods such as theta method [1], support vector machines, neural networks, local modeling [2], wavelet-decomposition [3] and many more.

This year, for the second time, the *European Symposium on Time Series Prediction* is held. This symposium always presents a challenging competition in the domain of time series prediction. This year the competition concerns the prediction of three different time series which can be found on the website `www.estsp.org`. Each time series has different properties which is interesting because this will reveal the strength and weaknesses of the many methods that are applied on the time series during the contest.

Although we have no profound experience in the domain of time series prediction, we wanted to join this competition in order to compare our method with many others in the forecasting domain. In this paper we describe how a combined approach of wavelet decomposition and reservoir computing can be used for forecasting. Before we continue the full explanation of our method we give a short overview of reservoir computing which will be the baseline of our method.

*This research is partially funded by FWO Flanders project G.0317.05 and the Photonics@be Interuniversity Attraction Poles program (IAP 6/10), initiated by the Belgian State, Prime Minister's Services, Science Policy Office.

2 Reservoir computing: a short overview

Reservoir computing is a novel technique for the fast training of large recurrent neural networks which have been successfully applied in a broad range of temporal tasks such as robotics [4], speech recognition [5, 6] and time series generation [7]. Last year, reservoir computing outperformed all other methods in the NN3 competition for financial time series prediction [8].

The reservoir computing technique is based on the use of a large untrained dynamical system, the reservoir, where the desired function is implemented by a linear memory-less mapping from the full instantaneous state of the dynamic system to the desired output. Only this linear mapping is learned. When the dynamical system is a recurrent neural network of analog neurons the method is referred to as echo state networks [7]. When spiking neurons are used, one often speaks of liquid state machines [9]. But both are now commonly referred to as reservoir computing [10].

Training is done in a supervised way by first driving the reservoir with teacher forced inputs and/or teacher forced output feedback. Secondly, the output layer is trained by using linear regression methods. This is summarized by the following equations:

$$\begin{aligned} \mathbf{x}[k+1] &= f\Big(W^{\text{res}}_{\text{res}}\mathbf{x}[k] + W^{\text{res}}_{\text{inp}}\mathbf{u}[k] + W^{\text{res}}_{\text{out}}\mathbf{y}[k] + W^{\text{res}}_{\text{bias}}\Big) \\ \widehat{\mathbf{y}}[k+1] &= W^{\text{out}}_{\text{res}}\mathbf{x}[k+1] + W^{\text{out}}_{\text{inp}}\mathbf{u}[k] + W^{\text{out}}_{\text{bias}}, \end{aligned} \tag{1}$$

where $\mathbf{x}[k]$ is the reservoir's state, $\mathbf{u}[k]$ is the input, $\mathbf{y}[k]$ is the desired output and $\hat{\mathbf{y}}[k]$ is the actual output. When analog neurons are used, the nonlinearity f often represents the sigmoid function. All the weight matrices denoted by $W^{\text{out}}_{\star}$ are trained, while those denoted by $W^{\text{res}}_{\star}$ are fixed and randomly created. During testing, the teacher forced output feedback $\mathbf{y}[k]$ is replaced by the actual output $\hat{\mathbf{y}}[k]$ which we call free run mode.

Because only the output weights are changed, training can be realized very quickly which can be an additional benefit in comparison with other methods. Additionally, reservoir computing doesn't suffer from local optima like other methods based on neural networks do. We conclude that reservoir computing gives us a powerful tool that can be easily used in a broad range of applications.

3 Time series prediction

Now we have introduced the baseline of our prediction mechanism we can formalize our overall prediction scheme which is illustrated in Fig. 1. We present now each of the modules in greater detail.

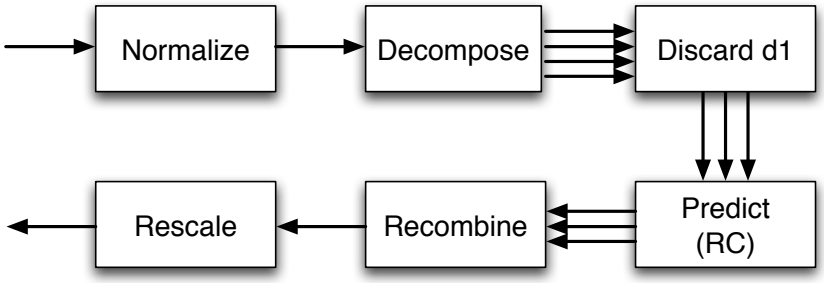


Fig. 1: This is a schematic overview of our prediction methodology. We start by normalizing the given time series. Next, the normalized time series is decomposed by means of wavelet decomposition what results in a trend and a bunch of detail coefficients. Hereafter the level 1 detail coefficient is discarded. The obtained components are predicted separately using reservoir computing (RC). Finally, the predicted components are combined and rescaled in order to get a prediction of the given time series.

3.1 Normalizing the time series

Because we want to work in both the linear and nonlinear part of our recurrent neural network we need to normalize the time series to the interval $[-1, 1]$. Otherwise all neurons would be saturated and thus loosing information. Normalization is done by removing the mean and dividing the outcome by the maximal absolute value:

$$\begin{aligned} \mathbf{x}' &= \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{x_{norm}} &= \mathbf{x}' / \max\left(|\mathbf{x}'|\right), \end{aligned} \tag{2}$$

3.2 Decomposition

Time series can be very often decomposed into components with different dynamics: a trend, periodical effects (sometimes denoted as seasonal effects) and irregular residual components. In [3] wavelet decomposition was motivated because of the easy analysis of the obtained components. A second motivation to use decomposition of the original time series is inherent to the use of reservoir computing which tend to be sensitive to a small temporal range [11]. This can be a problem with time series which contain information on different timescales.

When there is no additional information available about the time series, decomposition can be done by using a set of successive filters. This is also known as multiscale decomposition and described in more detail in [12]. The filters are obtained by rescaling the so called mother wavelet. When the filters are applied iteratively, one obtains a slow varying trend series and a hierarchy of detail components which contain the system's dynamics at different timescales [3]. Because

the system's dynamics are now splitted up into different timescales processing will be a lot easier with reservoir computing. The number of iterations L depends on the length N of the time series and is limited by $L_{max} = \lfloor \log_2 N \rfloor$. But less iterations can be considered by inspecting the derived detail components. After L iterations, time series $\mathbf{y}[k]$, $k = 1...N$, with length N can be written as the sum of the trend $\mathbf{c}_L[k]$ and L detail coefficients $\mathbf{d}_m[k]$, $m = 1...L$:

$$\mathbf{y}[k] \quad = \quad \mathbf{c}_L[k] + \sum_{m=1}^{L} \mathbf{d}_m[k], \tag{3}$$

For our experiments we used the MATLAB Wavelet toolbox for decomposition of the time series. The filters we used were obtained from the discrete Meyer filter because this gave components which have few discontinuities. But we suspect that a Daubechies filter of a sufficient high order is also feasible. In Fig. 3 the first time series of the ESTSP 2008 competition is shown with its trend and detail coefficients using a level eight decomposition. The most noisy coefficient $\mathbf{d_1}$ is not illustrated. By way of inspecting the derived trend and detail components we decided that level eight decomposition produced sufficient smooth components for the first and second time series of the ESTSP 2008 competition. For the third time series we used level 12 decomposition because this gave smoother and more predictable coefficients.

3.3 Prediction

The trend and detail components we obtain from decomposition are used to predict the original time series. We neglect the level 1 component $\mathbf{d_1}$ because it is too noisy to predict. The remaining components are predicted separately by means of reservoir computing. This has as an important implication that correlations between different components are neglected. We plan to investigate the use of many timescales within a reservoir so that all components can be predicted at once instead of training them separately. This would boost calculation time and has the benefit of combining possible correlation between the components.

For each component a fully connected reservoir with 500 sigmoid neurons, one output and only output feedback as an input is constructed. No other external inputs are used. An illustration of this topology can be seen in Fig. 2 The connection weights are scaled so that the largest eigenvalue is nearly 1 which makes the reservoir nearly unstable. The output feedback weights were scaled to 0.1. Depending on the desired output, classical neurons, leaky neurons [7] or band-pass neurons [11] are used. A rule of thumb here is that leaky neurons are used for the slowest components, band-pass neurons for the faster components and classical neurons for the fastest varying components. We determined the leak-rates and band-pass neurons manually based on the frequency spectrum of the components. Because we want to generate the future of a time series, the output is fed back to each of the neurons in the reservoir. During training the desired signal is used as feedback using teacher forcing as we previously explained.

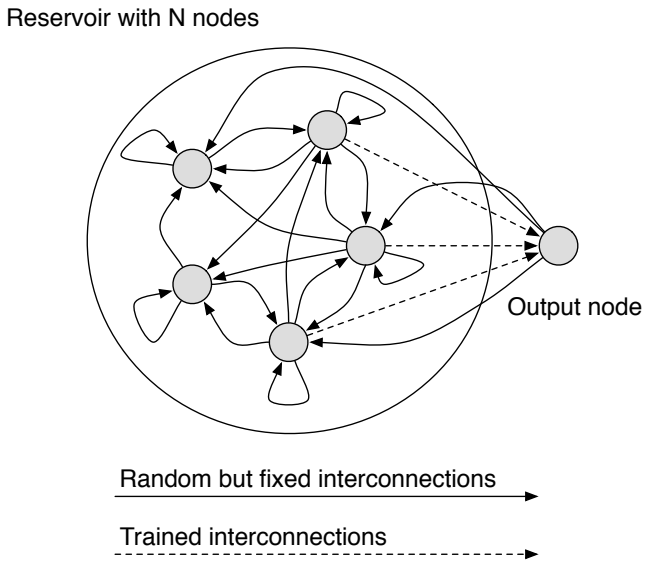


Fig. 2: Schematic overview of the described reservoir topology. The reservoir has only output feedback as an input. Only the output weights, presented by dashed lines, are trained. All other connection weights are initialized randomly and scaled so that the largest eigenvalue is nearly 1.

In order to avoid overfitting we use ridge regression to train the output which proved to have good regularization properties [13], even for generation tasks.

For training and testing we divide each component in three parts: one for training (the largest part), and the two last parts (which have lengths equal to the desired prediction horizon) for validation and testing. The final results for both testing and the competition were obtained by first training the reservoir using teacher forcing with the largest part. The optimal regularization parameter is determined using the performance of the reservoir in predicting the validation part. Next, the reservoir is retrained by teacher forcing it with the first and the second part and using the obtained optimal regularization parameter in order to predict the third (known) testing part. This part is used for evaluation of our approach and these results are presented in the next section. Finally, we train the reservoir again using the complete component in order to predict the unknown samples which are needed for the competition. We repeat this process ten times for each of the components, each time using an other reservoir. The unknown samples were generated by the reservoir which had the best performance on the testing part.

3.4 Composition and rescaling

Recombination of the components can be done by using equation 3. Afterward the composed time series is rescaled again to undo the normalization.

4 Experimental results

The goal of the ESTSP 2008 competition is to predict three different time series each for a different prediction horizon. The evaluation of time series $\mathbf{y}$ with length N and its prediction $\hat{\mathbf{y}}$ is done by calculating the $NMSE$:

$$NMSE = \frac{\sum_{t=1}^{N} (y_t - \hat{y}_t)^2}{L\sigma_y^2}, \tag{4}$$

For the first time series we have to predict the next 18 samples based on a history of 354 samples. Two additional time series were given which could be helpful for prediction of the first time series. After first trying a few prediction setups were these additional time series were used as an input of our reservoir we decided to neglect these external variables. This because they gave no significant improvement. The final result with decomposition of the time series is presented in Fig. 3. A $NMSE$ of 0.25 was obtained on the last known 18 samples. The complete training and prediction procedure takes nearly 2 minutes using an average desktop computer with two gigabyte of memory and a 2.4 GHz Intel based CPU.

The second time series of the ESTSP 2008 competition consists of 1300 samples of which we have to predict the next 100 samples. This time series has a period of 7 samples which makes it convenient to think that it was sampled from a daily updated variable. This thought becomes more pronounced when we cut this time series into sets of 365 samples and look to the correlations between the different sets. Altough we wanted to use this analysis first as additional information into a different approach, we choose to reject it because our results were comparable to the result we have now. We wanted to have one consistent methodology for the three time series. The predictions are shown in Fig. 4. A $NMSE$ of 0.14 was obtained which is the best of the three given time series. A total time of nearly 5 minutes was needed to complete the training and testing procedure.

For the third time series we got 31614 samples of which we had to predict the next 200 samples. Completion of the prediction procedure took three hours which is due to the long sample history and the use of more decomposition levels (and thus needing more reservoirs for prediction of the components). The results are shown in Fig. 5. A $NMSE$ of 0.42 was obtained which gives us the worst performance, this possibly due to the discrete jumps in the trend and the many noisy detail coefficients that we derived from decomposition.

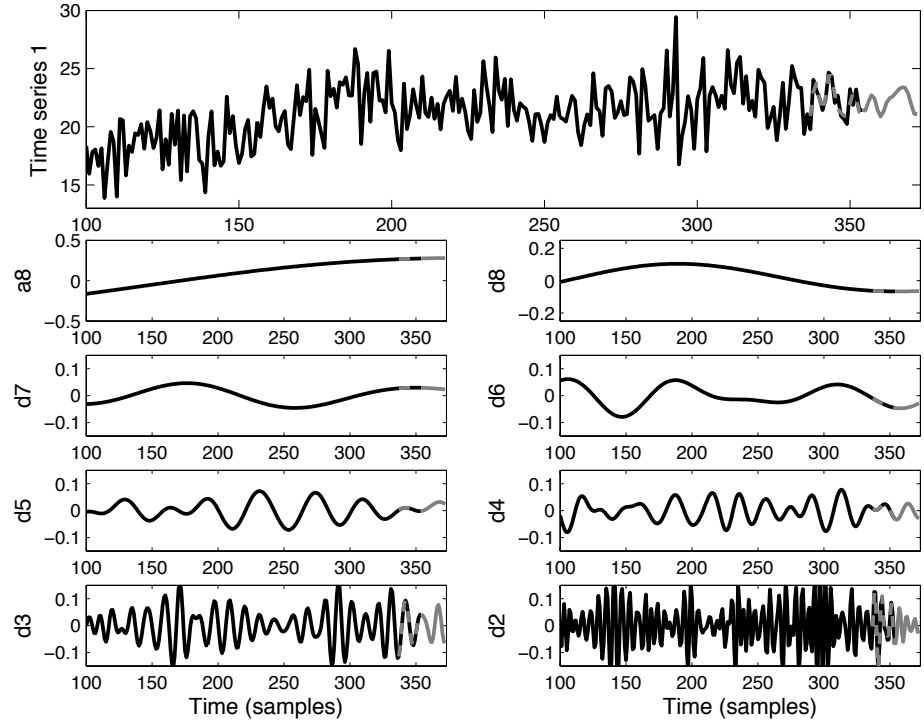


Fig. 3: In solid black lines the original time series 1 of the ESTSP 2008 competition and its decomposition (using level eight wavelet decomposition) into its trend and detail coefficients are shown. The level 1 detail coefficient is not shown because it was too noisy to predict. The last 18 samples of the original time series and its components were predicted in order to evaluate our prediction methodology which is given in a dashed gray line. This resulted in a $NMSE$ of 0.25. The 18 unknown samples which were predicted for the competition are shown in a solid gray line.

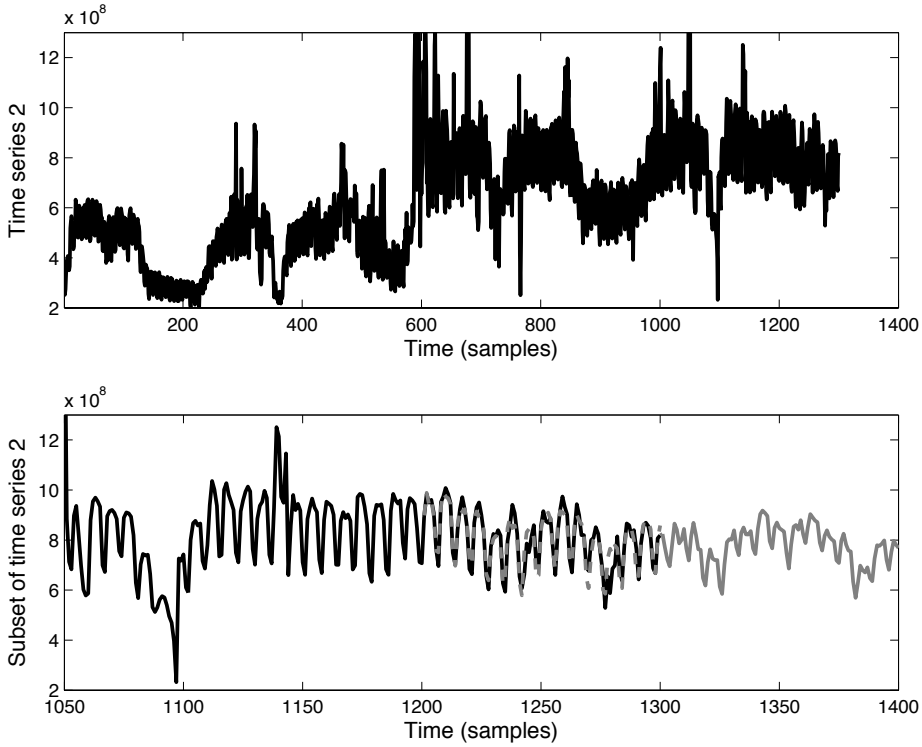


Fig. 4: At the top, the complete time series 2 of the ESTSP 2008 competition is shown in a solid black line. Our method was evaluated on the last 100 samples which gave a $NMSE$ of 0.14. At the bottom, these predictions are marked with a dashed gray line. The next 100 unknown samples for the competition are marked with a solid gray line.

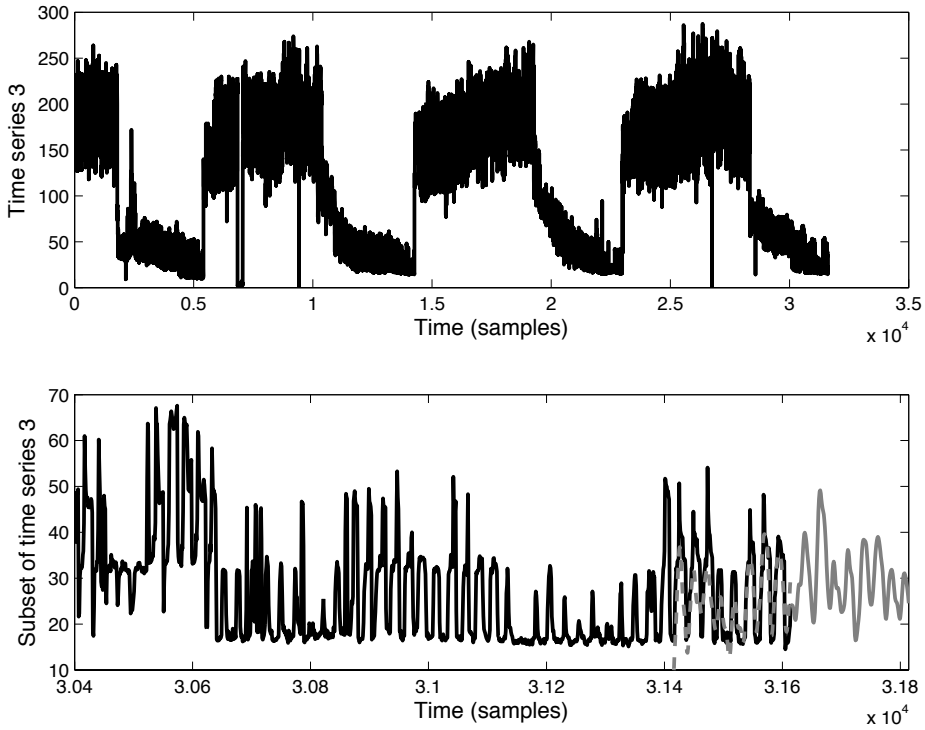


Fig. 5: At the top, an impression of the complete time series 3 of the ESTSP 2008 competition is given. The last 200 samples were used for evaluating our technique which resulted in a $NMSE$ of 0.42. These predicted samples are shown with a dashed gray line. Our prediction for the unknown future of 200 samples is illustrated with solid gray line.

5 Conclusions

In this work a prediction scheme for fast and accurate time series prediction based on wavelet decomposition and reservoir computing was presented. By using time series decomposition, components of different time scales were obtained which are easier to predict. The obtained trend series and detail coefficients were predicted using reservoir computing. We evaluated our method on a known part of the three time series of the ESTSP 2008 competition. In the end, the unknown samples of the three time series were generated. For future work we plan to use a setup using a single reservoir for both decomposition and prediction. This will give us a prediction mechanism which is able to use the interdependence between the obtained components. Therefore we will need to investigate first how many timescales can be used within one reservoir.

References

[1] V. Assimakopoulos and K. Nikolopoulos. The theta method: a decomposition approach for forecasting. *International journal of forecasting*, 16:521–530, 2000.

[2] J. McNames. *Innovations in local modeling for time series prediction*. PhD thesis, Stanford University, 1999.

[3] S. Soltani. On the use of the wavelet decomposition for time series prediction. *Neurocomputing*, 48:267–277, 2002.

[4] Eric. A. Antonelo, Benjamin Schrauwen, and Jan Van Campenhout. Generative modeling of autonomous robots and their environments using reservoir computing. *Neural Processing Letters*, 26(3):233–249, 2007.

[5] Mark D. Skowronski and John G. Harris. 2007 Special Issue: Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, 20(3):414–423, 2007.

[6] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout. Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, 95(6):521–528, 2005.

[7] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.

[8] H. Jaeger. Background information: Jacobs university "smart systems" seminar wins international financial time series competition, 2007.

[9] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[10] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20:391–403, 2007.

[11] F. wyffels, B. Schrauwen, D. Verstraeten, and D. Stroobandt. Band-pass reservoir computing. In *Proceedings of the International Joint Conference on Neural Networks*, 2008.

[12] Ingrid Daubechies. *Ten Lectures on Wavelets (C B M S - N S F Regional Conference Series in Applied Mathematics)*. Soc for Industrial & Applied Math, December 1992.

[13] F. wyffels, B. Schrauwen, and D. Stroobandt. Regularization methods for reservoir computing. In *Proceedings of the International Conference on Analog Neural Networks (ICANN)*, 2008. (accepted).

Time Series Prediction using LS-SVMs

Marcelo Espinoza, Tillmann Falck, Johan A. K. Suykens and Bart De Moor *

{marcelo.espinoza,tillmann.falck,johan.suykens}@esat.kuleuven.be
K.U.Leuven, ESAT - SCD
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium

Abstract. This paper describes the use of LS-SVMs as an estimation technique in the context of the time series prediction competition of ESTSP 2008 (Finland). Given three different time series, a model is estimated for each series, and subsequent simulations of several points after the last available sample are produced. For the first series, a NARX model is formulated after a careful selection of the relevant lags of inputs and outputs. The second and third series show cyclical or seasonal patterns. Series 2 is modelled by adding deterministic "calendar" variables into the nonlinear regression. Series 3 is first cleaned from the seasonal patterns, and a NAR model is estimated using LS-SVM on the deseasonalized series. In all cases, hyperparameters selection and input selection are made on a cross-validation basis.

1 Introduction

Time series prediction can be treated as a special case of system identification [1]. In nonlinear system identification [2, 3] it is common to use past lags of the output variable $y \in \mathbb{R}$ and, if available, of exogenous input variables $\boldsymbol{u} \in \mathbb{R}^d$ to build a NARX model of the form

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, \boldsymbol{u}_{t-1}, \boldsymbol{u}_{t-2}, \ldots, \boldsymbol{u}_{t-q}) + e_t, \tag{1}$$

where e_t is assumed to be a white noise process. The estimated model can then be used for prediction or simulation. Nonlinear effects can be identified when the function f is parameterized as a nonlinear function. In this article, we use Least-Squares Support Vector Machines (LS-SVMs) [4] as a tool for estimating f in NARX models in order to produce the required simulations for three time series available in the context of the time series competition of the Second European Symposium of Time Series Prediction (ESTSP 2008, Porvoo, Finland, http://www.estsp.org/).

LS-SVMs belong to the class of kernel methods [4, 5, 6], which use positive-definite kernel functions to build a nonlinear representation of the original inputs in a high-dimensional feature space. An optimization problem consisting of a

*The work presented was funded by Research Council KUL: GOA AMBioRICS, CoE EF/05/006, IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; FWO: PhD/postdoc grants, projects G.0452.04, G.0499.04, G.0211.05, G.0226.06, G.0321.06, G.0302.07, G.0320.08, G.0558.08, G.0557.08, research communities (ICCoS, ANMMM, MLDM); IWT: PhD Grants, McKnow-E, Eureka-Flite+, Helmholtz: viCERP, IUAP P6/04; EU: ERNSI;Contract Research: AMINAL. Johan Suykens is a professor and Bart De Moor is a full professor at the Katholieke Universiteit Leuven, Belgium.

regularized least-squares cost function and equality constraints is formulated in primal space and solved in the dual variables. This formulation is flexible in the sense that it allows the incorporation of different elements of knowledge about the problem at hand. In the primal domain the model is parametric and it is easy to incorporate some types of prior knowledge that gets automatically embedded in the dual representation. Examples for this are the inclusion of partially linear models [7], autocorrelated residuals [8, 9] and monotonicity [10].

The remainder of the paper is structured as follows. A general overview of LS-SVMs and a discussion about practical elements are given in Section 2. The study cases for time series of the competition are discussed in Sections 3 and 4.

2 General Methodology

2.1 Least Squares Support Vector Machine Regression

LS-SVMs belong to the class of kernel methods, which use positive-definite kernel functions to build a nonlinear representation of the original inputs in a high-dimensional feature space. We start by parameterizing NARX model (1) as

$$y_t = \boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}_t) + b + e_t \tag{2}$$

where $y_t \in \mathbb{R}$, $\boldsymbol{x}_t \in \mathbb{R}^n$ is the regression vector $[y_{t-1}, y_{t-2}, \ldots, y_{t-p}, \boldsymbol{u}_{t-1}, \boldsymbol{u}_{t-2}, \ldots, \boldsymbol{u}_{t-q}]$, $b \in \mathbb{R}$ is a bias term, $\boldsymbol{w} \in \mathbb{R}^{n_h}$ is an unknown coefficient vector, and $\boldsymbol{\varphi} : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$ is a nonlinear feature map, which transforms the original input $\boldsymbol{x}_t \in \mathbb{R}^n$ to a high-dimensional vector $\boldsymbol{\varphi}(\boldsymbol{x}_t) \in \mathbb{R}^{n_h}$, which can be infinite dimensional [6]. Consider the following constrained optimization problem with a regularized cost function:

$$\min_{\boldsymbol{w},b,e_t} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \gamma\frac{1}{2}\sum_{t=1}^{N} e_t^2 \tag{3}$$

$$\text{s.t.} \quad y_t = \boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}_t) + b + e_t, \quad t = 1, \ldots, N,$$

where γ is a regularization constant and K is a p.d. kernel function. With the application of the Mercer's theorem for the kernel matrix $\boldsymbol{\Omega}$ as $\boldsymbol{\Omega}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\varphi}(\boldsymbol{x}_i)^T\boldsymbol{\varphi}(\boldsymbol{x}_j), i, j = 1, \ldots, N$ it is not required to compute explicitly the nonlinear mapping $\boldsymbol{\varphi}(\cdot)$ as this is done implicitly through the use of positive definite kernel functions K. For $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ there are usually the following choices: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j$ (linear kernel); $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T\boldsymbol{x}_j + c)^d$ (polynomial of degree d, with $c \geq 0$ a tuning parameter); $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2/\sigma^2)$ (radial basis function, RBF), where σ is a tuning parameter.

The problem (3) is solved using Lagrange multipliers and the solution is expressed in dual form [4]. The final expression for the estimated f is given by

$$\hat{f}(\boldsymbol{x}) = \sum_{t=1}^{N} \alpha_t K(\boldsymbol{x}_t, \boldsymbol{x}) + b. \tag{4}$$

The one-step-ahead prediction is simply $\hat{y}_{N+1} = \hat{f}(\boldsymbol{x}_{N+1})$ using the estimated $\hat{f}$; simulation $n-$steps ahead can be obtained by iteratively applying the prediction equation replacing future outputs by its predictions [1, 2].

2.2 Practical Implementation

The training process of LS-SVM involves the selection of kernel parameters and the regularization constant γ. A good choice of these parameters is crucial for the performance of the estimator. In this paper, we use 5-fold cross-validation for selecting these parameters. The second important choice is the selection of regressors, i.e., which lags of inputs and outputs are going to be included in the regression vector $\boldsymbol{x}_t$. This selection is done by using a large number of initial components and then performing a greedy search to prune non-informative lags on a cross-validation basis. Therefore an initial model containing all regressors is estimated and optimal choices for the parameters are made. On each stage of the greedy backwards elimination process, a regressor is removed if the cross-validation Mean Squared Error (CV-MSE) improves. The final set of regressors is then used for the final predictions. For the purpose of model estimation, all series are normalized to zero mean and unit variance.

3 Analysis for Time Series 1

The available data for time series consists of a sequence $\{y_t, u_t, v_t\}_{t=1}^{N}$ for the output y and the two exogenous inputs u and v with $N = 354$ datapoints. The series are depicted in Figure 1. The goal is to produce a sequence of simulated future values $\hat{y}_{N+1}$ until $\hat{y}_{N+18}$. Based on autocorrelation analysis, it can be noticed that all three variables share some periodic behavior with a period of 12 samples.

3.1 Modeling Strategy

We test two model specifications, with and without the exogenous inputs (NAR and NARX models, respectively), to be estimated using LS-SVM, as follows:

- (NAR) $\hat{y}_t = \hat{f}(y_{t-1}, \ldots, y_{t-p})$ with $p \leq 48$
- (NARX) $\hat{y}_t = \hat{f}(y_{t-1}, \ldots, y_{k-p}, u_{t-1}, \ldots, u_{t-q_1}, v_{t-1}, \ldots, v_{t-q_2})$ with p, q_1, $q_2 \leq 48$

The performance measure used for 5-fold cross-validation is the the CV-MSE over two different prediction scenarios:

- One step ahead prediction for each test fold,
- Simulation of the time series for 18 time steps for every sample in the test fold, averaging the values at each time step over predictions (1 step, 2 steps, ..., 18 steps)

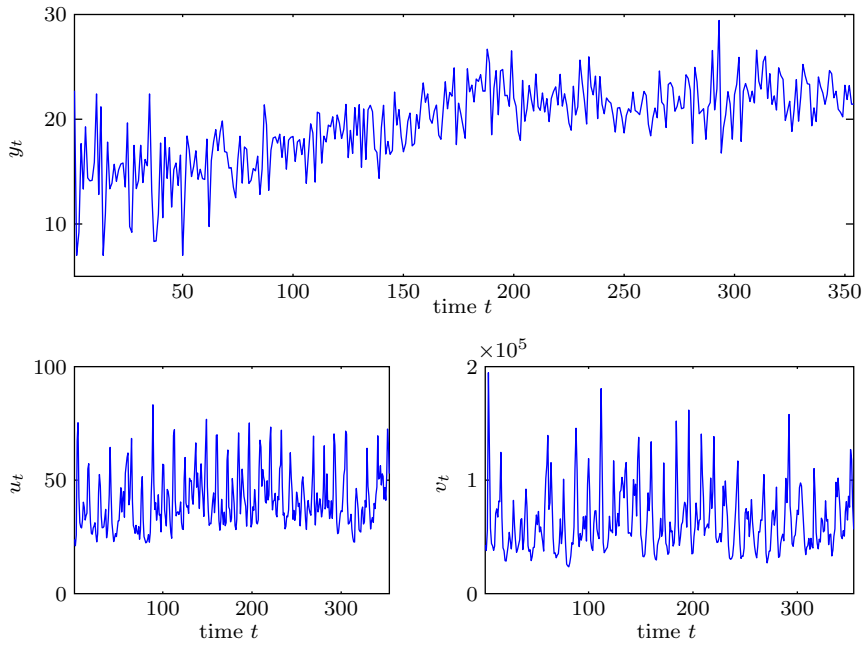


Fig. 1: Time plots of target variable y (top) and the exogenous variables u (bottom left) and v (bottom right) of time series 1

3.2 Results

Consider the NAR(48) model formulation. The selected lags to be included in the regression vector are pruned using a greedy backwards search. The CV-MSE as a function of the number of regressors included in the model is shown as the blue line in Figure 2. By actively selecting the regressors the MSE can be improved by 33%. The best NAR model uses 12 regressors and achieves a CV-MSE of 0.2834. However, further correlation analysis between the model residuals and the exogenous inputs reveal that some information has not been captured by the NAR model alone. This means we should test a NARX formulation.

Two different NARX models are evaluated. The first one includes all lags up to 48 for y, u and v, NARX(48,48,48). The second model assumes a delay between inputs and output of 18 time steps. The NARX(48,30,30) model contains the lags 19 up to 48 for the inputs u and v. Figure 2 shows the feature selection process for the models, both of which clearly outperform the NAR model. The model using all regressors achieves a CV-MSE of 0.2437 with 34 regressors, whereas the delayed model uses 25 regressors and has a CV-MSE of 0.2489. The improvement of the full over the delayed model is rather small with 2%. For the purpose of generating the results for the time series competition, the full model would require to simulate the exogenous inputs as well, which might be an additional source of errors for the simulated outputs. On the other hand,

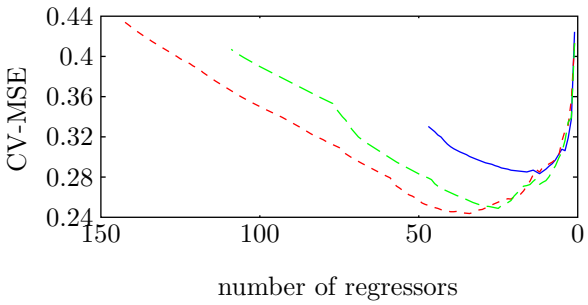


Fig. 2: Feature selection using greedy backward search. Solid blue: NAR model (lags y: 1-48), green long dashes: NARX (lags y: 1-48, u: 1-48, v: 1-48), red short dashes: NARX (lags y: 19-48, u: 19-48, v: 19-48)

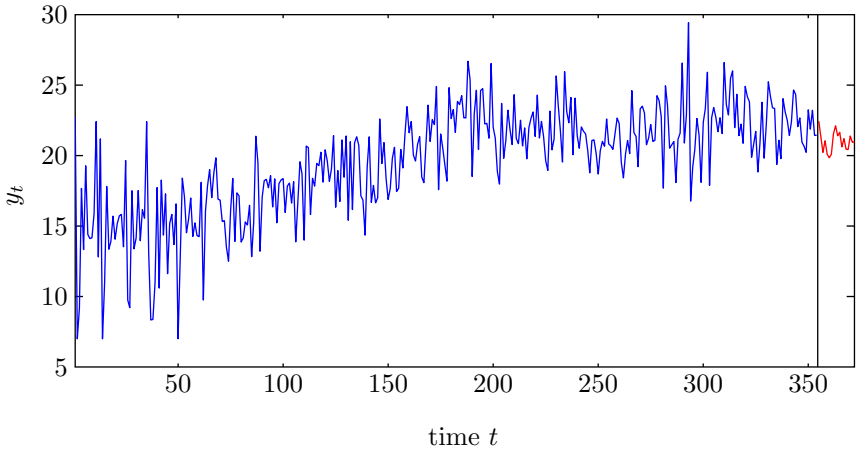


Fig. 3: Simulations for the next 18 points for Series 1, shown after the vertical line.

the delayed model does not have this problem, and we can always use actual observations to generate the simulated outputs. Considering that the difference in performance is small, we select the delayed NARX model to simulate 18 consecutive new samples. The simulated values together with the training data are shown in Figure 3.

4 Analysis for Time Series 2 and 3

Series 2 and Series 3 are modelled following a similar methodology. Series 2 (Figure 4, top) consists of 1,300 observations, and the goal is to simulate the next 100 points. Series 3 contains 31,614 samples (Figure 5, top), and the goal is to predict the next 200 points. The two series display a similar cyclical behavior. Series 2 display a strong correlation every 7 samples, as can be seen

in the autocorrelation plot (Figure 4, center). Such pattern would correspond to a "weekly" seasonal cycle for a series consisting of consecutive daily values. In the same manner, Series 3 displays cyclical patterns similar to those of "daily", "monthly" and "yearly" cycles for a series consisting of hourly values.

4.1 Modeling Strategy

The seasonal patterns detected in the series suggest to use a specific seasonal modeling strategy, following the golden-rule of "do not estimate what you already know" [2]. One of the most common approaches is the use of deterministic calendar information to keep track of the sequence of patterns involved. For the case of Series 2, we use the binary-valued vector $\boldsymbol{W}_t \in \{0,1\}^7$, which is a vector of zeros with a 1 in the position of the day of the week at time t to keep track of the day-to-day cycle. For example, Monday corresponds to $\boldsymbol{W}_t = [1, 0, \cdots, 0]$. In the same way, the variable $\boldsymbol{M}_t \in \{0,1\}^{12}$ is defined as a vector of zeros with a 1 in the position of the month at time t. A binary-valued vector $\boldsymbol{H}_t \in \{0,1\}^{24}$ is similarly defined to keep track of the hour of the day at time t. These variables are considered as exogenous inputs to the corresponding models for each series.

4.1.1 Model for Series 2

The estimated model is the following NARX formulation:

$$y_t = f(y_{t-1}, \ldots, y_{t-p}, \boldsymbol{W}_t, \boldsymbol{M}_t) + e_t \tag{5}$$

estimated using LS-SVM. The order p, the hyperparameters and the relevant regressors are determined on a 5-fold cross-validation basis using the greedy search optimization procedure described previously.

4.1.2 Model for Series 3

This series is modelled differently because of the strong presence of the seasonal patterns. Series 3 shows a very strong combination of seasonal patterns that can be considered the "backbone" of the observed series. Following a standard approach in seasonal modeling [11], we decompose the original series as the sum of a regular and an irregular component, $y_t = r_t + z_t$, where $r_t = \beta_1^T \boldsymbol{H}_t + \beta_2^T \boldsymbol{M}_t + \beta_3^T \boldsymbol{W}_t$ is the contribution of the deterministic seasonal variables. The identification of r_t and z_t is obtained by estimating the following linear regression:

$$y_t = \beta_1^T \boldsymbol{H}_t + \beta_2^T \boldsymbol{M}_t + \beta_3^T \boldsymbol{W}_t + z_t, \tag{6}$$

with $\beta_1 \in \mathbb{R}^{24}, \beta_2 \in \mathbb{R}^{12}, \beta_3 \in \mathbb{R}^7$ estimated with ordinary least-squares. The irregular component z_t corresponds to the residual of this regression. Figure 5 shows the original series (top) and the decomposition in the regular component r_t (center, left) and the irregular component z_t (center, right).

Predicting the regular component into the future is straightforward. For the prediction of the irregular component z_t, we estimate a NAR model using

LS-SVM,

$$z_t = f(z_{t-1}, \ldots, z_{t-p}) + e_t. \tag{7}$$

The irregular component z_t still displays significant autocorrelation with a 24 hours period (Figure 5 bottom left), which should be captured by the NAR model. Given that z_t is much more stationary than the original series, the LS-SVM model is estimated only using the last 1,000 observations. The order of this model, hyperparameters and relevant lags are determined as in the other models.

4.2 Results

For Series 2, the best order of the NARX model is found to be $p = 14$, which gives a total number of regressors of 33 (14 past values, 7 days of the week, 12 months in a year). The model with 33 regressors shows a CV-MSE= 0.23. From the 33 regressors, only 11 are found to be relevant, lowering the CV-MSE to 0.20. This final model with 11 regressors is used to produce the final simulations. The final simulations are shown on the bottom panel of Figure 4.

For Series 3, the best order of the NAR model on y_s is $p = 48$. The selection of relevant regressors yields no significant improvement. The autocorrelation present in z_t has been captured by the model, and the residuals e_t show no such correlation (Figure 5 bottom right). Overall, the CV-MSE obtained following this strategy is 0.004. The NAR model is used to produce the simulations for y_s, which are added to the simulations for the regular component. The final simulation requested for the competition is shown on Figure (6).

5 Conclusions

This paper described the use of LS-SVMs as an estimation technique in the context of the time series prediction competition of the Second European Symposium ESTSP 2008 (Porvoo, Finland). Given three different time series, a model is estimated for each series, and subsequent simulations of several points after the last available sample are produced. For the first series, a NARX model is formulated after a careful selection of the relevant lags of inputs and outputs. Supported by correlation analysis, it is determined that the exogenous inputs provided for this series have a significant influence on modeling of the output series.

The second and third series are modelled independently, yet following a similar methodology. Both series show seasonal variations, and we used deterministic seasonal variables as exogenous inputs. Series 2 is modelled by using a NARX formulation including the deterministic variables; Series 3 is first "deseasonalized" by using a linear regression of the series on the deterministic variables, and later a NAR model is estimated with LS-SVM on the residuals of the first regression.

