Matrix Auto-regressive Model with Vector Time-series Covariates

Hu Sun

January 29, 2023

Outline

Central Research Question

- Given a matrix time series $\{\mathbf{X}_t\}$, how to forecast the matrix in the future given a history of matrices? In other words, given the data $X_{t-p}, X_{t-p+1}, \ldots, X_t$, how to give a prediction for $X_{t+1}, X_{t+2}, \ldots$?

Central Research Question

- Given a matrix time series $\{\mathbf{X}_t\}$, how to forecast the matrix in the future given a history of matrices? In other words, given the data $X_{t-p}, X_{t-p+1}, \dots, X_t$, how to give a prediction for $X_{t+1}, X_{t+2}, \dots$?
- If there is an additional vector time-series $\{\mathbf{z}_t\}$ that are correlated with the matrix time series, how can one incorporate the vector time-series information to assist the forecast?

Central Research Question

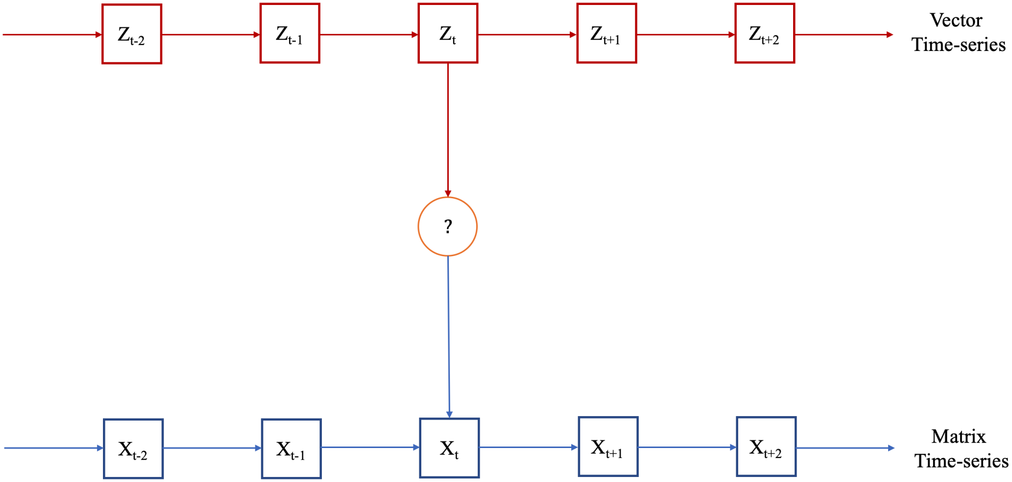

Figure: How can we build a statistics model to make vector time-series to forecast matrix time-series?

Vector Auto-regressive Model (VAR)

- We start building new models from some classical, well-studied model. Here, we focus on the Vector Auto-regressive (VAR) model.

Vector Auto-regressive Model (VAR)

- We start building new models from some classical, well-studied model. Here, we focus on the Vector Auto-regressive (VAR) model.
- A typical VAR(p) model for a d-dimensional vector time-series $\{x_t\}$ can be formulated as:

$$\mathbf{x}_t = \mathbf{\Phi}_1 \mathbf{x}_{t-1} + \mathbf{\Phi}_2 \mathbf{x}_{t-2} + \cdots + \mathbf{\Phi}_p \mathbf{x}_{t-p} + \mathbf{e}_t \tag{1}$$

  where $\mathbf{e}_t$ is often assumed to be a white-noise process, uncorrelated with the x_t.

Vector Auto-regressive Model (VAR)

- We start building new models from some classical, well-studied model. Here, we focus on the Vector Auto-regressive (VAR) model.
- A typical VAR(p) model for a d-dimensional vector time-series $\{x_t\}$ can be formulated as:

$$\mathbf{x}_t = \mathbf{\Phi}_1 \mathbf{x}_{t-1} + \mathbf{\Phi}_2 \mathbf{x}_{t-2} + \cdots + \mathbf{\Phi}_p \mathbf{x}_{t-p} + \mathbf{e}_t \tag{1}$$

  where $\mathbf{e}_t$ is often assumed to be a white-noise process, uncorrelated with the x_t.
- $\mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_p$ are $d \times d$ parameters to be estimated.

Vector Auto-regressive Model (VAR)

- We start building new models from some classical, well-studied model. Here, we focus on the Vector Auto-regressive (VAR) model.
- A typical VAR(p) model for a d-dimensional vector time-series $\{x_t\}$ can be formulated as:

$$\mathbf{x}_t = \mathbf{\Phi}_1 \mathbf{x}_{t-1} + \mathbf{\Phi}_2 \mathbf{x}_{t-2} + \cdots + \mathbf{\Phi}_p \mathbf{x}_{t-p} + \mathbf{e}_t \tag{1}$$

  where $\mathbf{e}_t$ is often assumed to be a white-noise process, uncorrelated with the x_t.
- $\mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_p$ are $d \times d$ parameters to be estimated.
- The degree of freedom of the model is $p \times d^2$

Matrix Auto-regressive Model (MAR), but with VAR

- Now consider we have a matrix time-series $\{\mathbf{X}_t\}$ of size $T \times m \times n$.

Matrix Auto-regressive Model (MAR), but with VAR

- Now consider we have a matrix time-series $\{\mathbf{X}_t\}$ of size $T \times m \times n$.
- If one "vectorize" matrices into long vectors, say for any matrix $\mathbf{X}_t$ of size $m \times n$, the vectorized matrix (column-major order) $\text{vec}\left(\mathbf{X_t}\right)$ is of shape $mn \times 1$. Then one can still apply the VAR model as follows:

$$\text{vec}\left(\mathbf{X_t}\right) = \mathbf{\Phi}_1 \text{vec}\left(\mathbf{X_{t-1}}\right) + \mathbf{\Phi}_2 \text{vec}\left(\mathbf{X_{t-2}}\right) + \cdots + \mathbf{\Phi}_p \text{vec}\left(\mathbf{X_{t-p}}\right) + \text{vec}\left(\mathbf{E_t}\right)$$

where $\mathbf{E}_t \in \mathbb{R}^{m \times n}$ is assumed as a white-noise matrix time-series with i.i.d entries.

Matrix Auto-regressive Model (MAR), but with VAR

- Now consider we have a matrix time-series $\{\mathbf{X}_t\}$ of size $T \times m \times n$.
- If one "vectorize" matrices into long vectors, say for any matrix $\mathbf{X}_t$ of size $m \times n$, the vectorized matrix (column-major order) $\text{vec}\left(\mathbf{X_t}\right)$ is of shape $mn \times 1$. Then one can still apply the VAR model as follows:

$$\text{vec}\left(\mathbf{X_t}\right) = \mathbf{\Phi}_1\text{vec}\left(\mathbf{X_{t-1}}\right) + \mathbf{\Phi}_2\text{vec}\left(\mathbf{X_{t-2}}\right) + \cdots + \mathbf{\Phi}_p\text{vec}\left(\mathbf{X_{t-p}}\right) + \text{vec}\left(\mathbf{E_t}\right)$$

  where $\mathbf{E}_t \in \mathbb{R}^{m\times n}$ is assumed as a white-noise matrix time-series with i.i.d entries.
- Each coefficient matrix $\mathbf{\Phi}_i, i = 1, 2, \ldots, p$ is of size $(mn) \times (mn)$, which can be astronomical for large matrices.

