

Sparse and Low-Rank Matrix Regularization for Learning Time-Varying Markov Networks

Jun-ichiro Hirayama^{1,*}, Aapo Hyvärinen^{2,1}, and Shin Ishii^{3,1}

¹*Brain Information Communication Research Laboratory Group, Advanced Telecommunications Research Institute International (ATR), Kyoto, Japan*

²*Department of Computer Science and HIIT, University of Helsinki, Helsinki, Finland*

³*Graduate School of Informatics, Kyoto University, Kyoto, Japan*

*hirayama@atr.jp

Abstract

Statistical dependencies observed in real-world phenomena often change drastically with time. Graphical dependency models, such as the Markov networks (MNs), must deal with this temporal heterogeneity in order to draw meaningful conclusions about the transient nature of the target phenomena. However, in practice, the estimation of time-varying dependency graphs can be inefficient due to the potentially large number of parameters of interest. To overcome this problem, we propose such a novel approach to learning time-varying MNs that effectively reduces the number of parameters by constraining the rank of the parameter matrix. The underlying idea is that the effective dimensionality of the parameter space is relatively low in many realistic situations. Temporal smoothness and sparsity of the network are also incorporated as in previous studies. The proposed method is formulated as a convex minimization of a smoothed empirical loss with both the ℓ_1 - and the nuclear-norm regularization terms. This non-smooth optimization problem is numerically solved by the alternating direction method of multipliers (ADMM). We take the Ising model as a fundamental example of an MN, and we show in several simulation studies that the rank-reducing effect from the nuclear norm can improve the estimation performance of time-varying dependency graphs. We also demonstrate the utility of the method for analyzing real-world datasets for improving the interpretability and predictability of the obtained networks.

1 Introduction

The sparse Markov Network (MN) [37, 52, 32, 18, 3, 23] is a powerful model for analyzing statistical dependency in multivariate data. Like the ordinary MN, it describes conditional (in)dependences between random variables of interest in terms of an undirected graph, quantified by model parameters associated with the graph. A sparse MN, in particular, learns the graph structure and parameters using *sparse estimation* techniques, such as the ℓ_1 -norm regularization [48, 11], formulated as a convex optimization that can be solved efficiently. This computational efficiency of the sparse MN is a great practical advantage, when compared to classical methods based on combinatorial optimization of the graph structure.

A challenging issue with the sparse MNs arises when attempting to deal with the dependency graph that is potentially heterogeneous in a given dataset. In particular, statistical dependency observed in dynamical phenomena often changes drastically with time. The dependency graphs varying over time then must be considered in order to analyze any transient nature of the target

phenomena. The conventional “static” case of the sparse MNs is not useful for this purpose, as it assumes that the dependency graph is consistent for all time points.

However, estimating time-varying dependency graphs is rather difficult due to the potentially large number of parameters of interest: the number of edges or their weights grows quadratically with the number of variables and linearly with the sample size (i.e., the number of time points). This high degree of freedom of the model typically degrades the parameter estimation due to relative insufficiency of data. Previous studies on the time-varying MNs [46, 28, 58, 29] have addressed this issue, by assuming that no substantial changes occur in the network over a relatively short time period, which is likely to be reasonable in many cases.

In the present study ¹, we propose a new framework for learning time-varying MNs, focusing on particular situations where the parameter *matrix* of the time-varying MN can be reasonably assumed to be of relatively *low rank* according to some prior knowledge. As illustrated in Fig. 1, every instance of the dependency graph of a time-varying MN is described by a single column of the parameter matrix Θ . The low-rank assumption of Θ equivalently means that the columns and the rows of Θ are constrained to low-dimensional subspaces. Intuitively, the basis vectors in the column and row subspaces correspond to the patterns in the dependency graph and to the time courses of these patterns’ coefficients, respectively. The low rank then implies that the number of these patterns is relatively small. Although, in reality, the true Θ may also contain unstructured noise and thus may not be exactly rank-deficient, we can still expect that the true Θ is well approximated by a low rank matrix in many situations.

We should emphasize here that we do not seek a method that works in every situation or for every dataset. Rather, we focus on improving the estimation in specific situations where the low-rank assumption is reasonable. However, we still expect the method to have a wide range of applicability covering many practical situations. An example of this type of situation, suggested directly by the visualization in Fig. 1, is a network with a *community* (or cluster) structure [38]. The low-rank property will emerge if the overall changes in the network are dominated by dynamic activities of major communities that produce specific dependency patterns in the observed data. Another important situation is *regime switching*, a well-known concept in time series analysis in which the network visits a limited number of regimes, possibly many times. Actually, the idea of the low rank parameter matrix is a generalization of regime switching because now each regime is not only selected from a finite set of dependency patterns but each may also be any linear combination of them.

Here, we focus on the time-varying Ising model [46, 28], a fundamental instance of time-varying MNs, and extend its basic framework of sparse estimation [46, 28], formulated as such to employ ℓ_1 -norm regularization and kernel smoothing. In particular, we introduce the nuclear (trace) norm regularization [15, 47] into this basic framework, which has been widely used for learning low-rank matrices in recent years. The proposed *sparse and low-rank estimation* framework of time-varying MNs can still be formulated as a convex optimization and thus solved efficiently. We also introduce an optimization algorithm based on the alternating direction method of multipliers (ADMM) [4, 14, 7], which solves the problem by effectively splitting the two non-smooth regularization terms.

The remainder of this paper is organized as follows. We first briefly introduce the Ising model and the sparse estimation method for this model, using the pseudolikelihood [5, 6] (Section 2). We then propose our new method for learning time-varying MNs based on the low-rank assumption (Section 3), and we briefly summarize related studies (Section 4). We also present simulation results using several artificially-generated datasets (Section 5), and demonstrate the applicability of the proposed method to a real-world problem (Section 6). Finally, we discuss

¹A preliminary version of this paper appeared in a conference proceeding [22], which introduced the basic algorithm and provided a simple simulation result (included as Section 5.1 in this extended version).

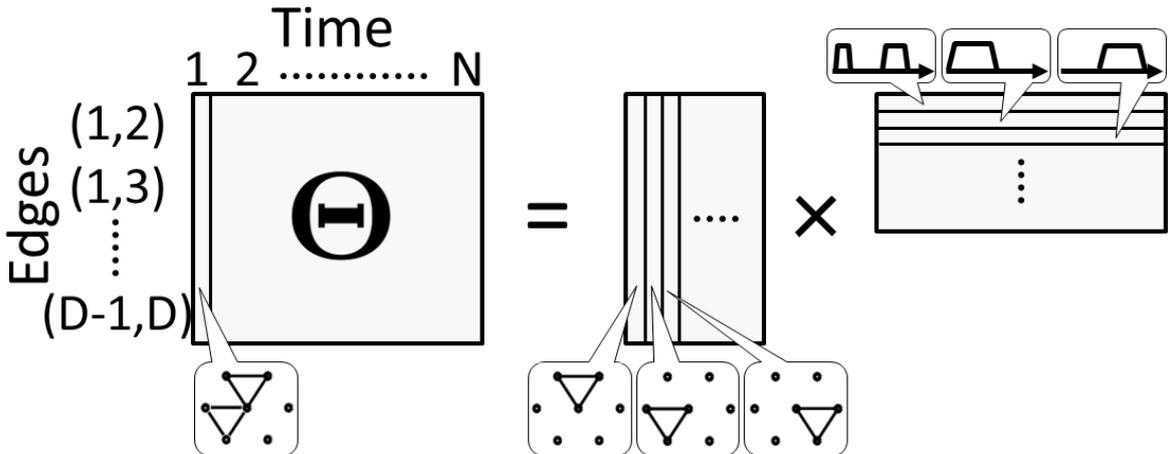


Figure 1: Illustration of the basic idea. In a time-varying Markov Network, the n -th instance of (undirected) dependency graph ($n = 1, 2, \dots, N$) is represented by the n -th column of the parameter matrix Θ , depicted as a block on the left-hand side of the equality. Each element of these column vectors corresponds, for example, to an edge weight between two particular nodes (out of D nodes), which describes a pairwise interaction between two random variables. Here, in particular, we assume that the parameter matrix has a relatively low rank, implicitly decomposed into the two matrices; i.e., the dependency patterns and the coefficient time-series, depicted as the first and second blocks on the right-hand side, respectively.

the results and open issues (Section 7).

2 Preliminaries

In this section, we introduce the Ising model and its sparse estimation using pseudolikelihood.

2.1 Ising model

An MN represents conditional independences between random variables by means of an undirected graph, where the parameters associated with cliques (i.e., complete subgraphs) quantify the full probability distribution of the random variables [40, 31]. The Ising model is one of the simplest examples of an MN, having only parameters associated with nodes and edges so that it represents, at most, pairwise interactions between binary variables.

