Forward-Backward Splitting for Time-Varying Graphical Models

Federico Tomasi FEDERICO.TOMASI@DIBRIS.UNIGE.IT
Veronica Tozzo VERONICA.TOZZO@DIBRIS.UNIGE.IT
Alessandro Verri ALESSANDRO.VERRI@UNIGE.IT
DIBRIS, Università degli Studi di Genova, Via Dodecaneso 35, 16146 Genova, Italy

Saverio Salzo SAVERIO.SALZO@IIT.IT
Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy

Abstract

Gaussian graphical models have received much attention in the last years, due to their flexibility and expression power. However, the optimisation of such complex models suffer from computational issues both in terms of convergence rates and memory requirements. Here, we present a forward-backward splitting (FBS) procedure for Gaussian graphical modelling of multivariate time-series which relies on recent theoretical studies ensuring convergence under mild assumptions. Our experiments show that a FBS-based implementation achieves, with very fast convergence rates, optimal results with respect to ground truth and standard methods for dynamical network inference. Optimisation algorithms which are usually exploited for network inference suffer from drawbacks when considering large sets of unknowns. Particularly for increasing data sets and model complexity, we argue for the use of fast and theoretically sound optimisation algorithms to be significant to the graphical modelling community.

Keywords: graphical inference; convex optimization; fused lasso; sparse models.

1. Introduction

The graphical modelling problem has received a lot of attention in recent years. In particular, graphical models have evolved in complexity, often challenging state-of-the-art minimisation methods. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, d\}$ is a finite set of vertices, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, an undirected *Gaussian graphical model* (GGM) is a multivariate Gaussian random variable $X = (X_1, \dots, X_d)$ on $\mathbb{R}^d$ with distribution $\mathcal{N}(\mu, \Sigma)$ where (i) there is no distinction between an edge $(i, j) \in \mathcal{E}$ and (j, i) (undirected graph), and (ii) the conditional independence between two variables X_i and X_j given all the others is encoded in $\mathcal{G}$, meaning that X_i and X_j are conditionally independent given the others if and only if $(i, j) \notin \mathcal{E}$ (Lauritzen, 1996). Moreover, the *precision matrix* $\Theta = \Sigma^{-1}$ encodes the structure of the graphs, since $(i, j) \notin \mathcal{E}$ if and only if $\Theta_{i,j} = 0$. In the context of *graphical inference*, which aims to recover the precision matrix, several models have been recently proposed. One of the best known method for such task is the *graphical lasso* (Friedman et al., 2008). Then, based on this method, different variations have been also proposed, which take into account more components, in order to be able to capture the complexity of a wide range of systems. Examples include the *latent variables graphical lasso*, which considers the presence of latent unmeasurable factors during the inference of the network (Chandrasekaran et al., 2010; Ma et al., 2013), the *joint graphical lasso*, for multi-class graph inference (Danaher et al., 2014), *time-varying graphical lasso*, to model dynamical systems by inferring multiple (while related) networks in time (Hallac et al., 2017), and the *latent variable time-varying graphical lasso*, which models a

dynamical system while taking into account latent factors that may change during the evolution of the system (Tomasi et al., 2018).

Such models, while powerful for real and complex systems, introduce non-trivial challenges from the computational point of view. In this context, a popular optimisation algorithm is the *alternating direction method of multipliers* (ADMM), which can cope with complex graphical lasso-based models (Boyd et al., 2010). ADMM partitions the problem into multiple (easier) sub-problems, so that the solution of the global problem is found as the consensus among the solutions of the smaller sub-problems. While offering a great flexibility for optimising even very complex models, a drawback of ADMM is the need of variable duplication, before finding a consensus. This may lead to a slow convergence rate and to a high computational cost (both in terms of computing resources and memory requirements). On the other hand, other options make assumptions that are not satisfied in the setting of graphical models.

Contribution In this paper, we propose a new algorithm for graphical modelling of multivariate time-series, based on the forward-backward splitting (FBS) method. We rely on recent advances on such method which introduces suitable line searches for the parameters of the algorithm and relax the assumptions, so to include the type of problems we are interested in, while maintaining strong theoretical convergence guarantees (Salzo, 2017). In particular, we focus on time-varying graphical lasso (TGL), which aims to infer a network among variables that can change during the time through specific evolutionary patterns. We cover two types of temporal transitions (Hallac et al., 2017): (i) a possibly discontinuous behaviour with few time changes in the links, by using a total variation penalty term; (ii) a smooth transition, adopting a square norm penalty term. We validate the performance of our algorithm by comparing it against the ground truth and the state-of-the-art method in this context, that is, ADMM. Our results show that our FBS-based methods are significantly faster than ADMM. Also, since FBS algorithms do not require variable duplication, the spatial complexity is lower than ADMM. This is a fundamental feature for the analysis of large networks. We emphasise the need of investigating alternative optimisation methods for such complex problems, which can deal with the increasing dimensionality of the data sets. Our work is an attempt in this direction, showing FBS-based graphical models to be a effective alternative.

Outline The rest of the paper is organised as follows. Section 2 recalls time-varying graphical lasso models and FBS minimisation algorithms. Our main contribution is detailed in Section 3.1, where we provide two alternatives FBS algorithms with line searches for two types of time-varying graphical lasso models. We extensively validated our FBS-based methods under synthetic experiments in Section 4. Finally, we conclude in Section 5 with a discussion on the results and future research directions.

