UNIVERSITÀ
DEGLI STUDI
FIRENZE

PHD PROGRAM IN SMART COMPUTING
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)

# Optimization and Machine Learning in support of Statistical Modeling

**Leonardo Di Gangi**

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

*PhD Program in Smart Computing*
*University of Florence, University of Pisa, University of Siena*

# Optimization and Machine Learning in support of Statistical Modeling

**Leonardo Di Gangi**

**Advisor:**

Prof. Fabio Schoen

**Head of the PhD Program:**

Prof. Paolo Frasconi

**Evaluation Committee:**
Prof. Bernardetta Addis, *Université de Lorraine*
Dr. Giampaolo Liuzzi, *Sapienza Universitá di Roma*

XXXIV ciclo — October 2022

*To my family and Azzurra*

# Acknowledgments

It seems yesterday that this adventure began and instead three long years passed. We even experienced the spread of Covid 19 pandemic which generated deaths and disorientation all over the world. Doing a PhD in such complicated situation was a great privilege, making me think even more about the significance of science and research.

I really have so many people to thank and I apologize if I forget someone. First, I would like to express my thanks to Prof. Fabio Schoen and Prof. Marco Sciandrone for having me helped and supported in this experience. Every talk with them has always given me enrichment and improvement. I also want to emphasize their great human depth that rendered the Global Optimization Lab (GOL) a pretty place to live. Many thanks also to Prof. Anna Gottard and Prof. Francesco Rinaldi, as supervisors, for their support and advice. A big thank you to Prof. Giorgio Calzolari for his precious suggestions.

And then there are all my friends and colleagues of GOL, past and present: Alessandro G., Guido C., Tommaso L. (Toh), Leonardo G., Giulio G., Luca T. (Doctor), Matteo L., Alessio S., Enrico C., Simone M., Pierluigi M., Tommaso A.. It was a great fortune to share GOL with you.

Tommaso L. (Toh), Luca T. (Doctor), Enrico C., we spent great times together: it was truly a privilege to know you. Tommaso L., I owe you dinner (I remember!). Guido C. you are like a brother for me. Alessio S. you are a great friend: you've helped me so many times and I'm grateful to you. Matteo L. you are the smartest person I've ever met, totally enthusiastic to do research and at the same time a leader, a very dear friend with great human qualities: thanking you is not enough.

A great thank to my lifelong friends Gian Marco F., Francesco B., Andrea B., Roberto S., Simone L. and to my past academic friends Marco L., Alessandro P., Marco M., Andrea B., Jacopo P..

Finally, I wish to address a special thought to my family and Azzurra. It would have been impossible to reach this goal without you. You have been of great help in these years in many different ways. For this reason this manuscript is dedicated to you.

**Abstract**

A crucial step in data analysis is to formulate the most appropriate model for reliable inference or prediction. Both optimization and machine learning assist the modeler towards this task.

Whenever the inference is the focus, a linear regression model represents a suitable tool for an initial understanding of the reality that the model aims to describe. An automatic and objective procedure to select the predictors of the regression model is fundamental to achieve this target. On this matter, as a first contribution, we propose an algorithm, based on Mixed Integer Optimization (MIO), for best subset selection problem in Gaussian linear regression scenario. The algorithm, with simple modifications, is also suitable for the order selection problem in Gaussian ARMA models. The proposed approach has the advantage of considering both model selection as well as parameter estimation as a single optimization problem. The core of the algorithm is based on a two-step Gauss-Seidel decomposition scheme which favors the computational efficiency of the procedure. The performed experiments show that the algorithm is fast and reliable although not guaranteed to deliver the optimal solution.

As a second contribution, we consider the maximum likelihood estimation problem of causal and invertible Gaussian ARMA models of a given order $(p, q)$. We highlight the convenience of fitting these models directly in the space of partial autocorrelations (autoregressive component) and in the space of partial moving average coefficients (moving average component) without having to exploit the additional Jones reparametrization. In our method, causality and invertibility constraints are handled by formulating the estimation problem as a bound constrained optimization problem. Our approach is compared to the classical estimation method based on the Jones reparametrization which leads to an unconstrained formulation of the problem. The convenience of our approach is assessed by the results of several computational experiments which reveal a significant reduction of fitting times and an improvement in terms of numerical stability. We also propose a regularization term in the model and we show how this addition improves the out of sample quality of the fitted model.

As a final contribution, the problem of forecasting univariate temporal data is considered. When the purpose of the model is prediction, combining forecasting models is a well known successful strategy leading to an improvement of the accuracy of prediction. Usually, knowledge of experts is needed to combine forecasting models in an appropriate way. However, especially in real-time applications, the need of automatic procedures, which replace the knowledge of experts, is evident. By learning from past forecasting episodes, a meta learning model can be properly trained to learn the combination task. On this matter, we introduce two meta-learning systems which recommend a weighting schema for the combination of forecasting models based on time series features. We focus on sparse convex combinations. Zero weighted forecasting models do not

contribute to the computation of the final forecast and their fit can be avoided. Therefore, the more the degree of sparsity increases, the more the computational time for producing final forecasts decreases. The methodology is tested on the M4 competition dataset. Obtained results highlight that it is possible to reduce significantly the number of models in the combination without affecting the quality of prediction.

# Contents

# Chapter 1

# Introduction

Optimization plays an important role in many branches of science but one of its main applications is statistical model fitting. Traditionally, the model selection phase is separated from the estimation phase: the modeler, thanks to the knowledge of context, detects a set of candidate models and, after their estimation, chooses the best one. Anyway, the estimation of a single candidate model requires caution. The associated optimization problem can hide pitfalls (e.g., non convexities and numerical instabilities).
More recently, research focused on the development of advanced optimization algorithms able to perform selection and fitting simultaneously. The employment of these algorithms makes the model building process as objective as possible and remedies the lack of context knowledge by the modeler who is not able to formulate a model.
Even machine learning can be supportive to the modeler in automating the model building process: given the experience of multiple learning episodes, a trained machine learning system can identify and recommend models or their combinations that achieve potentially best performance. More precisely, this subfield of machine learning is known as meta-learning.

This dissertation deals with three relevant statistical problems related to both optimization and machine learning. It is shown as optimization and machine learning can really assist the model building process (see Figure 1.1) as long as the knowledge and task of the modeler, necessary to select and fit a final model [1], vary.

Indeed, three different scenarios are considered:

- The modeler does not know the model structure. The task of the modeler is both to detect the subset of relevant predictors to formulate the model as well

[1] Following (Ding et al., 2018), the term *model* denotes a class of hypothetical probability distributions, formulated in order to approximate the Data Generator Process (DGP), which is the unknown but true distribution.

as to estimate its parameters.

- The modeler has already identified the model structure. Therefore, only a fitting problem remains to be solved in order to estimate the parameters of the model.
- The modeler considers appropriate to follow a model averaging strategy since there is not enough information in the observed data to infer a single model. Consequently, the modeler has to set up a suitable model averaging strategy, especially if his interest is to use the model for predictive purposes.

Each of these scenarios corresponds to each of the problems considered in the dissertation.

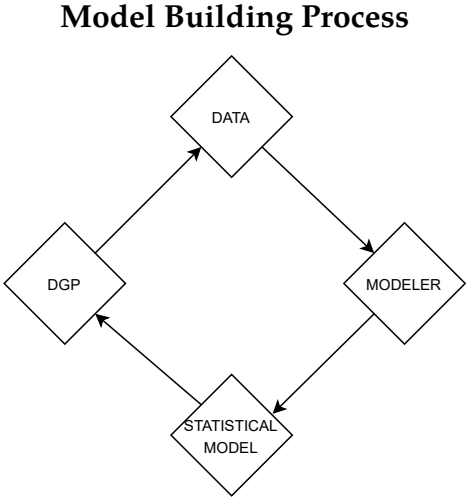



Figure 1.1: The modeler disposes of a dataset coming from a un unknown Data Generating Process (DGP). The aim of the modeler is to detect and fit a final model in order to approximate the DGP.

Best subset selection in linear regression is the first considered problem. It consists of finding the subset of predictors that produces the best fit in terms of squared error: there is need of both fitting and selection. A sparse linear model increases the degree of interpretability and reduces the risk of overfitting which impacts negatively on the predictive performance of the model. Furthermore, sparsity is also necessary due to the multicollinearity issue which invalidates the statistical inference process. In Chapter 2, the best subset selection problem is tackled by means of the Alternate Minimization algorithm (Di Gangi et al., 2019). Alternate Minimization is then generalized to the area of time series analysis, dealing with the problem of order selection of Auto Regressive Moving Average (ARMA) models. In this latter case, Alternate Minimization works like a preliminary estimation algorithm: the returned ARMA model needs to be refined by a maximum likelihood step having fixed

the model structure identified by the Algorithm. The carried out experiments highlight that the Alternate Minimization algorithm is fast and reliable in both these two scenarios and, although not guaranteed to deliver the globally optimal solutions, it generates very good solutions in low computational time.

Again in the context of time series, the problem of exact maximum likelihood estimation of ARMA models is investigated in Chapter 3. Now the modeler, after having found the model structure, has still to estimate the model's parameters. However, this estimation problem suffers from two main complications: (i) the evaluation of the likelihood is expensive, (ii) the causality and invertibility constraints, respectively for the autoregressive (AR) and moving average (MA) parameters, need to be carefully managed. By means of an extensive computational study, we highlight that a bound constrained formulation of the exact maximum likelihood estimation problem of causal and invertible ARMA is convenient in terms of fitting quality as well as computational time w.r.t. the state of the art methodology.

Finally, the last contribution is about forecasting. It is well known that combining forecasting methods is a successful and widespread strategy, leading to an improvement of forecast accuracy. The issue of model selection is especially prominent in time series analysis. In fact, forecasting problems possess small and finite history of points to formulate strict hypotheses about the DGP. In this regard, any combination strategy protects against the risk of selecting a wrong model.
Traditionally, in the literature three different schemes of combining forecasts can be found: (i) simple combinations approaches as simple average, median and trimmed mean (Clemen, 1989; Stock and Watson, 2004; Jose and Winkler, 2008), (ii) regression based approaches (Bates and Granger, 1969; Granger and Ramanathan, 1984; Diebold and Shin, 2019), (iii) meta-learning based approaches (Arinze et al., 1997; Prudêncio and Ludermir, 2004; Kück et al., 2016; Talagala et al., 2018; Montero-Manso et al., 2020; Ma and Fildes, 2021; Li et al., 2020).
With the development of novel machine learning models and technologies, especially the meta-learning approaches are spreading in the forecasting context. Indeed, the more recent meta-learning systems (Montero-Manso et al., 2020; Li et al., 2020; Kang et al., 2021; Ma and Fildes, 2021) are trained to combine in a suitable manner the forecasting methods. In particular, these meta-learners are able to recommend the weights of each method in the combination. However, in any case, the forecast combination strategy has the drawback to be time consuming. In Chapter 4, we introduce two meta-learning systems, Sparse Robust Forecast Averaging (SRFA) and Sparse Flexible Forecast Averaging (SFFA), which provide sparse convex combination of forecasting methods. Sparse combinations allows forecasting methods to be selected and weighted at the same time. There is no need to fit the zero weighted forecasting models and therefore it is evident the computational savings of the strategy and its reliability in real time applications.

The rest of the Chapter thoroughly introduces and contextualizes the three following chapters.

## 1.1 Best Subset Selection in Gaussian Linear Regression models with Extension to Gaussian ARMA models

Given a set of $N$ observations $\{x_i, y_i\}_{i=1}^N$, we assume the following stochastic linear relationship:

$$Y_i = \sum_{j=1}^{P} \beta_j x_{ij} + c + \epsilon_i \quad \epsilon_i \sim \mathbb{N}(0, \sigma^2), \tag{1.1}$$

with Independent and Identically Distributed (I.I.D) Gaussian error terms $\epsilon_i$. $x_i = (x_{i1}, \ldots, x_{iP}) \in \mathbb{R}^P$ represents the predictor vector of the $i$-th observation, while the scalar $y_i$ is the response variable of the $i$-th observation. Regression parameters $\beta = (\beta_1, \ldots, \beta_P)$ describe the relationship between each predictor and the response. The variance parameter $\sigma^2$ of the error terms quantifies the average degree to which the responses differ from their conditional expectations due to the absence of some predictors or measurement errors of the response variables.

The classical formulation of best subset selection problem does not involve any probabilistic assumption of the response variable:

$$\begin{aligned} &\min_{\beta, c} \frac{1}{2} \sum_{i=1}^{N} \left( y_i - c - \sum_{j=1}^{P} \beta_j x_{ij} \right)^2 \\ &\text{s.t } ||\beta||_0 \leq k, \end{aligned} \tag{1.2}$$

where $||\beta||_0 = |\{j \mid \beta_j \neq 0\}|$, the so-called $\ell_0$ norm [2] of parameters $\beta \in \mathbb{R}^{P \times 1}$, is constrained to not exceed the subset size $k$. Since the knowledge of $k$ is required in (1.2), this formulation is known as the cardinality constrained version of best subset selection problem. Problem (1.2) is NP-hard (Natarajan, 1995) and it has been widely dismissed as being intractable by the statistical community (Bertsimas et al., 2016). The Lagrangian version of Problem (1.2) is:

$$\min_{\beta, c} \frac{1}{2} \sum_{i=1}^{N} \left( y_i - c - \sum_{j=1}^{P} \beta_j x_{ij} \right)^2 + \lambda ||\beta||_0, \tag{1.3}$$

[2] The usage of term *norm* is wrong to refer to the function $||.||_0$. In fact, $||.||_0$ does not satisfy all the properties which characterize a *norm* function (Boyd et al., 2004).

where $\lambda > 0$ is a regularization parameter which encourages sparse linear regression models but does not guarantee a sparse solution.

To overcome the computational burden associated to both the constrained (1.2) and unconstrained formulations (1.3), many approximating procedures have been proposed in the literature. Forward selection and Backward elimination algorithms represent two intuitive but greedy strategies to identify subset of predictors, see (Friedman et al., 2001). However, the Lasso estimator (Tibshirani, 1996) is certainly the most popular approximating method for best subset selection problem. The Lasso formulation is based on the following convex programming problem:

$$\begin{aligned} &\min_{\beta,c} \frac{1}{2}\sum_{i=1}^{N}\left(y_i - c - \sum_{j=1}^{P}\beta_j x_{ij}\right)^2 \\ &\text{s.t } ||\beta||_1 \leq t, \end{aligned} \tag{1.4}$$

where the parameter $t > 0$ assesses the degree of sparsity of the solution. Crucial is the presence of the convex function $||.||_1$, called $\ell_1$ norm, which sums the absolute values of the elements of parameters $\beta$. Lasso tends to produce sparse regression models by the shrinkage of parameters towards zero induced by the occurrence of $||.||_1$. Sparsity of Lasso estimator is not guaranteed but for geometric reasons inherent to $\ell_1$ norm, it is very likely that some of the elements of $\beta$ are set exactly equal to zero. In particular, let $\hat{\beta}^{\text{ols}} = (\hat{\beta}_1^{\text{ols}}, \ldots, \hat{\beta}_P^{\text{ols}})$ the full least squares estimates and let $t^{\text{ols}} = ||\hat{\beta}^{\text{ols}}||_1$. Values of $t < t^{\text{ols}}$ will cause shrinkage of the solutions towards 0, and some coefficients may be exactly equal to 0. For example, if $t = \frac{t^{\text{ols}}}{2}$, the effect will be roughly similar to finding the best subset of size $\frac{P}{2}$.
Lagrangian version of Lasso is:

$$\min_{\beta,c} \frac{1}{2}\sum_{i=1}^{N}\left(y_i - c - \sum_{j=1}^{P}\beta_j x_{ij}\right)^2 + \lambda||\beta||_1. \tag{1.5}$$

Computational efficiency and scalability to practical sized problems are the two main appealing properties of Lasso estimator which have enhanced its popularity. However, its shortcomings are also well known in literature. Lasso suffers of estimation bias, tending to give biased estimates for large regression parameters (Fan and Li, 2001). On this matter, to correct this bias, a smoothly clipped absolute deviation (SCAD) penalty function, instead of $||.||_1$ in (1.5), is introduced in (Fan and Li, 2001). Furthermore, in (Zhao and Yu, 2006), it is shown that Lasso achieves consistent selection of predictors in a narrow range of problems where the technical *irrepresentable conditions* hold. Again, selecting optimal $\lambda$ for prediction gives inconsistent predictor selection results (Meinshausen and Bühlmann, 2006). To overcome these issues, in (Zou, 2006) Adaptive Lasso is proposed. This method replaces the $\ell_1$ norm Lasso penalty by a re-weighted version. The key idea is that the value of

parameter $\lambda$ is tuned individually for each predictor based on a previous fitted regression model. Both SCAD and Adaptive Lasso enjoy the Oracle properties (Fan and Li, 2001; Zou, 2006), i.e. consistency in selecting the true subset of predictors and asymptotic normality in estimation.

Penalized likelihood approaches (Fan and Lv, 2010) represent the natural generalization of penalized least squares methods (1.2), (1.3), (1.4), (1.5). Now, we assume that the observed data $\{x_i, y_i\}_{i=1}^N$ are generated according to (1.1). The formulation of penalized likelihood maximization problem for best subset selection in Gaussian linear regression is:

$$\max_{\beta,c,\sigma^2} \; l(\beta, c, \sigma^2) - \lambda ||\beta||_0, \tag{1.6}$$

where the term $l(\beta, c, \sigma^2)$ is the log-likelihood function of the Gaussian linear regression model. In this case, the log-likelihood term provides a measure of fit of the model, while the $\ell_0$ norm encourages parsimonious models exactly as in (1.3). The optimization of known information criteria as the AIC (Akaike Information Criterion) (Akaike, 1974), BIC (Bayesian Information Criterion) (Schwarz, 1978) or HQIC (Hannan Quinn Information Criterion) (Hannan and Quinn, 1979) represents a specific instance of Problem (1.6).

The proposed Alternate Minimization algorithm (Di Gangi et al., 2019) is conceived to minimize the AIC, BIC or HQIC information criteria of Gaussian linear regression models and also Gaussian ARMA models. The problem is addressed through a formulation based on mixed integer optimization for handling the $\ell_0$ norm.

## 1.2 Improved Maximum Likelihood Estimation of ARMA Models

The process $\{Y_t,\ t \in \mathbb{Z}\}$ is a Gaussian zero mean ARMA process of order $(p, q)$ if the following linear stochastic difference equation holds:

$$Y_t - \phi_1 Y_{t-1} - \cdots - \phi_p Y_{t-p} = \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t, \quad \epsilon_t \sim \mathcal{WN}(0, \sigma^2). \tag{1.7}$$

In Equation 1.7, $\phi = (\phi_1, \ldots, \phi_p)$, which refer to the AR part, relate current observations to past observations, while $\theta = (\theta_1, \ldots, \theta_q)$, which are the parameters of the MA part, encode the dependence of current observations to previous error terms. The error terms $\epsilon_t$ in 1.7 are modeled as a Gaussian white noise process, i.e. $\epsilon_t \sim \mathcal{WN}(0, \sigma^2)$.

Every ARMA model admits also a representation as a state space model which is in the form of a linear system of two sets of linear stochastic equations (Hamilton, 1994). This representation is quite general and flexible as it allows the handling of

many time series analysis problems, such as prediction, signal extraction, decomposition, parameter estimation and interpolation, within a single framework.

A Gaussian ARMA$(p,q)$ time series $\mathbf{y}_N = \{y_t\}_{t=1}^N$ , which is a single finite realization of such a stochastic process, has a zero mean Gaussian joint distribution:

$$\mathbf{Y}_N \sim \mathbb{N}\Big(\mathbf{0}, \Gamma_N(\phi,\theta,\sigma^2)\Big). \tag{1.8}$$

In the ARMA$(p,q)$ case, the covariance $\Gamma_N(\phi,\theta,\sigma^2) = \mathbb{E}(\mathbf{Y}_N\mathbf{Y}_N^T)$ is expressible in terms of a finite number of unknown parameters (Brockwell et al., 1991). Therefore, from (1.8), the Gaussian ARMA log-likelihood function $l(\phi,\theta,\sigma^2)$ referring to an observed time series $\mathbf{y}_N$ is:

$$l(\phi,\theta,\sigma^2) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log\big(\det(\Gamma_N)\big) - \frac{1}{2}\mathbf{y}_N^T\Gamma_N^{-1}\mathbf{y}_N \tag{1.9}$$

Calculation of (1.9) requires the inversion of the $\Gamma_N$ matrix and the computation of its determinant that may be complicated when $N$ is large. Kalman filter algorithm (Kalman, 1960), which works for any state space model, is also useful for efficient evaluation of the ARMA log-likelihood function (1.9). The algorithm avoids the direct computation of $\Gamma_N^{-1}$ and $\det(\Gamma_N)$ by employing the prediction error decomposition of the likelihood function. By means of this decomposition, the log-likelihood $l(\phi,\theta,\sigma^2)$ becomes:

$$l(\phi,\theta,\sigma^2) = \sum_{t=1}^{N}\log\Big(f_{t|t-1}(y_t|\mathbb{I}_{t-1})\Big), \tag{1.10}$$

with conditional Gaussian terms

$$f_{t|t-1}(y_t|\mathbb{I}_{t-1}) = -\frac{1}{2}\log(2\pi) - \log(\sigma_{e_t}) - \frac{1}{2}\frac{(y_t - y_{t|t-1})^2}{\sigma_{e_t}^2}.$$

$f_{t|t-1}(.)$ is the density of $y_t$ conditional on the past information up to time $t-1$, i.e. $\mathbb{I}_{t-1} = \{y_1,\ldots,y_{t-1}\}$ , with $y_{t|t-1} = \mathbb{E}[Y_t|\mathbb{I}_{t-1}]$ (Conditional expectation) and $\sigma_{e_t}^2 = \mathrm{Var}(Y_t|\mathbb{I}_{t-1}) = \mathrm{Var}(Y_t - y_{t|t-1}|\mathbb{I}_{t-1})$ (Prediction Error Variance). The building blocks of the decomposition are the forecast errors $y_t - y_{t|t-1}$ and their variances $\sigma_{e_t}^2$. Kalman filter provides a recursive and efficient computation of these quantities. Typically, in terms of software implementation, a compiled subroutine evaluates the value of the likelihood for a given set of parameters. Note also that the gradient of the log-likelihood function is usually approximated by numerical difference, requiring the evaluation of $l(\phi,\theta,\sigma^2)$ many times (Kitagawa, 2020).

One of the main applications of ARMA models is forecasting. Usually in the forecasting context, ARMA parameters are constrained to satisfy the causality and invertibility conditions (Brockwell et al., 1991). These conditions, which determine

the feasibility space of the ARMA$(p, q)$ structural parameters, i.e. $(\phi, \theta) \in S_p \times S_q$, can be stated in terms of the absolute values of the roots of the autoregressive $\Phi(z)$ and moving-average $\Theta(z)$ polynomials:

$$S_p = \{\phi \in \mathbb{R}^p \mid \Phi(z) \neq 0 \; \forall \, z \in \mathbb{C} \text{ s.t. } |z| \leq 1\} \tag{1.11}$$

$$S_q = \{\theta \in \mathbb{R}^q \mid \Theta(z) \neq 0 \; \forall \, z \in \mathbb{C} \text{ s.t. } |z| \leq 1\}, \tag{1.12}$$

where $\Phi(z)$ and $\Theta(z)$ are defined as

$$\Phi(z) = 1 - \phi_1 z - \ldots - \phi_p z^p \tag{1.13}$$

$$\Theta(z) = 1 + \theta_1 z + \ldots + \theta_q z^q. \tag{1.14}$$

The geometry of the space $S_p \times S_q$ is thoroughly described in (Piccolo, 1982; Shlien, 1985; Picinbono and Benidir, 1986; Combettes and Trussell, 1992).

According to the proposed fitting method, causality and invertibility are handled by reformulating the ARMA fitting problem as a bound constrained optimization problem. In our implementation, evaluation of the likelihood $\ell(\phi, \theta, \sigma^2)$ is still performed by means of a Kalman filter algorithm.

## 1.3 Sparse Convex Combinations of Forecasting Models By Meta Learning

Let $f_i, \; i = 1, \ldots, L$, be the $L$ forecasts to be combined. The forecast combination can be written as

$$f_{\text{comb}} = \sum_{i=1}^{L} w_i f_i \tag{1.15}$$

The forecast combination approach typically uses a linear combination of forecasts obtained from different models for the same time series. In the meta-learning context the aim is to learn a weighting function

$$h(.) : T \rightarrow W,$$

which maps a time series $\mathbf{y}_N \in T$ into an appropriate set of combination weights $w = \{w_i\}_{i=1}^{L}$. Usually, convex combinations of forecast methods are employed, i.e. $W$ is the probability simplex. Note that the choice of convex combinations of forecasting models, though it is the most intuitive, is not the only possible, see for example (Granger and Ramanathan, 1984; Radchenko et al., 2020).

However, the learning problem, for how it has been introduced, is ill-posed. Each time series is simply a sequence of data points representing the temporal evolution of a given phenomenon. This original representation does not emphasize the nature of data, e.g. the presence of trend and seasonality, the degree of noise and

non-linearity or any other measurable characteristic of a time series. Moreover, it is well known that these characteristics affect the performance of forecasting methods (Reid, 1972).

Therefore, in order to properly defining the learning problem, there is a need to introduce a feature extraction function

$$fe(.) : T \rightarrow X,$$

which returns a new representation $x \in X$ of the series as a set of its meaningful and measurable characteristic, i.e. the time series feature representation.

Hence, the whole problem consists in choosing both $fe(.)$ and a novel weighting function $\tilde{h}(.)$ which maps time series features into weights of combination:

$$\begin{aligned} & T \overset{fe(.)}{\rightarrow} X \overset{\tilde{h}(.)}{\rightarrow} W \\ & fe(.) : T \rightarrow X \quad \textit{(feature extractor)} \\ & \tilde{h}(.) : X \rightarrow W \quad \textit{(weighting function)} \end{aligned}$$

In the literature two different ways of extracting time series features can be found: (i) manual approaches of feature extraction (Wang et al., 2009; Widodo and Budi, 2013; Talagala et al., 2018; Montero-Manso et al., 2020; Kang et al., 2021), (ii) automatic approaches in which features are learned automatically from data by appropriate neural networks architectures (Li et al., 2020; Ma and Fildes, 2021).

Usually, the modeler chooses in advance how to extract time series features and the learning problem is substantiated in estimating $\tilde{h}(.)$. In order to estimate $\tilde{h}(.)$, the modeler needs to collect a set of past forecasting episodes with the associated performances of methods and characteristics of time series (meta-data). Training a learning model on meta-data, i.e. a meta-learner, the modeler can finally infer an appropriate weighting function $\tilde{h}(.)$ which can then be employed to recommend the weights of combination on new time series to be forecast.

Both the proposed SRFA and SFFA methods face the forecasting problem just as a meta-learning problem. These meta-learners are distinguished by the way the function $\tilde{h}(.)$, which recommends sparse combinations, is inferred from past forecasting episodes.

# Chapter 2

# Best Subset Selection in Gaussian Linear Regression models with Extension to Gaussian ARMA models

We employ the AIC (Akaike, 1974), BIC (Schwarz, 1978) or HQIC (Hannan and Quinn, 1979) information criteria to solve best subset selection problem in Gaussian linear regression. The use of information criteria as a tool for model selection, assumes the willingness of a parsimonious linear regression model, in the sense that not all the $P$ predictors in Equation (1.1) are really relevant in explaining the response variable [1]. Using these criteria, the modeler has the important advantage that the number of predictors to be selected is implicitly determined by the criterion without having to specify in advance this number. Hence, the modeler has only to choose the right criterion based on the purpose of the model.

