COMPLEX-VALUED GROUP LASSO FOR TENSOR AUTOREGRESSIVE MODELS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Yang Yang Hu

February 2016

ABSTRACT

I will introduce a group lasso algorithm for complex variables and demonstrate its application to a novel time series model called tensor autoregression (T-AR). T-AR utilizes the t-product tensor operation on 3-dimensional tensors and models time series exhibiting seasonality and geometric trend. The tensor structure of T-AR enables historical information from a selection of lag durations to simultaneously affect the prediction of a single series. I will first introduce the topic of tensor computations and motivate the t-product and T-SVD manipulations. Then, I will derive the T-AR model from the t-product definition and discuss its properties and interpretation. Next, I will adapt a group lasso algorithm to complex-valued problems and derive the fast algorithm for lag selection in T-AR. Finally, I will conclude with simulation results.

BIOGRAPHICAL SKETCH

Yang Y. Hu (Michael) grew up in Maryland and earned his Bachelor of Arts degree in 2012 from Columbia University in computer science and statistics. He has previously worked with the Boeing Company's research and development team on optimization algorithms. Yang is currently a Master of Science degree candidate in statistics at Cornell University.

ACKNOWLEDGEMENTS

My graduate study at Cornell University has been a rewarding and transformative experience thanks to the generous support and positive influence of the graduate faculty and collaborators.

First, I would like to thank my advisor, Professor Martin Wells, for providing me the opportunity to write this thesis and supporting me throughout its progress. Professor Wells provided the insightful guidance that has helped me maintain the coherence of my work. He has encouraged me in times of doubt and kept me focused on my goals, and because of this I have grown as a student. I appreciate his patience with my progress and confidence in my ability.

I would also like to thank my committee member, Professor Charles Van Loan, for his interest in my project and for sparking my interest in matrix computations. My discussions with Professor Van Loan have enriched my perspective on my thesis topic. His course on matrix computations has given me the confidence to solve my technical problems and the language to articulate the ideas in my thesis.

Next, I would like to thank Professor Giles Hooker for guiding me through my coursework and for keeping me on track with academic requirements. Professor Hooker has been very accommodating of my academic interests and supportive of my goals. I also enjoyed taking his course on statistical computing.

Finally, I would like to thank the Advertising Sciences team at Yahoo Labs in Sunnyvale for a productive summer internship. In particular, I give credit to Dr. James Li for providing the groundwork that motivated the new ideas in this thesis. In addition, I would like to thank Dr. Suleyman Cetintas for providing his expertise on time series and machine learning methodology to help define the scope of our summer project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Decisions on how to represent information can profoundly impact a model and lead to a variety of data structures and operations suited to applications where relational structure is crucial to the analysis. In this thesis, we will explore the theme of data representation through the concrete examples of our two novel contributions. Our first contribution is a tensor autoregressive model that manipulates tensor data structures into a time series model for seasonality and geometric trend. Our second contribution is a general group lasso algorithm for complex-valued variables. At the end of our discussion on methodology, we will unite these two contributions into a single framework incorporating complex group lasso regularization into tensor autoregression.

In higher dimensions, tensor computations often reduce to structured matrix computations bearing meaningful interpretations. The tensor autoregressive (T-AR) model is our example of how an interpretation of a particular tensor operation, the t-product [12], can bridge a gap between distinct modes of thinking. We will show how the t-product can be utilized to construct a time series model which, by virtue of its tensor structure, is amenable to feature selection techniques originally designed for ordinary linear regression models. The T-AR framework models a specific time series phenomenon, possibly characteristic of web traffic volume data [15], where the series exhibits seasonality and geometric rates of change. The t-product tensor operation enables T-AR to capture these characteristics exactly.

Just as tensor representations reflect structure in scientific problems, complex analogues to real-valued phenomena can expose useful structures in the complex domain that are difficult to exploit in the real domain. Many of these useful complex analogues can be exposed by the discrete Fourier transform (DFT), which has wide applications in signal processing, data compression, and polynomial multiplication where analysis often simplifies in the complex-valued Fourier space. In the work relevant to this thesis, the DFT is used to simplify the implementation of T-AR by transforming the block-circulant structure of the t-product into a complex-valued block-diagonal structure.

Thus, the t-product is an example of a real matrix problem that is solved optimally in a complex space, and therefore the T-AR implementation motivates the need for complex-valued feature selection methods. To address this need, we present a complex group lasso algorithm for feature selection in general complex-valued settings. The contribution of this algorithm is independent of T-AR and applies to any complex-valued regression problem, not necessarily time series. Complex datasets arise in many areas of natural science, but they arise in T-AR due to the design of the methodology rather than the phenomenon under study. We hope this application demonstrates that it may be worthwhile to search for complex analogues to real data analysis problems if opportunities arise in the complex domain. In the case of T-AR, considering a complex representation presents the opportunity to implement a group lasso algorithm using simplified derivations that are more efficient in terms of memory and speed than the general, naïve implementation.

The thesis will proceed as follows. First, we will briefly review background on tensors, focusing on a few of the most common tensor decompositions. Having discussed this foundational work, we then compare the t-product to these existing methods in order to highlight the t-product's innovation in the tensor literature. Next, we incorporate the t-product into our novel T-AR time series model, derive a general group lasso algorithm in complex variables, and show how the T-AR model structure leads to a natural optimization of the general algorithm. Finally, we conclude with experimental results, all produced using the R statistical language, to justify the correctness of our methodology.

CHAPTER 2

BACKGROUND ON TENSORS

2.1 Tensors and Tensor Decompositions

As structured data becomes more abundant, people have utilized higher order tensors to represent the structural relationships within data [13, 14, 16, 12]. We define a tensor to be a multi-dimensional (multi-mode) array, consistent with the definition by Kilmer et al. [12]. We do not mean tensors in the physical sense that carries directional connotations; our tensors are simply multi-dimensional arrangements of data. A vector is a 1-tensor, a matrix is a 2-tensor, and data structures with 3 or more modes are higher-order tensors. In this thesis, we restrict our attention to 3-mode tensors. There are many tensor decompositions, and the most appropriate often depends on the nature of the application. We summarize two popular decompositions, Tucker and CANDECOMP/PARAFAC (CP), and briefly describe a statistical application of each.

Often, tensor operations do not manipulate the entirety of a tensor, but parts of a tensor. For example, a tensor operation may manipulate "slices" or "tubes" of a tensor, where a slice is the matrix obtained by allowing two modes to vary and a tube is the vector obtained by allowing one mode to vary. One of the most commonly used tensor operations is the **n-mode product**, which defines the multiplication of an n-mode tensor with either a matrix or a vector.

Definition 1. *Let $\mathcal{X} \in \mathbb{R}^{m_1 \times m_2 \times \ldots \times m_N}$ be a tensor and $V \in \mathbb{R}^{p \times m_n}$ be a matrix. The **n-mode product** of $\mathcal{X}$ with V is denoted $\mathcal{X} \times_n V$ and has each tube along the nth mode of $\mathcal{X}$ multiplied by V. If $\mathcal{Y} = \mathcal{X} \times_n V$, then the nth tube of $\mathcal{Y}$ is $Y_{(n)} = VX_{(n)}$, and $\mathcal{Y}$ has dimensions $m_1 \times m_2 \times \ldots \times m_{n-1} \times p \times m_{n+1} \times \ldots \times m_N$.*

The n-mode product leads to two widely used tensor decompositions, Tucker and CANDECOMP/PARAFAC (CP), which we will explain for the three dimensional case.

The Tucker decomposition factors a tensor into a core tensor with each mode transformed by a matrix. Suppose we have a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. According to the Tucker decomposition, there exists the factorization

$$\mathcal{X} = \mathcal{G} \times_1 A \times_2 B \times_3 C \tag{2.1}$$

$$= \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} \mathcal{G}_{pqr} a_p \circ b_q \circ c_r, \tag{2.2}$$

where $\circ$ denotes the vector outer product, a_p is the pth column of $A \in \mathbb{R}^{I \times P}$, b_q is the qth column of $B \in \mathbb{R}^{J \times Q}$, and c_r is the rth column of $C \in \mathbb{R}^{K \times R}$. In the forecasting domain, authors have used Tucker decomposition and the n-mode product for tensor-matrix multiplication to derive a multilinear dynamical system to predict future realizations of a tensor-valued sequence [20].

The CP decomposition can be seen as a special case of the Tucker decomposition where $\mathcal{G}$ is diagonal and the second mode dimensions of the outer matrices are equal, or $P = Q = R$. The CP decomposition factorizes a tensor into a sum of rank one components so that, given an N-mode tensor $\mathcal{X}$, the CP decomposition is

$$\mathcal{X} = \sum_{r=1}^{R} \lambda_r a_r^{(1)} \circ a_r^{(2)} \circ \ldots \circ a_r^{(N)}. \tag{2.3}$$

The challenge in computing either of these decompositions is determining the number of components R in the factorization. Given R, both the Tucker and CP decompositions can be computed using alternating least squares (ALS). The CP decomposition has been applied recently in neuroimaging analysis [24] for methodology that combines the Kronecker product and a CP decomposition-based rank definition to form a multilinear regression model with matrix input and scalar response. The authors' multilinear model fits far fewer parameters than an ordinary linear model would, yet their method achieves promising accuracy as the reduction exploits the two-dimensional structure of the data in its original matrix format.

2.2 T-product

Unlike previous tensor operations and decompositions that are motivated by specific applications, Kilmer and Martin [12] devised their **t-product** to exhibit familiar linear algebraic properties. The t-product is a linear operation that leads to the existence of tensor inverse, identity, and transpose. The t-product definition also leads to the authors' novel tensor decomposition called the T-SVD, which has been applied to image compression [12, 11, 10, 18, 7].

The t-product is composed of two sub-operations: matrix vectorization and block circulant. Matrix vectorization can be imagined as stacking the slices along the third mode ($X_k \in \mathbb{R}^{n\times p\times 1}$) of a 3-tensor into a block matrix vector.

Definition 2. *For $\mathcal{X} \in \mathbb{R}^{n\times p\times t}$, the **matrix vectorization** of $\mathcal{X}$, denoted $\mathrm{matvec}(\mathcal{X})$, is defined as*

$$\mathrm{matvec}(\mathcal{X}) = \begin{bmatrix} X_1 \\ \vdots \\ X_t \end{bmatrix} \in \mathbb{R}^{nt\times p}.$$

Matrix vectorization is undone by

$$\mathrm{fold}(\mathrm{matvec}(\mathcal{X})) = \mathcal{X}.$$

The block circulant of a tensor $\mathcal{X}$ creates a block circulant matrix by downshifting the matrix vectorization of $\mathcal{X}$ (per Definition 2) at each new column.

Definition 3. *For $\mathcal{X} \in \mathbb{R}^{n \times p \times t}$, the **block circulant** of $\mathcal{X}$, denoted $\mathrm{bcirc}(\mathcal{X})$, is defined as*

$$\mathrm{bcirc}(\mathcal{X}) = \begin{bmatrix} X_1 & X_t & \dots & X_2 \\ X_2 & X_1 & \dots & X_3 \\ \vdots & \vdots & \ddots & \vdots \\ X_t & X_{t-1} & \dots & X_1 \end{bmatrix} \in \mathbb{R}^{nt \times pt}.$$

Using the operations given by Definitions 2 and 3, we can now define the t-product.

Definition 4. *For 3-tensors $\mathcal{A} \in \mathbb{R}^{n \times p \times t}$ and $\mathcal{B} \in \mathbb{R}^{p \times \ell \times t}$, the **t-product** of $\mathcal{A}$ and $\mathcal{B}$ is written $\mathcal{A} * \mathcal{B} \in \mathbb{R}^{n \times \ell \times t}$, where $\mathrm{matvec}(\mathcal{A} * \mathcal{B})$ entails matrix multiplication of $\mathrm{bcirc}(\mathcal{A})$ with $\mathrm{matvec}(\mathcal{B})$. Thus,*

$$\mathcal{A} * \mathcal{B} = \mathrm{fold}\left(\begin{bmatrix} A_1 & A_t & \dots & A_2 \\ A_2 & A_1 & \dots & A_3 \\ \vdots & \vdots & \ddots & \vdots \\ A_t & A_{t-1} & \dots & A_1 \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_t \end{bmatrix} \right) \in \mathbb{R}^{nt \times \ell}, \tag{2.4}$$

where $\{A_k\}_{k=1}^t$ and $\{B_k\}_{k=1}^t$ are the slices along the third mode of $\mathcal{A}$ and of $\mathcal{B}$, respectively.

The t-product definition leads to definitions of identity, inverse, and transpose that are reminiscent of the definitions in matrix algebra.