Let $\mathbf{y} = (y_1, y_2, \dots, y_D)^\top$ be a D -dimensional binary random vector that takes a value in $\{-1, 1\}^D$. The probability distribution of \mathbf{y} under the Ising model, parameterized by a real-valued vector $\boldsymbol{\theta}$, is then defined by

$$p(\mathbf{y}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{i < j} \theta_{ij} y_i y_j + \sum_i \theta_i y_i\right), \quad (1)$$

where the first summation in the exponent is taken over all pairs (i, j) that satisfy $i < j$ and the partition function, $Z(\boldsymbol{\theta}) = \sum_{\mathbf{y}} \exp\left(\sum_{i < j} \theta_{ij} y_i y_j + \sum_i \theta_i y_i\right)$, simply makes the total probability equal to unity. The parameter vector $\boldsymbol{\theta}$ contains the $C (= D(D + 1)/2)$ network parameters, θ_i 's and θ_{ij} 's, in a certain fixed order. Here, we refer to θ_i 's and θ_{ij} 's as node-wise and edge-wise network parameters (or “edge weights”), respectively. For convenience, we also introduce

auxiliary variables θ_{ji} for any $i < j$, and use θ_{ij} and θ_{ji} interchangeably by implicitly assuming $\theta_{ji} = \theta_{ij}$.

The undirected graph associated with this model has D nodes, each of which corresponds to a single variable y_i ($i = 1, 2, \dots, D$). The graph has an undirected edge (i, j) between the i -th and j -th nodes if θ_{ij} is non-zero, and otherwise has no edge. If the two nodes are connected (disconnected), then the variables y_i and y_j are conditionally dependent (independent), given their Markov blanket (i.e., all other variables that have direct connections to either or both of these variables). The dependence structure of the constituent binary variables is then specified by the graph structure (i.e., which edge-wise parameters are zero and which are non-zero).

2.2 Estimation by maximum pseudolikelihood

In the Ising model, as in many other MNs, computing the likelihood or its gradient is often difficult due to the intractability of the partition function $Z(\boldsymbol{\theta})$, unless D is sufficiently small. One approach for overcoming the intractability of likelihood is to use the pseudolikelihood [5, 6, 25, 30]; many variants are in popular use in the field of machine learning [52, 42, 23, 20].

The pseudolikelihood for a parametric model $p(\mathbf{y}; \boldsymbol{\theta})$, given a single datum \mathbf{y} , is generally defined as a product of conditional likelihoods; i.e., $\prod_{i=1}^D p(y_i | \mathbf{y}_{\setminus i}; \boldsymbol{\theta})$. Here, $\mathbf{y}_{\setminus i}$ denotes all the variables of interest other than y_i . The corresponding loss function $l(\mathbf{y}, \boldsymbol{\theta})$ is introduced as the negative logarithm of the pseudolikelihood, which is given specifically for the Ising model by

$$l(\mathbf{y}, \boldsymbol{\theta}) = - \sum_{i=1}^D \log p(y_i | \mathbf{y}_{\setminus i}; \boldsymbol{\theta}) = \sum_{i=1}^D \{ \log \cosh(\xi_i) - y_i \xi_i + \log 2 \}, \quad (2)$$

where $\xi_i := \sum_{j:j \neq i} \theta_{ij} y_j + \theta_i$ and $\cosh(x) = (e^x + e^{-x})/2$. Its partial derivatives needed for numerical optimization are given by

$$\frac{\partial}{\partial \theta_{ij}} l(\mathbf{y}, \boldsymbol{\theta}) = \tanh(\xi_i) y_j + y_i \tanh(\xi_j) - 2 y_i y_j, \quad \text{for any } i < j, \quad (3a)$$

$$\frac{\partial}{\partial \theta_i} l(\mathbf{y}, \boldsymbol{\theta}) = \tanh(\xi_i) - y_i, \quad \text{for any } i, \quad (3b)$$

where $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$. The minimization of the empirical loss, $\sum_{n=1}^N l(\mathbf{y}^n, \boldsymbol{\theta})$, then yields the maximum pseudolikelihood estimator of $\boldsymbol{\theta}$, which has been shown to be consistent [25]. Note that taking the second derivatives readily reveals that this loss function l is convex with respect to $\boldsymbol{\theta}$.

2.3 Sparse estimation using ℓ_1 -norm regularization

Determination of the graph structure can be facilitated by accommodating sparse estimation based on the ℓ_1 -norm regularization [37, 52, 32, 18, 3, 23] with the maximum pseudolikelihood. In a continuous optimization of the parameter vector $\boldsymbol{\theta}$, the additional regularization term encourages the optimal $\boldsymbol{\theta}$ to have many entries that are exactly zero (i.e., effectively determining the graph structure).

The loss function introduced above now formulates the sparse estimation of the Ising model as a convex minimization problem, such that

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{N} \sum_{n=1}^N l(\mathbf{y}^n, \boldsymbol{\theta}) + \|\boldsymbol{\lambda} \odot \boldsymbol{\theta}\|_1, \quad (4)$$

where $\|\cdot\|_1$ denotes the ℓ_1 -norm (i.e., the sum of all absolute values of the elements), $\boldsymbol{\lambda}$ is a vector of regularization coefficients, and \odot denotes element-wise multiplication. The objective function is convex, since both the loss function l and the ℓ_1 -norm are convex with respect to $\boldsymbol{\theta}$ and any sum of convex functions is again convex [8].

The second term of Eq. (4) can also be written as

$$\|\boldsymbol{\lambda} \odot \boldsymbol{\theta}\|_1 = \sum_{i < j} \lambda_{ij} |\theta_{ij}| + \sum_i \lambda_i |\theta_i|. \quad (5)$$

We typically use $\lambda_i = 0$ for any i , because only the edge-wise parameters are relevant for making the graph structure sparse. In addition, for simplicity, we often use a common λ for any $i < j$.

3 Proposed method

The time-varying MN [46, 1, 28, 58] extends the sparse MN to incorporate potential heterogeneity of the dependency graph over time. It assumes that N successive observations or measurements $\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^N$ are generated independently from the MNs $p(\mathbf{y}^n; \boldsymbol{\theta}^n)$ with different parameter vectors $\boldsymbol{\theta}^n$ at each time step. The corresponding time-varying dependency graphs are then associated with the parameter matrix $\boldsymbol{\Theta} = (\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \dots, \boldsymbol{\theta}^N) \in \mathbb{R}^{C \times N}$, and each column specifies an instance of the MN. The above sparse estimation framework now requires the incorporation of additional assumptions about $\boldsymbol{\Theta}$ other than sparsity; otherwise, each instance of the MN $p(\mathbf{y}^n; \boldsymbol{\theta}^n)$ is estimated only with a single datum \mathbf{y}^n .

In this section, we extend a previous approach [46, 58, 28] by incorporating an additional assumption whereby the parameter matrix $\boldsymbol{\Theta}$ has a relatively low rank. Based on this assumption, we formulate the sparse and low-rank estimation of the time-varying MN as a convex optimization problem; in particular, we use joint regularization by both the ℓ_1 -norm for sparsity and the nuclear norm for rank reduction. We then introduce an algorithm for solving this problem based on the ADMM.

3.1 Previous kernel smoothing method

Previous studies assumed that temporally adjacent $\boldsymbol{\theta}^n$'s are similar to each other. Here, we extend the existing approach based on kernel smoothing [46, 58, 28]. According to this approach, parameters are estimated using locally weighted averages of the loss function based on kernel smoothing, which makes the estimates temporally smooth. In the context of the present study, the smoothed version of empirical loss, averaged over all instances, is given by ²

$$f_w(\boldsymbol{\Theta}) := \frac{1}{N} \sum_{n=1}^N \sum_{m=1}^N \varphi_w(m-n) l(\mathbf{y}^m, \boldsymbol{\theta}^n), \quad (6)$$

for any $\boldsymbol{\Theta}$. In Eq. (6), l is the loss function of Eq. (2), and φ_w is the smoothing kernel (i.e., a nonnegative symmetric function centered at zero [34]), depending on the kernel width w . Examples of the smoothing kernels are the box kernel,

$$\varphi_w(z) = \begin{cases} 1/(2w+1) & |z| \leq w \\ 0 & \text{otherwise} \end{cases}, \quad (7)$$

²Here, we assume that \mathbf{y}^n are sampled regularly in time. For non-regular sampling intervals, the smoothing kernel can be defined from the difference between the time stamps of two data points \mathbf{y}^m and \mathbf{y}^n , as in [28].