The selected models over the three series are those who provided the best

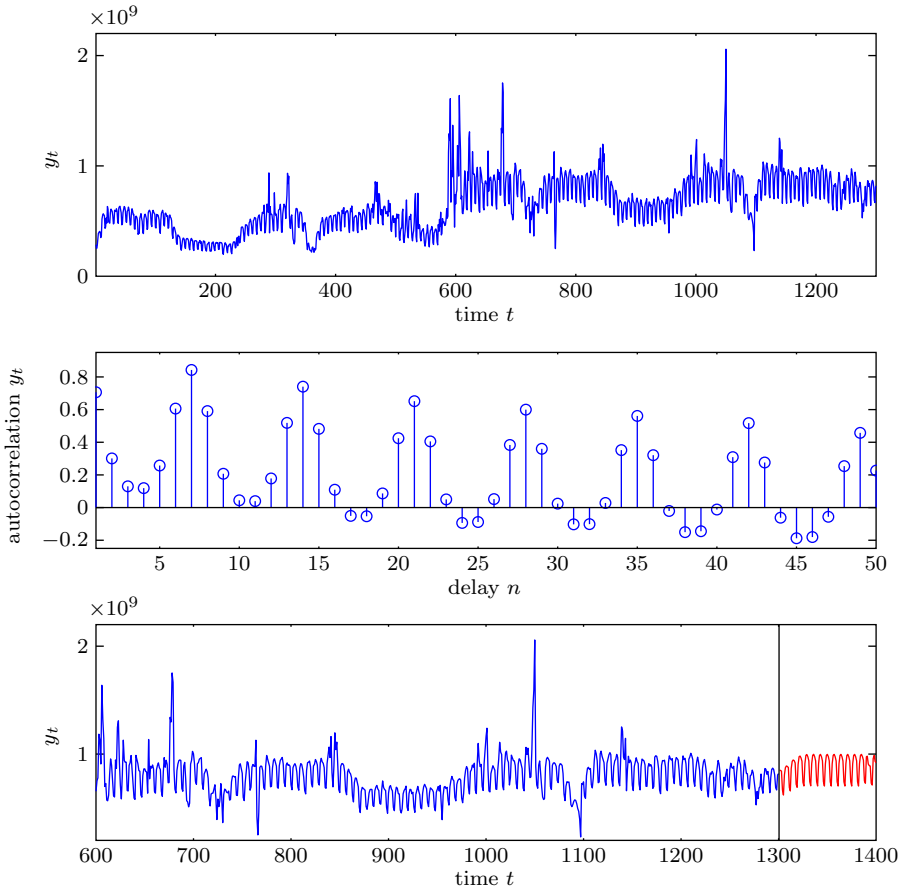


Fig. 4: The original data for Series 2 (top) shows a seasonal pattern visible in the autocorrelation plot (center) of the series. Using a NARX formulation, the requested simulations are computed for the next 100 points (bottom), shown after the vertical line.

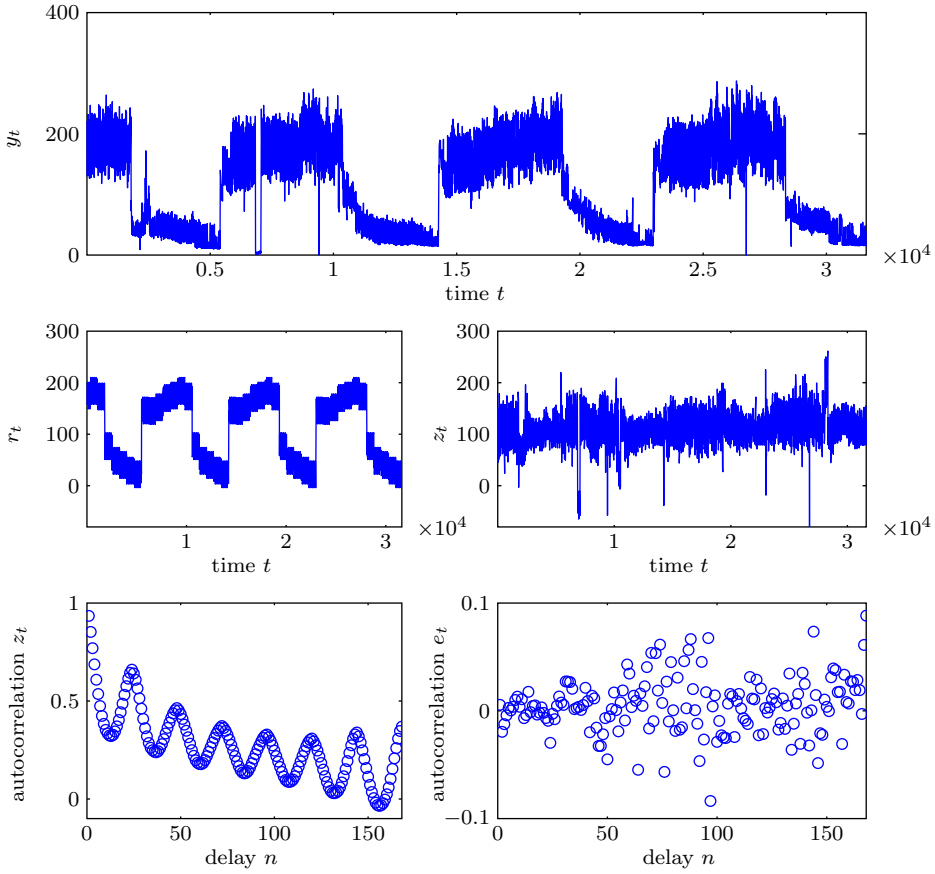


Fig. 5: The original data for Series 3 (top) can be decomposed as the sum of a regular component r_t (center, left) and an irregular component z_t (center, right). Using a NAR formulation to model the irregular component z_t, the final model captures the autocorrelation that was still present in z_t. This is visible by comparing the autocorrelation plot for z_t (Bottom, left) with that of the NAR model residuals e_t (bottom, right).

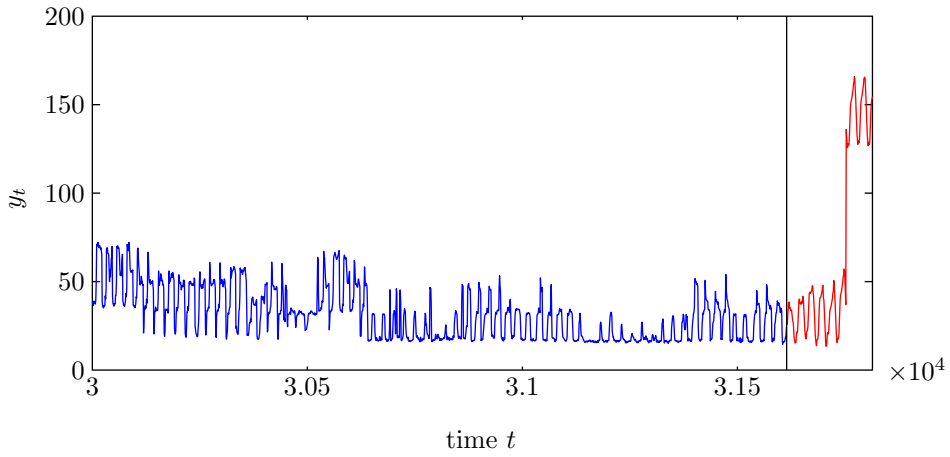


Fig. 6: Simulations for the next 200 points for Series 3, shown after the vertical line.

performance on a cross-validation basis, in terms of order selection, kernel parameters, regularization constant and relevant regressors.

References

[1] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, New Jersey, 1987.

[2] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear Black-box Modelling in System Identification: a Unified Overview. *Automatica*, 31:1691–1724, 1995.

[3] A Juditsky, H. Hjalmarsson, A. Benveniste, B. Deylon, L. Ljung, J. Sjöberg, and Q. Zhang. Nonlinear Black-box Modelling in System Identification: mathematical foundations. *Automatica*, 31:1725–1750, 1995.

[4] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[5] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[6] V. Vapnik. *Statistical Learning Theory*. Wiley, New-York, 1998.

[7] M. Espinoza, J.A.K. Suykens, and B. De Moor. Kernel based partially linear models and nonlinear identification. *IEEE Transactions on Automatic Control, Special Issue: Linear vs. Nonlinear*, 50(10):1602–1606, 2005.

[8] M. Espinoza, J.A.K. Suykens, and B. De Moor. LS-SVM regression with autocorrelated errors. In *Proc. of the 14th IFAC Symposium on System Identification (SYSID 2006)*, pages 582–587.

[9] M. Espinoza, J.A.K. Suykens, R. Belmans, and B. De Moor. Electric load forecasting: Using kernel-based modeling for nonlinear system identification. *IEEE Control Systems Magazine*, 27(5):43–57, 2007.

[10] K. Pelckmans, M. Espinoza, J. De Brabanter, J.A.K. Suykens, and B. De Moor. Primal-dual monotone kernel regression. *Neural Processing Letters*, 22(2):171–182, 2005.

[11] S. Hylleberg. *Modelling Seasonality*. Oxford University Press, 1992.

Exogenous Data and Ensembles of MLPs for Solving the ESTSP Forecast Competition Tasks

Paulo J. L. Adeodato[1,2], Adrian L. Arnaud[1,2], Germano C. Vasconcelos[1,2], Rodrigo C.L.V. Cunha[1,2], Domingos S.M.P. Monteiro[1,2]

1- Universidade Federal de Pernambuco - Center for Informatics
Recife, Pernambuco - Brazil
2- NeuroTech Ltd.
Recife, Pernambuco - Brazil

Abstract. This paper presents an extension of an NN3 award winning solution to the ESTSP competition. The problem consists of three tasks of multi-step ahead forecasting for series with different lengths. Analysis allowed for using "exogenous" data. Temporal preprocessing was followed by training several Multilayer Perceptron Networks, with architecture and algorithm optimized for the best MSE median. The multi-step ahead forecasts were produced by 31 networks and their median value was chosen. Afterwards, the values of each predicted series were de-normalized and had their trend re-inserted for producing the final forecasts. The competition results will show this solution worked well.

1 Introduction

Time series prediction has been one of the priority areas of research for statisticians, economists and computer scientists among others and several different approaches have been proposed for solving this problem along the last decades.

The statistical technique of Box & Jenkins (ARIMA models) [1] became one of the most popular among practitioners in actual world forecasting tasks. However, ARIMA models are linear, a feature which represents a limitation for predictive modeling. Nonlinear approaches have been proposed for overcoming this constraint. Bilinear models [2], threshold autoregressive models [3] and exponential autoregressive models [4] among others are examples of such attempts. These nonlinear approaches, however, are mathematically very complex. Artificial neural networks were a recent alternative proposed for non-linear modeling of time series [5], more recently combined with evolutionary approaches for the network parameters' optimization (e.g. topology, number of processing units, learning rate etc.) [6].

The strength of the work carried out here, however, relies strongly on an ensemble of ideas from different areas, more on its sound statistical procedures, and less on the particular forecasting techniques used.

The summary of the steps carried out in each task is listed below. Apart from the first step, the idea is focused on a systematic approach for predicting multiple time series, multiple steps ahead:

1. Series analyses (specific for each task)
2. Test set separation

3. Trend removal
4. Data normalization
5. Predictive model optimized selection
6. 31-Replicas model training without test set
7. Median model selection
8. Performance evaluation
9. Phase shift measurement
10. 31-Replicas model training with test set
11. Median forecast selection
12. Phase shift correction
13. Data de-normalization
14. Trend re-insertion

This paper is organized as follows. Section II presents the data analysis for each task. Section III shows the data selection approach. In Section IV, the data preparation is explained. Section V describes the predictive modeling. Section VI presents some results and interpretation on the test data. Section VII explains the process for forecast generation of the k values ahead for each task of the competition. Section VIII finalizes the paper with remarks on what the team has done, the results achieved on the modeling data and the competition results, along with what the team has yet to do.

2 Data Analysis

The modeling data series were plotted for the series of each task, separately, to help modelers get an intuitive feeling of the types of behavior present. In this sense, each task presented very different characteristics which will be detailed in this section.

2.1 **Task-1**

The task consists in predicting one time series for the next 18 steps in the future. The information available was the series itself plus two other exogenous series, all with 354 data points. Unfortunately, the level of correlation between the series to be predicted and the exogenous series was very low (correlation < 0.3) for lags within a 50 observations displacement. Maybe, further analyses and data transformation would have detected correlation of a more complex nature but there was not much time for available, so the exogenous series were discarded from modeling.

2.2 **Task-2**

Task-2 consists in predicting one time series for the next 100 steps in the future. The information available was the series with 1300 data points plotted in Fig. 1. Despite its irregular aspect, after some data transformation and serial correlation analysis, this series shows a daily behavior along slightly over 3 and half years, as the data labels indicate. This perception allows for much richer input information in the modeling stage such as yearly, monthly and weekly features.

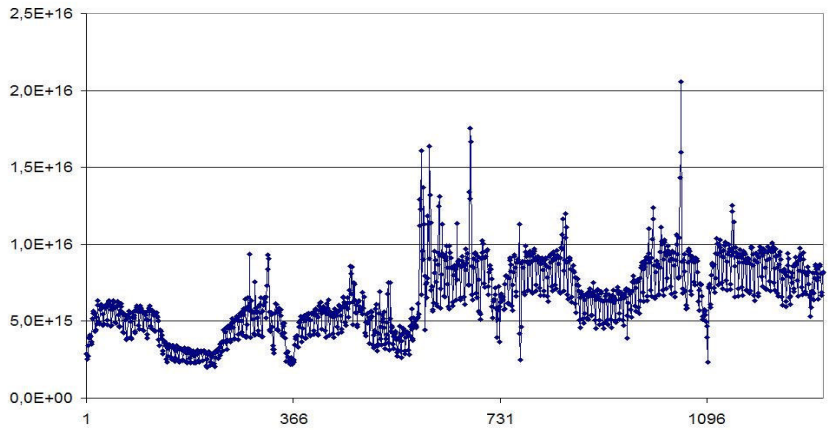


Fig. 1: Original Time Series of Task-2.

2.3 **Task-3**

Task-3 consists in predicting one time series for the next 200 steps in the future. The information available was the series with 31614 data points plotted in Fig. 2.

Despite its irregular aspect, with two phases, the future 200-step forecasting horizon is very unlikely to reach the next phase transition. The "low phase" has 3 complete plateaus with a minimum of around 3600 data points and the tail of an incomplete one with only around 3280 data points. By adding the 200 forecasting steps to this, results in 3480 data points which is much smaller than the smallest plateau previously observed. Taking this into account, the focus of this task becomes the "low phase" of the series, only.

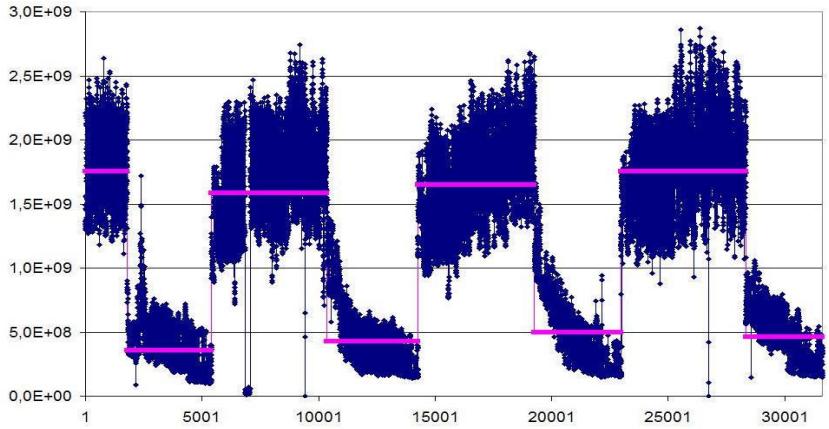


Fig. 2: Original Time Series of Task-3, along with the mean plateau of each "phase".

After serial correlation analysis, this series was decomposed into 4 pieces, from the "most stable" part of each "low phase" plateau. Unfortunately, the correlations were not strong enough for the forecasting system to benefit from them. Thus, the modeling was carried out with only the last 1000 data points from the last "low phase" plateau.

3 Data Selection

As stated above, the approach presented here is robust particularly because it relies on sound statistical procedures.

Hence, a sample of the last *k* observations of each time series to be predicted was separated only for performance assessment (the test set) aiming at preserving statistical independence from the parameter estimation process. The remainder of the data (the modeling set) was used for parameter setting and predictive modeling. The sizes of the test sets were 18, 100 and 200, corresponding to the number of steps ahead in the predictions task 1, 2 and 3, respectively.

Trend analysis and elimination were carried out only considering the parameters extracted from the modeling set. The same was done for all the analyses conducted for lag definition and for establishing the parameters for data transformation, such as normalization and outlier filtering. Accordingly, the predictive modeling (learning) was also estimated only with the modeling data set.

4 Data Preparation

The trends of the series were approximated by the best fitting linear functions which were subtracted from the series' data for later re-insertion, after the system final prediction.

After trend removal, the series were normalized in such a way that all data points stayed in the range from 0.1 to 0.9 allowing for over 10% increase beyond these boundaries; 0-1 range.

These transformations were then applied to the remaining data set (the test set). After these basic transformations, the temporal properties of the data series were checked through correlograms and Fourier analysis focusing on the time window needed for the predictive modeling. Since they (the correlograms and Fourier analysis) lead to different window sizes, both were considered on the optimization process by the modeling technique.

5 Predictive Modeling

Considering the complexity of systematically modeling several time series, this paper re-uses the methodology already successfully developed for NN3 Forecasting Competition, last year. That proved being an effective approach to develop robust solutions based only on each series data alone for the prediction of the future values. For this reason the solution relied on basic statistics over a number of independent trials: the median value predicted by 31 forecasting systems.

The well-known multilayer perceptron (MLP) was the modeling technique chosen. The MLP has been one of the neural network models most frequently used in pattern classification problems for its excellent generalization capacity, simplicity of operation and ability to perform universal function approximation [7]. It also presents robustness when compared to other techniques [8]. However, one drawback of this technique is the need of a validation data set for preventing over-fitting, which is critical in situations where there are few data observations available, such as in some forecasting problems. General approaches for data multiplication such as data division (n-fold cross-validation and leave-one-out) or data sampling strategies (Monte-Carlo and bootstrapping) would be useful but they are applicable to statistically independent samples [9]; not to time series data.

The MLP were chosen with just a single hidden layer with processing units varying from 1 to 30 and were trained either with the standard error back propagation algorithm [10] or with the Levenberg-Marquadt algorithm [11], having the minimum squared error on the validation set as the training stopping criterion.

If, in one side, the small amount of data is a drawback in terms of noisy solutions, on the other side, it yields low cost for a large amount of simulations for noise filtering through median forecasting.

Thus, each time series was exhaustively tested for the two algorithms, all the possible number of processing units within the pre-defined range (1 to 30) and input window sizes (lags) based on either correlograms or Fourier analyses [12].

The architecture, training algorithm and input window selected were then replicated to produce 31 systems trained from different initial states (weight initialization for symmetry breaking randomly drawn from a uniform distribution between -10^{-4} and $+10^{-4}$). The validation set was used for measuring the error for training stopping criterion again and also for defining the temporal phase shift [13] in case it was identified to improve performance.

Then the k step ahead forecasts were produced for each one of the series by all the 31 systems and compared to the values of the reserved test set. The 31 systems performance on both the validation and the test set were then compared for assessing two main aspects: quality degradation from the dependent to the independent data set and the quality itself, both measured in terms of the competition criterion: the NMSE metrics. The choice was for the solution simultaneously closest to the median in terms of quality and to the median in terms of degradation.

6 Results and Interpretation

The results presented below show the graphs of series selected from the known data separated into validation and test sets, as described in the previous sections, along with the median forecast produced. The figures present the results for the 3 tasks of the ESTSP 2008 Competition. The results are assessed with six metrics commonly used for time series forecasting assessment: MSE (Mean Squared Error), MAPE (Mean Absolute Percent Error), SMAPE (Symmetric Mean Absolute Percent Error), U-Theil, POCID (Prediction On Chance In Direction) and correlation, each capturing important aspects in forecasting performance.

Fig. 3 below shows the comparison of the actual and the predicted data points on the statistically independent test set for Task-1. The average predicted values are

not far from the actual ones, particularly considering that the Y-axis starts from the value 20. However, their oscillatory behaviors do not match much; the POCID and their correlation are low.

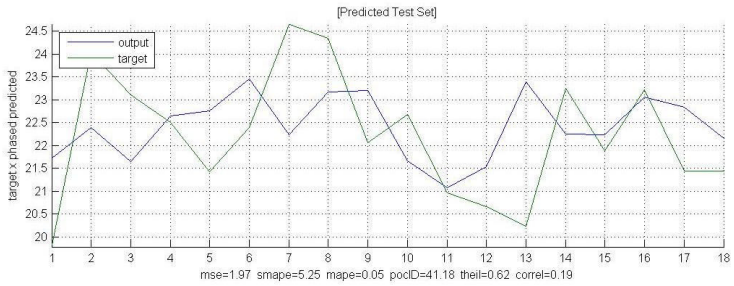


Fig. 3: Performance evaluation on the test set for time series of Task-1.

Fig. 4 below shows the comparison of the actual and the predicted data points on the statistically independent test set for Task-2. The excellent quality of the prediction is visually perceived. In this task, despite the series having a high coefficient of variation, the prediction follows the same oscillatory behavior as the actual data. This is even more relevant when taking into account that this is a 100-step ahead forecast. All these aspects are supported by the metrics below; this time the POCID and the correlation go up to 0.7.

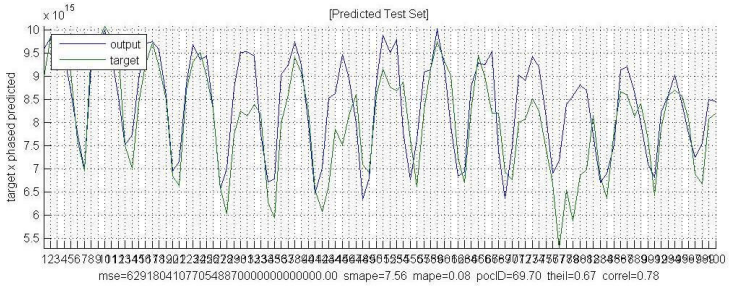


Fig. 4: Performance evaluation on the test set for time series of Task-2.

Fig. 5 below shows the comparison of the actual and the predicted data points on the statistically independent test set for Task-3. The quality of the prediction is good. In this task, the series oscillates a lot producing a coefficient of variation even higher than in Task-2. That is why the SMAPE performance is much worse than in the other series. An important aspect is that the prediction follows closely the oscillatory behavior of the actual data; both the POCID and the correlation high metrics confirm that. This result is even more relevant when considering that this is a 200-step ahead forecast.

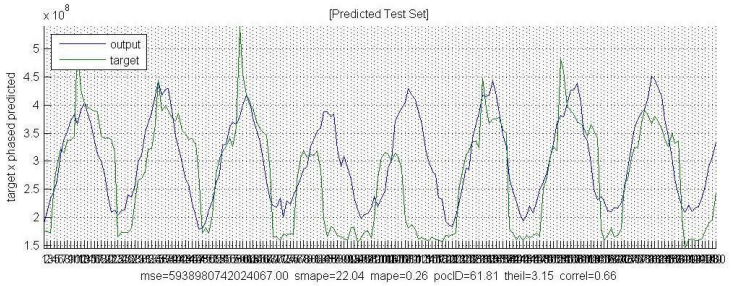


Fig. 5: Performance evaluation on the test set for time series of Task-3.

7 Competition's Forecast Generation

For producing the competition forecasts, the same methodology used above was applied to all 3 series with a single difference: this time the *k* last data observations reserved as test set before were now included in the modeling data as validation set. The 31 replicas of the selected MLP neural network architecture were retrained by the selected algorithm with this increased modeling data set. The validation set (former test set) was used for stopping training and for measuring all the correction factors on it.

Then the trained system was used to produce 31 values for each of the *k* forecasts ahead for each series. For the competition, the median of the 31 values was selected for each forecast ahead and further post-processing was carried out. Finally, the parameters estimated in the modeling stage were now applied to produce the corresponding phase correction, value de-normalization and trend re-insertion on each series for completing the results of the competition submission file.

8 Concluding Remarks

This paper has presented a combination of careful time series analysis and a principled methodology based on an ensemble of multilayer perceptron neural networks and basic statistics to produce robust multi-step ahead time series forecasting.

A thorough procedure has been carried out for defining the most adequate combination of MLP architecture, training algorithm and time window for each series aiming at optimizing the NMSE metrics, defined as the competition performance evaluation criterion. The robustness of the solution has been assured through the use of median forecasts of several systems.

Therefore, it was expected that the forecasts would not deviate from the actual values more than what had been measured during the modeling stage. The competition result would confirm that expectation.

The solution could be still further improved by deeper non-linear analysis for capturing relevant information from the exogenous series.

References

[1] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. New Jersey: Prentice Hall, 1994.

[2] T. S. Rao and M. M. Gabr, *Introduction to Bispectral Analysis and Bilinear Time Series Models*, ser. Lecture Notes in Statistics. Berlin: Springer, 1984, vol. 24.

[3] T. Ozaki, *Nonlinear Time Series Models and Dynamical Systems*, ser. Hand Books of Statistics, E. J. Hannan and P. R. K. eds, Eds. Amsterdam: Noth-Holland, 1985, vol. 5.

[4] M. B. Priestley, *Non-Linear and Non-Stationary time Series Analysis*. London: Academic Press, 1988.

[5] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, pp. 35–62, 1998.

[6] X. Yao, "Evolving artificial neural networks," in *Proceedings of IEEE*, vol. 87, no. 9, pp. 1423–1447, September 1999.

[7] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural Networks*, vol. 2, pp. 359–366, 1989.

[8] M. Y. Kiang, "A comparative assessment of classification methods", *Decision Support Systems*, 35, pp. 441-454, 2003.

[9] [A. K. Jain, R. P. W. Duin, J. Mao, "Statistical pattern recognition: A review". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22(1), pp.4-37, 2000.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd edition, Prentice Hall, New York, USA, 1998.

[11] N. Ampazis, and S. J. Perantonis, "Two highly efficient second-order algorithms for training feedforward networks," *IEEE Transactions on Neural Networks*, Vol. 13, pp. 1064-1073, September, 2002.

[12] A. L. Arnaud, *A Hybrid Approach for Artificial Neural Networks Optimization on Time Series Forecasting*, Ph.D. Thesis, Center for Informatics, Federal University of Pernambuco, Recife – PE, Brazil, 2007 (in Portuguese).

[13] T. A. E. Ferreira, G. C. Vasconcelos, and Paulo J. L. Adeodato, "A New Evolutionary Approach for Time Series Forecasting", in *Proc. IEEE Symposium on Computational Intelligence and Data Mining* - CIDM2007, Hawaii, 2007.

Use of specific-to-problem kernel functions for time series modeling

Gines Rubio Alberto Guillen Luis J. Herrera Hector Pomares
Ignacio Rojas *

Universidad de Granada, ETSI Informática - Dept. ATC
C/ Periodista Daniel Saucedo sn 18071 Granada - Spain

Abstract.
Although there is a large diversity in the literature related to kernel methods, there are few works which do not use kernels based on Radial Basis Functions (RBF) for regression problems and time series prediction. This type of kernels have a good behaviour in these type of problems due to their generalization capabilities and their smooth interpolation. Thus, this paper presents the settings of specific-to-problem kernels applied to the 3 time series proposed in the competition of the ESTSP 2008. The kernels proposed are based on the analysis of some feature of the data. The weighted k-nearest neighbours with kernel based distance is used to predict and a parallel version of it is utilized to deal with large data sets.

1 Introduction

Kernel methods such us Gaussian Process Regression [1, 2] and Least Square Support Vector Machines (LS-SVMs) [3] have been succesfully applied to regression and function approximation problems. Although they present a good performance, obtaining very accurate results, they present some drawbacks:

- the selection of the kernel function could be difficult
- the optimization of the parameters of the kernel function is computationally intensive, because they require the evaluation of some cross validation procedure or some Bayesian criteria with a complexity of $O(N) = N^3$, where N is the number of training point.
- the generated models could be huge, because they include all training data inside.

In literature, the kernel functions used are almost all variant of RBF [4] [5] [6] [7] and the analysis of the problem is centered in the feature selection. In this paper, we focus on the analysis on the specific data to extract some features to be used as guide to create some kernel functions. The method is based on the example for *Mauna Loa Atmospheric Carbon Dioxide concentration* in pp. 118-120, chapter 5, [1]. But, instead of Gaussian Process or LS-SVM, we use

*This work has been partially supported by the Spanish CICYT Project TIN2007-60587 and Junta Andalucia Project P07-TIC-02768.

a `kernelized` version of the weighted k-nearest neighbours algorithm (that is described in next section), that is less computational intensive.

Thus, the work presented in this paper illustrates the application of a kernel method for function approximation, specializing the kernel functions to the time series proposed in the ESTSP 2008 competition. The rest of the paper is organized as follows: Section 2 will briefly introduce the weighted K-nearest neighbour algorithm and the modification it need to use kernel based instead of Euclidean distance, Section 3 will present each kernel adapted to each time series analyzing the perfomance obtained and, in Section 4, conclusions will be drawn.

2 Weighted K nearest neighbours algorithm for regression

The k nearest neighbours (k-NN) is a simple algorithm that has been usually applied to classfication tasks [8] although some adaptations of this algorithm have been applied also to regression problems [9, 10].

Given a set of function samples $(\vec{x}_i, y_i), i = 1, \ldots, N$, where $x \in R^d$ and $y_i \in \mathbb{R}$, let $X = [\vec{x}_i], i = 1...N$ be a set where distance function D has been defined over it. The weighted K nearest neighbor algorithm for regression is able to compute the output for a given new input $\vec{x}_i$ as:

$$\hat{y} = \frac{\sum_{i=0}^{k} y_i^{\hat{x}} w\left(\hat{x}, x_i^{\hat{x}}\right)}{\sum_{i=0}^{k} w\left(\hat{x}, x_i^{\hat{x}}\right)}, \tag{1}$$

$$where\ w\left(\hat{x}, x_i^{\hat{x}}\right) = \left[1 - \frac{D\left(\hat{x}, x_i^{\hat{x}}\right)^2}{D\left(\hat{x}, x_{k+1}^{\hat{x}}\right)^2}\right]^2. \tag{2}$$

where $x_i^{\hat{x}}$ is the i-th neareast training point to $\vec{x}$ according to the distance function D and $y_i^{\hat{x}}$ its corresponding output. As the formulation shows, the method is based on the values of a predefined distance function and it is independent of any structure the input data could have. Therefore, the weighted K-nn can be seen as a kernel method where the distance function plays the role of kernel. The most commonly used distance measure is the Euclidean.

A kernel function k is defined as a scalar product of the inputs after their projection (Φ) from the *input space* to a *feature space* [11]. Thus a kernel can be defined as:

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \tag{3}$$

The quadratic norm of two points in feature space can be written in term of the values of the kernel function:

$$\begin{aligned} \|\Phi(x) - \Phi(x')\|^2 &= \langle \Phi(x), \Phi(x) \rangle + \langle \Phi(x'), \Phi(x') \rangle - 2 \langle \Phi(x), \Phi(x') \rangle \\ &= k(x,x) + k(x',x') - 2k(x,x') \end{aligned} \tag{4}$$

so the adaptation of the algorithm to work in feature space is quite simple and is obtained by redefining the distance measure D as:

$$D\left(x, x'\right)^2 = k(x,x) + k(x',x') - 2k(x,x') \tag{5}$$

So we only need to substitute in the equations 2 the quadratic terms of distance by the expression 5 to get the kernel-distance-based version of the weighted k-NN algorithm for regression.

3 The series of the ESTSP 2008 competition

Once the algorithm used to predict has been introduced, this section will describe the methodology used to create each kernel adapted to the time series as well as will present the results obtained in the different cases. Note that in the tables we use the Normalized Root Mean Square Error (nrmse, $\sqrt{mse/\sigma_y^2}$) instead of mse to present the results independently from scales.

3.1 Methodology to create, optimize parameters and choose the kernels

Following the same procedure used in [1], we will use the indeces of the time series, $t = 1 \cdots N$, as inputs for each one of the models. Each time series will be predicted using 3 kernels created by combining in sums of the kernel functions that are defined in Equations 6, 7, 8, and 9, that are classical kernels of the literature.

1. λ-Periodic kernel:

$$k_{\lambda-periodic}(x_i, x_j; \{\theta_1, \theta_2\}) = \theta_1 \exp\left(-2\frac{sin(\frac{\pi}{\lambda} * (x_i - x_j))^2}{\theta_2^2}\right). \tag{6}$$

   where λ is a fixed period length.

2. Gaussian or RBF kernel:

$$k_{gauss}(x_i, x_j; \{\theta_1, \theta_2\}) = \theta_1 \exp\left(-\frac{1}{{\theta_2}^2}\|x_i - x_j\|^2\right). \tag{7}$$

3. Linear kernel:

$$k_{linear}(x_i, x_j; \{\beta\}) = \langle x_i, x_j \rangle + \beta. \tag{8}$$

4. Rational Quadratic kernel:

$$k_{ratquad}(x_i, x_j; \{\theta, \alpha, \beta\}) = \theta\left(1 + \frac{\|x_i - x_j\|^2}{2\alpha\beta^2}\right)^{-\alpha} \tag{9}$$

To design the kernels, the first objective is to identify the possible periodicity of each time series. For each period of length λ detected a $\lambda - periodic$ (Eq. 6) term is added to the created kernels. Once done, we create combinations of the previously named terms with 7, 8, and 9. So for each series, 3 candidate kernels are created.

The parameters for each kernel must be set. In order to obtain the value for these parameters, the Leave-One-Out (LOO) Error of weighted k-NN algorithm were optimized using a variable neighbour search (VNS) [12]. The use of the VNS instead of the classical conjugate gradient is because there is no posibility of derivating the error function due to the use of the weighted k-NN. It is also required to set a value k for the weighted k-NN algorithm and the value assigned to it was the one recommended by the literature, this is, $k = 10$.

The following subsections specifies these formulations for each of the three time series as well as the results of the simulations.The computers used for the execution were a personal computer with a AMD Turion of 64 bits of 2.2 GHz and 2 GB of RAM running Matlab under linux for series 1 and 2, and a cluster of 8 nodes with an AMD Opteron of 64 bits of 2.6 GHz processor with 2 GB of RAM each node. The use of a parallel computer was needed due to the large size of the data set of the third time series.

3.2 Time series 1

The first time series has 354 values and it is requested to predict the next 18 values. The data set also contained 2 exogenous variables that could be used or not according to the competition rules. We decided to use the Automatic Relevance Determination (ARD) as a way to weight automatically the importance of each input (it is a wrapper method). The ARD consists in the application of kernels as the ones defined in the following equations:

$$k_{gaussARD}(x_i, x_j; \{\theta_1, \Sigma\}) = \theta_1 \exp\left(-\sum_{d=1}^{D} \frac{(x_i^d - x_j^d)^2}{{\sigma_d}^2}\right). \tag{10}$$

$$k_{linearARD}(x_i, x_j; \{\Sigma\}) = \sum_{d=1}^{D} \sigma_d x_i^d x_j^d. \tag{11}$$

$$k_{ratquadARD}(x_i, x_j; \{\theta, \alpha, \Sigma\}) = \theta \left(1 + \frac{1}{2\alpha} \sum_{d=1}^{D} \frac{(x_i^d - x_j^d)^2}{{\sigma_d}^2}\right)^{-\alpha} \tag{12}$$

These type of kernels have a parameter that scales the input on each dimension, so the value of these parameters (i.e. the importance of each input) is determinated while training.

For the first time series there is $N = 354$ and the prediction horizont is $h = 18$. As said before, we use the indices of the time series as inputs, so to predict next h values we only need to calculate the output for the inputs $N+1, ..., N+h$. To incorporate the exogeneous variables to inputs, the next 18

values of them are needed. So we apply the same methodology of kernel creation, and optimization of kernel parameters to use with weighted k-NN to get theses values. Each exogeous variable is quite similar, having a period of length 12, so we used the following kernels:

1. $k_1(x_i, x_j; \Theta_1) = k_{linear}(x_i, x_j; \Theta_1) + k_{12-periodic}(x_i, x_j; \Theta_1)$
2. $k_2(x_i, x_j; \Theta_2) = k_{gauss}(x_i, x_j; \Theta_2) + k_{12-periodic}(x_i, x_j; \Theta_2)$
3. $k_3(x_i, x_j; \Theta_3) = k_{ratquad}(x_i, x_j; \Theta_3) + k_{12-periodic}(x_i, x_j; \Theta_3)$

The optimization of the kernel parameters is done with 50 iterations of VNS as described in section 3.1. The results of the training can be shown in the tables 1 2.

kernel	nrmse-loo	time
1	3.234e+00	3.052e+01
2	4.338e+00	2.522e+01
3	4.494e+00	3.578e+01

Table 1: Final nrmse-loo values and time (seconds) of training for the exogeneous variable 1 of first series.

kernel	nrmse-loo	time
1	1.938e+00	2.956e+01
2	3.934e+00	2.864e+01
3	1.926e+00	3.616e+01

Table 2: Final nrmse-loo values and time (seconds) of training for the exogeneous variable 2 of first series.

The best kernel (with its parameters) according to mse-loo of training is used in weighted k-NN to predict the next 18 values of each exogeneous variable.

So we have for each value of the time series to predict y_t an input of three dimension $x_t = (t, ex1_t, ex2_t)$, where t is a time index, $ex1_t$ and $ex2_t$ the values of exogenous variables in time t. The kernels evaluated were the result of adding to the ARD kernels 10, 11 and 12 a term that reflect a periodicity of the series of length 12 (this term only works with the time index input). This kernels can be shown below:

1. $k_1(x_i, x_j; \Theta_1) = k_{12-periodic}(x_i, x_j; \Theta_1)$
   $+ k_{gaussARD}(x'_i, x'_j; \Theta_1)$
2. $k_2(x_i, x_j; \Theta_2) = k_{12-periodic}(x_i, x_j; \Theta_2)$
   $+ k_{linearARD}(x'_i, x'_j; \Theta_2)$
3. $k_3(x_i, x_j; \Theta_3) = k_{12-periodic}(x_i, x_j; \Theta_3)$
   $+ k_{ratquadARD}(x'_i, x'_j; \Theta_3)$

The results obtained are shown in Table 3 and the real output and the output of the kernel are represented in Figure 1.

kernel	nrmse-loo	time
1	1.290e+00	5.131e+01
2	2.314e+00	6.593e+01
3	2.314e+00	8.223e+01

Table 3: Final nrmse-loo values and time (seconds) of training for the first series.

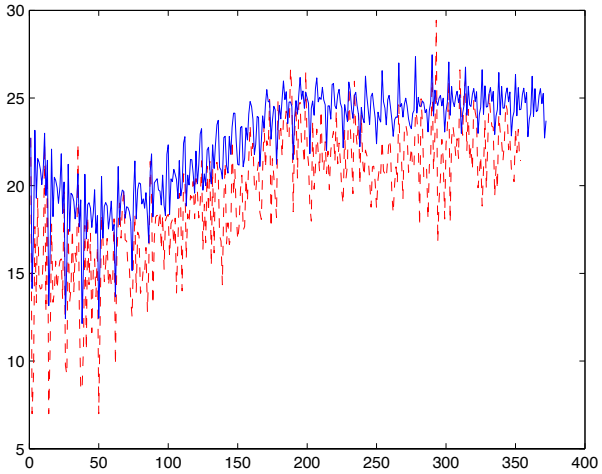


Fig. 1: Series 1 (dashed) and output of weighted k-nn for the selected kernel

As these results show, the prediction is not as accurate as desired so new kernels that use only the time index were used for a new training:

1. $k_1(x_i, x_j; \Theta_1) = k_{linear}(x_i, x_j; \Theta_1) + k_{12-periodic}(x_i, x_j; \Theta_1)$
2. $k_2(x_i, x_j; \Theta_2) = k_{gauss}(x_i, x_j; \Theta_2) + k_{12-periodic}(x_i, x_j; \Theta_2)$
3. $k_3(x_i, x_j; \Theta_3) = k_{ratquad}(x_i, x_j; \Theta_3) + k_{12-periodic}(x_i, x_j; \Theta_3)$

obtaining the results shown in Tabla 4 and the output of the best model is shown in Figure 2.

3.3 Time series 2

The data provided for the second time series consist in 1300 values and the aim is to predict the next 100. The time series 2 has two periods of lenght 354 and

kernel	nrmse-loo	time
1	1.963e-01	6.052e+01
2	1.901e-01	6.189e+01
3	1.333e-01	7.258e+01

Table 4: Final nrmse-loo values and time (seconds) of training for the first series.

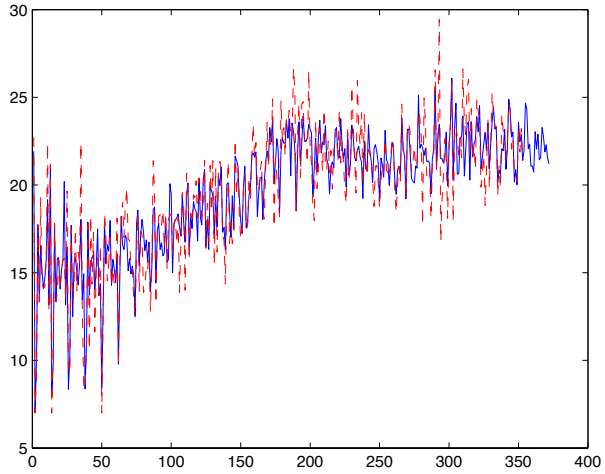


Fig. 2: Series 1 (dashed) and output of selected model

7 respectively, therefore the following kernels, whose parameters were optimized in the same way than in the previous time series, were applied:

1. $k_1(x_i, x_j; \Theta_1) = k_{linear}(x_i, x_j; \Theta_1) + k_{7-periodic}(x_i, x_j; \Theta_1)$
   $+ k_{364-periodic}(x_i, x_j; \Theta_1)$
2. $k_2(x_i, x_j; \Theta_2) = k_{gauss}(x_i, x_j; \Theta_2) + k_{7-periodic}(x_i, x_j; \Theta_2)$
   $+ k_{364-periodic}(x_i, x_j; \Theta_2)$
3. $k_3(x_i, x_j; \Theta_3) = k_{ratquad}(x_i, x_j; \Theta_3) + k_{7-periodic}(x_i, x_j; \Theta_3)$
   $+ k_{364-periodic}(x_i, x_j; \Theta_3)$

Table 5 shows the results of the executions and Figure 3 depicts the best model obtained using the LOO.