Definition 5. *The **identity** tensor, $\mathcal{I} \in \mathbb{R}^{n \times n \times t}$, has I_n in the first slice and 0 in the others. Straightforward computation verifies that, for $\mathcal{A} \in \mathbb{R}^{n \times p \times t}$, $\mathcal{A} * \mathcal{I} = \mathcal{A}$ where $*$ denotes the t-product.*

Definition 6. *If $n = p$, then $\mathcal{A} \in \mathbb{R}^{n \times p \times t}$ is **invertible** if and only if there exists some $\mathcal{A}^{-1} \in \mathbb{R}^{n \times n \times t}$ such that*

$$\mathcal{A}^{-1} * \mathcal{A} = \mathcal{I}$$

and

$$\mathcal{A} * \mathcal{A}^{-1} = \mathcal{I}.$$

Definition 7. *For $\mathcal{A} \in \mathbb{R}^{n \times p \times t}$, the **transpose** of $\mathcal{A}$ is*

$$\mathcal{A}^T := \text{fold}\left(\begin{bmatrix} A_1^T \\ A_t^T \\ \vdots \\ A_2^T \end{bmatrix}\right) \in \mathbb{R}^{p \times n \times t}.$$

*$\mathcal{A}$ is **orthogonal** if $\mathcal{A}^T = \mathcal{A}^{-1}$.*

The **generalized inverse** of $\mathcal{A} \in \mathbb{R}^{n \times p \times t}$ is defined as

$$\mathcal{A}^{\dagger} := \text{fold}(A^{\dagger}_{1:n}) \in \mathbb{R}^{p \times n \times t}, \tag{2.5}$$

where $A^{\dagger}_{1:n}$ denotes the first n columns of $A^{\dagger} \in \mathbb{R}^{pt \times nt}$, which is the matrix Moore-Penrose Inverse of $A = \text{bcirc}(\mathcal{A})$. For any given $\mathcal{A}$, $\mathcal{A}^{\dagger}$ uniquely exists (due to the uniqueness of Moore-Penrose Inverse) and satisfies the following pseudo-inverse properties:

1. $\mathcal{A} * \mathcal{A}^{\dagger} * \mathcal{A} = \mathcal{A}$,
2. $\mathcal{A}^{\dagger} * \mathcal{A} * \mathcal{A}^{\dagger} = \mathcal{A}^{\dagger}$,

3. $(\mathcal{A} * \mathcal{A}^\dagger)^T = \mathcal{A} * \mathcal{A}^\dagger$, and

4. $(\mathcal{A}^\dagger * \mathcal{A})^T = \mathcal{A}^\dagger * \mathcal{A}$.

The t-product given in Definition 4 entails redundant calculations due to the data-sparse block-circulant matricization of $\mathcal{A}$. Hence, the authors implement the t-product by utilizing the block-diagonalization of a block-circulant matrix through discrete Fourier transformation. Let F_{n_3} be the n_3-by-n_3 square DFT matrix and $F_{n_3}^H$ its conjugate transpose. Then, a block circulant matrix $\mathrm{bcirc}(\mathcal{X})$ may be block-diagonalized by

$$(F_{n_3} \otimes I_{n_1}) \cdot \mathrm{bcirc}(\mathcal{X}) \cdot (F_{n_3}^H \otimes I_{n_2}) = \begin{bmatrix} \widetilde{X}_1 & & & \\ & \widetilde{X}_2 & & \\ & & \ddots & \\ & & & \widetilde{X}_{n_3} \end{bmatrix} \tag{2.6}$$

$$= \widetilde{X}. \tag{2.7}$$

Using this diagonalization strategy, it is more efficient to implement the t-product between tensors $\mathcal{X}$ and $\mathcal{Y}$ by

$$\mathcal{X} * \mathcal{Y} = \mathrm{fold}((F_{n_3}^H \otimes I_{n_1}) \cdot \widetilde{X} \cdot (F_{n_3} \otimes I_{n_2}) \cdot \mathrm{matvec}(\mathcal{Y})) \tag{2.8}$$

$$= \mathrm{fold}\left((F_{n_3}^H \otimes I_{n_1}) \cdot \begin{bmatrix} \widetilde{X}_1 & & & \\ & \widetilde{X}_2 & & \\ & & \ddots & \\ & & & \widetilde{X}_{n_3} \end{bmatrix} \begin{bmatrix} \widetilde{Y}_1 \\ \widetilde{Y}_2 \\ \vdots \\ \widetilde{Y}_{n_3} \end{bmatrix} \right), \tag{2.9}$$

giving us the entry-sparse complex analog to the data-sparse real matrix multiplication.

2.3 T-SVD

The block-diagonalized complex representation of the t-product leads naturally to the authors' derivation of the T-SVD, an analog of matrix singular value decomposition for tensors. Given a 3-tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the authors show that there exist $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{V}$ such that $\mathcal{U}$ and $\mathcal{V}$ are orthogonal and $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$, $\mathcal{S}$ being a tensor with tubes along a diagonal. Their proof is constructive and invokes the existence of ordinary matrix SVD's in the block-diagonal setting. Suppose $\mathcal{A}$ can be diagonalized by

$$(F_{n_3} \otimes I_{n_1}) \cdot \mathrm{bcirc}(\mathcal{A}) \cdot (F_{n_3}^H \otimes I_{n_2}) = \begin{bmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_{n_3} \end{bmatrix}. \tag{2.10}$$

Then,

$$\begin{bmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_{n_3} \end{bmatrix} = \begin{bmatrix} U_1 & & & \\ & U_2 & & \\ & & \ddots & \\ & & & U_{n_3} \end{bmatrix} \begin{bmatrix} \Sigma_1 & & & \\ & \Sigma_2 & & \\ & & \ddots & \\ & & & \Sigma_{n_3} \end{bmatrix} \begin{bmatrix} V_1^T & & & \\ & V_2^T & & \\ & & \ddots & \\ & & & V_{n_3}^T \end{bmatrix}. \tag{2.11}$$

Orthogonality of $\mathcal{U}$ and $\mathcal{V}$ can be shown through explicit computation. Thus, we see that the complex representation of the t-product greatly simplifies analysis as well as improves computational efficiency.

CHAPTER 3

TENSOR AUTOREGRESSION (T-AR)

Having established the definition and properties of the t-product, we now introduce the tensor autoregressive (T-AR) time series model with the motivation of modeling the characteristics of time series exhibiting seasonality and geometric trend. The model we introduce combines the sequential aspect of univariate autoregression with the multi-feature framework of multivariate regression. While the time series literature has offered methods for learning between related series in a multivariate forecasting setting [6, 2, 21], the literature on multivariate input frameworks for the univariate forecasting setting is limited to autoregressive exogenous (ARX) models.

T-AR introduces this multivariate input context to univariate forecasting so that forecasts for a single target series are formed using the information from various lagged periods. This modeling framework bridges the gap between dimension reduction techniques for regression and univariate time series forecasting, enabling our time series application of structured regularization methods formulated for regression problems. We begin with explanations of our model assumptions and how these assumptions lead to the T-AR formulation. We will state the specific time series characteristics that T-AR is most suited to capture and analyze how T-AR achieves its accuracy. In relation to this analysis, we give our interpretation of the T-AR parameters and argue that the parameters can offer insight into the extent to which data conforms to model assumptions. The basic formulation leads to a closed-form solution for the parameter that minimizes the Frobenius norm of estimation error. Although this solution is determined in the complex Fourier domain, we can show that real-valued parameter coefficients are always recovered. We conclude the chapter with implementation analysis and a proof to guarantee real parameter coefficients.

3.1 Model Formulation

T-AR is structured to predict one period forward at a time, using history from the current period and a number of previous periods as training data. The working principle of T-AR is that changes in the series of interest occur over the course of periods rather than individual time points. This understanding implies that the unit of observation is one period, so we consider our training data to be a collection of periods observed in the series. Let $\mathbf{y}$ be the time series of interest consisting of n periods, each of length t. We structure the response $\mathcal{Y} \in \mathbb{R}^{n \times 1 \times t}$ as a slice along the second mode, where the ith tube of $\mathcal{Y}$ (a tube of $\mathcal{Y}$ is indexed along its first mode and has length t) contains data from the ith most recent period of $\mathbf{y}$. The first tube contains the most recent period of $\mathbf{y}$, the second tube contains the second most recent period, and so on. An order $\mathbf{p}$ tensor autoregressive model captures the additive effects on $\mathcal{Y}$ from data lagged $\mathbf{p} = [\ell_1, \ell_2, \ldots, \ell_p]$ periods behind $\mathcal{Y}$.

According to these specifications, we construct the input tensor $\mathcal{X}$ with dimensions $\mathbb{R}^{n \times p \times t}$. The first slice of $\mathcal{X}$ consists of the ℓ_1 period-lag of $\mathcal{Y}$, the second slice of $\mathcal{X}$ consists of the ℓ_2 period-lag of $\mathcal{Y}$, and so on. Overall, this formulation induces the following parameterization of the T-AR model, which is illustrated in Figure 3.1:

$$\mathcal{Y} = \mathcal{X} * \mathcal{B} + \mathcal{E}, \tag{3.1}$$

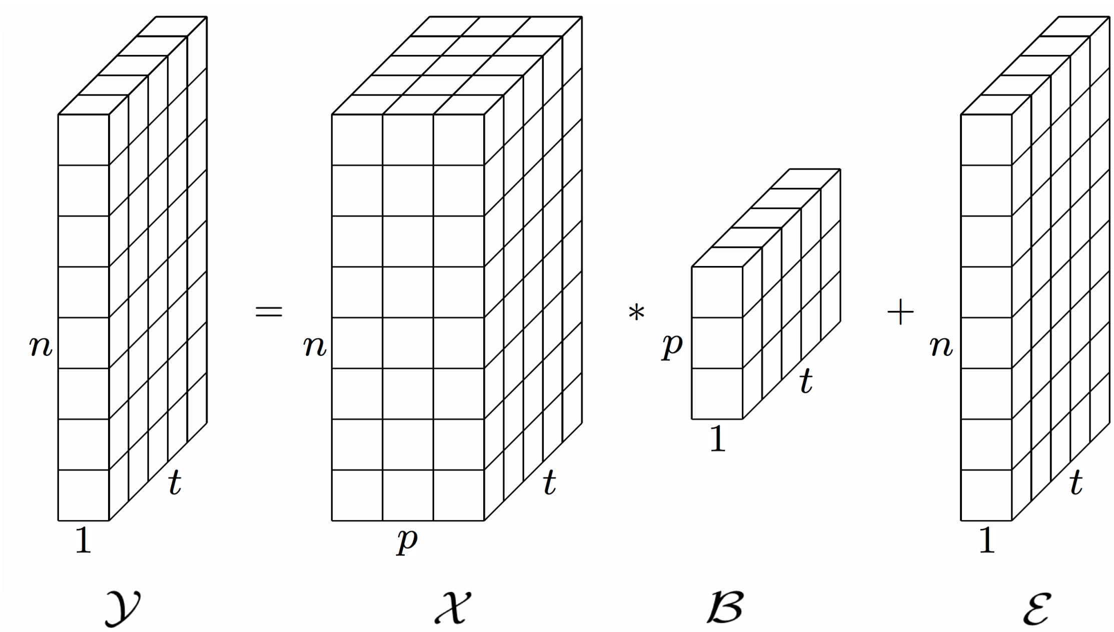

Figure 3.1: Diagram of the Tensor Autoregressive model.

where

$\mathcal{Y} \in \mathbb{R}^{n\times 1\times t}$ is the response slice,

$\mathcal{X} \in \mathbb{R}^{n\times p\times t}$ is the input tensor,

$\mathcal{B} \in \mathbb{R}^{p\times 1\times t}$ is the parameter slice, and

$\mathcal{E} \in \mathbb{R}^{n\times 1\times t}$ is the residual slice.

Thus, T-AR organizes a time series into a structure resembling an ordinary linear regression model. We treat the first mode of the data tensor $\mathcal{X}$ as the sample size dimension, where periods of data accumulate. In principle, if the time series were truly periodic, then a single parameter $\mathcal{B}$ would generate every period in the sample. This assumption imposes a restriction that, in order to stay consistent with the interpretation of $\mathcal{B}$, any features along the second mode of $\mathcal{X}$ must also be periodic. However, treating T-AR as an extension of ordinary autoregression, this is not a problem if the features are simply lagged periods of the series itself. The third mode simply accommodates the temporal aspect of the problem.

3.2 Parameter Estimation and Interpretation

We now derive the closed-form solution of the T-AR model parameter and prove its optimality in terms of least Frobenius norm of error.