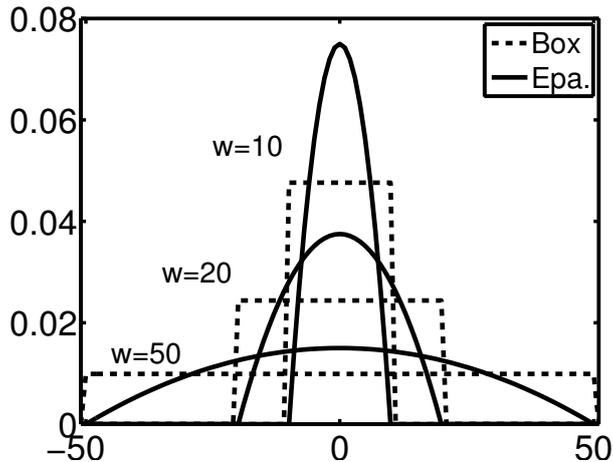


Figure 2: Smoothing kernels: Box (Eq. (7)) and Epanechnikov (Eq. (8)) with the kernel widths of $w = 10, 20,$ and 50 .

and the Epanechnikov kernel,

$$\varphi_w(z) = \begin{cases} 3(1 - (z/w)^2)/(4w) & |z| < w \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Figure 2 illustrates these smoothing kernels. The kernel width determines the smoothness and should be carefully selected, whereas the precise form of the kernel function is usually not very important [34].

3.2 Proposed sparse and low-rank estimation of time-varying MN

We now focus on situations where the parameter matrix Θ can be reasonably restricted to have relatively low rank according to any prior knowledge. This further reduces the effective degree of freedom associated with the large number of parameters in the time-varying MN, even when the temporal smoothness may not be very effective by its own.

Every instance θ^n of the parameter vectors is now implicitly represented in terms of K basis vectors (dependency patterns) \mathbf{a}^k and their coefficients s_k^n :

$$\theta^n = \sum_{k=1}^K s_k^n \mathbf{a}^k. \quad (9)$$

This also implies that the matrix Θ can be decomposed as $\Theta = \mathbf{A}\mathbf{S}$ with $\mathbf{A} = (\mathbf{a}^1, \mathbf{a}^2, \dots, \mathbf{a}^K)$ and $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_K)^\top$, as depicted in Fig. 1. We assume $K < \min\{C, N\}$ so that the matrix Θ is rank-deficient. Note that, in reality, true Θ might not be strictly rank-deficient but is instead approximately of low rank with the singular values approaching to zero relatively quickly; Eq. (9) then can be seen as an approximation of the true system.

The underlying idea is that networks in the real world often exhibit a limited number of communities or regimes, as already stated in Section 1. Intuitively, each \mathbf{a}^k in Eq. (9) may represent a specific dependency graph that corresponds to a community or a regime in the network. Equation (9) actually represents a generalized form of the regime switching because each network θ^n now can be any linear combination of the dependency patterns.

Here, we propose to extend the previous kernel smoothing method into that with the low-rank assumption. This could be achieved by minimizing the kernel-weighted empirical loss of

Eq. (6) with respect to \mathbf{A} and \mathbf{S} under the low-rank decomposition $\Theta = \mathbf{A}\mathbf{S}$ described above. However, joint minimization of the two decomposition terms is not a convex problem and thus cannot avoid the issue of local minima, which may limit its applicability in practice. We thus adopt an alternative approach which formulates the problem as a convex optimization directly on Θ without explicitly decomposing it.

The proposed sparse and low-rank estimation of Θ is now formulated as

$$\underset{\Theta}{\text{minimize}} \quad f_w(\Theta) + \|\mathbf{\Lambda} \odot \Theta\|_1 + \eta \|\Theta\|_*, \quad (10)$$

where the two regularization terms are introduced in the objective function, as well as the kernel-weighted empirical loss. Here, the ℓ_1 -norm $\|\cdot\|_1$ is of a long vector that concatenates the columns. The nuclear (or trace) norm [15, 47] $\|\cdot\|_*$ is defined as

$$\|\Theta\|_* = \sum_k \sigma_k(\Theta), \quad (11)$$

where $\sigma_k(\Theta) \geq 0$ is the k -th singular value of Θ . The nuclear norm therefore means the ℓ_1 -norm of the vector of singular values, and its minimization introduces sparsity on the singular values, encouraging the matrix to have a low rank [15, 47]. The two regularization coefficients (i.e., the non-negative matrix $\mathbf{\Lambda}$ and the positive scalar η) control the strengths of the ℓ_1 - and the nuclear-norm regularizations, respectively. For simplicity, we set $\mathbf{\Lambda} = (\boldsymbol{\lambda}, \boldsymbol{\lambda}, \dots, \boldsymbol{\lambda})$ with a common $\boldsymbol{\lambda}$ in every column. Note that the problem (10) is convex because all three terms in the objective function are convex [8].

In this formulation, the rank reduction by the nuclear norm introduces similarity between any columns θ^n or between any rows, as it is invariant under permutation of columns or rows. The kernel smoothing also introduces a similarity, but only between temporally-adjacent columns. The nuclear norm is thus expected to be particularly useful when some networks located at distant time points (relative to the specified kernel width) are potentially similar, or when the weights of some edges potentially have similar time courses. Note that the nuclear norm by itself never encourages any two networks θ^{n_1} and θ^{n_2} to differ. If the loss terms $l(\cdot, \theta^{n_1})$ and $l(\cdot, \theta^{n_2})$ actually promote differences in the networks, the nuclear norm oppositely encourages them to be similar, cooperatively with kernel smoothing.

3.3 Estimation algorithm by ADMM

The minimization problem (10) is convex and unconstrained, but the objective function is non-smooth due to the regularization terms. Hence, we cannot directly apply standard unconstrained optimization techniques. In this subsection, we derive a simple first-order algorithm to solve this problem (10) based on the alternating direction method of multipliers (ADMM) [4, 14, 7], which is a variant of the augmented Lagrangian (AL) method. The standard AL method has been effectively used to solve problems with either the ℓ_1 -norm regularization (e.g., in [50]) or the nuclear-norm regularization (e.g., in [51]³), but the joint use of the ℓ_1 and nuclear norms prevents an effective application of the standard AL. This motivated us to use the ADMM.

In the following sections, we first introduce the ADMM in a general form and then apply the ADMM to an equality-constrained problem that is equivalent to the problem considered herein (10).

³Note that these studies applied the standard AL to the Fenchel dual of the original ℓ_1 - or nuclear-regularized problems, rather than directly to the original problems, which is also possible.

3.3.1 General framework

Consider the following optimization problem that includes two convex functions ϕ and γ :

$$\begin{aligned} & \text{minimize} && \phi(\mathbf{x}) + \gamma(\mathbf{z}) \\ & \text{subject to} && \mathbf{J}\mathbf{x} = \mathbf{z} \end{aligned}, \quad (12)$$

with respect to real-valued vectors \mathbf{x} and \mathbf{z} , where \mathbf{J} is a matrix of appropriate size that represents linear constraints. To solve this problem, the ADMM iterates the following three steps from any initial conditions $\mathbf{x}^{(0)}, \mathbf{z}^{(0)}, \mathbf{r}^{(0)}$ until a given convergence criterion is satisfied:

$$\mathbf{x}^{(t)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \phi(\mathbf{x}) + \langle \mathbf{r}^{(t-1)}, \mathbf{J}\mathbf{x} \rangle + \frac{\alpha}{2} \|\mathbf{J}\mathbf{x} - \mathbf{z}^{(t-1)}\|_2^2 \right\}, \quad (13a)$$

$$\mathbf{z}^{(t)} = \underset{\mathbf{z}}{\operatorname{argmin}} \left\{ \gamma(\mathbf{z}) - \langle \mathbf{r}^{(t-1)}, \mathbf{z} \rangle + \frac{\alpha}{2} \|\mathbf{J}\mathbf{x}^{(t)} - \mathbf{z}\|_2^2 \right\}, \quad (13b)$$

$$\mathbf{r}^{(t)} = \mathbf{r}^{(t-1)} + \alpha(\mathbf{J}\mathbf{x}^{(t)} - \mathbf{z}^{(t)}), \quad (13c)$$

where $\langle \cdot, \cdot \rangle$ and $\|\cdot\|_2$ are the standard inner product and the ℓ_2 -norm, and $\alpha > 0$ is any positive constant. The superscript (t) denotes the iteration number. Each single iteration step of the ADMM can be seen as a single cycle in an alternating minimization of the augmented Lagrangian [4, 39]

$$\mathcal{L}_\alpha(\mathbf{x}, \mathbf{z}, \mathbf{r}) = \phi(\mathbf{x}) + \gamma(\mathbf{z}) + \langle \mathbf{r}, \mathbf{J}\mathbf{x} - \mathbf{z} \rangle + \frac{\alpha}{2} \|\mathbf{J}\mathbf{x} - \mathbf{z}\|_2^2, \quad (14)$$

with respect to primal vectors \mathbf{x} and \mathbf{z} , followed by the update of the dual vector \mathbf{r} . Note that in the standard AL method, the primal vectors \mathbf{x} and \mathbf{z} are simultaneously updated to jointly minimize \mathcal{L}_α with the dual vector \mathbf{r} fixed, instead of updating them by Eqs. (13a) and (13b) only once.