As mentioned in Section 1.1, the optimization of information criteria represents a specific instance of the general penalized likelihood maximization problem (1.6). A penalized likelihood maximization approach (1.6), with the presence of a $\ell_0$ norm regularization term, promotes parsimonious regression models by encouraging sparse estimates of $\beta$. Optimization of (1.6) requires the handling of the $\ell_0$ term.

Recent approaches (Bertsimas et al., 2016; Miyashiro and Takano, 2015a,b; Gómez and Prokopyev, 2018) for best subset selection problem have started to propose formulations based on mixed integer optimization to handle the $\ell_0$ norm. However, these formulations, as our experiment also highlight, have the drawback of being computationally expensive. On the contrary, Alternate Minimization Algorithm manages the presence of the $\ell_0$ norm by sequentially solving MIQP problems, i.e. models with quadratic objective, linear constraints and both continuous as well as discrete variables, whose solution can be found in low computational times by Gurobi (Gurobi Optimization LLC, 2018). In particular, the core of the Algorithm

[1] Full model view interpretation of regression parameters, see (Berk et al., 2013).

is based on a two-step Gauss-Seidel decomposition scheme, where at each iteration the first step involves the update of the regression parameters $\beta$ by solving a MIQP problem and the second step is about the closed form update of the variance parameter $\sigma^2$.

We start the Chapter with a review in Section 2.1 of the AIC, BIC and HQIC information criteria. A sketch derivation of both the AIC and BIC criteria, which are heavily used in practice, is also given in Appendix A. A description of the most known related works is reported in Section 2.2. Alternate Minimization is presented in Section 2.3 and its convergence analysis in Section 2.4. The three variants of Alternate Minimization for its adaption to Gaussian ARMA models are treated in Section 2.5. Finally, Sections 2.6 and 2.7 report results of computational experiments, respectively about linear regression and time series scenarios.

## 2.1 AIC, BIC and HQIC information criteria

AIC, BIC and HQIC information criteria refer to model selection methods that are based on log-likelihood function $l(\theta)$ and applicable to parametric model-based problems. These criteria are conceived to find a proper model for data under investigation providing a measure of information that strikes a balance between the goodness of fit and parsimonious specification of the model.

### AIC

The AIC criterion is based on Kullback–Leibler (KL) divergence [2] (Kullback and Leibler, 1951) which is a general tool to measure the distance between two statistical models or in general to compare two probability distributions. In particular, Akaike found a relationship between the maximum likelihood estimator and KL divergence which led to the definition of the following criterion:

$$\text{AIC} = -2l(\hat{\theta}) + 2k = -2l(\hat{\theta}) + 2||\hat{\theta}||_0, \tag{2.1}$$

where $k$ is the dimension of the fitted model, which can be computed as the $\ell_0$ norm of the maximum likelihood estimate $\hat{\theta}$ (Fan and Lv, 2010). The AIC provides an estimate, only asymptotically valid, of the expected, relative distance between the fitted model and the unknown true mechanism that generated the data. Comparing a set of candidate models, the one with smallest AIC is considered closer to the truth than the others. Note that the presence of the factor 2 in the AIC definition (2.1) is only for historical reasons.

[2] Also note as Kullback–Leibler information or relative entropy.

### BIC

The BIC criterion is derived as an asymptotic approximation of the posterior probability of the Bayesian posterior probability of a candidate model (Neath and Cavanaugh, 2012). Its derivation follows Bayesian arguments although its computation only requires the computation of maximum likelihood estimator. BIC criterion is defined as follows:

$$\text{BIC} = -2l(\hat{\theta}) + k\ln(N) = -2l(\hat{\theta}) + \ln(N)||\hat{\theta}||_0, \tag{2.2}$$

where $N$ is the sample size. The BIC criterion, interpreted as a penalized-likelihood, weighs down model complexity more heavily than the AIC criterion. Model selection based on BIC is advantageous in the sense that BIC has the property of consistency (Neath and Cavanaugh, 2012). Suppose that the DGP is of finite dimension, and that this model is represented in the candidate collection. A consistent criterion will asymptotically select, with probability one, the candidate model having the correct structure.

### HQIC

The HQIC (Hannan and Quinn, 1979), is defined as

$$\text{HQIC} = -2l(\hat{\theta}) + 2(||\hat{\theta}||_0)\log(\log N). \tag{2.3}$$

This criterion provides a consistent estimator of the order of an autoregressive model (Konishi and Kitagawa, 2008; Broersen, 2006). HQIC is mainly suited for the autoregression setting and seems to have little use in practice (Burnham and Anderson, 2002).

## 2.2 Related Works

As reported in Section 1.1, approximating approaches, mainly represented by the Lasso estimator (Tibshirani, 1996) with its extensions (see, for example (Fan and Li, 2001; Zou, 2006)) and greedy approaches as the Forward selection and Backward elimination algorithms have been proposed in the literature to solve best subset selection problem. Always in this context, Least Angle Regression (LARS) Algorithm (Efron et al., 2004) is also worth mentioning. In fact, a simple modification of the LARS algorithm implements the Lasso while a different one leads to an efficient version of the Forward selection heuristic.

More recently, formulations based on mixed integer models have been introduced to solve best subset selection problem (Bertsimas and Shioda, 2009; Konno and Yamamoto, 2009; Miyashiro and Takano, 2015a,b; Bertsimas and King, 2015; Bertsimas et al., 2016; Gómez and Prokopyev, 2018).

When the cardinality parameter $k$ is given, Problem (1.2) can be formulated as a MIQP problem (Bertsimas et al., 2016). In fact, by introducing binary indicator variables $z \in \{0,1\}^P$ s.t $z_i = 0$ only if $\beta_i = 0$, Problem (1.2) is equivalent to:

$$\begin{aligned} \min_{\beta,\, c,\, z} \; & \sum_{i=1}^{N} \left( y_i - c - \sum_{j=1}^{P} \beta_j x_{ij} \right)^2, \\ & e^T z \leq k, \\ & -Mz \leq \beta \leq Mz, \\ & z \in \{0,1\}^P, \beta \in \mathbb{R}^P, \end{aligned} \tag{2.4}$$

where $e$ is the vector of $P$ elements all equal to 1 and $M$ is a sufficiently large constant. However, the formulation (2.4), although simple and easy to be implemented, stands on the knowledge of $k$ that is not always available.

Non-cardinality constrained formulations of subset selection problem are based on the minimization of information criteria. The negative double log-likelihood of a linear regression model is

$$-2\ell(\beta, c, \sigma^2) = N \log(\sigma^2) + N \log(2\pi) + \frac{1}{\sigma^2} \sum_{i=1}^{N} \left( y_i - c - \sum_{j=1}^{P} \beta_j x_{ij} \right)^2.$$

Therefore, for the above reported information criteria (AIC, BIC and HQIC), we can use the following general notation:

$$-2\ell(\beta, c, \sigma^2) + \alpha(\|\beta\|_0 + 2),$$

where $\alpha$ depends on the chosen information criterion. By removing the constant terms from the objective function, we can consider the following optimization problem:

$$\min_{\beta,\, c,\, \sigma^2} N \log(\sigma^2) + \frac{1}{\sigma^2} \sum_{i=1}^{N} \left( y_i - c - \sum_{j=1}^{P} \beta_j x_{ij} \right)^2 + \alpha(\|\beta\|_0). \tag{2.5}$$

By first order conditions, we can easily derive the maximum likelihood estimator of the variance parameter $\sigma^2$, i.e.

$$\hat{\sigma}^2 = \frac{R(\beta,\ c)}{N},$$

being $R(\beta,\ c) = \sum_{i=1}^{N} (y_i - c - \sum_{j=1}^{P} \beta_j x_{ij})^2$. Substituting in (2.5), we get the following problem:

$$\min_{\beta,\, c} \; N \log \left( \frac{R(\beta, c)}{N} \right) + N + \alpha \|\beta\|_0. \tag{2.6}$$

Problem (2.6) can also be reformulated as the following Mixed-Integer Second Order Cone Programming (MISOCP) problem (Miyashiro and Takano, 2015a):

$$\min_{f} \; f \tag{2.7a}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \varepsilon_i^2 \leq f \cdot \sum_{j=0}^{P} \left( w_j \cdot \exp\left( -\frac{\alpha j}{N} \right) \right) \tag{2.7b}$$

$$\sum_{j=0}^{P} (j \cdot w_j) = \sum_{j=1}^{P} z_j \tag{2.7c}$$

$$\sum_{j=0}^{P} w_j = 1 \tag{2.7d}$$

$$-Mz_j \leq \beta_j \leq Mz_j \quad j = 1, \dots P \tag{2.7e}$$

$$\beta \in \mathbb{R}^P,\ c \in \mathbb{R},\ \epsilon \in \mathbb{R}^N,\ f \in \mathbb{R}_+,\ w \in \{0,1\}^{P+1},\ z \in \{0,1\}^P \tag{2.7f}$$

This mixed-integer model is elegant, but in fact, as our experiments in Section 2.6 highlight, cannot be solved so efficiently. In particular, as shown in Table 2.2, this method is in practice much slower not only w.r.t. the stepwise heuristics, but also with respect to the approach of solving problem (2.4) once for each value of size $k$.

A Mixed-Integer Fractional formulation (MIFO) has also been proposed (Gómez and Prokopyev, 2018):

$$\min_{\beta,\, c,\, s} \; \frac{\sum_{i=1}^{N} \left( y_i - c - \sum_{j=1}^{P} \beta_j x_{ij} \right)^2}{s} \tag{2.8a}$$

$$\text{s.t.} \quad s \leq \sum_{i=0}^{P} g(i) w_i \tag{2.8b}$$

$$\sum_{i=0}^{P} i \cdot w_i = \sum_{i=1}^{P} z_i \tag{2.8c}$$

$$\sum_{i=0}^{P} w_i = 1 \tag{2.8d}$$

$$-Mz \leq \beta \leq Mz \tag{2.8e}$$

$$z \in \{0,1\}^P,\ w \in \{0,1\}^{P+1},\ \beta \in \mathbb{R}^P,\ s \geq 0, \tag{2.8f}$$

where $s$ is an additional variable and $g(x) = \exp(-\alpha x/N)$. Problem (2.8) can be tackled exploiting the efficient MIQO solvers. In particular, the parametrized problems

$$d(t) = \min_{\beta,\, c,\, s} \; R(\beta,\ c) - ts \tag{2.9a}$$

$$\text{s.t.} \quad \text{(2.8b) – (2.8f)} \tag{2.9b}$$

are considered. If $d(t^\star) = 0$, then $t^\star$ is the optimal value of (2.8). The original problem is thus solved finding a root of the equation $d(t) = 0$ by using Newton method. The whole procedure thus reduces to solving a sequence of problems of the form (2.9), which can be proved to terminate at most after the $P+1$-th problem.

## 2.3 Alternate Minimization Algorithm

Let us consider optimization problems of a slightly more general form w.r.t. (2.5):

$$\begin{aligned} \min_{\beta,c,\sigma} \quad & N\log(\sigma^2) + \frac{1}{\sigma^2}R(\beta,c) + g(\beta) \\ \text{s.t.} \quad & \sigma > 0, \quad c \in \mathbb{R} \\ & \beta \in \mathbb{R}^P, \end{aligned} \tag{2.10}$$

where $g : \mathbb{R}^P \to S$, being $S$ a discrete set with finite cardinality, and $R : \mathbb{R}^P \times \mathbb{R} \to (\epsilon, +\infty)$ a quadratic convex function.

We propose a new method to solve such type of problems. Note that problem (2.5) is indeed an instance of (2.10), where

$$g(\beta) = \alpha\|\beta\|_0, \qquad R(\beta,c) = \sum_{i=1}^{N}\left(y_i - c - \sum_{j=1}^{P}\beta_j x_{ij}\right)^2.$$

Fixing $\bar\beta, \bar c$, we get the convex, univariate problem

$$\min_{\sigma>0} \quad N\log(\sigma^2) + \frac{R(\bar\beta,\bar c)}{\sigma^2},$$

whose closed form solution, if $R(\bar\beta,\bar c) > 0$, is immediately found to be

$$\bar\sigma^2 = \frac{R(\bar\beta,\bar c)}{N}. \tag{2.11}$$

Note that this relation has to hold at every optimal solution $\bar\beta, \bar c, \bar\sigma$ of problem (2.10), so we might equivalently reformulate the problem in a concentrated form as

$$\min_{\beta,c} N\log\left(\frac{R(\beta,c)}{N}\right) + g(\beta) + N, \tag{2.12}$$

similarly as in (2.6).

However, differently from e.g. the MISOCP (Miyashiro and Takano, 2015a) or the MIFO (Gómez and Prokopyev, 2018) formulations, we do not employ the concentrated formulation (2.12). On the contrary, we propose a two blocks Gauss-Seidel

type solving scheme (Bertsekas, 2016) for formulation (2.10). The proposed approach is able to produce a much more efficient method. The procedure is described by Algorithm 1.

**Algorithm 1** Alternate Minimization (AM)

**Input:** $\beta^0,\ c^0,\ \sigma_0, k = 0$

1: let $g(\beta^{-1}) = \texttt{NaN}$
2: **while** $g(\beta^k) \neq g(\beta^{k-1})$ **do**
3: set

$$\beta^{k+1},\ c^{k+1} = \arg\min_{\beta,\ c} \frac{R(\beta, c)}{{\sigma_k}^2} + g(\beta)$$

4: set

$$\sigma^2_{k+1} = \arg\min_{\sigma^2>0} N\log(\sigma^2) + \frac{R(\beta^{k+1},\ c^{k+1})}{\sigma^2} = \frac{R(\beta^{k+1},\ c^{k+1})}{N}$$

5: set $k = k + 1$
6: **end while**
7: **return** $\beta^k,\ c^k,\ \sigma_k$

The major computational effort required by Algorithm 1 is in the solution of the optimization problem defined at step 3 of the procedure.

At first glance, dealing with such subproblems may seem particularly hard, since the algorithm requires the global optimum and the objective function is discontinuous. However, problem

$$\min_{\beta,\ c} \quad \frac{R(\beta, c)}{{\sigma_k}^2} + g(\beta) \tag{2.13}$$

can be equivalently reformulated as a mixed-integer convex quadratic problem; in particular it is equivalent to:

$$\min_{\beta,\ c} \ R(\beta,\ c) + \Lambda\sigma_k^2 \tag{2.14a}$$

$$\text{s.t. } \Lambda = g(\beta) \tag{2.14b}$$

$$\Lambda \in S, \tag{2.14c}$$

$$\beta \in \mathbb{R}^P, \quad c \in \mathbb{R}, \tag{2.14d}$$

where, based on the structure of $g$, the constraint (2.14b) can be turned into a linear constraint by introducing auxiliary binary and integer variables and linear constraints. For example, if $g(\beta) = \alpha\|\beta\|_0$, as in the case of linear regression, we can

substitute constraints (2.14b)-(2.14c) with the following set of constraints:

$$\delta_i \in \{0,1\} \quad \forall\, i = 1,\dots,P \tag{2.15a}$$

$$-M\delta_i \leq \beta_i \leq M\delta_i \quad \forall\, i = 1,\dots,P \tag{2.15b}$$

$$\Lambda = \alpha \sum_{i=1}^{P} \delta_i. \tag{2.15c}$$

If $M$ is a large enough positive constant, the value of variable $\Lambda$, appearing in the objective function, will be equal to the number of non-zero components of $\beta$: indeed, if $\beta_i \neq 0$, then $\delta_i$ will have to assume the value 1 in order to satisfy constraint (2.15a), while if $\beta_i = 0$, then $\delta_i$, which may assume both values 0 and 1, will be 0, since it brings to a lower objective value.

Problem (2.14) with constraints of the type (2.15a)-(2.15c) is usually solved significantly faster than, e.g., the MISOCP problem (2.7) that directly tackles Problem (2.12) (Miyashiro and Takano, 2015a). Indeed, solving a sequence of problems of the form of (2.14) proved to be more efficient than solving (2.7), as we will show in Section 2.6. In fact, Problem (2.14) is similar, in terms of complexity, to Problem (2.4), which is solved $P$ times through enumeration. On the other hand, even though we will briefly proof that Algorithm 1 also solves, in the worst case, $P$ times problem (2.14), it in fact usually stops after much less than $P$ iterations.

## 2.4 Convergence Analysis

The following Proposition characterizes the properties of Algorithm 1. In order to simplify the notation throughout the analysis, we will ignore, with no loss of generality, the intercept term $c$. Before going on with the analysis, we also make an assumption, which is reasonable in practical implementations, about Algorithm 1:

**Assumption 1.** *Step 3 of Algorithm 1 is performed in such a way that if* $R(\beta^{k+1})/\sigma_k^2 + g(\beta^{k+1}) = R(\beta^k)/\sigma_k^2 + g(\beta^k)$ *then* $\beta^{k+1} = \beta^k$.

The above assumption substantially says that the current point is updated only if the new point is strictly better than the previous one in terms of objective value. For the sake of notation simplicity, let also $f(\beta) = N\log(R(\beta)/N) + g(\beta)$. Now, we can finally turn to the convergence analysis.

**Proposition 1.** *Consider Algorithm 1, under Assumption 1. Then, the following properties hold:*

*(a) For each iteration* $k$, *either* $g(\beta^k) = g(\beta^{k-1})$, *i.e. the algorithm terminates, or* $g(\beta^k) \neq g(\beta^h)$ *for all* $h < k$.

*(b) The algorithm terminates in at most* $|S|$ *iterations, returning a solution* $(\bar{\beta}, \bar{\sigma})$.

(c) *Let $\bar{k}$ be the index of the last iteration. Then $f(\bar{\beta}) \leq f(\beta)$ for all $\beta \in \{\beta \mid \exists k \in \{1, \ldots, \bar{k}\}$ s.t. $g(\beta^k) = g(\beta)\}$.*

(d) *If $\bar{k} = |S| + 1$, then the returned solution $\bar{\beta}$ is optimal.*

(e) *Let $\beta^\star$ be an optimal solution of problem* (2.12)*, i.e., the pair $\beta^\star$, $\sigma^\star = \sqrt{R(\beta^\star)/N}$ is optimal for problem* (2.10)*. Then, the following bound holds:*

$$0 \leq f(\bar{\beta}) - f(\beta^\star) \leq -N \log(1 - \eta^2 \exp(\theta - 1)), \tag{2.16}$$

*where $\theta \in (0, 1)$ and $\eta = (g(\bar{\beta}) - g(\beta^\star))/N$.*

*Proof.* We prove one property at a time.

(a) Since both step 3 and step 4 require to compute global minima of subproblems, the sequence of objective values $\{f(\beta^k)\}$ is monotone non-increasing.

By the instructions of the algorithm,

$$\beta^{k+1} = \arg\min_{\beta} \frac{R(\beta)}{\sigma_k^2} + g(\beta),$$

so it also holds

$$\beta^{k+1} = \arg\min_{\beta : g(\beta) = g(\beta^{k+1})} \frac{R(\beta)}{\sigma_k^2} + g(\beta) = \arg\min_{\beta : g(\beta) = g(\beta^{k+1})} R(\beta). \tag{2.17}$$

Therefore, if $g(\beta^h) = g(\beta^k)$ for two indexes $h$ and $k$, $R(\beta^k) = R(\beta^h)$ and thus $f(\beta^k) = f(\beta^h)$. Now, let $k > h$ and assume $g(\beta^k) = g(\beta^h)$. Since the sequence of objective values is non-increasing, we have

$$f(\beta^k) \leq f(\beta^\ell) \leq f(\beta^h) = f(\beta^k)$$

for all $k > \ell \geq h$, where the last equality comes from the previous considerations. Thus, we have

$$f(\beta^\ell) = f(\beta^k)$$

for all $k > \ell \geq h$.

Since the objective value has not decreased from iteration $h$ to $k$, from Assumption 1 it has to be $\beta^\ell = \beta^h$ for all $k > \ell \geq h$.

But then, $g(\beta^\ell) = g(\beta^h)$ for all $k > \ell \geq h$, which is only possible if $\ell = h = k - 1$, since $g(\beta^{h+1}) = g(\beta^h)$ triggers the stopping criterion.

Therefore, for all $h < k$, we have either $h = k - 1$ with $g(\beta^k) = g(\beta^{k-1})$ or $g(\beta^k) \neq g(\beta^h)$.

(b) Since $g$ can have at most $|S|$ possible different values and, from (a), at each iteration the algorithm either finds a solution with an unseen value of $g$ or maintains the previous solution, no later than at the beginning of the $|S|+1$-th iteration the stopping criterion gets satisfied.

(c) From the non-increasing property of $\{f(\beta^k)\}$, $f(\beta^{\bar{k}}) \leq f(\beta^k)$ for all $k$. Moreover, recalling that, for all $k$, (2.17) holds, it has to be

$$\beta^k = \underset{\beta\,:\,g(\beta)=g(\beta^k)}{\arg\min} f(\beta) = N \log\left(\frac{R(\beta)}{N}\right) + g(\beta).$$

Thus we can conclude that

$$\beta^{\bar{k}} = \underset{\beta\,:\,\exists\, h\in\{1,\dots,\bar{k}\}\,:\,g(\beta)=g(\beta^h)}{\arg\min} f(\beta).$$

(d) This property directly descends from (a), (b) and (c).

(e) Since $\beta^\star$ is the optimal solution, we have

$$\beta^\star = \underset{\beta}{\arg\min}\, N \log\left(\frac{R(\beta)}{N}\right) + g(\beta), \tag{2.18}$$

while, by the instructions of the algorithm, we know that the returned solution $(\bar{\beta}, \bar{\sigma})$ satisfies

$$\bar{\sigma}^2 = \frac{R(\bar{\beta})}{N}, \tag{2.19}$$

$$\bar{\beta} = \underset{\beta}{\arg\min}\, N\frac{R(\beta)}{R(\bar{\beta})} + g(\beta). \tag{2.20}$$

From (2.18), we have

$$N \log\left(\frac{R(\beta^\star)}{N}\right) + g(\beta^\star) \leq N \log\left(\frac{R(\bar{\beta})}{N}\right) + g(\bar{\beta}),$$

while (2.20) implies

$$N + g(\bar{\beta}) \leq N\frac{R(\beta^\star)}{R(\bar{\beta})} + g(\beta^\star).$$

Now, let $r = R(\beta^\star)/R(\bar{\beta})$ and $\Delta = g(\bar{\beta}) - g(\beta^\star)$. We can rewrite the previous inequalities in a more compact way:

$$N + \Delta \leq Nr \quad \Longleftrightarrow \quad r \geq 1 + \Delta/N \tag{2.21}$$

$$N\log(r) \leq \Delta \quad \Longleftrightarrow \quad r \leq \exp(\Delta/N) \tag{2.22}$$

Let us recall Taylor's theorem:

$$s(y) = s(x) + s'(x)(y-x) + s''(\xi)(y-x)^2, \quad \xi = x + \theta(y-x) \text{ for some } \theta \in (0,1),$$

and let $\eta = \Delta/N$. Setting $s(\eta) = \exp(\eta)$ and $x = 0$, we can write $\exp(\eta) = 1 + \eta + \eta^2 \exp(\theta\eta)$, for some $\theta \in (0,1)$. Rearranging we get

$$1 + \eta = \exp(\eta) - \eta^2 \exp(\theta\eta).$$

Combining inequalities (2.21) and (2.22) with the last equality, we obtain the following bounds for $r$:

$$\exp(\eta) - \eta^2 \exp(\theta\eta) \leq r \leq \exp(\eta).$$

Now, let us consider the gap, in terms of objective value, between the returned solution $\bar{\beta}$ and the optimal solution $\beta^\star$; we have

$$\begin{aligned}
0 \leq f(\bar{\beta}) - f(\beta^\star) &= N \log(1/r) + \Delta \\
&= \Delta - N \log(r) \\
&\leq \Delta - N \log(\exp(\eta) - \eta^2 \exp(\theta\eta)) \\
&= \Delta - N \log(\exp(\eta)(1 - \eta^2 \exp(\eta(\theta - 1)))) \\
&= \Delta - N \log(\exp(\eta)) - N \log(1 - \eta^2 \exp(\theta - 1)) \\
&= \Delta - N(\Delta/N) - N \log(1 - \eta^2 \exp(\theta - 1)) \\
&= -N \log(1 - \eta^2 \exp(\theta - 1)),
\end{aligned}$$

i.e., we have obtained the bound of property (e).

□

Note that the upper bound at point (e) of Proposition 1 is often very close to 0. Consider for example the case of linear regression, where $g(\beta) = \alpha\|\beta\|_0$. The bound becomes

$$f(\bar{\beta}) - f(\beta^\star) \leq -N \log\left(1 - \frac{\alpha^2(\|\bar{\beta}\|_0 - \|\beta^\star\|_0)^2}{N^2} \exp(\theta - 1)\right);$$

the bound goes to zero when $\alpha(\|\bar{\beta}\|_0 - \|\beta^\star\|_0)/N$ goes to zero. Therefore:

- the gap is zero if $\|\bar{\beta}\|_0 = \|\beta^\star\|_0$;
- if $\alpha = 0$, there is no penalty on the model complexity; $\bar{\beta}$ then minimizes $R(\beta)$ and is trivially the global optimum;

- if the absolute value of $\alpha(\|\bar{\beta}\|_0 - \|\beta^\star\|_0)/N$ is small, the optimality gap is bounded by a small quantity; moreover, note that $\left|\|\bar{\beta}\|_0 - \|\beta^\star\|_0\right| \leq P$; in most applications $P \ll N$ and for the information criteria we have considered it holds that $\alpha = o(N)$, so we are typically guaranteed to obtain, at least, a nearly-optimal solution.

Note that optimality of the returned solution $\bar{\beta}$ cannot be guaranteed. In fact, we can show by a numerical counter-example that, in unfortunate cases, Algorithm 1 may stop at suboptimal solutions:

**Example 1.** *Consider the best subset selection problem for linear regression, using AIC penalization, for the model*

$$Y = X\beta,$$

*where*

$$Y = \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix} \qquad X = \begin{bmatrix} 10 & 0.1 \\ 0.1 & 10 \\ 1 & 1 \end{bmatrix}$$

*and $\beta \in \mathbb{R}^2$ (no intercept term c). The AIC measure for the model as a function of $\beta$ is given, up to constants, by*

$$f(\beta) = 3\log\left(\frac{\|Y - X\beta\|_2^2}{3}\right) + 2\|\beta\|_0.$$

*The optimal parameters $\beta^\star$ for this problem are*

$$\beta^\star = \begin{bmatrix} 1.0673 \\ 1.0673 \end{bmatrix},$$

*with $f(\beta^\star) = 13.74$. However, if we run Algorithm 1 setting $\sigma_0 = 178.0219/3 = 59.34$, it stops at the end of the second iteration returning the first solution found:*

$$\bar{\beta} = \begin{bmatrix} 1.0989 \\ 0 \end{bmatrix},$$

*having value $f(\bar{\beta}) = 14.25 > f(\beta^\star)$.*

*Indeed, let $\beta_0$, $\beta_1 = \bar{\beta}$ and $\beta_2 = \beta^\star$ the solutions with $L_0$ norm respectively equal to 0, 1 and 2 minimizing $R(\beta) = \|Y - X\beta\|_2^2$. We have $R(\beta_0) = 300$, $R(\beta_1) = 178.022$ and $R(\beta_2) = 63.08$. With $\sigma_0 = 59.34$, the objective values of these solutions in the problem at step 3 of Algorithm 1 are respectively 5.056, 5 and 5.063: $\beta_1$ is therefore optimal; but then, the value of $\sigma_1$ is set to $R(\beta_1/3) = 59.34$, i.e. it does not change w.r.t. $\sigma_0$; thus, the first iteration is exactly repeated and at the end of it the stopping condition gets true, with the algorithm returning $\beta_1 = \bar{\beta}$.*

*It is interesting to note that the gap in terms of objective value is not so narrow in this case. However, this simple example does not match with the typical real-world problems where $P \ll N$.*

## 2.5 Alternate Minimization for Gaussian ARMA$(p,q)$ Models

In this Section, we properly modify the Alternate Minimization Algorithm to identify and estimate ARMA$(p,q)$ models (see Section 1.2). The identification of these models is substantiated in selecting the pair $(p,q)$ of ARMA orders, which measure the extent of temporal dependence of the autoregressive and moving average components.