2. Background

In this section we recall time-varying graphical lasso (TGL) models and recent advances on the forward-backward splitting (FBS) algorithm for the minimisation of composite convex and possibly nonsmooth objective functions.

In what follows, we denote by $\mathcal{H}$ a generic Euclidean space, and by $\langle \cdot, \cdot \rangle$ its scalar product. For every square symmetric matrix $A \in \mathbb{R}^{d \times d}$, $A \succ 0$ means that A is positive definite, and the space of such matrices is denoted by $\mathbb{S}^d_{++}$.

2.1 Time-Varying Graphical Lasso

Consider a series of observations $(\boldsymbol{x}^i(t))_{1\leq i\leq n_t}$, $\boldsymbol{x}^i(t) \in \mathbb{R}^d$, $t = 1, \ldots, T$, each one drawn from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \Sigma(t))$. Network inference aims at recovering at each time t the interaction structure $\Theta_t = \Sigma(t)^{-1}$ of the d variables, starting from n_t observations $\{\boldsymbol{x}^1(t), \ldots, \boldsymbol{x}^{n_t}(t)\}$ (Hallac et al., 2017). The goal is to find a set of precision matrices $\Theta_t \in \mathbb{S}^d_{++}$, $t = 1, \ldots, T$, that represents the dynamical network at different time points t. Then, the time-varying graphical lasso (TGL) problem is defined as follows:

$$\underset{\Theta_t \in \mathbb{S}^d_{++}}{\text{minimize}} \sum_{t=1}^{T} -n_t\, \ell(S_t, \Theta_t) + \alpha \left\|\Theta_t\right\|_{\text{od},1} + \beta \sum_{t=1}^{T-1} \psi(\Theta_{t+1} - \Theta_t), \tag{1}$$

where

- $S_t = (1/n_t) \sum_{i=1}^{n_t} \boldsymbol{x}^i(t) \otimes \boldsymbol{x}^i(t)$ is the empirical covariance matrix at time t;
- $\ell(S_t, \Theta_t) = \log\det(\Theta_t) - \operatorname{tr}(S_t\Theta_t)$ is the Gaussian log-likelihood (up to a constant and scaling factor), where Θ_t is positive definite;
- $\|\cdot\|_{od,1}$ is the off-diagonal ℓ_1-norm, which promotes sparsity in the precision matrix (excluding the diagonal);
- ψ encodes prior information on the qualitative time behaviour of the network.

Different choices of ψ reflect different evolutionary patterns of the interactions of variables in play and in (Hallac et al., 2017) several options are considered. Among these, here we focus on the ℓ_1 norm, which gives rise to a total variation penalty term, that is, $\psi(\Theta_{t+1} - \Theta_t) = \|\Theta_{t+1} - \Theta_t\|_1 = \sum_{i,j=1}^{d} |\theta_{t+1,i,j} - \theta_{t,i,j}|$ and the square of the ℓ_2 norm, meaning that $\psi(\Theta_{t+1} - \Theta_t) = \|\Theta_{t+1} - \Theta_t\|_2^2 = \sum_{i,j=1}^{d} |\theta_{t+1,i,j} - \theta_{t,i,j}|^2$. The first choice is suitable when one expects few edges to change between subsequent time points, whereas the second is appropriate when the dynamic smoothly varies over time. A solution to problem (1) has been proposed via ADMM in (Hallac et al., 2017). We stress that while favouring a relatively easy implementation for this model, ADMM requires a duplication of variables which may not always be feasible in practice, due to computational constraints, particularly for high-dimensional data.

2.2 Forward-Backward Splitting with Mild Differentiability Assumptions

Forward-backward splitting (FBS) (Combettes and Wajs, 2005) is an algorithm for the optimisation of objective functions of the following form:

$$\underset{x \in \mathcal{H}}{\text{minimize}}\, f(x) + g(x), \tag{2}$$

where $\mathcal{H}$ is an Euclidean space, f is convex and smooth, while g is convex and possibly non-smooth. The form (2) covers our objective problem (1), where depending on the different choices of ψ, the smooth part f may include the negative Gaussian log-likelihood only, or the last term too. The idea behind the method is to make a descent step of size γ towards the direction of gradient of the smooth part f (the *forward* step), then to project the point back via the proximity operator of g (the *backward* step), and finally to perform a relaxation step of size $\lambda \in\,]0, 1]$. Such algorithm

Algorithm 1: Forward-Backward splitting with Line Search (FBS-LS).

for $k = 1, \dots$ **do**
- choose $\gamma_k \in \mathbb{R}_{++}$;
- $\hat{x}^k = x^k - \gamma_k \nabla f(x^k)$;
- $y^k = \mathrm{prox}_{\gamma_k g}(\hat{x}^k) = \underset{x}{\mathrm{argmin}}\ \gamma_k g(x) + \frac{1}{2}\|x - \hat{x}^k\|^2$;
- choose $\lambda_k \in]0, 1]$;
- $x^{k+1} = x^k + \lambda_k(y^k - x^k)$;