Theorem 1. *Let $\widetilde{Y}$ and $\widetilde{X}$ be Fourier domain representations of $\mathcal{Y}$ and $\mathcal{X}$ respectively, and let $\widetilde{B}$ be the T-AR parameter in the Fourier domain. Let † be the generalized inverse provided in Equation (2.5), and let k index a block. Then*

$$\widehat{\widetilde{\mathcal{B}}}_k = (\widetilde{X}_k^T \widetilde{X}_k)^\dagger \widetilde{X}_k^T \widetilde{Y}_k$$

is the kth complex slice of the ***least Frobenius norm*** *estimator of $\mathcal{B}$, and $\mathcal{B}$ minimizes $||\mathcal{E}||_F^2$.*

Proof. We first show how $||\mathcal{E}_F^2||$ decouples along the third mode.

$$\begin{aligned}
\|\mathcal{Y} - \mathcal{X} * \mathcal{B}\|_F^2 &= \|\text{matvec}(\mathcal{Y} - \mathcal{X} * \mathcal{B})\|_F^2 &(3.2)\\
&= \|(F_n \otimes I_t)\text{matvec}(\mathcal{Y} - \mathcal{X} * \mathcal{B})\|_F^2 &(3.3)\\
&= \|(F_n \otimes I_t)(\text{matvec}(\mathcal{Y}) - \text{bcirc}(\mathcal{X})\text{matvec}(\mathcal{B}))\|_F^2 &(3.4)\\
&= \left\|\text{matvec}(\widetilde{\mathcal{Y}}) - \textbf{diag}(\widetilde{X}_1, \ldots, \widetilde{X}_t)\text{matvec}(\widetilde{\mathcal{B}})\right\|_F^2 &(3.5)\\
&= \sum_{k=1}^{t} \left\|\widetilde{\mathcal{Y}}[:,:,k] - \widetilde{X}_k \widetilde{\mathcal{B}}[:,:,k]\right\|_F^2, &(3.6)
\end{aligned}$$

where Equation (3.4) follows from the definition of the t-product and Equation (3.5) follows from

$$\text{bcirc}(\mathcal{X})\text{matvec}(\mathcal{B}) = (F_{n_3}^H \otimes I_{n_2}) \cdot \textbf{diag}(\widetilde{X}_1, \ldots, \widetilde{X}_t) \cdot (F_{n_3} \otimes I_{n_1}) \cdot \text{matvec}(\mathcal{B}), \tag{3.7}$$

which implies that

$$(F_{n_3} \otimes I_{n_1}) \cdot \mathrm{bcirc}(\mathcal{X})\mathrm{matvec}(\mathcal{B}) = \textbf{diag}(\widetilde{X}_1, \dots, \widetilde{X}_t) \cdot (F_{n_3} \otimes I_{n_1}) \cdot \mathrm{matvec}(\mathcal{B}) \tag{3.8}$$

$$= \textbf{diag}(\widetilde{X}_1, \dots, \widetilde{X}_t)\mathrm{matvec}(\widetilde{\mathcal{B}}). \tag{3.9}$$

We see that minimizing the criterion in Equation (3.6) amounts to minimizing each term of the sum, but each term in Equation (3.6) is a matrix least squares criterion, for which we know the optimal solutions to be

$$\widehat{\widetilde{\mathcal{B}}}[:,:,k] = (\widetilde{X}_k^T \widetilde{X}_k)^{-1} \widetilde{X}_k^T \widetilde{\mathcal{Y}}[:,:,k], \quad k = 1, \dots, t. \tag{3.10}$$

Taking inverse FFT's, we get

$$\widehat{\mathcal{B}}[i,j,:] = \mathrm{iFFT}(\widehat{\widetilde{\mathcal{B}}}[i,j,:]), \tag{3.11}$$

which give us the solution to the real T-AR estimation problem and are the quantities we use for forecasting. □

To discuss the interpretation of the coefficients in $\mathcal{B}$, we consider the matrix vectorization of Equation (3.1),

$$\begin{bmatrix} \widehat{Y}_1 \\ \widehat{Y}_2 \\ \vdots \\ \widehat{Y}_t \end{bmatrix} = \begin{bmatrix} X_1 & X_t & \dots & X_2 \\ X_2 & X_1 & \dots & X_3 \\ \vdots & \vdots & \vdots & \vdots \\ X_t & X_{t-1} & \dots & X_1 \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_t \end{bmatrix}, \tag{3.12}$$

where

$$\begin{aligned} \widehat{Y}_1 &= X_1 B_1 + X_t B_2 + \dots + X_2 B_t \\ \widehat{Y}_2 &= X_2 B_1 + X_1 B_2 + \dots + X_3 B_t \\ &\vdots \\ \widehat{Y}_t &= X_t B_1 + X_{t-1} B_2 + \dots + X_1 B_t, \end{aligned}$$

$$B_k = [\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}], \;\; k = 1, \ldots, t.$$

Our interpretation hinges on the periodicity of the series, which implies that the value indexed by X_t is both $t-1$ time steps ahead as well as one time step behind X_1. For concreteness (and alluding to our application), let t range from 1 to 7, indexing the days of week from Sunday to Saturday. Then, we interpret the first column of $\mathrm{bcirc}(\mathcal{X})$ to be the same days of week as $\mathcal{Y}$ lagged $\mathbf{p}$ weeks behind, the second column to be one day prior and lagged $\mathbf{p}$ weeks behind, and so on.

In order to more clearly convey how T-AR operates, in the following discussion we analyze the simplest case of regressing the current period against one period prior. In this simplification, each variable in Equation (3.12) would be scalar. The t-product imposes the restriction that each column of the input circulant matrix in Equation (3.12) is multiplied by the same parameter value. This restriction prevents overfitting whenever the data strongly exhibits patterns that T-AR is designed to model.

Now, suppose the previous period's observations are $X_1, X_2, \ldots, X_t$. If the series were seasonal, then the current observations would be $Y_1 = X_1, Y_2 = X_2, \ldots, Y_t = X_t$, and the solution to the regression problem given in Equation (3.12) is exactly determined to be $B_1 = 1, B_2 = \ldots = B_t = 0$. Next, suppose a geometric trend begins in the previous period. Assume, without loss of generality, that $X_1 = 1$ and that X_1 is compounded at rate $c > 0$ so that the previous period is $X_1 = 1, X_2 = c, X_3 = c^2, \ldots, X_t = c^{t-1}$. If the trend continues into the current period, then we would observe $Y_1 = c^t, Y_2 = c^{t+1}, \ldots, Y_t = c^{2t-1}$, and $B_1 = c^t, B_2 = \ldots = B_t = 0$ would capture the geometric trend exactly.

The first row of plots in Figure 3.2 illustrates how this parameterization can

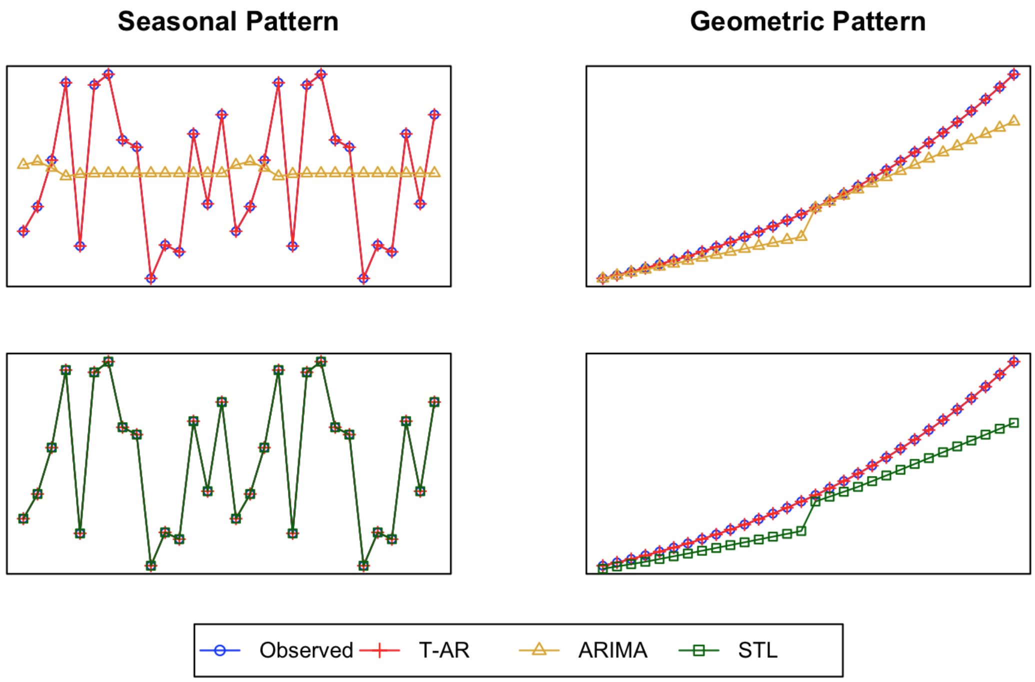

Figure 3.2: T-AR demonstrates advantages over ARIMA and STL.

outperform the autoregressive integrated moving average (ARIMA) model. In all of our experiments, we use the `forecast` R package's implementation of ARIMA [9], which optimizes ARIMA parameters automatically. In Figure 3.2, we see that the generality of ARIMA fails to capture the seasonality and geometric trend that T-AR captures precisely through the restriction in its parameterization. We also compare T-AR against an implementation of seasonal trend decomposition by Loess (STL) [4] that uses ARIMA as its forecasting model. The plots in the second row of Figure 3.2 show that STL is competitive with T-AR in modeling seasonality but still cannot capture geometric trend. T-AR captures both patterns exactly without any modification.

The role of the other circulant columns corresponding to $B_2, \ldots, B_t$ in Equation (3.12) is to stabilize T-AR's forecasting accuracy when the model frequency

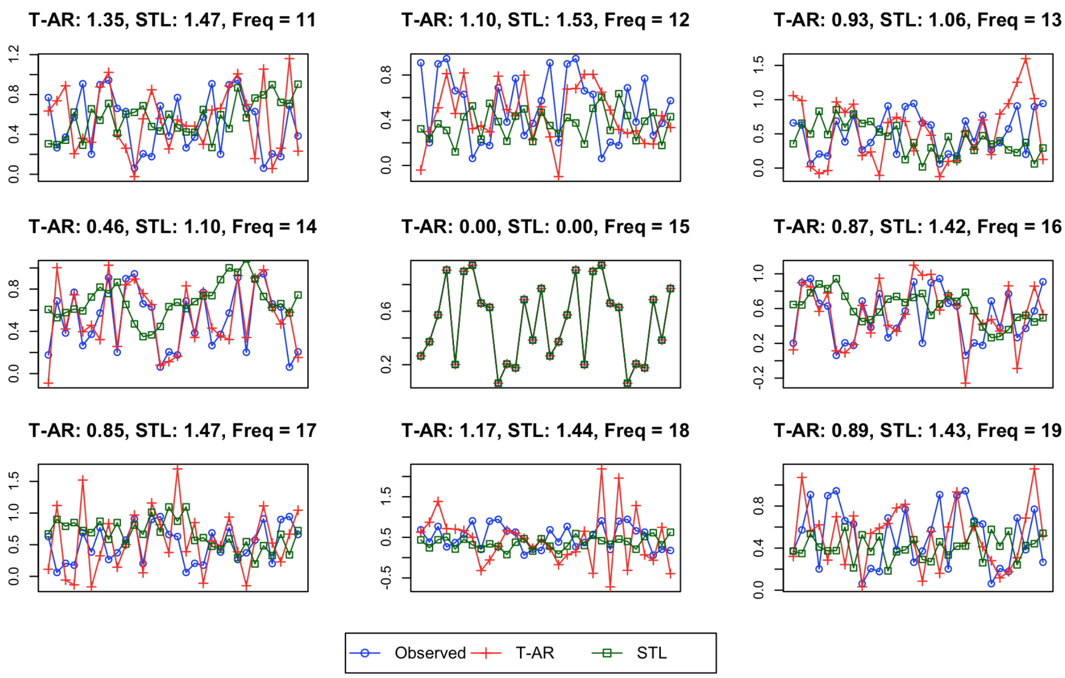

Figure 3.3: T-AR is more resistant to frequency misspecification than STL.

(the value set for t) is misspecified, or when the data's periodicity is irregular. Simulation results in Figure 3.3 demonstrate that this stabilizing effect enables T-AR to outperform STL on synthetic seasonal data at varying degrees of model misspecification. The titles of the plots in Figure 3.3 indicate average percentage error after the corresponding model name. We simulated a periodic series that repeats every 15 time points and compared the forecasting behaviors of T-AR and STL at various frequency settings, correctly and incorrectly specified. Figure 3.3 (center) shows that at the correct frequency specification, both T-AR and STL forecast properly. However, as the frequency is incrementally misspecified, T-AR retains more accuracy than STL does.