As pointed out in (Broersen, 2006), ARMA order selection is a challenging task with a greater difficulty than pure AR or MA order selection. In fact, in both the latter cases, the order selection problem is equivalent to selection in a hierarchically nested class of candidate models, where each higher order model contains all parameters of lower order models. Conversely, in ARMA models there may be more than two models with the same complexity and this increases the difficulty of selection. The number of candidate ARMA$(p,q)$ models, with $p \leq L$ and $q \leq L$, is $L^2$.

An exhaustive penalized likelihood approach over all possible orders $(p,q)$ is proper when $L \leq 3$. In fact, the maximum likelihood estimation of many different ARMA models can be cumbersome especially when the length of the time series is more than 150 observations (Hyndman et al., 2008). On this matter, the state of the art approaches, are conceived to keep to a minimum the maximum likelihood steps, seeking to avoid the estimation of the most unlikely models.

In the following, we briefly review the two main methodologies to select ARMA models.

**Box and Jenkins** The Box and Jenkins procedure (Brockwell et al., 1991; Box et al., 2015) has represented for years the state of the art concerning the ARMA order selection problem. This methodology separates the order selection phase, based on the visual inspection of both the sample autocorrelation function (ACF) and the sample partial autocorrelation function (PACF) plots, from the next fitting phase. By matching both the sample autocorrelation plots with the theoretical autocorrelation plots, the modeler subjectively detects a small set of candidate models. ARMA models, which belong to this set, are estimated. The selected model is the one which minimizes a previous chosen information criterion, as the AIC, BIC or HQIC, among the candidate models. Eventually, the procedure ends with a diagnostic phase to validate the selected model.

**Hyndman-Khandakar Search** The algorithm of Hyndman-Khandakar (Hyndman et al., 2008) implements an heuristic strategy to traverse the ARMA order space efficiently. Once again, information criteria act as metric for order selection. In the first step of the procedure, a low order ARMA$(p,q)$ model, among a set of

four candidate ARMA$(p,q)$, is identified and denoted as the current model. At the second step, a new set of models, close to the current, is obtained by considering small variations of the orders of the current model. Whenever a model with lower information criterion value is found, it becomes the new current model and the procedure is repeated. The process finishes when we cannot find a model close to the current model with lower information criterion value. The function `auto.arima()` from the `R` package forecast (Hyndman et al., 2008) implements this heuristic, considering for selection also the Auto Regressive Integrated Moving Average (ARIMA) models and Seasonal Auto Regressive Integrated Moving Average (SARIMA) models (Brockwell et al., 1991; Hamilton, 1994; Box et al., 2015).

Especially with the increase of the upper bounds $(P,Q)$ of the orders, both these procedures have obvious complications. In fact, it becomes almost impossible for the Box and Jenkins procedure, like any other subjective model selection strategy, to select an ARMA model when the order space has a lot of candidates. Similarly, the Hyndman-Khandakar search may require several steps of maximum likelihood, hence becoming computationally non efficient.

Based on the latter considerations, the employment of Alternate Minimization in this scenario is motivated by the need of an automatic and efficient tool to detect ARMA models. However, in such scenario, as anticipated in Chapter 1, Alternate Minimization works as a preliminary estimation algorithm, returning a final fitted ARMA$(p,q)$ model that needs to be refined by a final maximum likelihood step. We adapt Alternate Minimization to the ARMA case as follows.

We consider, as a useful approximating model for a zero mean stationary time series $\{y_t\}_{t=1}^N$, the following Gaussian ARMA linear regression model:

$$Y_t = \phi_1 Y_{t-1} + \cdots + \phi_P Y_{t-P} + \theta_1 \hat{\epsilon}_{t-1} + \cdots + \theta_Q \hat{\epsilon}_{t-Q} + \epsilon_t, \quad \epsilon_t \sim \mathbb{N}(0,\sigma^2)\ , \tag{2.23}$$

where the error terms have been previously estimated as in the first step of the Hannan Rissanen methodology (Hannan and Rissanen, 1982) [3].

Now, the Gaussian log-likelihood $\ell(\phi,\theta,\sigma^2)$ of ARMA linear regression model (2.23) is:

$$\ell(\phi,\theta,\sigma^2) = -\frac{(N-m-Q)}{2}\log(2\pi) - \frac{(N-m-Q)}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\,\mathrm{SSR}(\phi,\theta), \tag{2.24}$$

with $\mathrm{SSR}(\phi,\theta) = \sum_{t=m+1+Q}^{N}(y_t - \phi_1 y_{t-1} - \cdots - \phi_P y_{t-P} - \theta_1 \hat{\epsilon}_{t-1} - \cdots - \theta_Q \hat{\epsilon}_{t-Q})^2$ representing the sum of squares loss function. The general optimization problem to

[3] Errors terms are estimated as residuals of a high-order preliminary AR$(m)$ model with $m > max(P,Q)$

be solved is the following:

$$\min_{\phi,\theta,\sigma^2} \quad -2\ell(\phi,\theta,\sigma^2) + \alpha(||\phi||_0 + ||\theta||_0), \tag{2.25}$$

where $\alpha > 0$ is a generic regularization term. Here again, by the choice of specific values of $\alpha$, we can perform the direct minimization of known information, which are commonly employed for order selection in time series, see (Stoica and Selen, 2004) for a nice review on this topic.

To properly identify the orders $(p,q)$ within a framework based on mixed integer optimization, we consider the hierarchical sparsity constraints for the autoregressive parameters $\phi = (\phi_1,\ldots,\phi_l,\ldots,\phi_P)$ and moving average parameters $\theta = (\theta_1,\ldots,\theta_l,\ldots,\theta_Q)$ (Liu and Tajbakhsh, 2020):

$$\text{if } \phi_l = 0 \text{ then } \phi_{l^*} = 0 \;\forall\, l < l^* \iff \text{if } \phi_{l^*} \neq 0 \text{ then } \phi_l \neq 0 \;\forall\, l < l^* \tag{2.26}$$

$$\text{if } \theta_l = 0 \text{ then } \theta_{l^*} = 0 \;\forall\, l < l^* \iff \text{if } \theta_{l^*} \neq 0 \text{ then } \theta_l \neq 0 \;\forall\, l < l^*. \tag{2.27}$$

As a consequence of these latter constraints, penalty terms on the model complexity are proportional to the sum of orders of the model.

At a given iteration $k$ of the algorithm, fixing the variance parameter $\sigma^2 = \sigma_k^2$, the first sub-problem is the following:

$$\min_{\phi,\theta} \quad \frac{\text{SSR}(\phi,\theta)}{\sigma_k^2} + g(\phi,\theta) \tag{2.28}$$

$$\text{s.t.:} \quad \phi \in \mathbb{R}^P,\ \theta \in \mathbb{R}^Q, \tag{2.29}$$

$$\phi \text{ satisfies } 2.26,\ \ \theta \text{ satisfies } 2.27, \tag{2.30}$$

$$g(\phi,\theta) = \alpha(||\phi||_0 + ||\theta||_0). \tag{2.31}$$

With the introduction of $P+Q$ binary variables $(\delta_1,\ldots,\delta_P,\nu_1,\ldots,\nu_Q)$, Problem (2.28)

can be reformulated as a mixed-integer convex quadratic problem:

$$\min_{\phi,\theta} \quad \mathrm{SSR}(\phi,\theta) + \sigma_k^2 \ (\alpha \sum_{i=1}^{P} \delta_i + \alpha \sum_{j=1}^{Q} \nu_j) \tag{2.32}$$

$$\text{s.t.:} \quad \delta_i \in \{0,1\} \qquad \forall i = 1,\dots,P \tag{2.33}$$

$$\nu_j \in \{0,1\} \qquad \forall j = 1,\dots,Q \tag{2.34}$$

$$\sum_{i=1}^{P} \delta_i \geq 1, \quad \sum_{j=1}^{Q} \nu_j \geq 1 \tag{2.35}$$

$$\sum_{i=1}^{m} \delta_i \geq m\delta_m \qquad \forall m = 1,\dots,P \tag{2.36}$$

$$\sum_{j=1}^{n} \nu_j \geq n\nu_n \qquad \forall n = 1,\dots,Q \tag{2.37}$$

$$-M\delta_i \leq \phi_i \leq M\delta_i \quad \forall i = 1,\dots,P \tag{2.38}$$

$$-M\nu_j \leq \theta_j \leq M\nu_j \quad \forall j = 1,\dots,Q \tag{2.39}$$

Especially when the ARMA model is employed in forecasting, its parameters are usually constrained in the causal-invertible space, i.e. $(\phi,\theta) \in S_p \times S_q$ (see Section 1.2). We manage these constraints by exploiting the parametrization of the ARMA parameters $(\phi,\theta)$ in terms of partial autocorrelations, also known as reflection coefficients. Levinson recursion $\mathrm{Y}(\cdot)$ maps reflection coefficients into ARMA parameters $(\phi,\theta)$ (see Section 3.2). Through $\mathrm{Y}(\cdot)$, the feasible sets $\mathrm{S}_p$ or $\mathrm{S}_q$, can be approximated within any degree of accuracy by the following closed sets (Combettes and Trussell, 1992):

$$\bar{\mathrm{S}}_p = \mathrm{Y}(\bar{K}_p) \quad \text{with} \ \ \bar{K}_p = [\delta - 1, 1 - \delta]^p \tag{2.40}$$

$$\bar{\mathrm{S}}_q = \mathrm{Y}(\bar{K}_q) \quad \text{with} \ \ \bar{K}_q = [\delta - 1, 1 - \delta]^q \tag{2.41}$$

Let $(\phi^{(k)},\theta^{(k)}) \in \mathbb{R}^P \times \mathbb{R}^Q$ be the solution returned by the algorithm at a given iteration $k$ and indicate with $(\hat{\phi}^{(k)},\hat{\theta}^{(k)}) \in \mathbb{R}^p \times \mathbb{R}^q$ the first $p$ non zero elements of $\phi^{(k)}$ and the first $q$ non zero elements of $\theta^{(k)}$. Feasibility is obtained by computing the projection of $\hat{\phi}^{(k)}$ (for the moving average is analogous) onto the closed feasible set $\bar{\mathrm{S}}_p$. The projection problem is approached in the reflection coefficients space (Combettes and Trussell, 1992):

$$\min_{k_p \in \bar{K}_p} \Phi(k_p) = \left(\mathrm{Y}(k_p) - \hat{\phi}^{(k)}\right)^T \left(\mathrm{Y}(k_p) - \hat{\phi}^{(k)}\right) \tag{2.42}$$

Note that due to the potential non convexity of $\bar{\mathrm{S}}_p$, the projection may not be unique and there is the possibility of convergence to local minima. However, the lack of convergence to a global optimum of Problem 2.42 is a rare eventuality (Combettes and

Trussell, 1992). Furthermore, since finding an exact projection by means of a global optimization algorithm could be expensive for a preliminary estimation algorithm, Problem 2.42 is solved by L-BFGS-B to handle box constraints. Observe that L-BFGS-B is a local optimization algorithm and the projection is not certified. We define three different variants of the algorithm (see algorithms 2, 3, 4). According to the first variant, the projection step (2.42) is performed at the end of each iteration; conversely, the second variant computes the projection when the iterations of the algorithm have finished. This variant is conceived to promote the computational efficiency. For these two variants, the whole procedure terminates with an exact maximum likelihood step, starting the optimization from the solution $(\hat{\phi}^{(k)}, \hat{\theta}^{(k)})$ provided by the algorithm. Finally, the third variant exactly matches the second variant but it performs maximum likelihood estimation of ARMA models belonging to a neighborhood $A$ of the returned ARMA$(p, q)$, i.e. $A = \{(p, q), (p-1, q), (p+1, q), (q-1, p), (q+1, p)\}$ (Local Search Variant). The final returned model is the one which minimizes the chosen information criterion among all the models in $A$.

**Algorithm 2** Alternate Minimization (First Variant)

**Input:** $\{y_t\}_{t=1}^N$, $P$, $Q$, $\phi^0$, $\theta^0$, $\sigma_0$, $k = 0$

1: fit a high-order AR($m$) to $\{y_t\}_{t=1}^N$
2: compute residuals $\{\hat{\epsilon}_t\}_{t=m+1}^N$ from the fitted AR($m$)
3: let $g(\phi^{-1}, \theta^{-1}) = \texttt{NaN}$
4: **while** $g(\phi^k, \theta^k) \neq g(\phi^{k-1}, \theta^{k-1})$ **do**
5: set

$$\phi^{k+1},\, \theta^{k+1} = \underset{\phi,\, \theta}{\arg\min}\, \frac{\mathrm{SSR}(\phi, \theta)}{{\sigma_k}^2} + g(\phi, \theta)$$

6: set

$$\hat{\phi}_{\mathrm{pro}}^{k+1} = \begin{cases} \hat{\phi}^{k+1}, & \text{if } \hat{\phi}^{k+1} \in \mathrm{S}_p \\ \underset{k_p \in \bar{K}_p}{\arg\min} \left(\mathrm{Y}(k_p) - \hat{\phi}^{k+1}\right)^T \left(\mathrm{Y}(k_p) - \hat{\phi}^{k+1}\right), & \text{otherwise} \end{cases}$$

7: set

$$\hat{\theta}_{\mathrm{pro}}^{k+1} = \begin{cases} \hat{\theta}^{k+1}, & \text{if } \hat{\theta}^{k+1} \in \mathrm{S}_q \\ \underset{k_q \in \bar{K}_q}{\arg\min} \left(\mathrm{Y}(k_q) - \hat{\theta}^{k+1}\right)^T \left(\mathrm{Y}(k_q) - \hat{\theta}^{k+1}\right), & \text{otherwise} \end{cases}$$

8: set

$$\sigma_{k+1}^2 = \frac{\mathrm{SSR}(\phi_{\mathrm{pro}}^{k+1}, \theta_{\mathrm{pro}}^{k+1})}{(n - m - Q)}$$

9: set $k = k + 1$
10: **end while**
11: set $p = ||\phi_{\mathrm{pro}}^k||_0, \quad q = ||\theta_{\mathrm{pro}}^k||_0$
12: fit by exact maximum likelihood an ARMA($p.q$) model to $\{y_t\}_{t=1}^N$
13: **return** $\phi^k$, $\theta^k$, $\sigma_k$

**Algorithm 3** Alternate Minimization (Second Variant)

**Input:** $\{y_t\}_{t=1}^N$, $P$, $Q$, $\phi^0$, $\theta^0$, $\sigma_0$, $k=0$

1: fit a high-order AR$(m)$ to $\{y_t\}_{t=1}^N$
2: compute residuals $\{\hat{\epsilon}_t\}_{t=m+1}^N$ from the fitted AR$(m)$
3: let $g(\phi^{-1},\theta^{-1}) = \texttt{NaN}$
4: **while** $g(\phi^k,\theta^k) \neq g(\phi^{k-1},\theta^{k-1})$ **do**
5: set

$$\phi^{k+1},\theta^{k+1} = \arg\min_{\phi,\theta} \frac{\mathrm{SSR}(\phi,\theta)}{\sigma_k^2} + g(\phi.\theta)$$

6: set

$$\sigma_{k+1}^2 = \frac{\mathrm{SSR}(\phi^{k+1},\theta^{k+1})}{(n-m-Q)}$$

7: set $k = k+1$
8: **end while**
9: set

$$\hat{\phi}_{\mathrm{pro}}^{k+1} = \begin{cases} \hat{\phi}^{k+1}, & \text{if } \hat{\phi}^{k+1} \in \mathrm{S}_p \\ \arg\min\limits_{k_p \in \bar{K}_p} \left(\mathrm{Y}(k_p) - \hat{\phi}^{k+1}\right)^T \left(\mathrm{Y}(k_p) - \hat{\phi}^{k+1}\right), & \text{otherwise} \end{cases}$$

10: set

$$\hat{\theta}_{\mathrm{pro}}^{k+1} = \begin{cases} \hat{\theta}^{k+1}, & \text{if } \hat{\theta}^{k+1} \in \mathrm{S}_q \\ \arg\min\limits_{k_q \in \bar{K}_q} \left(\mathrm{Y}(k_q) - \hat{\theta}^{k+1}\right)^T \left(\mathrm{Y}(k_q) - \hat{\theta}^{k+1}\right), & \text{otherwise} \end{cases}$$

11: set $p = ||\phi_{\mathrm{pro}}^k||_0$, $q = ||\theta_{\mathrm{pro}}^k||_0$
12: fit by exact maximum likelihood an ARMA$(p.q)$ model to $\{y_t\}_{t=1}^n$
13: **return** $\phi^k$, $\theta^k$, $\sigma_k$

**Algorithm 4** Alternate Minimization (Third Variant)

**Input:** $\{y_t\}_{t=1}^N$, $P$, $Q$, $\phi^0$, $\theta^0$, $\sigma_0$, $k = 0$

1: fit a high-order AR($m$) to $\{y_t\}_{t=1}^N$
2: compute residuals $\{\hat{\epsilon}_t\}_{t=m+1}^N$ from the fitted AR($m$)
3: let $g(\phi^{-1}, \theta^{-1}) = \texttt{NaN}$
4: **while** $g(\phi^k, \theta^k) \neq g(\phi^{k-1}, \theta^{k-1})$ **do**
5: set

$$\phi^{k+1}, \theta^{k+1} = \arg\min_{\phi,\theta} \frac{\mathrm{SSR}(\phi, \theta)}{\sigma_k^2} + g(\phi.\theta)$$

6: set

$$\sigma_{k+1}^2 = \frac{\mathrm{SSR}(\phi^{k+1}, \theta^{k+1})}{(n - m - Q)}$$

7: set $k = k + 1$
8: **end while**
9: set

$$\hat{\phi}_{\mathrm{pro}}^{k+1} = \begin{cases} \hat{\phi}^{k+1}, & \text{if } \hat{\phi}^{k+1} \in \mathrm{S}_p \\ \underset{k_p \in \bar{K}_p}{\arg\min} \left(\mathrm{Y}(k_p) - \hat{\phi}^{k+1}\right)^T \left(\mathrm{Y}(k_p) - \hat{\phi}^{k+1}\right), & \text{otherwise} \end{cases}$$

10: set

$$\hat{\theta}_{\mathrm{pro}}^{k+1} = \begin{cases} \hat{\theta}^{k+1}, & \text{if } \hat{\theta}^{k+1} \in \mathrm{S}_q \\ \underset{k_q \in \bar{K}_q}{\arg\min} \left(\mathrm{Y}(k_q) - \hat{\theta}^{k+1}\right)^T \left(\mathrm{Y}(k_q) - \hat{\theta}^{k+1}\right), & \text{otherwise} \end{cases}$$

11: set $p = ||\phi_{\mathrm{pro}}^k||_0$, $q = ||\theta_{\mathrm{pro}}^k||_0$
12: define $A = \{(p, q), (p - 1, q), (p + 1, q), (q - 1, p), (q + 1, p)\}$
13: fit by exact maximum likelihood all the feasible ARMA models with orders $(p^*, q^*) \in A$
14: **return** $\phi^k$, $\theta^k$, $\sigma_k$ from the best fitted model

## 2.6 Experiments: Gaussian Linear Regression

Concerning Gaussian linear regression, similarly as in previous works on the topic (Miyashiro and Takano, 2015a; Kimura and Waki, 2018), our benchmark is made up of eight datasets from the UCI Machine Learning Repository (Dua and Graff, 2017). Table 2.1 synthetically describes these datasets, showing the number $N$ of points and the number $P$ of variables. The Solar Flare instance has three target variables (C, M and X) and therefore three instances of the problem have been prepared.

| Dataset | $N$ | $P$ |
|---|---|---|
| Housing | 506 | 13 |
| Servo | 167 | 19 |
| Auto MPG | 392 | 25 |
| Solar Flare C | 1066 | 27 |
| Solar Flare M | 1066 | 27 |
| Solar Flare X | 1066 | 27 |
| Breast Cancer Wisconsin | 194 | 33 |
| Forest Fires | 517 | 63 |
| Automobile | 159 | 65 |
| Communities and Crime | 1993 | 102 |

Table 2.1: List of datasets for experiments on subset selection for linear regression.

As for the Forest Fires dataset, interaction terms between $x$ and $y$ spatial coordinates have been created. For each dataset, we performed the one-hot encoding of the categorical variables and we normalized the other ones to zero mean and unit standard deviation, in order to prevent numerical issues. Moreover, data points with missing variables haves been removed.

Minimization of AIC, BIC and HQIC for the 10 problems has been carried out. With all the considered methods in the experiments, Gurobi 8.1.0 (Gurobi Optimization LLC, 2018) was employed as the quadratic programming solver for MINLP problems and subproblems. Indeed, all of the subproblems we are solving are MIQP. Although still complex in general, these models can be successfully solved in the convex case through general purpose solvers like Gurobi if the dimension is not excessively large. In order to enhance the efficiency of Gurobi, continuous variables were constrained to belong to the interval $[-10^3, 10^3]$. Being the datasets normalized, this in practice does not represent a restriction to the model. In order to prevent numerical problems with integer variables, we set the integer precision parameter to $10^{-9}$, which is the most accurate possible value with Gurobi. The value of $M$ for big$M$-type constraints was set to $10^4$.

All experiments were performed on a machine with Ubuntu Server 16.04 LTS OS, Intel Xeon E5-2430 v2 @ 2.50GHz CPU and 16GB RAM. Every run of each algorithm was performed on a single CPU core, in order to make time measurements fair, and a time limit of 10 000 seconds was set.

In the following, we will make use of performance profiles (Dolan and Moré, 2002) for comparing the performance of different algorithms on a set of benchmark problems. We recall that, in performance profiles, each curve represents the cumulative fraction of problems the corresponding solver could solve in a time which is at most a factor of $\tau$ worse than the best performing one. Separate analyses are performed for each different information criterion.

We compared several solvers on the subset selection for linear regression problems generated from datasets in Table 2.1. We considered Alternate Minimization algorithm (Algorithm 1), the step-wise heuristic with Forward selection and Backward elimination strategies, the exhaustive approach, the MISOCP model (2.7), the Newton's method to solve problem 2.9 (MIFO-Newton) and finally the method of solving the MIQP model (2.4) for all possible values of $k$.

Alternate Minimization, MIFO-Newton and the iterated MIQP algorithms employ mixed-integer solvers (Gurobi) as subroutines. With the largest problems (those generated from Automobile and Communities and Crime datasets) we had to set an inner time limit of 1 200 seconds for each iteration, since otherwise Gurobi never stops, failing at certificating the optimality of the current solution of the subproblem. This slight modification clearly spoils the theoretical properties of the algorithms, but in practice leads to good performance both in terms of runtime and quality of the solutions.

A warm-start strategy also speeds up algorithms Alternate Minimization and MIFO-Newton. The solution found at the $k$-th iteration is feasible (and likely good) for the $k+1$-th problem; using that solution as starting point at the $k+1$-th iteration provides some reduction in the computing time. On the other hand, the addition of redundant constraints based on normal equations, as outlined in Gómez and Prokopyev (2018), greatly improves the performance of MISOCP model. The same addition, on the contrary, turned out to be quite useless with the other mixed-integer models.

| Method | # successful runs | total time (sec) |
|---|---|---|
| **AM** | 29/30 | 31299 |
| BW Stepwise | 18/30 | 38666 |
| Exhaustive | 17/30 | 267405 |
| FW Stepwise | 17/30 | 3530 |
| MIFO-Newton | 29/30 | 32759 |
| MIQP | 25/30 | 155295 |
| MISOCP | 22/30 | 172215 |

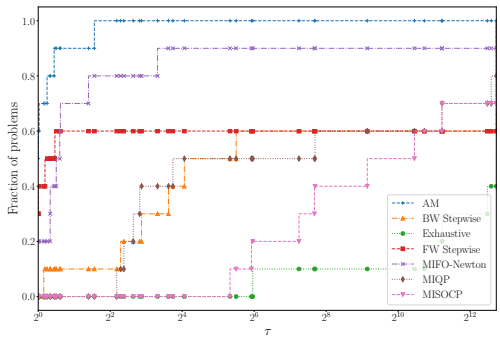
Table 2.2: For each considered algorithm, the number of times the optimal solution was found and the sum of runtimes is reported, out of the 30 linear regression problems (10 datasets with AIC/BIC/HQIC). A solution is considered optimal if the relative distance to the best overall objective value is lower than $10^{-3}$.

In Table 2.2 we show the overall performance of the algorithms. We can see that the heuristics are fast, especially the forward selection, but often lead to suboptimal solutions. The exhaustive search is very slow, and often exceeds the time limit, returning bad solutions. The MISOCP model has a slightly better behavior, but with similar shortcomings. Our proposed Alternate Minimization algorithm proved to

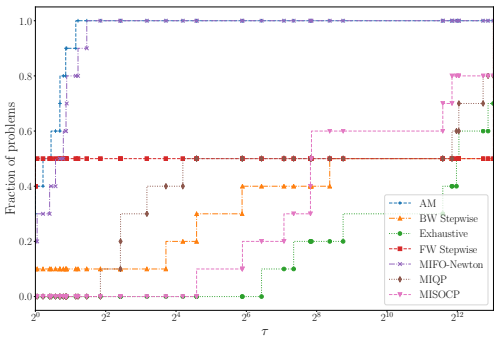
be the best choice, as it is the second one in terms of CPU time while being the top ranking in terms of the number of solved problems; like MIFO-Newton, it only fails once at finding an optimal solution, but it requires a smaller time to run on all the problems.

Figures 2.1 and 2.2 give a wider insight of results. In Figure 2.1 the performance profiles, in terms of runtime, of the seven considered algorithms on the ten regression problems are shown. Note that we considered the runtime to be "infinite" when the returned solution is suboptimal. We made up separate profiles for AIC (2.1a), BIC (2.1b) and HQIC (2.1c). We can observe that the profile of AM is almost completely dominant with respect to the others. The only comparable algorithm is MIFO-Newton, but its performance curves are always under those of AM, except for the final part of the HQIC scenario, where AM cannot find the optimal solution of one problem. The curves of all other methods considered are far below that of AM.

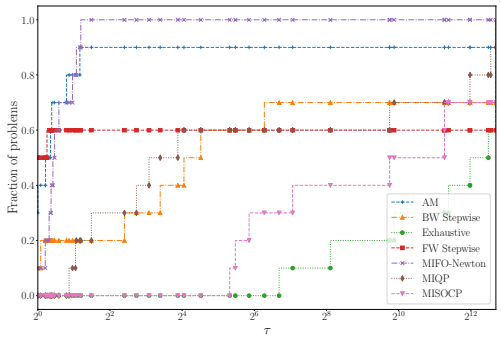
As for the quality of the returned solutions, we report in Figure 2.2 the cumulative distribution of the relative errors, in terms of objective value, attained by the various solvers. This plot confirms that not only AM is the fastest of the considered algorithms, but it is also, along with MIFO-Newton, the most accurate one.