has strong theoretical guarantees (Combettes and Wajs, 2005; Beck and Teboulle, 2009). However, these results require the smooth part f to have a Lipschitz continuous gradient on the whole space $\mathcal{H}$. This is not our case, unfortunately, since the negative Gaussian log-likelihood is defined on the open convex cone $\mathbb{S}^d_{++} \subset \mathbb{R}^{d\times d}$ and its gradient is not Lipschitz continuous. Only recently, global convergence guarantees along with rates of convergence in function values were extended to a wider class of functions (Salzo, 2017) that indeed covers our objective problem (1). Such guarantees rely on suitable line-search backtracking procedures that adaptively select the stepsize γ and/or the relaxation parameter λ, keeping the iterations inside the domain $\mathbb{S}^d_{++}$. Algorithm 1 presents a generic form of FBS. We adopt two alternative line-search strategies that are studied in (Salzo, 2017). In the first method the line search is only on the parameter γ_k, whereas the second method performs a line search on both γ_k and λ_k. For the sake of compactness we set

$$J(x, \gamma, \lambda) = x + \lambda(\mathrm{prox}_g(x - \gamma\nabla f(x)) - x),$$

so that $x^{k+1} = J(x^k, \gamma_k, \lambda_k)$.

LS(γ). Set $\lambda_k \equiv 1$ and let $\delta, \epsilon \in]0, 1[$, $\overline{\gamma} \in]0, 1]$. Then $\gamma_k = \bar{\gamma}\epsilon^i$ where i is the smallest integer so that

$$f(J(x^k, \gamma_k, 1)) - f(x^k) \leq \langle y^k - x^k, \nabla f(x^k)\rangle + \frac{\delta}{\gamma_k}\|y^k - x^k\|^2.$$

LS(γ, λ). Let $\delta, \epsilon \in]0, 1[$, $\overline{\gamma}, \overline{\lambda} \in]0, 1]$. Then $\gamma_k = \bar{\gamma}\epsilon^i$ and i is the smallest integer so that $y^k = \mathrm{prox}_{\gamma_k g}(x^k - \gamma_k \nabla f(x^k)) \in \mathbb{S}^d_{++}$ and $\lambda_k = \bar{\lambda}\epsilon^j$ where j is the smallest integer so that

$$f(J(x^k, \gamma_k, \lambda_k)) - f(x^k) \leq \lambda_k \left(\langle y^k - x^k, \nabla f(x^k)\rangle + \frac{\delta}{\gamma_k}\|y^k - x^k\|^2 \right).$$

In (Salzo, 2017) it is proved that Algorithm 1 with any of the above line searches gives a sequence $(x^k)_{k\in\mathbb{N}}$ converging to a minimiser of $f + g$ and such that $(f + g)(x^k) - \min_x(f + g)(x) = o(1/k)$.

3. Time-varying Network Inference via Forward-Backward Splitting

In this section, we formally present two problems of time-varying network inference under smooth and bounded variation temporal transitions. Then, we present our main contribution, which consists in two procedures, based on the forward-backward splitting algorithm, that are ensured to converge to a solution of the above problems.

3.1 Problem Formulation

We consider the following two time-varying graphical lasso models:

$$\underset{\Theta_t \in \mathbb{S}^d_{++}}{\text{minimize}} \sum_{t=1}^{T} -n_t\, \ell(S_t, \Theta_t) + \alpha \left\|\Theta_t\right\|_{\text{od},1} + \beta \sum_{t=1}^{T-1} \left\|\Theta_{t+1} - \Theta_t\right\|_1, \tag{TGL-ℓ_1}$$

and

$$\underset{\Theta_t \in \mathbb{S}^d_{++}}{\text{minimize}} \sum_{t=1}^{T} -n_t\, \ell(S_t, \Theta_t) + \alpha \left\|\Theta_t\right\|_{\text{od},1} + \beta \sum_{t=1}^{T-1} \left\|\Theta_{t+1} - \Theta_t\right\|_2^2, \tag{TGL-ℓ_2^2}$$

where S_t is the empirical covariance matrix, defined as in Section 2.1.
In order to put the above minimisation problems in the form (2), we set $\mathcal{H} = (\mathbb{R}^{d\times d})^T$, $\boldsymbol{\Theta} = (\Theta_t)_{1\le t\le T}$, with $\Theta_t = (\theta_{t,i,j})_{1\le i,j\le d} \in \mathbb{R}^{d\times d}$ being the precision matrix at time t, and f and g as follows:

Case (TGL-ℓ_1). $f(\boldsymbol{\Theta}) = \sum_{t=1}^{T} -n_t\, \ell(S_t, \Theta_t)$ and

$$\begin{aligned} g(\boldsymbol{\Theta}) &= \alpha \sum_{t=1}^{T} \left\|\Theta_t\right\|_{\text{od},1} + \beta \sum_{t=1}^{T} \left\|\Theta_{t+1} - \Theta_t\right\|_1 \\ &= \alpha \sum_{\substack{i,j=1 \\ i\neq j}}^{d} \sum_{t=1}^{T} |\theta_{t,i,j}| + \beta \sum_{i,j=1}^{d} \sum_{t=1}^{T} |\theta_{t+1,i,j} - \theta_{t,i,j}| \\ &= \sum_{i,j=1}^{d} \left[(1-\delta_{i,j})\alpha \left\|\theta_{\cdot,i,j}\right\|_1 + \beta TV(\theta_{\cdot,i,j})\right], \end{aligned}$$

where $\delta_{i,j}$ is the Kronecker symbol and $TV(\cdot)$ is the 1D total variation on $\mathbb{R}^T$.