Challenges for MAR

There are two major challenges for estimating the matrix auto-regressive model using the vector auto-regressive model:

- Over-parameterization of the coefficient matrices $\mathbf{\Phi}_i$. (size $= mn \times mn$)
- Over-parameterization of the covariance matrix of $\text{vec}(\mathbf{E_t})$. (size $= mn \times mn$)

Matrix Auto-regressive Model (Chen et al., 2021)

In R. Chen, Xiao, and Yang (2021), a lag-1 MAR model is proposed to reduce the dimensionality of the parameter space:

$$\mathbf{X}_t = \mathbf{A}\mathbf{X}_{t-1}\mathbf{B}^{'} + \mathbf{E}_t$$

where $\mathbf{A}, \mathbf{B}$ are model coefficients. For any matrix $\mathbf{X}_t$ of size $m \times n$, the coefficients $\mathbf{A}$ is of size $m \times m$ and $\mathbf{B}$ is of size $n \times n$.

- Note how the total amount of parameters gets reduced from m^2n^2 to $m^2 + n^2$.

Matrix Auto-regressive Model (Chen et al., 2021)

An equivalent way of formulating the model under vectorization:

$$\operatorname{vec}(\mathbf{X_t}) = [\mathbf{B} \otimes \mathbf{A}]\operatorname{vec}(\mathbf{X_{t-1}}) + \operatorname{vec}(\mathbf{E_t})$$

where $\otimes$ is the Kronecker product of two matrices.

A Quick Recap of Kronecker Product

The Kronecker Product of two matrices $\mathbf{A}_{m\times n}, \mathbf{B}_{p\times q}$, i.e. $\mathbf{A}\otimes\mathbf{B}$, is defined as:

$$\mathbf{A}\otimes\mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2n}\mathbf{B} \\ \dots & \dots & \dots & \dots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix}_{mp\times nq}$$

Matrix Auto-regressive Model (Chen et al., 2021)

Additionally, the covariance structure of the error process is assumed to have a similar Kronecker product form:

$$\mathbf{cov}\left(\mathrm{vec}\left(\mathbf{E_t}\right)\right) = \Sigma_c \otimes \Sigma_r$$

Matrix Auto-regressive Model (Chen et al., 2021)

Think about the interpretation of the model by think of: how does $\mathbf{X}_{t-1}$ help predict $\mathbf{X}_{t,ij}$ (the (i,j)-th element of $\mathbf{X}_t$)?

- (VAR Model) $\mathbf{X}_{t,ij} = \sum_{k,l} \mathbf{\Phi}_{ij,kl} \mathbf{X}_{t-1,kl}$, where $\mathbf{\Phi}_{ij,kl}$ are different for all (k,l) tuple.
- (MAR Model) $\mathbf{X}_{t,ij} = \sum_{k,l} \left(\mathbf{A}_{ik} \mathbf{B}_{jl}\right) \mathbf{X}_{t-1,kl}$

Matrix Auto-regressive Model (Chen et al., 2021)

Think about the interpretation of the model by think of: how does $\mathbf{X}_{t-1}$ help predict $\mathbf{X}_{t,ij}$ (the (i,j)-th element of $\mathbf{X}_t$)?

- (VAR Model) $\mathbf{X}_{t,ij} = \sum_{k,l} \mathbf{\Phi}_{ij,kl}\mathbf{X}_{t-1,kl}$, where $\mathbf{\Phi}_{ij,kl}$ are different for all (k,l) tuple.
- (MAR Model) $\mathbf{X}_{t,ij} = \sum_{k,l} \left(\mathbf{A}_{ik}\mathbf{B}_{jl}\right)\mathbf{X}_{t-1,kl}$
- In MAR, the (i,k)-th element $\mathbf{A}$ captures how the k-th row of $\mathbf{X}_{t-1}$ predicts the i-th row of $\mathbf{X}_t$.

Matrix Auto-regressive Model (Chen et al., 2021)

Think about the interpretation of the model by think of: how does $\mathbf{X}_{t-1}$ help predict $\mathbf{X}_{t,ij}$ (the (i,j)-th element of $\mathbf{X}_t$)?

- (VAR Model) $\mathbf{X}_{t,ij} = \sum_{k,l} \mathbf{\Phi}_{ij,kl} \mathbf{X}_{t-1,kl}$, where $\mathbf{\Phi}_{ij,kl}$ are different for all (k,l) tuple.
- (MAR Model) $\mathbf{X}_{t,ij} = \sum_{k,l} \left(\mathbf{A}_{ik} \mathbf{B}_{jl}\right) \mathbf{X}_{t-1,kl}$
- In MAR, the (i,k)-th element $\mathbf{A}$ captures how the k-th row of $\mathbf{X}_{t-1}$ predicts the i-th row of $\mathbf{X}_t$.
- Similarly, the (j,l)-th element $\mathbf{B}$ captures how the j-th column of $\mathbf{X}_{t-1}$ predicts the l-th column of $\mathbf{X}_t$.

Matrix Auto-regressive Model (Chen et al., 2021)

Think about the interpretation of the model by think of: how does $\mathbf{X}_{t-1}$ help predict $\mathbf{X}_{t,ij}$ (the (i,j)-th element of $\mathbf{X}_t$)?

- (VAR Model) $\mathbf{X}_{t,ij} = \sum_{k,l} \mathbf{\Phi}_{ij,kl} \mathbf{X}_{t-1,kl}$, where $\mathbf{\Phi}_{ij,kl}$ are different for all (k, l) tuple.
- (MAR Model) $\mathbf{X}_{t,ij} = \sum_{k,l} (\mathbf{A}_{ik} \mathbf{B}_{jl}) \mathbf{X}_{t-1,kl}$
- In MAR, the (i,k)-th element $\mathbf{A}$ captures how the k-th row of $\mathbf{X}_{t-1}$ predicts the i-th row of $\mathbf{X}_t$.
- Similarly, the (j,l)-th element $\mathbf{B}$ captures how the j-th column of $\mathbf{X}_{t-1}$ predicts the l-th column of $\mathbf{X}_t$.
- Finally, the prediction effect of $\mathbf{X}_{t-1,kl}$ is decomposed into the product of the row effect ($\mathbf{A}$) and column effect ($\mathbf{B}$).

More Relevant Works

- In Hsu, Huang, and Tsay (2021), the authors consider further decomposing the covariance structure of $\text{vec}\left(\mathbf{E_t}\right)$ with a fixed-rank kriging model:

$$\mathbf{cov}(\text{vec}\left(\mathbf{E_t}\right)) = \mathbf{FMF}^{'} + \sigma_{\eta}^{2}\mathbf{I}$$

where $\mathbf{F}$ is a rank-k basis, and $\mathbf{M}$ is a $k \times k$ "core" covariance matrix.

More Relevant Works

- In Hsu, Huang, and Tsay (2021), the authors consider further decomposing the covariance structure of $\text{vec}\left(\mathbf{E_t}\right)$ with a fixed-rank kriging model:

$$\mathbf{cov}(\text{vec}\left(\mathbf{E_t}\right)) = \mathbf{FMF}' + \sigma_\eta^2\mathbf{I}$$

  where $\mathbf{F}$ is a rank-k basis, and $\mathbf{M}$ is a $k \times k$ "core" covariance matrix.
- In Wang, Liu, and R. Chen (2019), a matrix auto-regressive model for large-scale matrices is proposed with the model applied to a "core" factor matrix time-series.