(a) AIC




(b) BIC




(c) HQIC

Figure 2.1: Performance profiles of runtimes of different algorithms for subset selection in 10 linear regression problems. The problems are generated from datasets in Table 2.1. Each figure refers to one of the considered information criteria (AIC, BIC and HQIC).

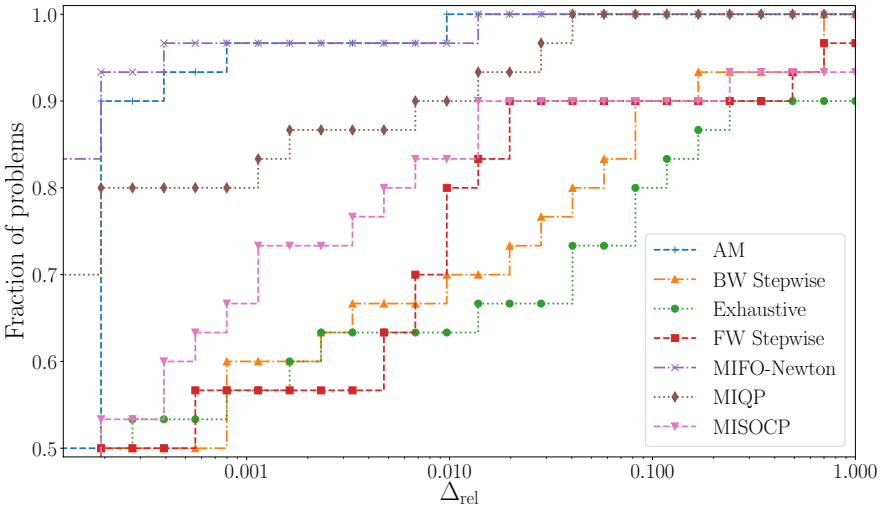



Figure 2.2: Each curve represents the fraction of the 30 linear regression problems for which the corresponding solver obtains a relative error less or equal than $\Delta_{\text{rel}}$ w.r.t. the optimal value.

## 2.7 Experiments: Gaussian ARMA models

W.r.t. Gaussian ARMA models, all the three variants of the Alternate Minimization algorithm and the heuristic search of Hyndman-Khandakar, revised just for the case of pure ARMA models, have been considered for the experiments.

The experiments have been carried out on a synthetic dataset of zero mean Gaussian ARMA time series. We simulated time series from nine different scenarios (see Table 2.3), any one of which is identified by the standard deviation $\sigma$ of the input white noise generator process and the length of the simulated series. A total of one

| Scenario | Length | $\sigma$ | Candidate models |
|---|---|---|---|
| I | 300 | 0.1 | 25 |
| II | 1000 | 0.1 | 25 |
| III | 10000 | 0.1 | 25 |
| IV | 300 | 1 | 25 |
| V | 1000 | 1 | 25 |
| VI | 10000 | 1 | 25 |
| VII | 300 | 5 | 25 |
| VIII | 1000 | 5 | 25 |
| IX | 10000 | 5 | 25 |

Table 2.3: Scenarios of the experiments: each scenario is identified by the standard deviation $\sigma$ of the noise process and length of the simulated series.

hundred time series have been simulated for each scenario, where each series represents a single realization of a different Gaussian ARMA$(p, q)$. We start the simulation of each series by generating the white noise error series. Then, we set the orders $(p, q)$ of each generator process by uniformly sampling orders up to a maximum of $(5, 5)$ [4]. Fixed the order, we choose the associated structural parameters $(\phi, \theta)$ of the process from a uniform distribution over the causality and invertibility region (Jones, 1987). Finally, each series is generated recursively according to the ARMA equations.

We assessed the algorithms both in terms of the accuracy of finding the true generator model and computational efficiency. In this case, the selection of models is based only on the BIC criterion: in fact, in such settings, when the true generator model belongs to the set of candidate models, the BIC criterion satisfies the consistency property and surely represents an appropriate choice.

Accuracy results are reported in Table 2.4. We note that none of the methods, involved in the experiments, prevails over the others neither varying the length of

[4]The choice of setting $(5, 5)$ as the upper bounds $(P, Q)$ of the ARMA order is coherent with the default settings of the `auto.arima()` function from the `R` package `forecast` (Hyndman et al., 2008).

| | (0.1,300) | (0.1,1000) | (0.1,10000) | (1,300) | (1,1000) | (1,10000) | (5,300) | (5,1000) | (5,10000) |
|---|---|---|---|---|---|---|---|---|---|
| First Variant | 0.33 | 0.53 | 0.65 | 0.33 | 0.46 | 0.66 | 0.27 | 0.49 | 0.67 |
| Second Variant | 0.33 | 0.39 | 0.71 | 0.28 | 0.49 | 0.68 | 0.27 | 0.44 | 0.66 |
| Third Variant | 0.34 | 0.42 | 0.65 | 0.33 | 0.41 | 0.66 | 0.30 | 0.55 | 0.63 |
| Heuristic | 0.33 | 0.43 | 0.66 | 0.26 | 0.43 | 0.67 | 0.30 | 0.54 | 0.68 |

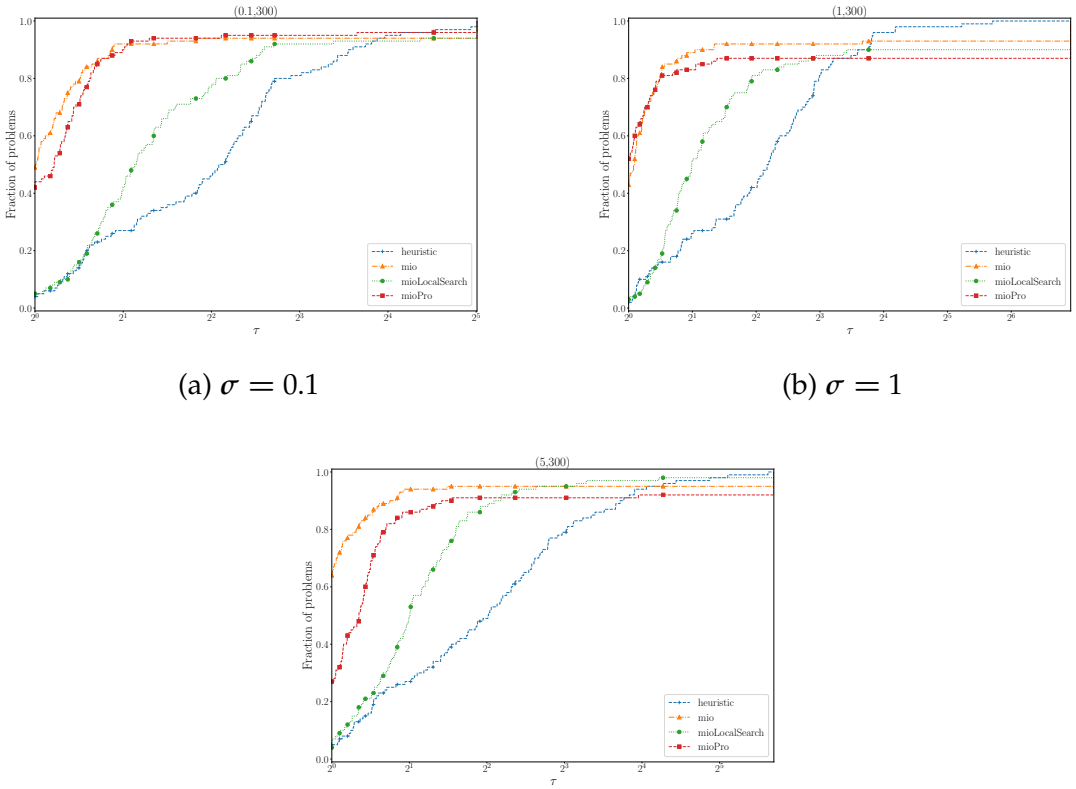
Table 2.4: Accuracy of finding the true generator ARMA$(p, q)$ model obtained by different algorithms. We compare all the three variants of the proposed method and the heuristic approach of Hyndman and Khandakar. Results are reported for each considered scenario. BIC criterion has been employed for order selection.

the series nor varying the standard deviation of the error process. We also observe that the quality of results increase, similarly for each algorithm, with the length of the series, pointing out the consistency property of BIC criterion.

The attention is then focused on the computational efficiency of Alternate Minimization. We aim to measure the computational efficiency of the mixed integer optimization strategy, in a default setting scenario consisting of twenty-five candidate ARMA model. Although in such settings a enumerative maximum likelihood approach is also feasible, we compare again with the heuristic approach of Hyndman and Khandakar. In Figures 2.3, 2.4, 2.5, we report the performance profiles, in terms of run time, of the four involved algorithms for each considered scenario.

By observing Figures 2.3, 2.4, 2.5, we note that the orange, the red and green curves, respectively referring to Algorithms 3, 2, 4, systematically lie above the green line which refers to the heuristic strategy. Hence, we can conclude that in a realistic scenario, the employment of Alternate Minimization is also adequate in terms of runtimes. Algorithm 3, which performs only a single projection step and a single exact likelihood step, is the most efficient Alternate Minimization variant, saving computational times. Instead, Algorithms 2, 4, although they respectively perform a larger number of projection-maximum likelihood steps and are slower than Algorithm 3, do not have higher accuracy as it is highlighted in Table 2.4.

Taking into account both the results of accuracy in Table 2.4 and the performance profiles plots 2.3, 2.4, 2.5, we conclude that the simple MIO ARMA variant of Alternate Minimization represents the best option to fit efficiently ARMA models without any subjective identification and in a totally *data-driven* manner.

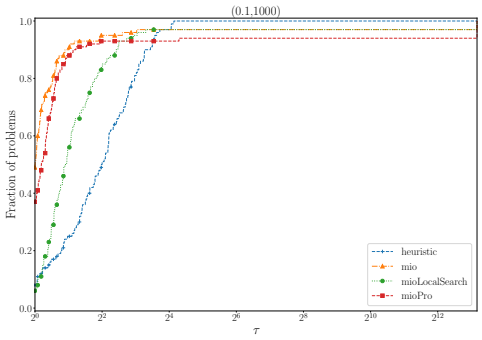



(a) $\sigma = 0.1$

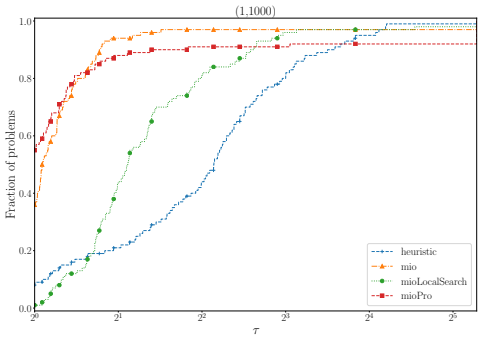
(b) $\sigma = 1$

(c) $\sigma = 5$

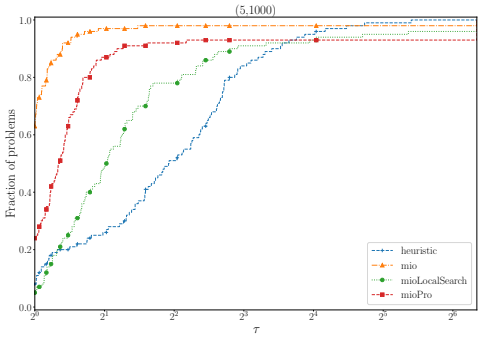
Figure 2.3: Performances profiles of runtimes for time series of 300 observations varying the standard deviations of the error process. The orange line refers to the Second Variant of our method. The red line refers to the First Variant of our method. The green line refers to the third variant of our method. The blue line refers to the heuristic of Hyndman-Khandakar.




(a) $\sigma = 0.1$

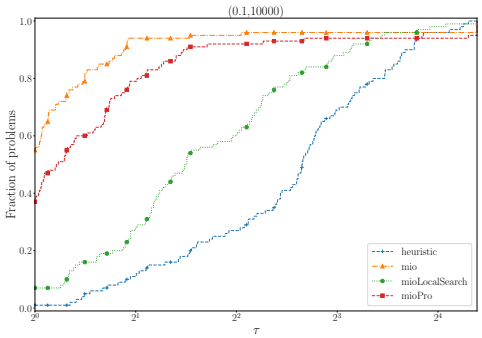



(b) $\sigma = 1$

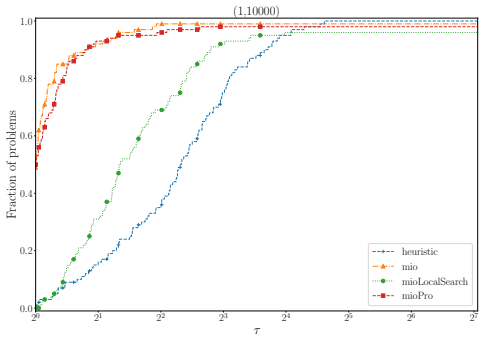



(c) $\sigma = 5$

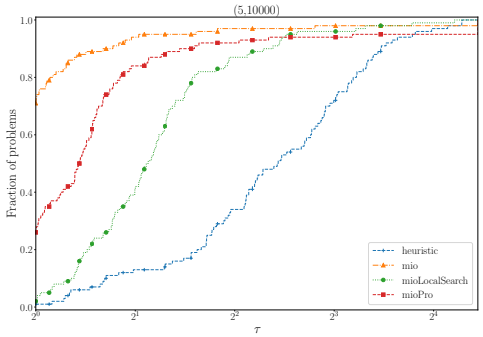
Figure 2.4: Performances profiles of runtimes for time series of 1000 observations varying the standard deviations of the error process. The orange line refers to the Second Variant of our method. The red line refers to the First Variant of our method. The green line refers to the third variant (Local Search Variant) of our method. The blue line refers to the heuristic of Hyndman-Khandakar.




(a) $\sigma = 0.1$




(b) $\sigma = 1$




(c) $\sigma = 5$

Figure 2.5: Performances profiles of runtimes for time series of 10000 observations varying the standard deviations of the error process. The orange line refers to the Second Variant of our method. The red line refers to the First Variant of our method. The green line refers to the third variant (Local Search Variant) of our method. The blue line refers to the heuristic of Hyndman-Khandakar.

## 2.8 Conclusions

We developed, theoretically analyzed and numerically experimented a decomposition approach, which rely on the solution of mixed integer quadratic programming sub-problems, to solve the best subset selection problem in Gaussian linear regression and the order selection problem in Gaussian ARMA models.

The proposed methodology directly minimizes known information criteria as the AIC, BIC and HQIC. The employment of these criteria is particularly useful, since the modeler has not to specify in advance the number of predictors to be selected and the whole methodology is as far as possible *data-driven*.

In the ARMA scenario, Alternate minimization is modified inducing a hierarchical sparsity structure that allows to properly identify the ARMA orders $(p, q)$. Furthermore, unlike in the linear regression case, the algorithm finishes with at least one ARMA maximum likelihood step in order to refine the solution provided by the algorithm.

Our experiments highlight that the approach is capable of delivering high quality solutions in very small CPU time in both the scenarios, although the optimality of the final solution is not guaranteed.

# Chapter 3

# Improved Maximum Likelihood Estimation of ARMA Models

A bound-constrained optimization model enables to avoid the employment of the Jones reparametrization (Jones, 1980), which is the classical method to deal with the causal and invertible conditions. The performed experiments highlight that the use of this reparametrization, although it allows to formulate the estimation problem as an unconstrained optimization problem, leads to higher computational times and numerical instability. We also show how the addition of a $\ell_2$-regularization term in our formulation improves the out of sample quality of the fitted model. This improvement is achieved thanks to an increased penalty on models close to the non-causality or non-invertibility boundary.

The rest of the chapter is organized as follows. In Section 3.1 the main historical steps of ARMA estimation problem are reported. Section 3.2 contains a review of the Jones reparametrization method. In Section 3.3 the notion of closeness of structural ARMA parameters $(\phi, \theta)$ to the feasibility boundary is defined. In Section 3.4 the bound constrained maximum likelihood estimation approach is introduced. The carried out computational experiments, which assess the reliability of the proposed method, are outlined in Section 3.5. Finally, we finish with conclusions in Section 3.6.

## 3.1 Estimation of ARMA models: a historical parentheses

Since the direct evaluation of the Gaussian exact log-likelihood function (1.9) is expensive both for storage and computational reasons, approximating procedures have been proposed in the 1970s to estimate the ARMA parameters.

Rewriting the zero mean ARMA process (1.7) as

$$\epsilon_t = Y_t - \phi_1 Y_{t-1} - \ldots - \phi_p Y_{t-p} - \theta_1 \epsilon_{t-1} - \ldots - \theta_q \epsilon_{t-q}, \quad t = 1, \ldots, N,$$

and setting the unknown pre-sample $\epsilon_t$ and $\epsilon_1, \ldots, \epsilon_p$ errors equal to 0, i.e. their unconditional expected value, parameters $(\phi, \theta)$ are estimated by minimizing $\sum_{t=p+1}^{N} \epsilon_t^2$. This estimation procedure is known as Conditional Least Squares (CLS) method (Dent and Min, 1978).

Alternatively, it is possible to employ the backcasting method of Box and Jenkins to obtain the estimates of the pre-sample errors and observations (Box et al., 2015). Once these quantities are obtained, Unconditional Least Square (ULS) estimates of parameters $(\phi, \theta)$ are found by minimizing $\sum_{t=-L}^{N} \epsilon_t^2$, for some large $L$. In (Newbold, 1974) it is pointed out that the approximation of the exact likelihood by ULS method can have difficulties when the series of available data is relatively short and when a root of either the autoregressive (1.11) or moving-average polynomials (1.12) lies close to the unit circle.

Newbold (1974) derived the ARMA exact likelihood using a generalization of the approach used by Box and Jenkins for pure moving average processes. Ansley (1979) computes the exact likelihood by transforming the process to obtain a band covariance matrix whose Cholesky decomposition can be readily computed. Finally, in the late 1970's, the theoretical and especially computational advantages of computing the exact likelihood by means of the Kalman Filter have been pointed out in (Harvey and Phillips, 1979). Kalman Filter based methods are considered the most efficient in terms of the number of operations involved to compute the likelihood (Mauricio, 2002). To date, Kalman Filtering represents the state-of-the-art of the methods employed to compute the exact likelihood for any given choice of the parameters $(\phi, \theta)$.

To run the the Kalman filter recursions, the ARMA process needs to be cast into the state space form. There is not a unique way to represent an ARMA process in this form. Common ARMA state space representations are the ones reported in (Harvey, 1990; Hamilton, 1994; Akaike, 1998).

## 3.2 Jones reparametrization

When causality and invertibility conditions hold, parameters $\phi = (\phi_1, \ldots, \phi_p)$ and $\theta = (\theta_1, \ldots, \theta_q)$ are constrained to belong to the set $S_p \times S_q$ corresponding to the polynomial operator root conditions (1.11), (1.12).
$S_k$ is easily identified for $k \leq 2$, but for $k > 2$ its form becomes complicated and for $k > 4$ Equations (1.11), (1.12) cannot be solved analytically (Marriott, 1995). To circumvent the problem of dealing with conditions (1.11) and (1.12) (Barndorff-Nielsen

and Schou, 1973) reparametrize $\phi = (\phi_1, \ldots, \phi_p)$ in terms of the partial autocorrelations $\rho = (\rho_1, \ldots, \rho_p)$ by means of the one-to-one continously differentiable Levinson mapping $\Upsilon(\cdot)$:

$$\begin{aligned} \phi_k^{(k)} &= \rho_k, \quad k = 1, \ldots, p, \\ \phi_i^{(k)} &= \phi_i^{(k-1)} - \rho_k \phi_{k-i}^{(k-1)}, \quad i = 1, \ldots, k-1. \end{aligned} \tag{3.1}$$

In (3.1), causality is simply obtained by $\rho_k \in (-1, 1) \ \ \forall k = 1, \ldots, p$. (Jones, 1980) introduces an additional mapping $J : \mathbb{R}^p \to (-1, 1)^p$, which allows to formulate the original problem as an unconstrained optimization problem introducing variables $u_k, \ k = 1, \ldots, p$:

$$\rho_k = \frac{1 - \exp(-u_k)}{1 + \exp(-u_k)}, \quad k = 1, \ldots, p. \tag{3.2}$$

Similar transformations can also be employed for the moving average parameters $\theta = (\theta_1, \ldots, \theta_q)$ in order to guarantee the invertibility condition. By writing the moving average polynomial (1.14) for the negative vector of MA parameters, $-\theta$, we get

$$\Theta(z) = 1 - (-\theta_1)z - \ldots - (-\theta_q)z^q, \tag{3.3}$$

and the following can be deduced:

$$\begin{aligned} \theta_k^{(k)} &= b_k, \quad k = 1, \ldots, q, \\ \theta_i^{(k)} &= \theta_i^{(k-1)} + b_k \theta_{k-i}^{(k-1)}, \quad i = 1, \ldots, k-1, \end{aligned} \tag{3.4}$$

where the variables $b_k \in (-1, 1) \ \ \forall k = 1, \ldots, q$. Jones reparametrization for the moving average part is equivalent to (3.2):

$$b_k = \frac{1 - \exp(-w_k)}{1 + \exp(-w_k)}, \quad k = 1, \ldots, q. \tag{3.5}$$

In (Jones, 1980), the variables $b_k$ are called partial moving average coefficients. The optimization of the exact log-likelihood in the causal and invertible feasible space is now carried out with respect to the variables $u = (u_1, \ldots, u_p) \in \mathbb{R}^p$ and $w = (w_1, \ldots, w_q) \in \mathbb{R}^q$.

Note that $\phi = \Upsilon(\rho)$, while $\theta = -\Upsilon(b)$. In fact, for any $u$ and $w$, the evaluation of the exact likelihood function in a causal and invertible feasible point can be computed by means of the transformations (3.1), (3.2), (3.4), (3.5), and the Kalman recursions. Inverse Jones transformations are easily found by solving (3.2), (3.5) respectively for $u_k, \ k = 1, \ldots, p$ and $w_k, \ k = 1, \ldots, q$. On the other hand, Monahan (1984) derives the expression of the inverse transformation $\Upsilon^{-1}(\cdot)$ of (3.1) which equivalently can be extended for the moving average part (3.4).

## 3.3 Closeness to the Feasiblity Boundary

In this Section, the notion of closeness of a feasible point $(\phi, \theta) \in S_p \times S_q$ to the set $\partial S_p \times \partial S_q$, i.e. the boundary of the invertibility and causality regions, is formalized. This will be useful later in the chapter, when investigating the relation between the closeness to the boundary and the numerical stability during the optimization of the Gaussian ARMA exact log-likelihood function.

It is partially documented [1] that log-likelihood evaluation by Kalman filter may fail when a point $(\phi, \theta)$ is close to the causality boundary. Furthermore, it is well known that closeness to the non-invertible region is problematic due to the presence of the so-called pile-up effect (Kang, 1975; Sargan and Bhargava, 1983; Kim and Kim, 2013). Indeed, when the true parameter of an MA$(1)$ process is close to unity, the model can be estimated to be non-invertible with a unit root even when the true process is invertible, with a considerably high probability in a finite sample. Ansley and Newbold (1980) confirm the presence of such effect in ARMA models too.

Inspired by the method of Zhang and McLeod (2006) for testing the presence of a parameter estimate on the boundary of an MA$(q)$ model, the closeness of a point $(\phi, \theta)$ to the boundary of the invertible and the causal-stationary regions is defined exploiting the parametrization of an ARMA$(p, q)$ in terms of $\rho$ and $b$:

$$
\begin{aligned}
&(\phi, \theta) = (\mathrm{Y}(\rho), -\mathrm{Y}(b)) \,, \\
&(\phi, \theta) \in S_p \times S_q \iff (\rho, b) \in (-1, 1)^p \times (-1, 1)^q .
\end{aligned}
$$

$\mathrm{Y}(\cdot)$ is not one-to-one on the hypercube boundary (Barndorff-Nielsen and Schou, 1973). However, as elegantly shown in (Zhang and McLeod, 2006), $\mathrm{Y}(\cdot)$ maps the boundary of $(-1, 1)^p$ onto $\partial S_p$. Since $\mathrm{Y}(\cdot)$ is a continuously differentiable function in $[-1, 1]^p$, the closeness of an estimate $\phi \in S_p$ to the non-causal-stationary boundary $\partial S_p$ can be defined respectively in terms of the partial autocorrelations $\rho$. The same reasoning holds for the moving average part.

As reported in (Zhang and McLeod, 2006), $\phi \in \partial S_p$ if and only if $\|\rho\|_\infty = 1$ and similarly $\theta \in \partial S_q$ if and only if $\|b\|_\infty = 1$. Now, by fixing a threshold parameter $\tau > 0$, closeness of $(\phi, \theta) = (\mathrm{Y}(\rho), -\mathrm{Y}(b)) \in S_p \times S_q$ to the boundary $\partial S_p \times \partial S_q$ is defined by the following:

(i) $(\phi, \theta) \in S_p \times S_q$ is close to $\partial S_p$ if and only if $1 - \|\rho\|_\infty < \tau$;

(ii) $(\phi, \theta) \in S_p \times S_q$ is close to $\partial S_q$ if and only if $1 - \|b\|_\infty < \tau$;

(iii) $(\phi, \theta) \in S_p \times S_q$ is close to both $\partial S_p$ and $\partial S_q$ if and only if $1 - \|\rho\|_\infty < \tau$ and $1 - \|b\|_\infty < \tau$.

[1] see, e.g., `https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/KalmanLike`

A point $(\phi, \theta) \in S_p \times S_q$ which does not satisfy any of the above conditions i, ii, iii is defined as a strictly feasible point of $S_p \times S_q$.

## 3.4 The Proposed Approach

It is proposed to fit causal and invertible ARMA$(p, q)$ models by solving the following bound constrained optimization problem:

$$\begin{aligned} &\max_{\rho, b, \sigma^2} \ell\left(\mathrm{Y}(\rho), -\mathrm{Y}(b), \sigma^2\right) \\ &\text{s.t. } \rho \in [-1+\varepsilon, 1-\varepsilon]^p, \quad b \in [-1+\varepsilon, 1-\varepsilon]^q, \quad \sigma \in \mathbb{R}_+. \end{aligned} \tag{3.6}$$

Optimizing w.r.t. the partial autocorrelation and the partial moving average coefficients avoids the use of the Jones reparametrization (3.2), (3.5). Note that this formulation cuts off a small part of the feasible space $S_p \times S_q$. However, as highlighted by thorough numerical experiments that we will describe in the following Section, our formulation provides some nice advantages:

- it allows to save a significant amount of running time, as there is no more the need to compute equations (3.2) and (3.5) $p$ and $q$ times respectively, each time the log-likelihood has to be computed during the optimization process (note that every gradient computation by finite differences requires $2(p+q)$ objective evaluations);
- it allows to avoid solutions too close to the feasibility boundary that typically lead to numerical errors.

A Tikhonov regularization term is also included in the objective function of Problem (3.6):

$$\begin{aligned} &\max_{\rho, b, \sigma^2} \ell\left(\mathrm{Y}(\rho), -\mathrm{Y}(b), \sigma^2\right) - \lambda(||\rho||_2^2 + ||b||_2^2) \\ &\text{s.t. } \rho \in [-1+\varepsilon, 1-\varepsilon]^p, \quad b \in [-1+\varepsilon, 1-\varepsilon]^q, \quad \sigma \in \mathbb{R}_+. \end{aligned} \tag{3.7}$$

In the following, it is experimentally shown that this term not only discourages solutions close to the feasibility boundary, but it also improves the predictive ability of ARMA models.