Case (TGL-ℓ_2^2). $f(\boldsymbol{\Theta}) = \sum_{t=1}^{T} -n_t\, \ell(S_t, \Theta_t) + \beta \sum_{t=1}^{T-1} \left\|\Theta_{t+1} - \Theta_t\right\|_2^2$ and

$$g(\boldsymbol{\Theta}) = \alpha \sum_{t=1}^{T} \left\|\Theta_t\right\|_{\text{od},1} = \sum_{i,j=1}^{d} (1-\delta_{i,j})\alpha \left\|\theta_{\cdot,i,j}\right\|_1.$$

We note that in both cases the function g is convex and separable, meaning that

$$g(\boldsymbol{\Theta}) = \sum_{i,j=1}^{d} g_{i,j}(\boldsymbol{\theta}_{i,j}), \quad \boldsymbol{\theta}_{i,j} = (\theta_{t,i,j})_{1\le t\le T},$$

where, for every $(i,j) \in \{1,\dots,d\}^2$,

$$g_{i,j} \colon \mathbb{R}^T \to \mathbb{R}, \quad g_{i,j}(\boldsymbol{\theta}) = \begin{cases} (1-\delta_{i,j})\alpha \left\|\boldsymbol{\theta}\right\|_1 + \beta TV(\boldsymbol{\theta}) & \text{in the case (TGL-}\ell_1\text{)}, \\ (1-\delta_{i,j})\alpha \left\|\boldsymbol{\theta}\right\|_1 & \text{in the case (TGL-}\ell_2^2\text{)}. \end{cases}$$

Therefore, the proximity operator of g can be computed component-wise (Combettes and Wajs, 2005). Moreover, in the case (TGL-ℓ_1) the components of g consist in a 1D total variation penalty

if $i = j$ and in a fused lasso penalty otherwise. We stress that it is possible to exactly compute the proximity operator of such penalties by mean of a finite termination procedure (Condat, 2013). On the other hand, in the case (TGL-ℓ_2^2), the proximity operator of $g_{i,j}$ reduces to a soft-thresholding operation (Combettes and Wajs, 2005). Finally, we remark that in each of the above cases the function f is defined on the open convex cone $\mathbb{S}_{++}^d$, it is convex, and its gradient is

$$\nabla f(\boldsymbol{\Theta}) = \begin{cases} \begin{pmatrix} n_1(S_1 - \Theta_1^{-1}) \\ n_1(S_2 - \Theta_2^{-1}) \\ \cdots\cdots \\ n_T(S_T - \Theta_T^{-1}) \end{pmatrix} & \text{in the case (TGL-}\ell_1\text{)} \\ \begin{pmatrix} n_1(S_1 - \Theta_1^{-1}) \\ n_1(S_2 - \Theta_2^{-1}) \\ \cdots\cdots \\ n_T(S_T - \Theta_T^{-1}) \end{pmatrix} + 2\beta \begin{pmatrix} \Theta_1 - \Theta_2 \\ 2\Theta_2 - \Theta_1 - \Theta_3 \\ \cdots\cdots \\ \Theta_T - \Theta_{T-1} \end{pmatrix} & \text{in the case (TGL-}\ell_2^2\text{).} \end{cases} \tag{3}$$

This shows that ∇f is (only) locally Lipschitz continuous on $\mathbb{S}_{++}^d$.

3.2 Algorithm

We now give two instances of Algorithm 1 which correspond to two types of line-search procedures and can be applied to both problems (TGL-ℓ_1) and (TGL-ℓ_2^2). Algorithm 2 implements a line-search on the step-size γ only, whereas Algorithm 3 performs a line-search on the relaxation parameter λ and an additional backtracking procedure on the step-size γ in order to keep the sequence of the iterates feasible. The operations $\mathrm{prox}_{\gamma g_{i,j}}$ and ∇f are those described in Section 3.1.

We note that, since f and g are convex and ∇f is locally Lipschitz continuous on $\mathbb{S}_{++}^d$, it follows from (Salzo, 2017) that Algorithms 2 and 3 are ensured to converge with a rate $o(1/k)$. We stress that both the proposed algorithms are equivalent in terms of convergence properties and computational cost. However, in practice, they may behave differently depending on the applications, as shown in Section 4.

Stopping Criterion. Since $\boldsymbol{Y}^k = \mathrm{prox}_{\gamma_k g}(\hat{\boldsymbol{\Theta}}^k)$, it follows that $(\hat{\boldsymbol{\Theta}}^k - \boldsymbol{Y}^k)/\gamma_k \in \partial g(\boldsymbol{Y}^k)$, where ∂g is the subdifferential of g. Our stopping criterion is based on the following residual

$$\boldsymbol{R}^k = \nabla f(\boldsymbol{Y}^k) + \frac{\hat{\boldsymbol{\Theta}}^k - \boldsymbol{Y}^k}{\gamma_k}, \tag{4}$$

which belongs to $\partial(f+g)(\boldsymbol{Y}^k)$ and, in view of the expression of $\hat{\boldsymbol{\Theta}}$ in Algorithm 2 and 3, can also be written as

$$\boldsymbol{R}^k = \nabla f(\boldsymbol{Y}^k) - \nabla f(\boldsymbol{\Theta}^k) + \frac{\boldsymbol{\Theta}^k - \boldsymbol{Y}^k}{\gamma_k}. \tag{5}$$