In the ADMM, the constant α can be chosen rather freely, as in the standard AL method. This is in contrast to the penalty method [39], in which the strength of the penalty should be sufficiently large, which can cause numerical instability.

3.3.2 Derivation for the problem of the present study

We apply the ADMM by reformulating the problem (10) in a similar manner as in [4, 16] as follows. This reformulation allows us to deal separately with the loss term and the two different regularization terms, leading to a simple iterative algorithm summarized in Algorithm 1.

We first introduce auxiliary variables $\mathbf{Z}_1, \mathbf{Z}_2$, and \mathbf{Z}_3 in $\mathbb{R}^{C \times N}$ and define

$$\gamma(\mathbf{Z}) = f_w(\mathbf{Z}_1) + \|\mathbf{\Lambda} \odot \mathbf{Z}_2\|_1 + \eta \|\mathbf{Z}_3\|_*, \quad (15)$$

with $\mathbf{Z} = (\mathbf{Z}_1^\top, \mathbf{Z}_2^\top, \mathbf{Z}_3^\top)^\top$. We can then rewrite the original problem (10) as

$$\begin{aligned} & \underset{\mathbf{X}, \mathbf{Z}}{\text{minimize}} && \phi(\mathbf{X}) + \gamma(\mathbf{Z}) \\ & \text{subject to} && \mathbf{J}\mathbf{X} = \mathbf{Z} \end{aligned}, \quad (16)$$

where $\phi(\mathbf{X}) \equiv 0$, $\mathbf{J} = (\mathbf{I}_C, \mathbf{I}_C, \mathbf{I}_C)^\top$ and \mathbf{I}_C is a $C \times C$ unit matrix. Note that the equality constraint implies that $\mathbf{X} = \mathbf{Z}_q$ for any q . The two problems (10) and (16) are equivalent in the sense that, given the optimal solutions Θ^* for (10) and $(\mathbf{X}^*, \mathbf{Z}^*)$ for (16), $\Theta^* = \mathbf{X}^* = \mathbf{Z}_q^*$ holds for any q .

Therefore, with the dual variables $\mathbf{M} = (\mathbf{M}_1^\top, \mathbf{M}_2^\top, \mathbf{M}_3^\top)^\top$, the ADMM procedure can be applied in a straightforward manner to this problem. The first step, Eq. (13a), is now a quadratic minimization and thus has a closed-form solution:

$$\mathbf{X}^{(t)} = \frac{1}{3} \left(\mathbf{Z}_1^{(t-1)} + \mathbf{Z}_2^{(t-1)} + \mathbf{Z}_3^{(t-1)} \right) - \frac{1}{3\alpha} \left(\mathbf{M}_1^{(t-1)} + \mathbf{M}_2^{(t-1)} + \mathbf{M}_3^{(t-1)} \right). \quad (17)$$

The second step, Eq. (13b), can be separately written as

$$\mathbf{Z}_1^{(t)} = \underset{\mathbf{Z}_1}{\operatorname{argmin}} \left\{ f_w(\mathbf{Z}_1) + \frac{\alpha}{2} \|\mathbf{Z}_1 - \mathbf{G}_1^{(t)}\|_{\mathbb{F}}^2 \right\}, \quad (18a)$$

$$\mathbf{Z}_2^{(t)} = \underset{\mathbf{Z}_2}{\operatorname{argmin}} \left\{ \|\mathbf{\Lambda} \odot \mathbf{Z}_2\|_1 + \frac{\alpha}{2} \|\mathbf{Z}_2 - \mathbf{G}_2^{(t)}\|_{\mathbb{F}}^2 \right\}, \quad (18b)$$

$$\mathbf{Z}_3^{(t)} = \underset{\mathbf{Z}_3}{\operatorname{argmin}} \left\{ \eta \|\mathbf{Z}_3\|_* + \frac{\alpha}{2} \|\mathbf{Z}_3 - \mathbf{G}_3^{(t)}\|_{\mathbb{F}}^2 \right\}, \quad (18c)$$

where $\mathbf{G}_q^{(t)} = \mathbf{X}^{(t)} + \alpha^{-1} \mathbf{M}_q^{(t-1)}$ ($q = 1, 2, 3$) and $\|\cdot\|_{\mathbb{F}}$ is the Frobenius norm. We can then obtain $\mathbf{Z}_1^{(t)}$ numerically using any standard unconstrained optimization technique, and $\mathbf{Z}_2^{(t)}$ and $\mathbf{Z}_3^{(t)}$ can be obtained as follows:

$$\mathbf{Z}_2^{(t)} = \operatorname{Soft} \left(\mathbf{X}^{(t)} + \alpha^{-1} \mathbf{M}_2^{(t-1)}, \mathbf{\Lambda}/\alpha \right), \quad (19a)$$

$$\mathbf{Z}_3^{(t)} = \operatorname{Svt} \left(\mathbf{X}^{(t)} + \alpha^{-1} \mathbf{M}_3^{(t-1)}, \eta/\alpha \right). \quad (19b)$$

Here, $\operatorname{Soft}(\cdot, \cdot)$ is an element-wise application of the soft-thresholding [19, 26, 12] operator, $\operatorname{soft}(a, b) = \operatorname{sign}(a) \max(|a| - b, 0)$, and $\operatorname{Svt}(\cdot, \cdot)$ denotes the singular value thresholding [9, 49] operator defined by

$$\operatorname{Svt}(\mathbf{A}, b) = \mathbf{U} \operatorname{diag}(\operatorname{Soft}(\boldsymbol{\sigma}, b)) \mathbf{V}^\top, \quad (20)$$

where $\mathbf{A} = \mathbf{U} \operatorname{diag}(\boldsymbol{\sigma}) \mathbf{V}^\top$ is the singular value decomposition of \mathbf{A} , and $\boldsymbol{\sigma}$ is the vector of singular values. Finally, the dual update step, Eq. (13c), is given separately for \mathbf{M}_q ($q = 1, 2, 3$) by $\mathbf{M}_q^{(t)} = \mathbf{M}_q^{(t-1)} + \alpha(\mathbf{X}^{(t)} - \mathbf{Z}_q^{(t)})$.

3.3.3 Stopping criterion and final estimates of Θ

The procedure described in Section 3.3.2 converges to the optimal solution under certain conditions on the accuracies of the two inner minimization problems denoted by Eqs. (13a) and (13b) [14]. Recent theoretical analysis on convergence behavior of ADMM can also be found, for example, in [21]. Although a detailed theoretical analysis of the proposed method is beyond the scope of our current study, the basic characteristics can be understood from these general results. In the simulation experiments in Section 5, there was no run that did not converge, which has empirically proved that the proposed method converges in ordinary situations.