## 3.5 Computational Experiments

In what follows the approximation parameter $\varepsilon$ is set to $10^{-2}$; the closeness parameter in i, ii, iii is fixed to $\tau = 2\varepsilon$, so that it is still possible for models (3.6) and (3.7) to produce points that are close to the border of the original feasible set.

All the experiments have been performed on a dataset of synthetically generated time series. We simulated a total of 2250 time series of different length $l \in \{100, 1000, 10000\}$ from ARMA $(p, q)$ Gaussian processes up to a maximum order $(p, q)$ of $(5, 5)$ and standard deviation $\sigma \in \{0.01, 0.1, 1\}$.

Specifically, for a given combination of length, order and standard deviation, we generated 10 time series, each one representing a finite realization of a particular ARMA process with its structural autoregressive and moving average parameters $(\phi, \theta)$. Each pair $(\phi, \theta)$ is selected according to the methodology described in (Jones, 1987). This methodology allows to choose $(\phi, \theta)$ from a uniform distribution over the feasible set $S_p \times S_q$.

Firstly, the interest lies in establishing the differences between solving problem (3.6) and the unconstrained one, based on Jones reparametrization, both from the standpoints of computational times and numerical stability. To this aim, a multi-start strategy has been employed: for each time series, the fitting process is repeated 30 times from different randomly chosen starting points. These starting points are again obtained by uniform sampling over the feasible region. For a fair comparison, the two considered methods share the sets of starting points.

Secondly, the predictive performance of ARMA models close to the boundary are investigated. As usual, the performance is evaluated on a test set, after fitting on training data. Our test set for each time series is given by the last three observations (short term forecasting scenario). Similarly as above, the process of model estimation and computation of forecasts is repeated 30 times in a multi-start fashion. Note that, here, ARMA models have been fitted only by means of the classical Jones methodology. Indeed, the aim is to characterize both the forecasting performance of ARMA models close to the border and how frequently they are obtained in the standard setting.

The last experiment assesses the impact of the $\ell_2$ regularization term in the short term forecasting scenario. For each time series of our dataset, a single starting point to initialize the optimization is selected. The fitting procedure is then repeated for different values of the regularization hyperparameter $\lambda$ in Equation 3.7.

All the experiments were performed on a machine with Ubuntu Server 20.04 LTS OS, Intel Xeon E5-2430 v2 @ 2.50GHz CPU and 32GB RAM.

## Fitting Procedure Runtimes

Our method provides a significant reduction of the computational time required to fit a time series with respect to the unconstrained fitting method of Jones. The time saving is estimated to be about 24% in relative terms.

This result is supported by the non parametric Wilcoxon signed-ranks test (Wilcoxon, 1992; Demšar, 2006). We considered as fitting time for a time series the average runtime of successful runs (i.e., with no numerical error) of our multi-start procedure.

Results of the Wilcoxon signed-ranks test are reported in Tables 3.1a and 3.1b. These results point out that the median of the differences of fitting times between the two methods can be assumed to be positive, i.e., the constrained method has significantly lower fitting times, considering a significance level $\alpha = 0.05$.

| Test statistic | P-value |
|---|---|
| -34.3807 | $< 1e{-}5$ |

(a) Two sided Wilcoxon signed-rank test

| Test statistic | P-value |
|---|---|
| 34.3807 | $< 1e{-}5$ |

(b) One sided Wilcoxon signed-rank test

Table 3.1: Null hypothesis for the two sided Wilcoxon signed-rank test: the median of the differences of the computational times $t_{\text{Jones}} - t_{\text{our}}$ is zero. Null hypothesis for the one sided Wilcoxon signed-rank test: the median of the differences of computational times $t_{\text{Jones}} - t_{\text{our}}$ is negative.

## Numerical Instability

Our fitting method prevents numerical issues during the optimization process of the ARMA exact likelihood function, thereby ensuring a higher level of computational stability.

| Method | Arithmetic issues | Kalman Filter errors |
|---|---|---|
| Our | 0 | 0.06 |
| Jones reparametrization | 2.65 | 0.22 |

Table 3.2: Occurrence of numerical instability issues per 1000 runs

The employment of the Jones reparametrization, where exponential operators are present, leads to a non-negligible probability of arithmetic issues, which almost always are divisions by zero and in rare cases overflows. Our method does not suffer at all from these issues.

The most critical errors, that completely undermine the fitting process, come from the Kalman Filter recursions. In general, it is well known that numerical instability often occurs in Kalman Filtering (Tusell et al., 2011), especially related to the computation of the state covariance matrix.

Experiments show that the closeness of a point $(\phi, \theta)$ to the feasibility boundary is related to numerical instability within the Kalman Filter recursions. In particular, we observed a total of 19 `LinAlgError` errors (15 by the classical method, 4 by using our model (3.6)) because of a convergence failure of the SVD numerical computation.

In Tables 3.3a and 3.3b a detailed description of these errors is reported. The error may be due to the evaluation of the log-likelihood in that point or the computation in the same point of the gradient, since it is approximated by finite differences.

| Model | Length | $\sigma$ | Starting point | Error point | Ground truth point |
|---|---|---|---|---|---|
| ARMA$(2,1)$ | 100 | 0.01 | strictly feasible | (iii) | strictly feasible |
| ARMA$(2,1)$ | 10000 | 0.01 | strictly feasible | (iii) | (i) |
| ARMA$(2,1)$ | 10000 | 0.01 | (i) | (ii) | strictly feasible |
| ARMA$(2,1)$ | 100 | 0.1 | strictly feasible | (iii) | strictly feasible |
| ARMA$(2,1)$ | 100 | 0.1 | (ii) | (i) | strictly feasible |
| ARMA$(2,1)$ | 100 | 0.1 | strictly feasible | (i) | strictly feasible |
| ARMA$(2,1)$ | 1000 | 0.1 | strictly feasible | (iii) | strictly feasible |
| ARMA$(2,1)$ | 10000 | 0.1 | strictly feasible | (iii) | strictly feasible |
| ARMA$(2,1)$ | 10000 | 0.1 | strictly feasible | (iii) | strictly feasible |
| ARMA$(2,1)$ | 100 | 1 | (i) | (iii) | strictly feasible |
| ARMA$(2,1)$ | 1000 | 1 | strictly feasible | (iii) | strictly feasible |
| ARMA$(2,1)$ | 10000 | 1 | strictly feasible | (iii) | strictly feasible |
| ARMA$(2,3)$ | 10000 | 1 | (ii) | (iii) | strictly feasible |
| ARMA$(3,2)$ | 100 | 0.01 | strictly feasible | (iii) | strictly feasible |
| ARMA$(5,1)$ | 10000 | 1 | strictly feasible | (i) | strictly feasible |

(a) Numerical errors in Kalman filtering when using Jones reparametrization

| Model | Length | $\sigma$ | Starting point | Error point | Ground truth point |
|---|---|---|---|---|---|
| ARMA$(4,2)$ | 10000 | 1 | strictly feasible | (iii) | strictly feasible |
| ARMA$(4,4)$ | 1000 | 0.1 | strictly feasible | (iii) | (ii) |
| ARMA$(5,5)$ | 100 | 0.1 | strictly feasible | (ii) | strictly feasible |
| ARMA$(5,5)$ | 1000 | 0.1 | strictly feasible | strictly feasible | strictly feasible |

(b) Numerical errors in Kalman filtering when using model (3.6)

Table 3.3: The first three columns contain information about the ARMA process that generated the tested series and the series itself (orders $p$ and $q$, series length, standard deviation of the white noise generator process). The fourth and fifth columns provide details about the optimization run: the starting point and the point where the error has been generated are characterized in terms of closeness to the feasibility boundary, according to the metrics introduced in Section 3.3. The last column provides the same information associated with the parameters of the model employed to generate the series.

Two patterns are clear from Tables 3.3a and 3.3b. Firstly, the classical method by Jones fails 4 times more frequently than ours. This means that the proposed reformulation protects from the occurrence of most numerical errors. Secondly, regardless of the type of parametrization employed, it is evident that these numerical errors are related to points close the boundary $\partial S_p \times \partial S_q$ of the feasible set. Furthermore, by observing the first column of both tables, it seems that most errors inside the unconstrained framework happen even when fitting low order models.

## Forecasting with Almost-Border Models

As reported above, again a multi-start approach has been employed to assess the predictive performance of close to the border ARMA models. For this analysis, time series having at least one strictly feasible solution and at least a solution that meets one of the conditions i, ii, iii have been picked. In doing so, a total of 614 time series with such features have been identified.

When multiple strictly feasible solutions are available, the best one, according to the exact log-likelihood value, have been considered. The same is done when multiple solutions close to the border are obtained for a single time series. Multi-step ahead predictions are then computed with the two selected models for each time-series.

Differences in predictive performance of these two distinct ARMA models are again investigated by means of the Wilcoxon signed-ranks test. Mean Absolute Scaled Error (MASE) (Hyndman and Koehler, 2006) has been used to measure the accuracy of forecasts. Indeed, the MASE can be used to compare forecast methods on a single series and, being scale-free, to compare forecast accuracy across series (Hyndman et al., 2006).

In the experiments, MASE at a given forecast horizon $h$ is computed as

$$\text{MASE}(h) = \frac{1}{h} \frac{\sum_{t=N+1}^{h} |y_t - \hat{y}_t|}{\frac{1}{N-1}\sum_{t=2}^{N} |y_t - y_{t-1}|}. \tag{3.8}$$

Single absolute scaled errors for each different forecast horizon $h$ are also reported:

$$\text{ScaledError}(h) = \frac{|y_{N+h} - \hat{y}_{N+h}|}{\frac{1}{n-1}\sum_{t=2}^{N} |y_t - y_{t-1}|}. \tag{3.9}$$

| **Error** | **Test statistic** | **P-value** |
|---|---|---|
| MASE(3) | -4.23197 | 2.31e−5 |
| ScaledError(1) | -1.49874 | 0.13394 |
| ScaledError(2) | -1.67521 | 0.09389 |
| ScaledError(3) | -4.35523 | 1.33e−5 |

Table 3.4: Results from the two-sided Wilcoxon test at different horizons. Null hypothesis: the median of the differences of the MASE errors, $\text{MASE}_{\text{border}} - \text{MASE}_{\text{strictly feasible}}$, is zero.

Results are reported in Tables 3.4 and 3.5. The observed P-value in the last row of Table 3.4 evidences that significant differences exist in forecast accuracy between strictly feasible ARMA$(p, q)$ models and close-to-the-border ARMA$(p, q)$ models. The significant differences involve only the MASE$(3)$ error and the absolute scaled error at horizon $h = 3$: in both cases the associated P-values are strictly lower than

| Error | Test statistic | P-value |
|---|---|---|
| MASE(3) | 4.23197 | 1.16e−5 |
| ScaledError(1) | 1.49874 | 0.06697 |
| ScaledError(2) | 1.67521 | 0.04695 |
| ScaledError(3) | 4.35523 | $<$ 1e−5 |

Table 3.5: Results from the one-sided Wilcoxon test at different horizons. Null hypothesis: the median of the differences of the MASE errors, $\text{MASE}_{\text{border}} - \text{MASE}_{\text{strictly feasible}}$, is negative.

the default significance level $\alpha = 0.05$. Furthermore, for these two metrics the one-sided test confirms that ARMA models close to the feasibility boundary perform poorer in terms of the predictive ability than the strictly feasible ARMA models.

Considering instead the remaining error metrics, results in Table 3.4 indicate that at forecast horizon $h = 1$ non-substantial difference exists in forecast accuracy between the two types of ARMA models. Differences in predictive ability become more evident as the forecast horizon grows. From Table 3.4 it is observed that at horizon 2, only assuming a significance level $\alpha = 0.1$, it is possible to deduce a statistically significant difference between the two ARMA models in forecasting performances.

The main conclusion of this experiment is that ARMA models close to the feasibility boundary perform poorer in terms of the predictive ability than the strictly feasible ARMA models. The practical meaning of this result is that caution is needed with close to the border ARMA models when forecasting is required. This is one of the motivations to modify the proposed fitting model (3.6) by adding to the objective an $\ell_2$ penalty term as in (3.7). The effects of this modification are discussed in detail in the next section.

## Forecasting with Regularized ARMA models

The next and final experiment investigates the effect of the addition of an $\ell_2$ regularization term from a forecasting accuracy perspective. Different values of the regularization hyperparameter $\lambda$ in Equation (3.7) give rise to different ARMA$(p, q)$ models with diverse forecasting performances.

For each time series, all different models are fitted starting the optimization at the same initial point. ARMA models are, in practice, fitted by iterative optimization algorithms that start at preliminary estimates obtained, for example, with the well-known Hannan and Rissanen (HR) method (Hannan and Rissanen, 1982). Precisely this setting has been considered to carry out the experiment.

Friedman test (Friedman, 1937, 1940; Demšar, 2006) has been employed to catch the differences between the methods. The test ranks the fitting methods for each

time series separately, the best performing method (lowest error) getting the rank of 1, the second best rank 2 and so on. The null-hypothesis, states that all the fitting methods are equivalent and so their ranks should be equal. Table 3.6 reports the average of ranks over all the time series in our dataset, w.r.t. the metrics of interest (3.8) and (3.9).

We observe from Table 3.6 that for the MASE$(3)$ and the absolute scaled error at horizon $h = 3$ the averages of ranks go down until a value of the hyperparameter $\lambda = 8$. For the other two errors the trend of the averages of the ranks seems quite stationary: this pattern is confirmed by the results of Friedman test as it is shown in Table 3.7.

| **Error** | **Jones** | $\lambda = 0$ | $\lambda = 1$ | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 8$ | $\lambda = 16$ |
|---|---|---|---|---|---|---|---|
| MASE$(3)$ | 4.228 | 4.201 | 4.056 | 3.947 | 3.882 | **3.825** | 3.862 |
| ScaledError$(1)$ | 4.022 | 3.996 | 4.018 | 3.999 | 3.972 | **3.968** | 4.025 |
| ScaledError$(2)$ | 4.082 | 4.095 | 4.01 | 3.972 | 3.958 | **3.935** | 3.948 |
| ScaledError$(3)$ | 4.220 | 4.226 | 4.081 | 3.980 | 3.885 | **3.798** | 3.809 |

Table 3.6: Average of ranks between different ARMA models performance w.r.t. different error metrics.

| **Error** | **Test statistic** | **P-value** |
|---|---|---|
| MASE$(3)$ | 78.06724 | $< 1e{-5}$ |
| ScaledError$(1)$ | 1.57091 | 0.95465 |
| ScaledError$(2)$ | 12.13886 | 0.05894 |
| ScaledError$(3)$ | 94.93939 | $< 1e{-5}$ |

Table 3.7: Results of Friedman test for the difference in forecasting performance of various ARMA models w.r.t. different error metrics.

Friedman test, whose results are reported in Table 3.7, suggests that the forecasting performance of the considered fitting models statistically differ (assuming a significance level of $\alpha = 0.1$) for all the errors except for the absolute scaled forecasting error at horizon $h = 1$.

Therefore, based on these results, post-hoc analysis is performed w.r.t. the MASE$(3)$, the absolute scaled forecasting error at horizon $h = 3$ and $h = 2$ (although the P-value in the latter case is not negligible).

Post-hoc analysis is performed by means of the Nemenyi test (Nemenyi, 1962; Demšar, 2006). Critical differences between two generic methods are assessed in terms of the differences between the averages of the ranks. Results of the Nemenyi test are reported in Tables 3.8, 3.9 and 3.10.

| | **Jones** | $\lambda = 0$ | $\lambda = 1$ | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 8$ | $\lambda = 16$ |
|---|---|---|---|---|---|---|---|
| **Jones** | 1.00000 | 0.90000 | 0.10395 | 0.00100 | 0.00100 | 0.00100 | 0.00100 |
| $\lambda = 0$ | 0.90000 | 1.00000 | 0.26546 | 0.00154 | 0.00100 | 0.00100 | 0.00100 |
| $\lambda = 1$ | 0.10395 | 0.26546 | 1.00000 | 0.60537 | 0.10031 | 0.00630 | 0.04196 |
| $\lambda = 2$ | 0.00100 | 0.00154 | 0.60537 | 1.00000 | 0.90000 | 0.48698 | 0.82448 |
| $\lambda = 4$ | 0.00100 | 0.00100 | 0.10031 | 0.90000 | 1.00000 | 0.90000 | 0.90000 |
| $\lambda = 8$ | 0.00100 | 0.00100 | 0.00630 | 0.48698 | 0.90000 | 1.00000 | 0.90000 |
| $\lambda = 16$ | 0.00100 | 0.00100 | 0.04196 | 0.82448 | 0.90000 | 0.90000 | 1.00000 |

Table 3.8: Post-hoc analysis of the performance forecasting: pairwise comparison of the MASE(3) error.

| | **Jones** | $\lambda = 0$ | $\lambda = 1$ | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 8$ | $\lambda = 16$ |
|---|---|---|---|---|---|---|---|
| **Jones** | 1.00000 | 0.90000 | 0.90000 | 0.60131 | 0.46951 | 0.25145 | 0.37172 |
| $\lambda = 0$ | 0.90000 | 1.00000 | 0.82448 | 0.48264 | 0.34176 | 0.16502 | 0.25839 |
| $\lambda = 1$ | 0.90000 | 0.82448 | 1.00000 | 0.90000 | 0.90000 | 0.90000 | 0.90000 |
| $\lambda = 2$ | 0.60131 | 0.48264 | 0.90000 | 1.00000 | 0.90000 | 0.90000 | 0.90000 |
| $\lambda = 4$ | 0.46951 | 0.34176 | 0.90000 | 0.90000 | 1.00000 | 0.90000 | 0.90000 |
| $\lambda = 8$ | 0.25145 | 0.16502 | 0.90000 | 0.90000 | 0.90000 | 1.00000 | 0.90000 |
| $\lambda = 16$ | 0.37172 | 0.25839 | 0.90000 | 0.90000 | 0.90000 | 0.90000 | 1.00000 |

Table 3.9: Post-hoc analysis of the performance forecasting: pairwise comparison of the absolute scaled error at horizon $h = 2$.

| | **Jones** | $\lambda = 0$ | $\lambda = 1$ | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 8$ | $\lambda = 16$ |
|---|---|---|---|---|---|---|---|
| **Jones** | 1.00000 | 0.90000 | 0.31753 | 0.00357 | 0.00100 | 0.00100 | 0.00100 |
| $\lambda = 0$ | 0.90000 | 1.00000 | 0.27263 | 0.00259 | 0.00100 | 0.00100 | 0.00100 |
| $\lambda = 1$ | 0.31753 | 0.27263 | 1.00000 | 0.67435 | 0.03709 | 0.00100 | 0.00100 |
| $\lambda = 2$ | 0.00357 | 0.00259 | 0.67435 | 1.00000 | 0.73116 | 0.07136 | 0.11154 |
| $\lambda = 4$ | 0.00100 | 0.00100 | 0.03709 | 0.73116 | 1.00000 | 0.80825 | 0.90000 |
| $\lambda = 8$ | 0.00100 | 0.00100 | 0.00100 | 0.07136 | 0.80825 | 1.00000 | 0.90000 |
| $\lambda = 16$ | 0.00100 | 0.00100 | 0.00100 | 0.11154 | 0.90000 | 0.90000 | 1.00000 |

Table 3.10: Post-hoc analysis of the performance forecasting: pairwise comparison of the absolute scaled error at horizon $h = 3$.

Regarding the absolute scaled error at horizon $h = 2$, results from the Nemenyi test indicate no significant differences between the fitting methods in terms of the forecasting performances. All the P-values reported in Table 3.9 are greater than 0.1.

On the other end, results about absolute scaled error at horizon $h = 3$ and the MASE(3) are equivalent. By observing both Table 3.8 and Table 3.10, no significant difference is found between the two non-regularized methods. Furthermore, no significant differences in forecasting performance have been identified between both the non-regularized methods and the regularized one with $\lambda = 1$.

Instead, stronger regularization leads to significantly better forecasts w.r.t. the non-regularized methods. Forecasting performance, as mentioned above, starts to deteriorate as the regularization hyperparameter grows to $\lambda = 16$. In summary, the constrained fitting method with regularization leads to causal and invertible ARMA models with better short term predictive ability than the non-regularized ones.

## 3.6 Conclusions

Fitting causal and invertible ARMA models by constrained optimization in the partial autocorrelation and partial moving-average coefficients space has several advantages w.r.t. the classical unconstrained approach based on the Jones reparametrization. First of all, the proposed approach leads to a significant reduction of the fitting times. Moreover, almost-border solutions are often avoided. Such solutions, as further experiments highlight, are bad both because they lead to numerical errors during the optimization of the ARMA exact log-likelihood and because they do not perform well at forecasting.

Based on these results $\ell_2$-regularization is suggested to discourage almost-border solutions. As non parametric statistical tests assess, $\ell_2$-regularization also improves the short term forecasting performances of causal and invertible ARMA models.

# Chapter 4

# Sparse Convex Combinations of Forecasting Models By Meta Learning

As anticipated in Chapter 1, we introduce SRFA (Sparse Robust Forecast Averaging) and SFFA (Sparse Flexible Forecast Averaging) meta-learners. Both the meta-learners are trained to recommend, based on time series features, sparse convex combinations of forecasting methods. Sparse combinations are motivated as the right compromise between the computational savings of the strategy, which does not require the fitting of the zero-weighted methods, and the advantages that any combination strategy provides in terms of predictive accuracy (Atiya, 2020).

The idea of combining only a subset of the available forecasting methods is a successful and popular strategy (Jose and Winkler, 2008; Diebold and Shin, 2019; Lichtendahl Jr and Winkler, 2020) but non in the context of meta-learning in support of time series forecasting.

The proposed methodology is tested on the recent M4 competition dataset (Makridakis et al., 2020), which is now retained as the benchmark dataset in the community of forecasting, by evaluating both the degree of sparsity and the predictive performance obtained by SRFA and SFFA.

Based on the predictions of SRFA, we also developed an analysis, which focuses on understanding the relationship between the exclusion of a forecasting method and the features of the series. This analysis deepens the content of the obtained results in terms of *intepretability* (Molnar, 2020).

The remainder of the chapter is structured as follows. Section 4.1 contains a review of the meta-learning methods in time series forecasting. Following (Atiya, 2020), in Section 4.2 we analyze the effects of the combination strategy on the bias and the variance of the forecast. A summary of data and a description of both the employed forecasting methods and time series features are given in Section 4.3. Section 4.4 outlines the proposed methodology. Section 4.5 describes the implementation of the methodology w.r.t. the M4 competition time series dataset. Experiments

and results are discussed in Section 4.6. Finally, conclusions are reported in Section 4.7.

## 4.1 Forecasting by Meta-Learning: related works

The plan of using time series features to select an appropriate forecasting model has been pursued since the 1990's and is still actual and enhanced by the development of flexible machine learning models acting as meta-learners.

In (Arinze et al., 1997), a completely automatic system, precursor of the more recent approaches, is firstly proposed. Training time series, each represented by a set of six features, are employed to create an induction tree able to recommend the most appropriate forecasting method or a combination of methods.

Prudêncio and Ludermir (2004) are the first to talk about meta-learning in the context of automatic model selection for time series. First, they proposed a decision tree to select between two forecasting methods based on six features of the time series. Then, they adapted the NOEMON approach (Kalousis and Theoharis, 1999) to rank and select three forecasting models.

Later, Lemke and Gabrys (2010) highlight how a ranking approach of combining four methods outperforms other meta-learning approaches and also improves upon the best individual method.

More recently, meta-learner systems based both on decision trees (Talagala et al., 2018; Montero-Manso et al., 2020) and neural networks (Kück et al., 2016; Li et al., 2020; Ma and Fildes, 2021) are spreading out. Neural networks can automate the process of feature extraction which becomes *learnable* from the series. Especially Convolution Neural Network (CNN) can discover and extract the suitable internal structure to generate deep features of the input time series automatically by using convolution and pooling operations (Zhao et al., 2017).

In (Talagala et al., 2018) a Random Forest, employed as a meta-learner, is used to pick the best forecasting method, from a pool of candidate forecasting methods, based on a set of manual selected time series features. This framework is known as FFORMS (Feature-Based FORecast Model Selection) and it is considered the precursor of the FFORMA (Feature-Based FORecast Model Averaging) methodology (Montero-Manso et al., 2020). In FFORMA, a XGBoost (Chen and Guestrin, 2016) model is trained to obtain non zero and sum one weights for each of nine well known forecasting methods based on a set of fourty-two manually extracted time-series features. FFORMA performed very well in the M4 forecasting combination (Makridakis et al., 2020) finishing in second place.

In (Kück et al., 2016), a single-hidden-layer Multilayer Perceptron is trained to select the best of four exponential smoothing models. Error based features and statistical tests are employed for the selection of the forecasting models.

Differently from the other mentioned approaches, in (Ma and Fildes, 2021) time series features are automatically extracted by the employment of a CNN and then linked with a set of weights which are used to combine the forecasting methods. Similarly, in (Li et al., 2020), time series features are learned by CNN from the recurrence plots of time series. These features are then mapped into weights of forecasting methods by minimizing the same weighted average loss function used in FFORMA.

Finally in (Kang et al., 2021), a novel approach of extracting time series features, named *Forecast with Forecasts*, is proposed. Pairwise diversity measures, computed from the forecasts of methods to be combined, are used as time series features. Through meta-learning, these features are then used to fit combination of forecasting models. In fact, the diversity of methods in a combination improves the predictive accuracy of the final forecast as we show in the next section.

## 4.2 Why does forecast combination work well?

Let $y = (y_1, \ldots, y_H)$ be the time series values for the horizon $H$ to be forecast. Following the derivation of (Atiya, 2020), we consider the bias-variance decomposition of the Mean Squared Error (MSE) in predicting $y$ by means of a generic convex combination $f_{\text{comb}}$ of forecasts $f_1, \ldots, f_L$, with weights $w = \{w_i\}_{i=1}^{L}$:

$$\begin{aligned} & f_{\text{comb}} = \sum_{i=1}^{L} w_i f_i \\ & \text{s.t. } \; w \in W = \{w : w_i \geq 0 \, , \sum_{i=1}^{L} w_i = 1\} \text{ (probability simplex)} \end{aligned} \tag{4.1}$$

The Bias $B$ of $f_{\text{comb}}$ is the weighted average of the individual bias $B_i$ of forecasts $f_i$:

$$\begin{aligned} B &= \mathbb{E}(f_{\text{comb}}) - \mathbb{E}(y) = \\ &= \sum_{i=1}^{L} w_i \mathbb{E}(f_i) - \mathbb{E}(y) = \\ &= \sum_{i=1}^{L} w_i B_i, \end{aligned} \tag{4.2}$$

where the expectation is over the variations of the error terms of the DGP. For example, a structural break of the DGP can lead to bias of the individual forecasts. However, in (Tjøstheim and Paulsen, 1983), it is pointed out that rarely the biases of the individual forecasts share the same direction. This means that the strategy of combining forecasts is favourable in terms of bias reduction allowing individual bias cancellations.