Since $\boldsymbol{Y}^k - \boldsymbol{\Theta}^k \to 0$ as $k \to +\infty$ (Salzo, 2017) and ∇f is locally Lipschitz continuous, it follows that $(\boldsymbol{R}^k)_{k\in\mathbb{N}}$ is a sequence of subgradients of $f+g$ (each one at the point $\boldsymbol{Y}^k$), that converges to

Algorithm 2: FBS-LS(γ) for time-varying network inference.

choose $\epsilon \in]0,1[$, $\bar{\gamma} > 0$, and $\delta \in]0,1[$;
choose $\boldsymbol{\Theta}^0$ and set $\gamma_{-1} = \bar{\gamma}\epsilon$;
for $k = 0, 1, \ldots$ *(until convergence)* **do**
  initialise $\gamma = \gamma_{k-1}/\epsilon$;
  do
    $\gamma \leftarrow \gamma\epsilon$;
    $\hat{\boldsymbol{\Theta}}^k = \boldsymbol{\Theta}^k - \gamma \nabla f(\boldsymbol{\Theta}^k) = (\hat{\boldsymbol{\theta}}^k_{ij})_{1 \leq i,j \leq d}$;
    for *each interaction* ij **do**
      $\boldsymbol{y}^k_{i,j} = \mathrm{prox}_{\gamma g_{i,j}}(\hat{\boldsymbol{\theta}}^k_{ij})$;
    $\boldsymbol{Y}^k = (\boldsymbol{y}^k_{i,j})_{1 \leq i,j,\leq d}$;
    $\zeta_k = \langle \boldsymbol{Y}^k - \boldsymbol{\Theta}^k, \nabla f(\boldsymbol{\Theta}^k) \rangle + (\delta/\gamma) \left\| \boldsymbol{Y}^k - \boldsymbol{\Theta}^k \right\|_2^2$;
  while $Y^k_1 \not\succ 0$ *or* $Y^k_2 \not\succ 0, \cdots$ *, or* $Y^k_T \not\succ 0$ *or* $f(\boldsymbol{Y}^k) - f(\boldsymbol{\Theta}^k) > \zeta_k$;
  $\gamma_k = \gamma$;
  $\boldsymbol{\Theta}^{k+1} = \boldsymbol{Y}^k$;

Algorithm 3: FBS-LS(γ, λ) for time-varying network inference.

choose $\epsilon \in]0,1[$, $\bar{\gamma} > 0$, $\bar{\lambda} \in]0,1]$, and $\delta \in]0,1[$;
choose $\boldsymbol{\Theta}^0$ and set $\gamma_{-1} = \bar{\gamma}\epsilon$, $\lambda_0 = \bar{\lambda}$;
for $k = 0, 1, \ldots$ *(until convergence)* **do**
  initialise $\gamma = \gamma_{k-1}/\epsilon$ and $\lambda = \lambda_{k-1}/\epsilon$;
  do
    $\gamma \leftarrow \gamma\epsilon$;
    $\hat{\boldsymbol{\Theta}}^k = \boldsymbol{\Theta}^k - \gamma \nabla f(\boldsymbol{\Theta}^k) = (\hat{\boldsymbol{\theta}}^k_{ij})_{1 \leq i,j \leq d}$;
    for *each interaction* ij **do**
      $\boldsymbol{y}^k_{i,j} = \mathrm{prox}_{\gamma g_{i,j}}(\hat{\boldsymbol{\theta}}^k_{ij})$;
    $\boldsymbol{Y}^k = (\boldsymbol{y}^k_{i,j})_{1 \leq i,j,\leq d} = (Y^k_1, \ldots, Y^k_T)$;
  while $Y^k_1 \not\succ 0$ *or* $Y^k_2 \not\succ 0, \cdots$ *, or* $Y^k_T \not\succ 0$;
  do
    $\lambda \leftarrow \lambda\epsilon$;
    $\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}^k + \lambda(\boldsymbol{Y}^k - \boldsymbol{\Theta}^k)$;
    $\zeta_k = \langle \boldsymbol{Y}^k - \boldsymbol{\Theta}^k, \nabla f(\boldsymbol{\Theta}^k) \rangle + (\delta/\gamma) \left\| \boldsymbol{Y}^k - \boldsymbol{\Theta}^k \right\|_2^2$;
  while $f(\boldsymbol{\Theta}^{k+1}) - f(\boldsymbol{\Theta}^k) > \lambda\zeta_k$;
  $\gamma_k = \gamma$;

zero. A scale invariant stopping criterion can be obtained by adopting the condition $r^k_r < \epsilon_{\text{abs}}$ or

$r_n^k < \epsilon_{\text{abs}}$, where

$$r_r^k = \frac{\left\|\boldsymbol{R}^k\right\|_2}{\max\left\{ \left\|\nabla f(\boldsymbol{Y}^k)\right\|_2, \left\|\frac{\hat{\boldsymbol{\Theta}}^k - \boldsymbol{Y}^k}{\gamma^k}\right\|_2 \right\} + \epsilon_r} \qquad \text{and} \qquad r_n^k = \frac{\left\|\boldsymbol{R}^k\right\|_2}{\left\|\boldsymbol{R}^1\right\|_2 + \epsilon_n},$$

are the *relative* and *normalised* residuals respectively, ϵ_{abs} is an absolute tolerance parameter, and ϵ_r and ϵ_n are small constants to prevent the denominator from being zero (Goldstein et al., 2014).