A criterion for stopping an ADMM algorithm was presented in [7]: The algorithm terminates after the t -th iteration if the following two conditions are satisfied.

$$\delta_p^{(t)} := \|\mathbf{J}\mathbf{X}^{(t)} - \mathbf{Z}^{(t)}\|_{\mathbb{F}} \leq \epsilon_p^{(t)}, \quad \delta_d^{(t)} := \alpha \|\mathbf{J}^\top(\mathbf{Z}^{(t)} - \mathbf{Z}^{(t-1)})\|_{\mathbb{F}} \leq \epsilon_d^{(t)}. \quad (21)$$

Here, the tolerances $\epsilon_p^{(t)}$ and $\epsilon_d^{(t)}$ are practically chosen by

$$\epsilon_p^{(t)} = \sqrt{3CN} \epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\{\|\mathbf{J}\mathbf{X}^{(t)}\|_{\mathbb{F}}, \|\mathbf{Z}^{(t)}\|_{\mathbb{F}}\}, \quad (22a)$$

$$\epsilon_d^{(t)} = \sqrt{CN} \epsilon^{\text{abs}} + \epsilon^{\text{rel}} \|\mathbf{J}^\top \mathbf{M}^{(t)}\|_{\mathbb{F}}, \quad (22b)$$

Algorithm 1: The ADMM for the problem (16) (equivalent to (10))

Given: w (kernel width), λ (ℓ_1 -norm coefficient) and η (nuclear-norm coefficient)
 $\mathbf{X} \leftarrow \mathbf{0}$; $\mathbf{Z}_1 \leftarrow \mathbf{0}$; $\mathbf{Z}_2 \leftarrow \mathbf{0}$; $\mathbf{Z}_3 \leftarrow \mathbf{0}$; $\mathbf{M}_1 \leftarrow \mathbf{0}$; $\mathbf{M}_2 \leftarrow \mathbf{0}$; $\mathbf{M}_3 \leftarrow \mathbf{0}$;
while a specific convergence criterion is not satisfied, **do**

$$\begin{aligned} \mathbf{X} &\leftarrow \frac{1}{3} (\mathbf{Z}_1 + \mathbf{Z}_2 + \mathbf{Z}_3) - \frac{1}{3\alpha} (\mathbf{M}_1 + \mathbf{M}_2 + \mathbf{M}_3) \\ \mathbf{Z}_1 &\leftarrow \underset{\mathbf{Z}_1}{\operatorname{argmin}} \left\{ f_w(\mathbf{Z}_1) + \frac{\alpha}{2} \|\mathbf{Z}_1 - (\mathbf{X} + \alpha^{-1}\mathbf{M}_1)\|_{\mathbb{F}}^2 \right\} \\ \mathbf{Z}_2 &\leftarrow \operatorname{Soft}(\mathbf{X} + \alpha^{-1}\mathbf{M}_2, \mathbf{A}/\alpha) \\ \mathbf{Z}_3 &\leftarrow \operatorname{Svt}(\mathbf{X} + \alpha^{-1}\mathbf{M}_3, \eta/\alpha) \\ \mathbf{M}_q &\leftarrow \mathbf{M}_q + \alpha(\mathbf{X} - \mathbf{Z}_q) \quad \text{for } q = 1, 2, 3 \end{aligned}$$

end

where ϵ^{abs} and ϵ^{rel} are the absolute and relative tolerances, respectively, set at small positive numbers, e.g., $(\epsilon^{\text{abs}}, \epsilon^{\text{rel}}) = (10^{-5}, 10^{-4})$ [7]. In addition, slow convergence is avoided by adjusting α in each iteration, such that $\alpha^{(t+1)} = 2\alpha^{(t)}$, $0.5\alpha^{(t)}$ and $\alpha^{(t)}$, if $\delta_p^{(t)} > 10\delta_d^{(t)}$, $\delta_d^{(t)} > 10\delta_p^{(t)}$ and otherwise, respectively.

After the ADMM algorithm has converged, we obtain the primal solutions, $\widehat{\mathbf{X}}$, $\widehat{\mathbf{Z}}_1$, $\widehat{\mathbf{Z}}_2$, and $\widehat{\mathbf{Z}}_3$. Although in theory, these solutions should be equal to each other and also to Θ^* , in practice, these solutions are not exactly the same because we stop the algorithm when the equality constraints hold only approximately. In the experiments in Section 5, we define the final estimate $\widehat{\Theta}$ by $\widehat{\mathbf{X}}$, whereas each element of $\widehat{\Theta}$ is explicitly set at zero if the corresponding element of $\widehat{\mathbf{Z}}_2$ is zero. The estimated graph structure can finally be given by whether each element of $\widehat{\Theta}$ is zero or non-zero.

4 Related studies

The proposed method particularly assumes $\eta > 0$ in order to explicitly include the nuclear norm in the objective function. If we set $\eta = 0$, the optimization problem of Eq. (10) almost reduces to the one solved by KELLER [46, 28] (kernel-reweighted logistic regression); KELLER minimizes the same objective function but with a slightly relaxed constraint. To see this, suppose that $\lambda_i = 0$ and $\lambda_{ij} = \lambda$ for any $i < j$, as well as $\eta = 0$. It then follows from Eq. (2) and Eq. (6) that Eq. (10) has an equivalent optimization problem, given by

$$\begin{aligned} \text{minimize} \quad & \sum_{n=1}^N \sum_{i=1}^D \left\{ f_w^{in}(\boldsymbol{\theta}_i^n) + (\lambda/2) \sum_{j:j \neq i} |\theta_{ij}^n| \right\}, \\ \text{subject to} \quad & \theta_{ij}^n = \theta_{ji}^n \quad \text{for any } i < j \text{ and any } n. \end{aligned} \quad (23)$$

Here, the optimization variables include both θ_{ij} and θ_{ji} for any $i < j$, with the equality constraints explicitly introduced. The smoothed empirical loss f_w in Eq. (6) is decomposed into the ND terms, given by

$$f_w^{in}(\boldsymbol{\theta}_i^n) := \frac{1}{N} \sum_{m=1}^N \varphi_w(m-n) \{-\log p(y_i^m | \mathbf{y}_{\setminus i}^m; \boldsymbol{\theta}_i^n)\}, \quad (24)$$

where $\boldsymbol{\theta}_i^n := \{\theta_i^n\} \cup \{\theta_{ij}^n : j \neq i\}$ is the parameter subset on which the conditional probability actually depends. The problem (23) now can be seen as jointly solving the ND different ℓ_1 -regularized logistic regression problems based on the weighted loss, while every set of D problems

cannot be further decomposed due to the equality constraints. On the other hand, KELLER relaxes the equality constraints, so that the ND problems can be solved separately. However, it requires a post-processing step to resolve the inconsistency between $\hat{\theta}_{ij}$ and $\hat{\theta}_{ji}$.

A large body of literature describes regime switching or statistical change-point detection, with a motivation closely related to that of the present study. For example, Carvalho and West [10] studied Gaussian covariances in their dynamic matrix-variate graphical modeling framework. Yoshida et al. [53] estimated time-dependent gene networks from microarray data by dynamic linear models with Markov switching. The low-rank assumption on Θ actually generalizes the concept of regime switching, and this assumption is likely valid in change-point detection when the number of change-points are relatively small. Note that the method proposed here has an advantage in its computational efficiency due to the convexity of the optimization problem.

We introduced a new idea of using low-rank regularization in the specific context of the time-varying MNs. A closely-related idea has recently been presented in [2], which used a low-rank *decomposition* to improve the estimation of the time-varying Gaussian Graphical Model (GGM), the fundamental MN for real-valued data. This explicitly decomposed the GGM parameters into the fixed patterns and their time-varying coefficients, as in Eq. (9), and directly estimated both of them. It solved a non-convex optimization problem which would suffer from the issue of local optima; the convex nature of our method can be beneficial in practice, while a detailed comparison between these two approaches is left for a future study.

Joint ℓ_1 - and nuclear-norm regularization has been studied recently in e.g., [13, 41, 35, 57], independently from our preliminary work [22]. However, their target applications were quite different from ours: finding rank-one submatrices [13], multi-task learning [35], matrix completion [41], and estimating block or community structures in a single, *static* network [57]. Algorithms similar to our ADMM algorithm were developed in [41, 57]. In particular, Zhou et al. [57] applied the joint ℓ_1 - and nuclear-norm regularization, using a slightly different form of ADMM, to enforce a point-process network model (multi-dimensional Hawkes model) to have a static community structure. Although technically very similar, the underlying idea is quite different from ours. That is, they considered learning only a single network at a time, assuming that its (weighted) adjacency matrix is of low rank. In contrast, we here consider learning many instances of time-varying networks simultaneously, assuming that the entire parameter matrix Θ is of low rank but the adjacency matrix of each instance (obtained from the corresponding column θ^n appropriately) is not necessarily of low rank. The ideas are thus mutually orthogonal and they could even be combined in principle.

5 Simulation experiments

We conducted several simulation studies in order to validate the effect of the rank reduction by nuclear-norm regularization in estimating time-varying dependency graphs. In all these experiments, the regularization coefficient for the ℓ_1 -norm was set such that $\lambda_i = 0$ and $\lambda_{ij} = \lambda$ for all $i < j$ (see Section 2.3). The minimization in the first step of the ADMM, Eq. (18a), was numerically solved by a quasi-Newton method ⁴.