More Relevant Works

- In Hsu, Huang, and Tsay (2021), the authors consider further decomposing the covariance structure of vec $(\mathbf{E_t})$ with a fixed-rank kriging model:

$$\mathbf{cov}(\text{vec}(\mathbf{E_t})) = \mathbf{FMF}' + \sigma_\eta^2 \mathbf{I}$$

where $\mathbf{F}$ is a rank-k basis, and $\mathbf{M}$ is a $k \times k$ "core" covariance matrix.
- In Wang, Liu, and R. Chen (2019), a matrix auto-regressive model for large-scale matrices is proposed with the model applied to a "core" factor matrix time-series.
- In X. Chen and Sun (2021), the authors consider forecasting a tensor time-series with vector time-series, but vector time-series are latent variables.

Baseline Model

Our model undertakes two tasks:

- Build an auto-regressive model for $\{\mathbf{X}_t\}$, without incurring latent variable.
- Incorporate the vector time-series $\{\mathbf{z}_t\}$ explicitly in the model.

Baseline Model

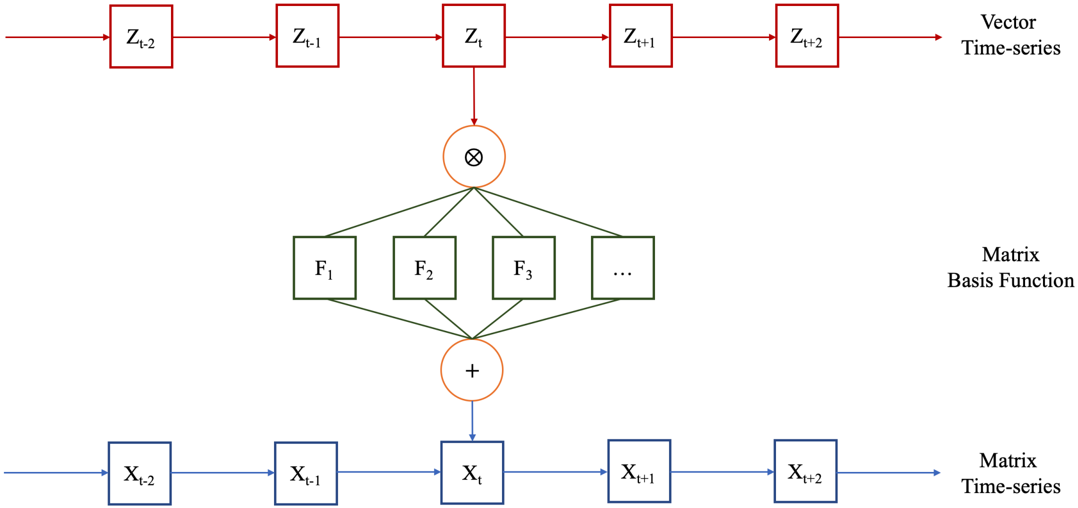

Figure: Matrix Auto-regressive Model with Temporal Covariates: Graphical Illustration.

Baseline Model

Our model can be formulated as:

$$\mathbf{X}_t = \sum_{l=1}^{p} A_l \mathbf{X}_{t-l} B_l{}' + \sum_{k=1}^{K} \left(\mathbf{z}_{t-1}' \beta_k \right) \cdot \mathbf{F}_k + \mathbf{E}_t \tag{2}$$

where:

- $(A_l, B_l)_{l=1}^p$ are pairs of $m \times m, n \times n$ auto-regressive coefficients.
- $\mathbf{F}_k, k = 1, 2, \ldots, K$ are $m \times n$ basis functions
- β_k are auxiliary data regression coefficients.
- $\mathbf{cov}\left(\mathrm{vec}\left(\mathbf{E_t}\right)\right) = \Sigma_c \otimes \Sigma_r$, the common Kronecker product covariance structure for the error process.

Baseline Model

One can still obtain a familiar vectorization form of the model as:

$$\mathbf{x}_t = \sum_{l=1}^{p} (B_l \otimes A_l)\, \mathbf{x}_{t-l} + [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K][\beta_1, \beta_2, \ldots, \beta_K]' \mathbf{z}_{t-1} + \mathbf{e}_t \tag{2}$$

where:

- $\mathbf{x}_t$: the vectorized matrix time-series $(mn \times 1)$
- $\mathbf{f}_k$: the vectorized matrix basis function $(mn \times 1)$
- $\mathbf{e}_t$: the vectorized noise term $(mn \times 1)$

Matrix Basis Function

The matrix basis functions $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_K$ are $m \times n$ matrices, i.e. the same size as the matrix time-series. **In this work, we select the basis from some parametric families instead of estimating the basis using non-parametric approach.** Potential choices of the basis functions include:

- Wavelet basis
- Multi-resolution spline basis (Jing et al. 2018)
- (**Our choice**) Spherical Harmonics basis (Nortje et al. 2015)

Matrix Basis Function

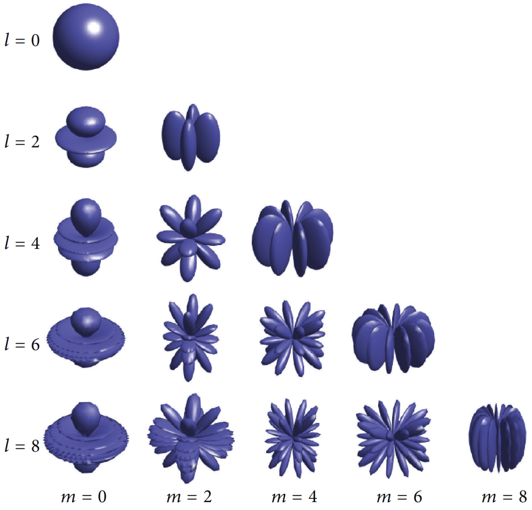

Figure: Spherical Harmonics Basis, source: (Nortje et al. 2015)

Matrix Basis Function

To evaluate the value of the basis function at every (i, j)-th cell, one first needs to define a *spatial grid* over the matrix time-series. This grid contains the **location information** of each cell of the matrix time-series, examples of such grid include:

- Longitude-Latitude Coordinates
- Width-Height in Digital Images

Matrix Basis Function

To evaluate the value of the basis function at every (i, j)-th cell, one first needs to define a *spatial grid* over the matrix time-series. This grid contains the **location information** of each cell of the matrix time-series, examples of such grid include:

- Longitude-Latitude Coordinates
- Width-Height in Digital Images

The basis function thus contains an extra layer of information, i.e. the location information of all data points, to help the vector covariates predict the future matrix time-series.

Model Estimation with Maximum Likelihood Estimator (MLE)

In our model:

$$\mathbf{X}_t = \sum_{l=1}^{p} A_l \mathbf{X}_{t-l} B_l{}' + \sum_{k=1}^{K} \left(\mathbf{z}_{t-1}' \beta_k \right) \cdot \mathbf{F}_k + \mathbf{E}_t$$
$$\mathrm{vec}\left(\mathbf{E_t}\right) \sim \mathcal{N}\left(\mathbf{0}, \Sigma_c \otimes \Sigma_r\right)$$

we need to estimate all parameters in red.