3.3 Complexity

In Algorithms 2 and 3 the most expensive step lies in the inversion of T matrices, required by the gradient of f (Equation (3)). The complexity per iteration is equivalent to that of the ADMM proposed by Hallac et al. (2017). Indeed, the optimisation of TGL with ADMM requires an eigenvalue decomposition at each iteration. Hence, both minimisation methods have complexity of $O(Td^3)$ per iteration. We recall that both the matrix inversion problem and the eigenvalue decomposition problem can be solved by more efficient algorithms with lower complexity. Employing such algorithms can significantly improve the time performance of our FBS-based algorithms as well as of ADMM. Finally, exploiting a particular structure of the $\boldsymbol{\Theta}$ matrix (such as a block structure) may be an additional benefit in the matrix inversion operation. We leave such improvements as a future work.

4. Experiments

The performance of the proposed methods has been assessed on synthetic data in terms of the number of iterations, execution time, and space scalability. In particular, we compared the two proposed algorithms FBS-LS(γ) (Algorithm 2) and FBS-LS(γ, λ) (Algorithm 3) with the ADMM algorithm, which is the state of the art in the context of time-varying network inference (Hallac et al., 2017).

4.1 Convergence

Data were generated starting from a set of precision matrices $\boldsymbol{\Theta} = (\Theta_1, \ldots, \Theta_T)$, related in time according to a specific behaviour while guaranteeing that $\Theta_t \in \mathbb{S}_{++}^d$ for $t = 1, \ldots, T$. In particular, we generated two data sets according to different temporal behaviours, consisting of $n_t = 200$ samples in $\mathbb{R}^d$ with $d = 200$ and $T = 10$ time stamps. The first data set was obtained by modelling the interactions between variables across time according to a square waveform. Under such schema, the interactions may be zero or positive at particular time points, but the transition between those states is non-smooth. The second data set is generated modelling variable interactions according to a smooth sinusoidal behaviour. Hence, the interactions were constrained to change slowly in time. Additional details on data generation can be found in the implementation (see Section 6).
We considered the time-varying graphical lasso with the two temporal penalties (TGL-ℓ_1) and (TGL-ℓ_2^2), according to the type of the data set. As for the hyperparameters (α, β), we considered the search space $[0.1, 1] \times [0.1, 5]$ for (TGL-ℓ_1) and $[0.1, 1] \times [0.01, 0.1]$ for (TGL-ℓ_2^2). We performed a Bayesian optimisation procedure, and we checked that the best hyperparameters lie in the interior of the search space (do not belong to the boundary). In particular, $(\alpha^*, \beta^*) = (0.111, 4.855)$ for (TGL-ℓ_1), while $(\alpha^*, \beta^*) = (0.789, 0.020)$ for (TGL-ℓ_2^2). Then, we set a grid on the search

	precision	0.1		0.01		0.001	
	score	iter.	time [s]	iter.	time [s]	iter.	time [s]
ℓ_1	FBS-LS(γ)	22 ± 1	4.8 ± 0.7	24 ± 1	5.1 ± 0.4	26 ± 1	5.0 ± 0.3
	FBS-LS(γ,λ)	22 ± 1	4.6 ± 0.7	24 ± 1	5.4 ± 0.5	26 ± 1	5.6 ± 0.5
	ADMM	1060 ± 553	75.2 ± 39.8	2623 ± 1757	184.3 ± 122.3	4312 ± 1536	301.6 ± 107.4
ℓ_2^2	FBS-LS(γ)	72 ± 19	7.9 ± 2.7	107 ± 30	11.7 ± 4.1	137 ± 40	14.9 ± 5.5
	FBS-LS(γ, λ)	72 ± 19	8.3 ± 2.8	104 ± 31	12.1 ± 4.5	129 ± 41	14.9 ± 6.0
	ADMM	192 ± 24	13.2 ± 1.7	252 ± 42	17.3 ± 3.1	453 ± 66	30.4 ± 3.8

Table 1: Comparison between FBS with line search and ADMM. We ran the algorithms for several values of (α, β). The table displays the average and standard deviation of the number of iterations and CPU times across the different runs for achieving ${}^{|\text{obj}_k - m_*|}/_{|m_*|} \leq \varepsilon$, with $\varepsilon \in \{0.1, 0.01, 0.001\}$. For each pair of hyperparameters, the minimum m_* is estimated as the best value obtained in 500 iterations among the different algorithms.

space and ran the two proposed algorithms FBS-LS(γ) and FBS-LS(γ, λ) as well as ADMM, for the corresponding values of the hyperparameters.