5.1 Simulation study I: Basic effects of rank reduction and kernel smoothing

First, we examined the effect of the nuclear-norm regularization in a simple simulation setting corresponding to Fig. 1. The dataset consisted of a seven-dimensional ($D = 7$) binary time-

⁴We used a Matlab implementation [43] of a limited-memory BFGS presented by Mark Schmidt, which is available at <http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html>

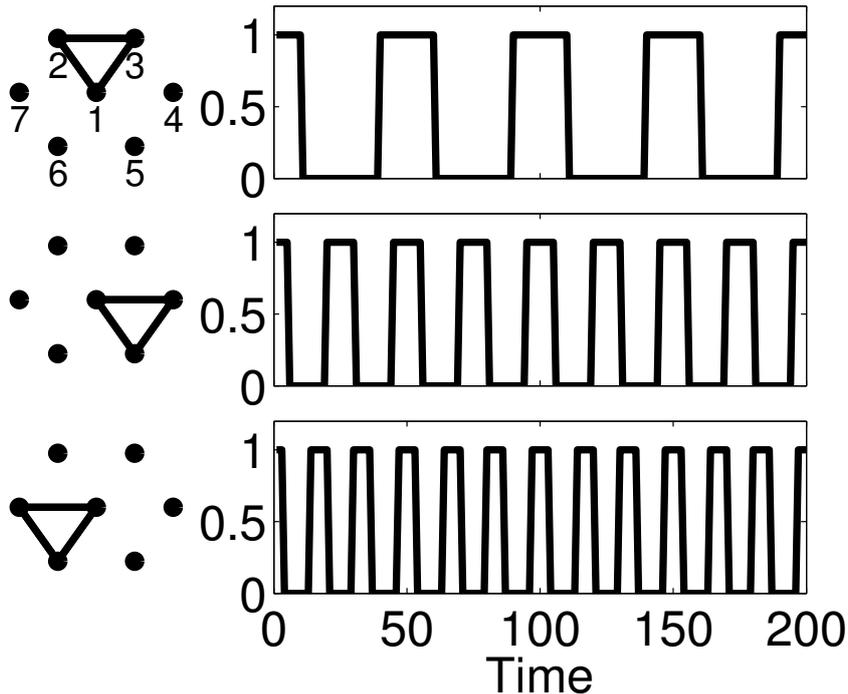


Figure 3: Simulation study I: Three graphs corresponding to the basis vectors (left) and the time-series of their coefficients (right).

series of length 200 ($N = 200$). We sampled $\mathbf{y}^n \in \{-1, 1\}$ at each time step from an Ising model $p(\mathbf{y}^n; \boldsymbol{\theta}^n)$ according to the joint probabilities directly calculated by Eq. (1); every $\boldsymbol{\theta}^n \in \mathbb{R}^{28}$ was created by

$$\boldsymbol{\theta}^n = s_1^n \mathbf{a}^1 + s_2^n \mathbf{a}^2 + s_3^n \mathbf{a}^3, \quad (25)$$

with three basis vectors $\{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$ and their coefficients $\{s_1^n, s_2^n, s_3^n\}$, so that every $\boldsymbol{\theta}^n$ was effectively constrained to be in a three-dimensional subspace, and the rank of Θ was at most three.

The dependency graphs corresponding to the three vectors \mathbf{a}^k ($k \in \{1, 2, 3\}$) are shown in Fig. 3. Specifically, for each k , we set an edge-wise parameter a_{ij}^k at a positive constant of 0.5 if the k -th graph has an edge between the nodes i and j , or set it at zero otherwise; we simply set all the node-wise parameters a_i^k ($i = 1, 2, \dots, D$) at zero. Figure 3 also shows the time-series of their coefficients, s_1^n , s_2^n , and s_3^n , which switched between 0 or 1 periodically with different cycle lengths.

We ran the proposed algorithm with every combination of the following values of the tuning parameters (w, λ, η) : $w \in \{2, 4, 6, 8\}$, λ from zero and positive values such that $\log_{10} \lambda$'s were regularly spaced by 0.05 in $[-4, -2]$, and η from positive values such that $\log_{10} \eta$'s were regularly spaced by 0.05 in $[-3, -1]$. We used the box kernel for kernel smoothing (see Fig. 2). We also ran a baseline algorithm with $\eta = 0$ for comparison, which basically solves a problem equivalent to that of KELLER [46, 28] as described in Section 4. In this experiment, the algorithm was terminated simply when $\max\{\delta_z^{(t)}, \delta_m^{(t)}\} \leq \epsilon$ ($= 10^{-5}$) was achieved, as in [55], where $\delta_z^{(t)}$ and $\delta_m^{(t)}$ are the maximum absolute values of all of the elements in $\mathbf{Z}^{(t)} - \mathbf{Z}^{(t-1)}$ and $\mathbf{M}^{(t)} - \mathbf{M}^{(t-1)}$, respectively. The constant α was initially set at 10^{-3} , but in order to accelerate the final convergence, it was multiplied by a factor of 1.5 at each iteration after reaching $\delta_m^{(t)} \leq 0.8\epsilon$.

The evaluation was performed in a similar manner to that of binary classification. In other

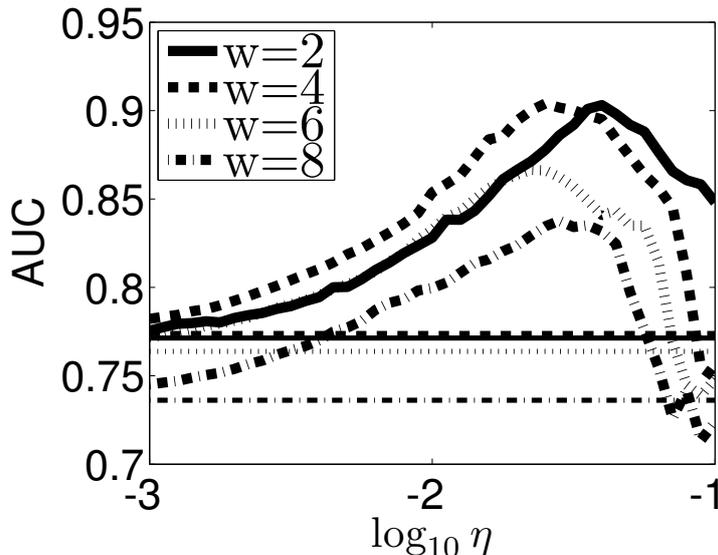


Figure 4: Simulation study I: The area under the ROC curve (AUC) versus $\log_{10} \eta$ (η , which is the regularization coefficient for the nuclear norm). For each w , the AUC value with $\eta = 0$ is depicted by a horizontal line.

words, as described in Section 3.3.3, each edge weight $\hat{\theta}_{ij}^n$ according to the final estimate $\hat{\Theta}$ was examined to determine whether the weight was nonzero or exactly zero, corresponding to edge existence or non-existence, respectively. Every such binary estimate was then compared with the truth for all $i < j$ and n . We quantified the performance by examining the area under the ROC curve (AUC) for every choice of w and η , as shown in Fig. 4. This figure clearly shows that for a wide range of η , the proposed method outperformed the previous method without the nuclear-norm regularization ($\eta = 0$).

Figure 5 also shows some examples of the ROC curves. For each w , the curves correspond to the best value of the AUC in Fig. 4 and the baseline algorithm of $\eta = 0$. The vertical and horizontal axes respectively denote the true positive and false positive rates (within all correct and incorrect outputs, respectively), where positive refers to a nonzero weight (i.e., the existence of an edge). Note that the ROC curve was plotted in a non-standard manner: rather than varying the threshold for classification (which is lacking in the setting of the present study), each single curve shows the variation of the classification performance according to the value of the regularization coefficient λ (i.e., the strength of sparsity). The point at the top-right corner corresponds to $\lambda = 0$, and the other points to positive λ 's whose values increase as the bottom left corner is approached.

These results showed that the rank reduction can improve the overall performance of edge detection in terms of the AUC measure. We will use the same scheme also in the following simulation studies (Sections 5.2, 5.3 and 5.4) to evaluate the average performance.