By plotting the mean magnitudes of the entries of T-AR parameter estimates in Figure 3.4, we see that the magnitudes of the parameter entries justify our analytical understanding of the T-AR parameters. In the ideal situation where the

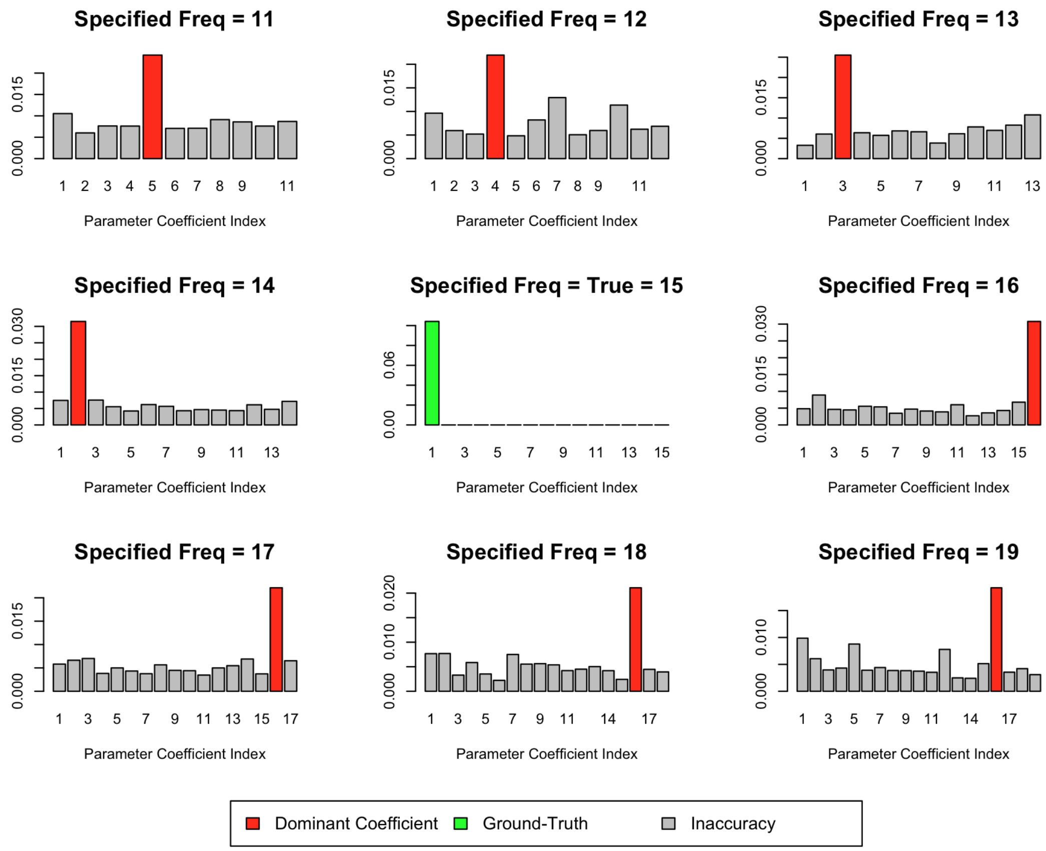

Figure 3.4: Colored bar indicates the circulant column providing most accuracy.

data is perfectly periodic and the model frequency is correctly specified (center of Figure 3.4), the first entry of the parameter is dominant while the rest are nearly zero. As the frequency specification varies, the dominant entry shifts along the indexes of the parameter, decreasing in magnitude as the specification moves further from the true frequency. Thus, the magnitude of the dominant entry and the asymmetry of its size relative to the other entries are indications of how well the data conforms to model assumptions. In addition, since the dominant entry shifts by one for each incremental misspecification, the location of the dominant entry relative to the first index also suggests how to correct the frequency misspecification.

As an aside, we also briefly discuss the "intercept" T-AR feature, which we find improves prediction performance in real data applications. The intercept is designed to compensate the T-AR forecasting model for systematic departures from T-AR assumptions (for this reason, we find the intercept overparameterizes the model in simulation studies). The intercept is constant throughout an application; hence, it may be treated as a periodic feature. Our analysis of the T-AR intercept slice begins with the matrix vectorization of Equation (3.1),

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_t \end{bmatrix} = \begin{bmatrix} X_1 & X_t & \dots & X_2 \\ X_2 & X_1 & \dots & X_3 \\ \vdots & \vdots & \vdots & \vdots \\ X_t & X_{t-1} & \dots & X_1 \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_t \end{bmatrix} + \operatorname{matvec}(\mathcal{E}), \tag{3.13}$$

and highlights the residual term $\operatorname{matvec}(\mathcal{E})$. If $\operatorname{matvec}(\mathcal{E})$ exhibits a predictable pattern, we attempt to model the residual behavior using a substitution for $\operatorname{matvec}(\mathcal{E})$ that leads to

$$\begin{aligned} \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_t \end{bmatrix} = \begin{bmatrix} X_1 & X_t & \dots & X_2 \\ X_2 & X_1 & \dots & X_3 \\ \vdots & \vdots & \vdots & \vdots \\ X_t & X_{t-1} & \dots & X_1 \end{bmatrix} \cdot \begin{bmatrix} \beta_{1,1} \\ \beta_{1,2} \\ \vdots \\ \beta_{1,t} \end{bmatrix} + \\ \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \cdot \begin{bmatrix} \beta_{0,1} \\ \beta_{0,2} \\ \vdots \\ \beta_{0,t} \end{bmatrix} . \end{aligned} \tag{3.14}$$

Putting $B_k = [\beta_{0k}, \beta_{1k}]^T$, we derive the intercept T-AR model with the matrix vectorized expression

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_t \end{bmatrix} = \begin{bmatrix} [1, X_1] & [0, X_t] & \dots & [0, X_2] \\ [0, X_2] & [1, X_1] & \dots & [0, X_3] \\ \vdots & \vdots & \vdots & \vdots \\ [0, X_t] & [0, X_{t-1}] & \dots & [1, X_1] \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_t \end{bmatrix}. \tag{3.15}$$

Thus, we implement the intercept slice by writing the vector $[1, 0, 0, \dots, 0] \in \mathbb{R}^t$ into each index along the first mode of the first second-mode slice of $\mathcal{X}$, which is convolved via $\mathrm{circ}([1, 0, 0, \dots, 0]^T)$ into the intercept slice.

3.3 Implementation and Scalability

Our implementations of T-AR algorithms are based on Fourier block-diagonalization of the block-circulant structure, as given in Equation (3.7). This transformation circumvents explicit construction of the block-circulant matrix.

Algorithm 1 states the procedure for estimating $\widehat{\mathcal{B}}$ given $\mathcal{X}$ and $\mathcal{Y}$, where p is the number of entries in the vector of period-lag specifications $\mathbf{p}$. Assuming that Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (iFFT) execute in $O(t \log t)$ time and that the Moore-Penrose pseudo-inverse executes in $O(p^3)$ time, the first outer loop executes in $O(npt \log t + nt \log t)$ time, the second in $O(t(p^3 + 2np^2 + np))$ time, and the third in $O(pt \log t)$ time. The prediction step is essentially a t-product between the estimated parameter and the most recent period of the data. The runtime complexity of prediction, whose procedure is stated in Algorithm 2, is $O(2pt \log t + pt + nt \log t)$. The memory complexities of these algorithms come predominantly from allocating space for the data arrays.

Input: $\mathcal{X} \in \mathbb{R}^{n \times p \times t}$, $\mathcal{Y} \in \mathbb{R}^{n \times 1 \times t}$
for $i = 1, \ldots, n$ **do**
  for $j = 1, \ldots, p$ **do**
    $\widetilde{\mathcal{X}}[i, j, :] = \text{FFT}(\mathcal{X}[i, j, :])$
  end
  $\widetilde{\mathcal{Y}}[i, 1, :] = \text{FFT}(\mathcal{Y}[i, 1, :])$
end
for $k = 1, \ldots, t$ **do**
  $\widetilde{\mathcal{B}}[:, 1, k] = (\widetilde{X}_k^T \widetilde{X}_k)^\dagger \widetilde{X}_k^T \widetilde{Y}_k$
end
for $j = 1, \ldots, p$ **do**
  $\widehat{\mathcal{B}}[j, 1, :] = \text{iFFT}(\widetilde{\mathcal{B}}[j, 1, :])$
end
Output: $\widehat{\mathcal{B}} \in \mathbb{R}^{p \times 1 \times t}$

Algorithm 1: Estimating $\widehat{\mathcal{B}}$ given $\mathcal{X}$ and $\mathcal{Y}$.

Input: $\mathcal{X} \in \mathbb{R}^{1 \times p \times t}$, $\widehat{\mathcal{B}} \in \mathbb{R}^{p \times 1 \times t}$
for $j = 1, \ldots, p$ **do**
  $\widetilde{\mathcal{X}}[1, j, :] = \text{FFT}(\mathcal{X}[1, j, :])$
end
for $j = 1, \ldots, p$ **do**
  $\widetilde{\widehat{\mathcal{B}}}[j, 1, :] = \text{FFT}(\widehat{\mathcal{B}}[j, 1, :])$
end
for $k = 1, \ldots, t$ **do**
  $\widetilde{\mathcal{Y}}[:, 1, k] = \widetilde{X}_k \widetilde{\widehat{B}}_k$
end
for $i = 1, \ldots, n$ **do**
  $\widehat{\mathcal{Y}}[i, 1, :] = \text{iFFT}(\widetilde{\mathcal{Y}}[i, 1, :])$
end
Output: $\widehat{\mathcal{Y}} \in \mathbb{R}^{1 \times 1 \times t}$

Algorithm 2: Predicting $\widehat{\mathcal{Y}}$ using $\widehat{\mathcal{B}}$ and the current period.

3.4 Proof of Real Estimation

Although we solve the T-AR problem in the Fourier space, we can prove that closed-form solutions for T-AR are always real-valued. The proof will exploit symmetries in the conjugacy patterns within the DFT matrix. In order to illustrate the intuition, we first give the proof for the simple one period training window, one lag, frequency $= t$ case, and then we extend to the general case with no restriction on the number of periods in the training window, denoted n, or number of lags in the feature mode, denoted p.

In the simple setting, our data are the input $x \in \mathbb{R}^t$ and response $y \in \mathbb{R}^t$ from two distinct periods in a single univariate time series. The T-AR algorithm treats the vectors x and y as tubes of $\mathbb{R}^{1\times1\times t}$ tensors and transforms the input and response by

$$\widetilde{x} = F_t x \tag{3.16}$$

$$\widetilde{y} = F_t y, \tag{3.17}$$

where the DFT matrix F_t is defined as

$$F_t = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_t & \omega_t^2 & \omega_t^3 & \dots & \omega_t^{t-1} \\ 1 & \omega_t^2 & \omega_t^4 & \omega_t^6 & \dots & \omega_t^{2(t-1)} \\ 1 & \omega_t^3 & \omega_t^6 & \omega_t^9 & \dots & \omega_t^{3(t-1)} \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & \omega_t^{t-1} & \omega_t^{2(t-1)} & \omega_t^{3(t-1)} & \dots & \omega_t^{(t-1)^2} \end{bmatrix} \tag{3.18}$$

and where $\omega_t = \exp(-2\pi i/t) = \cos(2\pi/t) - i\sin(2\pi/t)$. From here on, we will let F_j denote the jth row of the DFT matrix in Equation (3.18).

Before deriving results, we first state some useful facts:

1. $\omega_t^t = 1$. When t is even, $\omega_t^{\frac{t}{2}} = -1$.

   Proof. Follows from definition. □

2. If $m + n = t$, then $\omega_t^m = \overline{\omega_t^n}$.

   Proof. Putting $n = t - m$, we see

$$\omega_t^n = \exp(-2\pi i(t-m)/t) \tag{3.19}$$

$$= \cos(2\pi(t-m)/t) - i\sin(2\pi(t-m)/t) \tag{3.20}$$

$$= \cos(2\pi - 2\pi m/t) - i\sin(2\pi - 2\pi m/t) \tag{3.21}$$

$$= \cos(2\pi m/t) + i\sin(2\pi m/t) \tag{3.22}$$

$$= \overline{\omega_t^m}. \tag{3.23}$$

   □

3. $\overline{x}\,\overline{y} = \overline{xy}$.