Model Estimation with Maximum Likelihood Estimator (MLE)

A natural choice is to estimate with the Maximum Likelihood Estimator (MLE):

$$\max_{\substack{A_1,A_2,\ldots,A_p,B_1,B_2,\ldots,B_p;\\ \beta_1,\beta_2,\ldots,\beta_K;\\ \Sigma_r,\Sigma_c}} \left\{ -\frac{T-p}{2}\left(\log|\Sigma_c|^m|\Sigma_r|^n\right) - \frac{1}{2}\sum_{t=p+1}^{T} \mathbf{r}_t' \left(\Sigma_c \otimes \Sigma_r\right)^{-1} \mathbf{r}_t \right\} \tag{3}$$

where $\mathbf{r}_t$ is simply the residual:

$$\mathbf{r}_t = \mathbf{x}_t - \sum_{l=1}^{p} \left(B_l \otimes A_l\right) \mathbf{x}_{t-l} - [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K][\beta_1, \beta_2, \ldots, \beta_K]' \mathbf{z}_{t-1}$$

Denote the negative log-likelihood function above as $\mathcal{L}(A_{1:p}, B_{1:p}, \beta_{1:K}, \Sigma_r, \Sigma_c)$.

Model Estimation with Maximum Likelihood Estimator (MLE)

- $\mathcal{L}(A_{1:p}, B_{1:p}, \beta_{1:K}, \Sigma_r, \Sigma_c)$ is convex for $\beta_{1:K}, \Sigma_r, \Sigma_c$, but is only *bi-convex* for pairs of $(A_l, B_l), l = 1, 2, \ldots, p$.
- Bi-convexity means that $\mathcal{L}(A_l, B_l, \ldots)$ is convex for A_l, conditioning on B_l being fixed, and vice versa.

Model Estimation with Maximum Likelihood Estimator (MLE)

In addition to the bi-convexity of the log-likelihood function, we also have an **identifiability concern** regarding pairs of $(A_l, B_l), l = 1, 2, \ldots, p$ and (Σ_c, Σ_r):

- For every pair of A_l, B_l, we can identify them only up to a scaling constant because:

$$B_l \otimes A_l = \left(\frac{1}{c} B_l\right) \otimes (cA_l), c \neq 0$$

  same issue for (Σ_c, Σ_r)

- To tackle this, we fix these pairs of parameters subject to the constraint that:

$$\|A_l\|_F = 1, \quad \text{sign}\,(tr(A_l)) = 1, \forall l$$
$$\|\Sigma_r\|_F = 1, \quad \text{sign}\,(tr(\Sigma_r)) = 1$$

Solve MLE with Block Coordinate Descent (BCD)

- A common choice of optimizing bi-convex functions is the block coordinate descent (BCD) method, or known as the cyclic coordinate minimization (CCM), or the alternating minimization (AM).

Solve MLE with Block Coordinate Descent (BCD)

- A common choice of optimizing bi-convex functions is the block coordinate descent (BCD) method, or known as the cyclic coordinate minimization (CCM), or the alternating minimization (AM).
- Basically at iteration $k+1$, when estimating A_l:

$$A_l^{(k+1)} = \arg\max \mathcal{L}(A_1^{(k+1)}, B_1^{(k+1)}, \ldots, A_{l-1}^{(k+1)}, B_{l-1}^{(k+1)}, A_l, B_l^{(k)}, \ldots,)$$

Solve MLE with Block Coordinate Descent (BCD)

- A common choice of optimizing bi-convex functions is the block coordinate descent (BCD) method, or known as the cyclic coordinate minimization (CCM), or the alternating minimization (AM).
- Basically at iteration $k+1$, when estimating A_l:

$$A_l^{(k+1)} = \arg\max \mathcal{L}(A_1^{(k+1)}, B_1^{(k+1)}, \ldots, A_{l-1}^{(k+1)}, B_{l-1}^{(k+1)}, A_l, B_l^{(k)}, \ldots,)$$

- Similarly when estimating B_l:

$$B_l^{(k+1)} = \arg\max \mathcal{L}(A_1^{(k+1)}, B_1^{(k+1)}, \ldots, A_{l-1}^{(k+1)}, B_{l-1}^{(k+1)}, A_l^{(k+1)}, B_l, \ldots,)$$

Solve MLE with Block Coordinate Descent (BCD)

- A common choice of optimizing bi-convex functions is the block coordinate descent (BCD) method, or known as the cyclic coordinate minimization (CCM), or the alternating minimization (AM).
- Basically at iteration $k+1$, when estimating A_l:

$$A_l^{(k+1)} = \arg\max \mathcal{L}(A_1^{(k+1)}, B_1^{(k+1)}, \ldots, A_{l-1}^{(k+1)}, B_{l-1}^{(k+1)}, {\color{red}A_l}, B_l^{(k)}, \ldots,)$$

- Similarly when estimating B_l:

$$B_l^{(k+1)} = \arg\max \mathcal{L}(A_1^{(k+1)}, B_1^{(k+1)}, \ldots, A_{l-1}^{(k+1)}, B_{l-1}^{(k+1)}, {\color{blue}A_l^{(k+1)}}, {\color{red}B_l}, \ldots,)$$

- We update all the parameters to be estimated cyclically in the order of:

$$A_1 \to B_1 \to A_2 \to B_2 \to \cdots \to A_p \to B_p \to (\beta_1, \beta_2, \ldots, \beta_K) \to \Sigma_c \to \Sigma_r \to \ldots$$

MLE with BCD: Algorithm Overview

Luckily, every step of our block coordinate descent algorithm has a closed-form solution, allowing exact maximization at every step. For instance:

- To update A_l:

$$A_l^{(k+1)} \leftarrow \left(\sum_{t=p+1}^{T} \widetilde{\mathbf{X}}_{t,-l} \Sigma_c{}^{-1} B_l \mathbf{X}'_{t-l} \right) \left(\sum_{t=p+1}^{T} \mathbf{X}_{t-l} B_l{}' \Sigma_c{}^{-1} B_l \mathbf{X}'_{t-l} \right)^{-1} \tag{3}$$

  one needs to replace the parameters in red with their current value at step k in the algorithm.

- The $\widetilde{\mathbf{X}}_{t,-l}$ is the residual of $\mathbf{X}_t$, excluding the lag-l prediction:

$$\widetilde{\mathbf{X}}_{t,-l} = \mathbf{X}_t - \sum_{s<l} A_s^{(k+1)} \mathbf{X}_{t-s} \left(B_s^{(k+1)} \right)' - \sum_{s>l} A_s^{(k)} \mathbf{X}_{t-s} \left(B_s^{(k)} \right)' - \sum_{\tau=1}^{K} \left(\mathbf{z}'_{t-1} \beta_\tau^{(k)} \right) \cdot \mathbf{F}_k$$

MLE with BCD: Algorithm Overview

- To update $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_K)$ jointly, we have:

$$\operatorname{vec}\left(\boldsymbol{\beta}'\right) \leftarrow \left[\sum_{t=p+1}^{T} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)' \left(\Sigma_c^{(k)} \otimes \Sigma_r^{(k)}\right)^{-1} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)\right]^{-1} \cdot \left[\sum_{t=p+1}^{T} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)' \left(\Sigma_c^{(k)} \otimes \Sigma_r^{(k)}\right)^{-1} \widetilde{\mathbf{x}}_t\right]$$

which is similar to the formula used for generalized least square (GLS).