It should be noted that kernel smoothing may locally have a detrimental effect on edge detection especially when the smoothing is strong (i.e., with large w). To illustrate this, we examined a simple change-point problem in which the data were generated in the same manner as above except that the true dependency structures (i.e., true Θ^n) changed only at the 101-th time point. The two graphs before and after the change point are illustrated at the top of Fig. 6. Only the three edge weights changed from zero to nonzero (edges 1 and 3) or nonzero to zero (edge 2), with every nonzero weight θ_{ij}^n being set at 0.5. We set η as $\log_{10} \eta = -1.5$, and λ as $\log_{10} \lambda = -2.5$ in Fig. 6 (a) and $\log_{10} \lambda = -2.3$ in Fig. 6 (b). To clarify the effect of smoothing,

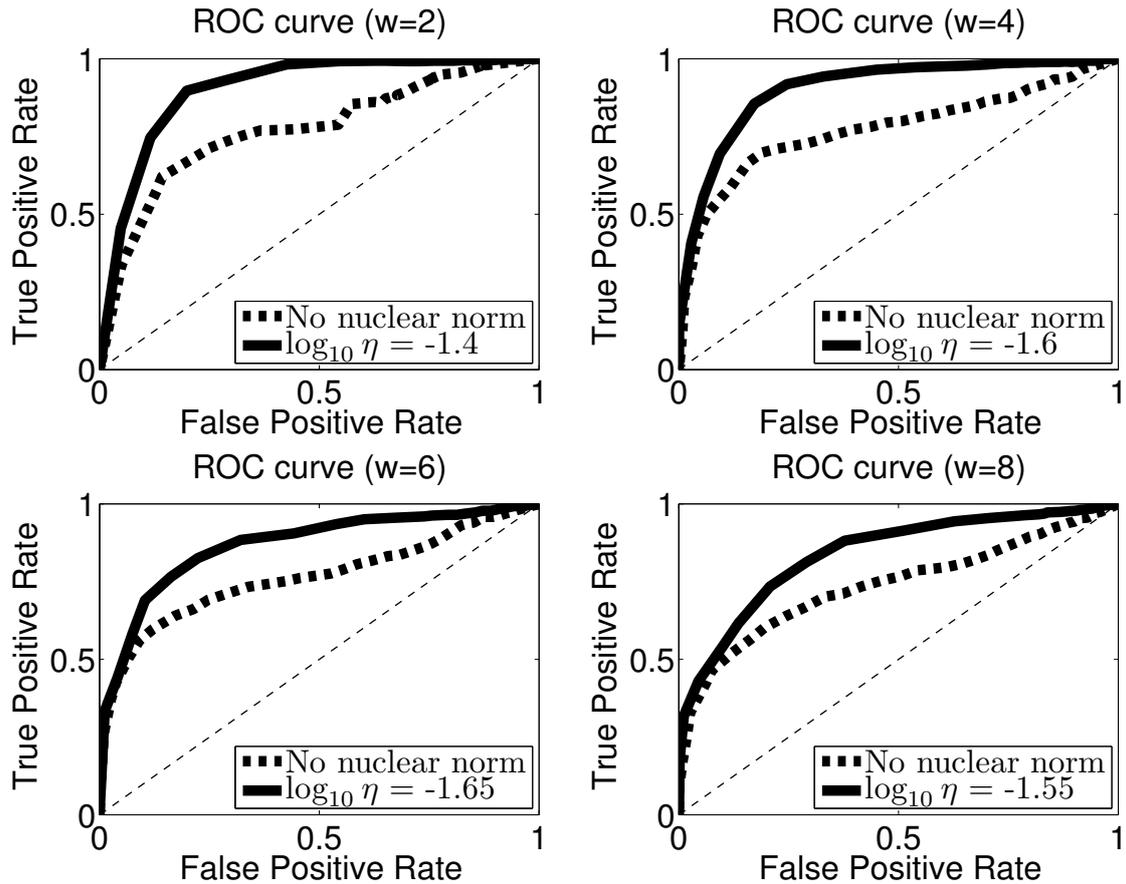


Figure 5: Simulation study I: Estimation of graph structure improved by nuclear-norm regularization. The ROC curves were plotted by varying the strength of sparsity (regularization coefficient of the ℓ_1 norm) λ , instead of varying the threshold for classification (which is lacking in the proposed method). Each panel corresponds to a specific choice of the kernel width w . The vertical and horizontal axes denote the true positive and false positive rates, respectively, and the two curves show the results without the nuclear norm ($\eta = 0$) and with a positive η (η : the regularization coefficient for the nuclear norm), where only the result with the best η value (in terms of AUC shown in Fig. 4) is shown for clarity.

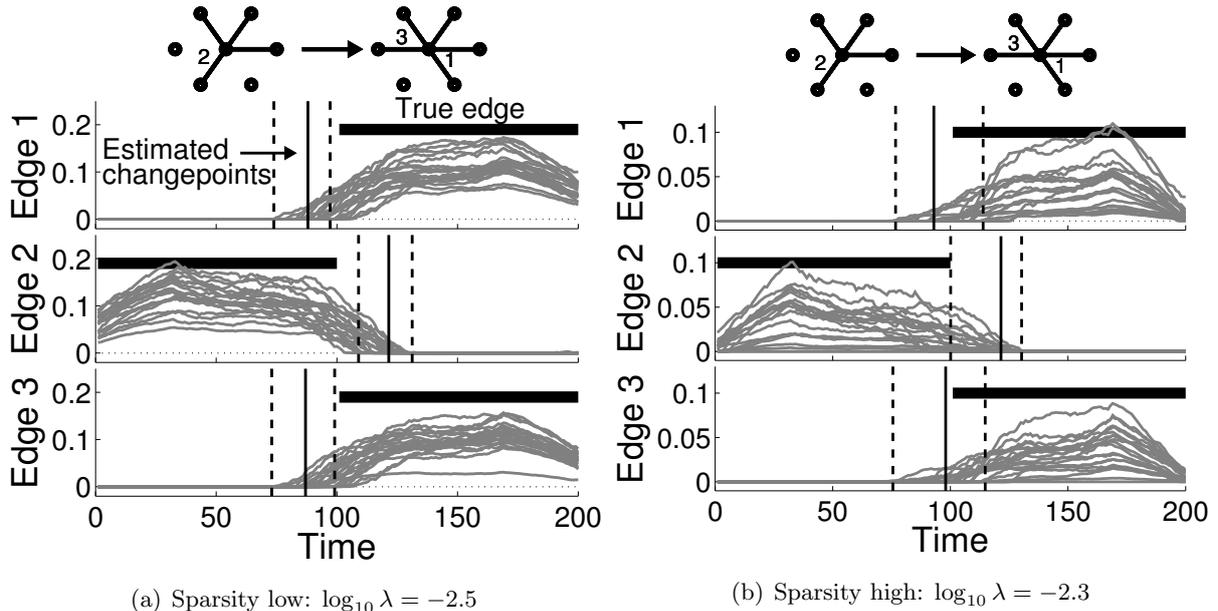


Figure 6: Simulation study I: Strong kernel smoothing ($w = 32$, box kernel) possibly biases the edge detection around steep change points (while improving it in stationary regimes), depending on the value of regularization coefficient: $\log_{10} \lambda = -2.5$ in (a) and -2.3 in (b), and $\log_{10} \eta = -1.5$ (both (a) and (b)). Top: True dependency graphs in the former and latter halves of the total duration $N = 200$, with positive edge weights fixed at 0.5. Bottom: Time series of the estimated edge weights (gray lines, for 20 randomly-generated datasets) corresponding to the three edges indicated in the graphs. Distribution of the estimated change points are indicated by vertical solid (median) and dashed (10- and 90-percentiles) lines. Black horizontal bars indicate the duration of the existence of true (nonzero) edges.

we set the kernel width relatively large, i.e., $w = 32$. In both figures, estimated time series of the three edge weights $\hat{\theta}_{ij}^n$ (for 20 randomly-generated datasets) are drawn by gray lines, and the median and two percentiles (of 10% and 90%) of the detected change points (from zero to nonzero in edges 1 and 3, and from nonzero to zero in edge 2) are indicated by the solid and dashed vertical lines, respectively. The black horizontal bars represent the time period during which the true edge existed.

As is obvious in Fig. 6 (a), the relatively strong kernel smoothing enabled the accurate identification of zero or nonzero within the stationary regimes, while it caused the estimated weights (gray lines) to be wrongly nonzero around the true change points, so that the timing of estimated change points (vertical solid/dashed lines) was biased. Strong kernel smoothing may thus lead to locally inaccurate edge detection around a step change point. As seen in Fig. 6 (b), a stronger sparsity (or possibly another threshold on weights) could compensate for such a biased estimation of change points by shrinking the estimated weights toward zero. However, the combined effect should depend on different choices of tuning parameters, and is not easily predictable.

However, such a detrimental effect of kernel smoothing would be minor on the whole due to the relatively small fraction of change points among all the time points, so that the AUC can be a valid measure for evaluating overall edge detection performance. If the objective is to know the precise timing of change points, results must be carefully interpreted. We discuss possible solution to this issue in Section 7.

5.2 Simulation study II: Smooth transition with dependency patterns

Next, we evaluated the performance of the estimation of graph structures with a variety of synthetic time-varying networks. The objective of this experiment is to clarify the situations where the rank reduction works efficiently.