   Proof. Let $x = a + ib$ and $y = c + id$. Then,

$$\overline{x}\,\overline{y} = (a - ib)(c - id) \tag{3.24}$$

$$= ac - i(bc + ad) - bd \tag{3.25}$$

$$= (ac - bd) - i(bc + ad) \tag{3.26}$$

$$= \overline{(a + ib)(c + id)} \tag{3.27}$$

$$= \overline{xy}. \tag{3.28}$$

   □

4. If $x = \overline{y}$, then $x^{-1} = \overline{y^{-1}}$.

Proof. Let $x = a + ib$ and $y = a - ib$. Then, we have

$$x^{-1} = \frac{1}{a+ib}\frac{a-ib}{a-ib} \tag{3.29}$$

$$= \frac{a-ib}{a^2+b^2}, \tag{3.30}$$

$$y^{-1} = \frac{1}{a-ib}\frac{a+ib}{a+ib} \tag{3.31}$$

$$= \frac{a+ib}{a^2+b^2} \tag{3.32}$$

$$\implies x^{-1} = \overline{y^{-1}}. \tag{3.33}$$

□

With these facts available, we proceed to derive the results on the T-AR estimation outcome.

Lemma 1. *$F_j^T x$ is conjugate to $F_{t-j+2}^T x$ for all $x \in \mathbb{R}^t$, for $j = 2, \ldots, \lfloor \frac{t}{2} \rfloor$.*

Proof. First, write

$$F_j^T x = x_1 + \sum_{k=1}^{t-1} \omega_t^{k(j-1)} x_{k+1}, \tag{3.34}$$

$$F_{t-j+2}^T x = x_1 + \sum_{k=1}^{t-1} \omega_t^{k(t-j+1)} x_{k+1}. \tag{3.35}$$

The real parts of Equations (3.34) and Equations (3.35) are equal by the following argument,

$$\mathrm{Re}(F_j^T x) = x_1 + \sum_{k=1}^{t-1} \cos(\frac{2\pi k(j-1)}{t}) x_{k+1} \tag{3.36}$$

$$= x_1 + \sum_{k=1}^{t-1} \cos(-\frac{2\pi k(j-1)}{t}) x_{k+1} \tag{3.37}$$

$$= x_1 + \sum_{k=1}^{t-1} \cos(\frac{2\pi k(t-j+1)}{t}) x_{k+1} \tag{3.38}$$

$$= \mathrm{Re}(F_{t-j+2}^T x). \tag{3.39}$$

Likewise, the imaginary parts of Equations (3.34) and Equations (3.35) are opposites by a similar argument,

$$\mathrm{Im}(F_j^T x) = \sum_{k=1}^{t-1} \sin(\frac{2\pi k(j-1)}{t}) x_{k+1} \tag{3.40}$$

$$= \sum_{k=1}^{t-1} -\sin(-\frac{2\pi k(j-1)}{t}) x_{k+1} \tag{3.41}$$

$$= -\sum_{k=1}^{t-1} \sin(\frac{2\pi k(t-j+1)}{t}) x_{k+1} \tag{3.42}$$

$$= -\mathrm{Im}(F_{t-j+2}^T x). \tag{3.43}$$

□

Using this result, we prove the next result.

Lemma 2. *For all* $x, y \in \mathbb{R}^t$, $x^T F_j F_j^T y = \overline{x^T F_{t-j+2} F_{t-j+2}^T y}$.

Proof. Using Lemma 1 and Fact 3, we have

$$x^T F_j F_j^T y = (\overline{x^T F_{t-j+2}})(\overline{F_{t-j+2}^T y}) \tag{3.44}$$

$$= \overline{x^T F_{t-j+2} F_{t-j+2}^T y}. \tag{3.45}$$

□

The least squares problem is solved independently for each slice along the third mode. According to Algorithm 1, we have that the Fourier representation of the parameter is

$$\widetilde{B} = \begin{bmatrix} (x^T F_1 F_1^T x)^{-1} x^T F_1 F_1^T y \\ (x^T F_2 F_2^T x)^{-1} x^T F_2 F_2^T y \\ (x^T F_3 F_3^T x)^{-1} x^T F_3 F_3^T y \\ (x^T F_4 F_4^T x)^{-1} x^T F_4 F_4^T y \\ \vdots \\ (x^T F_t F_t^T x)^{-1} x^T F_t F_t^T y \end{bmatrix}. \tag{3.46}$$

Thus, the transformed parameter vector has conjugate pairs at indices j and $t-j+2$ for for $j = 2, \ldots, \lfloor \frac{t}{2} \rfloor$ (notice the first entry is always real). Now, we can summarize all the results.

Theorem 2. *For input $x \in \mathbb{R}^{1\times 1\times t}$ and response $y \in \mathbb{R}^{1\times 1\times t}$, the estimated T-AR parameter coefficients are real-valued.*

Proof. With the parameter stated in Equation (3.46), it remains to show that applying the inverse DFT to the complex parameter results in a real solution. According to the algorithm, we have

$$\text{iFFT}(\widetilde{B}) \propto F^H \widetilde{B} \tag{3.47}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \overline{\omega_t} & \overline{\omega_t^2} & \overline{\omega_t^3} & \ldots & \overline{\omega_t^{t-1}} \\ 1 & \overline{\omega_t^2} & \overline{\omega_t^4} & \overline{\omega_t^6} & \ldots & \overline{\omega_t^{2(t-1)}} \\ 1 & \overline{\omega_t^3} & \overline{\omega_t^6} & \overline{\omega_t^9} & \ldots & \overline{\omega_t^{3(t-1)}} \\ \vdots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & \overline{\omega_t^{t-1}} & \overline{\omega_t^{2(t-1)}} & \overline{\omega_t^{3(t-1)}} & \ldots & \overline{\omega_t^{(t-1)^2}} \end{bmatrix} \begin{bmatrix} (x^T F_1 F_1^T x)^{-1} x^T F_1 F_1^T y \\ (x^T F_2 F_2^T x)^{-1} x^T F_2 F_2^T y \\ (x^T F_3 F_3^T x)^{-1} x^T F_3 F_3^T y \\ (x^T F_4 F_4^T x)^{-1} x^T F_4 F_4^T y \\ \vdots \\ (x^T F_t F_t^T x)^{-1} x^T F_t F_t^T y \end{bmatrix}. \tag{3.48}$$

We know the parameter has conjugate pairs at j and $t-j+2$, so call them B_j and $\overline{B_j}$. B_j is multiplied with $\overline{\omega_t^{(k-1)(j-1)}}$ and $\overline{B_j}$ with $\overline{\omega_t^{(k-1)(t-j+1)}}$, where k is the row of the inverse Fourier matrix, F^H. But from Fact 2, $\overline{\omega_t^{(k-1)(j-1)}}$ and $\overline{\omega_t^{(k-1)(t-j+1)}}$ are a conjugate pair, and using Fact 3, $B_j \cdot \overline{\omega_t^{(k-1)(j-1)}}$ and $\overline{B_j} \cdot \overline{\omega_t^{(k-1)(t-j+1)}}$ are conjugate. Therefore, the inner product between every row of the inverse Fourier matrix with the Fourier-domain coefficient vector is a sum of real values and conjugate pairs. The imaginary components cancel, and the end result is real. $\square$

The extension of Theorem 2 to the general case begins with an inspection of the ordinary least squares subproblem for the kth slice. According to Algorithm (1), the real data are $Y_i \in \mathbb{R}^{1\times 1\times t}$ and $x_{i,j} \in \mathbb{R}^{1\times 1\times t}$, for $i = 1, \dots, n$ and $j = 1, \dots, p$. The complex transformations are

$$\widetilde{Y}_k = \begin{bmatrix} F_k^T Y_1 \\ \vdots \\ F_k^T Y_n \end{bmatrix} \tag{3.49}$$

and

$$\widetilde{X}_k = \begin{bmatrix} x_{1,1}^T F_k & x_{1,2}^T F_k & \dots & x_{1,p}^T F_k \\ x_{2,1}^T F_k & x_{2,2}^T F_k & \dots & x_{2,p}^T F_k \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1}^T F_k & x_{n,2}^T F_k & \dots & x_{n,p}^T F_k \end{bmatrix}. \tag{3.50}$$

For each slice k, T-AR estimates the parameter for each of p features at the kth index by solving the normal equation

$$\widetilde{B}_k = (\widetilde{X}_k^T \widetilde{X}_k)^{-1} \widetilde{X}_k^T \widetilde{Y}_k. \tag{3.51}$$

Now, each entry of every matrix in the right-hand side of Equation (3.51) is an expression of form $x^T F_k F_k^T y$ as in Lemma 2. Hence, $\widetilde{B}_k$ is either real or is conjugate with $\widetilde{B}_{t-k+2}$. Using the similar reasoning as in the proof of Theorem 2, the proof of real estimation in the general matrix case follows as well.

CHAPTER 4
COMPLEX GROUP LASSO

Extending the group lasso [23] to T-AR illustrates how the t-product can provide a natural mechanism by which regression thinking can flow into a time series modeling framework. Indeed, structured regularization for time series modeling is currently an active area of research aimed at solving problems of overparameterization in conventional time series models [1, 3]. Recent regularization methods for time series have almost exclusively addressed real-valued problems, and complex-valued regularization is less studied as it normally arises due to the complex-valued phenomena under study [17]. Unlike previous work on complex-valued regression methodology, our complex-valued problem is an outcome of the T-AR implementation and not an outcome of the time series phenomenon.

In this chapter, we contribute our adaptation of the group lasso framework to complex-valued problems by extending a block-coordinate descent algorithm for group lasso to complex variables. Our complex group lasso (CGL) algorithm imposes structured sparsity regularization to any complex-valued regression problem, and our T-AR model is one application. Our work utilizes a unique type of derivative called the Wirtinger derivative, and we begin this chapter with an introduction to Wirtinger differentiation. Then, we derive a block-coordinate descent algorithm for our complex group lasso problem and show how the T-AR model structure simplifies the derivations of the general algorithm.

4.1 Relevant Background from Wirtinger Differentiation

The complex group lasso unconstrained objective function, despite being a function of complex variables, is real-valued. To extend the group lasso to complex variables, we therefore first discuss differentiation of real functions of complex variables, which differs from complex differentiation. Complex differentiability is a strong condition, requiring the function of interest to satisfy Cauchy-Riemann equations that pose restrictions on partial derivatives of the function's real and imaginary components. However, for real functions of complex variables, both real and imaginary components of the variable contribute to the real part of the function. The appropriate differential operator for these functions is the Wirtinger derivative, named after Wilhelm Wirtinger who introduced the ideas in 1927.

Consider, for example, a complex variable $z = a + ib$, whose norm is defined to be

$$f(z) = \|z\|^2 \tag{4.1}$$

$$= \overline{z} \cdot z. \tag{4.2}$$

The function $f(z)$ does not have a complex derivative because it does not satisfy the Cauchy-Riemann equations. However, the Wirtinger derivatives of $f(z)$ with respect to z and to $\overline{z}$ are defined by partial derivatives of f (as opposed to partial derivatives of real and imaginary components of f as for the complex derivative),

$$\frac{\partial f}{\partial z} = \frac{1}{2} \cdot \Big(\frac{\partial f}{\partial a} - i\frac{\partial f}{\partial b}\Big), \tag{4.3}$$

and

$$\frac{\partial f}{\partial \overline{z}} = \frac{1}{2} \cdot \left(\frac{\partial f}{\partial a} + i\frac{\partial f}{\partial b}\right). \tag{4.4}$$

Furthermore, the following important functions can be shown to have gradients with Wirtinger derivatives [5].

$f(z)$	$\frac{\partial f}{\partial z}$	$\frac{\partial f}{\partial \overline{z}}$
$c^T z = z^T c$	c	0
$c^T \overline{z} = z^H c$	0	c
$z^H z = z^T \overline{z}$	$\overline{z}$	z
$z^H M z = z^T M^T \overline{z}$	$M^T \overline{z}$	Mz

We will use the results from this table for the derivations to follow.