Now, we consider the variance $V$ of the combination $f_{\text{comb}}$. Let $\sigma_i^2$ be the individual forecast variance and $\rho_{ij}$ the correlation coefficient of forecast $i$ and $j$. The variance $V$ of $f$ is:

$$\begin{aligned} V &= \mathbb{E}\left(f_{\text{comb}} - \mathbb{E}(f_{\text{comb}})\right)^2 \\ &= \left(\sum_{i=1}^{L} w_i \sigma_i\right)^2 - 2\sum_{i=1}^{L}\sum_{j=i+1}^{L} w_i w_j (1-\rho_{ij})\sigma_i \sigma_j. \end{aligned} \tag{4.3}$$

By the RMS-arithmetic weighted mean inequality, we have:

$$\sum_{i=1}^{L} w_i \sigma_i \leq \sqrt{\sum_{i=1}^{L} w_i \sigma_i^2}, \tag{4.4}$$

which implies the following upper bound of the variance $V$:

$$V \leq \sum_{i=1}^{L} w_i \sigma_i^2 - 2\sum_{i=1}^{L}\sum_{j=i+1}^{L} w_i w_j (1-\rho_{ij})\sigma_i \sigma_j. \tag{4.5}$$

Since the second term in (4.5) is positive, it is possible to observe that many positive terms are subtracted from the first term (average variance), indicating that the variance of the forecast combination tends to decrease substantially. The extent of the decrease depends on the correlation coefficients $\rho_{ij}$ among the forecasting methods: if the correlation coefficients are smaller, the decrease in variance becomes larger. This means that diversity improves the performance of the forecast combination.

Now, we focus on sparse convex forecast combinations:

$$\begin{aligned} &f_{\text{comb}} = \sum_{i=1}^{L} w_i f_i \\ &\text{s.t. } \; w \in \partial W \text{ (boundary of probability simplex)} \end{aligned} \tag{4.6}$$

As reported in (Atiya, 2020), it is likely that the correlation coefficients of the base forecasting models are typically larger than 0.5. Assuming values of correlations coefficients approaching 1, the upper bound (4.5) of the forecast combination variance approximates the weighted average of individual variances:

$$\sum_{i=1}^{L} w_i \sigma_i^2 - 2\sum_{i=1}^{L}\sum_{j=i+1}^{L} w_i w_j (1-\rho_{ij})\sigma_i \sigma_j \approx \sum_{i=1}^{L} w_i \sigma_i^2. \tag{4.7}$$

In such scenario, where some forecasting models are redundant, it is evident that an appropriate sparse weighting can reduce the variability of the resulting forecast.

## 4.3 Forecasting Methods, Data, Feature Set and Error Metrics

Our pool of methods is reported in Table 4.1. These are well known forecasting algorithms, used in recent related studies (Montero-Manso et al., 2020; Kang et al., 2021; Li et al., 2020). All the methods are implemented in the R `forecast` package (Hyndman et al., 2008).

| Forecasting Method | R implementation |
|---|---|
| Automated ARIMA model | `auto.arima()` |
| Automated exponential smoothing state space model | `ets()` |
| Feed-forward neural network with autoregressive (AR) inputs | `nnetar()` |
| TBATS model | `tbats()` |
| Seasonal and trend decomposition using Loess with AR seasonally adjusted series. | `stlm(modelfunction = ar)` |
| Random walk with drift | `rwf(drift = TRUE)` |
| Theta method | `thetaf()` |
| Naive method | `naive()` |
| Seasonal naive method | `snaive()` |

Table 4.1: Methods used for forecast combination.

We provide an overview of each forecasting method employed for the proposed SRFA and SFFA meta-learners. We also dwell into the main details of the R implementation.

- `auto.arima()` returns best ARIMA model or SARIMA model according to the value of a chosen information criterion. Best model is found by means of a heuristic search (algorithm of Hyndman-Khandakar in (Hyndman et al., 2008)) over the potential ARIMA and SARIMA models. For further details, see Section 2.5.

- `ets()` returns a model belonging to the class of exponential smoothing state space models. These models are specified by a set of stochastic equations describing the underlying components of the series, i.e. the level, trend and seasonality. This class of models include known forecasting algorithms, as the Simple Exponential Smoothing (Brown, 1959), Holt's Linear Method (Holt, 2004), Damped Trend Method (Gardner Jr and McKenzie, 1985), Holt-Winters' Trend and Seasonality Method (Holt, 2004; Winters, 1960) and their main variations, reformulating them as state space models (Hyndman et al., 2002). This reformulation allows easy calculation of the likelihood and model selection criteria. `ets()` implements an automatic forecasting procedure which picks the best model based on a full enumerative strategy.

- `nnetar()` implements the only machine learning forecasting algorithm in our pool. This function fits a feed-forward neural networks with a single hidden layer and lagged inputs for forecasting, i.e a nonlinear autoregressive model.

The optimal number of input lags and the number of nodes in the hidden layer are selected according to the AIC criterion as reported in (Hyndman and Athanasopoulos, 2018).

- `tbats()` function is conceived to model both high-frequency seasonal time series, which often exhibit multiple seasonal patterns, and seasonal time series with non integer-valued period. On this matter, `tbats()` provides the R implementation of the Trigonometric Box-Cox Arma-errors Trend Seasonality (TBATS) model (De Livera et al., 2011), which belongs to the class of state space models. TBATS model employs a trigonometric representation of each seasonal components based on Fourier series. This trigonometric representation accommodates also non-integer seasonality. Since the TBATS model has several parameters to be estimated, its estimation can be computationally challenging.

- `stlm(modelfunction = ar)` computes the required forecast in a sequential way. First, the function uses the STL decomposition (Robert et al., 1990) to decompose the input time series into its trend, seasonal and residual components. Then, an autoregressive model is fitted to the seasonally adjusted time series. By means of the fitted model, final forecasts are computed by summing up the autoregressive forecasts and the corresponding estimated seasonal components.

- `rwf(drift = TRUE)` returns forecasts for a random walk with drift model applied to the input time series. The presence of the drift term allows the forecasts to increase or decrease over time. Drift is estimated as the average between consecutive observations seen in historical data.

- `thetaf()` is the R implementation of Theta forecasting method (Assimakopoulos and Nikolopoulos, 2000). Theta method is based on the concept of modifying the local curvatures of the time series. The initial time series, eventually deseasonalised if any seasonality is detected, is decomposed into a set of two new time series, called Theta-lines, each corresponding to a particular value of the theta coefficient. The first time series describes the input time series through a linear trend in order to boost the long term behavior of the original series. The second time series has its second differences exactly twice the original time series, hence magnifying its short term behavior. The first time series is then extrapolated in the usual way for a linear trend, while the second is extrapolated via simple exponential smoothing. Final forecast is them calculated as the average of these two new forecasts.

- `naive()` returns forecasts simply equal to the value of the last observation. This method works remarkably well for many economic and financial time series (Hyndman and Athanasopoulos, 2018).
- `snaive()` sets each forecast equal to the last observed value from the same season of the year.

M4 dataset consists of a set of 100,000 time series from different domains and frequency, including high-frequency data (weekly, daily and hourly) along with low-frequency data (yearly, quarterly and monthly).

| Frequency | Micro | Industry | Macro | Finance | Demographic | Other | Total |
|---|---|---|---|---|---|---|---|
| **Yearly** | 6538 | 3716 | 3903 | 6519 | 1088 | 1236 | **23000** |
| **Quarterly** | 6020 | 4637 | 5315 | 5305 | 1858 | 865 | **24000** |
| **Monthly** | 10975 | 10017 | 10016 | 10987 | 5728 | 277 | **48000** |
| **Weekly** | 112 | 6 | 41 | 164 | 24 | 12 | **359** |
| **Daily** | 1476 | 422 | 127 | 1559 | 10 | 633 | **4227** |
| **Hourly** | 0 | 0 | 0 | 0 | 0 | 414 | **414** |
| **Total** | 25121 | 18798 | 19402 | 24534 | 8708 | 3437 | **100000** |

Table 4.2: Number of M4 series per data frequency and domain

Yearly time series have been managed separately. A different meta-learner has been trained independently for yearly time series using both a restricted number of time series features and forecast methods w.r.t. the non-yearly time series. Since the aim is to reduce at most the number of methods in the forecast combination, `tbats()`, `stlm(modelfunction = ar)` and `snaive()` methods have been excluded for yearly time series. In fact, these methods are conceived to deal with high frequency data or seasonal patterns.

The same set of 42 features of the FFORMA meta-learner (Montero-Manso et al., 2020), has been employed to represent all the time series except the yearly ones. This set of features includes trend, seasonality, linearity, curvature, and correlation features calculated by means of the R package `tsfeatures` (Hyndman et al., 2019). Conversely, a restricted number of features has been employed to represent yearly time series: only the 36 features for non seasonal data as reported in (Montero-Manso et al., 2020) were used.

We evaluate the accuracy of forecasts, employing both the symmetric mean absolute percentage error (sMAPE) (Makridakis, 1993) and the seasonal Mean Absolute Scaled Error (MASE) (Hyndman and Koehler, 2006):

$$\text{sMAPE} = \frac{2}{h} \sum_{t=N+1}^{N+h} \frac{|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|} * 100\% \tag{4.8}$$

$$\text{MASE} = \frac{1}{h} \frac{\sum_{t=N+1}^{n+h} |y_t - \hat{y}_t|}{\frac{1}{N-m} \sum_{t=m+1}^{N} |y_t - y_{t-m}|}, \tag{4.9}$$

where $\hat{y}_t$ is the forecast produced over up to horizon $h$, $m$ is the time interval between successive observations considered by the organizers for each data frequency, i.e. 12 for monthly, 4 for quarterly, 24 for hourly and 1 for yearly, weekly and daily data, as stated in (Makridakis et al., 2020).

## 4.4 Inducing Sparsity of Forecast Combination

Forecast combinations are obtained with the aid of a deep neural network, properly trained as a meta-learner for the purpose of gaining knowledge about the identification of sparse averaging strategies which can lead to an improvement of predictive performances. More in detail, the proposed SRFA and SFFA meta-learners differ in the way this knowledge is acquired from the performances of forecasting methods on previous prediction problems.

SRFA is trained to learn the mapping which links the features of a time series to the most promising subset of methods in terms of forecast accuracy. The learning process of this meta-learner happens in a fully supervised way, by solving a a multi-label classification problem (Zhang and Zhou, 2006; Nam et al., 2014; Yu et al., 2014) on a dataset $\mathcal{D}_{\text{srfa}} = \{(x_i, v_i)\}_{i=1}^{N}$ of meta-examples which consist of pairs $(x_i, v_i)$ of time series feature representations $x_i$ and multi-hot encoding vector $v_i$ [1] of the best performing methods for that series. At inference time, when the prediction is required on a new time series, the forecast is computed as the simple average of forecasts obtained from methods that SRFA identifies as well performing on that series: this is actually equivalent to the prediction obtained from a sparse convex combination of all available methods in which all the methods are zero weighted except the ones identified by SRFA that receive the same weight. Note that this strategy does not take in account the different contributions of the methods in the computation of the final forecast (**sparse robust strategy**).

SFFA is directly trained to combine forecasts of different methods. In particular, this meta-learner aims to learn the mapping which links the features of a time series to a sparse set of $L$ weights, each multiplying a specific forecast of the combination. In training phase, the meta-data for SFFA are represented by a dataset $\mathcal{D}_{\text{sffa}} = \{(x_i, F_i, y_i)\}_{i=1}^{N}$ of examples, where $F_i = (f_1^{(i)}, \ldots, f_L^{(i)})$ is the forecast matrix of the base forecasting methods, whose columns contain the forecasts of base methods, computed over a given horizon $h$, to predict the corresponding ground true values $y_i = (y_{n+1}^{(i)}, \ldots, y_{n+h}^{(i)})^T$. Both $F_i$ and $y_i$ enter in the computation of the training loss whose aim is to calibrate the ability of the meta-learner to create proper forecast combinations. Given a new time series to be predicted, the trained SFFA

[1] Each element $v_i^{(l)}$ of the vector $v_i$ is equal to 1 if the $l$-th forecasting method performs well, according to some pre-defined criterion, on the i-*th* time series, otherwise is set to 0.

transforms the feature representation of this series into a sparse set of combination weights. Sparsity of the weights, is obtained by means of the Sparsemax activation function (Martins and Astudillo, 2016), which transforms the output of the network into a sparse set of weights. The use of this activation function allows a non uniform weighting of the active (non zero weighted) forecasts to be combined (**sparse flexible strategy**).

The two meta-learners partially share a common neural network architecture, i.e. a Multi-Layer-Perceptron (MLP), composed of one input layer, two hidden layers and one output layer, as shown in Figure 4.1, but are different for the way the output of the network is managed as will be clarified in the following. We highlight that this simple structure has been chosen to guarantee the fastest possible inference. In fact, in a real time scenario, especially for very long time series, the process of feature extraction, which determines the input signal to the network, can be computationally expensive and for this reason this simple architecture can compensate for the time-consuming due to feature extraction.

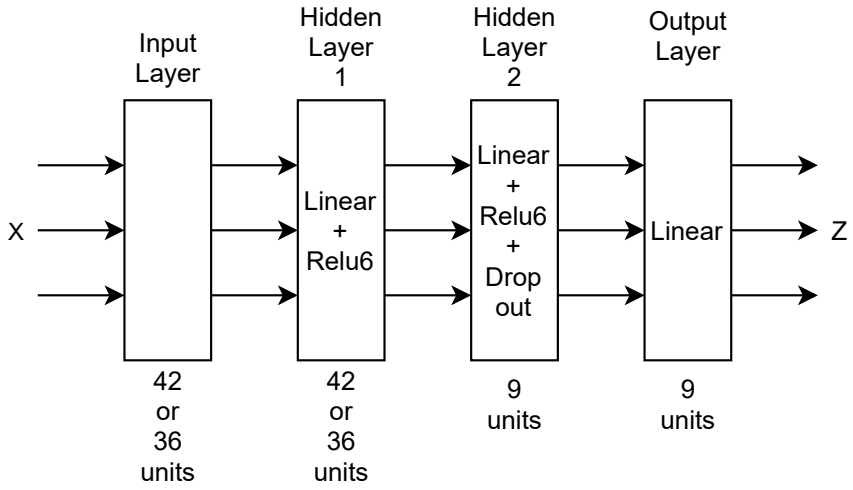



Figure 4.1: Neural network architecture of SRFA and SFFA: the network outputs the un-normalized vector $z(x, \theta) \in \mathbb{R}^L$ for a generic time series feature representation $x$. In the present case, since there are 9 base forecasting methods, $L$ is equal to 9.

In both cases, the network takes as input the extracted time series features $x \in \mathcal{X}$ and returns a set of un-normalized weights $z(x, \theta) \in \mathbb{R}^L$ whose dimension matches the number of base forecasting methods. The information flow passes through two hidden layers before reaching the final output layer. Both the first and the second hidden layers linearly transform the input from the previous layers and then they utilize a Relu 6 activation function (Howard et al., 2017) to increase robustness and to mitigate the problem of vanishing gradient. This last non-linearity is followed by a dropout regularizer with rate $(0.1)$ to attenuate the overfitting. Finally, the resulting signal is linearly processed by the output layer which returns $z(x, \theta)$.

In what follows, we more formally introduce SRFA and SFFA and get insight into the differences between them.

## Multi-label Classification (SRFA)

Let $V = \{0,1\}^L$ be the $L$-dimensional binary indicator vector space of subsets of the label set $\mathcal{Y}$ which is the output domain of possible forecasting methods as reported in Table 4.1. Given a training set $\mathcal{D}_{\text{srfa}} = \{(x_i, v_i) |\ x_i \in \mathcal{X}, v_i \in V\}_{i=1}^N$ of examples (meta-data), the aim of SRFA is to learn the mapping function $m(.)$:

$$m(.) : \mathcal{X} \to \{0,1\}^L,$$

which maps a time series representation $x \in \mathcal{X}$ into a binary indicator vector $v \in V$ of a subset of forecasting methods that performs well for that series.

Training labels $v_i$ are obtained as follows. Firstly, for each training series, the forecasting performances of base methods are evaluated by means of the MASE metric. Two cases are then considered: (i) for time series in which all the forecasting methods perform well (MASE $< 1$) or all bad (MASE $\geq 1$), the first three methods, according to the ranking of MASE, are regarded as best methods and marked with a 1 while the others with a 0 (ii) for the other time series, we labelled with a 1 only the methods with MASE $< 1$ and the others with a 0.

The employment of a single multi-output neural network, as the one in Figure 4.1, is an intuitive and effective approach to solve multi-label classification tasks (Zhang and Zhou, 2006; Nam et al., 2014; He and Xia, 2018). In the present case, multiple sigmoid functions $\sigma(.)$ are used as activation functions of each output neuron, to predict the probability that a forecasting method is relevant, in terms of its performance, for the input time series. An additional threshold function converts each predicted probability to a 0-1 binary label.

Network is fit by minimizing the Binary Cross Entropy (BCE) loss function $L(\theta)$ w.r.t. the weights of the network $\theta$:

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{L} \text{BCE}\left(v_i^{(l)}, \hat{v}^{(l)}(\theta, x_i)\right), \tag{4.10}$$

where $v_i^{(l)}$ is the binary label for the $l$-th forecasting method of the $i$-th training time series, while $\hat{v}^{(l)}(\theta, x_i) = \sigma\big(z^{(l)}(\theta, x_i)\big)$ is the related predicted conditional probability, computed by means the sigmoid activation function which operates over the $l$-th unnormalized output neuron $z^{(l)}(\theta, x_i)$ of the network.

The weighting function $\tilde{h}(.)$ works on the binary predictions of the neural network by assigning the same weight (equal to the inverse of the number of predicted methods) to each of the survived methods and zero to the remaining methods.

### Minimum Loss of Combined Forecasts (SFFA)

In this case, given a training set $\mathcal{D}_{\text{sffa}} = \{(x_i, F_i, y_i)\}_{i=1}^{N}$ of examples (meta-data), the network is directly trained to learn a sparse weighting function $\tilde{h}(.)$:

$$\tilde{h}() : \mathcal{X} \rightarrow W,$$

which returns a sparse set of weights based on time series features. The Sparsemax activation function in the output layer returns the projection $w(x, \theta)$ of the vector $z(x, \theta)$ onto the probability simplex $W$:

$$\text{sparsemax}\big(z(x,\theta)\big) = \underset{w \in W}{\arg\min}\, ||w - z(x,\theta)||^2.$$

This projection is likely to hit the boundary of $W$, in which case $w(x, \theta)$ becomes sparse.

As shown in Figure 4.2, forecasts and ground truth observations enter directly as input to the network for the computation of the loss.

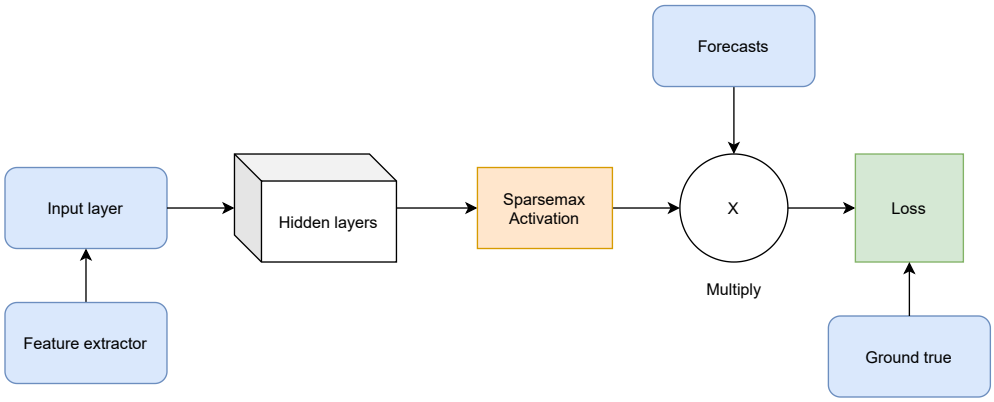



Figure 4.2: SFFA meta-Learner with Sparsemax final activation function. Sparsemax transforms the network processed signal into a sparse set of weights, each referring to a forecasting method of the combination.

Network is fit by minimizing the following scaled squared loss $L(\theta)$ w.r.t. the weights of the network $\theta$:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{MSE}\big(F_i w(x_i, \theta),\ y_i\big)}{\text{MSE}\big(F_i w,\ y_i\big)}, \tag{4.11}$$

where $F_i$ is the input forecast matrix for the $i$-th training time series. Each $l$-column of $F_i$ contains the forecast obtained by the $l$- method in order to predict the ground true observations $y_i$. The numerator of Equation 4.11 is the Mean Squared Error (MSE), related to the $i$-th training time series, of the forecast combination with weights $w(x_i, \theta)$ predicted by the network. The denominator is the Mean Squared Error (MSE) of the simple average forecast combination, i.e. $w = \left(\frac{1}{L}, \ldots, \frac{1}{L}\right)^T$, for the $i$-th training time series.

## 4.5 Implementation

In what follows, the way SRFA and SFFA meta-learners have been implemented with the M4 competition data is in detail described.

Each available time series of the M4 dataset occurs already divided into a training (historical data) and a testing period (future data). Time series observations, corresponding to the testing period, have been exclusively used to evaluate the forecasts computed by means of the trained meta-learners. In training phase, the only information known about the testing period is represented by the length of the period, which is equal to the number of required multi-step-ahead forecasts: forecast horizons for yearly, quarterly, monthly, weekly, daily and hourly time series are set up by the organizers of the competition equal to 6, 8, 18, 13, 14, 48 (Makridakis et al., 2020).

### Metadata Generation

The datasets $\mathcal{D}_{\text{srfa}} = \{(x_i, v_i)\}_{i=1}^{N}$ and $\mathcal{D}_{\text{sffa}} = \{(x_i, F_i, y_i)\}_{i=1}^{N}$, necessary for the training of the two meta-learners, are determined based on the training period of all the time series of the M4 competition according to a temporal hold-out strategy in 4.3.

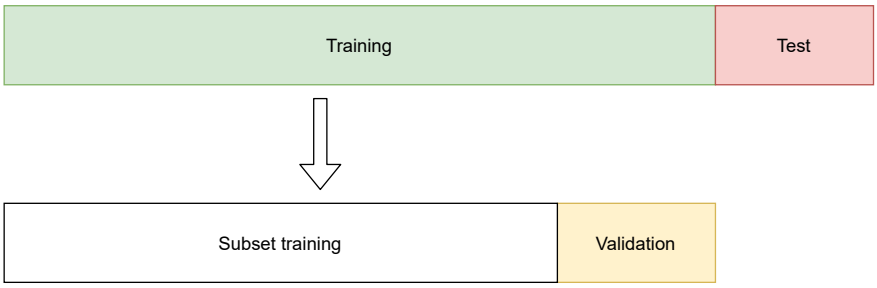



Figure 4.3: Temporal hold-out strategy to generate the training datasets $\mathcal{D}_{\text{srfa}} = \{(x_i, v_i)\}_{i=1}^{N}$ and $\mathcal{D}_{\text{sffa}} = \{(x_i, F_i, y_i)\}_{i=1}^{N}$ of the meta-learners. The features $x_i$ are calculated using the initial part of the training period (subset training period). Then, for each base model, the forecasts $F_i$ are calculated for the validation period $y_i$. Labels $v_i$ of dataset $\mathcal{D}_{\text{srfa}}$ are assigned based on the MASE error of forecasts $F_i$.

The initial part of the training period is employed to extract the time series features $x_i$, which provide the training inputs of the meta-learners. Labels $v_i$ of SRFA meta-learner are obtained by the computation of the MASE over the validation periods as explained in Figure 4.3. Both forecasts $F_i$ and validation period observations $y_i$ represent itself the input data for the training of the SFFA meta-learner as shown in Figure 4.2.

Since each meta-learner is independently trained for yearly and non-yearly time series, four training datasets $\mathcal{D}_{\text{srfa}}^{(no\ yearly)}$, $\mathcal{D}_{\text{srfa}}^{(yearly)}$, $\mathcal{D}_{\text{sffa}}^{(no\ yearly)}$ and $\mathcal{D}_{\text{sffa}}^{(yearly)}$ have been generated in total.

### Training SRFA and SFFA

The learning process of the meta-learners is performed on the training datasets $\mathcal{D}_{\text{srfa}}^{(no\ yearly)}$, $\mathcal{D}_{\text{srfa}}^{(yearly)}$, $\mathcal{D}_{\text{sffa}}^{(no\ yearly)}$ and $\mathcal{D}_{\text{sffa}}^{(yearly)}$ by optimizing the respective losses in Equations (4.10) and (4.11).

ADAM optimizer (Kingma and Ba, 2014) in combination with a batch size of 32 samples has been employed to minimize our losses. A grid search strategy with a 5 fold cross-validation mechanism has been set up to validate the optimization hyperparameters. In particular, we considered the following hyperparameters for every neural network optimization:

- *learning rate* $\in \{10^{-5}, 10^{-3}, 10^{-2}\}$
- *weight decay* $\in \{0, 10^{-3}\}$
- *epochs* $\in \{15, 50, 100\}$

Obtained results are reported in Table 4.3.

| **Meta-learner** | ***learning rate*** | ***weight decay*** | ***epochs*** |
|---|---|---|---|
| **SRFA (no yearly)** | $10^{-5}$ | $10^{-3}$ | 50 |
| **SFFA (no yearly)** | $10^{-3}$ | $10^{-3}$ | 15 |
| **SRFA (yearly)** | $10^{-3}$ | $10^{-3}$ | 100 |
| **SFFA (yearly)** | $10^{-3}$ | $10^{-3}$ | 50 |

Table 4.3: Results of grid search for hyperparameter selection: the reported values optimize the F1 score evaluation metric for the SRFA multi-label classifier and the scaled MSE for the SFFA meta-learner.

## 4.6 Experiments

Firstly, the reliability of the proposed approach is assessed both in terms of quality of point forecasting and achieved degree of sparsity by computing the forecasts over the test period of the M4 competition with the support of the trained meta-learners. Degree of sparsity is defined as the percentage reduction of the total number of fitted models provided by our methodology w.r.t. the non sparse approaches based on the same pool of models (Montero-Manso et al., 2020; Li et al., 2020; Kang et al., 2021). Forecasts over the test period are also computed by means of the simple average

combination of the methods in Table 4.1. It is well known in literature that this latter is an effective forecasting strategy [2].

Then, we focus to understand the relationship between the exclusion of a model and the features of the series. For each forecasting model, predictions from SFFA meta-learner (trained on all the time series of M4 dataset except the yearly ones) have been used to assign binary exclusion labels of the algorithms. In this way, we obtained, for each forecasting model, a dataset $\{(x_i, v_i^{(l)})\}_{i=1}^{N}$, where $x_i$ is the feature representation of the $i$-th time series and $v_i^{(l)}$ is the respective label of exclusion referring to the $l$-th forecasting model. Since the presence of multicollinearity in the original feature set has been detected, which is problematic when the modeling focus is the interpretability rather than the predictability, we reduced the set of time series features based on the ranking of features provided by ReliefF algorithm (Kononenko, 1994) for each dataset. ReliefF calculates a score for each time series feature that can be used to estimate the relevance in predicting the exclusion of the forecasting method. By selecting, according to the ranking of ReliefF, the six most relevant features, new smaller datasets have been obtained with a degree of multicollinearity strongly reduced. Then, we fitted a gradient boosting classifier to model the probability of exclusion of each model as a function of the selected time series features.

Finally, for purposes of interpretation, based on the fitted gradient boosting classifiers, we report in Appendix B the partial dependence plots (Friedman, 2001) that show the non-linear relationship between the probability of exclusion of each forecasting method and its three most influential features, according to the ReliefF algorithm ranking.