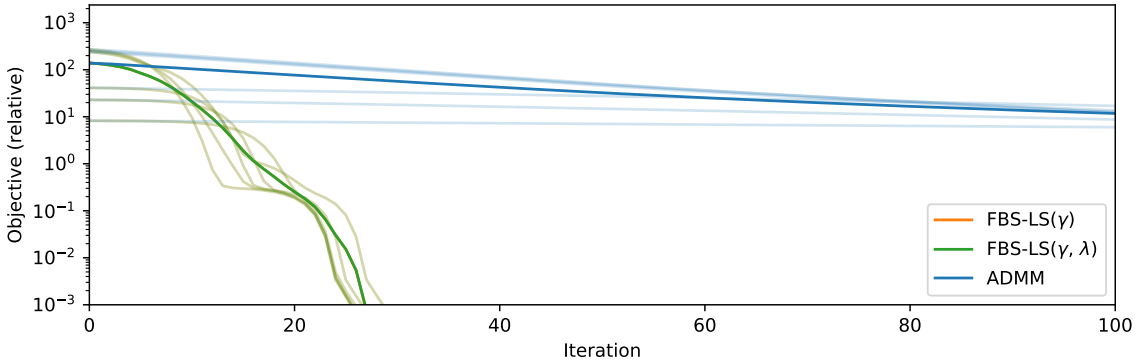
We evaluated the performance of the proposed methods with respect to the ground truth. We computed the mean squared error (MSE) for each algorithm after convergence, defined as MSE $= \frac{2}{Td(d-1)} \sum_{t=1}^{T} \|\Theta_t^{(u)} - \tilde{\Theta}_t^{(u)}\|_2^2$, where Θ_t denotes the ground truth, $\tilde{\Theta}_t$ is the inferred precision matrix, and the superscript (u) refers to the upper triangular part of the matrix (excluding the diagonal). The achieved MSE was the same for each algorithm ($0.648 \cdot 10^{-4}$ for (TGL-ℓ_1), $0.498 \cdot 10^{-4}$ for (TGL-ℓ_2^2)).

Table 1 reports the performance of the three algorithms across the different runs in terms of the number of iterations and CPU times for achieving a given precision. In this experiment, both FBS-based algorithms clearly outperform the ADMM. FBS-based algorithms are able, in only a few iterations, to increase the precision of order of magnitudes, for both ℓ_1 and ℓ_2^2 set of experiments. We note, however, that the difference in the convergence behaviour with respect to ADMM is less substantial in the case of ℓ_2^2. In the case of ℓ_1, FBS has a higher cost per iteration with respect to ADMM. This is due to the computation of the proximity operator of the fused lasso penalty. In the case of ℓ_2^2, instead, the cost is lower because the proximity operator of the nonsmooth (penalty) term simplifies to a soft-thresholding. Finally, for (TGL-ℓ_2^2), we point out the better performance of FBS-LS(γ, λ) against FBS-LS(γ).

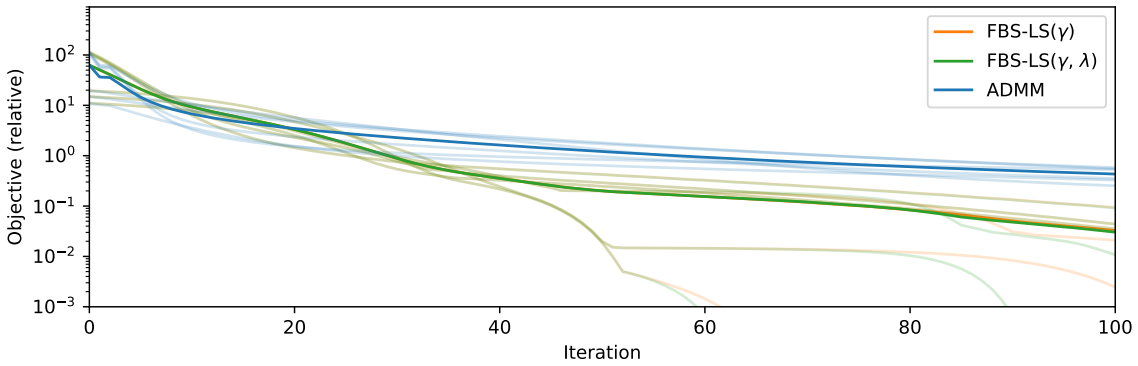
Figure 1 shows the relative objective value across the first 100 iterations and multiple runs for FBS-based algorithms and ADMM. The averaged value is depicted in bold line. In particular, in the case of the (TGL-ℓ_1), FBS-based algorithms clearly surpass the ADMM in terms of convergence rate (Figure 1a). We note that the two algorithms FBS-LS(γ) and FBS-LS(γ, λ) completely overlap in the case of (TGL-ℓ_1), whereas FBS-LS(γ, λ) shows to convergence slightly faster than FBS-LS(γ). The poor convergence rate of ADMM may be due to the need of reaching a consensus among a large number of variables which a typical scenario in the inference of time-varying networks.

4.2 Scalability

FBS- and ADMM-based optimisations feature different memory requirements. In particular, FBS-based implementation requires $O(2d^2T)$ in space, for keeping in memory both the precision and empirical covariance matrices at all time points. Instead, ADMM-based implementation requires



(a) ℓ_1-norm temporal penalty.



(b) ℓ_2^2-norm temporal penalty.

Figure 1: Relative objective value (decreasing) at each iteration. The relative value is obtained as $|\text{obj}_k - m_*|/|m_*|$, where m_* is the minimum objective value obtained across 500 iterations, and obj_k is the value of the objective function at iteration k. In both cases, FBS-based algorithms converge to the minimum faster with respect to ADMM.

more variables due to the consensus framework and the presence of dual variables. More specifically, in our setting, it requires $O(4d^2(2T-1))$ space complexity (Hallac et al., 2017). The difference between the two complexities consists in a multiplicative factor which, however, may have an impact in the analysis of large data sets.

Figure 2 shows the difference in space complexity as the number of unknowns ($Td(d+1)/2$) of the problem grows. We note that such computations do not take into account the use of optimised data structures for sparse data. Better performance may be achieved by exploiting the structure and the sparsity of the involved matrices, but we leave such investigation for future work.