4.2 Block Coordinate Descent for Complex Variables

The following derivations extend the BCD-GL algorithm of Qin *et al.* [19] to complex variables. We begin our derivation with the original group lasso unconstrained problem

$$\min_x \frac{1}{2} \|Ax - b\|^2 + \lambda \sum_{j=1}^{J} \|x_j\|, \tag{4.5}$$

where A is the data, x is the parameter, and b is the response from a least squares setting. The complex analog to Equation (4.5) is

$$\min_x \frac{1}{2}(Ax - b)^H (Ax - b) + \lambda \sum_{j=1}^{J} \sqrt{x_j^H x_j}. \tag{4.6}$$

Looking at the jth subiteration, we solve

$$\min_{x_j} \frac{1}{2}\Big(x_j^H M_j x_j + \big(\sum_{i\neq j} x_i^H A_i^H - b^H\big) A_j x_j + x_j^H A_j^H \big(\sum_{i\neq j} A_i x_i - b\big)\Big) + \lambda\sqrt{x_j^H x_j}, \tag{4.7}$$

where $M_j = A_j^H A_j$. Notice Equation (4.7) is again a real-valued objective, so it is amenable to Wirtinger differentiation. Put $p_j = \Big[\big(\sum_{i\neq j} x_i^H A_i^H - b^H\big)A_j\Big]^T$. Taking the Wirtinger derivative of Equation (4.7) with respect to x_j and setting to zero, we get that the first order optimality condition implies

$$\big(M_j + \frac{\lambda}{\|x_j\|} I\big)\overline{x_j} = -p_j. \tag{4.8}$$

Next, we convert Equation (4.6) into the trust-region subproblem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}\big(x_j^H M_j x_j + p_j^T x_j\big) \\ \text{subject to} \quad & \|x_j\| \leq \Delta, \end{aligned}$$

where the minimizer x_j^* satisfies $\|x_j^*\| = \Delta$ and has the form

$$x_j^* = -\overline{\big(M_j + \frac{\lambda}{\Delta} I\big)^{-1} p_j} \tag{4.9}$$

$$= \Delta y_j(\Delta). \tag{4.10}$$

Thus, $y_j(\Delta)$ has the form

$$y_j(\Delta) = -\overline{\big(\Delta M_j + \lambda I\big)^{-1} p_j}. \tag{4.11}$$

Now, the task is to search for Δ such that $\|y_j(\Delta)\| = 1$. M_j is Hermitian, so the Schur decomposition $M_j = Q\Gamma Q^H$ yields a diagonal Γ and unitary Q, leading to the derivation

$$\|y_j(\Delta)\|^2 = \left\| \overline{\left(\Delta M_j + \lambda I\right)^{-1} p_j} \right\|^2 \tag{4.12}$$
$$= \left\| \left(\Delta M_j + \lambda I\right)^{-1} p_j \right\|^2 \tag{4.13}$$
$$= \left\| \left(\Delta Q \Gamma Q^H + \lambda I\right)^{-1} p_j \right\|^2 \tag{4.14}$$
$$= \left\| Q \left(\Delta \Gamma + \lambda I\right)^{-1} Q^H p_j \right\|^2 \tag{4.15}$$
$$= \left\| \left(\Delta \Gamma + \lambda I\right)^{-1} Q^H p_j \right\|^2 \tag{4.16}$$
$$= \left\| \sum_i \frac{q_i^H p_j}{\Delta \gamma_i + \lambda} \right\|^2 \tag{4.17}$$
$$= \sum_i \overline{\left(\frac{q_i^H p_j}{\Delta \gamma_i + \lambda} \right)} \frac{q_i^H p_j}{\Delta \gamma_i + \lambda}. \tag{4.18}$$

We apply Newton's method to

$$\phi(\Delta) = 1 - \frac{1}{\|y_j(\Delta)\|}, \tag{4.19}$$

using the following derivations,

$$\frac{d}{d\Delta}\Big(\frac{1}{\|y_j(\Delta)\|} \Big) = \frac{d}{d\Delta} \big(\|y_j(\Delta)\|^2 \big)^{-\frac{1}{2}} \tag{4.20}$$
$$= -\frac{1}{2} \big(\|y_j(\Delta)\|^2 \big)^{-\frac{3}{2}} \frac{d}{d\Delta} \|y_j(\Delta)\|^2 \tag{4.21}$$

and, letting $\gamma_i = \alpha_i + i\beta_i$,

$$\frac{d}{d\Delta} \|y_j(\Delta)\|^2 = \frac{d}{d\Delta} \sum_i \frac{\overline{(q_i^H p_j)}(q_i^H p_j)}{(\Delta\alpha_i + \lambda)^2 + \Delta^2 \beta_i^2} \tag{4.22}$$
$$= -\sum_i \frac{\overline{(q_i^H p_j)}(q_i^H p_j)}{\big[(\Delta\alpha_i + \lambda)^2 + \Delta^2 \beta_i^2\big]^2} \big(2\alpha_i(\Delta\alpha_i + \lambda) + 2\Delta\beta_i^2 \big). \tag{4.23}$$

Note that this process finds $\overline{y_j(\Delta)}$, so we conjugate once more before substituting $x_j = \Delta y_j(\Delta)$. We now summarize the results into Algorithm 3, a general block-coordinate descent algorithm for complex variables.

Input: $A \in \mathbb{C}^{n \times p}$, $b \in \mathbb{C}^{n \times 1}$, $x \in \mathbb{C}^{p \times 1}$ randomly initialized, $\lambda \in \mathbb{R}$, J group labels

while *x not converged* **do**

  for $j = 1, \ldots, J$ **do**

    A_j = columns of A belonging to group j

    Compute Schur decomposition of $M_j = A_j^H A_j$

    $d = \sum_{i \neq j}(x_i^H A_i^H - b^H)$

    $p_j = (dA_j)^T$

    if $\|p_j\| \leq \lambda$ **then**

      $x_j = 0$

    else

      Compute solution to equation (4.11) and substitute $x_j = \Delta y_j(\Delta)$.

    end

  end

end

Output: $\widehat{x} \in \mathbb{C}^{p \times 1}$, the complex group lasso parameter estimate

Algorithm 3: Complex BCD-GL

4.3 Optimizing Complex BCD-GL for T-AR

For T-AR, the complex block-coordinate descent algorithm can be accelerated by exploiting the block-diagonal structure of the t-product's implementation in the Fourier domain. The jth sub-iteration given by Equation (4.7) involves a data matrix of form

$$A_j = \begin{bmatrix} a_{1,j} & & & \\ & a_{2,j} & & \\ & & \ddots & \\ & & & a_{t,j} \end{bmatrix} \tag{4.24}$$

where $a_{k,j}$ are column vectors in $\mathbb{C}^{n \times 1}$. Thus, $M_j = A_j^H A_j$ is a real diagonal matrix for all T-AR problems and we do not need to compute the Schur form of M_j (an $O(n^3)$ computation) at every iteration through J groups. Instead, letting m_j be the column vector of M_j's diagonal elements, we simplify the T-

AR calculation of $y_j(\Delta)$ by using

$$\left\|y_j^{\text{T-AR}}(\Delta)\right\|^2 = \left\|\overline{\left(\Delta M_j + \lambda I\right)^{-1} p_j}\right\|^2 \tag{4.25}$$

$$= \left\|\left(\Delta m_j^T I + \lambda I\right)^{-1} p_j\right\|^2 \tag{4.26}$$

$$= \sum_i \frac{\left\|[p_j]_i\right\|^2}{(\Delta[m_j]_i + \lambda)^2} \tag{4.27}$$

with

$$\frac{d}{d\Delta}\left\|y_j^{\text{T-AR}}(\Delta)\right\|^2 = -2\sum_i \frac{\left\|[p_j]_i\right\|^2 [m_j]_i}{(\Delta[m_j]_i + \lambda)^3} \tag{4.28}$$

and proceed with Newton's method to search for the appropriate Δ in Equation (4.27). Test results show a noticeable speedup from the naïve implementation by using these simplified derivations; comparing implementations in the R statistical language, we generally notice the fast method computes the same result in about 90% of the time of the general algorithm. However, the simplified derivation replaces a Fortran implementation of the Schur decomposition, the `gqz` function of the `geigen` package on CRAN, which is already much faster than the R code that wraps it.

CHAPTER 5

EXPERIMENTS AND APPLICATIONS

The simulation results of this chapter are intended to substantiate the effectiveness of T-AR forecasting in ideal settings as well as the correctness of complex group lasso on both general complex regression problems and T-AR problems. First, we show that T-AR demonstrates an obvious competitive advantage for time series simulated in the conditions for which T-AR was designed. Then, we show that CGL works for general complex-valued regression problems with no time series context. Next, we combine CGL with T-AR to demonstrate how CGL can improve the prediction performance of the basic T-AR implementation. Finally, we demonstrate a simple application of T-AR to stock price data as a case study of the model's use in practice. All results are computed using the R statistical language.

5.1 Simulation Study of T-AR

In our first simulation study, we illustrate the ideal circumstances in which T-AR has a significant advantage over its competitors. Our synthetic dataset consists of time series generated by a seasonality component that is compounded to form a geometric trend. We specify a period length of 7 and a total series length of 104 periods to emulate a weekly forecasting problem across the duration of two years. We generated different series by varying a rate parameter from -0.3 to 0.3 in steps of 0.01. When the rate is non-negative, we generate a series by compounding each previous week by $(1 + \text{rate})$ and then adding daily noise from $N(\mu = 0, \sigma = 0.1)$, achieving a nondecreasing trend. When the rate

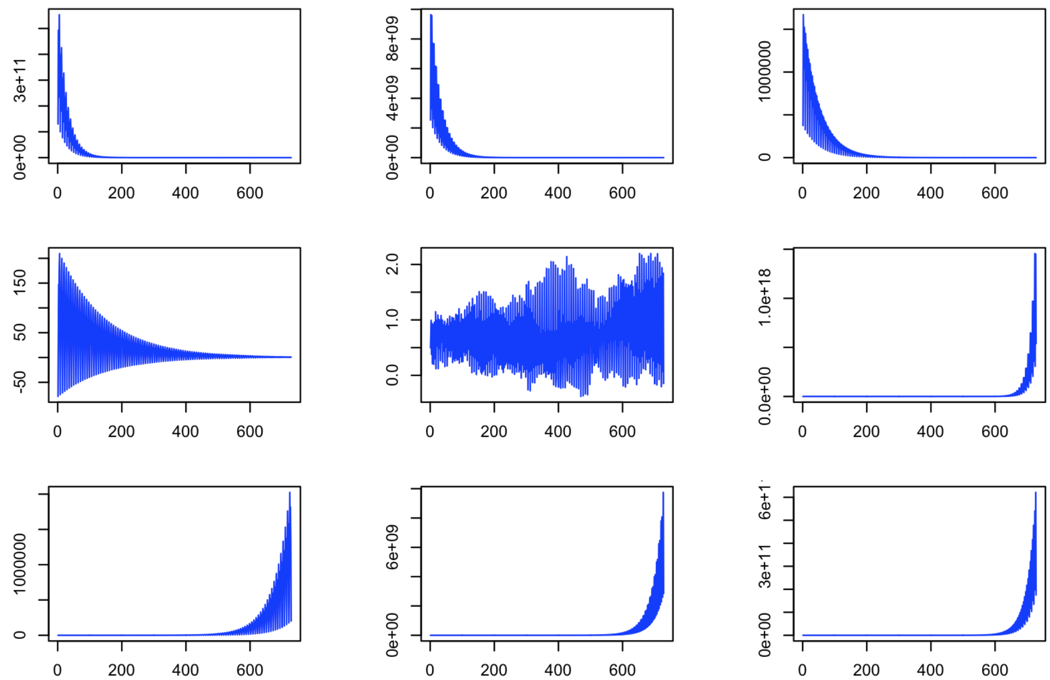

Figure 5.1: Nine example simulated series.

is negative, we reverse the series generated by compounding with $(1 - \text{rate})$ so that the result is a nonincreasing trend. The examples in Figure 5.1 show that this approach generates a representative set of noisy, quasi-seasonal series with geometric trends varying from fast decrease to fast increase.

For each series, we slide a window forward week by week until reaching one week prior to the end of the series. This window is the training data available for all competing models, and the task is to forecast the next week's values using the information in the current window. The window sizes we tested were 4 weeks, 13 weeks, 18 weeks, and 26 weeks. For each model and each window size, we report the average Mean Average Percentage Error (MAPE) across all

MAPE	$n = 4$	$n = 13$	$n = 18$	$n = 26$
T-AR($[1]$)	0.05	0.07	0.07	0.04
T-AR($[1, 2]$)	0.16	0.11	0.10	0.16
ARIMA	2.20	4.44	4.52	4.12
STL	0.55	4.48	5.23	20.49
HW	0.22	0.37	0.21	0.27

Table 5.1: Simulation average MAPE results.

forecasts, across all series. MAPE is defined as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\text{Actual}_i - \text{Forecast}_i}{\text{Actual}_i}\right|. \tag{5.1}$$

The simulation results in Table 5.1 show that T-AR models on average outperform the competition on synthetic time series that simultaneously exhibit seasonality and geometric trend patterns. ARIMA and STL struggle to adapt to the geometric curvature of our boundary cases. However, we notice that the performance of Holt-Winters comes close to that of T-AR; indeed, the original motivation of the Holt-Winters method was to adapt quickly to changes in the trend and seasonality patterns of sales [8, 22].