3.4 Time series 3

For last time series, 31614 values were given with the intention of prediction the next 200. The data were analyzed as in the previous cases showing a main period of length 8736 and many smaller periods. We decided to use the bigger

kernel	nrmse-loo	time
1	2.017e-01	2.521e+02
2	1.425e-01	2.805e+02
3	1.450e-01	3.205e+02

Table 5: Final nrmse-loo values and time (seconds) of training for the second series.

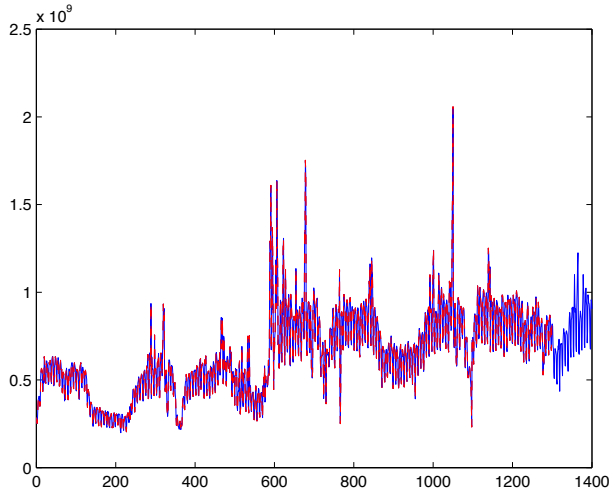


Fig. 3: Series 2 (dashed) and output of selected model

and the smaller, of length 7, periods. One of the challenges when trying to apply the methodology was the large amount of memory and computational effort that requires to work with 31614 values. Therefore, the algorithm k-NN had to be parallelized in order to make practical it use in this case. In order to speedup the training and reduce the memory requirements, the weighted k-NN was parallelized by the distribution of the data among the different processes. Each node compute the k nearest neighbours of an input to their local data, and after that all nodes shares their results to get the k nearest neighbours to the input of the distributed data set. Although the excution time was significantly decreased, only 10 iterations of the VNS algorithm were made. For this time series the kernels defined were:

1. $k_1(x_i, x_j; \Theta_1) = k_{linear}(x_i, x_j; \Theta_1) + k_{7-periodic}(x_i, x_j; \Theta_1)$
   $+ k_{8736-periodic}(x_i, x_j; \Theta_1)$
2. $k_2(x_i, x_j; \Theta_2) = k_{gauss}(x_i, x_j; \Theta_2) + k_{7-periodic}(x_i, x_j; \Theta_2)$
   $+ k_{8736-periodic}(x_i, x_j; \Theta_2)$

3. $k_3(x_i, x_j; \Theta_3) = k_{ratquad}(x_i, x_j; \Theta_3) + k_{7-periodic}(x_i, x_j; \Theta_3)$
   $+ k_{8736-periodic}(x_i, x_j; \Theta_3)$

As for the previous time series, the results of the executions are in Table 6 and the output of the best model is represented in Figures 4.

kernel	nrmse-loo	time
1	2.281e+00	6.240e+04
2	1.858e+00	8.756e+04
3	1.691e+00	1.076e+05

Table 6: Final nrmse-loo values and time (seconds) of training for the third series.

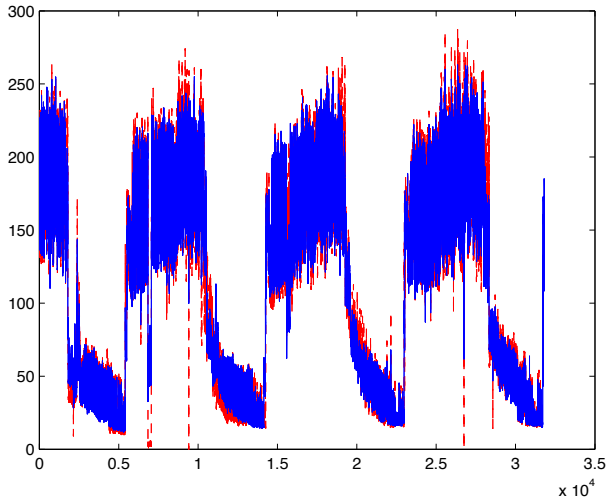


Fig. 4: Series 2 (dashed) and output of selected model

4 Conclusions

The use of specific-to-problem kernels are strongly recommend in the literature of kernel methods [1], but it require a deep understanding of the studied problem and the kernels itself to be effective. This paper has presented the design of specific kernel methods to predict 3 time series, during the experiments was shown how a specialization and custom design of a kernel can be made. The results obtained for each time series were coherent with the previous analysis of them showing how the kernels were able to model the tendencies and periodicities of each time series. Another element to remark is the utility of the parallel

programming as an almost obligated use in order to obtain accurate results in a reasonable time with big data sets.

References

[1] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[2] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer, 1998.

[3] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing, Singapore, 2002.

[4] K.-R. Müller, A.J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Using support vector machines for time series prediction, 2000.

[5] T. Van Gestel, J.A.K. Suykens, D.-E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework. *Neural Networks, IEEE Transactions on*, 12(4):809–821, 2001.

[6] R. van Brakel Thiessen, U. Using support vector machines for time series prediction. *Chemometrics and Intelligent Laboratory Systems*, 69:35–49, 2003.

[7] Sofiane Brahim-Belhouari and Amine Bermak. Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47(4):705–712, November 2004. available at http://ideas.repec.org/a/eee/csdana/v47y2004i4p705-712.html.

[8] B. V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques*. Los Alamitos: IEEE Computer Society Press, 1990, 1990.

[9] A.T. Lora and J.R. Santos. A comparison of two techniques for next-day electricity price forecasting. In *Third International Conference on Intelligent Data Engineering and Automated Learning, LNCS*, volume 2412, pages 384–390, 2002.

[10] Antti Sorjamaa, Jin Hao, and Amaury Lendasse. Mutual Information and k-Nearest Neighbors Approximator for Time Series Prediction. *Lecture Notes in Computer Science*, 3697:553–558, 2005.

[11] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[12] N. Mladenović and P. Hansen. Variable neighborhood search. *Comps. in Opns. Res.*, 24:1097–1100, 1997.

Tabu Search with Delta Test for Time Series Prediction using OP-KNN

Dušan Sovilj, Antti Sorjamaa and Yoan Miche

Helsinki University of Technology - Department of Information and Computer Science
P.O.Box 5400, 02150 HUT - Finland

Abstract. This paper presents a working combination of input selection strategy and a fast approximator for time series prediction. The input selection is performed using Tabu Search with the Delta Test. The approximation methodology is called Optimally-Pruned k-Nearest Neighbors (OP-KNN), which has been recently developed for fast and accurate regression and classification tasks. In this paper we demonstrate the accuracy of the OP-KNN with the Tabu Search using the ESTSP 2008 Competition datasets.

1 Introduction

The amount of information is increasing rapidly in many fields of science. It creates new challenges for storing the massive amounts of data as well as to the methods, which are used in the data mining processes. In many cases, when the amount of data grows, the computational complexity of the used methodology also increases. In this paper, we combine a clever way of selecting the input data and a very fast approximation method for time series prediction task.

In the time series prediction the problem is prediction of future values based on appropriate number of previous values in the series. This number can be decided empirically with trial and error and/or by looking at the periodicity of the series itself. Since the approximation model we are using is very fast, we used both conditions upon decisions of good regressor sizes for the ESTSP 2008 competition datasets.

Further performance improvement can be achieved with adequate input selection strategy. Delta Test is a nonparametric noise estimator which can be used for that purpose [1]. Since exhaustive search is the only way to obtain global optimum, but it is impractical to use with large datasets, a common strategy is to rely on suboptimal Forward-Backward selection. To overcome the problem of local optima we use Tabu Search, which incorporates additional memory and can be extended with various heuristics.

After variable selection with Tabu Search using Delta Test as input selection criterion, actual prediction is done with Optimally-Pruned k-Nearest Neighbors (OP-KNN). It is based on the inspiring work on Extreme Learning Machine (ELM) by Guang-Bin Huang *et al.* in [2] and on Optimally-Pruned ELM by Yoan Miche *et al.* in [3]. Unlike the two previous methods, the OP-KNN is deterministic and simple, while still retaining the speed of the ELM and the accuracy of the OP-ELM. Since we are predicting several values ahead, we focus

on using Direct Strategy [4] over Recursive for better long-term prediction accuracy. The Direct Strategy includes building a separate model for every time step. This also includes separate variable selection.

Section 2 presents the Tabu Search and the Delta Test estimation methods. In Section 3, the OP-KNN methodology is explained and finally, Section 4 presents the experimental results using ESTSP 2008 Competition benchmarks.

2 Tabu Search (TS)

Tabu Search [4] resembles the Forward-Backward Selection procedure [5], which only continues along selections that improve the objective function. Tabu Search tries to overcome the problem of local optima with additional use of memory and by considering solutions which do not improve the objective function. Memory is used to keep track of already visited solutions and those are prevented from being explored again, as they are in *tabu* state, hence the name of the method. On the other hand, there are no guarantees that the TS will find the global optimum, but results found with it are usually better than the ones obtained with plain greedy approach.

Consider the optimization problem $f(x), x \in X$, where $f(x)$ is the objective or cost function and x is one possible solution for the problem. The $f(x)$ in our case is the Delta Test. In the context of Tabu Search the neighborhood of solution x, denoted $N(x)$, plays an important role. $N(x)$ is a set of solutions reachable from x via transition moves.

The weakening of search criterion, moving to solutions with worse f values, introduces a problem of cycles, and this is *one* of the reasons for using memory. One part of memory is used to prevent already visited solutions in $N(x)$ to be explored again. Memory is divided into two parts, short term and long term memory. The short term memory strategies are focused on directly modifying the set of neighbors $N(x)$. The long term memory strategies can include solutions, which are not part of the $N(x)$ and which can generate solutions based on the attributes of the good solutions.

The simplest approach is to use only the short term memory, to mark solutions already visited as tabu and to choose solutions from $N(x)$ that have the best f value. This approach is called Simple Tabu Search [4]. The neighborhood $N(x)$ will be influenced by the contents of the memory and the set will change during the search. This is why the TS is sometimes called *dynamic neighborhood search technique*. The size of solutions would make TS impractical to use and it is much more efficient to store transition moves or some other attributes of vector x.

In the case of variable selection, the neighborhood is easily formed by having two solutions being neighbors if they *disagree* on the selection of exactly one variable v_k. The move that links these two neighbors is just flipping the state of the variable v_k. The size of the search space grows exponentially with the dimensionality of the inputs and for a dataset with a lot of variables it is very difficult to find optimal selection.

2.1 Delta Test (DT)

Delta Test is a Nonparametric Noise Estimator based on a nearest neighbor principle. The nearest neighbor of a point is defined as the (unique) point, which minimizes a distance metric to that point. Distance metric is usually the Euclidean distance, but other metrics can also be used.

In function approximation, we have a set of input points $(x_i)_{i=1}^N$ and associated scalar outputs $(y_i)_{i=1}^N$. The assumption is that there is a functional dependence between them, but with an additive noise term $y_i = f(x_i) + \epsilon_i$. The function f is assumed to be smooth, and the noise terms ϵ_i are independent and identically distributed with zero mean. Noise variance estimation is the study of how to give an *a priori* estimate for $\mathrm{Var}(\epsilon)$ given some data *without* considering any specifics of the shape of f.

Using the previous notation, the Delta Test is usually written as

$$\mathrm{Var}(\epsilon) \approx \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_{P(i)})^2,$$

where $P(i)$ defines the nearest neighbor of x_i in the input space. Hence, using the DT, we consider the estimate of the noise as the mean of the differences in the outputs associated with neighboring points (divided by 2). This is a well-known and widely used estimator, and it has been shown for example in [6] that the estimate converges to the true value of the noise variance in the limit $N \to \infty$.

3 Optimally-Pruned *k*-Nearest Neighbors (OP-KNN)

The OP-KNN is similar to the OP-ELM [3], which is an original and efficient way of training a feedforward neural network. The three main steps of the OP-KNN are summarized in Figure 1.

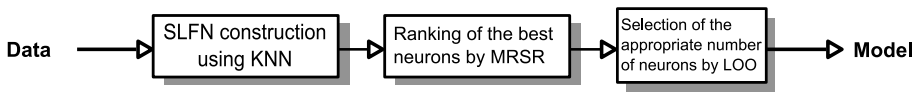


Fig. 1: The three steps of the OP-KNN algorithm.

3.1 Single hidden Layer Feedforward Neural Network (SLFN)

The first step of the OP-KNN algorithm is similar to the original ELM: building of a Single hidden Layer Feedforward Neural network. The idea of the ELM has been proposed by Guang-Bin Huang *et al.* in [2].

In the context of a single hidden layer network, let us denote the weights between the hidden layer and the output by $\mathbf{b}$. Activation functions used with the OP-KNN differ from the original SLFN choice since the original sigmoid activation functions of the neurons are replaced by the k-Nearest Neighbors,

hence the name OP-KNN. For the output layer, the activation function remains as a linear function.

A theorem proposed in [2] states that the output weights $\mathbf{b}$ can be computed from the hidden layer output matrix $\mathbf{H}$: the columns $\mathbf{h_i}$ of $\mathbf{H}$ are the corresponding output of the k-nearest neighbors.

Finally, the output weights $\mathbf{b}$ are computed by $\mathbf{b} = \mathbf{H}^{\dagger}\mathbf{y}$, where $\mathbf{H}^{\dagger}$ stands for the Moore-Penrose inverse [7] and $\mathbf{y} = (y_1, \ldots, y_M)^T$ is the output.

The only remaining parameter in this process is the initial number of neurons N of the hidden layer.

3.2 *k*-Nearest Neighbors (KNN)

The k-Nearest Neighbors model is a very simple, but powerful tool. It has been used in many different applications and particularly in classification tasks. The key idea behind the KNN is that similar training samples have similar output values. In the OP-KNN, the approximation of the output is a weighted sum of the outputs of the k-nearest neighbors as

$$\hat{y}_i = \sum_{j=1}^{k} b_j y_{P(i,j)},$$

where $\hat{y}_i$ represents the output estimation, $P(i,j)$ is the index of the j^{th} nearest neighbor of sample $\mathbf{x}_i$ and b is the result of the Moore-Penrose inverse introduced in the previous section.

3.3 Multiresponse Sparse Regression (MRSR)

For the removal of the useless neurons of the hidden layer, the Multiresponse Sparse Regression proposed by Timo Similä and Jarkko Tikka in [8] is used. It is an extension of the Least Angle Regression (LARS) algorithm [9] and hence is actually a variable ranking technique, rather than a selection one.

An important detail shared by the MRSR and the LARS is that the ranking obtained is exact in the case where the problem is linear. In our case this is true, since the neural network built in the previous step is linear between the hidden layer and the output. Therefore, the MRSR provides the exact ranking of the neurons for our problem.

MRSR is hence used to rank the kernels of the model: the target is the actual output y_i while the "variables" considered by MRSR are the outputs of the k-nearest neighbors.

3.4 Leave-One-Out (LOO)

Since the MRSR only provides a ranking of the kernels, the decision over the actual best number of neurons for the model is taken using a Leave-One-Out method. One problem with the LOO error is that it can be very time consuming

to calculate if the dataset has a high number of samples. Fortunately, the PRESS (PREdiction Sum of Squares) statistics provides a direct and exact formula for the calculation of the LOO error for linear models [10, 11]. The LOO error using the PRESS statistics can be written as

$$\epsilon^{\text{PRESS}} = \frac{y_i - \mathbf{h}_i \mathbf{b}}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T},$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T \mathbf{H})^{-1}$ and $\mathbf{H}$ as the hidden layer output matrix defined in Section 3.1.

The final decision over the appropriate number of neurons for the model is taken by minimizing the LOO error versus the number of neurons used (properly ranked by the MRSR already).

4 Experiments

In the experiments, we are using the datasets of the ESTSP 2008 Competition. The section is divided into three subsections and each presents one dataset. The first dataset is explained more deeply and the latter two are less detailed.

What is common to all datasets is the selection of the initial regressor sizes for the TS. The OP-KNN was used with several different regressor sizes with all the variables and the ones with the smallest LOO errors were selected for each dataset. For the dataset 3, also the periodicity of the data was taken into account. The inputs and outputs are normalized before running the Tabu Search.

After the selection of the initial regressor sizes, the Tabu Search with the Delta Test as the cost function was used. The set of variables, which had the lowest delta value, was selected individually for each prediction step. This means, that the prediction strategy used was the Direct Strategy [5], where each prediction step needs its own selection of variables and its own model to be built.

Tabu Search was implemented as a Simple Tabu Search, where only recent moves, or changes of variables, were kept in memory. The memory was set to 1/5 of the dimensionality of samples. For example, for sample with 10 dimensions, the memory size would be set to 2 so that only the last 2 applied moves are considered tabu.

Stopping conditions for TS varied from one dataset to another due to the different sizes of the initial regressors. After the selection of the variables, the final model was obtained using the OP-KNN. All the data was used in the training after the preliminary tests showed that the LOO and the test errors behaved in the same way with each other.

4.1 Dataset 1: Multidimensionality

Dataset 1 consists of 3 time series with one of them being the target series, and each of them have 354 values. All 3 series are shown in Figure 2 with the target one being on the top of the figure.

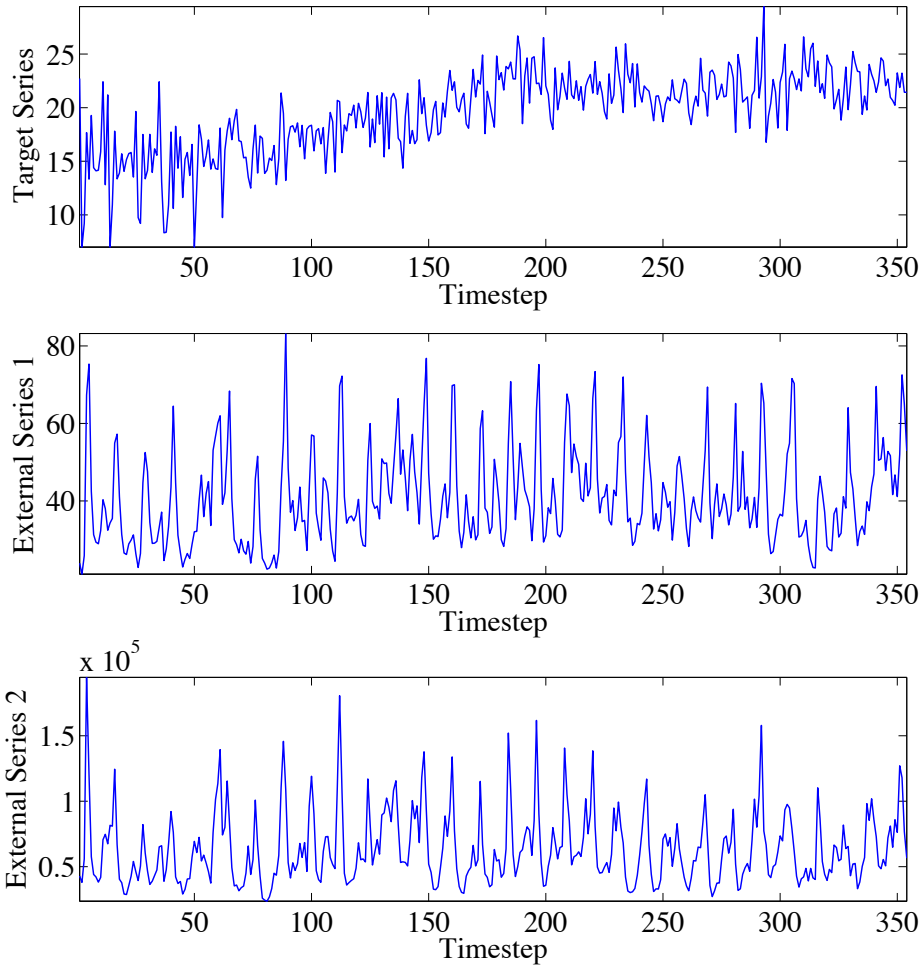


Fig. 2: Dataset 1. Top one is the series itself and the two lower ones are the external series.

First step was the removal of the trend present in the target series by transforming it using first order difference. The OP-KKN results showed that regressor of size 20 is the optimum for the target series. From both external series we chose 12 variables as extra inputs, based on the autocorrelations with the target series. As the number of selected variables is lower than for the target series, some of the values from the beginning of the two external series are removed. The alignments of two external 12 variables are done in such a way so that we always use the latest measurements as possible. Thus, the final regressor size was 44 and the final number of training samples was 316.

Since the search space has $2^{44} - 1$ instances, the TS was run 24 times from

random starting points with 1 hour each. Roughly 10^6 solutions were examined in every run, a very small percentage of the whole space, and the found delta value is probably not the global optimum. Figure 3 shows that the variable selection with Tabu Search using Delta Test does improve overall performance of the OP-KNN with respect to the LOO error. In the same figure the Delta values found by the TS are also shown.

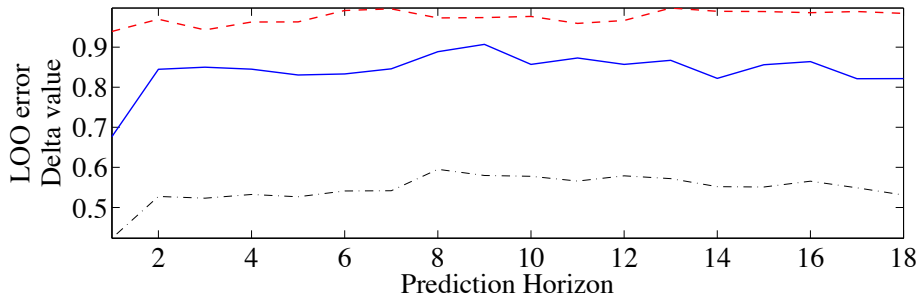


Fig. 3: LOO errors of 18 OP-KNN models with all variables (dashed line) and the selected variables with the Tabu Search (solid line). For comparison, also the Delta Test values found with the Tabu Search (dot-dashed line).

The Delta Test gives an estimate that one can achieve in terms of the training error without overfitting the model. Using the TS the LOO error was decreased more than 25% with respect to the Delta value. The Delta values are fairly high for this dataset (over 0.5), which suggests that the target series is very noisy, and therefore, hard to predict.

The predictions for the 18 steps ahead are shown in Figure 4.

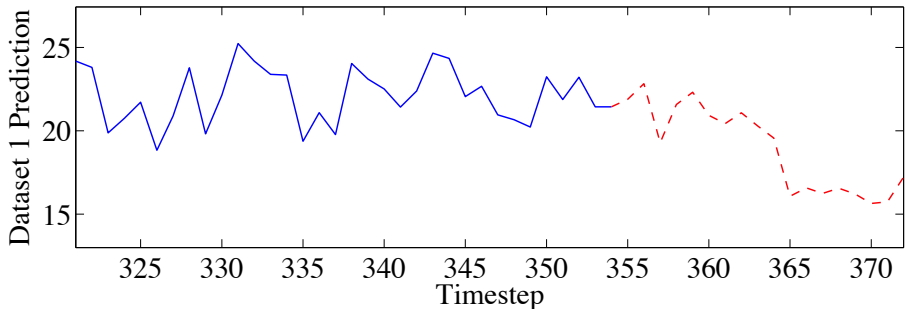


Fig. 4: Prediction of 18 steps for target series of Dataset 1. Solid line represents last known values and dashed one the prediction.

4.2 Dataset 2: Easy Sailing

Dataset 2 consists of 1300 values and it is shown in Figure 5.

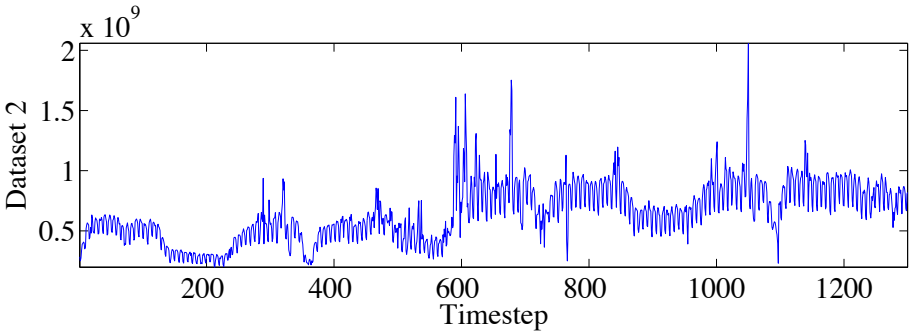


Fig. 5: Dataset 2.

Because the data seems to have two distinct levels, which changes just before the timestep 600, we decided to normalize the dataset in two parts: first part including the values before the 600 and the second part after it. At the same time, we have the whole set normalized for the TS.

The initial regressor size for the TS was selected to 20, which leaves us with roughly 1180 samples for the TS phase and the training of the OP-KNN. The stopping criterion for the TS was set to 10 percent of the search space, where the running times were around 1.5 hours for each prediction step.

The predictions for 100 steps ahead are shown in Figure 6.

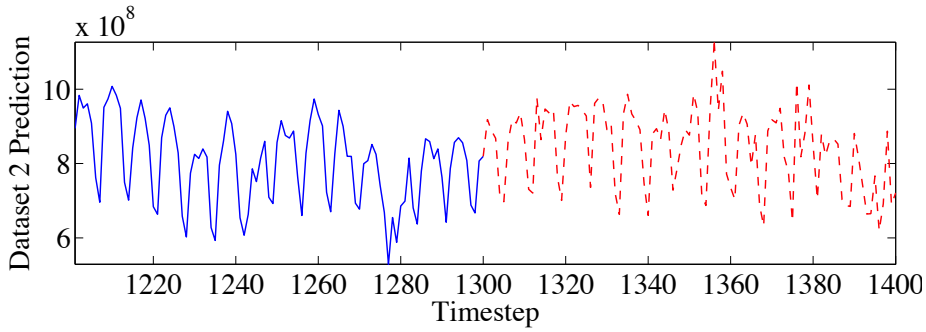


Fig. 6: Dataset 2 prediction of 100 steps ahead using the OP-KNN with Tabu selected variables. Solid line represents the known values and the dashed one is the prediction.

4.3 Dataset 3: Measurements Aplenty

Data 3 includes vast amount of data, more than 31 000 samples, shown in Figure 7.

Since the dataset had a clear periodicity of 24, we decided to cut the dataset into slices of 24 rather than build the traditional regressor by taking the samples

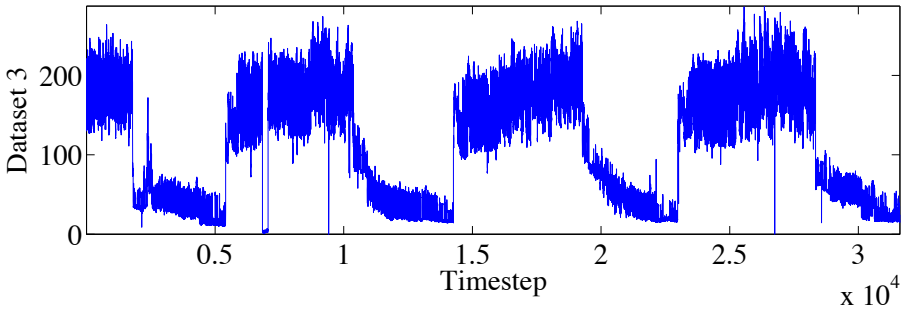


Fig. 7: Dataset 3.

using moving window through the dataset. After the slicing, our dataset has roughly 1300 samples of 24 dimensions.

Because the dataset exhibits heavy long-term seasonality, we decided to use sample-wise normalization and predict the normalized series, the means and the standard deviations separately. It means that in order to get 200 step ahead prediction for the original dataset, we needed to predict 200 steps for the normalized samples and 9 steps ahead for the means and standard deviations to be added and multiplied with the normalized prediction.

In the prediction of the normalized samples, we used two previous days as the initial regressor size for the TS. Hence, we have a regressor size 48. For the means and standard deviations the regressor size was selected to be 15. For normalized samples we had only 1 run with TS lasting 6 hours, while for the means and standard deviations the stopping criterion was 10 percent of solution space (same as in Dataset 2).

After similar steps than with the dataset 1, we obtain the prediction for 200 steps shown in Figure 8.

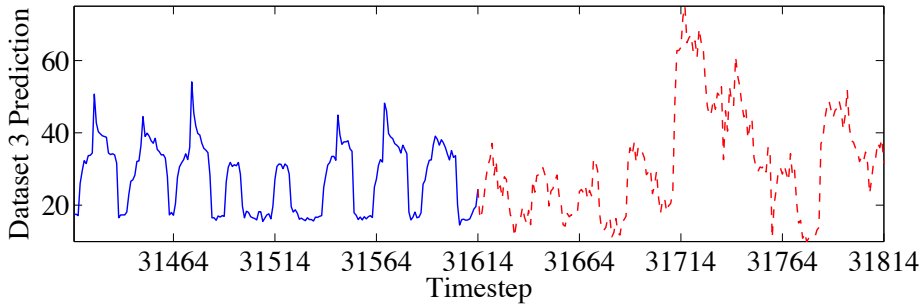


Fig. 8: Dataset 3 with the prediction of 200 steps using the OP-KNN with Tabu selected variables. The solid line denotes the known values and the dashed one the predicted.

5 Conclusions

In this paper we have shown the efficiency of the combination of a clever and adaptable input variable selection method, the Tabu Search, and a fast prediction method, the OP-KNN.

The results are satisfying and obtained with a small calculation time. The Tabu Search can be given a predefined amount of time for its search, which covers the search space better than plain greedy methods. However, it is also able to give acceptable results in reduced time, if necessary.

The OP-KNN is very fast to train and to use in the prediction task. The accuracy of the method can be improved with input selection, which on the other hand will increase the total calculation time. This creates the need for fast input selection methodology.

For further work, the Tabu Search will be improved in terms of speed and used heuristics with better finetuning for different datasets. The OP-KNN methodology will be tested more extensively with different input selection methods as well as with different regression tasks.

References

[1] E. Eirola, E. Liitiäinen, A. Lendasse, F. Corona, and M. Verleysen. Using the delta test for variable selection. In *ESANN 2008, European Symposium on Artificial Neural Networks, Bruges (Belgium)*, April 2008.

[2] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3):489–501, December 2006.

[3] Y. Miche, P. Bas, C. Jutten, O. Simula, and A. Lendasse. A methodology for building regression models using extreme learning machine: OP-ELM. In *ESANN 2008, European Symposium on Artificial Neural Networks, Bruges, Belgium*, April 23-25 2008.

[4] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[5] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16-18):2861–2869, October 2007.

[6] E. Liitiäinen, F. Corona, and A. Lendasse. Nearest neighbor distributions and noise variance estimation. In *ESANN 2007, European Symposium on Artificial Neural Networks, Bruges (Belgium)*, April 25-27 2007.

[7] C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and Its Applications*. John Wiley & Sons Inc, January 1972.

[8] T. Similä and J. Tikka. Multiresponse sparse regression with application to multidimensional scaling. In *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005*, volume 3697/2005, pages 97–102. 2005.

[9] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. In *Annals of Statistics*, volume 32, pages 407–499. 2004.

[10] R.H. Myers. *Classical and Modern Regression with Applications, 2nd edition*. Duxbury, Pacific Grove, CA, USA, 1990.

[11] G. Bontempi, M. Birattari, and H. Bersini. Recursive lazy learning for modeling and control. In *European Conference on Machine Learning*, pages 292–303, 1998.

Long-term prediction of nonlinear time series with recurrent least squares support vector machines

Indir Jaganjac

ArcelorMittal Zenica
Department of Iron Making
Bulevar Kralja Tvrtka 17/I
Bosnia and Herzegovina

Abstract. This paper is about applying recurrent least squares support vector machines (LS-SVM) on three ESTSP08 competition datasets. Least squares support vector machines are used as nonlinear models in order to avoid local minima problems. Then prediction task is re-formulated as function approximation task. Recurrent LS-SVM uses nonlinear autoregressive exogenous (NARX) model to build nonlinear regressor, by estimating in each iteration the next output value, given the past output and input measurements.

1 Introduction

Support Vector Machines (SVM) is a powerful methodology for solving problems in nonlinear classification, function estimation and density estimation [4]. It has been originally introduced within the context of statistical learning theory and structural risk minimization. These methods use quadratic programming to solve convex optimization problems [1]. Least Square Support Vector Machines (LS-SVM) are re-formulation to the standard SVM [6,7]. In this paper, recurrent least squares support vector machines are used as nonlinear models in order to avoid local minima problems [4,8]. The cost function is a regularized least squares function with equality constraints, leading to linear Karush-Kuhn-Tucker systems [4,7,8]. LS-SVMs are closely related to regularization networks and Gaussian processes [6,9]. Accurate prediction of nonlinear time series is very important in many fields: wind power systems, seismology, econometrics, industrial process automation systems, biomedicine, life sciences and etc. The main difficulty is lack of sufficient and necessary information for an accurate prediction. The challenge in the field of time series prediction is the long-term prediction, where typically more than 100 steps ahead ought to be predicted. Long-term prediction methods must solve many problems because of accumulation of errors, noise and perturbations from the environment. This paper is organized as follows. In Section 2, main principle of LS-SVM for nonlinear function estimation is presented. Section 3 presents how recurrent LS-SVM and nonlinear autoregressive exogenous (NARX) models are used for nonlinear regression and prediction. Section 4 presents ESTSP08 3 datasets and results. Section 5 concludes with some final remarks and pointers to further works.

2 Least-squares support vector machines for nonlinear function estimation

For a given training set of N data points $\{x_k, y_k\}$ with x_k as n-dimensional input and y_t as 1-dimensional output, feature space SVM models take the form [3,5]:

$$y(x) = \omega^T \varphi(x) + b,$$

where the nonlinear mapping $\varphi(.)$ maps the input data into a higher dimensional feature space. In least-squares support vector machines (LS-SVM) for nonlinear function estimation, the following optimization problem is formulated:

$$\min_{\omega, e} J(\omega, e) = \tfrac{1}{2}\, \omega^T \omega + \gamma\, \tfrac{1}{2} \sum_{k=1}^{N} e_k^2,$$

subject to equality constraints:

$$y(x) = \omega^T \varphi(x) + b + e_k, \quad k=1, \ldots, N$$

and the solution is:

$$h(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i) + b$$

In the above equations, i refers to the index of a sample and $K(x, x_i)$ is the Kernel function defined as the dot product between the $\varphi(x)^T$ and $\varphi(x)$. In this paper, Gaussian kernels are used:

$$K(x, x_i) = \exp \{ \|x-x_i\|^2 / \sigma^2 \}$$

The model hyperparameters σ and γ are trained and optimized according to [2,7,8].

3 Recurrent least-squares support vector machines

To predict more than 100 steps ahead values of time series, recurrent least-squares support vector machines can be used [3,5]. It uses the predicted values as known data to predict the next ones. The recurrent LS-SVM model can be constructed by first making one-step ahead prediction:

$$y_{t+1}' = f_1(y_t, y_{t-1}, \ldots, y_{t-M+1})$$

where M denotes the number of inputs and y_{t+1}' denotes predicted value. The regressor of the model is defined as the vector of inputs: $y_t, y_{t-1}, \ldots, y_{t-M+1}$. To predict the next value, the same model is used:

$$y_{t+2}' = f_1(y_{t+1}', y_t, y_{t-1}, \ldots, y_{t-M+1})$$

In this equation, the predicted value of y_{t+1}' is used instead of the true value, which is unknown. Then, for the H-steps ahead prediction, y_{t+2} to y_{t+H}' are predicted iteratively. When the regressor length M is larger than H, there are M-H real data in regressor to predict H_{th} step. When H exceeds M, all inputs are predicted values.

Nonlinear autoregressive exogenous (NARX) models are built based on nonlinear regression by estimating in each iteration the next output value, given the past output and input measurements. A dataset is converted into a new input (past measurements) by function *windowize*. Prediction is done by the function *predict*, iteratively in recurrent mode, and next output is computed, based on the previous predictions and starting values [2].

4 ESTSP08 times series datasets and results

4.1 Time Series 1

Function estimation is done for the 3_{rd} variable only, which has 354 data points. Then prediction is computed in recurrent mode for the next 18 values. The recurrent LS-SVM model is trained and fine-tuning of hyperparameters σ^2 and γ was done with cross-validation, according to [2].

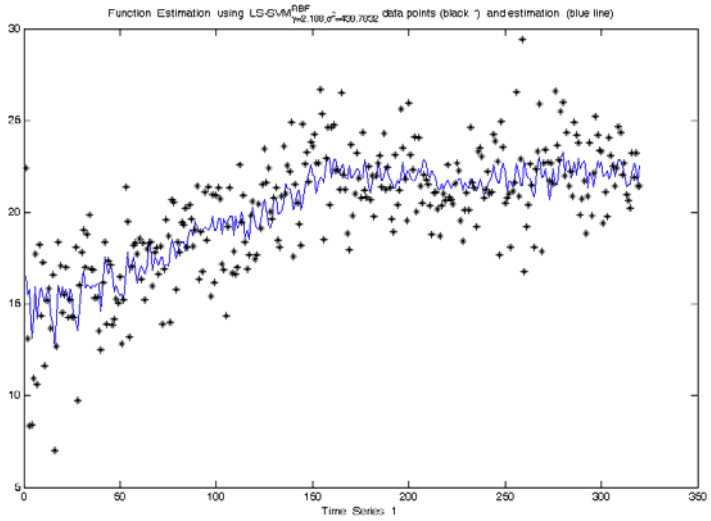


Figure 1: Function Estimation of Time Series 1

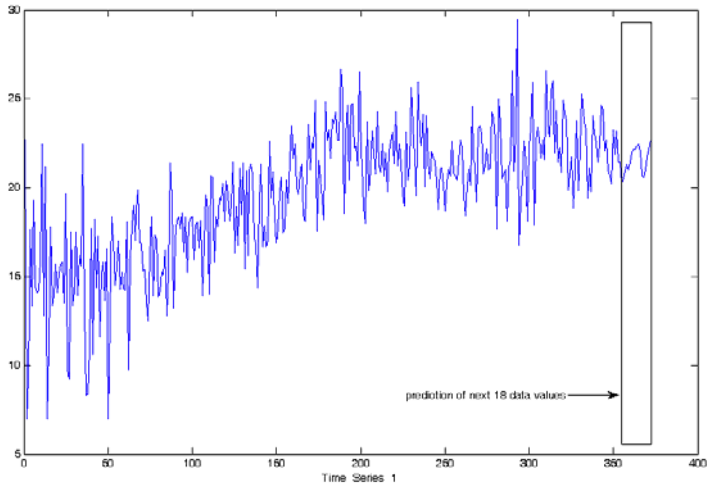


Figure 2: Prediction of next 18 data values for Time Series 1

4.2 **Time Series 2**

Nonlinear function estimation is done for the 1300 data points. Then prediction is computed in recurrent mode for the next 100 data values. The recurrent LS-SVM model is trained and fine-tuning of hyperparameters σ^2 and γ was done with cross-validation, according to [2].

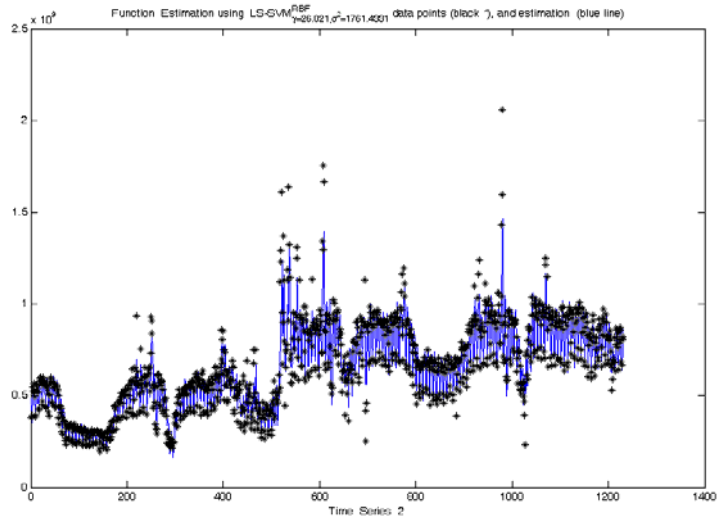


Figure 3: Function Estimation of Time Series 2

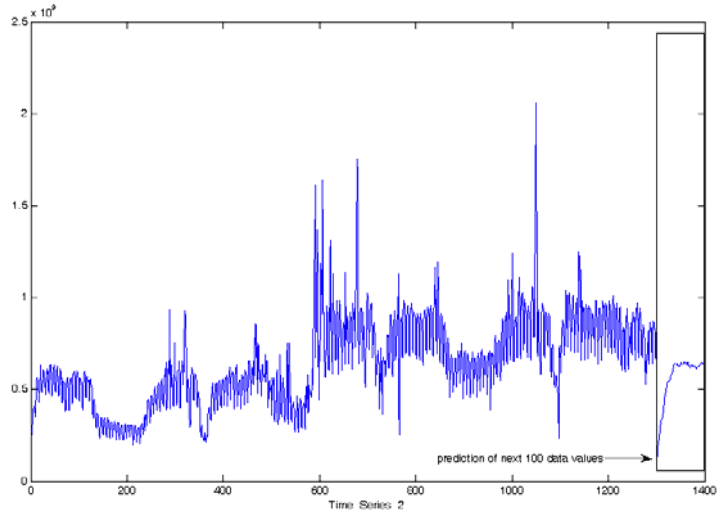


Figure 4: Prediction of next 100 data values for Time Series 2

4.3. **Time Series 3**

Nonlinear function estimation is done for the 3900 data points, which is sufficient for the accurate estimation. Then prediction is computed for the next 200 data values. Figure 5 shows nonlinear function estimation for 3900 data points and Figure 6 shows prediction of next 200 data values. The rationale for truncating the original dataset is that after testing various lower bounds, the best candidate lower bound converged to 3900 data points [10]. This lower bound captured all necessary fluctuations and dynamics in the original dataset. The recurrent LS-SVM model is trained and fine-tuning of hyperparameters σ^2 and γ was done with cross-validation, using Matlab and LS-SVMlab Toolbox. Details can be found in [2].