- $\widetilde{\mathbf{x}}_t$ is the residual at t, excluding the vector prediction:

$$\widetilde{\mathbf{x}}_t = \operatorname{vec}\left(\mathbf{X}_t - \sum_{l=1}^{p} A_l^{(k+1)} \mathbf{X}_{t-l} \left(B_l^{(k+1)}\right)'\right)$$

MLE with BCD: Algorithm Overview

To update Σ_c, Σ_r, one has:

$$\Sigma_c \leftarrow \frac{\sum_{t=p+1}^{T} \mathbf{R}_t' \left(\Sigma_r^{(k)}\right)^{-1} \mathbf{R}_t}{m(T-p)}$$

$$\Sigma_r \leftarrow \frac{\sum_{t=p+1}^{T} \mathbf{R}_t \left(\Sigma_c^{(k+1)}\right)^{-1} \mathbf{R}_t'}{n(T-p)}$$

where $\mathbf{R}_t$ is the residual at t, i.e. $\mathbf{X}_t$ subtracting all predictions, using all the **updated** value of $A_{1:p}, B_{1:p}, \beta_{1:K}$

MLE with BCD: Algorithm Overview

Algorithm 1 Iterative Least-square for Matrix Auto-regressive Model with Vector Covariates

Input: $m \times n$ matrix data $\{\mathbf{X}_1\}_{t=1}^T$, $d \times 1$ associated vector covariates $\{\mathbf{z}_t\}_{t=1}^T$.

Initialization: Randomly initialize $A_{1:p}, B_{1:p}, \boldsymbol{\beta}$ from standard normal distribution. Initialize both Σ_r and Σ_c as identity matrices. Set convergence threshold at $\tau = 10^{-5}$ and $\Delta = 1$.

while $\Delta > \tau$ **do**

  for $l = 1 : p$ **do**

    Update A_l based on Eq. 7

    Update B_l based on Eq. 8

  end for

  Update $\boldsymbol{\beta}$ based on Eq. 9, with the calculation simplified by Eq. 13 and Eq. 14.

  Update Σ_c based on Eq. 10

  Update Σ_r based on Eq. 11

  Re-scale each pair of $(\widehat{A_l}, \widehat{B_l})$ such that $\|\widehat{A_l}\|_F = 1$ and $\text{sign}(tr(\widehat{A_l})) = 1$

  Re-scale $\widehat{\Sigma}_c, \widehat{\Sigma}_r$ such that $\|\widehat{\Sigma}_r\|_F = 1$.

  Calculate the convergence of $B_l \otimes A_l$ with the upper bound in Eq. 12, denoted as Δ_1.

  $\Delta_2 \leftarrow \|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}\|_F$; $\Delta_3 \leftarrow \|\Sigma_c^{(k+1)} - \Sigma_c^{(k)}\|_F + \|\Sigma_r^{(k+1)} - \Sigma_r^{(k)}\|_F$

  $\Delta \leftarrow \max(\Delta_1, \Delta_2, \Delta_3)$

end while

Output: $\widehat{A}_{1:p}, \widehat{B}_{1:p}, \widehat{\boldsymbol{\beta}}, \widehat{\Sigma}_c, \widehat{\Sigma}_r$

Figure: Algorithm Overview

The Feature Selection Problem

Given the computational algorithm, the performance of our model also relies on the selection of the following hyperparameters:

- p: the maximum lag of the auto-regressive term, i.e. the maximum number of $\mathbf{X}_l$ used as the predictor (AIC, BIC)

The Feature Selection Problem

Given the computational algorithm, the performance of our model also relies on the selection of the following hyperparameters:

- p: the maximum lag of the auto-regressive term, i.e. the maximum number of $\mathbf{X}_l$ used as the predictor (AIC, BIC)
- q: the maximum lag of the vector predictor $\mathbf{z}_{t-1}$. We use $\mathbf{z}_{t-1}$ in our previous discussion, but actually one can use multi-lag predictors: $\mathbf{z}_{t-1}, \mathbf{z}_{t-2}, \ldots, \mathbf{z}_{t-q}$

The Feature Selection Problem

Given the computational algorithm, the performance of our model also relies on the selection of the following hyperparameters:

- p: the maximum lag of the auto-regressive term, i.e. the maximum number of $\mathbf{X}_l$ used as the predictor (AIC, BIC)
- q: the maximum lag of the vector predictor $\mathbf{z}_{t-1}$. We use $\mathbf{z}_{t-1}$ in our previous discussion, but actually one can use multi-lag predictors: $\mathbf{z}_{t-1}, \mathbf{z}_{t-2}, \ldots, \mathbf{z}_{t-q}$
- K: the amount of basis functions to use

The Feature Selection Problem

Given the computational algorithm, the performance of our model also relies on the selection of the following hyperparameters:

- p: the maximum lag of the auto-regressive term, i.e. the maximum number of $\mathbf{X}_l$ used as the predictor (AIC, BIC)
- q: the maximum lag of the vector predictor $\mathbf{z}_{t-1}$. We use $\mathbf{z}_{t-1}$ in our previous discussion, but actually one can use multi-lag predictors: $\mathbf{z}_{t-1}, \mathbf{z}_{t-2}, \dots, \mathbf{z}_{t-q}$
- K: the amount of basis functions to use

The selection of p has been discussed in many relevant works, and here we discuss how to select q and K using an ad-hoc procedure called Sparse Group Lasso (Simon et al. 2013).

The Feature/Basis Selection Problem

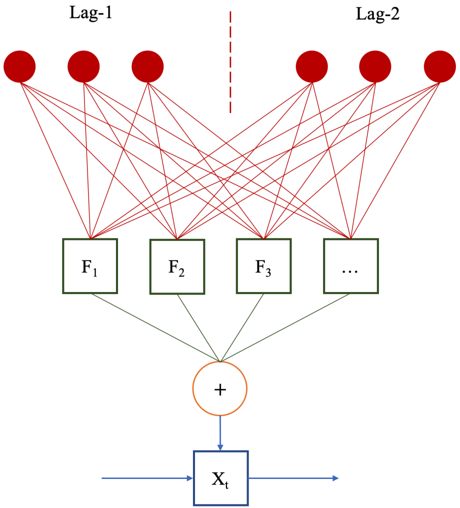

Figure: Our baseline model has no sparsity.

The Feature/Basis Selection Problem

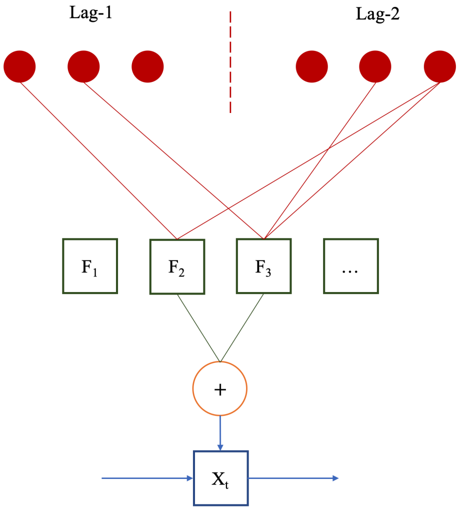

Figure: We need to ensure both basis sparsity and feature sparsity.