5.2 Evaluation of Complex Group Lasso

Next, we give simulation evidence that CGL is correct and discuss to what extent the behavior meets our expectations. The simulation involves generating a random complex data matrix complex parameter $x \in \mathbb{C}^{p\times 1}$, and complex noise $\epsilon \in \mathbb{C}^{n\times 1}$ all with real and imaginary components sampled from $N(0, 1)$. The response vector $b \in \mathbb{C}^{n\times 1}$ is generated by

$$b = Ax + \epsilon. \tag{5.2}$$

CMSE	$p = 4$	$p = 16$	$p = 64$
$n = 200$	0.084	1.122	27.829
$n = 500$	0.033	0.425	4.524
$n = 1000$	0.015	0.209	1.958
$n = 2000$	0.010	0.094	0.955

Table 5.2: Squared error of complex block coordinate descent.

For the following evaluations, we consider two metrics of correctness. The first metric is an extension of mean squared error; we define the complex mean squared error (CMSE) to be

$$CMSE = \frac{1}{2n} \sum_{i=1}^{n} \text{Re}(b_i - A(i,:)\hat{x})^2 + \text{Im}(b_i - A(i,:)\hat{x})^2. \tag{5.3}$$

However, CMSE will naturally degrade as the problem size increases. To control for growing dimensionality, we also consider proportional error,

$$\text{Error}_{\text{prop}} = \frac{\|b - A\hat{x}\|^2}{\|b\|^2}, \tag{5.4}$$

which we will see is more stable as the problem size grows.

Setting $\lambda = 0$ and $\epsilon = 0$ tests the algorithm's ability to recover exact solutions for ordinary, unpenalized least squares problems. We ran tests on problems of various dimensions, testing $n = 200, 500, 1000, 2000$ and $p = 4, 16, 64$. For each combination of n and p, we ran 100 tests and recorded the mean of the metrics for that combination over 100 simulations. Complex MSE and proportional error results are summarized in Tables 5.2 and 5.3 respectively. Observations of the average number of iterations until convergence are shown in Table 5.4. The tables show an intuitive outcome that CGL results degrade as the dimensionality p increases and improve as sample size n increases. This pattern is consistent for all three metrics.

We also test the group-wise sparsity-inducing facility of CGL in order to ensure that the sparsity is induced at the right locations. To set up this test, we

$\text{Error}_{\text{prop}}$	$p = 4$	$p = 16$	$p = 64$
$n = 200$	0.011	0.035	0.226
$n = 500$	0.004	0.013	0.036
$n = 1000$	0.002	0.006	0.015
$n = 2000$	0.001	0.003	0.007

Table 5.3: Proportional error of complex block coordinate descent.

Iterations	$p = 4$	$p = 16$	$p = 64$
$n = 200$	9.0	28.4	275.2
$n = 500$	8.5	22.1	78.6
$n = 1000$	7.6	21.0	57.6
$n = 2000$	7.1	18.4	49.0

Table 5.4: Average number of iterations until convergence.

fix $n = 1000$ and $p = 16$ with a parameter structured into four groups of four indexed [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]. We simulate three variations where each involves a ground-truth combination of relevant and extraneous groups of variables. The first ground-truth is that only group 1 is relevant, and the test is whether group 1 is the last to be eliminated by CGL process. The second ground-truth is that only group 1 is extraneous, and the test is whether group 1 is eliminated first. The third ground-truth involves two extraneous groups. To challenge the algorithm, we corrupt the response vector b with noise simulated from $N(0, 0.5)$ for all three testing variations.

We simulate 100 such scenarios allowing λ to vary from 0 to 100 in steps of 0.25. For both the first and second ground-truths, CGL passed 100 out of 100 times. However, when we simulated two extraneous feature groups, CGL passed 97 out of 100 times. Figure 5.2 shows trace plots of the extraneous CGL coefficients shrinking as the penalty λ increases. Trace plots are common diagnostics in conventional group lasso software and show that the CGL is indeed functioning as expected for general complex-valued regression problems.

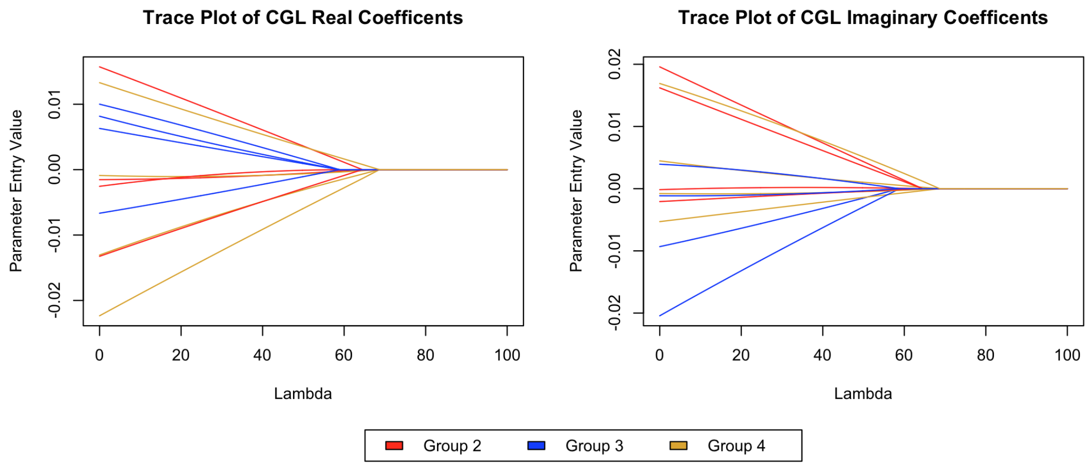

Figure 5.2: Simultaneous shrinkage within extraneous CGL groups.

5.3 Complex Group Lasso with T-AR

Whereas the simulations in Section 5.1 illustrate the advantages of T-AR over conventional time series benchmarks, in this section we demonstrate the advantages of T-AR's multi-lag facility along with the efficacy of its group lasso regularization. We simulate a random weekwise lag-l series with frequency t starting with l seed random periods, $X_n \in \mathbb{R}^t$ for weeks $n = 1 \ldots l$ and then generate a set of l coefficients that sum to 1, $\alpha_1 + \ldots + \alpha_l = 1$. All subsequent periods (weeks) are a convex combination of the l periods prior, so the weekwise (T-AR) autoregressive process is generated by

$$X_{n+1} = \alpha_1 X_n + \alpha_2 X_{n-1} + \ldots + \alpha_l X_{n-l} + \epsilon, \tag{5.5}$$

where ϵ is distributed $N(0, 0.1)$. In these tests, we let $t = 7$ and training window $n = 26$ to replicate the setting of forecasting weekly data using two quarters (26 weeks) of history. Figure 5.3 shows an example of a series generated by a weekwise autoregressive process when $l = 2$.

We consider the setup with two weeks of lag, setting $\alpha_1 = 0.1$ and $\alpha_2 = 0.9$

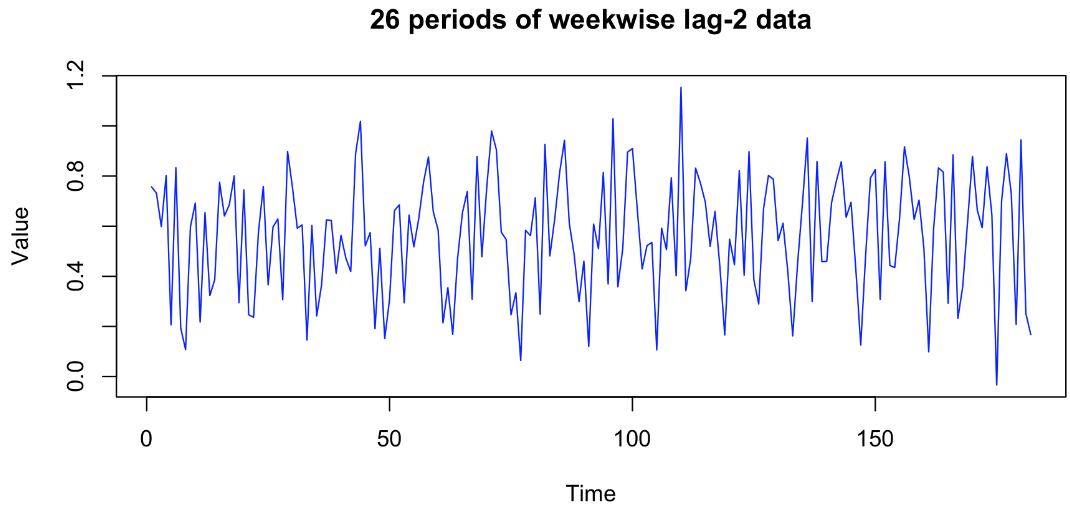

Figure 5.3: Synthetic series generated by a lag-2 T-AR process.

MAPE	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 10$
T-AR($[1]$)	0.5279	0.5299	0.5319	0.5360	0.6096
T-AR($[1, 2]$)	0.2362	0.2383	0.2405	0.2449	0.3570
T-AR($[1, 2, 3]$)	0.2399	0.2377	0.2384	0.2424	0.3596
T-AR($[1, 2, 3, 4]$)	0.2466	0.2374	0.2382	0.2426	0.3883

Table 5.5: Regularized T-AR performance in simulation.

in order to emphasize the influence of two weeks ago. This simulation is analogous to an AR(2) process except that the "2" is in units of weeks, each with 7 days. We simulate 104 periods of data for 728 total time points. Our metric of accuracy is, again, the MAPE metric. We run T-AR with the following lag parameterizations: [1], [1, 2], [1, 2, 3], and [1, 2, 3, 4], and we test CGL at various settings of λ to see how MAPE improves with regularization. Note that in this simulation, lags 1 and 2 are relevant and lags 3 and 4 are extraneous. We see that the regularization results agree with the ground-truth that unpenalized TAR($[1, 2]$) is the underlying model. In Table 5.5, the MAPE result for TAR($[1, 2]$) and $\lambda = 0$ is the lowest in the MAPE table. Notice also that regularization does not improve the MAPE score of a model incorporating only relevant lags. Instead, regularization improves prediction accuracy whenever the model incor-

Timing: Fast CGL in R	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 10$
T-AR($[1]$)	0.215s	0.239s	0.248s	0.247s	0.269s
T-AR($[1, 2]$)	22.275s	26.917s	26.215s	25.883s	352.347s
T-AR($[1, 2, 3]$)	68.862s	82.362s	89.335s	124.569s	794.376s
T-AR($[1, 2, 3, 4]$)	138.970s	168.960s	146.938s	231.521s	972.705s

Table 5.6: Profiling the fast CGL implementation for T-AR.

Fast/Naïve	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 10$
T-AR($[1]$)	0.919	0.912	0.936	0.915	0.947
T-AR($[1, 2]$)	0.823	0.873	0.875	0.891	1.061
T-AR($[1, 2, 3]$)	0.819	0.874	0.948	0.854	0.962
T-AR($[1, 2, 3, 4]$)	0.822	0.876	0.832	1.324	0.947

Table 5.7: Speedup from using simplified CGL derivations for T-AR.

porates extraneous lags. Notice that the second best MAPE score belongs to the overparameterized T-AR($[1, 2, 3, 4]$) model subjected to $\lambda = 0.25$ regularization.

We also measured the runtimes of T-AR using CGL lag selection. Table 5.6 shows that lag selection is a slow process. However, Table 5.7 shows that the fast CGL algorithm for T-AR gives a consistent, albeit moderate, speedup over the naïve algorithm.