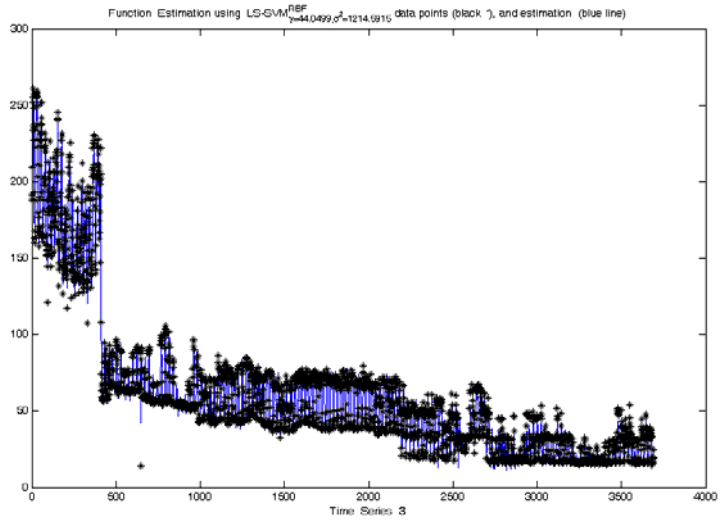


Figure 5: Function Estimation of Time Series 3

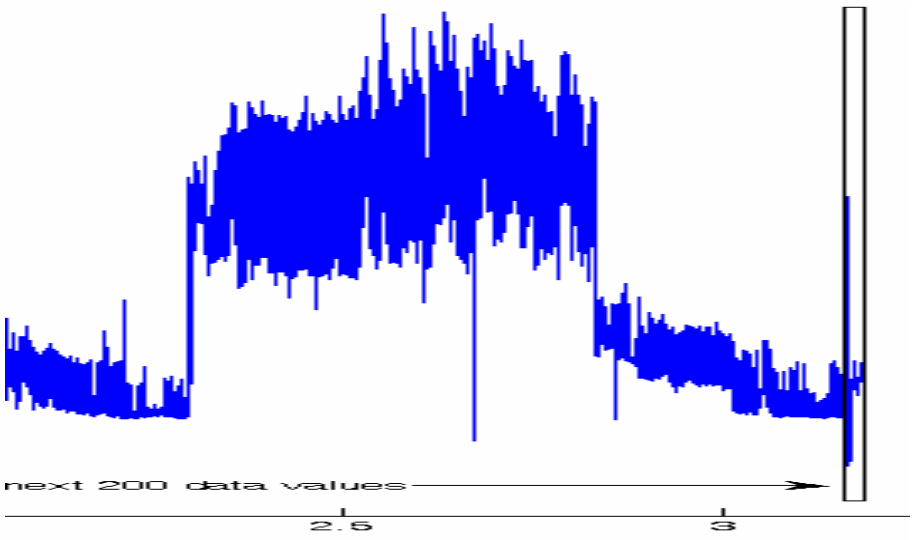


Figure 6: Prediction of next 200 data values for Time Series 3

5 Conclusion

This paper presents solutions for the long-term predictions of ESTSP08 datasets. LS-SVM was chosen for modeling nonlinear time series, because of its ability to avoid local minima problems. The prediction task was re-formulated as the nonlinear function estimation task. The predictions were computed using recurrent least squares support vector machines.

The recurrent least squares support vector machine iteratively predicts next output, based on the previous predictions and starting values. The time series 3 dataset was truncated to 3900 points, which was obtained as satisfactory lower bound for capturing all underlying fluctuations and dynamics in the original dataset.

In further works:

- research on parallelization of recurrent LS-SVM models will be done,
- on-line implementation of recurrent LS-SVM for continuous data analysis of industrial processes and measurements.

6 References

[1] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[2] K. Pelckmans, J.A.K. Suykens, T. Van Gestel, J. De Brabanter, L. Lukas, B. Hamers, B. De Moor, J. Vandewalle, LS-SVM Toolbox User's Guide, ESAT-SCD-SISTA Technical Report 02-145, Katholieke Universiteit Leuven, February 2003.

[3] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, A. Lendasse, Methodology for long-term prediction of time series, Neurocomputing, 2007.

[4] J.A.K. Suykens, Support Vector Machines: A Nonlinear Modeling and Control Perspective, April, 2001.

[5] A. Lendasse, V. Wertz, G. Simon, M. Verleysen, Fast Bootstrap applied to LS-SVM for Long Term Prediction of Time Series, IJCNN '2004 proceedings, July 2004, IEEE pp. 705-710.

[6] C.E. Rasmussen and C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.

[7] J.A.K. Suykens and J. Vandewalle, Recurrent Least Squares Support Vector Machines, IEEE Transactions on Circuits and Systems – I, Vol.47, No.7, July, 2000.

[8] J.A.K. Suykens, J.D. Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines, robustness and sparse approximation, Neurocomputing 48, 2002.

[9] T. Poggio and F. Girosi, Networks for Approximation and Learning, Proceedings of the IEEE, Vol.78, No.9, September 1990.

[10] P. Denbigh, System Analysis & Signal Processing, Addison – Wesley, 1998.

Regressive Fuzzy Inference Models with Clustering Identification: Application to the ESTSP´08 Competition.

Federico Montesino Pouzols[1] and Angel Barriga[2] *

1- CSIC, Scientific Research Council - Microelectronics Institute of Seville
Avda. Reina Mercedes s/n. Edif. CICA. E-41012 Seville - Spain

2- University of Seville - Department of Electronics and Electromagnetism
Computer Engineering School, Avda. Reina Mercedes s/n. E-41012 Seville - Spain

Abstract. In the context of a previously proposed methodology framework for time series prediction, we use a clustering technique in order to identify fuzzy inference systems for the regressive modeling of time series. We propose a modified version of the method for the identification of fuzzy rules based on the subtractive clustering method proposed by Chiu. The proper number of rules is derived from an a priori nonparametric residual variance estimate that is also used for an initial input selection stage. In addition, systems are optimized through the Levenberg-Marquardt second order method. The proposed method is applied to the three datasets of the ESTSP´08 competition and the results are discussed.

1 Introduction

In the context of a previously proposed methodology framework for time series prediction [1], we propose a time series prediction method intended to be fast and robust against noise and perturbations. In this paper, we develop a method for regressive time series prediction by means of fuzzy inference systems. We will call fuzzy autoregressors those autoregressors implemented as fuzzy inference systems. This is not to be confused with what is usually called fuzzy regression in the literature [2]. The method proposed here is intended to apply to crisp time series.

In practice, one finds two problems when building a fuzzy inference model for a time series: choosing the inputs and identifying the structure of the system. Here, the first problem is addressed by means of an a priori feature selection scheme based on nonparametric residual variance estimation [3]. The second problem is adressed by an identification method based on clustering techniques [4].

We introduce the use of clustering methods within the aforementioned methodology and propose an scheme for identifying fuzzy inference systems by means of the subtractive clustering method. This new approach to identification within the methodology has been found to attain an accuracy similar to that of the grid partition based method analyzed in [1].

*This work has been supported in part by project TEC2005-04359/MIC from the Spanish Ministry of Education and Science as well as project TIC2006-635 and grants IAC07-I-0205:33080 and IAC08-II-3347:56263 from the Andalusian regional Government.

The proposed method, implemented in the Xfuzzy environment [5, 6], is applied to the three datasets of the ESTSP´08 time series competition. To this end, we extend the original methodology framework to multivariate time series prediction. A simple downsampling is applied in order to speed up the prediction of one of the datasets.

The next section describes nonparametric residual variance estimation. In section 3, we describe the method used for fuzzy systems identification based on cluster estimation. In section 4, an automatic time series prediction method is proposed. This method is applied to the three datasets of the ESTSP´08 competition in section 5.

2 Nonparametric Residual Variance Estimation: Delta Test

Nonparametric residual variance estimation (or nonparametric noise estimation, NNE) is a well-known technique in statistics and machine learning, finding many applications in nonlinear modeling [3]. Delta Test (DT) is a NNE method for estimating the variance of the noise, or the lowest mean square error (MSE) that can be achieved by a model without overfitting the training set [3]. Given N multiple input-single output pairs, $(\bar{x}_i, y_i) \in R^M \times R$, the theory behind the DT method considers that the mapping between $\bar{x}_i$ and y_i is given by the following expression:

$$y_i = f(\bar{x}_i) + r_i,$$

where f is an unknown perfect fitting model and r_i is the noise. DT is based on hypothesis coming from the continuity of the regression function. When two inputs x and x' are close, the continuity of the regression function implies that outputs $f(x)$ and $f(x')$ will be close enough. When this implication does not hold, it is due to the influence of the noise.

Let us denote the first nearest neighbor of the point $\bar{x}_i$ in the set $\{\bar{x}_1, \ldots, \bar{x}_N\}$ by $\bar{x}_{NN}$. Then the DT, δ, is defined as follows:

$$\delta = \frac{1}{2N} \sum_{i=1}^{N} \left| y_{NN(i)} - y_i \right|^2,$$

where $y_{NN(i)}$ is the output corresponding to $\bar{x}_{NN(i)}$. For a proof of convergence, refer to [7].DT has been shown to be a robust method for estimating the lowest possible mean squared error (MSE) of a nonlinear model without overfitting. DT is useful for evaluating nonlinear correlations between random variables, namely, input and output pairs. This method will be used for a priori input selection.

3 Identification of Fuzzy Systems via Subtractive Clustering

The results of fixed clustering algorithms heavily depend on its initialization stage. To overcome this, the subtractive clustering method [8] was proposed as a modification to the mountain method for approximate estimation of cluster centers by Yager and Filev [9], where the the notion of potential of a cluster was introduced.

The method requires the initialization of two parameters: neighborhood radius and maximum number of clusters to be identified. The algorithm for cluster identification as implemented in version 3.2 of the Xfuzzy environment [5, 6] comprises the following steps:

1. Data in the input pattern, considered as a set of input vectors $\bar{x}_i$ in an $M+1$ dimensional space with $i = 1 \ldots N$, are normalized so that every scalar value falls within the $[0, 1]$ range.

2. An initial potential, P_i, is assigned to each input vector:

$$P_i = \sum_{j=1}^{N} e^{-\alpha \|\bar{x}_i - \bar{x}_j\|^2},$$

   where $\alpha = \frac{4}{r_a^2}$, with r_a being the neighborhood radius, a parameter that specifies the radius beyond which points have a negligible influence on each datum. The euclidean norm is used.

3. The input vector with the highest potential, $\bar{v}_s$ with potential P_s, is selected as the center of a cluster.

4. $\bar{v}_s$ is removed from the input pattern set, and the potencial for the remaining vectors is recomputed according to the following expression:

$$P_{i|new} = P_{i|old} - P_s \cdot e^{-\beta \|\bar{x}_i - \bar{v}_s\|^2},$$

   where $\beta = \frac{4}{r_b^2}$ and r_b is a positive constant with value $1.25 \cdot r_a$. Thus, an amount of potential as a function of its distance from the identified cluster is subtracted from each input vector. This way, those patterns near the identified cluster center will have a lower chance of being selected as a new cluster center in next iterations.

5. Stop conditions are checked. The first condition to be considered is:

$$P_{i|new} < \varepsilon \cdot P_s$$

   Thus, ε is a parameter key to the performance of the incremental clustering algorithm. The higher ε is, the less clusters will be identified. A common accepted value for ε is 0.15.The second stop condition is an a priori specified maximum number of clusters. If no stop condition holds, a new iteration is started from step 3.

Once the clustering algorithm has finished and Q clusters have been identified, a fuzzy inference system is generated with Q rules. These clusters are considered as prototypes or models of the whole input pattern sequence. First, values are denormalized. For simplicity, let us consider a multiple scalar input, single scalar output where the input patterns to the clustering algorithm consist

of M inputs and one output. Let us call c_i the denormalized values corresponding to each $\bar{v}_s$ identified cluster center. A rule is generated for each identified cluster. If a cluster has the following general form:

$$c_j : (a_1^j, \ldots, a_{M+1}^j), \text{ with } j = 1, \ldots, Q,$$

where a_{M+1}^j corresponds to the output (y) of a fuzzy inference model whereas the $a_1^j, \ldots, a_M^j$ correspond to the inputs $(x_1, \ldots, x_M)$ to the fuzzy model. For each cluster, the matching rule is generated with the following form:

$$\text{IF } x_1 \text{ is } A_1^j \text{ AND } x_2 \text{ is } A_2^j \text{ AND } \ldots \text{ AND } x_M = A_M^j \text{ THEN } y \text{ is } A_{M+1}^j,$$

where A_i^j are linguistic terms. Thus, Q different membership functions are generated for each input and output variable. Gaussian input membership functions are defined centered in the corresponding components of the cluster centers, i.e., the a_i^j values are the centers of the A_i^j linguistic terms. The width is set as a constant value based on the neighborhood radius. Output membership functions are defined as singleton functions centered in the corresponding component of the cluster centers (a_{M+1}^j) as well.

4 Method for Time Series Prediction with Fuzzy Inference Systems

Consider a discrete time series as a vector, $\bar{y} = y_1, y_2, \ldots, y_{t-1}, y_t$ that represents an ordered set of values, where t is the number of values in the series. The problem of predicting one future value, y_{t+1}, using an autoregressive model (autoregressor) with no exogenous inputs can be stated as follows[1]:

$$\hat{y}_{t+1} = f_r(y_t, y_{t-1}, \ldots, y_{t-M+1}),$$

where $\hat{y}_{t+1}$ is the prediction of model f_r and M is the number of inputs to the regressor.

Predicting the first unknown value requires building a model, f_r, that maps regressor inputs (known values) into regressor outputs (predictions). When a prediction horizon higher than 1 is considered, the unknown values can be predicted following two main strategies: recursive and direct prediction.

With the recursive strategy, the same model is applied iteratively, using predictions as known data to predict the next unknown values. It is the most simple and intuitive strategy and does not require any additional modeling after an autoregressor for 1 step ahead prediction is built. However, recursive prediction suffers from accumulation of errors.

Direct prediction requires that the process of building an autoregressor be applied for each unknown future value. Thus, for a maximum prediction horizon H, H direct models are built, one for each prediction horizon h:

$$\hat{y}_{t+h} = f_h(y_t, y_{t-1}, \ldots, y_{t-M+1}), \text{ with } 1 \leq h \leq H$$

[1]For simplicity, exogenous inputs are not considered here. The generalization to models with exogenous inputs is straightforward and will be addressed in section 5

In this paper, we follow the direct prediction strategy. In order to build each autoregressor, a fuzzy inference system is defined as a mapping between a vector of crisp inputs, and a crisp output. We use the following operators: minimum t-norm for conjunction and implication (or, more precisely, joint constraint) operators, and fuzzy mean as defuzzification method. Thus, the Mamdani inference model is followed. In this particular case a fuzzy autoregressor with M inputs for prediction horizon h is formulated as:

$$\mathcal{F}_h(\bar{y}) = \frac{\sum_{l=1}^{Q_h} \min\left(\mu_{R_l^h}, \min_{1\leq v\leq M} \mu_{L_l^{i,h}}(y_v)\right)}{\sum_{l=1}^{Q_h} \min_{1\leq v\leq M} \mu_{L_l^{i,h}}(y_v)},$$

where Q_h is the number of rules (identified clusters) in the rulebase for horizon h. $\mu_{L_l^{i,h}}$ are gaussian membership functions for the input linguistic labels and $\mu_{R_l^h}$ are singleton membership functions, both derived from clusters as specified in section 3.

The problem of building a regressor can be precisely stated as that of defining a proper number and configuration of membership functions and building a fuzzy rulebase from a data set of t sample data from a time series such that the fuzzy systems $\mathcal{F}_h(\bar{y})$ closely predict the $h-$th next values of the time series. The error metric to be minimized is the mean squared error (MSE). We propose a method in which a fuzzy inference system is defined for each prediction horizon throughout the stages shown in figure 1. These stages are detailed in the following subsections.

4.1 Variable Selection

As first step in the methodology, DT estimates are employed so as to perform an a priori selection of the optimal subset of inputs from the initial set of M inputs, given a maximum regressor size M. Variable selection requires a selection criterion. We use the result of the DT applied to a particular variable subset selection as as a measure of the goodness of the selection. The input selection that minimizes the DT estimate is chosen for the next stages. This way, the set of selected variables is the one that represents the mapping between inputs and outputs in the most deterministic manner.

In addition, a selection procedure is required. For medium regressor sizes (up to around 10-20) , an exhaustive evaluation of DT for all the possible selections (a total of $2^M - 1$) is feasible. We will call this procedure *exhaustive DT search*. Its main advantage is that the optimal selection is found. For higher regressor sizes, forward-backward search of selections (FBS) [10] can be used. Although this procedure does not guarantee optimality, it provides a convenient balance between performance and computational requirements.

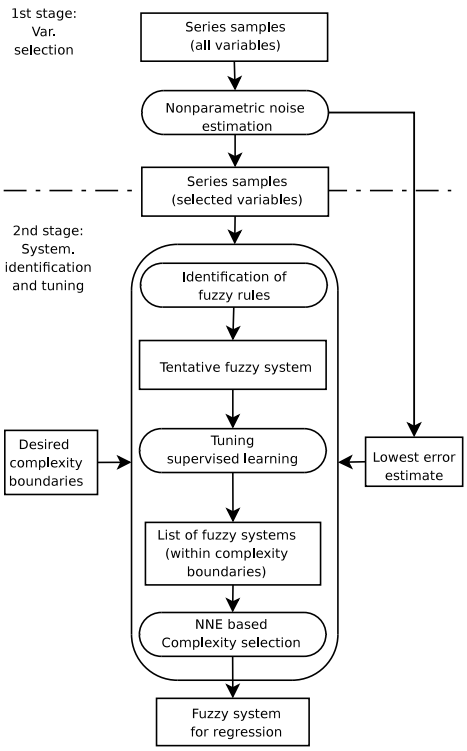


Fig. 1: Methodology Framework for Time Series Prediction.

4.2 System Identification and Optimization

This stage comprises three substages that are performed iteratively and in a coordinated manner. The process is driven by the third substage until a system that satisfies a condition on the training error derived from the DT estimate is built.

Substage 2.1: System identification. In this substage, the structure of the inference system (linguistic labels and rule base) is defined. The off-line clustering algorithm described in section 3 is used. In general, for fuzzy systems identification, one or more parameters are required that specify the potential complexity of the inference system. Thus, the desired boundaries of complexity for the systems being built are additional inputs to the process.

Besides other differences, the original method proposed by Chiu for the identification of fuzzy rules based on subtractive clustering [8] requires to set the maximum number of clusters in advance. Here, we overcome this limitation by using the DT estimates as an estimation of the optimal training error of the system. Since the training error is non-monotonic with the number of clusters, an exploration has to be performed. Systems are explored in an increasing order of complexity, from the lowest possible number of clusters up to a maximum specified as complexity boundary. Thus, the inital second stop condition of the

algorithm is replaced by a check of the training error against the DT estimate.

This iterative identification process for increasing number of clusters stops when a system is built such that the training error is lower than the DT estimate. The selection is made in the third stage by comparing the error after the next (optimization) stage.

Substage 2.2: System optimization. We consider an additional optimization step in the methodology as a substage separated from the identification substage. The Levenberg-Marquardt second order optimization method [11], driven by the normalized MSE (NMSE) is used. All the parameters of the membership functions of every input and output are adjusted using the algorithm implementation in the Xfuzzy development environment [12], i.e., self-tuning inference systems are defined.

Substage 2.3: Complexity selection. As last step, the complexity of the fuzzy autoregressors (measured as the number of clusters or rules) is selected depending on the DT estimate. The first (simplest) system that falls within the error range defined by the DT-NNE is selected.

5 Application to the ESTSP´08 Competition Datasets

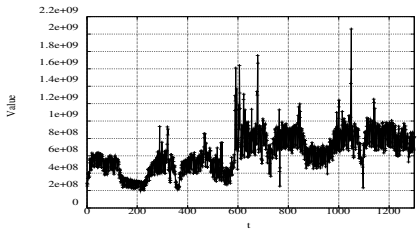
The three datasets of the ESTSP´08 time series competition are analyzed in the next sections. The datasets of the competition are used as training sets for building predictive models without applying any preprocessing technique. Considering the universal approximation capability of the models used and our previous experience with the prediction methodology, no preprocessing is required to capture nonlinear dynamics in general. For dataset 2, the methodology described above is applied. For dataset 1, we extend the methodology in order to cope with multivariate series. For dataset 3, we use a simple downsampling technique in order to provide a fast approach to predicting large and noisy time series.

5.1 Dataset 2

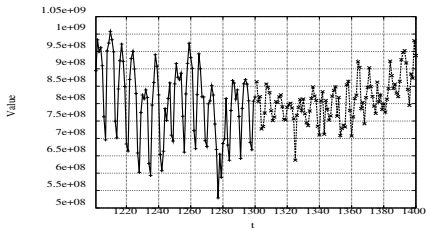
Dataset 2 is a univariate time series consisting of 1300 samples. The next 100 values are to be predicted. Figure 2 shows the known values together with the next 100 values as predicted through the method presented above.

5.2 Dataset 3

Dataset 3 is a univariate time series consiting of 31614 samples. The next 200 values are to be predicted. The series exhibits a high noise level or amount of highly unpredictable events. As the competition encourages the submission of fast methods, and under the assumption that predicting these events is not a goal in the application field of dataset 2, we simplify the problem by using a basic linear temporal aggregation scheme. It is well known that increasing the level of aggregation increases the signal to noise ratio. We note though that other more advanced preprocessing techniques, such as wavelet multiresolution analysis, could provide better results.



(a) Known values



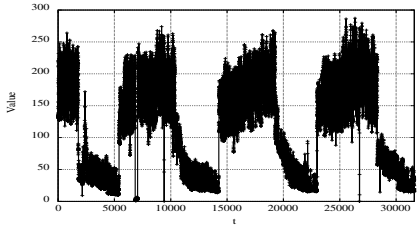
(b) Last 100 known values followed by 100 predicted values (dashed line)

Fig. 2: Dataset 2: Known 1300 samples and next 100 predicted samples.

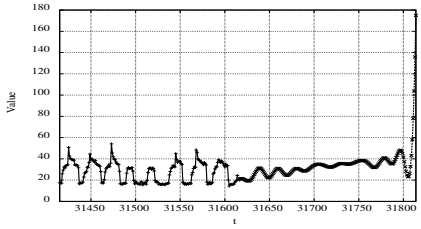
Let y_t be the time series for dataset 2, we will analyze y_t^{10}, the 10th order non-overlapping temporal aggregation of y_t performed by computing average values for each term. In general, an aggregate time series is defined as

$$y_t^{agg(A)} = \sum_{j=0}^{A} w_j y_{t-j}$$

This is a linear combination of current and past values of y_t. For dataset 3, we set $A = 9$ and $w_j = 1/10$. This way, we will apply the proposed prediction method on the dataset 3 series at a higher timescale. For the competition, the original task of predicting the next 200 values of y_t is replaced by the prediction of the next 20 values of $y_t^{agg(10)}$. Then, the next 200 values of the original time series are derived by a simple cubir spline interpolation technique. Figure 3 shows the known values of the time series together with the next 200 predicted values.



(a) Known values



(b) Last 200 known values followed by 200 predicted values (dashed line, $\times$)

Fig. 3: Dataset 3: Known 31614 samples and next 200 predicted samples.

5.3 Dataset 1

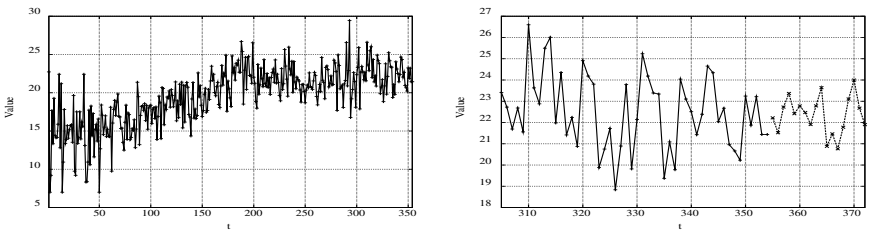
Dataset 1 is a multivariate time series consisting of 3 variables. Let us call these variables y^1, y^2 and y^3 for the first, second and third columns of the dataset file,

respectively. The next 18 values of y^3 are to be predicted.

We outline an extension of the proposed method for regression with exogenous inputs. By using a linear projection technique, we define a straightforward extension that also allows for considering a higher number of past known values as inputs to the regressors.

The projection to latent structures/partial least squares (PLS) technique [13] is used in order to generate a linear projection of an initial input-output regression matrix derived from dataset 1 into a lower dimensional set that is used for input selection and model building. An NxM matrix of linearly projected input variables is generated using PLS on the initial regression matrix, where N is the number of original samples of the time series and M is the desired dimension of the projected input space.

The set of three variable-specific regressor sizes, M^1, M^2 and M^3, was explored within reasonable limits considering that the dataset does not have a high number of samples. The sizes that minimize the DT estimate after projection of the initial set are chosen. M was set to 10 and the following values were chosen: $M^1 = 0, M^2 = 20$ and $M^3 = 30$. Figure 4 shows the known values of the time series for y_3 together with the predicted next 18 values.



(a) Known values

(b) Last 50 known values followed by 18 predicted values (dashed line)

Fig. 4: Dataset 1: Known 354 samples of the third variable and next 18 predicted samples.

5.4 Discussion

As a major advantage over usual prediction techniques, the proposed method can be used in order to understand the dynamics underlying these datasets with a simplicity-accuracy tradeoff under the direct control of the user of a time series prediction tool. Each rule can be interpreted as a linguistic map between regions of interest of the input and output spaces.

The joint use of a robust technique for NNE and input selection as well as the optimization of models through supervised learning allows for the development of accurate models while keeping the number of clusters low. Rules are ordered by significance, from the main underlying dynamics to the minor details. For the competition, no regressor required more than 16 rules, with an average number of rules below 10. As an example, for the 20 models built for dataset 3, between 5 and 7 inputs are selected out of 10 for the 18 models built, with 5.9 variables

selected on average. Between 2 and 26 rules are identified for each model, with 5.8 rules on average. In practice, according to our experience, systems built with the proposed method have a very low number of rules while attaining a high accuracy for a number of time series benchmarks [1].

6 Conclusion

We have proposed an automatic method for long-term time series prediction by means of fuzzy inference systems. Regressive inference systems are identified by a variant of the subtractive clustering algorithm that uses nonparametric residual variance estimates together with the Levenberg-Marquardt second order optimization algorithm in order to set the proper number and configuration of clusters and rules. The method is fast and provides linguistically interpretable models with an adjustable simplicity-accuracy tradeoff. Projection and downsampling techniques have been applied in combination with the proposed method to the ESTSP´08 competition datasets.

References

[1] Federico Montesino Pouzols, Amaury Lendasse, and Angel Barriga. Fuzzy Inference Based Autoregressors for Time Series Prediction Using Nonparametric Residual Variance Estimation. In *17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'08), IEEE World Congress on Computational Intelligence*, Hong Kong, China, June 2008.

[2] Yun-Hsi O. Chang and Bilal M. Ayyub. Fuzzy regression methods - a comparative assessment. *Fuzzy Sets and Systems*, 119(2):187–203, April 2001.

[3] Antonia J. Jones. New Tools in Non-linear Modelling and Prediction. *Computational Management Science*, pages 109–149, September 2004.

[4] Stephen L. Chiu. A Cluster Estimation Method with Extension to Fuzzy Model Identification. In *IEEE Conference on Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence.*, volume 2, pages 1240–1245, Orlando, FL, 1994.

[5] Xfuzzy: Fuzzy Logic Design Tools. Available from https://forja.rediris.es/projects/xfuzzy, April 2008.

[6] Federico Montesino Pouzols, Amaury Lendasse, and Angel Barriga. xftsp: a Tool for Time Series Prediction by Means of Fuzzy Inference Systems. In *4th IEEE International Conference on Intelligent Systems (IS´08)*, Varna, Bulgaria, September 2008.

[7] Elia Liitiäinen, Amaury Lendasse, and Francesco Corona. Non-parametric Residual Variance Estimation in Supervised Learning. In *IWANN 2007, International Work-Conference on Artificial Neural Networks*, pages 63–71, San Sebastián, Spain, June 2007.

[8] Stephen L. Chiu. Fuzzy Model Identification Based on Cluster Estimation. *Journal of Intelligent & Fuzzy Systems*, 2(3):267–278, September 1994.

[9] R. R. Yager and D. P. Filev. Approximate clustering via the mountain method. *IEEE Transactions on Systems, Man and Cybernetics*, 24(8):1279–1284, August 1994.

[10] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse. Methodology for Long-Term Prediction of Time Series. *Neurocomputing*, 70(16-18):2861–2869, October 2007.

[11] Roberto Battiti. First and Second Order Methods for Learning: Between Steepest Descent and Newton's Method. *Neural Computation*, 4(2):141–166, March 1992.

[12] Francisco José Moreno-Velo, Iluminada Baturone, Angel Barriga, and Santiago Sánchez-Solano. Automatic Tuning of Complex Fuzzy Systems with Xfuzzy. *Fuzzy Sets and Systems*, 158(18):2026–2038, September 2007.

[13] P. Geladi and B. R. Kowalski. Partial least-squares regression: A tutorial. *Analytica Chimica Acta*, 185:1–17, 1986.

Multiple Local ARX Modeling for System Identification Using the Self-Organizing Map

Luís Gustavo M. Souza[1] and Guilherme A. Barreto[2] *

1,2- Federal University of Ceará
Department Teleinformatics Enginnering
Av. Mister Hull, S/N - Campus of Pici, Center of Technology
CP 6005, CEP 60455-760, Fortaleza, CE, Brazil

Abstract. In this paper we build global NARX (Nonlinear Auto-Regressive with eXogenous variables) models from multiple local linear ARX models whose state spaces have been partitioned through Kohonen's Self-Organizing Map. The studied models are evaluated in the task of identifying the inverse dynamics of a flexible robotic arm. Simulation results demonstrate that SOM-based multiple local ARX models perform better than a single ARX model and an MLP-based global NARX models.

1 Problem Formulation

Several complex dynamical systems which can be described by the NARX model:

$$y(t) = f[y(t-1), \ldots, y(t-n_y); u(t), u(t-1), \ldots, u(t-n_u+1)], \qquad (1)$$

where n_y and n_u are the (memory) orders of the dynamical model. In words, Eq. (1) states that the system output y at time t depends, on the past n_y output values and on the past n_u values of the input u. In many situations, it is also desirable to approximate the inverse mapping of a nonlinear plant, given by

$$u(t) = f^{-1}[y(t-1), \ldots, y(t-n_y); u(t-1), \ldots, u(t-n_u)]. \qquad (2)$$

In system identification, the goal is to obtain estimates of $f(\cdot)$ and/or $f^{-1}(\cdot)$ from available input-output time series data $\{u(t), y(t)\}_{t=1}^{M}$.

SOM-based local dynamic modeling and control approaches have been successfully applied to complex system identification and control tasks [1, 2, 3, 4, 5], but these contributions still remain widely unknown by the Machine Learning and Statistics communities. In this paper, we compare the performances of system identification techniques which rely on the self-organizing map (SOM) [6] for local function approximation. To the best of our knowledge, such an evaluation has not been reported elsewhere. In this sense, this is one of the main contributions of this paper.

For the SOM and other unsupervised networks to be able to learn dynamical mappings, they must have some type of *short-term memory* (STM) mechanism. That is, the SOM should be capable of temporarily storing past information about the system input and output vectors. There are several STM models, such

*The authors thank FUNCAP (grant #1469/07) and CAPES/PRODOC for the financial support.

as delay lines, leaky integrators, reaction-diffusion mechanisms and feedback loops [7, 8], which can be incorporated into the SOM to allow it to approximate a dynamical mapping $f(\cdot)$ or its inverse $f^{-1}(\cdot)$. In order to draw a parallel with standard system identification approaches, we limit ourselves to describe the VQTAM approach in terms of time delays as STM mechanisms.

The remainder of the paper is organized as follows. In Section 2, SOM architecture and its learning process are described. In Section 3, two SOM-based local ARX models are introduced. Simulations and performance are presented in Section 4. The paper is concluded in Section 5.

2 The Self-Organizing Map

The SOM is composed of two fully connected layers: an input layer and a competitive layer. The input layer simply receives the incoming input vector and forwards it to the competitive layer through weight vectors. The goal of SOM is to represent the input data distribution by the distribution of the weight vectors. Competitive learning drives the winning weight vector to become more similar to the input data. Throughout this paper, we represent the weight vector between input layer and neuron i as

$$\mathbf{w}_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,j}, \ldots, w_{i,p})^T, \tag{3}$$

where $w_{i,j} \in \mathbb{R}$ denotes the weight connecting node j in the input layer with neuron i, and p, is the dimension of the input vector. In what follows, a brief description of the original SOM algorithm is given.

Firstly we use Euclidean distance metric to find the current winning neuron, $i^*(t)$, as given by the following expression:

$$i^*(t) = \arg\min_{\forall i \in \mathcal{A}} \|\mathbf{x}(t) - \mathbf{w}_i(t)\| \tag{4}$$

where $\mathbf{x}(t) \in \mathbb{R}^p$ denotes the current input vector, $\mathbf{w}_i(t) \in \mathbb{R}^p$ is the weight vector of neuron i, and t denotes the iterations of the algorithm. Secondly, it is necessary to adjust the weight vectors of the winning neuron and of those neurons belonging to its neighborhood:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)] \tag{5}$$

where $0 < \alpha(t) < 1$ is the learning rate and $h(i^*, i; t)$ is a gaussian weighting function that limits the neighborhood of the winning neuron:

$$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)}\right) \tag{6}$$

where $\mathbf{r}_i(t)$ and $\mathbf{r}_{i^*}(t)$, are respectively, the positions of neurons i and i^* in a predefined output array where the neurons are arranged in the nodes, and $\sigma(t) > 0$ defines the radius of the neighborhood function at time t.

The variables $\alpha(t)$ and $\sigma(t)$ should both decay with time to guarantee convergence of the weight vectors to stable steady states. In this paper, we adopt an exponential decay for both, given by:

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_T}{\alpha_0}\right)^{(t/T)} \quad \text{and} \quad \sigma(t) = \sigma_0 \left(\frac{\sigma_T}{\sigma_0}\right)^{(t/T)} \tag{7}$$

where α_0 (σ_0) and α_T (σ_T) are the initial and final values of $\alpha(t)$ ($\sigma(t)$), respectively. The operations defined by Eqs. (4) and (7) are repeated until a steady state of global ordering of the weight vectors has been achieved. In this case, we say that the map has converged.

The resulting map also preserves the topology of the input samples in the sense that adjacent input patterns are mapped into adjacent neurons on the map. Due to this topology-preserving property, the SOM is able to cluster input information and spatial relationships of the data on the map. This clustering ability of the SOM has shown to be quite useful for the identification of nonlinear dynamical systems [9]. However, the number of neurons required by the SOM to provide a good approximation of a given input-output mapping is very high, specially when compared to the MLP and RBF neural networks. To alleviate this limitation of the plain SOM algorithm to some extent, we introduce two SOM-based multiple local ARX models.

3 Multiple Local ARX Models Based on the SOM

In this section, we describe two approaches to the system identification problem that use the SOM as a building block. The basic idea behind both is the partitioning of the input space into non-overlapping regions, called Voronoi cells, whose centroids correspond to the weight vectors of the SOM. Then an interpolating hyperplane is associated with each Voronoi cell or to a small subset of them, in order to estimate the output.

3.1 Local Linear Mapping

The first architecture to be described is called *Local Linear Mapping* (LLM) [10]. The basic idea of the LLM is to associate each neuron in the SOM with a conventional FIR/LMS linear filter. The SOM array is used to quantize the input space in a reduced number of prototype vectors (and hence, Voronoi cells), while the filter associated with the winning neuron provides a local linear estimator of the output of the mapping being approximated.

Thus, for the inverse modeling task of interest, each input vector $\mathbf{x}(t) \in \mathbb{R}^{p+q}$ is defined as

$$\mathbf{x}(t) = [u(t-1), \ldots, u(t-q); y(t-1), \ldots, y(t-p)]^T. \tag{8}$$

Clustering (or vector quantization) of the input space $\mathcal{X}$ is performed by the LLM as in the usual SOM algorithm, with each neuron i owning a prototype vector $\mathbf{w}_i$, $i = 1, \ldots, N$.

Additionally, there is a coefficient vector $\mathbf{a}_i \in \mathbb{R}^{p+q}$ associated to each weight vector $\mathbf{w}_i$, which plays the role of the coefficients of an (linear) ARX model:

$$\mathbf{a}_i(t) = [b_{i,1}(t), \ldots, b_{i,q}(t), a_{i,1}(t), \ldots, a_{i,p}(t)]^T. \tag{9}$$

The output value is provided by one of the local ARX model as follows

$$\begin{aligned} \hat{u}(t) &= \sum_{k=1}^{q} b_{i^*,k}(t)u(t-k) + \sum_{l=1}^{p} a_{i^*,l}(t)y(t-l) \\ &= \mathbf{a}_{i^*}^T(t)\mathbf{x}(t), \end{aligned} \tag{10}$$

where $\mathbf{a}_{i^*}(t)$ is the coefficient vector associated with the winning neuron $i^*(t)$. From Eq. (10), one can easily note that the coefficient vector $a_{i^*}(t)$ is used to build a local linear approximation of the output of the desired nonlinear mapping.

Since the adjustable parameters of the LLM algorithm are the set of prototype vectors $\mathbf{w}_i(t)$ and their associated coefficient vectors $\mathbf{a}_i(t), i = 1, 2, \ldots, p+q$, we need two learning rules. The rule for updating the prototype vectors $\mathbf{w}_i$ follows exactly the one given in Eq. (5). The learning rule of the coefficient vectors $\mathbf{a}_i(t)$ is an extension of the normalized LMS algorithm, that also takes into account the influence of the neighborhood function $h(i^*, i; t)$:

$$\mathbf{a}_i(t+1) = \mathbf{a}_i(t) + \alpha' h(i^*, i; t)\Delta\mathbf{a}_i(t), \tag{11}$$

where $0 < \alpha' \ll 1$ denotes the learning rate of the coefficient vector, and $\Delta\mathbf{a}_i(t)$ is the error correction rule of Widrow-Hoff, given by

$$\Delta\mathbf{a}_i(t) = \left[u(t) - \mathbf{a}_i^T(t)\mathbf{x}(t)\right] \frac{\mathbf{x}(t)}{\|\mathbf{x}(t)\|^2}, \tag{12}$$

where $u(t)$ is the desired output of the inverse mapping being approximated.

3.2 Prototype-Based Local Least-Squares Model

The algorithm to be described in this section, called K-winners SOM (KSOM), was originally applied to nonstationary time series prediction [5]. In this paper we aim to evaluate this architecture in the context of nonlinear system identification. For training purposes, the KSOM algorithm depends on the VQTAM (*Vector-Quantized Temporal Associative Memory*) model [9], which is a simple extension of the SOM algorithm that simultaneously performs vector quantization on the input and output spaces of a given nonlinear mapping.

In the VQTAM model, the input vector at time step t, $\mathbf{x}(t)$, is composed of two parts. The first part, denoted $\mathbf{x}^{in}(t) \in \mathbb{R}^{p+q}$, carries data about the input of the dynamic mapping to be learned. The second part, denoted $x^{out}(t) \in \mathbb{R}$, contains data concerning the desired output of this mapping. The weight vector of neuron i, $\mathbf{w}_i(t)$, has its dimension increased accordingly. These changes are formulated as follows:

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{x}^{in}(t) \\ x^{out}(t) \end{pmatrix} \text{ and } \mathbf{w}_i(t) = \begin{pmatrix} \mathbf{w}_i^{in}(t) \\ w_i^{out}(t) \end{pmatrix}, \tag{13}$$

where $\mathbf{w}_i^{in}(t) \in \mathbb{R}^{p+q}$ and $w_i^{out}(t) \in \mathbb{R}$ are, respectively, the portions of the weight (prototype) vector which store information about the inputs and the outputs of the desired mapping. Depending on the variables chosen to build the vector $\mathbf{x}^{in}(t)$ and scalar $x^{out}(t)$ one can use the SOM algorithm to learn the forward or the inverse mapping of a given plant (system). For instance, if the interest is in inverse identification, then we define

$$\mathbf{x}^{in}(t) = [u(t-1), \ldots, u(t-q); y(t-1), \ldots, y(t-p)]^T \quad \text{and} \quad x^{out}(t) = u(t). \tag{14}$$

The winning neuron at time step t is determined based only on $\mathbf{x}^{in}(t)$, i.e.

$$i^*(t) = \arg \min_{\forall i \in \mathcal{A}} \{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}. \tag{15}$$

For updating the weights, however, both $\mathbf{x}^{in}(t)$ and $x^{out}(t)$ are used:

$$\Delta \mathbf{w}_i^{in}(t) = \alpha(t) h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)] \tag{16}$$

$$\Delta w_i^{out}(t) = \alpha(t) h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)] \tag{17}$$

where $0 < \alpha(t) < 1$ is the learning rate, and $h(i^*, i; t)$ is a time-varying Gaussian neighborhood function defined as in Eq. (6).