Selection with Sparse Group Lasso

- Our original updating rule for $\beta_1, \beta_2, \ldots, \beta_K$ is:

$$\operatorname{vec}\left(\boldsymbol{\beta}'\right) \leftarrow \left[\sum_{t=p+1}^{T} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)' \left(\Sigma_c^{(k)} \otimes \Sigma_r^{(k)}\right)^{-1} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)\right]^{-1} \cdot \left[\sum_{t=p+1}^{T} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)' \left(\Sigma_c^{(k)} \otimes \Sigma_r^{(k)}\right)^{-1} \widetilde{\mathbf{x}}_t\right]$$

Selection with Sparse Group Lasso

- Our original updating rule for $\beta_1, \beta_2, \ldots, \beta_K$ is:

$$\mathrm{vec}\left(\boldsymbol{\beta}'\right) \leftarrow \left[\sum_{t=p+1}^{T} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)' \left(\Sigma_c^{(k)} \otimes \Sigma_r^{(k)}\right)^{-1} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)\right]^{-1} \cdot \left[\sum_{t=p+1}^{T} \left(\mathbf{z}_{t-1}' \otimes \mathbf{F}\right)' \left(\Sigma_c^{(k)} \otimes \Sigma_r^{(k)}\right)^{-1} \widetilde{\mathbf{x}}_t\right]$$

- Now:

$$\min_{\beta_1,\beta_2,\ldots,\beta_K} \frac{1}{2(T-p)} \sum_{t=p+1}^{T} \widetilde{\mathbf{x}}_t' \left(\widehat{\Sigma}_c \otimes \widehat{\Sigma}_r\right)^{-1} \widetilde{\mathbf{x}}_t + (1-\alpha)\lambda \sum_{k=1}^{K} \|\beta_k\|_2 + \alpha\lambda \sum_{k=1}^{K} \|\beta_k\|_1$$

Selection with Sparse Group Lasso

We can fit the sparse group lasso with accelerated gradient descent. We can end up with a series of estimates of $\beta_1, \beta_2, \ldots, \beta_K$, such that:

- Some $\beta_k = \mathbf{0}$, which means the basis has null effect.
- Some β_k contains 0 coefficient, meaning the basis is not null, but some features at some lag have null effect.

Selection with Sparse Group Lasso

Algorithm 2 Matrix Auto-regressive Model with Vector Covariates + Sparse Group Lasso

Input: $m \times n$ matrix data $\{\mathbf{X}_1\}_{t=1}^T$, $d \times 1$ associated vector covariates $\{\mathbf{z}_t\}_{t=1}^T$. Group sparsity tuning parameter α.

1: Fit the non-sparse, fully-connected model using Algorithm 1, and get the initial estimates: $(\widehat{A}_l, \widehat{B}_l)_{l=1}^p$, $\text{vec}(\hat{\boldsymbol{\beta}}_0^{'})$, $\widehat{\Sigma}_r, \widehat{\Sigma}_c$. Set the initial value of $\boldsymbol{\beta}$ at $\hat{\boldsymbol{\beta}}_0$.

2: Calculate the partial residual time-series $\mathbf{R}_t = \mathbf{X}_t - \sum_{l=1}^p (\widehat{B}_l \otimes \widehat{A}_l)\mathbf{X}_{t-l}$

3: Set a grid of λ: $0 = \lambda_0 < \lambda_1 < \lambda_2 < \cdots < \lambda_J$

4: **for** j in $1 : J$ **do**

5: Initialize $\boldsymbol{\beta}$ at $\hat{\boldsymbol{\beta}}_{j-1}$.

6: Fit Sparse Group Lasso with (α, λ_j), with regression targets being $\mathbf{R}_t$ and the regressors being $[\mathbf{Z}_{t,q}' \otimes \mathbf{F}_1, \mathbf{Z}_{t,q}' \otimes \mathbf{F}_2 \ldots]$, and get the penalized estimates as $\hat{\boldsymbol{\beta}}_j^{'}$

7: **end for**

8: **Output:** $\widehat{A}_{1:p}, \widehat{B}_{1:p}, \widehat{\Sigma}_c, \widehat{\Sigma}_r, \hat{\boldsymbol{\beta}}_\mathbf{0}, \hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_J$

Figure: Ad-hoc sparse group lasso for feature selection.

Algorithm Convergence Guarantee

Theorem (Algorithm Convergence)

The block coordinate descent (BCD) algorithm guarantees that, from iteration k to $k+1$, the loss function descent, denoted as:

$$\Delta_k = \mathcal{L}(\phi^{(k+1)}) - \mathcal{L}(\phi^{(k)}), \quad \phi = (A_{1:p}, B_{1:p}, \boldsymbol{\beta}, \Sigma_c, \Sigma_r)$$

has a lower bound:

$$\Delta_k \geq \sum_{l=1}^{p} \lambda_{\min}\left(\sum_{t=p+1}^{T} \mathbf{X}_{t-l}(B_l^{(k)})' B_l^{(k)} \mathbf{X}_{t-l}'\right) \|A_l^{(k)} - A_l^{(k+1)}\|^2$$
$$\sum_{l=1}^{p} \lambda_{\min}\left(\sum_{t=p+1}^{T} \mathbf{X}_{t-l}'(A_l^{(k+1)})' A_l^{(k+1)} \mathbf{X}_{t-l}\right) \|B_l^{(k)} - B_l^{(k+1)}\|^2$$
$$\lambda_{\min}\left(\sum_{t=p+1}^{T} \mathbf{z}_{t-1}\mathbf{z}_{t-1}'\right) \cdot \lambda_{\min}\left(\sum_{\tau=1}^{K} \mathbf{f}_\tau \mathbf{f}_\tau'\right) \|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(k+1)}\|^2$$

Large Sample Asymptotics of the Estimators

Theorem (Large Sample Asymptotics)

Assume that the BCD algorithm reaches the global minimum of the empirical loss function $\widehat{\mathcal{L}}(\phi)$*, and denote the global minimum reached as* $\hat{\phi}_*$*, then with probability 1, we have:*

$$\sqrt{T}\|\hat{\phi}_* - \phi_0\| \leq c_T$$

where T *is the total number of frames of the matrix time-series,* $\{c_T\} \to +\infty$ *is an arbitrary sequence and* ϕ_0 *is the ground truth parameter of the data generating model.*

Numerical Experiments Design

To validate our proposed model and algorithm, we design two numerical experiments with simulated data:

- Non-sparse $\beta_1, \beta_2, \ldots, \beta_K$. (Non-sparse scenario)

Numerical Experiments Design

To validate our proposed model and algorithm, we design two numerical experiments with simulated data:

- Non-sparse $\beta_1, \beta_2, \ldots, \beta_K$. (Non-sparse scenario)
- Sparse $\beta_1, \beta_2, \ldots, \beta_K$. (Sparse scenario)

Scenario I: Non-sparse

We generate our simulated data with the specification of:

- 5000 frames of 3-dimensional vector time-series $\{\mathbf{z}_t\}$, generated via a stationary VAR(1) process.
- $K = 1$, a single basis function chosen from the Spherical Harmonics family.
- Spatial grid is defined using $(5i, 5j), i, j = 1, 2, \ldots, 10$.
- 5000 frames of 10×10 matrix time-series $\{\mathbf{X}_t\}$, generated via our model.
- We specify the true model with $p = q = 3$, namely the correct time lag of both the auto-regressive term and the vector covariates term are 3.

Scenario I: Non-sparse

To generate the model parameters $A_{1:3}, B_{1:3}, \beta_1, \Sigma_r, \Sigma_c$:

- $A_l, l = 1, 2, 3$ having a banded structure:

$$A_l(i,j) = \begin{cases} 0.5^{|i-j|}, & \text{if } |i-j| \leq 5 \\ 0, & \text{if } |i-j| > 5 \end{cases}$$

  and we generate $B_l, l = 1, 2, 3$ randomly from standard normal.
- The covariance structures are generated based on:

$$\Sigma_{s,ij} = \exp\left\{-\frac{|i-j|}{5}\right\}, \quad s \in \{c, r\}$$

  note that this means the variance of every matrix cell is 1.
- $\beta_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- We re-scale the $(A_l, B_l), l = 1, 2, 3$ and Σ_r, Σ_c to have $\|A_l\|_F = 1, \text{sign}(tr(A_l)) = 1$ for $l = 1, 2, 3$, and $\|\Sigma_r\|_F = 1$.