5.4 Application to Stock Price Data

Finally, we explore an application of T-AR to historical stock price data obtained from Yahoo Finance. We use this opportunity to demonstrate T-AR's diagnostic parameter interpretation as well as comment on some observed characteristics of T-AR forecasts. We focus specifically on the closing prices of Apple (AAPL) stock from December 22nd, 1983 to February 19th, 1986. The data was chosen to be evaluated in this time interval due to its volatility. Every seven days, T-AR forecasts AAPL closing prices for the next seven days using a range of historical

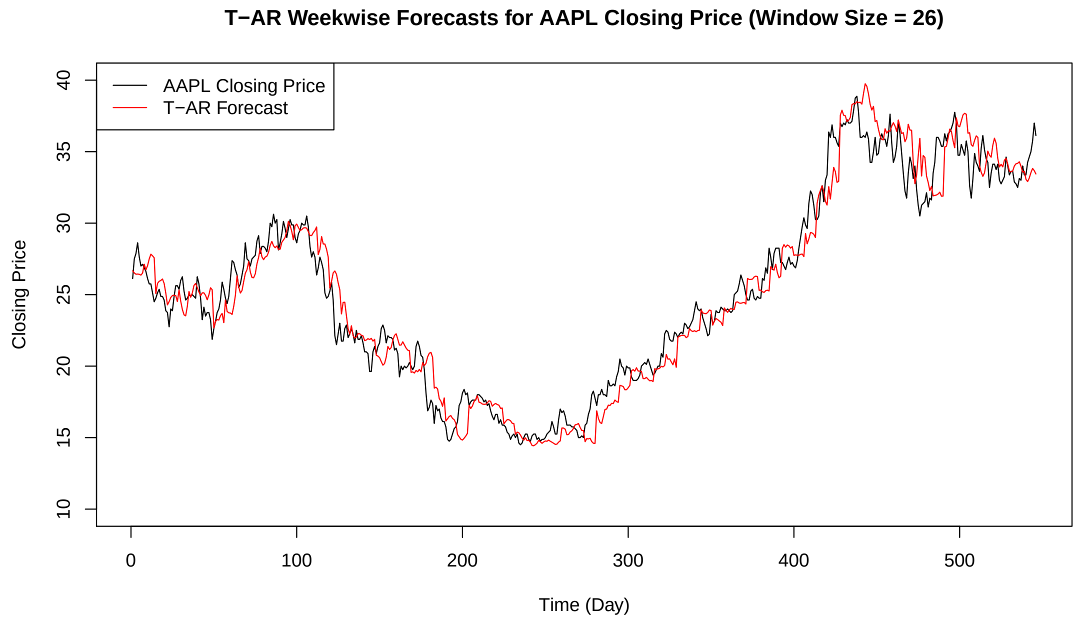

Figure 5.4: T-AR forecasts of AAPL closing prices.

data as training input. In Figure 5.4, we present forecasted closing prices from T-AR([1]) using training window size 26, performing with average daily MAPE equal to 0.0504. Figure 5.4 shows how the T-AR weekwise forecasts compare to actual closing prices.

Notice, however, that the 26-week window T-AR series exhibits some inertia, as if the forecasts would be better if one could slide the red line slightly to the left. This "inertia" can be manipulated by adjusting the size of the training window. In general, increasing window size increases forecasting inertia but also increases forecasting smoothness. The ideal window size depends on the specific application, and Figure 5.5 shows an example where, by reducing training window size to 4 weeks, the decrease in inertia is not worth the loss of smoothness (MAPE = 0.0594).

The diagnostic chart in Figure 5.6 give insights into the T-AR behavior on

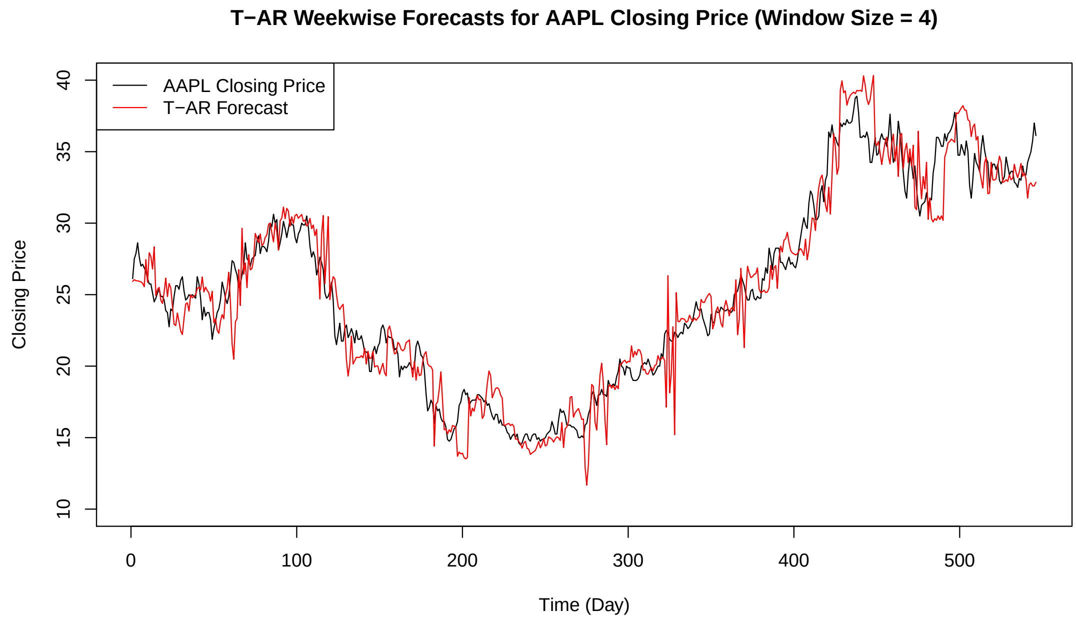

Figure 5.5: Smaller window size decreases inertia but also decreases smoothness.

the AAPL stock data. According to the parameter interpretation in Section 3.2 (see, in particular, Figure 3.4), the prominent skew in the distribution of the proportional magnitudes of the coefficient indicates that AAPL price data exhibits some degree of seasonality and geometric trend and, therefore, can be tracked by T-AR to an extent. However, the fact that there are no clearly dominant coefficients explains the limitation of using T-AR to forecast closing prices of this stock.

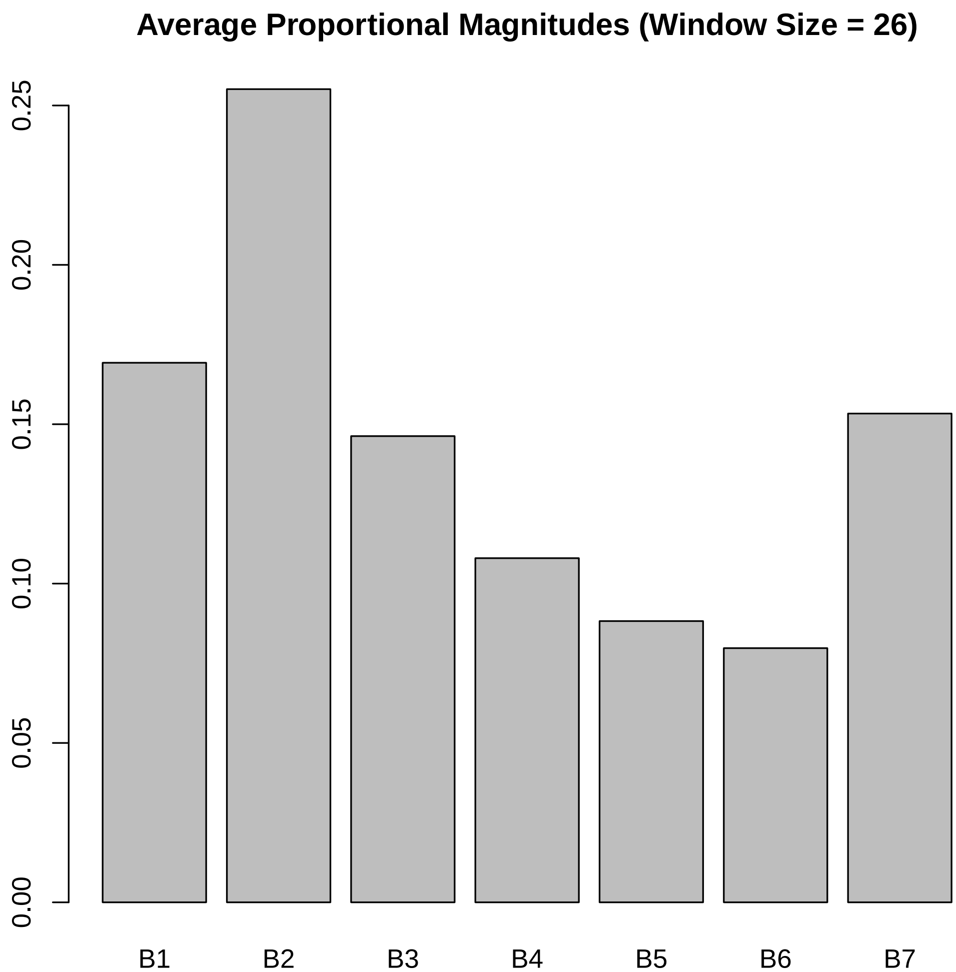

Figure 5.6: AAPL forecasting coefficients.

CHAPTER 6

CONCLUSIONS

In this thesis, we have presented the novel tensor autoregressive model that utilizes the t-product tensor operation to extend the ordinary linear model into the temporal dimension, transforming the ordinary linear model into a time series autoregressive model for periodic time series data. The regression-like framework of the T-AR connects regression methodology to a time series application, and we showed how to incorporate structured regularization for our time series problem by devising a complex-valued group lasso algorithm compatible with the implementation of T-AR's tensor operations.

We now conclude the thesis by addressing some possibilities of further work based on our explorations. First, we notice that the T-AR model assumes two specific and quantifiable time series traits, seasonality and geometric trend, and that the proportional magnitudes of the parameter coefficients indicate how prominent these traits are in the data. Therefore, one possible direction in which to extend this work is to form a rigorous null hypothesis statement for the T-AR time series condition. The T-AR parameters' proportional magnitudes suggest a test statistic that may be connected to the Dirichlet prior to the Multinomial distribution. A concrete application of a successful theory would be in anomaly detection, detection of a "T-AR condition", and could lead to an additional diagnostic facility that quantifies the reliability of T-AR on specific applications.

Furthermore, in connection with T-AR feature selection, an alternative approach to determining the ideal model specification is to interpret some partial autocorrelation function in tensor format. Indeed, the partial autocorrelation function (PACF) plays an important role in order determination of autoregressive models, and a tensor extension of PACF would narrow the gap between T-AR and conventional AR while addressing the practical issue of determining a T-AR model.

BIBLIOGRAPHY

[1] S. Basu and G. Michailidis. Regularized estimation in sparse high-dimensional time series models. *The Annals of Statistics*, 43:1535–1567, 2015.

[2] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural networks*, 5(6):961–970, 1992.

[3] N. Chan, C. Yau, and R. Zhang. Group lasso for structural break time series. *Journal of the American Statistical Association*, 109:590–599, 2014.

[4] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[5] R. F. H. Fisher. *Precoding and Signal Shaping for Digital Transmission*. Wiley and Sons, Inc., New York, NY, 2002.

[6] Clive William John Granger and Paul Newbold. *Forecasting Economic Time Series*. Academic Press, 2014.

[7] N. Hao, M. E. Kilmer, K. Braman, and R. C. Hoover. Facial recognition using tensor-tensor decompositions. *Society for Industrial and Applied Mathematics*, 6(1):437–463, 2013.

[8] Charles Holt. Forecasting trends and seasonals by exponentially weighted moving averages. *ONR Research Memorandum, Carnegie Institute of Technology*, 52, 1957.

[9] RJ Hyndman. *forecast: Forecasting Functions for Time Series and Linear Models*, 2014. R package version 2.4.0.

[10] M. Kilmer, K. Braman, N. Hao, and R. Hoover. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM Journal on Matrix Analysis and Applications*, 34(1):148–172, 2013.

[11] M. Kilmer and C. Martin. Facial recognition using tensor-tensor decomposition. *SIAM Linear Algebra and its Applications*, 435(3):641–658, 2011.

[12] M. E. Kilmer and C. D. Martin. Factorization strategies for third-order tensors. *Linear Algebra and its Applications*, 435:641–658, 2011.

[13] T. Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories, April 2006.

[14] T.G. Kolda and B.W. Bader. Tensor decomposition and applications. *Society for Industrial an Applied Mathematics*, 51(3):455–500, 2009.

[15] J. Li and A. W. Moore. Forecasting web page views: Methods and observations. *Journal of Machine Learning Research*, 9:2217–2250, October 2008.

[16] Haiping Lu, Konstantinos Plataniotis, and Anastasios Venetsanopoulos. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44(7):1540 – 1551, 2011.

[17] A. Maleki, L. Anitori, Z. Yang, and R. G. Baraniuk. Asymptotic analysis of complex lasso via complex approximate message passing (camp). *IEEE Transactions on Information Theory*, 59(7):4290–4308, 2013.

[18] C. Martin, R. Shafer, and B. LaRue. An order-p tensor factorization with applications in imaging. *SIAM Journal on Scientific Computing*, 35(1):A474–A490, 2013.

[19] Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2):143–169, 2013.

[20] M. Rogers, L. Li, and S. Russell. Multilinear dynamical systems for tensor time series. *Advances in Neural Information Processing Systems*, 26, 2013.

[21] Raja Velu and Gregory C Reinsel. *Multivariate Reduced-Rank Regression: Theory and Applications*, volume 136. Springer Science & Business Media, 2013.

[22] Peter Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6:324–342, 1960.

[23] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68(1):49–67, 2006.

[24] H. Zhou, L. Li, and H. Zhu. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, June 2013.