The learning rule in Eq. (16) performs topology-preserving vector quantization on the input space, while the rule in Eq. (17) acts similarly on the output space of the mapping being learned. As the training proceeds, the SOM learns to associate the input prototype vectors $\mathbf{w}_i^{in}$ with the corresponding output prototype vectors $\mathbf{w}_i^{out}$. The SOM-based associative memory implemented by the VQTAM can then be used for function approximation purposes.

Since the VQTAM is essentially a vector quantization algorithm, it requires too many neurons to provide small prediction errors when approximating continuous mappings. This limitation can be somewhat alleviated through the use of interpolation methods specially designed for the SOM architecture, such as geometric interpolation [11] and topological interpolation [12]. Another possibility is to devise a local linear interpolation strategy over the neighborhood of the winning neuron. For example, after training the VQTAM model, the coefficient vector $\mathbf{a}(t)$ of a local ARX model for estimating the mapping output is computed for each time step t by the standard least-squares estimation (LSE) technique, using the weight vectors of the K ($K \gg 1$) neurons closest to the current input vector, instead of using the original data vectors.

Let the set of K winning weight vectors at time t to be denoted by $\{\mathbf{w}_{i_1^*}, \mathbf{w}_{i_2^*}, \ldots, \mathbf{w}_{i_K^*}\}$. Recall that due to the VQTAM training style, each weight vector $\mathbf{w}_i(t)$ has a portion associated with $\mathbf{x}^{in}(t)$ and other associated with $x^{out}(t)$. So, the KSOM uses the corresponding K pairs of prototype vectors $\{\mathbf{w}_{i_k^*}^{in}(t), w_{i_k^*}^{out}(t)\}_{k=1}^K$, with the aim of building a local linear function at time t:

$$w_{i_k^*}^{out} = \mathbf{a}^T(t) \mathbf{w}_{i_k^*}^{in}(t), \qquad k = 1, \ldots, K \tag{18}$$

where $\mathbf{a}(t) = [b_1(t), \ldots, b_q(t), a_1(t), \ldots, a_p(t)]^T$ is a time-varying coefficient vector. Equation (18) can be written in a matrix form as

$$\mathbf{w}^{out}(t) = \mathbf{R}(t)\mathbf{a}(t), \tag{19}$$

where the output vector $\mathbf{w}^{out}$ and the regression matrix $\mathbf{R}$ at time t are defined as follows

$$\mathbf{w}^{out}(t) = [w^{out}_{i^*_1,1}(t) \;\; w^{out}_{i^*_2,1}(t) \;\; \cdots \;\; w^{out}_{i^*_K,1}(t)]^T \tag{20}$$

and

$$\mathbf{R}(t) = \begin{pmatrix} w^{in}_{i^*_1,1}(t) & w^{in}_{i^*_1,2}(t) & \cdots & w^{in}_{i^*_1,p+q}(t) \\ w^{in}_{i^*_2,1}(t) & w^{in}_{i^*_2,2}(t) & \cdots & w^{in}_{i^*_2,p+q}(t) \\ \vdots & \vdots & \vdots & \vdots \\ w^{in}_{i^*_K,1}(t) & w^{in}_{i^*_K,2}(t) & \cdots & w^{in}_{i^*_K,p+q}(t) \end{pmatrix}. \tag{21}$$

The coefficient vector $\mathbf{a}(t)$ is then computed by the following Tikhonov-regularized pseudoinverse (minimum norm) procedure

$$\mathbf{a}(t) = \left(\mathbf{R}^T(t)\mathbf{R}(t) + \lambda\mathbf{I}\right)^{-1}\mathbf{R}^T(t)\mathbf{w}^{out}(t), \tag{22}$$

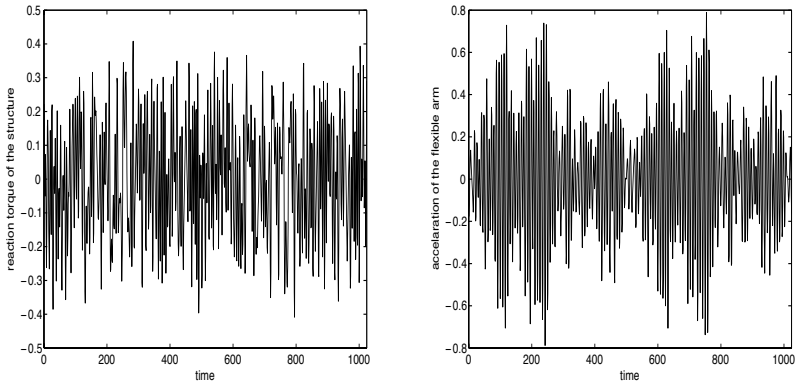
where $\mathbf{I}$ is a identity matrix of order K and $\lambda > 0$ (e.g. $\lambda = 0.001$) is a small regularization constant. Once $\mathbf{a}(t)$ is estimated, we can locally approximate the output of the nonlinear mapping by the output of the following ARX model:

$$\hat{u}(t) = \sum_{k=1}^{q} b_k(t)u(t-k) + \sum_{l=1}^{p} a_l(t)y(t-l) = \mathbf{a}^T(t)\mathbf{x}^{in}(t)$$

The KSOM can be considered a local (linear) ARX model due to the use of a subset of K weight vectors chosen from the whole set of N weight vectors. This is one of the differences between KSOM and the LLM approaches. While the former uses $K \ll N$ prototype vectors to build the local linear model, the latter uses a single prototype. Another difference is that the LLM approach uses a LMS-like learning rule to update the coefficient vector of the winning neuron. Once training is completed all coefficient vectors $\mathbf{a}_i$, $i = 1, \ldots, N$, are frozen for posterior use. The KSOM, instead, uses a LSE-like procedure to find the coefficient vector $\mathbf{a}(t)$ each time an input vector is presented, so that a single linear mapping is built at each time step.

Cho *et al.* [3] proposed a neural architecture that is equivalent to the KSOM in the sense that the coefficient vector $\mathbf{a}(t)$ is computed from K prototype vectors of a trained SOM using the LSE technique. However, the required prototype vectors are not selected as the K nearest prototypes to the current input vector, but rather automatically selected as the winning prototype at time t and its $K-1$ topological neighbors. If topological defects are present, as usually occurs for multidimensional data, the KSOM provides more accurate results.

Chen and Xi [13] also proposed a local linear regression model whose coefficient vectors are computed using the prototypes of a competitive learning network through the recursive least-squares algorithm. However, the competitive network used by Chen and Xi does not have the topology-preserving properties of the SOM algorithm, which has shown to be important for system identification purposes [9].



(a) (b)

Fig. 1: Measured values of reaction torque of the structure (a) and acceleration of the flexible arm (b).

4 Computer Simulations and Discussion

The proposed SOM-based multiple local ARX models are evaluated in the identification of the inverse dynamics of a flexible robot arm. The arm is installed on an electric motor. We want to model the input-output mapping from the measured reaction torque of the structure on the ground to the acceleration of the flexible arm[1]. Figure 1 shows the measured values of the reaction torque of the structure (input time series, $\{u(t)\}$) and the acceleration of the flexible arm (output time series, $\{y(t)\}$).

For the sake of completeness, the LLM- and KSOM-based local ARX models are compared with an one-hidden-layer MLP trained by the standard backpropagation algorithm (MLP-1h), another one-hidden-layer MLP trained by the Levenberg-Marquardt (MLP-LM) algorithm and, finally, a two-hidden-layer MLP (MLP-2h) trained by the standard backpropagation algorithm. All these global NARX models are also compared with the linear *Auto-Regressive with eXogenous variables* (ARX) model, trained on-line through the plain LMS algorithm.

For all MLP-based global NARX models, the activation function of the hidden neurons is the hyperbolic tangent function, while the output neuron uses a linear one. After some experimentation, the best configuration of the MLP-1h and MLP-LM models have 30 hidden neurons. For the MLP-2h, the number of

[1]These data were obtained in the framework of the Belgian Programme on Interuniversity Attraction Poles (IUAP-nr.50) initiated by the Belgian State.

Table 1: Performances of the global and local models for the robotic arm data.

Neural Models	**NMSE**			
	mean	*min*	*max*	*variance*
KSOM	0.0064	0.0045	0.0117	1.83×10^{-6}
KSOM-PL	0.0187	0.0082	0.0657	8.47×10^{-5}
MLP-LM	0.1488	0.0657	0.4936	0.0107
MLP-1h	0.1622	0.1549	0.1699	1.03×10^{-5}
VQTAM-T	0.1669	0.1199	0.2263	6.14×10^{-4}
LLM	0.3176	0.2685	0.3558	2.23×10^{-4}
ARX	0.3848	0.3848	0.3848	0.0445
VQTAM-G	0.4968	0.3700	0.6458	0.0024
MLP-2h	0.6963	0.5978	1.5310	0.0368

neurons in second hidden layer is heuristically set to half the number of neurons in the first hidden layer. The learning rate for the MLPs was set to 0.1.

During the prediction phase, the neural models should compute the estimation error (residuals) $e(t) = u(t) - \hat{u}(t)$, where $u(t)$ is desired output and $\hat{u}(t)$ is the estimate provided by each neural model. The performances of all models are assessed through the *normalized mean squared error* (NMSE):

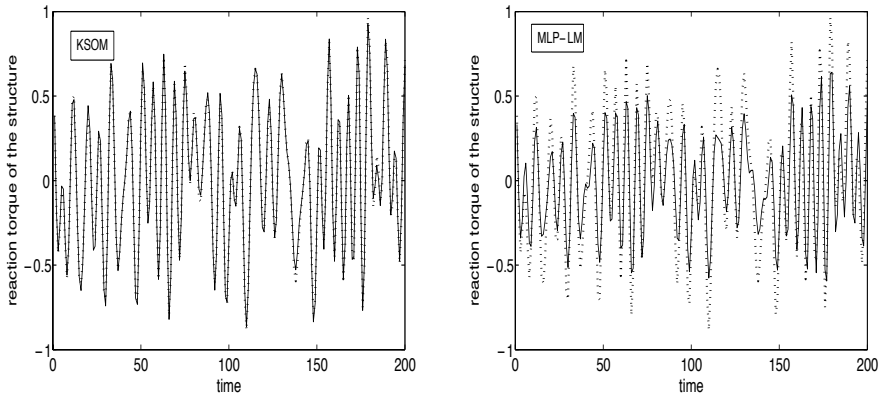
$$NMSE = \frac{\sum_{t=1}^{M} e^2(t)}{M \cdot \hat{\sigma}_u^2} = \frac{\sum_{t=1}^{M} (u(t) - \hat{u}(t))^2}{M \cdot \hat{\sigma}_u^2} \tag{23}$$

where $\hat{\sigma}_u^2$ is the variance of the original time series $\{u(t)\}_{t=1}^{M}$ and M is the length of the sequence of residuals.

The models are trained using the first 820 samples of the input-output signal sequences (approximately, 80% of the total) and tested with the remaining 204 samples. The input and output memory orders are set to $p = 4$ and $q = 5$, respectively. The obtained results are shown in Table 1, where are displayed the mean, minimum, maximum and variance of the NMSE values, measured along the 100 training/testing runs. The weights of the neural models were randomly initialized at each run. In this table, the models are again sorted in increasing order of the mean NMSE values.

The number of neurons for all SOM-based local ARX models is set to 30. For the KSOM-based local NARX model, K is equal to 25. For each SOM-based model, the initial and final learning rates are set to $\alpha_0 = 0.5$ and $\alpha_T = 0.01$. The initial and final values of radius of the neighborhood function are $\sigma_0 = N/2$ and $\sigma_T = 0.001$, where N, the number of neurons in the SOM, is set to 30. The learning rate α' is set to 0.1.

For the sake of curiosity, the VQTAM with topological (VQTAM-T) and geometric (VQTAM-G) interpolations have been tested with the hope of improving the approximation accuracy of the plain VQTAM model. The KSOM-based local ARX model was also implemented using the recently proposed *Parameter-less* SOM (PLSOM) architecture [14], which requires no annealing of the learning



(a) (b)

Fig. 2: Typical estimated sequences of reaction torque of the structure provided by the KSOM and MLP-LM models. Dashed lines denote actual sample values, while the solid line indicates the estimated sequence.

rate and neighborhood width parameters. The fundamental difference between the PLSOM and the SOM is that while the SOM depends on the learning rate and neighborhood size to decrease over time, e.g., as a function of the number of iterations of the learning algorithm, the PLSOM calculates these values based on the local quadratic fitting error of the map to the input space.

The performance of KSOM-based local ARX model on this real-world application is by far the best one, even better than the MLP-based global NARX models. A better performance of the KSOM-based model in comparison to the LLM-based model is also verified. This can be partly explained by the fact that the parameters of the KSOM-based local model are estimated in a batch mode from the K closest prototypes, while the parameters of the LLM-based model are estimated in an online mode.

The LLM-based local ARX model performed only better than the ARX, VQTAM-G and MLP-2h models. The performances of these three models were very poor. The performance of the VQTAM-T is statistically equivalent to that of the MLP-1h model. Among the MLP-based models, the use of second-order learning algorithm was crucial to the good performance of the MLP-LM model.

Finally, Figure 2 shows typical sequences of estimated values of the reaction torque of the structure provided by the best local and global NARX models. Figure 2a shows the sequence generated by the KSOM-based model, while Figure 2b shows the sequence estimated by the MLP-LM model.

5 Conclusion

We have attempted to tackle the problem of nonlinear system identification using the local linear modeling methodology. For that purpose we presented two multiple local ARX models based on Kohonen's self-organizing map and evaluated them in the identification of the inverse dynamics of one real-world data set, a robot arm. The first local ARX model builds a fixed number of local ARX models, one for each Voronoi region associated with the prototype vectors of the SOM. The second one builds only a single local ARX model using the prototypes vectors closest to the current input vector. It has been shown for the robot arm data set the KSOM-based local ARX model presented the best performance among all models.

References

[1] J. Cho, J. Principe, D. Erdogmus, and M. Motter. Quasi-sliding mode control strategy based on multiple linear models. *Neurocomputing*, 70(4-6):962–974, 2007.

[2] I. Díaz-Blanco, A. A. Cuadrado-Vega, A. B. Diez-González, J. J. Fuertes-Martínez, M. Domínguez-González, and P. Reguera-Acevedo. Visualization of dynamics using local dynamic modelling with self-organizing maps. *Lecture Notes on Computer Science*, 4668:609–617, 2007.

[3] J. Cho, J. Principe, D. Erdogmus, and M. Motter. Modeling and inverse controller design for an unmanned aerial vehicle based on the self-organizing map. *IEEE Transactions on Neural Networks*, 17(2):445–460, 2006.

[4] J. Lan, J. Cho, D. Erdogmus, J. C. Principe, M. A. Motter, and J. Xu. Local linear PID controllers for nonlinear control. *International Journal of Control and Intelligent Systems*, 33(1):26–35, 2005.

[5] G.A. Barreto, J.C.M. Mota, L.G.M. Souza, and R.A. Frota. Nonstationary time series prediction using local models based on competitive neural networks. *Lecture Notes in Computer Science*, 3029:1146–1155, 2004.

[6] T. K. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, Heidelberg, 2nd extended edition, 1997.

[7] J. C. Principe, N. R. Euliano, and S. Garani. Principles and networks for self-organization in space-time. *Neural Networks*, 15(8–9):1069–1083, 2002.

[8] G. A. Barreto and A. F. R. Araújo. Time in self-organizing maps: an overview of models. *International Journal of Computer Research*, 10(2):139–179, 2001.

[9] G. A. Barreto and A. F. R. Araújo. Identification and control of dynamical systems using the self-organizing map. *IEEE Transactions on Neural Networks*, 15(5):1244–1259, 2004.

[10] J. Walter, H. Ritter, and K. Schulten. Non-linear prediction with self-organizing map. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'90)*, volume 1, pages 587–592, 1990.

[11] J. Göppert and W. Rosenstiel. Topology preserving interpolation in selforganizing maps. In *Proceedings of the NeuroNIMES'93*, pages 425–434, 1993.

[12] J. Göppert and W. Rosenstiel. Topological interpolation in som by affine transformations. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN'95)*, pages 15–20, 1995.

[13] J.-Q. Chen and Y.-G. Xi. Nonlinear system modeling by competitive learning and adaptive fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics-Part C*, 28(2):231–238, 1998.

[14] E. Berglund and J. Sitte. The parameterless self-organizing map algorithm. *IEEE Transactions on Neural Networks*, 17(2):305–316, 2006.

Time series opportunities in the petroleum industry

Roar Nybø *

SINTEF Petroleum Research
Thormøhlensgate 55, 5008 Bergen - Norway

Abstract. Soft computing techniques have gained greater interest and acceptance in the oil industry in recent years. Some, who advocate the education of more interdisciplinary petroleum engineers, even list soft computing as one of the core competencies for such engineers. This paper will give a brief introduction to the challenges and opportunities for applied time series prediction in the oil industry and recent trends in research, with a focus on fault prediction.

1 Introduction

The petroleum industry, while traditionally conservative, has a surprisingly long history of testing and deploying artificial intelligence (AI) or soft computing systems. Early examples include expert systems like "Prospector" from the late 70's for evaluating mineral deposits and "Dipmeter advisor" from the 80's [1], which dealt with inferring 3D geological structures from measurements taken along the borehole. The early 90's saw the commercial launch of "ODDA", an expert system advisor for directional drilling developed by Total and Norsk Hydro [2] and the "Analysis While Drilling" package developed by Total and Nordic Offshore Systems [3].

When an oil well is drilled, equipment failure or a misjudgement of downhole conditions may delay the operation by days or weeks. One need only consider the cost of renting a drilling rig, now exceeding half a million dollars per day, to see that the cost of faults may easily enter the million dollar range. These high stakes increase the risk or perceived risk of trying out unproven technology, partly explaining the conservative attitude [4]. On the other hand, the drilling contractor would get an immediate return on their investments in fault prediction software even if it delivered only a small increase in the ability to predict and avoid faults. Thus ideally, a fault prediction system could be developed incrementally and still be useful and justify industry support in its early stages.

This paper seeks to give an overview of recent developments in the petroleum industry, its use of time series prediction methods as well as the characteristics of its time series, research challenges, open problems and possible development.

* This work is funded in part by the "Center for Integrated Operations in the Petroleum Industry" (http://www.ntnu.no/iocenter)

2 Integrated operations

Currently AI or "soft computing" methods are finding increased acceptance as one of the tools for deploying "Integrated Operations" (IO). Also known variously as e-operations and digital oil fields, the term loosely encompass a move for cutting costs and increasing oil recovery using new computer technology. Some broad themes can be outlined. One is how the oil industry is importing ideas from the process industries, such as a tighter integration between the oil companies and their suppliers when it comes to logistics and project management, as well as analyzing and optimizing offshore oil platform performance on the same terms as for a factory.

Another eye-catching feature of IO is the use of extensive video-conferencing between on- and offshore facilities and 3D visualization of the oil field and ongoing well drilling [5]. This has the aim of integrating different disciplines into planning and real-time operations. It also advances the industry's goal of keeping more of their personnel in onshore offices, being available for consultation with several platforms.

Of most interest may be the increase in real-time data that the oil industry has seen in recent years. This is mainly due to new downhole measurement equipment and an increase in bandwidth between this equipment and the offshore rig [6, 7] as well as the rig and land based facilities. Much of the ongoing research in IO seeks to take advantage of this torrent of data. Efforts include real-time production optimization [8] detailed monitoring of fluid flow [9] and adjusting the path of a well during drilling, based on real-time downhole surveys of the rock formation. While such real-time measurements have been available for years, their bandwidth was previously limited to around 20 bits/sec [10]. Challenging optimization problems also abound in the area of time series data analysis, such as predicting the interactions between a large number of wells in order to optimize their total production.

All this has created a need for a stronger ICT-literacy in the oil industry, where people such as Prof. Ershaghi at the Center for Interactive Smart Oilfield Technologies [†] at U. of Southern California are among the ones arguing for a revision of the petroleum engineering education, with data mining and soft computing as two of the core competencies.

3 Properties of oil industry time series

Time series in the oil industry are of course generated from a multitude of different processes, but a short overview may still give a feel of how it differs from the textbook examples of time series. Asking an industry professional about the series most prominent feature, the answer is likely to be "*noise*". Grave inaccuracies in the measurements contribute substantially, but "noise" may also be aspects of the system not covered by our models. For instance, the drillstring (Figure 1), as any rotating equipment, may fall prone to vibrations and wobbling. This may affect not just measurements of the drillstring's torque and weight, but also fluid flow and pressure [11]. The drillstring, several kilometres long, may in turn have had its movement affected by the type and amount of gravel in the well.

[†] http://cisoft.usc.edu/

This messy and very much "real world" interconnectedness of different processes has long been acknowledged as a challenge for traditional models [3]. However, it also lets a feature such as wobbling make its fingerprint on many variables. It is enticing that this correlation may let a multivariate analysis extract early warning signs from what is generally regarded as noise.

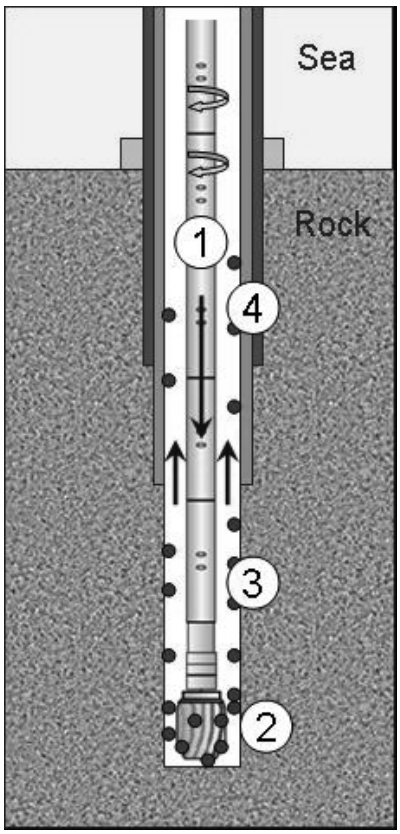


Figure 1: Simplified schematics of oil well drilling. A rotating pipe (1) extends from the rig to the bottom of the well, where it drives a drillbit. (2) At the same time fluid is being pumped down the pipe. This returns to the rig along the outside of the pipe, carrying the crushed rock (3) along with it. As drilling progresses, the wall of the well is periodically fitted with a protective casing (4).

3.1 **Pre-processing and problem definition**

For the purpose of downhole monitoring, our task can often be framed as that of an inverse problem: Given our measurements, reconstruct the downhole conditions that

caused them. Measurements of rock formation properties are coarse and real-time measurements along the well are sparse with current technology, frequently making the inverse problem an ill-posed one [12].

Fault detection and prevention may also be framed as a time-series prediction problem: Given the time-series up to now, predict if a fault is likely to occur. The horizon of such a task will be problem-specific. While the first signs of gas having entered a well become visible only minutes before the operator must respond, bad hole-cleaning is a situation that may deteriorate gradually over several hours.

Current alarm systems tend to employ simple pattern classification such as threshold values and trend detection, with more sophisticated systems focusing on recognizing the safe events that cause false alarms [13]. In the case of drilling, false alarms are today a major complaint among the users [14]. Attempts at pattern recognition by supervised learning may learn to foresee these common events, but the most severe events are rare in comparison. With few examples, a straight-forward approach taking into account all system parameters and using a large sliding window is then bound to experience the "curse of dimensionality" [15].

3.2 **The Hierarchy**

To get a grip on the data and underlying processes, one approach is a hierarchical decomposition. In [16] Saputelli et.al introduced the "Field Operations Hierarchy" in Figure 2 as a convenient structuring for the problem of optimizing the production of oil and gas.

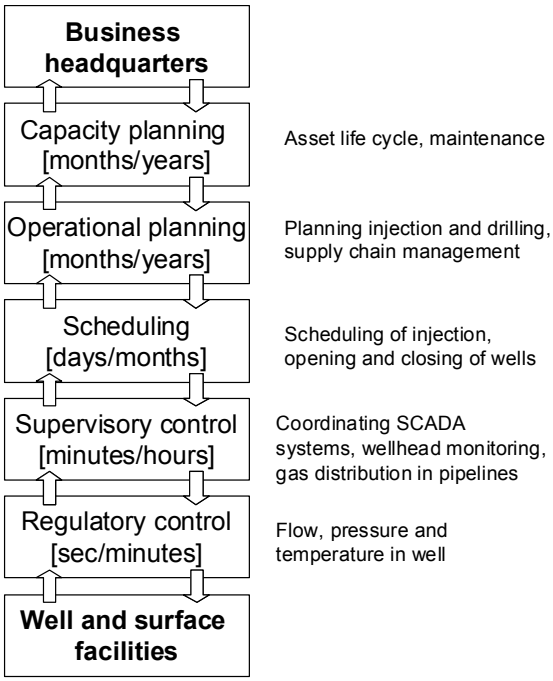


Figure 2: The Field Operations Hierarchy according to Saputelli

The structure will be familiar from other industries. In this figure, information travel upwards and orders are sent downwards. Both scales of time and space increase for higher levels. These levels are a result not just of management structure but of the time-scales of the physical processes involved. For instance, a flow measurement has a time-scale of seconds and may relate to a branch of a single well. The measurement is relayed to the scheduling level which may plan for days ahead taking the gradual wear of equipment into account. Operational planning in turn must plan for the even slower depletion of the whole oil-field.

Orders are subsequently relayed downward e.g. for the closing of valves in the well. This forms a closed loop of supervisory control, where time series fault detection and prediction as well as predictive control becomes important.
Such a hierarchy draws on theory from supervisory control theory, where such nested loops may also be associated with the supervisor's learning process [17, 18].

3.3 **The characteristics of different levels**

In addition to being a layout for optimization problems, the hierarchy is a useful roadmap for time series prediction. It appears that the demands placed on a time series prediction system depends very much on where in the hierarchy it is implemented. One may for instance notice that the information relayed becomes increasingly symbolic and aggregated as one move upward in the hierarchy. From numerical values that are interpreted higher up as states of the equipment and status reports, on to "net present value" at headquarters. It is telling that we find a symbolically based method like Case Based Reasoning analyzing job reports in the day to month range [19, 20], while typical applications of more numerical methods like neural networks focus on the lower levels [21-23]. In the lower levels it also usually demanded that we restrict ourselves to algorithms that work in real-time systems.

An exception to the symbolic trend is the task of simulating oil and gas reservoirs. This deals with large scales of time and space but mainly numerical data. Prediction of the movement of gas, oil and water in the rock is a computing-intensive problem, made harder by sparse measurements.

Soft computing on time series is here found in two niches. The first is as an aid in history-matching of the model. With many free parameters and much time spent on each run, it is tempting to use soft computing methods to optimize the parameter search. Efforts include evolutionary algorithms [24] and ensemble Kalman filters [25]. This also allows us to use deterministic models while moving towards a probabilistic assessment of subsurface conditions. This probabilistic viewpoint is another trend in the petroleum industry made possible by increased computing power.

The second application sees the time-consuming simulator replaced by a surrogate model, such as a neural network. Trained on input and output from a traditional model, the neural network gives quicker predictions, allowing us to e.g. try out a larger number of different well placements, or explore more of the parameter space. This approach is sometimes referred to as "neuro-simulation" in the literature [26].

Moving down to real-time measurements, a typical issue here is the non-stationarity. Time series from drilling record a system with frequent exogenous

inputs, as the drilling operator frequently intervenes to change rates of flow, pipe rotation or type of fluid used. A drilling operation is composed of several different tasks and a parameter value that indicates imminent danger in one situation may be in the normal range in another. The classification of "drilling modes" would therefore feature prominently as a pre-processing step on the way to more sophisticated fault predictions.

The drilling mode classification is also becoming an increasingly pressing issue for symbolic analysis at the higher levels. Much of the system knowledge gathered by methods such as CBR derives from human-made logs and reports of operations. But if such systems are to offer analysis and advice in real-time, they would need real-time reports. A drilling mode classification could correspond to such reports, which shows how applications of hybrid systems may arise naturally in the field operations hierarchy.

Recent efforts at automated classification include a rule-based system by Thonhauser et.al. for the automatic generation of drilling reports [27, 28], but the problem of a reliable *real-time* classification is still an unsolved problem.

4 Combined approaches

The hierarchical approach gives us some leads on overcoming the curse of dimensionality, but not all methods rely on this. For instance, in [29] Lorentzen et.al study an optimization problem where they make a leap directly from choke control to net present value. A common factor in their approach and the previously discussed soft computing methods in reservoir simulation is the combination of soft computing with physical models. Advanced simulators exist for all levels from reservoirs to well drilling [5] and is in a sense an encoding of our knowledge of the system.

It is recognized in system identification and grey-box modelling [30] that "fictious data" is a convenient way to encode expert knowledge, which the simulators readily provide. It is the author's opinion that a combined hard and soft computing approach would be viable not only for the aforementioned optimization problems, but also for fault prediction in time series. However, as mentioned, the physical models do not necessarily reproduce fault signatures; properties of the noise or some complex effects may lead to false alarms.

An approach taken by e.g. Forssell and Lindskog in [31] is to run the best available model alongside measurements and train the AI on their difference or the unexplained "residual". That is, to predict:

$$T_{residual} = T - T_{model\ prediction}$$

We may then re-order the equation to yield an improved prediction:

$$T_{combined\ prediction} = T_{Prediction\ of\ residual} + T_{Model\ prediction} \approx T$$

This improved prediction may in turn be used to remove false alarms or increase the sensitivity of established fault detection methods, as implemented by this author in [32]. However, this approach tends to assume that the task of predicting $T_{residual}$ is a simpler or lower-dimensional task than the prediction of T. While often true, counterexamples show that this is not true in general. Other possibilities for injecting

prior knowledge from simulations exist, but the author is not aware of well-established methods for the general case.

5 Conclusions

Petroleum exploration and production is an industry that provides researchers with multivariate time-series with challenging "real-world" properties. The time series call for different prediction tasks which seem suited to wildly different schools of prediction systems, while at the same time hinting at a need for a "deeper", perhaps hybrid, system architecture.

Regarding applied research and commercial applications of time series prediction, we find that management now has an open mind towards new methods, under the umbrella of Integrated Operations. However, applications such as real-time fault detection will find that there is a low tolerance of false alarms while time series prediction as part of e.g. production optimization, would have to compete against successful traditional methods. To find acceptance in the industry, and more importantly, to be useful, it is the authors' opinion that time series prediction results must be in a form that can be combined with those from existing physical models. This approach has the potential of yielding better accuracy, stability and generalisation capability than each method alone. It would also be in the spirit of Integrated Operations for us to integrate the experience inherent in time series with the knowledge inherent in physical models.

References

[1] J. Liebowitz, *The handbook of applied expert systems*, Boca Raton, FL: CRC Press, 1998.

[2] M. H. Amara, and B. Martin, "SPE 20419: The Offshore Directional Drilling Advisor: An Expert System for Directional Drilling Optimization," in SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, 1990.

[3] N. Hytten, L. Havrevold, and P. Parigot, "SPE 23052: Getting More Out of Drilling Data by Analysis-While-Drilling," in Offshore Europe, 3-6 September 1991, Aberdeen, United Kingdom, 1991.

[4] D. Govia, and K. Carpenter, "SPE 112237-MS: Valuation Models for Intelligent Strategies," in Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 2008.

[5] R. Rommetveit, K. S. Bjørkevoll, S. I. Ødegård *et al.*, "Automatic Real-Time Drilling Supervision, Simulation, 3D Visualization and Diagnosis on Ekofisk," in 2008 IADC/SPE Drilling Conference, Orlando, Florida, 2008.

[6] Nygaard, Jahangir, Gravem *et al.*, "SPE 112742-MS: A Step Change in Total System Approach Through Wired Drillpipe Technology," in IADC/SPE Drilling Conference, Orlando, Florida, 2008.

[7] T. S. Olberg, H. Laastad, B. Lesso *et al.*, "SPE 112702-MS: The Utilization of the Massive Amount of Real-Time Data Acquired in Wired Drillpipe Operations," in IADC/SPE Drilling Conference, Orlando, Florida, USA, 2008.

[8] A. Shere, Y. Roberts, and S. Bakkevig, "SPE 112130-MS: Online Production Optimisation on Ekofisk," in Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 2008.

[9] X. Wang, J. Lee, B. Thigpen *et al.*, “SPE 111790-MS: Modeling Flow Profile Using Distributed Temperature Sensor (DTS) System,” in Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 2008.

[10] M. Hernandez, D. MacNeill, M. Reeves *et al.*, “SPE 113157-MS: High-Speed Wired Drillstring Telemetry Network Delivers Increased Safety, Efficiency, Reliability and Productivity to the Drilling Industry,” in SPE Indian Oil and Gas Technical Conference and Exhibition, Mumbai, India, 2008.

[11] R. Ahmed, and S. Miska, “SPE 112604-MS: Experimental Study and Modeling of Yield Power-Law Fluid Flow in Annuli with Drillpipe Rotation,” in IADC/SPE Drilling Conference, Orlando, Florida, 2008.

[12] M. Khasanov, R. Khabibullin, and V. Krasnov, “SPE 88557-MS: Interactive Visualization of Uncertainty in Well Test Interpretation,” in SPE Asia Pacific Oil and Gas Conference and Exhibition, Perth, Australia, 2004.

[13] D. Hargreaves, S. Jardine, and B. Jeffryes, “SPE 71369 - Early Kick Detection for Deepwater Drilling: New Probabilistic Methods Applied in the Field,” in SPE Annual Technical Conference and Exhibition, New Orleans, 2001.

[14] P. S. A. Norway, *Human factors i bore og brønnoperasjoner - Borernes arbeidssituasjon*, 2007.

[15] M. Verleysen, “Learning high-dimensional data,” *Limitations and Future Trends in Neural Computation, IOS Press*, pp. 141-162, 2003.

[16] L. Saputelli, H. Malki, J. Canelon *et al.*, “SPE 77703-MS, A Critical Overview of Artificial Neural Network Applications in the Context of Continuous Oil Field Optimization,” in SPE Annual Technical Conference and Exhibition, San Antonio, Texas, 2002.

[17] T. B. Sheridan, "Supervisory Control," *Handbook of Human Factors*, G. Salvendy, ed., pp. 1243-1268, 1987.

[18] J. Rasmussen, "Information Processing and Human-Machine Interaction, An Approach to Cognitive Engineering," 1986.

[19] P. Skalle, and A. Aamodt, "Knowledge-Based Decision Support in Oil Well Drilling," *Intelligent Information Processing II*, pp. 443-455, 2005.

[20] Popa, Popa, Malamma *et al.*, “SPE 114229-MS: Case-Based Reasoning Approach for Well Failure Diagnostics and Planning,” in SPE Western Regional and Pacific Section AAPG Joint Meeting, Bakersfield, California, 2008.

[21] R. K. Fruhwirth, G. Thonhauser, and W. Mathis, "SPE 103217, Hybrid simulation using Neural Networks to Predict Drilling Hydraulics in Real Time," 2006.

[22] M. et.al, *Method and apparatus for prediction control in drilling dynamics using neural networks*, USA US 6,732,052 B2,to Baker Hughes Incorporated, 2004.

[23] Dashevskiy, Dubinsky, and Macpherson, “56442-MS: Application of Neural Networks for Predictive Control in Drilling Dynamics,” in SPE Annual Technical Conference and Exhibition, Houston, Texas, 1999.

[24] Selberg, Ludvigsen, Diab *et al.*, “SPE 102349-MS: New Era of History Matching and Probabilistic Forecasting--A Case Study,” in SPE Annual Technical Conference and Exhibition, San Antonio, Texas, 2006.

[25] Lorentzen, Nævdal, Vallès *et al.*, “SPE 96375-MS: Analysis of the Ensemble Kalman Filter for Estimation of Permeability and Porosity in Reservoir Models,” in SPE Annual Technical Conference and Exhibition, Dallas, Texas, 2005.

[26] Hari, Ertekin, and Grader, “SPE 39962-MS: Methods of Neuro-Simulation for Field Development,” in SPE Rocky Mountain Regional/Low-Permeability Reservoirs Symposium, Denver, Colorado, 1998.

[27] G. Thonhauser, “Using Real-Time Data for Automated Drilling Performance Analysis,” *Oil Gas European Magazine*, no. 4/2004, pp. 170-173, 2004.

[28] G. Thonhauser, and W. Mathis, "SPE 103211, Automated Reporting Using Rig Sensor Data Enables Superior Drilling Project Management," 2006.

[29] R. J. Lorentzen, A. M. Berg, G. Nævdal *et al.*, “SPE 99690-MS: A New Approach for Dynamic Optimization of Waterflooding Problems,” in Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 2006.

[30] M. Kárný, A. Halousková, and P. Nedoma, “Recursive approximation by ARX model: a tool for grey box modelling,” *International journal of adaptive control and signal processing,* vol. 9, no. 6, pp. 525-546, 1995.

[31] U. Forssell, and P. Lindskog, “Combining semi-physical and neural network modeling: an example of its usefulness,” in 11th IFAC Symposium on System Identification (SYSID'97), 1997.

[32] R. Nybø, K. S. Bjørkevoll, and R. Rommetveit, “SPE 112212-MS, Spotting a False Alarm—Integrating Experience and Real-Time Analysis With Artificial Intelligence,” in Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 2008.

Homicide Flash-up Prediction Algorithm Studying

D.V Serebryakov[1], I.V. Kuznetsov[2] *

1- Keldysh Institute for Applied Mathematics RAS – 3d Dept
125047 Moscow, Miusskaya pl., 4 – Russia
2- International Institute of Earthquake Prediction Theory and Mathematical Geophysics RAS – Dept of Nonlinear dynamics
117997 Moscow, Profsoyuznaya str, 84/32 – Russia

Abstract. This report represents a homicide flash-up prediction algorithm especially detailed from general criminality prediction method based on universal behavior rules of complex nonlinear hierarchical systems. We study algorithm sensitivity to algorithm parameters and data variation. A created automotive prediction computer-based program allows us to obtain a huge number of the algorithm's realizations. We show that there are groups of data where small variations of algorithm parameters lead to small variation of prediction results. Moreover, we show that developed algorithm represents a similar result over times series for two different towns.

1 Introduction

Developed prediction algorithm [1] is based on supposition about hierarchicality of crime regime, i.e. existence of heaviness levels of accident, crime or group of crimes. In this work we consider three heaviness groups where the third group, the group of the most heaviness crimes, includes only homicides. Thus our algorithm was specialized here for homicide flash-ups prediction. We compare results of the algorithm realization for its different parameters such as averaging and accumulation windows and, that is very important, we compare algorithm realization for two different time series representing two towns, Tambov and Yaroslavl.

1.1 Object-to-predict

Object or *object-to-predict* is a time moment where a flash-up of the serious crimes arises plus a special condition. We believe it is advisable to determine a flash-up as an overshoot of present crimes' number comparatively to mean number of crimes for a previous time interval. We say that there is *a flash-up* at the time moment i, if satisfied

$$R_i \equiv N_i - n_i \geq \sigma,$$
$$n_i = \frac{1}{w_\sigma} \sum_{j=1}^{w_\sigma} N_{i-j} \ ,$$

* This work was supported by Russian Fund for Basic Research (project № 07-01-00618).

where σ is intercepting threshold, N_i is a number of homicides at the moment i, w_σ is averaging windows, R_i's is residue series [1-3].

1.2 Alarm

We say that there is *a predictor signal* at the time moment i, if satisfied

$$r_i \equiv b_i - k_i \geq \beta,$$
$$k_i = \frac{1}{w_\beta} \sum_{j=1}^{w_\beta} b_{i-j} ,$$

where β is intercepting threshold, $b_i = (V_1(i) - V_3(i)) / 2$, where V_1 and V_3 are sums of events for the 1st and 3d groups of heaviness correspondently, for instance $V_3(i) = \lg (N_i + N_{i-1} + \ldots + N_{i-(w-1)})$, where w is an accumulation window, w_β is averaging window, r_i's is residue series.

If a predictor signal presents at time moment i, then one *declares an alarm* for following d serial moments $i+1, i+2, \ldots, i+d$, i.e. one should wait for an object appearance over these moments named as *alarm interval* or *alarm*. If there is other predictor signal at the moment j during the alarm interval, then the alarm is prolonged for the next d moments $j+1, j+2, \ldots, j+d$. If there is an object s during the alarm, then the final alarm moment is s after which this alarm is cancelled. If an object s and a predictor signal i present at the same moment or $0 \leq i - s \leq 2$, then an alarm is not declared: two serial moments followed an object are considered as relaxation period when system behavior is special and isn't applicable for the prediction [1-3].

We say that *an alarm is successful* if the alarm consists an object-to-predict, *an alarm is false* (*false alarm*) if there is no an object over the alarm interval.

We say that *the object is predicted* if it is in an alarm, an object-to-predict is *fail to predict* (*a fail-to-predict object*) if it is not covered any alarm interval.

Fig. 1 represents a result of algorithm realization in graphic.

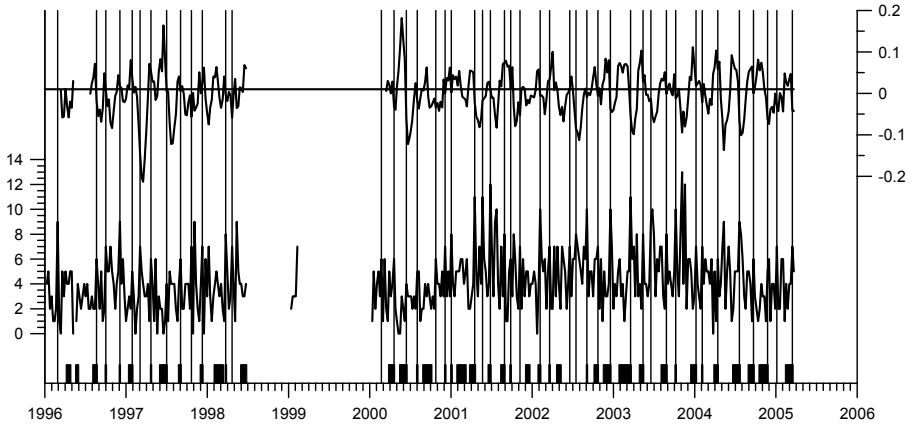


Figure 1. A graphical result of one of algorithm realizations. Lower graph – homicides (left vertical axis); upper graph – residue series r_i (right vertical axis), intercepting threshold β (horizontal line); vertical lines – objects; rectangles – alarms.