4.3 Model Selection

The hyperparameters of the methods have been selected by using a cross-validation procedure. In particular, we used the Monte Carlo cross-validation that repeatedly splits the data set in two mutually exclusive sets, namely *learning* and *test* sets (Molinaro et al., 2005). For each hyperparameter combination, the model was trained on the learning set and the likelihood of the model was esti-

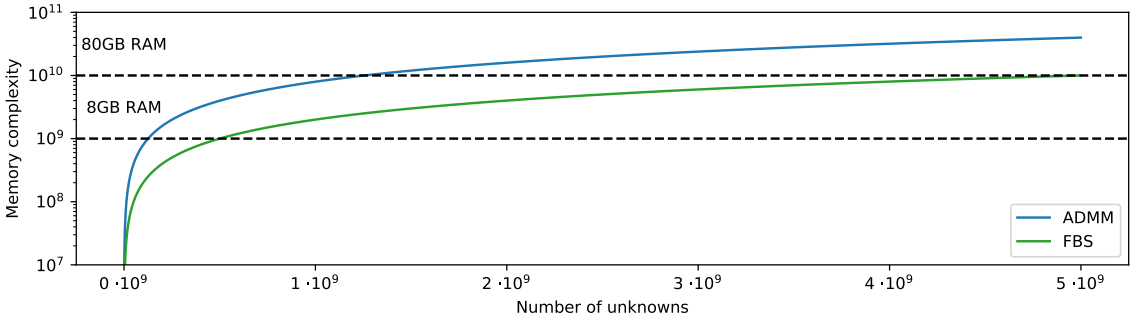


Figure 2: Memory requirements as the number of unknowns grows, with $T = 50$ and d varying. Each matrix entry is stored in double precision.

mated on the independent test set. We selected the combination of hyperparameters based on the average maximum likelihood of the model across multiple splits of the data set. Then, the selected hyperparameters were used for both FBS- and ADMM-based algorithms, for which the functional is the same. Since the number of TGL-FB hyperparameters is large, we used a Gaussian process-based Bayesian optimisation procedure to choose the best combination of hyperparameters for each data set, based on the expected improvement strategy (Snoek et al., 2012).

We note that, in real cases when the temporal dynamics is not known, it is possible and advantageous to include the choice of the temporal behaviour in a cross-validation procedure.

5. Conclusions and Future Work

In this work, we presented a new algorithm for the graphical modelling of multivariate time-series, which is based on a forward-backward splitting procedure. We cover two significant types of temporal behaviours, that is, a nonsmooth transition with few interaction changes, and a smooth transition where the system evolves slowly. Our experiments proved that the proposed method is more efficient than ADMM in terms of number of iterations, CPU time and memory requirements.

Further improvements on our contribution may be achieved by exploiting the structure of the involved matrices (*e.g.*, the block structure of precision matrices) in order to increase the efficiency in the computation of ∇f. Also, we plan to apply the FBS algorithm on different graphical models, such as those considered by Danaher et al. (2014) and Tomasi et al. (2018). We expect that, when increasing the complexity of models, FBS-based graphical models would prove even more effective.

We believe that this work could pave the way to develop solid graphical models with increasing complexity, leading to further advances in pattern recognition.

6. Availability and Implementation

The minimisation algorithm is implemented in an open-source Python framework, available under BSD-3-Clause at `https://github.com/fdtomasi/regain`. It is fully compatible with the *scikit-learn* library of machine learning algorithms. It provides a straightforward and intuitive interface, while relying on low-level high-performance libraries for numerical computations that lead to a fast and scalable optimisation algorithm, even with an increasing number of unknowns.

References

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, 3 (1), 2010. ISSN 1935-8237. doi: 10.1561/2200000016. URL http://www.nowpublishers.com/article/Details/MAL-016.

V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky. Latent variable graphical model selection via convex optimization. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 1610–1613. IEEE, 2010.

P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

L. Condat. A direct algorithm for 1-d total variation denoising. *IEEE Signal Processing Letters*, 20(11): 1054–1057, 2013.

P. Danaher, P. Wang, and D. M. Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 76(2):373–397, 2014. ISSN 13697412. doi: 10.1111/rssb.12033.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

T. Goldstein, C. Studer, and R. Baraniuk. A field guide to forward-backward splitting with a fasta implementation. *arXiv preprint arXiv:1411.3406*, 2014.

D. Hallac, Y. Park, S. Boyd, and J. Leskovec. Network inference via the time-varying graphical lasso. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 205–213, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098037. URL http://doi.acm.org/10.1145/3097983.3098037.

S. L. Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

S. Ma, L. Xue, and H. Zou. Alternating Direction Methods for Latent Variable Gaussian Graphical Model Selection. *Neural Computation*, 25(8):2172–2198, aug 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00379. URL http://www.mitpressjournals.org/doi/10.1162/NECO{_}a{_}00379.

A. M. Molinaro, R. Simon, and R. M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 2005.

S. Salzo. The variable metric forward-backward splitting algorithm under mild differentiability assumptions. *SIAM Journal on Optimization*, 27(4):2153–2181, 2017.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

F. Tomasi, V. Tozzo, S. Salzo, and A. Verri. Latent variable time-varying network inference. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 2338–2346, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3220121. URL http://doi.acm.org/10.1145/3219819.3220121.