Python has been used for these experiments. `Rpy2` library provided the Python interface to the R language. Forecasts have been computed by means of the R package `forecast` (Hyndman et al., 2008) while the time series features have been extracted by the `tsfeatures` package (Hyndman et al., 2019). Neural networks have been implemented using `PyTorch` (Paszke et al., 2019). All the experiments were performed on a machine with Ubuntu Server 20.04 LTS OS, Intel(R) Core(TM) i7-6700 @ 3.40GHz CPU and 32GB RAM.

## Forecast Accuracy and Degree of Sparsity

Results in Tables 4.4, 4.5, 4.6 clearly show that both the meta-learners strongly limit the requirement of fitting all the available methods in building forecast combinations. The degree of sparsity, obtained by SRFA, exceeds the sparsity of SFFA. Roughly, this means that, for our pool of forecasting methods, which consists of nine methods, SRFA builds on average combinations of five methods while SFFA six methods.

[2] This fact is known as *forecasting puzzle* (Stock and Watson, 2004; Smith and Wallis, 2009).

Although this considerable reduction, forecasting performances of the two meta-learners, reported in Tables 4.4, 4.5, 4.6, are comparable with the top ten methods of the M4 competition (Makridakis et al., 2020).

| Method | sMAPE | MASE | Decrease of fitted models |
|---|---|---|---|
| SRFA | 11.221 | 1.135 | -42.555% |
| SFFA | 11.254 | 1.13 | -30.101% |
| Simple average | 11.642 | 1.209 | 0 |

Table 4.4: Results over Monthly, Quarterly, Weekly, Daily, Hourly time series of M4 competition

| Method | sMAPE | MASE | Decrease of fitted models |
|---|---|---|---|
| SRFA | 13.602 | 3.085 | -48.096% |
| SFFA | 13.574 | 3.069 | -56.716% |
| Simple average | 14.146 | 3.195 | 0 |

Table 4.5: Results over Yearly time series of M4 competition

| Method | sMAPE | MASE | Decrease of fitted models |
|---|---|---|---|
| SRFA | 11.769 | 1.584 | -43.475% |
| SFFA | 11.787 | 1.576 | -34.521% |
| Simple average | 12.218 | 1.666 | 0 |

Table 4.6: Results over all time series of M4 competition

Statistics about the importance and the frequency of exclusion of the forecast algorithms are reported in Tables 4.7, 4.9, 4.8, 4.10. Importance is simply estimated as the average of weights assigned by the meta-learner overall the time series.
Considering the non-annual time series, which represent more than $\frac{2}{3}$ of the available time series, both the meta-learners are quite coherent regarding the importance of methods: as expected, `auto.arima()`, `tbats()` and `thetaf()` are estimated as the most important methods. In fact, these three methods are the most sophisticated requiring heavier computation, especially `tbats()`. The only machine learning method of the pool, i.e. `nnetar()`, does not work well according to the assigned importance by the meta-learners. Its failure is probably due to its tendency of overfitting dealing with univariate time series. The probabilities of exclusion follow essentially the same pattern of the assigned importance: most important methods are the one with lowest probability of exclusion. It is important to note in Table 4.8 that SFFA almost never excludes `tbats()`.

Regarding annual time series, SRFA confirms `auto.arima()` as the most important and least excluded forecasting method, while `nnetar()` is practically never included in the combination. By observing Tables 4.9, 4.10, it is interesting to note

how for annual time series both the meta-learners give great importance to a simple forecasting method as `rwf(drift = TRUE)`. Indeed, as shown in Table 4.10, SFFA gives even more importance to `rwf(drift = TRUE)` than `auto.arima()`. Furthermore, since for annual time series, as reported in Table 4.5, SFFA strictly outperforms the other forecast combination strategies, the policy of forming forecast combinations which include `rwf(drift = TRUE)` is made even corroborated.

| Forecasting Method | Average of Weights (SD) | Exclusion Prob. |
|---|---|---|
| `auto.arima()` | 0.196 (0.090) | 0.043 |
| `ets()` | 0.185 (0.077) | 0.068 |
| `nnetar()` | 0.039 (0.073) | 0.773 |
| `tbats()` | 0.188 (0.081) | 0.061 |
| `stlm(modelfunction = ar)` | 0.010 (0.040) | 0.932 |
| `rwf(drift= TRUE)` | 0.108 (0.105) | 0.411 |
| `theta()` | 0.177 (0.085) | 0.096 |
| `naive()` | 0.086 (0.113) | 0.528 |
| `snaive()` | 0.012 (0.042) | 0.92 |

Table 4.7: Importance and probability of exclusion assigned by the SRFA meta-learner to the forecasting methods over Monthly, Quarterly, Weekly, Daily, Hourly time series

| Forecasting Method | Average of Weights (SD) | Exclusion Prob. |
|---|---|---|
| `auto.arima()` | 0.220 (0.139) | 0.053 |
| `ets()` | 0.075 (0.058) | 0.199 |
| `nnetar()` | 0.021 (0.048) | 0.676 |
| `tbats()` | 0.226 (0.084) | 0.005 |
| `stlm(modelfunction = ar)` | 0.089 (0.091) | 0.315 |
| `rwf(drift= TRUE)` | 0.056 (0.077) | 0.504 |
| `theta()` | 0.178 (0.127) | 0.041 |
| `naive()` | 0.056 (0.068) | 0.454 |
| `snaive()` | 0.078 (0.102) | 0.461 |

Table 4.8: Importance and probability of exclusion assigned by the SFFA meta-learner to the forecasting methods over Monthly, Quarterly, Weekly, Daily, Hourly time series

| Forecasting Method | Average of Weights (SD) | Exclusion Prob. |
|---|---|---|
| auto.arima() | 0.280 (0.137) | 0.157 |
| ets() | 0.216 (0.155) | 0.316 |
| nnetar() | 0.012 (0.058) | 0.956 |
| rwf(drift= TRUE) | 0.278 (0.147) | 0.19 |
| theta() | 0.135 (0.160) | 0.557 |
| naive() | 0.079 (0.126) | 0.71 |

Table 4.9: Importance and probability of exclusion assigned by the SRFA meta-learner to the forecasting methods over Yearly time series

| Forecasting Method | Average of Weights (SD) | Exclusion Prob. |
|---|---|---|
| auto.arima() | 0.337 (0.324) | 0.281 |
| ets() | 0.116 (0.126) | 0.404 |
| nnetar() | 0.025 (0.065) | 0.806 |
| rwf(drift= TRUE) | 0.432 (0.393) | 0.284 |
| theta() | 0.024 (0.075) | 0.831 |
| naive() | 0.066 (0.177) | 0.798 |

Table 4.10: Importance and probability of exclusion assigned by the SFFA meta-learner to the forecasting methods over Yearly time series

## Interpreting the Exclusion of Forecasting Methods

Considering the ReliefF ranking of features in Figure 4.4, the main consideration is about the recurrence of autocorrelation based features. For each forecasting method, at least one feature, which measures some type of autocorrelation, is one of the three most relevant features.

A substantial presence of features related to the seasonal patterns of the series, mostly in the scores referring to simple forecasting methods, is also observed in Figure 4.4. In fact, seasonal_period and seasonal_strength time series features, which respectively measure the length of the seasonal period and its strength (Wang et al., 2006), are relevant in predicting the exclusion of forecasting methods as naive() and rwf(drift=True). Furthermore, as shown in Figures B.3 and B.5 in the Appendix B, it is possible to see that for these methods, the probability of exclusion from the combination increases with the increase of the values of seasonal_strength. This result confirms our expectations since these latter methods do not consider any seasonality of the series in their forecasts. The seasonal_strength is also the most relevant feature in predicting the exclusion of the nnetar() forecasting method, which is the only considered machine learning method. This time, however, it is clear in Figure B.4 that the increase of the values of the seasonal_strength feature leads to a drastic decrease of the probability of exclusion of nnetar(). We deem that in time series with a strong seasonal component, nnetar(), which implements a non-linear forecasting model, succeeds well to capture the seasonality of the time

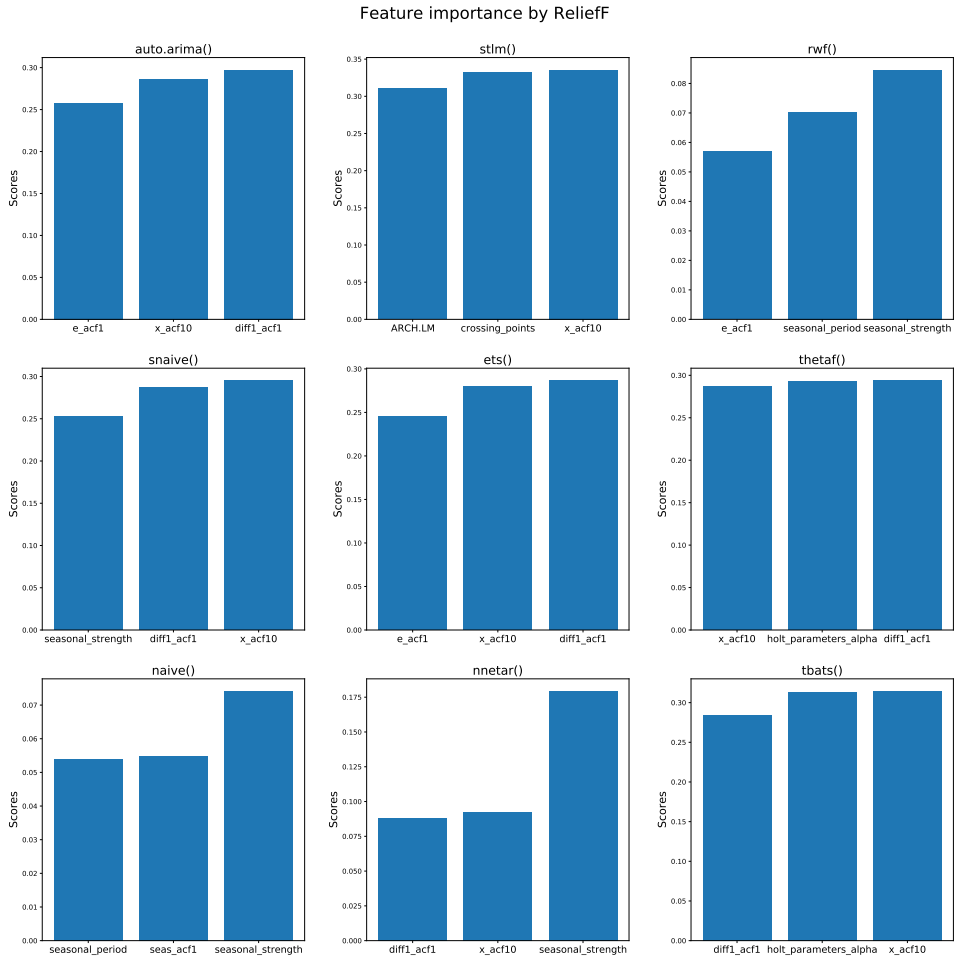



Figure 4.4: Ranking by ReliefF algorithm: each plot shows the three most influential features in predicting the exclusion of a forecast method.

series.

An interesting fact is that the pair of methods `auto.arima()`, `ets()` shares the same set of three most relevant features according to ReliefF ranking in Figure 4.4. Furthermore, the respective partial dependence plots in Figure B.1 and in Figure B.2 are quite similar. This is justifiable since the class of Auto Regressive Integrated Moving Average (ARIMA) models and the class of Exponential Smoothing State Space models, known as ETS models, partially overlap. Finally, note in Figure 4.4 that the two pairs of forecasting methods `snaive()`, `nnetar()` and `thetaf()`, `tbats()` share the same sets of relevant features.

## 4.7 Conclusions

We introduced SRFA and SFFA meta-learners in order to sparsely combine forecasting methods based on handcrafted time series features. Weighting and selection of the forecasting methods occur at the same time in a completely automatic manner which replaces the human knowledge.

The performances of SRFA and SFFA are evaluated on the M4 time-series competition dataset. Accuracy of point forecasting, provided by the two meta-learners, are remarkable according to the MASE and sMAPE error metrics. More in detail, the obtained results are slightly worse in terms of forecast accuracy than the FFORMA meta-learner (Montero-Manso et al., 2020), which ranked 2-th in the competition. Furthermore, both the meta-learners succeed in reducing significantly the overall number of fitted models in combining forecasts and in particular SRFA obtains a higher level of sparsity than SFFA: at inference time, when prediction is required on a new time series, SRFA rules out on average 4 out 9 forecasting methods while SFFA 3 out 9. We deem that with such a level of sparsity the proposed approach fits well within real time forecasting systems.

The feature extraction process of SRFA and SFFA uses the same approach of the FFORMA meta-learner (Montero-Manso et al., 2020). For this reason, as regards the part of feature extraction, it has to be recognised that our method is not more efficient than FFORMA. On this matter, a-posteriori descriptive analysis has been conduct to detect the most relevant time series features in predicting the exclusion of a forecasting method. According to the results of this analysis, the features that summarize the autocorrelation structure and seasonality patterns of the series, are almost always the most important in predicting the exclusions of the methods. This finding may be useful in future works to identify low-cardinality but meaningful sets of time series features, whose determination requires low computational effort and therefore a better adaption to real-time forecasting systems.

# Chapter 5

# Conclusions and Future Developments

Some relevant problems in the area of statistical modeling have been covered in this dissertation. For each of these problems, novel optimization approaches or meta-learning systems, have been proposed to assist the modeler in the building of a final model. The whole research has been inspired by the idea that the modeler, in a real scenario, needs efficient and performing tools to specify a model. On this matter, each of the proposed contributions pursues this insight.

As first contribution, we improved the state of the art by introducing the Alternate Minimization algorithm for best subset selection problem in Gaussian linear regression and order selection problem in Gaussian ARMA time series. In both of these scenarios, the algorithm has distinguished for its computational efficiency in providing fitted linear models in a data-driven manner. As possible development, it could be interesting to compare Alternate Minimization with LASSO and all the related approaches which are widely spread especially in the statistical community.

As second contribution, we proposed an efficient bound constrained formulation for exact maximum likelihood estimation of causal and invertible Gaussian ARMA$(p, q)$ models. The carried out experiments highlight computational time saving, better numerical stability and improvement of short term forecasting performances if $\ell_2$ regularization is applied. Possibly, this fitting method can refine the ARMA$(p, q)$ model returned by Alternate Minimization. Therefore the modeler, first using Alternate Minimization for preliminary estimation and then this fitting method, is able to fit ARMA$(p, q)$ models of good quality without preliminary knowledge.

Finally, we contributed to the forecast averaging problem. The strategy of combining forecasting methods improves the predictive performances but requires high computational times. On this matter, two meta learning systems, i.e. SRFA and SFFA, have been introduced. Both the meta-learners automate the averaging pro-

cess. The fundamental intuition of these meta-learners lies in their ability of identifying, based on time series features, an appropriate subset of available forecasting methods to be combined. This characteristic is appropriate in real time forecasting problems, since the fit of the excluded methods can be avoided, thereby reducing the total computational time to obtain the combined forecast. Our experiments highlight that both the meta-learners succeed in reducing significantly the overall number of fitted models in combining forecasts, without affecting the predictive performances. The reduction of computational times is also encouraged by the architecture of the two meta-learners, i.e. a shallow neural network composed of two hidden layers. As a future development, a systematic study that compares the meta-learning approaches with all the other forecast averaging methods, may be of great interest in the forecasting community. A further possible development could be about the investigation of forecasting performances with machine learning models. Are there any time series patterns that require the use of more sophisticated forecasting methods like for example recurrent neural networks?

# Appendix A

# Derivation of AIC and BIC

We provide a sketch derivation of the AIC and BIC information criteria. Further details can be found in (Burnham and Anderson, 2002; Neath and Cavanaugh, 2012; Portet, 2020).

Let us define the notation that we will use in the following:

- $f$ is the unknown DGP
- $G = \{g_l(x|\theta_l) : \theta_l \in \Theta(k_l)\}_{l=1}^{L}$ is a set of candidate models in terms of probability density functions
- $\mathcal{L} = \{L_l(\theta_l|x) : \theta_l \in \Theta(k_l)\}_{l=1}^{L}$ is a set of candidate models in terms of likelihood functions
- $K = \{k_l\}_{l=1}^{L}$ is the set of dimensions of parameter spaces referring to each candidate model
- $\Pi(l)$, $l \in \{1, \ldots, L\}$ denotes a discrete prior over the models

## Derivation of AIC

For models $f$ and $g \in G$, KL is defined as follows:

$$\mathrm{KL}(f, g) = \int f(x) \ln\left(\frac{f(x)}{g(x|\theta)}\right) dx = C - \mathbb{E}\left[\ln(g(x|\theta))\right], \tag{A.1}$$

where the expectation is clearly w.r.t. $f$. Ideally, the modeler should choose the model $g \in G$ by maximizing:

$$\max_{g \in G} \mathbb{E}\left[\ln(g(x|\theta))\right]. \tag{A.2}$$

It is not possible to solve Problem (A.2), since the model $f$ is unknown. In a real scenario, the modeler is limited to observe a sample $y$ from $f$ by which the parameter

$\theta$ of a statistical model $g$ are estimated by the maximum likelihood estimator $\hat{\theta}(y)$. Therefore, it makes sense to compute the expectation, always w.r.t. $f$, of the random variable $\mathbb{E}\left[\ln(g(x|\hat{\theta}(y)))\right]$, by obtaining the Expected Relative Kullback–Leibler information (ERKL):

$$\text{ERKL} = \mathbb{E}\Big[\mathbb{E}\left[\ln(g(x|\hat{\theta}(y)))\right]\Big], \tag{A.3}$$

which is the quantity that, in a real estimation scenario, the modeler should be able to maximize in order to select the best fitted model belonging to the set $G$ of candidate models. The problem is that its computation requires again the knowledge of $f$ which is unknown. The key result, obtained by Akaike, is that asymptotically (for a large sample) an unbiased estimator of the ERKL is given by:

$$l(\hat{\theta}|y) - k, \tag{A.4}$$

where $l(\hat{\theta}|y)$ is the log-likelihood function referring to a fitted statistical model, evaluated at its maximum $\hat{\theta}$ and $k$ is the dimension of the model. For historical reasons, Akaike multiplied the estimator (A.4) by –2, hence defining the AIC criterion:

$$\text{AIC} = -2l(\hat{\theta}|y) + 2k. \tag{A.5}$$

Therefore the best model within the set $G$ is the one with the minimum AIC value. In this derivation is clear as maximum likelihood estimation theory is linked to KL information theory, providing an useful tool for model selection.

## Derivation of BIC

Assuming to observe a sample $y$ of $N$ observations from $f$, the posterior probability for the $l$-th candidate model is given by:

$$P(l|y) = P(y)^{-1}\Pi(l)\int_{\Theta(k_l)} L(\theta_l|y)P(\theta_l|l)\,d\theta_l, \tag{A.6}$$

where $P(\theta_l|l)$ is a prior over $\theta_l$ given the $l$-th model and $P(y)$ is the probability of the observed sample $y$ which does not need for the purpose of model selection. Hence, by discarding $P(y)$, maximizing $P(l|y)$ is equivalent to minimize the following:

$$S(l|y) = -2\ln\big(\Pi(l)\big) - 2\ln\Big(\int_{\Theta(k_l)} L(\theta_l|y)P(\theta_l|l)\,d\theta_l\Big). \tag{A.7}$$

By taking the second order approximation of the log-likelihood function $l(\theta_l|y)$ around its maximum $\hat{\theta}_l$, likelihood function in (A.7) can be approximated as:

$$L(\theta_l|y) \approx L(\hat{\theta}_l|y) \times \exp\Big\{-\frac{1}{2}(\theta_l - \hat{\theta}_l)^T\left[N\bar{\mathcal{I}}(\hat{\theta}_l, y)\right](\theta_l - \hat{\theta}_l)\Big\}, \tag{A.8}$$

where $\bar{\mathcal{I}} = -\frac{1}{N}\frac{\partial^2 l(\hat{\theta}_l|y)}{\partial\theta_l\partial\theta_l^T}$ is average observed Fisher information matrix. It follows that the integral in Equation (A.7) becomes:

$$\begin{aligned}&\int_{\Theta(k_l)} L(\theta_l|y)P(\theta_l|l)\,d\theta_l \approx \\ &L(\hat{\theta}|y)\int_{\Theta(k_l)} \exp\Big\{-\frac{1}{2}(\theta_l-\hat{\theta}_l)^T\big[N\bar{\mathcal{I}}(\hat{\theta}_l,y)\big](\theta_l-\hat{\theta}_l)\Big\}P(\theta_l|l)\,d\theta_l.\end{aligned} \tag{A.9}$$

As long as the prior $P(\theta_l|l)$ is flat over the neighborhood of $\hat{\theta}_l$ and the number of observations $N$ goes to infinity, by using the Laplace approximation method, the integral in Equation (A.7) can be finally approximated as:

$$\begin{aligned}&\int_{\Theta(k_l)} L(\theta_l|y)P(\theta_l|l)\,d\theta_l \approx \\ &L(\hat{\theta}|y)(\frac{2\pi}{N})^{(\frac{k_l}{2})}|\bar{\mathcal{I}}(\hat{\theta}_l,y)|^{-\frac{1}{2}}.\end{aligned} \tag{A.10}$$

We can now approximate $S(l|y)$:

$$S(l|y) \approx -2\ln\big(\Pi(l)\big) - 2l(\hat{\theta}_l|y) + k_l\Big[\ln(\frac{N}{2\pi})\Big] + \ln|\bar{\mathcal{I}}(\hat{\theta}_l,y)|^{-\frac{1}{2}} \tag{A.11}$$

Ignoring the terms in Equation (A.11) that are bounded as the sample size grows to infinity, BIC criterion for a generic model of dimension $k$ is finally derived

$$\text{BIC} = -2l(\hat{\theta}|y) + k\ln(N). \tag{A.12}$$

Since BIC criterion is derived from the posterior probability distribution (A.6) over a candidate model, it is clear that the motivation behind BIC can be seen through a Bayesian development of the model selection problem.

# Appendix B

# Partial Dependence Plots

The following Figures B.1, B.2, B.3, B.4, B.5, B.6, B.7, B.8, B.9 show, for every forecasting method, how the predicted probability of exclusion of a method from the combination changes according to the changes of the most three relevant time series features of that method. We remember that the relevance of time series features is set out by the ReliefF algorithm.

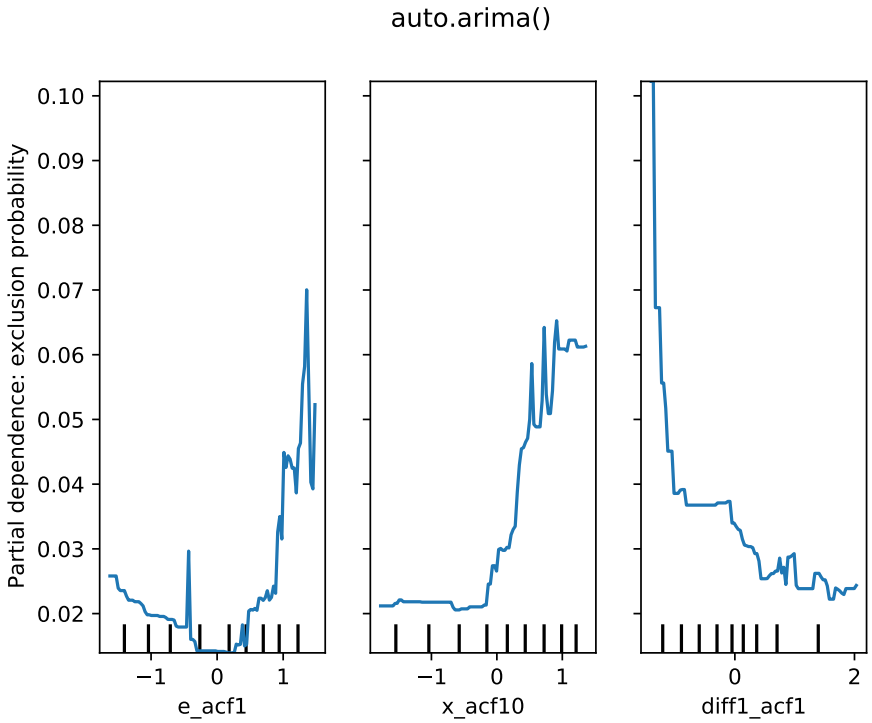



Figure B.1: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for auto.arima method.

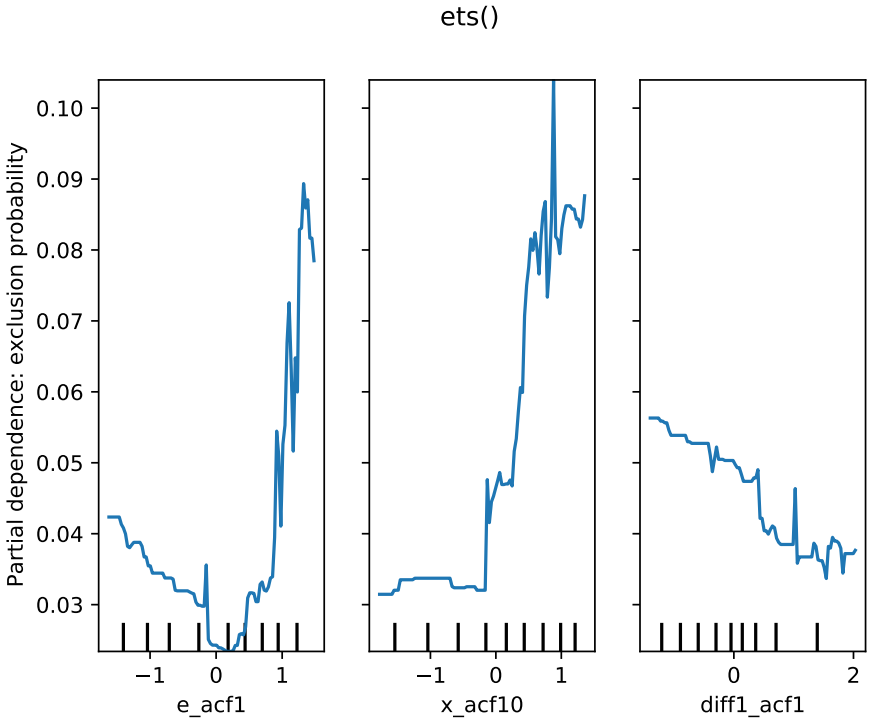



Figure B.2: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for ets method.

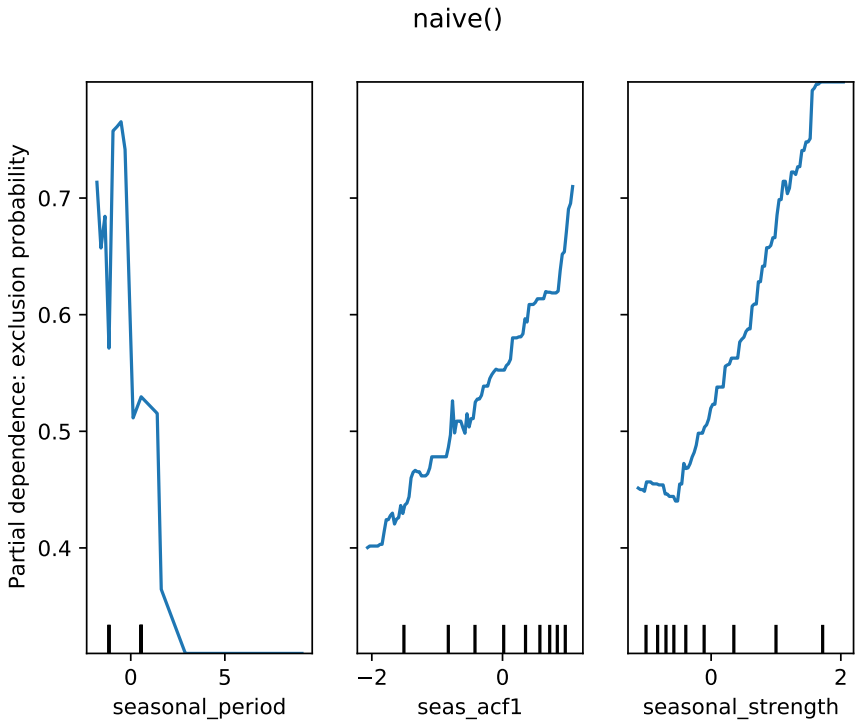



Figure B.3: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for naive method.