It is obvious if we change any parameter of the algorithm we obtain another result of prediction. Investigation of some relations between algorithm parameters and its prediction results is a goal of this work.

2 Data

In this work we use weekly crime data from towns Tambov and Yaroslavl, administrative centers of Tambovskaya and Yaroslavskaya oblast (regions) in Russia. Monitoring time for Tambov is period from 10.01.1996 till 23.03.2005 without period from 01.07.1998 till 05.01.2000, for Yaroslavl – period from 9.02.1993 till 19.06.2001, we defined a beginning of weeks as Tuesday. The monitoring time periods are 405 and 436 weeks, i.e. there are 405 and 436 cases or observations in initial time series for Tambov and Yaroslavl correspondently.

3 Numerical experiments

Here we vary for every town windows w_β and w, σ and β, when $w_\sigma = 5$. We consider values of w_β and w are equaled to 5, 7, 10, 17 and 26 weeks, values of σ and β are located in intervals $[1; \sigma_{max}]$ with step $\delta_\sigma = 0{,}1$ and $[0{,}001; \beta_{max}]$ with step $\delta_\beta = 0{,}001$ correspondently. For every algorithm realization we save the following information:

1) N (number of objects),
2) N^- (number of fail-to-predict objects),
3) T_a (duration of alarms),
4) W (number of alarms),
5) W^- (number of false alarms)

Quality of prediction is estimated by set of quantities $\eta = N^- / N$, $\tau = T_a / T$, $\varphi = \eta + \tau$ [1-3], $\chi = W^- / W$ and $\psi = \chi + \varphi$ which are calculated for every algorithm realization. It is clear to understand sense of η and τ on instance of two extreme points. As $\eta = 1$, $\tau = 0$ we have one extreme case when all objects are fail-to-predict as time alarm equals zero. As $\eta = 0$, $\tau = 1$ we predict all objects when time alarm is all monitoring period.

An integral quality of prediction might be evaluated by quantity $\varphi = \eta + \tau$. Condition $\varphi < 1$ corresponds to non-trivial prediction. The φ is smaller the prediction is better. But also it is necessary to take into account values of η and τ themselves. Non-trivial prediction is considered as acceptable if $\eta < 1/2$ and $\tau < 1/2$ and successful if $\eta < 1/3$ and $\tau < 1/3$. As we see from Table 1 the maximum of minimal for η equals 0,333 and for τ – 0,030.

town	w_β	w	η	τ	χ	φ	ψ
Tambov	5	5	,000	,014	,000	,322	,529
	5	7	,000	,013	,000	,169	,518
	5	10	,000	,023	,000	,302	,583
	5	17	,000	,022	,000	,243	,614
	5	26	,000	,013	,000	,235	,664
	7	5	,000	,013	,000	,419	,518

Tambov	7	7	,000	,016	,000	,155	,544
	7	10	,000	,024	,000	,270	,565
	7	17	,000	,022	,000	,198	,621
	7	26	,250	,014	,000	,443	,693
	10	5	,000	,023	,000	,552	,583
	10	7	,000	,024	,000	,520	,565
	10	10	,000	,010	,000	,273	,595
	10	17	,000	,014	,000	,184	,666
	10	26	,250	,011	,000	,503	,681
	17	5	,000	,022	,000	,380	,614
	17	7	,000	,022	,000	,373	,621
	17	10	,000	,014	,000	,434	,666
	17	17	,227	,018	,000	,519	,707
	17	26	,238	,012	,000	,498	,692
	26	5	,000	,013	,000	,291	,664
	26	7	,250	,014	,000	,533	,693
	26	10	,250	,011	,000	,620	,681
	26	17	,238	,012	,000	,665	,692
	26	26	,333	,008	,000	,528	,759
Yaroslavl	5	5	,111	,016	,000	,432	,556
	5	7	,000	,019	,000	,528	,573
	5	10	,000	,020	,024	,376	,627
	5	17	,000	,019	,026	,498	,692
	5	26	,083	,021	,000	,451	,564
	7	5	,111	,019	,000	,523	,523
	7	7	,000	,021	,000	,418	,594
	7	10	,000	,018	,000	,507	,573
	7	17	,000	,017	,000	,521	,607
	7	26	,000	,019	,000	,538	,561
	10	5	,000	,030	,000	,527	,542
	10	7	,000	,020	,000	,347	,603
	10	10	,000	,020	,000	,509	,572
	10	17	,000	,017	,000	,423	,605
	10	26	,000	,021	,000	,531	,586
	17	5	,000	,022	,000	,537	,586
	17	7	,000	,025	,000	,439	,633
	17	10	,000	,023	,000	,583	,583
	17	17	,000	,021	,000	,538	,564
	17	26	,000	,022	,000	,570	,591
	26	5	,000	,022	,000	,500	,559
	26	7	,000	,016	,000	,324	,558
	26	10	,000	,016	,000	,418	,548
	26	17	,000	,017	,000	,340	,578
	26	26	,000	,016	,000	,425	,575

Table 1. Minimal values of quantities η, τ, χ, φ, ψ for fixed w_β and w.

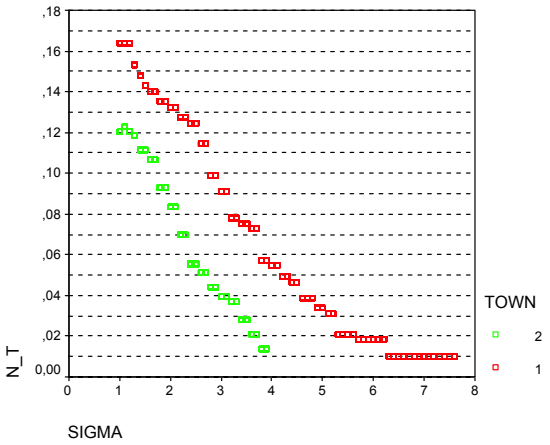


Figure 2. Dependence n (σ). There are following labeling: SIGMA – σ, N_T – n, TOWN = 1 corresponds to Tambov, TOWN = 2 – to Yaroslavl.

Another important quantity evaluating quality of prediction is percentage of false alarms, χ. Prediction is considered as effective if $\chi < 1/2$. From Table 1 we see the minimum values of χ are located in interval from 0,155 to 0,665. To evaluate together these three quantities we consider them sum $\psi = \eta + \tau + \chi$, which values from limitations above for appropriate prediction should be less 1,5. In our executed experiments values of ψ are located in interval from 0,518 to 0,759.

town	*p*	η		τ		χ		$\varphi = \eta + \tau$		$\psi = \eta + \tau + \chi$	
		min	*max*	*min*	*max*	*min*	*max*	*min*	***max***	*min*	***max***
Tambov	1	,20	,98	,01	,43	,00	,50	,52	**1,01**	,52	**1,51**
	2	,11	1,00	,02	,58	,00	1,00	,62	**1,05**	,76	**2,05**
	3	,05	1,00	,03	,65	,24	1,00	,50	**1,08**	,90	**2,08**
	4	,00	1,00	,03	,71	,70	1,00	,16	**1,27**	,96	**2,16**
Yaroslavl	1	,15	,96	,02	,46	,00	,57	,52	**,99**	,52	**1,60**
	2	,08	,96	,02	,60	,00	,67	,53	**,99**	,55	**1,65**
	3	,05	,96	,02	,66	,17	,83	,56	**1,01**	,90	**1,83**
	4	,00	1,00	,02	,70	,55	1,00	,32	**1,08**	1,08	**2,08**

Table 2. Minimal and maximal values of quantities η, τ, χ, φ, ψ for fixed p.

In addition to the heaviness levels we introduce another feature that distinguishes objects by its frequency – ratio of number of objects N to monitoring time T for corresponding town, $n = N / T$. It is clear that n depends on σ. A form of this dependence is obvious: the higher σ the smaller n (see Figure 2). We may consider this ratio as power of event, power of flash-up which we denote p. According to the form

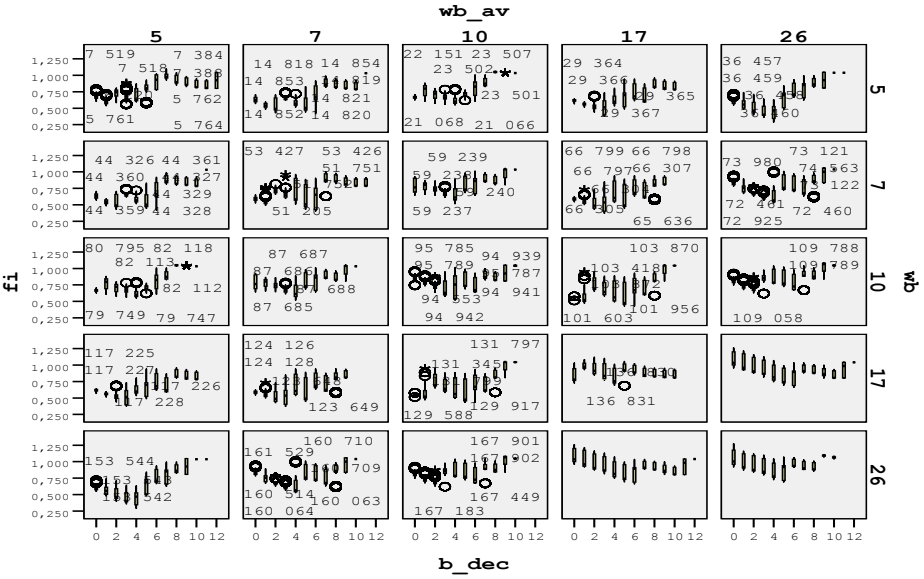


Figure 3. Dependence $\varphi(\Delta\beta)$, where $\Delta\beta = j$ if $0{,}1j < \beta \le 0{,}1\,(j+1)$, $j = 0,1,2,\ldots$ for Tambov as $p = 4$. There are following labeling: fi – φ, wb_av – w, wb – w_β.

of dependence of $n(\sigma)$ we mark out four intervals of n: $p = 4$ if $n \le 0{,}03$, $p = 3$ if $0{,}03 < n \le 0{,}06$, $p = 2$ if $0{,}06 < n \le 0{,}115$ and $p = 1$ if $n > 0{,}115$.

Table 2 represents minimal and maximal values of η, τ, χ, φ and ψ for corresponding p for each town. We see that for three first groups of p the quantity φ has close to each other values located nearly 0,55 when for the rarest objects φ is much less of these values: 0,16 for Tambov and 0,32 for Yaroslavl. But here, for $p = 4$, we obtain very large value of χ: 0,70 for Tambov and 0,55 for Yaroslavl.

Maximal values of φ are little bit higher 1. We should exclude cases where $\varphi > 1$. To do it we should study how φ depended on β, w and w_β. These dependences are represented on Fig.3 and Fig.4. It is clear from these figures that there are local minimums in dependences of $\varphi(\Delta\beta)$ almost for all combinations of parameters w and w_β. These minimums are located in interval 3-7 for $\Delta\beta$, i.e. 0,3-0,8 for β. In addition values of these local minimums are less 1. Hence, we find a way how to exclude the highest values of φ which are close to 1 and work in that parameters intervals which do not leads to inappropriate results.

4 Conclusions

As written in [1-3] crime prediction methods, in particular, homicide prediction could be used in tactical controlling of emergency town services, for instance, police, ambulance, hospitals etc. Having information about possible homicide flash-up over the next few weeks authorities could reinforce such services what allows to guarantee their faster reaction upon crime events. Decreasing of time reaction upon crime will

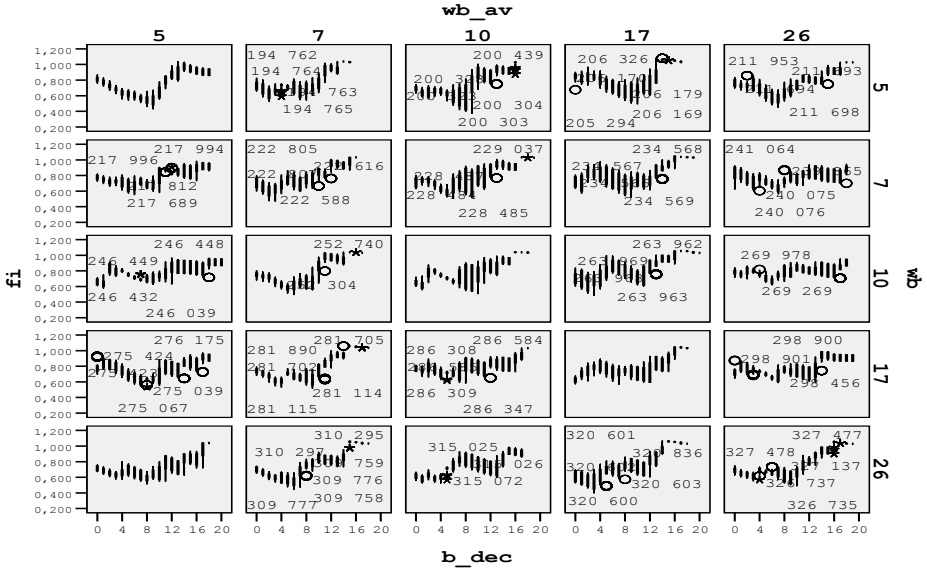


Figure 4. Dependence $\varphi(\Delta\beta)$, where $\Delta\beta = j$ if $0{,}1j < \beta \le 0{,}1\,(j+1)$, $j = 0,1,2,\ldots$ for Yaroslavl as $p = 4$. There are following labeling: fi – φ, wb_av – w, wb – w_β.

lead to decreasing of deceases and dangerous health hazards as a result of violent acts. Such information could help to effectively control working regime of emergency services' staff: vacations, free days, duties etc.

From economical point of view it is very important to have small enough percentage of false alarms which value is acceptable for frequent event and unfortunately unacceptable for most powerful, rare events.

Nevertheless we believe that developed algorithm is good enough and has a property of an universality – it is advisable to use our algorithm to predict serious crimes, in particular, homicides for different towns, regions with similar quality [4-5]. At least carried out investigations do not reveal properties of the algorithm which could indicate that the algorithm has narrow sphere of implementation: for huge range of parameters we obtain gradual changes of results' quantity.

As we didn't reveal a disproof of the main supposition of our method that regime of crime number dynamic is analogous in many respects to behavior of complex systems with precursor activation effects before serious crime flash-ups we will continue to study the developed algorithm. As we found a way to decrease value of one of the main quantity estimating prediction quality: a special range of β, the first following step is decreasing percentage of false alarms for rare events.

References

[1] D.V. Serebryakov, I.V. Kuznetsov, M.V. Rodkin. A serious crime flash-up forecast based on hierarchy of criminality behavior. Preprint, Inst. Appl. Math., the Russian Academy of Science, number 15, 2005. http://www.keldysh.ru/papers/2005/prep12/prep2005_12.html

[2] I.V. Kuznetsov, M.V. Rodkin, D.V. Serebryakov, O.B. Uryadov. Hierarchical way to crime dynamics. New in synergetic. New reality, new problems, new generation. Collected articles. Part 1 / Edited by G.G. Malinetskii. – M.: Radiotekhnika, 2006.

[3] D.V. Serebryakov, I.V. Kuznetsov and M.V. Rodkin. Hierarchical Approach to Dynamics of Criminality. Proceedings of European Symposium on Time Series Prediction, ESTSP'07, 7-8-9 February 2007, Espoo, Finland, pages 221-230.

[4] Dmitry V. Serebryakov, Igor V. Kuznetsov, Mikhail V. Rodkin, Oleg B. Uryadov. Hierarchical approach to criminality prediction. Programm and Abstracts, 27th International Symposium on Forecasting (ISF2007), June 24-27, 2007, New York City, p. 92

[5] D.V. Serebryakov, I.V. Kuznetsov, M.V. Rodkin, O.B. Uryadov. Prediction of serious crime flash-ups. Thesis of reports of II International conference "Mathematical modeling of historical prossecies" // Preprint, Inst. Appl. Math., the Russian Academy of Science, number 56, 2007, pages 58-59.

Neural Networks and their application in the fields of corporate finance

Eric Séverin[1]

1- University of Lille 1 - Dept GEA
Batiment SHS n°3, BP 179, 59653 Villeneuve d'Ascq cedex - France

Abstract.

This article deals with the usefulness of neuronal networks in the area of corporate finance.

Firstly, we highlight the initial applications of neural networks. One can distinguish two main types: layer networks and self organizing maps. As Altman al. (1994) underlined, the use of layer networks has improved the reclassifying rate in models of bankruptcy forecasting.

These first applications improved bankruptcy forecasting by showing a relationship between capital structure and corporate performance. The results highlighted in our second part, show the pertinence of the use of the algorithm of Kohonen applied to qualitative variables (KACM). More particularly, in line with Altman (1968, 1984), one can suggest the coexistence of negative and positive effects of financial structure on performance. This result allows us to question scoring models and to conclude as to a non-linear relationship.

In a larger framework, the methodology of Kohonen has allowed a better perception of the factors able to explain the leasing financing (Cottrell et al., 1996). This research, carried out with Belgian accounting data, highlights a relationship between leasing and the corporate financial strength. A following paper of this first study has been made using recent French data. The objective is here to explain the factors of the choice between leasing and banking loans. By using different variables, we highlight the characteristics of firms which most often use leasing. The corporate financing policy could be explained by: the cost of the financing, advantages of leasing or by the minimization of agency costs in leasing, we highlight a relationship between resorting to leasing and credit rationing.

1 Introduction

As underlined by Cottrell et al. [1], maps Kohonen allowed many applications and more specifically to finance. The areas in which they were used are varied: the detection of firms in financial distress, the choice of debt policy (for instance leasing)… In the case of financial distress, the objective is to develop a function able to discriminate 'good'(healthy) and 'bad' (financial distress) companies. So Altman [2] tried to determine -on the basis of variables observed during a long period of time- a Z function by using a discriminant analysis. The most of these methods are based on discriminant analysis and logistic regression but some statistical properties are not always checked. This leads to question the relevance of results.

Neural networks, contrary to the traditional statistical methods most often used in finance, do not make assumptions *a priori* on the variables. This is why they are able to deal with not structured problems (i.e. problems where it is not possible to specify the discriminating function *a priori*). With these algorithms, these systems are able to learn the relations between the variables starting from a unit data. This approach, still called step 'connexionnist' differs from the 'expert systems' because the user creates the base of knowledge and the rules which must be applied. The interest of the connexionnist approach is the following: the neural networks are able to learn themselves the relationships between the variables.
From this point, our discussion will be organized in three parts. In the first part, we will present the neural networks will show the relevance of their application in order to highlight the firms in financial distress. The second part will deal with the other recent applications of these networks in the field of corporate finance and more particularly leasing. The third part turns into the problem of leasing. The last section concludes.

2 Neural networks and the capacity of detect firms in financial distress

This section aims to introduce two types of neural networks: layered networks and self-organized maps called Kohonen maps.

2.1 Layered networks: a tool useful to detect firms in financial distress

These networks are organized in layers, each one of them has several neurons. Each neuron is an autonomous calculating unit and is connected with whole or part of the other neurons (located on the same layer or on the preceding layers).
In the field of corporate finance, the neurons located in the first layer receive some information which characterizes the firm. Generally these data are financial ratios. The exit neuron takes a binary value, zero or one, according to the firm is considered as financial distress or healthy. Each neuron collects information of the preceding layer with which it has relationship and calculates an activation potential.

Setting a network is done using a sample of learning. It is the learning algorithm that adjusts the synaptic weights by researching a minimization of cost function (Rumelhert et al.) [3].

2.2 **Artificial neural networks**

The principle is the following. After a first calculation, the exit result obtained is compared with the researched result exit. The total error made by the system is then 'backpropagated' from exit layer to entry layer and the synaptic weights are changed. It allows a new calculation. The implementation of the network requires: a sample of data used to parameter, a sample used for the validation and a third sample used to evaluate the capacities of generalization of the network.

During the training, the error decreases, until tending towards zero if the network architecture were correctly selected. However more the error is weak, less it is able to generalization.

2.3 **Empirical results concerning firms in financial distress**

An interesting contribution is realized by Altman et al. [4]. The authors test complex networks with several different sets of ratios. Our results highlighted a classification rate of 97.7% correct for healthy firms and 97% for firms in difficulty. These rates appear higher than those of discriminant analysis which is respectively 90.3% and 86.4%. Other studies (Bardos and Zhu) [5] found results that are consistent with the same first results. Besides, in the line of Udo [6] and De Almeida and Dumontier [7], .Casta and Prat [8] demonstrated the ability of artificial neural networks to deal with incomplete data, which remains common in this area of analysis. From this point, we will highlight the main results in the field of corporate finance.

3 Recent applications: SOM

This section highlights the application of unsupervised networks to corporate finance. It focuses on study dealing with the capital structure and performance through the algorithm Kohonen and more particularly KACM.
Capital structure can be view in two different ways. The first is the leverage, the second is the ownership structure. We try to show the interest of Kohonen algorithm to deal with the relationship between leverage and performance. Our section is organized as follows. In a first step, we will sum up the relationship between capital structure and performance. In a second step we will present the Kohonen methodology Kohonen. Finally, we will show results on empirical study.

3.1 **Leverage, performance and value: a complex relationship**

The financial literature has sought to measure the advantages of debt. Modigliani and Miller have shown that debt, in the absence of taxation, had not influence on the value of the firm [9 but, in case of taxation, this relationship is modified [10]. In the latter case, debt can create value. From this work, numerous studies have sought to assess the impact of debt on value. The results are contradictory. Some authors like Altman [1, 11], Collongues [12], debt is a source of bankruptcy and therefore destruction of value. Contrary to Modigliani and Miller, some authors such as Jensen[13] or Wruck [14] show that the debt has a positive influence on the firm value. Indeed, the debt can be used as a "sword of Damocles" because it constrained the leaders to undertake profitable projects able to generate liquidities to face their engagements. Even if the effects of the debt remains unclear, one can consider that there is a relationship between debt and value.

In line of Altman, we can consider the following relationship:
Decrease in performance ⇨ Increase of debt ⇨ Value destruction

For Opler and Titman [15], the relationship is different. They note that the debt is certainly destroying of value but they show, through an empirical work, that this value destruction value also has an influence on corporate performance and on the debt level. Indeed, Opler and Titman [15] specify that debt is a factor of « financial distress » likely to endanger the firm. Indeed, if a firm is in financial distress, the shakeholders can doubt its durability. For example, customers may be reluctant to do business with distressed firms. In other words, the stakeholders have no confidence in a firm which is not able to meet its commitments. The originality of Opler and Titman is to highlight the existence of indirect costs harmful to the firm before bankruptcy. In other words, they reverse the causality assumed by Altman.

In line of Titman and Opler [15], we can consider the following relationship:
Increase in debt in performance ⇨ Decrease of performance ⇨ Value destruction

3.2 **The justification and advantages ok Kohonen methodology**

If one considers the financial structure, the relationship between this concept and the performance is unclear.

When we try to determine if a non-linear relation exists between leverage and performance, we cannot use the techniques traditionally used in finance. Indeed, a great proportion of modern finance is based on the simplifying hypotheses; in particular, the normality of variables and the linearity of causality relations (Quintart) [16].

Our hypothesis of non-linearity led us to focus our attention on self-organized maps (SOM) and more especially on one of the variants called the Kohonen Map (Kohonen) [17 and 18]. One of the major advantages in its use is its capacity to deal with, in particular, non-linear problems (Quintart) [16].

Our objective was to determine several groups of homogenous individuals. Secondly, we used non-parametric tests (Wilcoxon and $\chi 2$) to highlight significant differences between our groups.

Many traditional methods assume strong hypotheses; in particular, the assumption of normality. To test this, we examined the distribution of the ratios. As our ratios do not have a normal distribution. Extreme values require the use of qualitative data. This non-normality and the presence of extreme values led us to cluster our individuals into 4 classes. Hence, firstly we transformed each character X_i into 4 categories (very strong, strong,weak, very weak[1]) and, secondly, transformed our variables into binary variables (De Bodt et al.) [18].

In a first step, in order to highlight the nature of relationship between debt and performance, we have realized a MCA. The findings are not satisfactory. The total inertia shown is weak (26%). The results are not reported.

Taking into consideration this disappointing result, we have decided to use a specific kind of self-organized map (SOM) called the Kohonen map[2]. The Kohonen algorithm[3] is a well-known unsupervised learning algorithm which produces a map composed of a fixed number of units (figure 1 presents a one-dimensional map, frequently called a string). Each unit has a specific position on the map and is associated with an n-dimensional vector Wi (which will define its position in the input space), n being the number of dimensions of the input space. A physical neighborhood relation between the units is defined (in figure 2, units 1 and 3 are neighbors of unit 2) and for each unit i, Vr(i) represents the neighborhood with the radius r centered at i.

After learning, each unit represents a group of individuals with similar features. The correspondence between the individuals and the units (more or less) respects the input space topology: individuals with similar features correspond to the same unit or to neighboring units. The final map is said to be a self-organized map, which preserves the topology of the input space as much as possible.
The learning algorithm takes the following form:

- at step 0, Wi(0) is randomly defined for each unit i,
- at step t, we present a vector x(t) randomly chosen according to the input density f and we determine the winning unit i*, which minimizes the Euclidean distance between x(t) and Wi(t),
- we then modify the Wi in order to move the weights of the winning unit i* and its physical neighbors towards x(t), using the following relations :

$$W_i(t+1) = W_i(t) + \left[\varepsilon(t) \times \left(x(t) - W_i(t)\right)\right] \text{ for } i \in V_{r(t)}(i^*) \quad (1)$$

[1] We divided the sample into four classes of equal size.
[2] An extensive presentation can be found in Kohonen [18].
[3] The Kohonen algorithm led to numerous theoretical studies: Cottrell, Fort and Pages [20] and Ritter and Schulten [21].

$$W_i(t+1) = W_i(t) \text{ for other } i \quad (2)$$

where ε(t) is a small positive adaptation parameter, r(t) is the radius of Vr(t) and ε(t) and r(t) are progressively decreased during the learning[4].

This is a competitive kind of algorithm (each unit competes to be the closest to the presented individual) which will perform two interesting tasks for data analysis:
- a clustering : each unit will be associated with a similar kind of individual, the Wi vector associated with the unit converging toward the mean profile of the associated individuals.
- a reduction in the number of dimensions : the (at least local) proximities between the units will give us an idea of the proximities of clusters of individuals in the input space.

A last remark concerning the neighborhood: it is reduced progressively to finish at value 0 (only the winning unit is displaced). The Kohonen algorithm then turns into a vectorial quantification. To assess the statistical significance of the results obtained with the Kohonen map, we used traditional non parametric tests (Wilcoxon[5], chi-square[6]).

3.3 **Financial structure and performance: an unclear relationship**

If one considers the financial structure, the relationship between this concept and the performance is unclear.

DeBodt et al. [19] made a paper on the relationship between financial structure and performance on the French firms. The results are the following.

[4] For stochastic algorithm ε(t) must follow the requirements of Robins-Monro [22].
[5] It is a test on ranks. Its justification is due to the no normality of data. Tests on ranks are very robust. By arranging the different observations (i.e. by giving them a rank), one identifies the place of every observation in the sample. One substitutes rank for observation. Thus one neutralises problems bound to the accurate measure of the value for every observation. We can note too that the results of rank tests are not altered by the distributions of observations (symmetrical, non-symmetrical...).
[6] The χ^2 test is a test of independence that serves to determine if samples come from the same population.

LEVVS VSGVW VSRVW VOIVW	LEVW VSGW VSRW VOIW	LEVVW VSGS VSRS VOIS	LEVS VSGVS VSRVS VOIS
Unit 1	Unit 2	Unit 3	Unit 4

Legend:
LEVVS, LEVS, LEVW, LEVVW: Leverage very strong, strong, weak, very weak.
VSGVS, VSGS, VSGW, VSGVW: Variation in sales growth very strong, strong, weak, very weak.
VSRVS, VSRS, VSRW, VSRVW: Variation in stock return very strong, strong, weak, very weak.
VOIVS, VOIS, VOIW, VOIVW: Variation in operating income very strong, strong, weak, very weak.
Leverage is measured for 1991, Variation in sales growth, of stock return and operating income are measured for 1993.
Table 1: Relationship between performance and financial structure
(From de DeBodt et al., 19)

The representation above leads to the conclusion that the more leveraged firms (firms located in the unit 1) have the worst performances (measured with accounting and market ratios). However, the influence of leverage on the performance seems complex. Indeed companies whose debt is high are also more efficient. These companies located in Unit 4 seem to confirm the hypothesis of free cash flow Jensen [13]. Coexistence of positive and negative effects of debt on performance leads us to note that the link debt-performance is not linear.
This leads us to question the relevance of most scoring models. Indeed, in theses models the leverage has a negative (and significant) influence on performance and increases the probability of bankruptcy.

3.3.1 Explicatives variables able to predict difficulties

Observations			Mean		1° Quartile		Median		3° Quartile	
Distressed firms	Effective firms	Characteristics of the firms	Distressed firms	Effective firms	Distressed firms	Effective firms	Distressed firms	Effective firms	Distressed firms	Effective firms
43	157	WC/SA 91 (%)	32.30	24.81	19.02	13.86	29.71	22.96	43.4	32.77
26	87	II/SA 91 (%)	9.16	7.0	3.45	3.28	5.91	4.77	10.27	7.59
43	157	FA/TA 91 (%)	36.34	36.16	22.44	24.18	34.09	33.89	45.65	48.55
43	157	CA/FA 91 (%)	14.27	14.48	2.65	3.22	8.89	9.84	20.83	20.23
43	157	SA 91 in millions of French Francs	7173.593	13286.340	431.060	577.911	1309.697	1649.535	4001.968	7926.325

43	157	TA 91 in millions of French Francs	9314.354	13235.791	397.016	471.895	1159.066	1480.734	3612.653	7190.321

Table 2: Descriptive statistics of industrial and investment features (in 1991) of distressed firms (unit 1) and other firms (units 2, 3 and 4). (From de De Bodt et al., 19)

Firms' characteristics	Wilcoxon statistic	P value	Significance
WC/SA91	-2.187[7]	0.029	**
II/SA91	-1.139	0.255	NS
FA/TA91	-0.219	0.827	NS
CA/FA91	-0.346	0.729	NS
SA91	-1.286	0.198	NS
TA91	-1.075	0.282	NS

Note : *** significant at the 1% threshold ; significant at the 5% threshold ; significant at the 10% threshold ; NS (Not significant).

Table 3: Wilcoxon test on the industrial and investment features of distressed (firms in unit 1) and others firms (units 2, 3 and 4). (From de De Bodt et al., 19)

The last table is interesting and suggests that the negative impact of debt on performance is even more pronounced when companies have long operating cycles (result significant at the 1% threshold). During a crisis period, firms can be tempted to reduce their working capital by using more credit supplier.

3.3.2 Financial structure and performance: a dynamic relation

We wished to know if distressed firms increased their leverage correlatively with their difficulties. In other words, does decrease in performance lead to an increase in leverage? If this relationship is right, our results will be consistent with those of Altman [11] We used the same methodology as before.

Variation in leverage from 1991 to 1993	Number of observations	Mean	Quartile One	Median	Quartile Three
Distressed firms (%)	43	2.80	9.2	-0.73	15.01
Effective and very effective firms (%)	157	-0.5	-24.32	-10.18	7.35

Table 4: Descriptive Statistics of leverage variation of distressed (unit 1), effective and very effective firms (units 2, 3 and 4). (From de De Bodt et al., 19)

[7]When comparing unit 4 with the others, a similar result is found, that is the working cycle of firm in that category compared with others is lower at the significance threshold of 5% (-2.047). Likewise when comparing firms in units 1 and 4 only, the results confirm those already evidenced (-2.065 significant at the 5% threshold).

Variation in leverage from 1991 to 1993	Number of observations	Mean	Quartile One	Median	Quartile Three
Distressed firms (%)	43	2.80	9.2	-0.73	15.01
Effective and very effective firms (%)	157	-0.5	-24.32	-10.18	7.35

	Wilcoxon statistic	P value	Significance
Leverage (Δ 1991- 1993)	-2.518	.012	**

Note : ** significant at the 5% threshold.

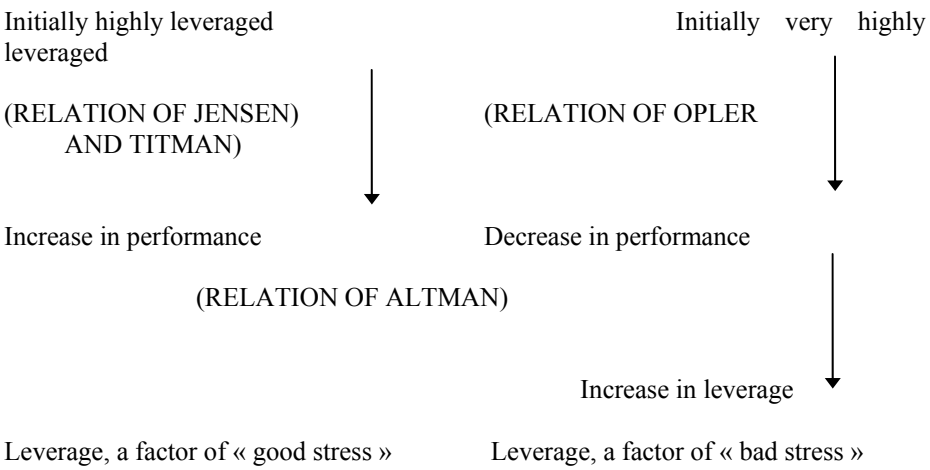
Table 5: The Wilcoxon test on the variation of leverage (between 1991 and 1993) of distressed and others firms. (From de De Bodt et al., 19)

Descriptive statistics (table 4) indicated that on average distressed firms increased their debt (2.80%) whereas effective firms decreased their debt (-5.06% on average). This result allows us to think that debt-increase is passive and highlights a mechanical consequence of working operating difficulties. In other words, drop in performance has an adverse effect on the debt level. Thus decrease in performance mechanically led to an increase in difficulties for distressed firms.

The Wilcoxon test showed us that the variation in leverage between distressed and effective firms is statistically different (significant at the 5% threshold).

Hence highly-leveraged firms are less efficient than others and are also subject to the greatest variation in their leverage (significant at the 5% level).

The relationship between leverage and performance is dynamic and our results are consistent with those of Opler and Titman [15] and Altman [11]. We can summarize it as follows:



(From De Bodt et al., 2001)

4 Another example: the leasing case

4.1 An first approach: leasing as a substitute for bank loans

The question of substitutability or complementarity of leasing and debt has been studied in details by the financial literature but without reaching a consensus of unanimous reply. According to Myers et al. [23], the use of leasing would be probably accompanied by a lesser utilization of a debt, reducing, in the same proportion, the firm's debt capacity (Levasseur and Quintart) [24]. The validity of such an argument is based on the perfect substitution hypothesis between leasing and bank debt. Therefore, this perfect substitution hypothesis can be rejected in favour of an imperfect substitution and sometimes in favor a complementarity between banking loans and leasing.

Two arguments can be advanced.

The first argument against a perfect substitution between leasing and debt concerns the substantial costs borne by the (banking) creditor or (credit) lessor in case of corporate bankruptcy. For Krishnan and Moyer [25], costs created by the bankruptcy of the tenant would be lesser for the lessor that for any other creditor. The quality of ownership allows him to recapture the goods in case of corporate bankruptcy when the contract is not stopped. In fact, it avoids opportunity costs associated with the slowness of the resolution process of the bankruptcy and more rapidly may allow to resale the asset.

The second argument contradicting perfect substitutability between leasing and debt is based on properties of leasing. By the terms of the contract and rights given to the lessor, leasing has characteristics close both to the secured and unsecured debt. The secured debt, because lessor has a real guarantee in the ownership of asset in the contract; the non-secured debt, because unpaid rents and banking loan prior before bankruptcy have the same rank of priority. In most of cases, the funds are lost by the creditor. The diversity of the debts contracted by the firm, introducing a distortion between creditors in case of bankruptcy, favours the thesis of the imperfect substitution leasing/ classical debt (Stulz and Johnson) [26].

The question of complementarity or substitutability of debt and leasing seems to be unsolved because there is no consensus between theoretical and empirical approaches. Leasing is often analyzed as a last resort solution, especially for very weak firms. Firms' financing preferences nevertheless depend on the characteristics. It allows lessor to gain a clientele of firms with atypical profiles. The reasons for the choice of leasing versus bank loan can therefore be several, which we develop in the following point.

As underlined by Smith and Wakeman (p. 907) [27]: "the coexistence of both leased and purchased assets suggests that the net benefits of leasing are uniformly neither positive or negative ". In this vein, the understanding leasing choice versus bank loan for the firm would depend on costs/advantages.

The empirical study, proposed in the next paragraph, based on a sample of French firms differs from previous ones, in two points.
The first concerns the methodology employed. The empirical analysis of De Bodt et al. [28] and Cottrell et al. [29] confirms viewpoints advanced in the financial literature. They do not allow to develop a discrimant analysis able to distinguish between firms that use leasing and those that do not. Moreover, by considering the very particular distribution of some ratios, the authors make a comparative analysis of a technique of linear data analysis (multiple correspondence analysis) and a technique of non linear analysis (Kohonen maps). The interest is in the results obtained by the KACM. Indeed, observations by the authors suggest firstly, the existence of subgroups within the population of firms that use leasing, and secondly, a clear association between the use of leasing and financial health of firms. Even if the KACM method presents weak points (The first is that the visualization of the Kohonen map does not allow evaluation of the distance between the units. The second is the quality of the representation), its use can bring progress in the understanding of leasing by the use of Kohonen maps. The use of Kohonen maps favors a roundup of individuals according to their propensity to use leasing utilization, which allows a finer analysis of firms' profiles.
The second contribution of our study concerns the utilization of a direct measure of the credit rationing.

4.2 **Data and methodology on French data**

4.2.1 Sample presentation and variables choice

Our study deals with a sample of 11436 French SME. We used the date from Dun and Bradstreet, for 1999. As we sought accounting data which represent leasing financing, we proceeded in several steps. Firstly, we retained all firms with a staff between 20 and 500 employees. By this, we wanted to select SME by excluding very small firms in order that the information would be less difficult to deal with. From this point, we choose to retain only industrial and services firms using leasing. Indeed, we excluded firms in the financial sector (the accounting treatment of profit for these firms is significantly different than that of other sectors). After applying these criteria 12669 were retained.

The validity of accounting data was validated by using a series of coherence tests. Hence our sample was composed of 11233 firms. Then, we worked from the Dun database.
In this database, we used the following information: number of workers, long term debt, leasing, equity, short term assets, short term liabilities, EBITDA, financial fees, fiscal debt and firm age.
The collection of these data allowed the construction of variables used in empirical tests.

In this study, we seek to clarify explanatory factors in the propensity of the firm to finance by leasing. To take into account the intensity of credit lease utilization, we

calculated the variable L (Leasing), by calculating the ratio of leasing divided by the long term debt for each observation.

The literature analysis presented in the first section allowed several arguments to be identified to explain the choice of leasing/banking loan for the firm. It concerns the firm's risk level, information asymmetry between lender and borrower, the firm's debt capacity and the dissociation between judicial and economic ownership. We summarize in table 6 for each determinant the financing policy by leasing. The lack of some data and information explains we have limited our investigation to these main four arguments. Table 7 presents the variable measurement used and the expected effects on the propensity of the firm to be financed by leasing.

Theoretical viewpoint	Argument	Chosen Variables
The risk of the borrower	The use of leasing is positively associated with the bankruptcy risk (Krishnan and Moyer) [25],	- Probability of bankruptcy - Firm's solvability
Informational asymmetry on quality of growth opportunities	Leasing is all the more used when the firms are young and small (Sharpe and Nguyen) [30].	- Firm size - Age
Limited debt capacity	Firms with real debt capacity should use leasing more frequently (Krishnan and Moyer) [25].	- Leverage
The separation between judicial and economic ownership	The acquisition of judicial ownership is optional in leasing contract. The firm avoids transaction costs due to retrading the good on the second market. So the small-sized firm would probably used leasing more frequently (Smith and Wakeman,) [27]. The loss of the legal ownership leads to loss in flexibility for the firm.	- Firm size

Table 6: Theoretical viewpoint for using leasing and variables choosen

Variable	Ratio	Expected sign
Probability of bankruptcy *(DEF)*	EBITDA / Financial expenses	positive
Solvability *(SOL)*	Cash flow / financial debt + leasing	negative
Size *(TA)*	Log (total workers)	negative
Age (AGE)	Number of years of life since the firm's creation	negative
Leverage (*LEV)*	Long term debt + leasing / equity	positive
Control Variable	Ratio	
Credit rationing *(CR)*	Fiscal and corporate debt/going concern debt	In the case of substitution hypothesis, we expect a negative relationship between credit rationing and the amount of leasing

Table 7: Explanatory variables and expected effects

4.2.2 Results obtained by Kohonen maps

In this section, we present the main results obtained from the firms' sample.

We show the results obtained for the different variables. We associated the dependent variable to the explanatory variable, by highlighting different levels: VS: very strong, S: strong, W: weak and VW: very weak.