Scenario I: Non-sparse

We evaluate our model based on two major statistics:

- (Estimation Accuracy): The element-wise the root-mean-square error (RMSE) of all model parameters estimators: $\widehat{A}_l, \widehat{B}_l, \widehat{\boldsymbol{\beta}}, \widehat{\Sigma}_r, \widehat{\Sigma}_c$, after the model converges.
- (Prediction Accuracy):

$$\mathrm{RMSE}_{\mathrm{pred}} = \sqrt{\frac{1}{(T-p)mn} \sum_{t=p+1}^{T} \|\widehat{X}_t - X_t\|^2}$$

where $\widehat{X}_t$ is the one-step prediction of X_t.

Scenario I: Non-sparse (Results)

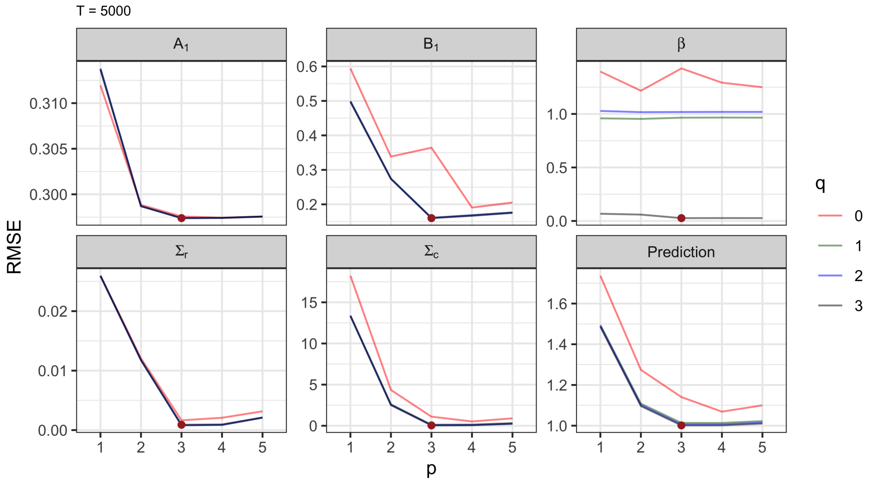

Figure: Model fitting results for $p \in \{1, 2, 3, 4, 5\}$, and $q \in \{0, 1, 2, 3\}$. Results are the average of 20 repeated model runs. The ground truth is $p = q = 3$. Round dot highlights the "correctly-specified" model.

Scenario I: Non-sparse (Results)

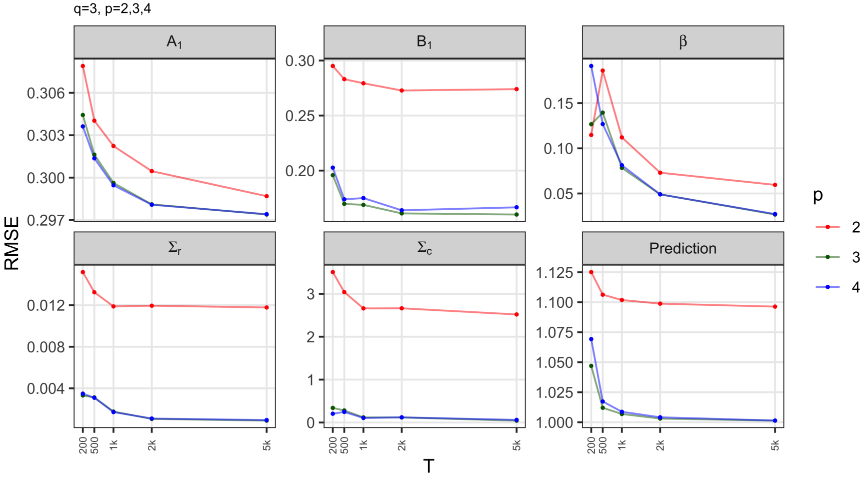

Figure: Model fitting results for $T = 200, 500, 1000, 2000, 5000$. An under-specified (red, $p = 2$) model, a correct (green, $p = 3$) model and an over-specified (blue, $p = 4$) are shown respectively.

Scenario I: Non-sparse (Results)

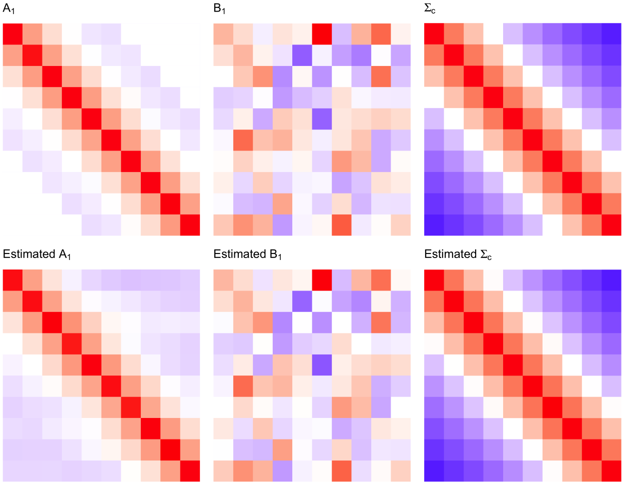

Figure: The ground truth of A_1, B_1, Σ_c (top row) and the estimated A_1, B_1, Σ_c (bottom row) for model $p = q = 3, T = 5,000$.

Scenario II: Sparse

Similar to the data generating scheme as the non-sparse case, we generate a simulated dataset with:

- 8000 frames of 3-dimensional vector time-series $\{\mathbf{z}_t\}$, generated via a stationary VAR(1) process.
- 8000 frames of 20×20 matrix time-series $\{\mathbf{X}_t\}$, generated via our model.
- Spatial grid is defined using $(5i, 5j), i, j = 1, 2, \ldots, 10$.
- We specify the true model with $p = q = 1$, namely the correct time lag of both the auto-regressive term and the vector covariates term are 1.

Scenario II: Sparse

- (Basis sparsity) $K = 30$, and 15 out of the 30 basis functions have null effect on the auto-regressive process, i.e. their $\beta_k = \mathbf{0}$. Denote the collection of these basis functions as $\mathcal{K}^0$.
- (Feature sparsity) For the remaining 15 basis functions, we coerce 40% of the elements of their corresponding β_k to be zero. Denote the collection of these basis functions as $\mathcal{K}^1$.

Scenario II: Sparse

We run our algorithm on the new simulated data and evaluate the following metrics:

- $\sum_{k \in \mathcal{K}^0 \cup \mathcal{K}^1} \mathcal{I}(|\hat{\beta}_k|_1 = 0)$: total group sparsity.
- $\sum_{k \in \mathcal{K}^0} \mathcal{I}(|\hat{\beta}_k|_1 = 0)$: total group sparsity for the truly sparse basis functions.
- $\sum_{k \in \mathcal{K}^1} \sum_d \mathcal{I}(|\hat{\beta}_{k,d}|_1 = 0)$: total feature sparsity, restricted to the non-sparse basis functions.
- $\sum_{k \in \mathcal{K}^1} \sum_d \mathcal{I}(|\hat{\beta}_{k,d}|_1 = 0 \wedge |\beta_{k,d}|_1 = 0)$: total feature sparsity out of all truly sparse features, restricted to the non-sparse basis functions.