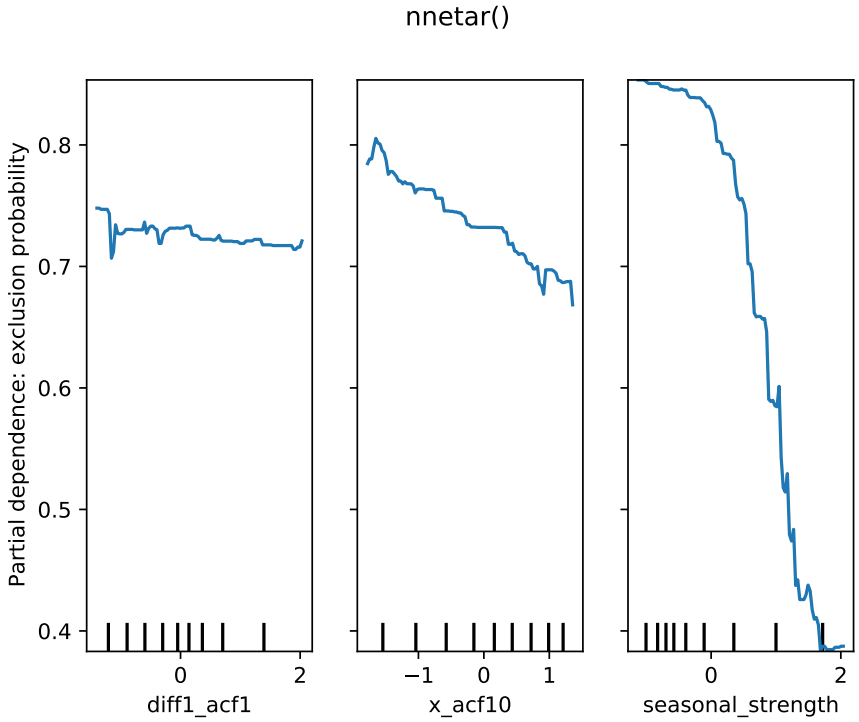



Figure B.4: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for nnetar method.

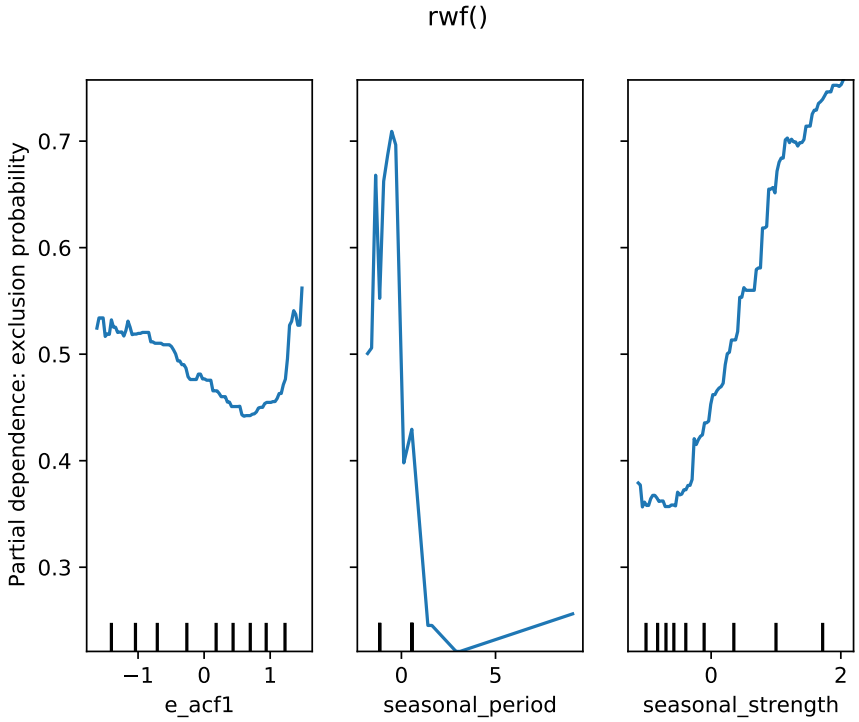



Figure B.5: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for rwf method.

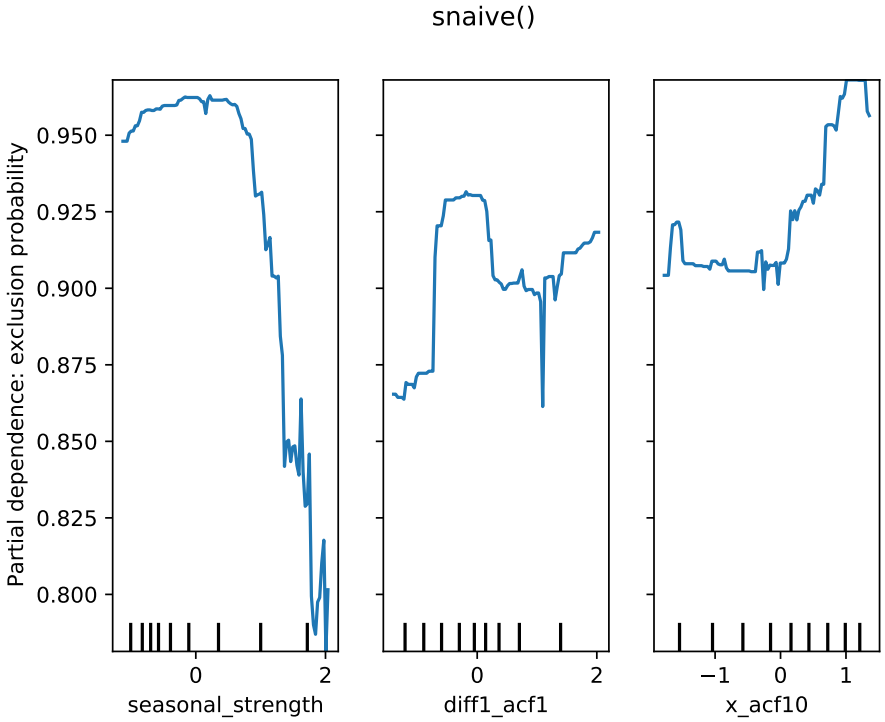



Figure B.6: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for snaive method.

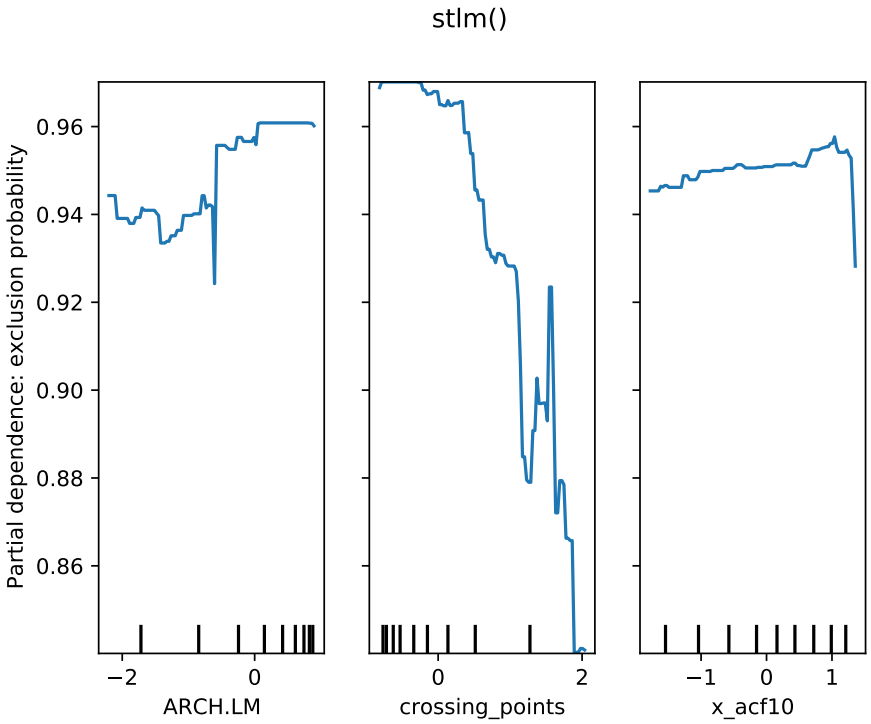



Figure B.7: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for stlm method.

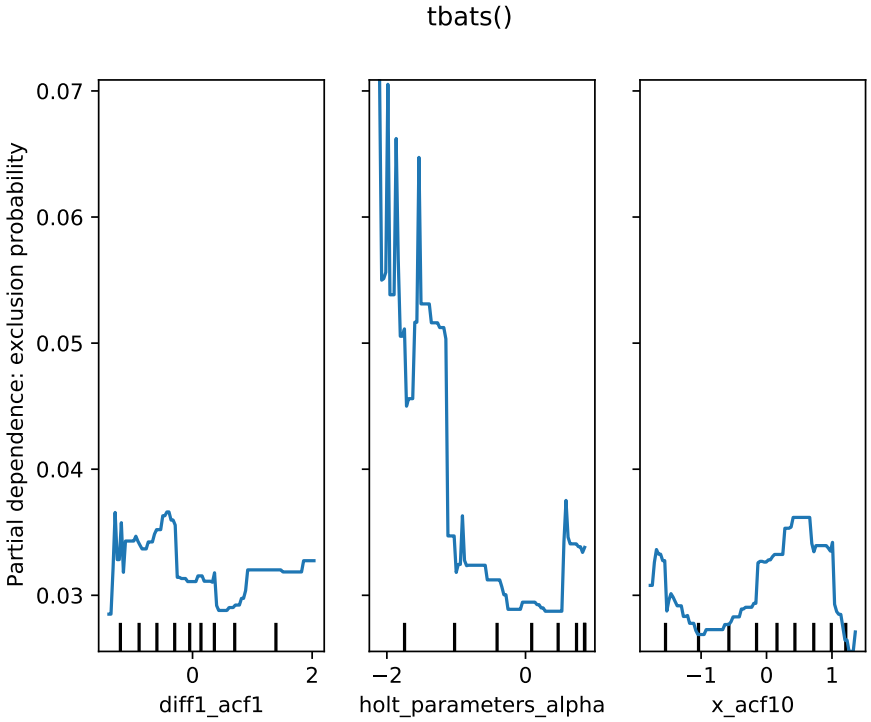



Figure B.8: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for tbats method.

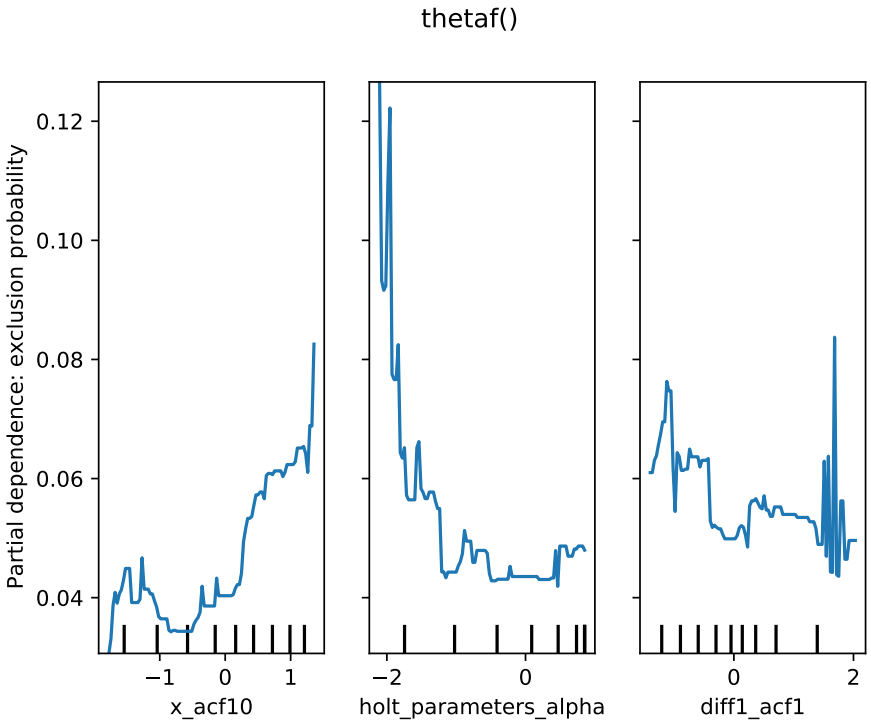



Figure B.9: Partial dependence plots showing top three features (according to ReliefF ranking) of time series that affect probability of exclusion for theta method.

# Appendix C

# Publications

## Journal papers

- **L. Di Gangi**, M. Lapucci, F. Schoen, A. Sortino, "An efficient optimization approach for best subset selection in linear regression, with application to model selection and fitting in autoregressive time-series", *Computational Optimization and Applications*, 74(3):919-948, 2019. **Candidate's contributions**: designed algorithms, carried out numerical experiments.

- M. Gulino, **L. Di Gangi**, A. Sortino, D. Vangi, "Injury risk assessment based on pre-crash variables: The role of closing velocity and impact eccentricity", *Accident Analysis & Prevention*, volume: 150, 2021. **Candidate's contributions**: designed algorithms, carried out numerical experiments.

- **L. Di Gangi**, M. Lapucci, F. Schoen, A. Sortino, "Improved Maximum Likelihood Estimation of ARMA Models", accepted at *Lobachevskii Journal of Mathematics*. **Candidate's contributions**: designed algorithms, carried out numerical experiments.

## Papers under review

- **L. Di Gangi**, "Sparse Convex Combinations Of Forecasting Models By Meta Learning", *Expert Systems with Applications*. **Candidate's contributions**: designed algorithms, carried out numerical experiments.

- T. Aldinucci, E. Civitelli, **L. Di Gangi**, A. Sestini, "Locally Explainable Random Forest by Reinforcement Learning", *Knowledge-Based Systems*. **Candidate's contributions**: designed algorithms.

## Other

- **L. Di Gangi**, "Mixed Integer Optimization in ARMA models". **Candidate's contributions**: designed algorithms, carried out numerical experiments, contributed to theoretical analyses.
- T. Aldinucci, E. Civitelli, **L. Di Gangi**, A. Sortino, "Leaving The Decision To The Best Trees of a Random Forest". **Candidate's contributions**: designed algorithms, carried out numerical experiments, contributed to theoretical analyses.

# Bibliography


Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723.

Akaike, H. (1998). Markovian representation of stochastic processes and its application to the analysis of autoregressive moving average processes. In *Selected Papers of Hirotugu Akaike*, pages 223–247. Springer.

Ansley, C. F. (1979). An algorithm for the exact likelihood of a mixed autoregressive-moving average process. *Biometrika*, 66(1):59–65.

Ansley, C. F. and Newbold, P. (1980). Finite sample properties of estimators for autoregressive moving average models. *Journal of Econometrics*, 13(2):159–183.

Arinze, B., Kim, S.-L., and Anandarajan, M. (1997). Combining and selecting forecasting models using rule based induction. *Computers & operations research*, 24(5):423–433.

Assimakopoulos, V. and Nikolopoulos, K. (2000). The theta model: a decomposition approach to forecasting. *International journal of forecasting*, 16(4):521–530.

Atiya, A. F. (2020). Why does forecast combination work so well? *International Journal of Forecasting*, 36(1):197–200.

Barndorff-Nielsen, O. and Schou, G. (1973). On the parametrization of autoregressive models by partial autocorrelations. *Journal of multivariate Analysis*, 3(4):408–419.

Bates, J. M. and Granger, C. W. (1969). The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468.

Berk, R., Brown, L., Buja, A., Zhang, K., and Zhao, L. (2013). Valid post-selection inference. *The Annals of Statistics*, pages 802–837.

Bertsekas, D. P. (2016). *Nonlinear programming*. Athena Scientific.

Bertsimas, D. and King, A. (2015). Or forum—an algorithmic approach to linear regression. *Operations Research*, 64(1):2–16.

Bertsimas, D., King, A., and Mazumder, R. (2016). Best subset selection via a modern optimization lens. *The annals of statistics*, 44(2):813–852.

Bertsimas, D. and Shioda, R. (2009). Algorithm for cardinality-constrained quadratic optimization. *Computational Optimization and Applications*, 43(1):1–22.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Brockwell, P. J., Davis, R. A., and Fienberg, S. E. (1991). *Time series: theory and methods: theory and methods*. Springer Science & Business Media.

Broersen, P. M. (2006). *Automatic autocorrelation and spectral analysis*. Springer Science & Business Media.

Brown, R. G. (1959). *Statistical forecasting for inventory control*. McGraw/Hill.

Burnham, K. P. and Anderson, D. R. (2002). *A practical information-theoretic approach*, volume 2. Springer New York.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Clemen, R. T. (1989). Combining forecasts: A review and annotated bibliography. *International journal of forecasting*, 5(4):559–583.

Combettes, P. L. and Trussell, H. J. (1992). Best stable and invertible approximations for arma systems. *IEEE Transactions on signal processing*, 40(12):3066–3069.

De Livera, A. M., Hyndman, R. J., and Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association*, 106(496):1513–1527.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

Dent, W. and Min, A.-S. (1978). A monte carlo study of autoregressive integrated moving average processes. *Journal of Econometrics*, 7(1):23–55.

Di Gangi, L., Lapucci, M., Schoen, F., and Sortino, A. (2019). An efficient optimization approach for best subset selection in linear regression, with application to model selection and fitting in autoregressive time-series. *Computational Optimization and Applications*, 74(3):919–948.

Diebold, F. X. and Shin, M. (2019). Machine learning for regularized survey forecast combination: Partially-egalitarian lasso and its derivatives. *International Journal of Forecasting*, 35(4):1679–1691.

Ding, J., Tarokh, V., and Yang, Y. (2018). Model selection techniques: An overview. *IEEE Signal Processing Magazine*, 35(6):16–34.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213.

Dua, D. and Graff, C. (2017). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. `http://archive.ics.uci.edu/ml`.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360.

Fan, J. and Lv, J. (2010). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, 20(1):101.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

Gardner Jr, E. S. and McKenzie, E. (1985). Forecasting trends in time series. *Management science*, 31(10):1237–1246.

Gómez, A. and Prokopyev, O. (2018). A mixed-integer fractional optimization approach to best subset selection. Technical Report Optimization On Line, 6795, Swanson School of Engineering, University of Pittsburgh.

Granger, C. W. and Ramanathan, R. (1984). Improved methods of combining forecasts. *Journal of forecasting*, 3(2):197–204.

Gurobi Optimization LLC (2018). Gurobi optimizer reference manual. `http://www.gurobi.com`.

Hamilton, J. D. (1994). *Time series analysis*, volume 2. Princeton New Jersey.

Hannan, E. J. and Quinn, B. G. (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):190–195.

Hannan, E. J. and Rissanen, J. (1982). Recursive estimation of mixed autoregressive-moving average order. *Biometrika*, 69(1):81–94.

Harvey, A. C. (1990). Forecasting, structural time series models and the kalman filter.

Harvey, A. C. and Phillips, G. D. (1979). Maximum likelihood estimation of regression models with autoregressive-moving average disturbances. *Biometrika*, 66(1):49–58.

He, H. and Xia, R. (2018). Joint binary neural network for multi-label learning with applications to emotion classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 250–259. Springer.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hyndman, R., Kang, Y., Montero-Manso, P., Talagala, T., Wang, E., Yang, Y., and O'Hara-Wild, M. (2019). tsfeatures: Time series feature extraction. *R package version*, 1(0).

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Hyndman, R. J. et al. (2006). Another look at forecast-accuracy metrics for intermittent demand. *Foresight: The International Journal of Applied Forecasting*, 4(4):43–46.

Hyndman, R. J., Khandakar, Y., et al. (2008). Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, 27(3):1–22.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688.

Hyndman, R. J., Koehler, A. B., Snyder, R. D., and Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of forecasting*, 18(3):439–454.

Jones, M. (1987). Randomly choosing parameters from the stationarity and invertibility region of autoregressive–moving average models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 36(2):134–138.

Jones, R. H. (1980). Maximum likelihood fitting of arma models to time series with missing observations. *Technometrics*, 22(3):389–395.

Jose, V. R. R. and Winkler, R. L. (2008). Simple robust averages of forecasts: Some empirical results. *International journal of forecasting*, 24(1):163–169.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Kalousis, A. and Theoharis, T. (1999). Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, 3(5):319–337.

Kang, K. M. (1975). A comparison of estimators for moving average processes. *Unpublished Paper, Australian Bureau of Statistics*.

Kang, Y., Cao, W., Petropoulos, F., and Li, F. (2021). Forecast with forecasts: Diversity matters. *European Journal of Operational Research*.

Kim, C.-J. and Kim, J. (2013). Thepile-up problem'in trend-cycle decomposition of real gdp: Classical and bayesian perspectives.

Kimura, K. and Waki, H. (2018). Minimization of Akaike's information criterion in linear regression analysis via mixed integer nonlinear program. *Optimization Methods and Software*, 33(3):633–649.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kitagawa, G. (2020). Computation of the gradient and the hessian of the log-likelihood of the state-space model by the kalman filter. *arXiv preprint arXiv:2011.09638*.

Konishi, S. and Kitagawa, G. (2008). *Information criteria and statistical modeling*. Springer Science & Business Media.

Konno, H. and Yamamoto, R. (2009). Choosing the best set of variables in regression analysis using integer programming. *Journal of Global Optimization*, 44(2):273–282.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer.

Kück, M., Crone, S. F., and Freitag, M. (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1499–1506. IEEE.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Lemke, C. and Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10-12):2006–2016.

Li, X., Kang, Y., and Li, F. (2020). Forecasting with time series imaging. *Expert Systems with Applications*, 160:113680.

Lichtendahl Jr, K. C. and Winkler, R. L. (2020). Why do some combinations perform better than others? *International Journal of Forecasting*, 36(1):142–149.

Liu, Y. and Tajbakhsh, S. D. (2020). Fitting arma time series models without identification: A proximal approach. *arXiv preprint arXiv:2002.06777*.

Ma, S. and Fildes, R. (2021). Retail sales forecasting with meta-learning. *European Journal of Operational Research*, 288(1):111–128.

Makridakis, S. (1993). Accuracy measures: theoretical and practical concerns. *International journal of forecasting*, 9(4):527–529.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74.

Marriott, J. (1995). Bayesian analysis of arma processes: Complete sampling-based inferences under full likelihood. *Bayesian Statistics and Econometrics: Essays in Honor of Arnold Zellner*.

Martins, A. and Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623. PMLR.

Mauricio, J. A. (2002). An algorithm for the exact likelihood of a stationary vector autoregressive-moving average model. *Journal of Time Series Analysis*, 23(4):473–486.

Meinshausen, N. and Bühlmann, P. (2006). Variable selection and high-dimensional graphs with the lasso. *Ann Stat*, 34:1436–1462.

Miyashiro, R. and Takano, Y. (2015a). Mixed integer second-order cone programming formulations for variable selection in linear regression. *European Journal of Operational Research*, 247(3):721–731.

Miyashiro, R. and Takano, Y. (2015b). Subset selection by Mallows' Cp: A mixed integer programming approach. *Expert Systems with Applications*, 42(1):325–331.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Monahan, J. F. (1984). A note on enforcing stationarity in autoregressive-moving average models. *Biometrika*, 71(2):403–404.

Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., and Talagala, T. S. (2020). Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1):86–92.

Nam, J., Kim, J., Mencía, E. L., Gurevych, I., and Fürnkranz, J. (2014). Large-scale multi-label text classification—revisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer.

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234.

Neath, A. A. and Cavanaugh, J. E. (2012). The bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2):199–203.

Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. International Biometric Soc 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210.

Newbold, P. (1974). The exact likelihood function for a mixed autoregressive-moving average process. *Biometrika*, 61(3):423–426.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Piccolo, D. (1982). The size of the stationarity and invertibility region of an autoregressive-moving average process. *Journal of Time Series Analysis*, 3(4):245–247.

Picinbono, B. and Benidir, M. (1986). Some properties of lattice autoregressive filters. *IEEE transactions on acoustics, speech, and signal processing*, 34(2):342–349.

Portet, S. (2020). A primer on model selection using the akaike information criterion. *Infectious Disease Modelling*, 5:111–128.

Prudêncio, R. B. and Ludermir, T. B. (2004). Meta-learning approaches to selecting time series models. *Neurocomputing*, 61:121–137.

Radchenko, P., Vasnev, A. L., and Wang, W. (2020). Too similar to combine? on negative weights in forecast combination. *On Negative Weights in Forecast Combination (July 1, 2020)*.

Reid, D. (1972). A comparison of forecasting techniques on economic time series. *Forecasting in Action. Operational Research Society and the Society for Long Range Planning*.

Robert, C., William, C., and Irma, T. (1990). Stl: A seasonal-trend decomposition procedure based on loess. *Journal of official statistics*, 6(1):3–73.

Sargan, J. D. and Bhargava, A. (1983). Maximum likelihood estimation of regression models with first order moving average errors when the root lies on the unit circle. *Econometrica: Journal of the Econometric Society*, pages 799–820.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, pages 461–464.

Shlien, S. (1985). A geometric description of stable linear predictive coding digital filters (corresp.). *IEEE Transactions on information theory*, 31(4):545–548.

Smith, J. and Wallis, K. F. (2009). A simple explanation of the forecast combination puzzle. *Oxford Bulletin of Economics and Statistics*, 71(3):331–355.

Stock, J. H. and Watson, M. W. (2004). Combination forecasts of output growth in a seven-country data set. *Journal of forecasting*, 23(6):405–430.

Stoica, P. and Selen, Y. (2004). Model-order selection: a review of information criterion rules. *IEEE Signal Processing Magazine*, 21(4):36–47.

Talagala, T. S., Hyndman, R. J., Athanasopoulos, G., et al. (2018). Meta-learning how to forecast time series. *Monash Econometrics and Business Statistics Working Papers*, 6:18.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tjøstheim, D. and Paulsen, J. (1983). Bias of some commonly-used time series estimates. *Biometrika*, 70(2):389–399.

Tusell, F. et al. (2011). Kalman filtering in r. *Journal of Statistical Software*, 39(2):1–27.

Wang, X., Smith, K., and Hyndman, R. (2006). Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13(3):335–364.

Wang, X., Smith-Miles, K., and Hyndman, R. (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing*, 72(10-12):2581–2594.

Widodo, A. and Budi, I. (2013). Model selection using dimensionality reduction of time series characteristics. In *International Symposium on Forecasting, Seoul, South Korea*, pages 57–118.

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342.

Yu, Y., Pedrycz, W., and Miao, D. (2014). Multi-label classification by exploiting label correlations. *Expert Systems with Applications*, 41(6):2989–3004.

Zhang, M.-L. and Zhou, Z.-H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Zhang, Y. and McLeod, A. I. (2006). Fitting ma (q) models in the closed invertible region. *Statistics & probability letters*, 76(13):1331–1334.

Zhao, B., Lu, H., Chen, S., Liu, J., and Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169.

Zhao, P. and Yu, B. (2006). On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.