<u>Theoretical hypothesis</u>: The risk level of the borrower

LVW DEFS	LW DEFVW	LS DEFW	LVS DEFVS
Unit 1	Unit 2	Unit 3	Unit 4

Legend
LVS, LS, LW, LVW: Leasing very strong, strong, weak, very weak.
DEFVS, DEFS, DEFW, DEFVW: Probability of bankruptcy very strong, strong, weak, very weak.
U1, U2, U3 et U4: Unit 1, Unit 2, Unit 3, Unit 4.
Table 8: Probability of the bankruptcy and the use of leasing**.**

Theoretical argument: The risk level of the borrower

LVS SOLW	LS SOLVS	LW SOLS	LVW SOLVW
Unit 1	Unit 2	Unit 3	Unit 4

Legend
LVS, LS, LW, LVW: Leasing very strong, strong, weak, very weak.
SOLVS, SOLS, SOLW, SOLVW: Solvability very strong, strong, weak, very weak.
U1, U2, U3 et U4: Unit 1, Unit 2, Unit 3, Unit 4.
Table 9: Debt capacity and the use of leasing.

Theoretical argument:- Informational asymmetry between lender and borrower
-separation between judicial and economic ownership

LS TAW	LVW TAVS	LW TAS	LVS TAVW
Unit 1	Unit 2	Unit 3	Unit 4

Legend
LVS, LS, LW, LVW: Leasing very strong, strong, weak, very weak.
TAVS, TAS, TAW, TAVW: Size (total assets) very strong, strong, weak, very weak.
U1, U2, U3 et U4: Unit 1, Unit 2, Unit 3, Unit 4.
Table 10: Firm size and the use of leasing.

Theoretical argument: Informational asymmetry between borrower and lender

LVS AGEVW	LW AGEW	LS AGES	LVW AGEVS
Unit 1	Unit 2	Unit 3	Unit 4

Legend
LVS, LS, LW, LVW: Leasing very strong, strong, weak, very weak.
AGEVS, AGES, AGEW, AGEVW: Age very strong, strong, weak, very weak.
U1, U2, U3 et U4: Unit 1, Unit 2, Unit 3, Unit 4.
Table 11 : Firm age and the use of leasing.

Theoretical argument: debt capacity of the firm

LVW LEVVW	LVS LEVVS	LS LEVS	LW LEVW
Unit 1	Unit 2	Unit 3	Unit 4

Legend
LVS, LS, LW, LVW: Leasing very strong, strong, weak, very weak.
LEVVS, LEVS, LEVW, LEVVW: Leverage very strong, strong, weak, very weak.
U1, U2, U3 et U4: Unit 1, Unit 2, Unit 3, Unit 4.
Table 12: Leverage and the use of leasing.

4.2.3 Comments and analysis

Empirical test results highlight significant effects for the different factors explaining the use of leasing. Indeed, firms use leasing especially when:
they have small size
they are young
they have a smaller solvability
they present a strong likelihood of bankruptcy.

These profiles testify a stronger risk of failure. The results suggest that leasing is often used when the firms are constrained by credit rationing. (Krishnan and Moyer, [25] ; Sharpe and Nguyen) [30]).

We have consequently sought to check if our results were robust, by seeing the relationship between explanatory variables and the credit rationing. To do this, we evaluated the level of credit rationing of the firm by the following ratio: fiscal and corporate debt/ going concern debts. The Wilcoxon test shows significant results for the following variables: size, leverage, probability of bankruptcy and solvency. We verify consequently that relationships have their origin in credit rationing. Table 13 summarizes the results.

Variables	Wilcoxon statistics	Interpretation
Size	-2.147**	Credit rationing is positively associated with a small size
Leverage	-16.03***	Credit rationing is positively associated with highly leveraged firms
Solvency	-5.31***	Credit rationing is negatively associated with firms' solvency
Probability of Bankruptcy	-10.1***	Credit rationing is positively associated with the probability of bankruptcy
Age	-1.11	Credit rationing is negatively associated with firms' age

Note: *, ** and *** significant at the 10%, 5% and 1% threshold.
Table 13: Credit rationing intensity and financing policy by leasing: Wilcoxon test.

On the French SME sample, our results suggest that the use of leasing is positively associated with credit rationing. This confirms that the use of leasing is a "last resort solution". Otherwise, the financial literature does not totally explain the motives of credit rationing. Is credit rationing due to too high leverage or informational asymmetry on the borrower? In this context, leasing could be preferred by young firms and start-ups.

5 Conclusion

We will end this paper with:
the advantages and the disadvantages of neural networks systems
our results.

The advantages are numerous compared to classical statistical analyses. On the one hand, they allow problems to be investigated for which we have a priori non information. Thus in the framework of the detection of firms in financial distress, it is not necessary to know the variable distribution (contrary to discriminant anlysis). Secondly, the neuronal systems discover by themselves relationships between variables which allow us to study non-linear problems. Thirdly, the uncompleted data can be taken into account by the supplementary neuron addition. Fourthly, the stop of the iterative process -when the system produces the best results on the validation sample- gives robust results. One can consider that the relevant information is integrated in the system. Fifthly, neuronal systems allow to work on qualitative and quantitative variables.

Despite these advantages, several criticism can be addressed to neural networks. These are following. Firstly, it does not exist theory allowing to determine the optimal structure of the system. Especially the determination of the hidden layers number and the number of neurons are, the most often, dependant from the user and its capacity to experiment several architectures. Secondly, neural networks often assimilated to "black boxes" in which it is difficult to extract relevant relationships among variables.

However, the studies presented above show that neural networks give good results in the classification area. To be able to improve the approach by neuronal networks, efforts have to focus on: the construction of the system, the clustering of entry variables and the adjustment of learning parameters that depend greatly on human intervention.

After these papers, the next research will be focus on the problem of prediction which will be developed in the next presentations.

References

[1] M. Cottrell, E. De Bodt and M. Levasseur., Les réseaux de neurones en finance : Principes et revue de la littérature, *Finance*, 16(1): 25-91, 1996

[2] .E.I. Altman, Financial ratios, discriminant analysis and the prediction of corporate bankruptcy », *Journal of Finance*, XXIII(4): 589-609, 1968.

[3] D.E. Rumelhart, G. Hinton and Williams R., Learning representation by back propagation errors, *Nature*, 323: 9, 1986.

[4] Altman, G. Marco and F. Varetto, Corporate distress diagnosis: comparison using linear discriminant analysis and neural networks (the italian experience), *Journal of Banking and Finance*, 18, 1994.

[5] Bardos and W. Zhu, Comparaison de l'analyse discriminante linéaire et des réseaux de neurones : application à la détection de défaillance d'entreprises, in Actes de la deuxième rencontre internationale sur l'approche neuronale en sciences de gestion, Poitiers, 1995

[6] G. Udo, Neural network performance on the bankruptcy classification problems, *Computer and Industrial Engineering*, 25: 4, 1993.

[7] F. De Almeida and P. Dumontier, 1993, Réseaux de neurones, information comptable et détection du risque de défaillance, 14ème congrès de l'Association française de comptabilité, 1993.

[8] J.F. Casta and B. Prat, Approche connexioniste de la classification des entreprises : contribution au traitement d'informations incomplètes, CREREG, Paris Dauphine, 1994.

[9] Modigliani and M. Miller, The cost of capital, corporation finance and the theory of investment, *American Economic Review*, 49: 261-297, 1958.

[10] F. Modigliani and M. Miller, Corporate income taxes and the cost of capital : a correction, *American Economic Review*, volume 53: 433-443, 1963.

[11] E.I. Altman, A Further empirical investigation of the bankruptcy cost question, *Journal of Finance*, XXXIX(4): 1067-1089, 1984.

[12] Y. Collongues, Ratios financiers et prévision des faillites des PME, *La Revue Banque*, 365 : 963-970, 1977.

[13] J.C. Jensen, Agency costs of free cash flow, corporate finance and takeovers, *American Economic Review*, volume 76: 323-329, 1986.

[14] K.H. Wruck, Financial distress, reorganization and organizational efficiency, *Journal of Financial Economics*, volume 27: 662-676, 1990.

[15] T. Opler and S. Titman, 1994, Financial distress and corporate performance, *Journal of Finance*, XLIX(3): 1015-1040, 1994.

[16] A. Quintart, Les fondements de la théorie financière classique et les limites de l'hypothèse générale de linéarité, in De Bodt et Henrion (ed), Les réseaux de neurone en finance : conception et applications, Bruxelles, D Facto Publications, pages 1-21, 1995.

[17] T. Kohonen, «Self organized formation of topologically correct feature maps, *Biological Cyberbetics*, 43, 59 p, 1982.

[18] T. Kohonen, Self organizing maps, Springer Series in Information Sciences », vol 30, Springer, Berlin, 1995.

[19] E. De Bodt, M. Levasseur and E. Séverin, Debt a factor of good and bad stress during an economic recession : Evidence from France, *Fuzzy Economic Review*, Vol VI, n°1: 63-87, 2001.

[20] M. Cottrell, J.C. Fort and G. Pages, Etude d'un algorithme d'auto-organisation, *Annales de l'Institut Henri Poincaré*, vol 23, #1 : 1-20, 1987.

[21] H. Ritter and K. Schulten., Convergence properties of kohonen's topology conserving maps: fluctuations, stability and dimension selection, *Biol Cybernetics*, 60: 59-71, 1988.

[22] H. Robins and H. Monro, A stochastic approximation method, *Annales de Mathématiques et de Statistiques*, 22 : 400-407, 1951.

[23] S.C. Myers, D. Dill and A. Bautista, 1976, Valuation of financial leasing contracts, *Journal of Finance*, volume 31: 799-819, 1976.

[24] M. Levasseur and A. Quintart, 2000, La capacité d'endettement, *Banque et marchés*, n° 45 : 5-20, 2000.

[25] V. Krishnan and R. Moyer, Bankruptcy costs and the financial leasing decision, *Financial Management*, volume 23, n° 2, summer: 31-42, 1994.

[26] R. Stulz and H. Johnson, An analysis of secured debt, Journal of Financial Economics, volume 14: 501-521, 1985.

[27] C.W. Smith and L. Wakeman, Determinants of corporate leasing activity, *Journal of Finance*, volume 40, n°3: 895-911, 1985.

[28] E De Bodt, E.F. Henrion, C. Van Wymeersch and A. Wolfs, Le leasing financier : complément ou substitut du crédit à l'investissement ? Quelques résultats empiriques, in Approches Neuronales en Sciences Economiques de gestion, Futuroscope de Poitiers, 1995.

[29] M. Cottrell, E. De Bodt and E. Henrion, Understanding the leasing decision with the help of a Kohonen map -A empirical study of the Belgian market-, ICCN Congress, 1996.

[30] S. Sharpe and H. Nguyen, Capital market imperfections and the incentive to lease, *Journal of Financial Economics*, volume 39: 271-294, 1995.

Evolution of Interest Rate Curve: Empirical Analysis of Patterns Using Nonlinear Clustering Tools

M. Kanevski[1], V. Timonin[1], A. Pozdnoukhov[1], M. Maignan[2] *

1 - Institute of Geomatics and Analysis of Risk (IGAR), University of Lausanne, Switzerland
2 - Banque Cantonale de Geneve (BCGE), Geneva, Switzerland

Abstract. The present study deals with the empirical analysis of patterns in the evolution of interest rate curves (IRC). The main topic is to consider IRC as objects (curves) embedded into high-dimensional space and to study similarities and differences between them. This is a typical problem of clustering and classification in machine learning. In fact, theses data – IRC, can be considered as functional data set. Machine learning algorithm, namely Self-Organising Map - SOM (Kohonen map) [1], was used to study the evolution of interest rates and to reveal the potential patterns and clusters in IRC. Case study is based on Swiss franc (CHF) data on daily interest rates. For the analysis both raw data (curves composed of 13 non-regularly distributed maturities - from 1 week to 10 years) and data completed by interest rates mapping in a feature space of date-maturity were studied [2]. In the latter case curves are composed of 120 regularly (by month) distributed maturities. Feasibility study and preliminary results on IRC patterns analysis first time were presented in [3].

1 Introduction

Interest Rates Curve (IRC), or yield curve, is the relation between the interest rate (or cost of borrowing) and the time to maturity of the debt for a given borrower in a given currency [Wikipedia]. So interest rates depend on time and maturity which defines term structure of the interest rate curves. Temporal evolution of IR data for all maturities considered in this study is presented in Figure 1. Data cover the period from the end of 1998 to the beginning of 2006. Missing data can be observed at the end of year 2000.

In general, the information on interest rates dynamics is available for several fixed time intervals (daily, weekly, monthly) and for some definite maturities. In this study interest rate curves are composed of LIBOR daily rates with maturities 1 week, 1, 2, 3, 6, 9 months and SWAP rates with maturities 1, 2, 3, 4, 5, 7 and 10 years. The behavior of different maturities is coherent and consistent in time which can be confirmed by the corresponding global correlation matrix between different maturities (Figure 2).

There are some well known important stylised facts (typical and stable behaviour) that have to be taken into account during analysis, modelling and interpretation of IRC [4]:

* The work was supported in part by the Swiss National Science Foundation projects N 200021-113944 and N 100012-113506.

- The average yield curve is increasing and concave.
- The yield curve assumes a variety of shapes through time, including upward sloping, downward sloping, humped, and inverted humped.
- IR dynamics is persistent, and spread dynamics is much less persistent.
- The short end of curve is more volatile than the long end.
- Long rates are more persistent than short rates.

Fig. 1. Temporal evolution of CHF interest rates

	1W	1M	2M	3M	6M	9M	12M	2_YEAR	3_YEAR	4_YEAR	5_YEAR	7_YEAR	10_YEAR
1W	1.00	0.99	0.98	0.98	0.96	0.95	0.93	0.88	0.84	0.82	0.79	0.74	0.68
1M	0.99	1.00	1.00	0.99	0.98	0.96	0.95	0.91	0.87	0.84	0.82	0.77	0.71
2M	0.98	1.00	1.00	1.00	0.99	0.98	0.97	0.92	0.89	0.86	0.84	0.79	0.73
3M	0.98	0.99	1.00	1.00	0.99	0.99	0.98	0.94	0.91	0.88	0.86	0.81	0.75
6M	0.96	0.98	0.99	0.99	1.00	1.00	0.99	0.96	0.93	0.91	0.89	0.84	0.78
9M	0.95	0.96	0.98	0.99	1.00	1.00	1.00	0.97	0.95	0.93	0.90	0.86	0.80
12M	0.93	0.95	0.97	0.98	0.99	1.00	1.00	0.98	0.96	0.94	0.92	0.88	0.82
2_YEAR	0.88	0.91	0.92	0.94	0.96	0.97	0.98	1.00	0.99	0.98	0.97	0.94	0.89
3_YEAR	0.84	0.87	0.89	0.91	0.93	0.95	0.96	0.99	1.00	1.00	0.99	0.96	0.92
4_YEAR	0.82	0.84	0.86	0.88	0.91	0.93	0.94	0.98	1.00	1.00	1.00	0.98	0.95
5_YEAR	0.79	0.82	0.84	0.86	0.89	0.90	0.92	0.97	0.99	1.00	1.00	0.99	0.96
7_YEAR	0.74	0.77	0.79	0.81	0.84	0.86	0.88	0.94	0.96	0.98	0.99	1.00	0.99
10_YEAR	0.68	0.71	0.73	0.75	0.78	0.80	0.82	0.89	0.92	0.95	0.96	0.99	1.00

Fig. 2. Global correlation matrix for all maturities. Light gray cells are with values more than 0.95

Some interesting studies concerning interest rates analysis were presented in [5] and [6]. In [5] a general method to study the hierarchical organization of financial data by embedding the structure of their correlations into multi-dimensional spaces with a clear cluster differentiation was proposed. In [6] an empirical analysis of interest rates in money and capital markets was performed on the set of different weekly interest

rates. The study was focused on the collective behavior of the stochastic fluctuations of the time series which was investigated with a clustering linkage procedure. The separation in several clusters organized in a hierarchical structure was demonstrated.
The main task of the present study deals with the analysis of IRC as objects embedded into 13-dimensional space (equal to the number of maturities) and application of nonparametric nonlinear tool – Self-Organising maps in order to find some typical classes of IRC. A priori hypothesis is that typical curves are clustered in time reflecting some market conditions.

2 SOM classification of interest rate curves

Detailed description of SOM theory and corresponding learning algorithms can be found in [1].

SOM structure used for the current study is 10x10 hexagonal grid with bubble neighbour function. A 13-dimensional input feature space corresponding to the number of maturities was build. Only daily interest rate values were used for the classification. No date/time information was presented to the model. U-matrix of the trained SOM is used as a visualization tool describing the structure of data. The following step of the analysis is to apply another declustering algorithm to the trained SOM, e.g. k-Means algorithm, to detect cluster structure of the data.

Let us consider some results. It was found that 3 classes do not reveal IRC patterns but with 4 classes the major IRC groups were detected. U-matrix of the trained SOM with 4 predefined clusters is presented in Figure 3. In Figure 4 the examples of the IR curves (rate vs. maturity) for each of 4 clusters are presented. Figure 5 gives temporal structure (distribution of IRC classes in time) for all 4 classes. One can see that the behaviour of the curves is similar inside clusters and dissimilar for different clusters.

Let us examine the U-matrix in details (Figure 3). The advantage of the SOM mapping is that one can not only divide the space into clusters in some manner but also it is possible to see some details in the structure of the data. For example, let us compare cluster 1 with cluster 2. Cluster 1 is located in the area where dark colours (corresponds to smaller distances between cells) are dominated. On the contrary, cluster 2 is located in the area where light colours are dominated. It means that IR curves from class 1 are more similar to each other than between class 2 (see Figures 4 and 5 for demonstration). Another useful feature of the SOM mapping is a presentation of a topological structure of the data. For example, obtained topology demonstrates that class 3 is closer (more similar) to the class 4 than to the class 2 (see Figures 4 and 5 for the demonstration).

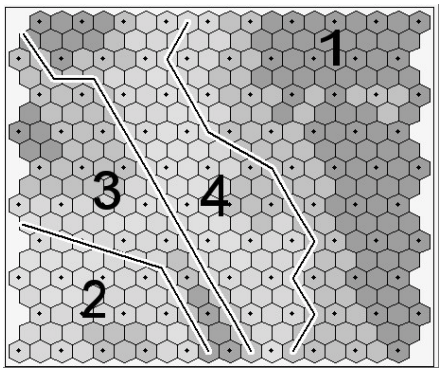


Fig. 3. SOM U-matrix map classified by k-Means clustering method. Predefined number of clusters equals to 4

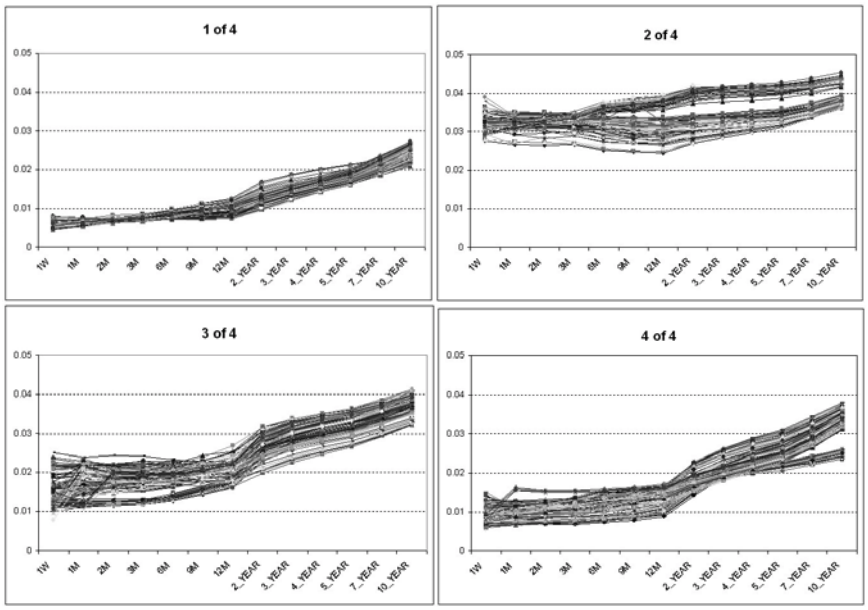


Fig. 4. Examples of the IR curves (rate vs. maturity) for each of 4 clusters

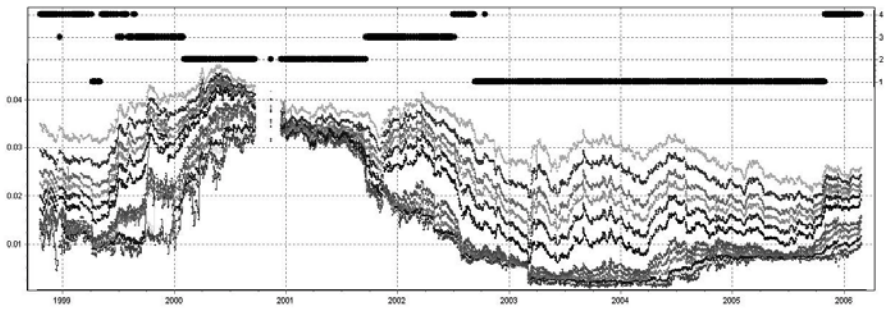


Fig. 5. Classification result for all maturities in time graphs. Black points on the top correspond to the 4 clusters detected by SOM (see right axis)

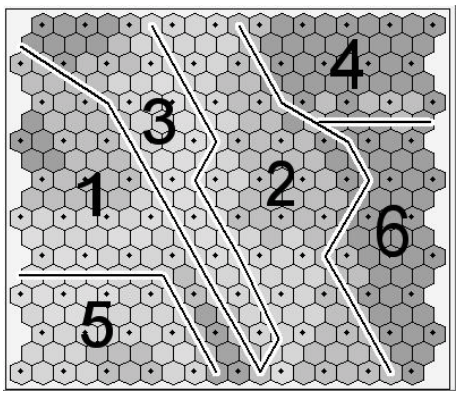


Fig. 6. SOM U-matrix map classified by k-Means clustering. Predefined number of clusters is 6

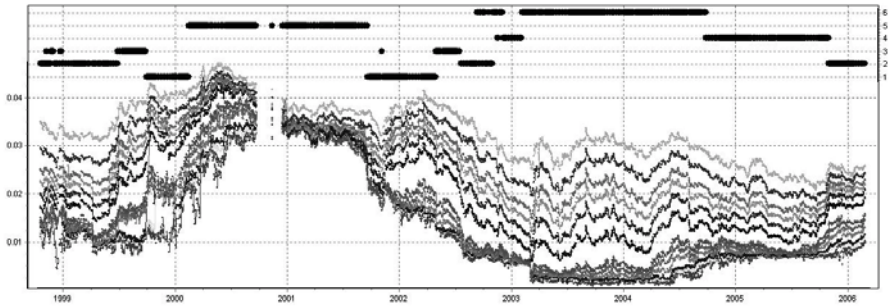


Fig. 7. Classification result for all maturities in time graphs for 6 classes

More detailed structure of the IRC can be elaborated when considering 6 basic classes (Figures 6 and 7). Note that in this case class 1 from 4-classes division was

divided into two (4 and 6) in the 6-classes division. Also additional class 3 (6-classes division) looks like a border between classes 3 and 4 (4-classes division).

3 Missing data reconstruction

One of the possible approaches proposed to solve the problem of missing data deals with using two-dimensional interest rates mapping in a feature space using geostatistical and machine learning algorithms [2]. In this case interest rates are considered as a function in a two-dimensional feature space – time and maturity. Thus, the problem can be formulated as a traditional problem of spatial predictions (interpolation), i.e. predicting values either on a grid or for some particular dates/maturities.

Interpolation problem of missing data deals only with historical data and is related to the tasks of interpolation of missing data in time and to the reconstruction of curves for any maturity. Example of such 2D map produced by Multilayer perceptron having structure [2-50-1] (two inputs, 50 hidden neurons and one output) is presented in Figure 8. MLP was trained using conjugate gradients algorithm with simulated annealing initialization of the weights. Raw data were split into training (80%) and validation (20%) subsets.

Horizontal axis is a maturity (in months), vertical axis is time (in days). Interpolation was produced on a regular grid with x-step 1 month and y-step 1 week. Of course, finer simulation grid can be used as well. Color scale is an interest rate value.

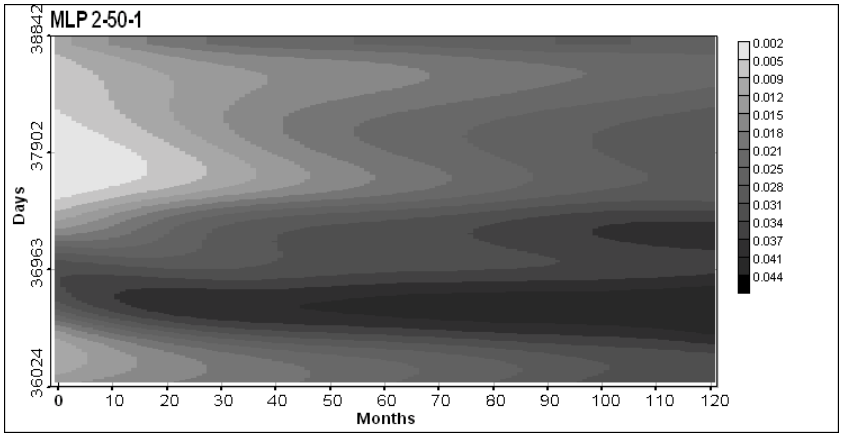


Fig. 8. Interest rates mapping using [2-50-1] multilayer perceptron (MLP) model. Step by horizontal axis is one month; by vertical axis is one week

As it was mentioned earlier, dataset used has one large period with missing values (from September to December of 2000, see Figure 1). Reconstructed time series for some maturities (1, 6 months, 1, 5, and 10 years) are presented in Figure 9. Note that interpolation step in time was one week, so we have smoothed

reconstruction of curves on a daily graph. For the current exploratory empirical study of patterns it was enough. But if it is necessary to produce less smoothed curves with more details one should use smaller step in interpolation grid and more complicated structures of the interpolating model (more hidden layers/neurons in the MLP model; geostatistical simulations, Support Vector Machines) [2].

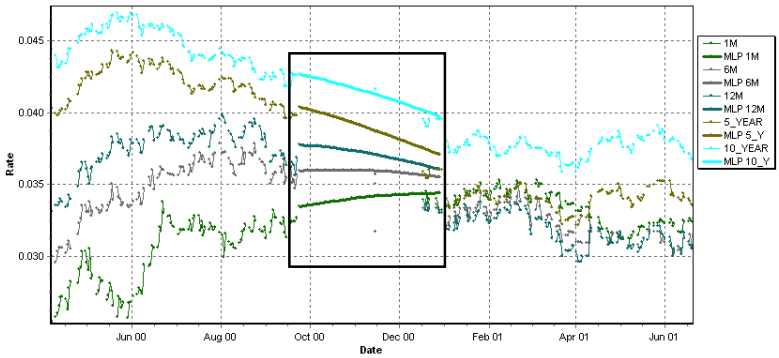


Fig. 9. Interest rates reconstruction of the missing data using 2-50-1 multilayer perceptron model for selected maturities (1, 6 months, 1, 5, and 10 years). Reconstructed period is in the rectangle

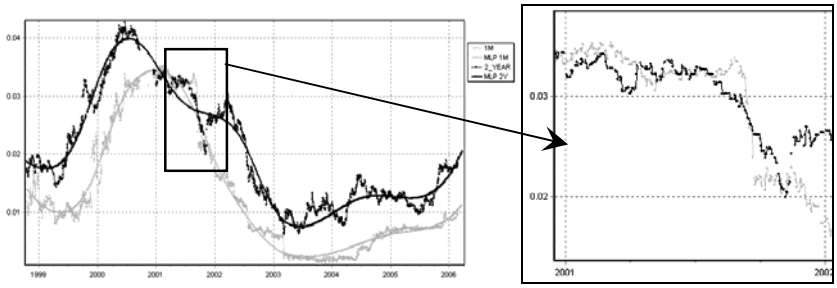


Fig. 10. Examples of reconstructed and real temporal evolution of CHF interest rates in time graphs for one month (gray) and two years (black). 2001 year, period with unusual behaviour (inversion) is outlined and presented with zoom (right).

In Figure 10 examples of reconstructed and real temporal evolution of CHF interest rates for one month (gray) and two years (black) are presented. 2001 year, period with unusual behavior (inversion) is outlined and presented with a zoom (right). Despite of reconstructed curves are rather smooth they were able to describe the phenomena of inversion during the selected period of time.

Now, let us apply SOM model for the analysis of clustering for the curves extracted from this map. These curves can be considered as objects embedded into 120 dimensional space according to the grid (120 cells in x-axis). It is interesting to compare the results with SOM analysis of raw data (original curves).

The result of U-matrix map classified by k-Means method with predefined number of clusters is 4 is given in Figure 11. Figure 12 is the same as Figure 5 but shows clustering results both for original data (13 maturities) and for 120 inputs (maturities) model after mapping (gray dots and black dots). Some differences can be detected only in a year 2001 when unusual behavior of the curve (inversion) was observed.

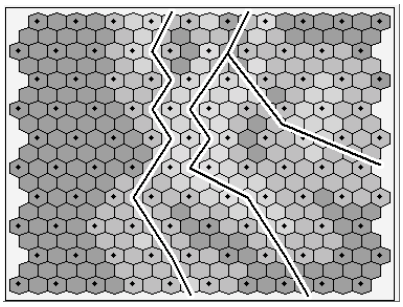


Fig. 11. SOM U-matrix map classified by k-Means clustering method. Model with 120 inputs. Predefined number of clusters is 4

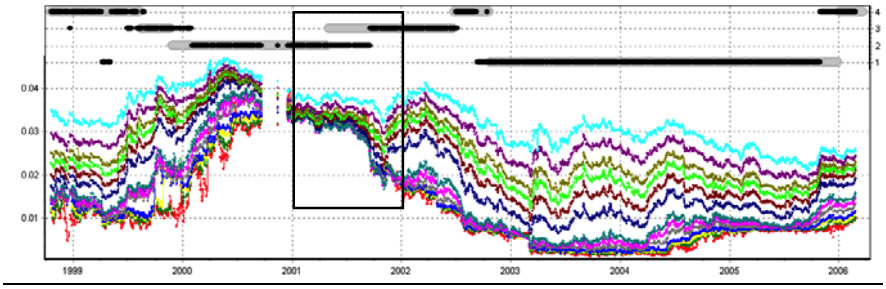


Fig. 12. Classification results for all maturities in time graphs. Black points on the top correspond to clusters (same as in Figure 5), gray dots - clustering result on SOM with 120 inputs after mapping. 2001 year, period with unusual behaviour (inversion) is outlined

4 Conclusions

Interest rate curves form the background for economic and financial decisions and risk management. They are composed of multiple time series corresponding to different maturities. In the present research they were considered as a functional data – curves, evolving in time. Self-Organising Kohonen maps were applied to study interest rate curves patterns and their clustering in time. Interesting finding deals with the observation of few typical behaviors of curves and their clustering in time around low level rates, high level rates, and periods of transition between the two. Such

analysis can help in the prediction of interest rate curves, evaluation of financial instruments and in financial risk management.

In order to solve the problem of missing data an approach based on interest rates mapping was successfully applied.

Future studies dealing with IRC classification and corresponding general market conditions are in progress. New studies concern the analysis of Euro and USD interest rates and their dynamics as well.

References

[1] Kohonen T. *Self-organising maps*. 3d edition. Springer, 2000.

[2] M. Kanevski, M. Maignan, A. Pozdnoukhov, and V. Timonin. *Interest rates mapping*. Physica A 387, Issue 15: 3897-3903, 2008.

[3] M. Kanevski, M. Maignan, A. Pozdnoukhov, and V. Timonin. *Classification of interest rate curves using Self-Organising Maps*. arXiv:0709.4401v1 [physics.data-an], 8 p., 2007

[4] Diebold F. and Canlin Li. *Forecasting the term structure of government bond yields*. Journal of Econometrics 130: 337-364, 2006.

[5] Di Matteo T., Aste T, Hyde ST and Ramsden S. *Interest rates hierarchical structure*. Physica A 355:21-33, 2005.

[6] T. Di Matteo, T. Aste and R. N. Mantegna. *An interest rates cluster analysis*. Physica A 339: 181-188, 2004.

Bankruptcy prediction and neural networks: the contribution of variable selection methods

Philippe du Jardin

Edhec Business School – Information Technology Department
393, Promenade des Anglais – BP 3116 – 06202 Nice Cedex 3 – France

Abstract. Of the methods used to build bankruptcy prediction models in the last twenty years, neural networks are among the most challenging. Despite the characteristics of neural networks, most of the research done until now has not taken them into consideration for building financial failure models, nor for selecting the variables to be included in the models. The aim of our research is to establish that to improve the prediction accuracy of the models, variable selection techniques developed specifically for neural networks may well offer a useful alternative to conventional methods.

1 Introduction

The history of bankruptcy prediction models may be divided into two main periods. The first starts with Altman's [1] and Ohlson's [2] models. During this period, which goes from the late 1960's to the late 1980's, research relied largely on discriminant analysis and logistic regression as methods of building the most accurate models. But as much research has since shown, these methods suffer from major drawbacks and the real input-output variables dependency (i.e., the dependency between financial ratios as explanatory variables and the probability of failure) may be neither linear nor logistic; in other words, it has gradually become clear that other methods should be studied and used to create bankruptcy models.[3]

The second period begins in the late 1980's, when many authors, in attempts to overcome the limitations described above, undertook research to assess the ability of non-parametric methods to accurately predict the risk of bankruptcy or the risk of financial failure. It was also during this period that non-linear techniques such as neural networks emerged in this field of research and demonstrated their frequent ability to outperform most existing techniques, whether parametric or not.

But, whatever the method, when the goal of the research is to seek an effective means of improving the accuracy of a prediction, the variables to be included in the models are commonly selected either because they are among those commonly used in the field of financial analysis, such a set being historically validated through univariate statistical tests (most of the time, t of F test–in which case there is no guarantee that this historical reference is sufficient to create the best models), or because selection is the result of automated processes, often optimized for linear methods (and in this case, there is no guarantee that such processes are relevant in any situation). For instance, it is not particularly relevant to use a variance-based criterion or a likelihood-based criterion to select a set of variables for a model with a method to which these parameters are not well suited, especially with non-linear methods. The

subsets which could be estimated in such a way may be under-optimized because the criterion used to assess their legitimacy does not make sense in a non-linear context.

Thus, we have seen the influence of different variable selection processes on the accuracy of a model and studied the fitness of the most widely used methods for designing bankruptcy models and several well known variable selection techniques. The content of the paper is organized as follows. In section 2 we describe the methods traditionally used to identify variables when the aim of a research is to build the most reliable bankruptcy prediction models. In section 3, we describe the methods and sample used in our experiments. Then, in section 4, we present and discuss the empirical results, and, in section 5, we summarize the main findings of the study.

2 Literature review

A reading of the major articles published over the past 50 years shows that, when developing business failure models, researchers usually use a two-step procedure to choose the « best » variables to be included in their models. Whereas a large set of variables is first identified based on general considerations (financial, empirical, and so on), only a few are finally chosen based on statistical issue.

The first group, often made up of a few tens of variables, is most often identified without using any automatic process but is arbitrarily chosen based on the popularity of variables in literature or on their predictive ability as assessed in previous studies. This « historical » set was built up on the strength of the seminal work done by researchers who, in the 1930's, first assessed the usefulness of financial ratios as a means of predicting corporate failure and by those who contributed to an understanding of the role played by multivariate statistical methods in the field of bankruptcy prediction. Among these latter researchers are Altman [1], Odom and Sharda [4], Zmijewski [5] and Zavgren [6]. All of this work may be viewed as the initial step towards the elaboration of a comprehensive set of essential bankruptcy predictors, which has been complemented over the years by other variables, whether they are accounting-based measures of the financial health of a firm or not (statistical variables calculated with financial data, variables measuring the evolution of financial indicators, non-financial variables describing a quantitative or qualitative characteristic of a company, market variables as a means to explain and quantify the way financial markets may evaluate the performance of companies through the price or the return they place on firm equity).

The second group, on the other hand, is most often selected through a computer-based procedure designed to mine the former group for the best set of variables, depending on an evaluation criterion to define *a priori*.

The evaluation criterion is very often a criterion that does not depend on the method used to develop models. This independence means that the inductive algorithm is not used to assess the value of a set of variables. However, whatever the criterion considered, it may not be without some influence on this algorithm. Indeed, a probabilistic distance, such as a Mahalanobis distance, or a distance calculated through a transformation of an intra- or inter-group covariance matrix, such as a Wilks Lambda, may be considered the criterion most suited to select variables to be used with a discriminant analysis, while a likelihood criterion may be most suited to select variables for a logistic regression. Nevertheless, the use of these criteria with methods for which they

are not entirely optimized or suited is common practice. Indeed, many authors use such criteria to build neural network models [7, 8, 9, 10, 11, 12]. But Leray and Gallinari [13] have stated that since many parametric variable selection methods rely on the hypothesis that input-output variable dependence is linear or that input variable redundancy is well measured by the linear correlation of these variables, such methods are clearly ill-suited to non-linear methods, and hence to neural networks.

Moreover, many of those who have developed neural models have identified their final sets of variables simply on account of their popularity in the financial literature. If one analyzes the linking these studies, it is clear that the criteria used to assess the legitimacy of most of these variables make sense only in a linear context [4, 14, 15, 16, 17, 18, 19, 20, 21, 22]. Very little research has used either a genetic algorithm [23, 24, 25, 26, 27] or a method suitable for non-linear techniques to take into account the characteristics of neural networks, and in each case, with only few variables, small samples, and without attempting comparisons of several methods or criteria [28, 29, 30, 31]; the significance of these experiments is thus reduced. Many authors also strongly recommend comparing the results obtained with different classification or regression techniques, but do not apply the same reasoning to the selection methods that will choose the variables relied on by these techniques.

The point is to show that many authors of bankruptcy models use variable selection methods without considering the very characteristics of the modelling techniques. It is for this reason that the aim of our research is to use « modelling method-variable selection technique » pair analysis to examine the influence of this common practice and to analyze the influence of the latter on the former, in terms of prediction accuracy. Only one study [23] has compared a pair of sets of variables optimized for a discriminant analysis (stepwise method and F test), a logistic regression (stepwise method, Rao's score test to add variables and a likelihood ratio test to discard variables) and a neural network (genetic algorithm), but just to analyze the differences between the models in terms of accuracy over different prediction timeframes (one, two or three years).

3 Methods and samples

3.1 Modelling techniques

Modelling methods are chosen for their popularity in the financial literature. Of the more than 50 regression or discriminant techniques, three predominate: discriminant analysis, logistic regression and a special type of neural network, known as multilayer perceptron, trained with a steepest descent method. A ten-cross validation technique is used to define all neural network parameters (topology, learning rate, momentum term, weight decay) and those that lead to the best out-of-sample error are then selected for our experiments. The final architecture is then composed of a single hidden layer with four nodes, an output node, a bias node in each layer and two weight decay parameters (one for the hidden layer and one for the output); a hyperbolic tangent is used as an activation function. We set the learning rate to 0.4, the momentum term to 0.4, and the weight decay to 10^{-4} and 10^{-3}. The learning process was stopped after 1,000 iterations, as no change could be observed in the error rate.

3.2 **Variable selection procedures**

The variable selection techniques we choose are those most commonly used in the literature. First, we have chosen a technique that relies on a forward search procedure to explore a (sub)space of possible variable combinations, a Fisher F test to interrupt the search, and a Wilks Lambda to compare variable subsets and determine the « best » one. This technique was complemented with two others: a forward stepwise search and a backward stepwise search, with a likelihood statistic as an evaluation criterion of the solutions and a Khi^2 as a stopping criterion. We then select three of the most commonly used [13] methods especially designed for neural networks, two of them evaluating the variables without using the inductive algorithm (filter methods) and one using the algorithm as an evaluation function (wrapper method). The first is a zero-order technique, which uses the evaluation criteria designed by Yacoub and Bennani [32] and the second is a first-order method that uses the first derivatives of network parameters with respect to variables as an evaluation criterion. The last one relies on the evaluation of an out-of-sample error calculated with the neural network. We do not choose a second-order method, based on second derivatives of network parameters, so as to investigate an equivalent number of points of comparison. With all these criteria, we use only a backward search procedure, rather than a forward or a sequential search, and the network is retrained after each variable removal. The zero and first order criterion were calculated as follows. With a network composed of n inputs, one hidden layer with h neurons and one output, where w_{ji} is the weight between input i and neuron j in the hidden layer, and w_j the weight between neuron j and the output, the relevance or the saliency S of a variable i may be defined as:

$$S_i = \sum_{j=1}^{h} \left(\frac{|w_{ji}|}{\sum_{k=1}^{n} |w_{jk}|} \frac{|w_j|}{\sum_{j'=1}^{h} |w_{j'}|} \right) \quad (1)$$

$$S_i = \frac{1}{N} \sum_{j=1}^{N} \left| \frac{\partial y_i}{\partial x_{ji}} \right| \quad (2)$$

where x_i is a variable, y the output of the network calculated with only one neuron and N the sample size.

3.3 **Variable selection procedure**

To select variables, 1,000 random bootstrap samples were drawn from the original dataset. Each bootstrap sample involved selection. To identify important variables, those that were included in more than 70% of the selection results are included in the final models. To avoid discarding potentially relevant but highly correlated variables, variable pairs in which one or both variables are included in more than 90% of the bootstrap selections are considered pairs containing a relevant variable. Then, for each identified pair, the variable that occurs in most of the selection results is ultimately chosen. Once these selections are done, the entire process is repeated to choose the final subsets.