Scenario II: Sparse (Results)

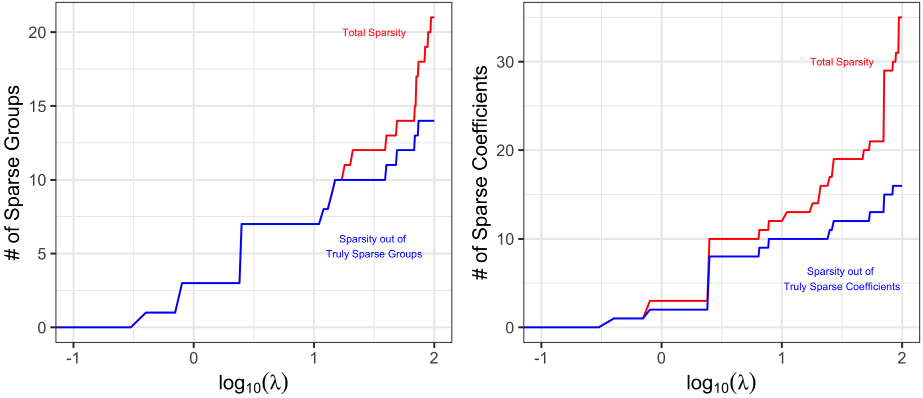

Figure: Group and Individual Sparsity Along the Solution Path: Large Sample Case ($T = 8000$). The ground truth group sparsity is 15, and the ground truth feature sparsity is 24. $\alpha = 0.95$

Real Data: The Total Electron Content Map

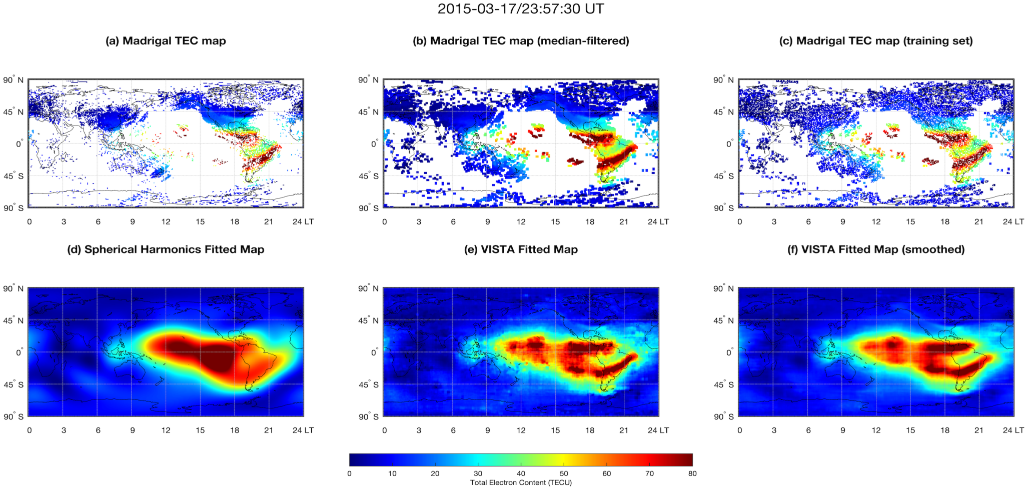

Figure: The Total Electron Content Map: Example at 23:57:30, Mar 17, 2015.

Real Data: The Total Electron Content Map

- There are $2,000+$ matrices from Jun 2017 $\sim$ Sept 2017, with individual matrix having size 181×361.
- We split the data into a train set (Jun 2017 $\sim$ Aug 2017) and a test set (Sept 2017).
- We apply our model with $p = q = 1$ and use all spherical harmonics basis at or below order 5 as our basis functions.
- The test set 1-hour prediction RMSE is 1.88 TECu, while the persistence model (simply predict $t + 1$ with t) has RMSE at 2.56 TECu.

Real Data: The Total Electron Content Map

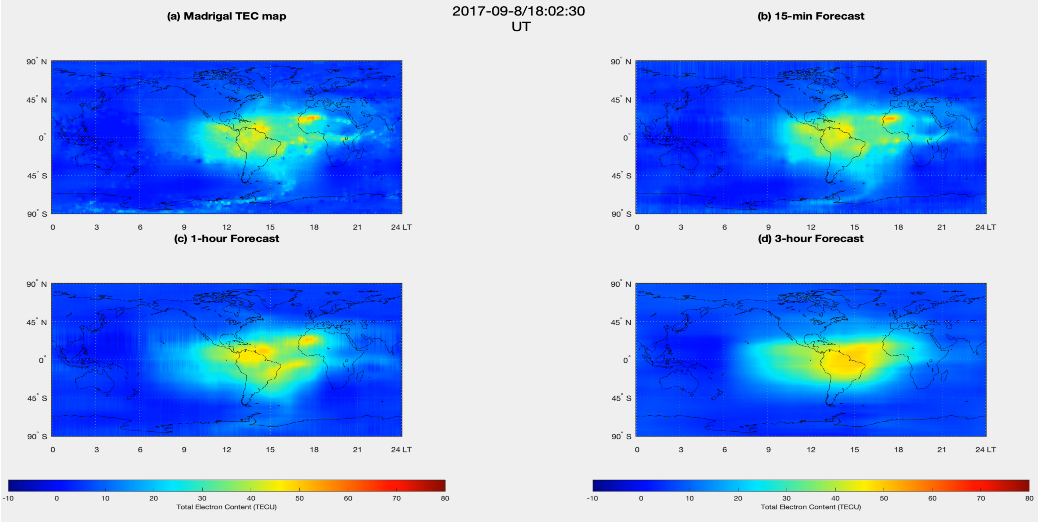

Figure: TEC Map 15-min, 1-hour, 3-hour forecasting results at 18:02:30, Sep 8, 2017.

Concluding Remarks

In this research project, we:

- Build a novel time-series auto-regressive model with matrix covariates and auxiliary vector covariates.
- Propose an optimization algorithm for model estimation, together with theoretical guarantees.
- Apply the model on simulated and real data, ended up with decent prediction performance and interpretability.

Concluding Remarks

What remains to be done/extended include:

- Find scalable implementation of the algorithm to large-scale data problem.
- Estimate the basis function using non-parametric, instead of parametric approach.
- Derive the joint asmptotics of the model parameters.

References

Chen, Rong, Han Xiao, and Dan Yang (2021). "Autoregressive models for matrix-valued time series". In: *Journal of Econometrics* 222.1, pp. 539–560.

Chen, Xinyu and Lijun Sun (2021). "Bayesian temporal factorization for multidimensional time series prediction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Hsu, Nan-Jung, Hsin-Cheng Huang, and Ruey S Tsay (2021). "Matrix autoregressive spatio-temporal models". In: *Journal of Computational and Graphical Statistics* 30.4, pp. 1143–1155.

Jing, Peiguang et al. (2018). "High-order temporal correlation model learning for time-series prediction". In: *IEEE transactions on cybernetics* 49.6, pp. 2385–2397.

Nortje, Caitlin R et al. (2015). "Spherical harmonics for surface parametrisation and remeshing". In: *Mathematical Problems in Engineering* 2015.

Simon, Noah et al. (2013). "A sparse-group lasso". In: *Journal of computational and graphical statistics* 22.2, pp. 231–245.

Wang, Dong, Xialu Liu, and Rong Chen (2019). "Factor models for matrix-valued high-dimensional time series". In: *Journal of econometrics* 208.1, pp. 231–248.