3.4 Model development

We used the following procedure to develop the models. The sample was randomly divided into two sub-samples: a learning sample A of 450 companies and a test sample T of fifty companies. 25 bootstrap samples are drawn from A and, for each selected set of variables, used to estimate as many models as bootstrap samples. Finally, the resulting models are used to classify the observations of sample T thanks to a majority voting scheme. These steps were repeated 100 times and the out-of-sample error is first estimated, along with a test sample, and then re-estimated using the 25 x 100 models, along with a validation sample of 520 companies.

3.5 Samples

The datasets (learning, test and validation sets) are drawn from a French database, Diane, which provides financial data on more than 2 million French companies. The learning and test samples consist of 250 bankrupt and 250 non-bankrupt retail firms which have assets of less than 750,000 €. Annual reports from 2002 are taken from this database to calculate a set of financial ratios, and we add one variable (shareholders' funds) from 2001. The validation set consists of companies belonging to the same sector and the same asset size category (260 bankrupt and 260 non-bankrupt firms), but the data are from 2003, with one variable (shareholders' funds) from 2002.

3.6 Variables

We have selected a set of 41 initial financial ratios that can be broken up into six categories that best describe company financial profiles: liquidity, solvency, financial structure, profitability, efficiency and turnover.

4 Results

4.1 Selected variables and individual discrimination power

Table 1 ranks the variables by frequency of appearance in the six sets of variables, and table 2 shows the same ranking but only for variables that are identified with the criteria optimized for a neural network. This ranking is compared in table 3, where the variables are ranked by their discrimination ability, as assessed by an F test. In this table, we have added their rank as it appears in the previous table. The first half of table 3 (line 1 to line 21) shows the variables for which the F test reveals the highest discrimination power. This part of the table also contains 13 of the 14 variables selected with the neural network. This result indicates that there is a relationship between a parametric measure of discrimination and all the others we used in this study and which are non-parametric. However, this relationship is fairly rough because the two rankings are quite different. For instance, as table 3 shows, the six variables that are most frequently selected with a neural network (EBITDA/total assets, change in equity position, shareholders' funds/total assets, (cash + marketable securities)/total assets, EBIT/total assets, and cash/current liabilities) are ranked 4th, 20th, 12th, 3rd and 13th respectively. By contrast, variables with high discrimination ability, such as EBITDA/total sales, cash/total assets, current liabilities/total assets, or cash/total debt, are not selected with any selection techniques.

	Number of selections	Rank of appearance in the 6 models					
EBITDA/Total Assets	6	4	4	5	5	6	6
Shareholder's Funds /Total Assets	5	1	1	2	3	7	
Change in Equity Position	5	1	3	3	4	7	
(Cash + Marketable Securities)/Total Assets	4	2	4	4	7		
EBIT/Total Assets	3	2	4	5			
Total Debt/Shareholders' Funds	2	1	2				
Cash/Total Debt	2	3	3				
Cash/Current Liabilities	2	3	5				
EBIT/Total Sales	2	5	6				
Cash/Total Sales	2	7	8				
Net Income/Total Assets	1	1					
Cash/Total Assets	1	1					
Current Assets/Current Liabilities	1	2					
Profit before Tax/Shareholders' Funds	1	2					
(Cash + Marketable Securities)/Total Sales	1	5					
Operating Cash Flow/Total Sales	1	6					
Total Liabilities/Total Assets	1	6					
Accounts Receivable/Total Sales	1	8					

Table 1: Ranking of the variables

Rank		Number of selections
1	EBITDA/Total Assets	3
1	Change in Equity Position	3
3	Shareholder's Funds/Total Assets	2
3	(Cash + Marketable Securities)/Total Assets	2
3	EBIT/Total Assets	2
3	Cash/Current Liabilities	2
7	Current Assets/Current Liabilities	1
7	Accounts Receivable/Total Sales	1
7	Operating Cash Flow/Total Sales	1
7	EBIT/Total Sales	1
7	Net Income/Total Assets	1
7	Cash/Total Sales	1
7	Total Debt/Shareholders' Funds	1
7	Total Debt/Total Assets	1

Table 2: Ranking of the variables selected with a neural network

As a consequence, it appears that using a t or an F test for a selection or pre-selection of the inputs of a neural network is unreliable, as these tests may lead to the choice of useless variables as well as to the removal of variables of great interest.

Such might well have been the case here, with the change in equity position, for which the F test is quite low, even though this variable is in fact relevant according to the neural network. Indeed, selection with a Wilks Lambda removes this variable. But when the value of an F test falls below a certain level, the only other variable selected is accounts receivable/total sales, which is selected only once.

		F	p-val.	Rank[1]
1	EBIT/Total Sales	220,15	0,000	7
2	EBITDA/Total Sales	219,49	0,000	
3	EBIT/Total Assets	218,96	0,000	3
4	EBITDA/Total Assets	213,91	0,000	1
5	Net Income/Total Assets	210,01	0,000	7
6	Shareholder's Funds/Total Assets	207,59	0,000	3
7	Total Debt/Total Assets	202,20	0,000	7
8	Total Debt/Shareholders' Funds	201,14	0,000	7
9	Cash/Total Assets	195,01	0,000	
10	Cash/Total Sales	179,60	0,000	7
11	Current Liabilities/Total Assets	179,32	0,000	
12	(Cash + Marketable Securities)/Total Assets	171,62	0,000	3
13	Cash/Current Liabilities	168,19	0,000	3
14	Cash/Total Debt	150,50	0,000	
15	(Cash + Marketable Securities)/Total Sales	145,63	0,000	
16	Current Assets/Current Liabilities	133,77	0,000	7
17	Quick Ratio	131,30	0,000	
18	Accounts Payable/Total Sales	85,95	0,000	
19	Value Added/Total Sales	68,37	0,000	
20	Change in Equity Position	44,29	0,000	1
21	Operating Cash Flow/Total Sales	28,57	0,000	7
22	Net Operating Working Capital/Total Assets	27,21	0,000	
23	Net Operating Working Capital/Total Sales	21,10	0,000	
24	Operating Cash Flow/Total Assets	19,40	0,000	
25	Long Term Debt/Total Assets	19,32	0,000	
26	Inventory/Total Sales	16,00	0,000	
27	Accounts Receivable/Total Sales	13,38	0,000	7
28	Gross Trading Profit/Total Sales	10,53	0,001	
29	Profit before Tax/Shareholders' Funds	8,97	0,003	
30	Quick Assets/Total Assets	7,13	0,008	
31	Current Assets/Total Sales	4,83	0,028	
32	Financial Expenses/Total Sales	4,04	0,045	
33	Quick Assets/Total Assets	3,47	0,063	
34	Change in Other Debts	2,20	0,139	
35	Total Sales/Shareholders' Funds	2,16	0,142	
36	Labor Expenses/Total Sales	0,62	0,431	

37	Net Income/Shareholders' Funds	0,20	0,651
38	Financial Debt/Cash Flow	0,18	0,669
39	Long Term Debt/Shareholders' Funds	0,17	0,681
40	EBITDA/Permanent Assets	0,11	0,743
41	Total Sales/Total Assets	0,02	0,878

[1] Rank of the variables in table 2

Table 3: Rank of the variables according to an F test

4.2 **Model Accuracy**

Several techniques are used to assess the prediction accuracy of the models. To define several points of comparison, we have first analyzed to what extent the two groups (i.e., bankrupt vs. non-bankrupt) could be discriminated using variables drawn at random. For each bootstrap sample, we have evaluated the accuracy of discriminant analysis models, logistic regression models and neural network models. This is a powerful way of measuring the distance between a hazard and a deterministic process, and estimating the economy of the latter. Indeed, if the discrepancy is small, we can expect that this process is useless, and the more it increases, the higher its added value. We then calculate the accuracy of models built with the 41 initial variables. This measure can be used to evaluate the performance of pruning strategies, and hence to analyze the relationship between a dimensionality reduction process and model accuracy. In a third and last step, we calculate the performance of the models built with the six final sets of variables and the three selected classification techniques: discriminant analysis, regression analysis and neural network. The aim of this final step is to discover the way modelling techniques may be influenced by a selection procedure and to identify points of compatibility. For instance, is there any difference between two neural models, one built with variables selected by a Wilks Lambda criterion, and the other by a zero or first-order criterion? And what about a logistic model compared to a discriminant model using the same set of variables?

4.2.1 Model accuracy with variables drawn at random

To assess to what extent our samples can be discriminated, we have drawn 50 sets of variables at random and calculated the correct classification rates with bootstrap samples. Table 4 shows the overall results.

	DA	LR	NN
Non-bankrupt	83.22%	82.39%	83.99%
Bankrupt	73.62%	76.88%	78.45%
Total	78.42%	79.64%	81.22%

DA: Discriminant analysis – LR: Logistic regression – NN: Neural network

Table 4: Model accuracy with variables drawn at random

These results demonstrate that it is not easy to discriminate the groups, since the correct classification rate is roughly equal to 80%. However, this rate is not bad if we take into account the fact that the variables are drawn at random, which reveals that

the initial 41 predictors demonstrate a good discriminatory ability when applied to our samples.

4.2.2 Model accuracy with all variables

Is there any gap in terms of accuracy between a set of randomly selected variables and a set including all variables? The results are shown in table 5. When all variables are taken into consideration, the correct classification rate increases slightly, but the main drawback of this model is its great complexity. In tables 4 and 5, the neural network offers better results than the two other methods.

	DA	LR	NN
Non-bankrupt	93.56%	91.18%	93.60%
Bankrupt	77.72%	81.76%	86.94%
Total	85.64%	86.47%	90.27%

DA: Discriminant analysis – LR: Logistic regression – NN: Neural network

Table 5: Model accuracy with all variables calculated on test samples

4.2.3 Model accuracy as shown by pairs "modelling method–selection technique"

We then analyze the relationship between modelling techniques and variable selection methods. The aim is to investigate whether there are any pairs that perform better than others and to study especially the behaviour of a neural network while using sets of variables that were optimized for other methods.

We first measure the accuracy of different combinations « modelling method–selection technique », but only for those for which the evaluation criterion suits the classification technique. We have compared the results of the following six pairs of methods: discriminant analysis–Wilks Lambda, logistic regression–likelihood criterion (with two search procedures), and neural network–zero-order, first-order, and error criteria. As tables 6 and 7 show, the neural network outperforms discriminant analysis and to a lesser extent logistic regression. Indeed, the best result – 93.85% – is achieved with a neural network on the validation samples, followed by that for logistic regression with 90.77% and discriminant analysis with 85.19%.

	DA Wilks Step.	LR Lik. B Step.	LR Lik. F Step.	NN Error B	NN 0 Order B	RN NN 1st Order B
Non-bankrupt	91.20%	93.60%	89.56%	92.78%	91.96%	92.82%
Bankrupt	83.20%	90.42%	88.84%	95.28%	95.22%	92.82%
Total	87.20%	92.01%	89.20%	94.03%	93.59%	92.82%

DA: Discriminant analysis – LR: Logistic regression – NN: Neural network
Lik.: Likelihood – B: Backward – F: Forward – Step.: Stepwise

Table 6: Model accuracy for different pairs « modelling technique–variable selection method » calculated on test samples

	DA Wilks Step.	LR Lik. B Step.	LR Lik. F Step.	NN Error B	NN 0 Order B	RN NN 1st Order B
Non-bankrupt	89.62%	91.15%	88.85%	93.08%	92.69%	91.15%
Bankrupt	80.77%	90.38%	88.46%	94.62%	91.92%	88.85%
Total	85.19%	90.77%	88.65%	93.85%	92.31%	90.00%

DA: Discriminant analysis – LR: Logistic regression – NN: Neural network
Lik.: Likelihood – B: Backward – F: Forward – Step.: Stepwise

Table 7: Model accuracy for different pairs « modelling technique–variable selection method » calculated on validation samples

We then analyze the results obtained when a modelling technique is used with a selection procedure for which the fit is not deemed acceptable. Table 8 displays the results obtained with the set of variables selected with a Wilks Lambda and those selected with a likelihood criterion, and table 9 gives the results calculated with the three sets of variables optimized for a neural network.

Table 8 shows that a variable selection process based on a variance criterion (i.e., Wilks Lambda) leads to bad results; the adequate classification rate of 87.20% achieved with discriminant analysis is slightly lower with the two other methods. The criterion used here relies on assumptions that dovetail with those on which discriminant analysis is founded. It is little wonder then that variables that cannot satisfactorily classify a high percentage of firms with discriminant analysis are unable to provide good results with other methods; the models built with logistic regression and the neural network produce nearly equal results. Therefore, this criterion is clearly ill-suited to non-linear techniques.

	Wilks Lambda Stepwise			Likelihood Backward Stepwise			Likelihood Forward Stepwise		
	AD	RL	RN	AD	RL	RN	AD	RL	RN
Non-bankrupt	91.20%	88.06%	90.02%	87.28%	93.60%	89.68%	87.98%	89.56%	88.08%
Bankrupt	83.20%	79.18%	77.20%	84.84%	90.42%	92.74%	82.42%	88.84%	91.14%
Total	87.20%	83.62%	83.61%	86.06%	92.01%	91.21%	85.20%	89.20%	89.61%

DA : Discriminant analysis – LR : Logistic regression – NN : Neural network

Table 8: Model accuracy according to modelling techniques and two variable selection criteria (Wilks Lambda–Likelihood) calculated on test samples

The sets of variables that are selected with a likelihood criterion lead to less accurate results with discriminant analysis than with logistic regression – 86.06% – as opposed 92.01% with a backward search, and 85.20% as opposed to 89.20% with a forward search. However, with a neural network, the results of these two sets are fairly good – 91.21% and 89.61% – similar to the results obtained with logistic regression. As it happens, the network leads to better results in one case out of two. With the likelihood criterion, logistic and neural models lead to broadly similar

results, but this is no longer the case with neural network-based criteria. The error criterion achieved an accuracy of 94.03% compared with 90.00% for logistic regression, and only 84.39% for discriminant analysis. The discrepancy between the results of the three methods is nearly the same with a zero-order criterion, with respective figures for correct classification of 93.59%, 88.01% and 83.60%, but with a first-order criterion there is a decrease, with figures of 92.82 %, 89.19 % and 84.45 %.

	Error Backward			0 Order Backward			1st Order Backward		
	AD	RL	RN	AD	RL	RN	AD	RL	RN
Non-bankrupt	83.38%	90.44%	92.78%	83.20%	86.38%	91.96%	87.06%	88.16%	92.82%
Bankrupt	85.38%	89.56%	95.28%	84.00%	89.64%	95.22%	81.84%	90.22%	92.82%
Total	84.28%	90.00%	94.03%	83.60%	88.01%	93.59%	84.45%	89.19%	92.82%

DA: Discriminant analysis – LR: Logistic regression – NN: Neural network

Table 9: Model accuracy according to modelling techniques and three variable selection criteria (Error, Zero and First-Order) calculated on test samples

Therefore, the neural network leads to far better results than other methods, especially with an error criterion, which is not really surprising, since this criterion is both the evaluation criterion of the variable relevance and the measure of this relevance. This is a very characteristic feature of wrappers, because the inductive algorithm is used directly during variable selection. This result is then consistent with what we might expect. The zero-order criterion's outperformance of a first-order criterion can be put down primarily to chance, as there is no evidence that the former is better than the latter.

Neural models, when developed with appropriate variables, are thus much more reliable than logistic or discriminant models. Nevertheless, logistic models seem to better fit the data than discriminant models, whatever the variables used. In addition, with an error criterion, a logistic model produces 90.00% accuracy, whereas the neural model achieves 94.03%, leaving the logistic model – at 84.38% – in the dust.
The accuracy of a model is in part the result of the intrinsic characteristics of the modelling technique and in part that of the fit between this technique and the variable selection procedure involved in its design. In the field of bankruptcy prediction, all the experiments that have been down with large samples show that both financial ratios and a probability of bankruptcy behave in a non-linear manner. It is precisely for this reason that, as long as this non-linearity cannot be taken into account, it is hardly possible to develop accurate models. Although using a selection criterion that fits logistic regression to design a neural model may be relevant, the choice of a criterion that fits discriminant analysis for the same purpose should not be recommended. It is necessary, at the very least, to consider other solutions.

5 Conclusion

We have demonstrated that a neural network-based model for predicting bankruptcy performs significantly better when designed with appropriate variable selection techniques rather than other types, and particularly those commonly used in the financial literature. Unlike the former, the latter are fast and easy to use, which may account for their under-use. However, a few studies have looked into other techniques, mainly genetic algorithms. So the reasons for the failure of neural network-based variable selection methods to be adopted more widely must be found elsewhere, perhaps in the absence of cross-disciplinary approaches to this particular field. Neural network algorithms are in exactly the same situation: while many types are commonly used in many scientific disciplines, only one is systematically used in the field of corporate finance. And variable selection techniques face the same issue: they come from a field of knowledge that has little to do with corporate finance. Of course, all these results should be confirmed by additional studies in a variety of other settings, such as other samples, types of firms, sectors, and so on, but they point to the need to use relevant variable selection techniques to develop neural models. As it happens, the most recent research papers continue to rely on traditional methods: variables are still selected because they were selected in earlier [33] or as a result of their popularity in the field of financial analysis [34].

We have also demonstrated that there is a relationship between the discrimination ability of a variable, as measured with a t test or an F test, and its ability to be selected by an automatic procedure that relies on other measures, but we have also found a discrepancy in this relationship, which indicates that such statistical tests should not be used alone if the purpose of the selection is to create a neural model.

As a consequence, we may use them – but with extreme caution – to build non-linear models, and if we intend to do so, we would do well to use them in conjunction with other techniques.

References

[1] E. I. Altman, Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy, *Journal of Finance*, 23:589-609, 1968.

[2] J. A. Ohlson, Financial Ratios and the Probabilistic Prediction of Bankruptcy, *Journal of Accounting Research*, 18:109-131, 1980.

[3] E. K. Laitinen and T. Laitinen, Bankruptcy Prediction: Application of the Taylor's Expansion in Logistic Regression, *International Review of Financial Analysis*, 9:327-349, 2000.

[4] M. C. Odom and R. Sharda, A Neural Network Model for Bankruptcy Prediction, *proceedings of the IEEE International Joint Conference on Neural Networks* (IJCNN 2001), pages 163-168, San Diego (California), 1990.

[5] M. E. Zmijewski, Methodological Issues Related to the Estimation of Financial Distress Prediction Models, *Journal of Accounting Research*, 22:59-82, 1984.

[6] C. V. Zavgren, Assessing the Vulnerability to Failure of American Industrial Firms: A Logistic Analysis, *Journal of Business Finance and Accounting*, 12:19-45, 1985.

[7] R. L. Wilson and R. Sharda, Bankruptcy Prediction Using Neural Networks, *Decision Support System*, 11:545-557, 1994.

[8] K. C. Lee, I. Han and Y. Kwon, Hybrid Neural Network Models for Bankruptcy Predictions, *Decision Support Systems*, 18:63-72, 1996.

[9] N. Kumar, R. Krovi and B. Rajagopalan, Financial Decision Support with Hybrid Genetic and Neural Based Modeling Tools, *European Journal of Operational Research*, 103:339-349, 1997.

[10] Z. R. Yang and R. G. Harrison, Analysing Company Performance Using Templates, *Intelligent Data Analysis*, 6:2002.

[11] A. Charitou, E. Neophytou and C. Charalambous, Predicting Corporate Failure: Empirical Evidence for the UK, *European Accounting Review*, 13:465-497, 2004.

[12] P. P. M. Pompe and J. Bilderbeek, The Prediction of Bankruptcy of Small- and Medium-Sized Industrial Firms, *Journal of Business Venturing*, 20:847-868, 2005.

[13] P. Leray and P. Gallinari, Feature Selection with Neural Networks, *Behaviormetrika*, 26:145-166, 1998.

[14] K. Y. Tam and M. Y. Kiang, Managerial Applications of Neural Networks: The Case of Bank Failure Predictions, *Management Science*, 38:926-947, 1992.

[15] N. Wilson, K. S. Chong and M. J. Peel, Neural Network Simulation and the Prediction of Corporate Outcomes: Some Empirical Findings, *International Journal of the Economics of Business*, 2:31-50, 1995.

[16] J. E. Boritz and D. B. Kennedy, Effectiveness of Neural Network Types for Prediction of Business Failure, *Expert Systems with Applications*, 9:503-512, 1995.

[17] R.C. Lacher, P. K. Coats, S. C. Sharma and L. F. Fant, A Neural Network for Classifying the Financial Health of a Firm, *European Journal of Operational Research*, 85:53-65, 1995.

[18] C. Serrano-Cinca, Feedforward Neural Networks in the Classification of Financial Information, *European Journal of Finance*, 3:183-202, 1997.

[19] T. Laitinen and M. Kankaanpaa, Comparative Analysis of Failure Prediction Methods: The Finnish Case, *European Accounting Review*, 8:67-92, 1999.

[20] Z. R. Yang, M. B. Platt and H. D. Platt, Probabilistic Neural Networks in Bankruptcy Prediction, *Journal of Business Research*, 44:67-74, 1999.

[21] P. C. Pendharkar, A Threshold-Varying Artificial Neural Network Approach for Classification and its Application to Bankruptcy Prediction Problem, *Computers and Operations Research*, 32:2561-2582, 2005.

[22] C. H. Wu, G. H. Tzeng, Y. J. Goo and W. C. Fang, A Real-Valued Genetic Algorithm to Optimize the Parameters of Support Vector Machine for Predicting Bankruptcy, *Expert Systems with Applications*, 32:397-408, 2007.

[23] B. Back, T. Laitinen and K. Sere, Choosing Bankruptcy Predictors Using Discriminant Analysis, Logit Analysis and Genetic Algorithms, Technical Report, Turku Centre for Computer Science, 1996.

[24] J. Wallrafen, P. Protzel and H. Popp, Genetically Optimized Neural Network Classifiers for Bankruptcy Prediction – An Empirical Study, *proceedings of the 29th Hawaii International Conference on System Sciences* (HICS 1996), pages 419-426, Maui (Hawaii), 1996.

[25] K. Kiviluoto, Predicting Bankruptcies with the Self-Organizing Map, *Neurocomputing*, 21:191-220, 1998.

[26] R. S. Sexton, R. S. Sriram and H. Etheridge, Improving Decision Effectiveness of Artificial Neural Networks: A Modified Genetic Algorithm Approach, *Decision Sciences*, 34:421-442, 2003.

[27] A. Brabazon and P. B. Keenan, A Hybrid Genetic Model for the Prediction of Corporate Failure, *Computational Management Science*, 1:293-310, 2004.

[28] E. W. Tyree and J. A. Long, Bankruptcy Prediction Models: Probabilistic Neural Networks versus Discriminant Analysis and Backpropagation Neural Networks, Working Paper, Department of Business Computing, City University, 1996.

[29] C. Charalambous, A. Charitou and F. Kaourou, Application of Feature Extractive Algorithm to Bankruptcy Prediction, *proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks* (IIE-IJCNN 2000), pages 303-308, July, Como (Italy), 2000.

[30] T. K. Sen, P. Ghandforoush and C. T. Stivason, Improving Prediction of Neural Networks: A Study of Two Financial Prediction Tasks, *Journal of Applied Mathematics and Decision Sciences*, 8:219-233, 2004.

[31] I. Bose and R. Pal, Predicting the Survival or Failure of Click-and-Mortar Corporations: A Knowledge Discovery Approach, *European Journal of Operational Research*, 174:959-982, 2006.

[32] M. Yacoub and Y Bennani, HVS: A Heuristic for Variable Selection in Multilayer Artificial Neural Network Classifier*, proceedings of the International Conference on Artificial Neural Networks and Intelligent Engineering* (ICANNII 1997), pages 527-532, Saint-Louis (Missouri) 1997.

[33] S. Jones and D. A. Hensher, Predicting Firm Financial Distress: A Mixed Logit Model, *Accounting Review*, 79: 1011-1038, 2007.

[34] Z. Zhu, H. He, J. A. Starzyk and C. Tseng, Self-Organizing Learning Array and its Application to Economic and Financial Problems, *Information Sciences*, 177:1180-1192, 2007.

A methodology for time series prediction in Finance

Qi Yu[1], Antti Sorjamaa[1], Yoan Miche[1],
and Eric Séverin[2]

1- Helsinki University of Technology - Information and Computer Science Department
Konemiehentie 2, Espoo - Finland

2- University of Lille 1 - Laboratoire Economie Management
59653 Villeneuve d'Ascq cedex - France

Abstract. Aims to predict the Return on assets (ROA)of the company for the next year correctly and efficiently, this paper proposes a methodology called OP-KNN, which builds a one hidden-layer feedforward neural network, using nearest neighbors neurons with extremely small computational time. The main strategy is to select the most relevant variables beforehand, then to build the model using KNN kernels. Multiresponse Sparse Regression (MRSR) is used as the third step in order to rank each k^{th} nearest neighbor and finally as a fourth step Leave-One-Out estimation is used to select the number of neighbors and to estimate the generalization performances. This new methodology is tested on a toy example and experiment on 200 French companies to predict ROA value for next year.

1 Introduction

Return on assets (ROA) is an important indicator to explain corporate performance, showing how profitable a company is before leverage, and is frequently compared with companies in the same industry. However, it is not easy to analyse what characters of the companies mainly affect the ROA value, especially when you try to predict it, the problem becomes more risky. Thus, we investigate the mothodology: Optimal Pruned K-Nearest Neighbors (OP-KNN) to realize these tasks in this paper, using neural network with KNN.

As we know, it is usual to have very long computational time for training a feedforward network using existing classic learning algorithms even for simple problems. Thus, Guang-Bin Huang in his paper [1] proposed an original algorithm called Extreme Learning Machine (ELM) for single-hidden layer feedforward neural networks (SLFN) which randomly chooses hidden nodes and analytically determines the output weights of SLFNs. The most significant characteristics of this method is that it tends to provide good generalization performance and a comparatively simple model at extremely high learning speed. But the remaining problem is the selection of the kernel, i.e. the activation function used between input data and the hidden layer. In [8], Optimal Pruned Extreme Learning Machine (OP-ELM) has been proposed as an improvement of the original ELM. However, as explained in [10], it isn't so appropriate for our case. Thus, this paper presents OP-KNN which uses KNN as the kernel and solves the problems properly. This method has several notable achievements:

- keeping good performance while being simpler than most learning algorithms for feedforward neural network,
- using KNN as the deterministic initialization,
- the computational time of OP-KNN being extremely low (lower than OP-ELM or any other algorithm). In our financial experiments, the computational time is less than a second (for a regression problem with 650 samples and 36 variables),
- for our application, Leave-One-Out (LOO) error is used both for variables selection [12] and OP-KNN complexity selection.

In the experimental Section, this paper deals with the explanation and prediction of corporate performance which is measured by ROA. We try to determine if the features of assets, the debt level or the cost structure have an influence on corporate performance. Our results highlight that the industry, the size, the liquidity and the dividend are the main determinants of corporate performance.

The main steps of the OP-KNN methodology are SLFN with KNN, MRSR (for Multiresponse Sparse Regression) [7] and finally the LOO error validation [11], using PRESS statistic [3]. All these steps are detailed in the Section 2. To improve the methodology, a prior Variable Selection is performed to remove irrelevant input variables beforehand [12]. Section 3 shows the results on a toy example and on financial modeling.

2 Optimal Pruned – k-Nearest Neighbors

OP-KNN is similar to OP-ELM, which is a original and efficient way of training a Multilayer Perceptron (MLP) network. The three main steps of the OP-KNN are summarized in Figure 1.

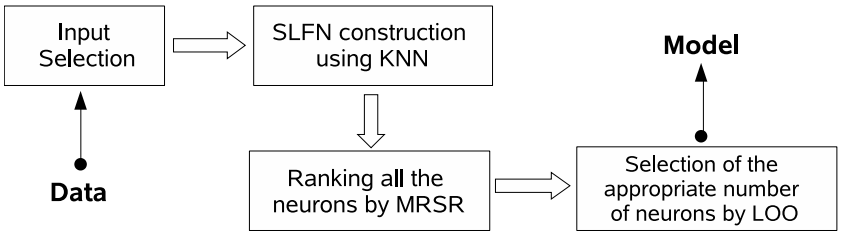


Figure 1: The three steps of the OP-KNN algorithm.

2.1 Input Selection

Obviously, the input variables can have different importance with respect to the output. Therefore, in this section, a methodology which optimizing the Nonparametric Noise Estimation (NNE) provided by Delta Test (DT) is used

for input selection [12]. As a result, the input data are preprocessed and scaled before building the model

Moreover, variable scaling can be seen as a generalization of variable selection; instead of restricting the scalars to attain either values0 or 1, the entire range $[0, 1]$ is allowed. That means we increase the scalar by $1/h$ from 0 to 1, Integer h is a constant grid parameter. In this paper, our experiments choose $h = 1$ to select variables at the beginning and then choose $h = 10$ on the selected variables to give them different scalars from $[0, 0.1, 0.2, ..., 1]$.

Small scalar weight in the result indicate that the variable is more irrelevant and weight zero shows the variable can be pruned. Moreover, the variable dimension could be decreased according to the scaling factors to reduce the complexity of the modeling process.

2.2 Single-hidden Layer Feedforward Neural Networks (SLFN)

The first step of the OP-KNN algorithm is the core of the original ELM: the building of a single-layer feed-forward neural network. The idea of the ELM has been proposed by Guang-Bin Huang *et al.* in [1].

In the context of a single hidden layer perceptron network, let us denote the weights between the hidden layer and the output by $\mathbf{b}$. Activation functions used with the OP-KNN differ from the original SLFN choice since the original sigmoid activation functions of the neurons are replaced by the k-Nearest Neighbors, hence the name OP-KNN. For the output layer, the activation function remains as a linear function.

A theorem proposed in [1] states that the activation functions, output weights $\mathbf{b}$ can be computed from the hidden layer output matrix $\mathbf{H}$: the columns $\mathbf{h_i}$ of $\mathbf{H}$ are the corresponding output of the k-nearest-neighbors. Finally, the output weights $\mathbf{b}$ are computed by $\mathbf{b} = \mathbf{H}^{\dagger}\mathbf{y}$, where $\mathbf{H}^{\dagger}$ stands for the Moore-Penrose inverse [6] and $\mathbf{y} = (y_1, \ldots, y_M)^T$ is the output.

The only remaining parameter in this process is the initial number of neurons N of the hidden layer.

2.3 *k*-Nearest Neighbors

The k-Nearest Neighbors (KNN) model is a very simple, but powerful tool. It has been used in many different applications and particularly in classification tasks. The key idea behind the KNN is that similar training samples have similar output values. In OP-KNN, the approximation of the output is the weighted sum of the outputs of the k-nearest neighbors. The model introduced in the previous section becomes:

$$\hat{y}_i = \sum_{j=1}^{k} b_j y_{P(i,j)} \tag{1}$$

where $\hat{y}_i$ represents the output estimation, $P(i, j)$ is the index number of the j^{th}

nearest neighbor of sample $\mathbf{x}_i$ and b is the results of the Moore-Penrose inverse introduced in the previous Section.

2.4 Multiresponse Sparse Regression (MRSR)

For the removal of the useless neurons of the hidden layer, the Multiresponse Sparse Regression proposed by Timo Similä and Jarkko Tikka in [7] is used. It is an extension of the Least Angle Regression (LARS) algorithm [2] and hence is actually a variable ranking technique, rather than a selection one. The main idea of this algorithm is the following: denote by $\mathbf{T} = [\mathbf{t}_1 \dots \mathbf{t}_p]$ the $n \times p$ matrix of targets, and by $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_m]$ the $n \times m$ regressors matrix. MRSR adds each regressor one by one to the model $\mathbf{Y}^k = \mathbf{X}\mathbf{W}^k$, where $\mathbf{Y}^k = [\mathbf{y}_1^k \dots \mathbf{y}_p^k]$ is the target approximation by the model. The $\mathbf{W}^k$ weight matrix has k nonzero rows at kth step of the MRSR. With each new step a new nonzero row, and a new regressor to the total model, is added.

An important detail shared by the MRSR and the LARS is that the ranking obtained is exact in the case where the problem is linear. In fact, this is the case, since the neural network built in the previous step is linear between the hidden layer and the output. Therefore, the MRSR provides the exact ranking of the neurons for our problem.

Details on the definition of a cumulative correlation between the considered regressor and the current model's residuals and on the determination of the next regressor to be added to the model can be found in the original paper about the MRSR [7].

MRSR is hence used to rank the kernels of the model: the target is the actual output y_i while the "variables" considered by MRSR are the outputs of the k-nearest neighbors.

2.5 Leave-One-Out (LOO)

Since the MRSR only provides a ranking of the kernels, the decision over the actual best number of neurons for the model is taken using a Leave-One-Out method. One problem with the LOO error is that it can get very time consuming if the dataset tends to have a high number of samples. Fortunately, the PRESS (or PREdiction Sum of Squares) statistics provide a direct and exact formula for the calculation of the LOO error for linear models. See [3, 4] for details on this formula and implementations:

$$\epsilon^{\text{PRESS}} = \frac{y_i - \mathbf{h}_i \mathbf{b}}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T}, \tag{2}$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T\mathbf{H})^{-1}$ and $\mathbf{H}$ the hidden layer output matrix defined in subsection 2.2.

The final decision over the appropriate number of neurons for the model can then be taken by evaluating the LOO error versus the number of neurons used (properly ranked by MRSR already).

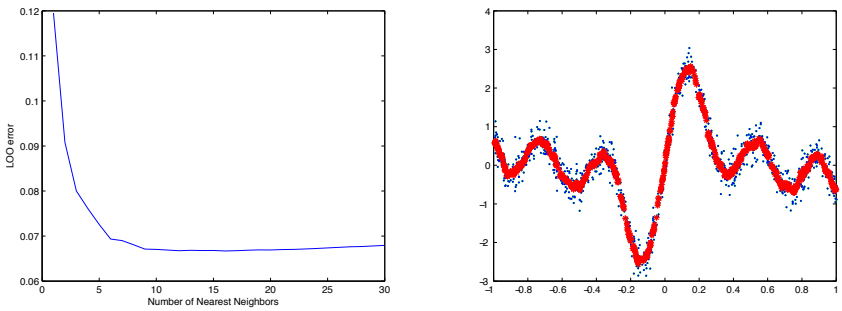
2.6 Discussion on the Advantages of the OP-KNN

In order to have a very fast and still accurate algorithm, each of the four presented steps have a special importance in the whole OP-KNN methodology. Input selection helps to reduce the variables deminsion and the modeling complexity beforehand at the very beginning. The K-nearest neighbor ranking by the MRSR is one of the fastest ranking methods providing the exact best ranking, since the model is linear (for the output layer), when creating the neural network using KNN. Without MRSR, the number of nearest neighbor that minimizes the Leave-One-Out error is not optimal and the Leave-One-Out error curve has several local minima instead of a single global minimum. The linearity also enables the model structure selection step using the Leave-One-Out, which is usually very time-consuming. Thanks to the PRESS statistics formula for the LOO error calculation, the structure selection can be done in a small computational time.

3 Experiments

3.1 Sine in one dimension

In this experiments, a set of 1000 training points are generated (and represented in Fig. 2B), the output is a sum of two sines. This single dimension example is used to test the method without the need for variable selection beforehand. The Fig. 2A shows the LOO error for different number of nearest neighbors and the model built with OP-KNN using the original dataset. This model approximates the dataset accurately, using 18 nearest neighbors; and it reaches a LOO error close to the noise introduced in the dataset which is 0.0625. The computational time for the whole OP-KNN is one second (using Matlab© implementation).



(a) (b)

Figure 2: Sine Toy example

3.2 Financial Modeling

In this experiment, we use the data [12] related to 200 French companies during a period of 5 years. 36 input variables on 535 samples are finanlly used without any missing value, the input variables are financial indicators that are measured every year (for example debt, number of employees, amount of dividends, ...) and the last variable is ROA value of the same year. The target variable is the ROA of the next year for each sample.

Table 1 shows the real meaning in financial field about all the variables we have used.

All these four targets are tested one by one using OP-KNN; Variable Selection ($h = 1$) and Scaling ($h = 10$) are performed as first step [12] for comparison. The results are listed in the following Tables 2 and 3.2 where we can see the LOO error are decreased almost half with variable selection step for each cases. The minimum LOO error appears when using OP-KNN on the scaled selected input variables as expected. Moreover, the final LOO error reach roughly the same stage as the value we estimated while doing variable selection [9]. Thus, for this financial dataset, this methodology not only build the model in a simple and fast way, but also prove the accuracy of our previous selection algorithm [9, 12], and the most important point is the method successfully predict the ROA value for the next year. It should also be noted that on the experiments of this financial data, the OP-KNN with Variable Scaling on the selected variables shows the best efficiency and accuracy, meanwhile it selected the most important variables with their ranking and build the model to predict. The computational time for the whole OP-KNN is one second for each output.

4 Conclusions

In this paper, we proposed a methodology OP-KNN based on SLFN which gives better performance than existing OP-ELM or any other algorithms for the financial modeling we have tested. Using KNN as a kernel, the MRSR algorithm and the PRESS statistic are all required for this method to build an accurate model. Besides, to do the prestep Variable Selection is clearly a wise choice to raise the interpretability of variables and increase the efficiency of the built model.

We test our methodology on 200 French industrial firms listed on Paris Bourse (Euronext nowadays) within a period of 5 years (1991-1995). Our results highlight that the first twelve variables are the best combination to explain corporate performance measured by the ROA of next year. Afterwards, the new variables do not allow improving the explanation of corporate performance. For example, we show that the company size is a variable that improves performance. Moreover, we use these selected variables to build a model for prediction. Furthermore, it is interesting to notice that the discipline of market allows to put pressure on firms to improve corporate performance.

Table 1: The meaning of variables

index	Variable	Meaning
1	Sector	Industry
2	Transaction	Number of shares exchanged during the year
3	Rotation	Security turnover rate
4	Vrif Rotation	Not useful
5	Net dividend	Amount of dividend for one share during the year
6	Effectifs	Number of employees
7	CA	Sales
8	II	Other assets
9	AMORII	Dotations on other assets
10	IC	Property, plant and equipement
11	AMORIC	Dotations on property, plant and equipement
12	IF	Not useful
13	AI	Fixed assets
14	S	Stocks or inventories
15	CCR	Accounts receivables
16	CD	Not useful
17	L	Cash in hands and at banks
18	AC	Total of current assets
19	CPPG	Total of capital of group (in book value)[a]
20	PRC	Not useful
21	FR	Accounts payables
22	DD	Not useful
23	DEFI	Financial debt
24	Debt-1AN	Debt whose maturity is inferior to 1 year
25	Debt+1AN	Debt whose maturity is superior to 1 year
26	TD	Total Debt
27	CPER	Cost of workers
28	CPO	Not useful
29	DA	Dotations on amortizations
30	REXPLOI	Operating income before tax
31	CFI	Interests taxes
32	RFI	Financial income
33	RCAI	Operating income before tax + Financial income
34	REXCEP	Extraordinary item
35	IS	Taxes from State
36	ROA	net income / total assets
Output1	ROA	the value of next year

[a]By construction the total debt is equal to Total assets

Table 2: VS+Scaling: ROA

index	Variable	Scaling value
11	AMORIC	1.0
21	FR	1.0
19	CPPG	0.9
28	CPO	0.8
9	AMORII	0.7
16	CD	0.6
10	IC	0.5
15	CCR	0.5
18	AC	0.4
32	RFI	0.2
14	S	0.1
2	Transaction	0
DT result	0.2811	0.2446

Table 3: Normalized result for output 1

	LOO error	Num of NN selected
all	0.5827	12
VS	0.4845	10
	0.4852	7
VS+Scaling	0.4602	11
	0.4616	8

References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew (2006), Extreme learning machine: Theory and applications, *Neurocomputing*, p. 489-501.

[2] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani (2004), Least angle regression, *Annals of Statistics*, **vol. 32** p. 407-499.

[3] R.H. Myers, (1990), Classical and Modern Regression with Applications, Duxbury, Pacific Grove, CA, USA

[4] G. Bontempi, M. Birattari, H. Bersini (1998), Recursive lazy learning for modeling and control, *European Conference on Machine learning*, p. 292-303.

[5] W.T. Miller, F.H. Glanz, L.G. Kraft (1990), Cmas: An associative neural network alternative to backpropagation, *Proceeding of the IEEE*, **vol. 70**97-102 p. 1561-1567.

[6] C.R. Rao, S.K. Mitra (1972), Generalized Inverse of Matrices and its Applications, *John Wiley & Sons Inc*,

[7] T. Similä, J. Tikka (2005), Multiresponse sparse regression with application to multidimensional scaling, *Proceedings of the 15th International Conference on Artificial Neural Networks*, **Part II** p. 97-102.

[8] Y. Miche, P. Bas, C. Jutten, O. Simula, A. Lendasse (2008), A methodology for building regression models using Extreme Learning Machine: OP-ELM, *European Symposium on Artificial Neural Networks*, p. 23-25.

[9] Q. Yu, E. Séverin, A. Lendasse (2007), Feature Selection Methodology for Financial Modeling, *Computational Methods for Modeling and Learning in Social and Human Science 2007*, Brest, France.

[10] Q. Yu, Antti Sorjamaa, Yoan Miche, E. Séverin, A. Lendasse (2008), Optimal Pruned K-Nearest Neighbors: OP-KNN – Application to Financial Modeling, *Computational Methods for Modeling and Learning in Social and Human Science 2008*, Créteil, France.

[11] A. Lendasse, V. Wertz, M. Verleysen (2003), Model Selection with Cross-Validations and Bootstraps - Application to Time Series Prediction with RBFN Models, *Joint International Conference on Artificial Neural Networks*, Istanbul, Turkey.

[12] Q. Yu, E. Séverin, A. Lendasse (2007), Variable Selection for Financial Modeling, *Computing in Economics and Finance*, Montréal, Canada.