Here, the number of variables was $D = 12$, and a total of $N = 360$ instances \mathbf{y}^n were sampled from time-varying Ising models $p(\mathbf{y}^n; \boldsymbol{\theta}^n)$, where $\boldsymbol{\theta}^n \in \mathbb{R}^{66}$ is the vector of edge-wise parameters at the n -th time step; all the node-wise parameters were set to zero for simplicity. Every parameter vector $\boldsymbol{\theta}^n$ was given by a linear combination of K basis vectors according to $\boldsymbol{\theta}^n = \sum_{k=1}^K s_k^n \mathbf{a}^k$. Each basis vector again defines a three-node clique (triangle) in the dependency graph, as in Section 5.1, where the K cliques were randomly chosen in each run without any overlapped edges between cliques; every nonzero element of \mathbf{a}^k was fixed at one.

We created more realistic time courses of the network than those in the previous section by generating the time-varying basis coefficients s_i^n in a similar scheme to that used in [58]. The network smoothly transitions between stationary regimes, and each dependency pattern appears non-periodically with non-constant strength.

Figure 7 illustrates this scheme. First, we divided the 360 time points into $360/\text{duration} + 1$ blocks, where `duration` means the length of the time period during which the structure of a dependency graph was maintained. The first and the last blocks had half of the length of the others. In the first block, a fixed number `num_edges/3` of active (nonzero) coefficients was randomly generated. As each clique has three edges, `num_edges` means the number of total edges in an instance graph. The nonzero values of these active coefficients were sampled uniformly in the range of $[2, 6]$. To generate the next block, we randomly set half (rounded-up if not an integer) of the active coefficients at zero, and activated the same number of other inactive coefficients with their values sampled in the same way as in the first block. This procedure was repeated until the final block was generated. Finally, the transitions between active and inactive coefficients were replaced by smooth sigmoidal curves.

Here, the values of `duration` and `num_edges` respectively controlled the level of stationarity and the level of sparsity of the generated time-varying networks. For each specification of these values, we randomly created ten different time-series of time-varying networks. A single binary time-series was generated from each of these by the corresponding Ising model. For each single time-series, we ran the proposed algorithm with various combinations of the tuning parameters (w, λ, η) with the Epanechnikov kernel (see Fig. 2) and the stopping criterion described in Section 3.3.3. We also ran the baseline algorithm with $\eta = 0$ for comparison.

Figure 8 shows how the AUC varied against η for different kernel widths w . Each panel corresponds to a particular combination of `duration` and `num_edges`. The number of underlying dependency patterns (maximum rank) K was also varied as $K = 7$ (a) and $K = 14$ (b). For the larger maximum rank of $K = 14$, the number of active coefficients (i.e., `num_edges/3`) was also set larger than that of $K = 7$. Note that when `num_edges` = 3, only a single coefficient s_i^n was nonzero at each time, resembling the ordinary situation of regime switching. In every panel, the AUC of the proposed method increased over that of the baseline method ($\log_{10} \eta \rightarrow -\infty$) as $\log_{10} \eta$ increased, until $\log_{10} \eta$ reached around -1.5 ; the increase was more obvious with smaller kernel widths w . This tendency is more clearly seen in those panels corresponding to smaller `num_edges` and smaller maximum rank (i.e., $K = 7$). Note that sparser Θ tends to have a lower rank. The rank reduction was thus more effective when the true rank was lower.

In this simulation, a particular dependency (regime) may repeatedly appear especially when the network changes more frequently. Figure 8 thus confirmed our statement in Section 3.2 that the rank reduction works efficiently, especially when temporally-distant networks (relative to the kernel width) can be similar to each other. This is most evident if we select the best kernel widths (among the four specific ones) for `duration` = 5, 15 and 60 as $w = 10, 30$ and

60, respectively, with which the highest AUC value was achieved. The increase in AUC then appeared to be large, moderate, and small when the `duration` was 5, 15, and 60, respectively. The more significant improvement of the AUC for a smaller `duration` clearly indicates that the proposed method effectively incorporated the similarity between temporally-distant networks which could not be (re-)used by the local kernel smoothing. The same thing is also seen when the kernel width w was set smaller than the best one (e.g., with $w = 3$ or 10) for the most cases of `duration` = 60.

The conjoint effect of the temporal smoothing and the rank reduction in our method naturally raises a question: How does the proposed method work solely with the rank reduction but without temporal smoothing? Figure 9 examines this issue. This again shows the AUC versus $\log_{10} \eta$ in the same setting as that of Fig. 10 (b) (i.e., $K = 14$), but in particular with $w = 1$ (i.e., the kernel smoothing was not effective). Clearly, in every panel, even the highest AUC was not comparable to the AUC obtained by the proposed or the baseline method with kernel smoothing (see Fig. 10 (b)). This implies that the temporal smoothing is essential for successful estimation of time-varying dependency graphs; the rank reduction then further improves its performance in the particular situations considered here.

5.3 Simulation study III: No explicit dependency patterns

To examine a limitation of the proposed method, we also studied how the proposed method works when the network does not *a priori* have a limited number of communities or regimes. Here, we directly generated time-varying edge weights θ_{ij}^n without explicitly using any dependency patterns \mathbf{a}^k . In other words, we replaced the three-node cliques \mathbf{a}^k by all the edges ($K = 66$) and generated their coefficients s_k^n in the same manner as in Section 5.2. Note that the meaning of the two simulation parameters `duration` and `num_edges` does not change at all.

Figure 10 shows the result in a similar manner to that in Figs. 8 and 12. Here, we examined the four levels of sparsity, `num_edges` = 3, 6, 9 and 12, and the three levels of stationarity, `duration` = 5, 15, and 60. In this figure, the increase in AUC by rank reduction disappeared in many cases. This is reasonable because the parameter matrix is now not necessarily of low rank. However, a slight increase in AUC over that of the baseline method in some kernel widths w is still seen especially in those panels of small `num_edges` (say 3 or 6), where the true rank could be approximately low. Most notably, the increase in AUC was rather large when `duration` was 60 with small kernel width $w = 3$ or 10 (Interestingly, the top-right panel of Fig. 10 is quite similar to that of Fig. 8). This was probably because the number of stationary regimes was limited *a posteriori* in these cases, as a consequence of the small number of transitions during the entire time-series, as shown intuitively by Fig. 11. The rank reduction was thus sometimes helpful to improve the estimation, even when the network does not *a priori* have a limited number of communities or regimes.

5.4 Simulation study IV: Small sample size (short time-series)

A difficult situation arises in practice because the total sample size, i.e., the length of the time-series, is smaller than the total number of parameters. In order to see how the proposed method works in this situation, we repeated the simulation study II (Section 5.2) with a reduced sample size of $N = 60$, which was smaller than the number of edge-wise parameters.

The result is shown in Fig. 12 in the same format as in Fig. 8. Note that `duration` = 60 now means that the transition of the network occurred only once during the whole period. Every plot exhibited larger variance as expected, while the increase in AUC with increasing $\log_{10} \eta$ is still evident in some cases. For example, in the upper three panels of `duration` = 5, the AUC clearly increased over that of the baseline method within an appropriate range of $\log_{10} \eta$, particularly

with kernel width of $w = 3$ or 10. Similarly, in the upper three panels of `duration = 60`, the AUC increased particularly with kernel width of $w = 10$ or 30 (but not of $w = 3$). Thus, even if the time-series length is quite short, the rank reduction can be useful, at least when the estimation variance is not too large, as in the bottom-most panels of Fig. 12. The choice of kernel width seems to be important for the method to be effective.

5.5 Computation time

Here, we compared the computation time between the ADMM algorithms and the regression-based KELLER [46, 28]. All the algorithms were implemented on Matlab 7.14 and run on a Linux computer with 4 cores, 3.30-GHz CPU and 126-GB RAM. We modified the original implementation of KELLER⁵ in order to use the same smoothing kernel and the same coefficient on the ℓ_1 -norm (according to Eqs. (23) and (24)). We measured the elapsed CPU time at every iteration of the ADMM algorithm on the ten datasets above of $N = 360$ (Section 5.2), with $K = 14$, `num_edges = 6` and `duration = 5`. Throughout this experiment and the other simulation studies described so far, we observed no runs that did not converge.

Figure 13 shows the values of the objective function versus the CPU time for three different λ 's. Note that the baseline ADMM ($\eta = 0$) and KELLER minimize the same objective function, while the low-rank ADMM includes the additional term $\eta \|\Theta\|_*$, with η being specifically set as 0.01. Each point for KELLER also indicates the objective value versus the total CPU time of a single run (i.e. to solve the ND logistic regression problems), obtained with varying termination conditions in the logistic regression. Note that the ADMM does not necessarily decrease the objective value monotonically (as in the case of $\log_{10} \lambda = -2.5$). The result shows that KELLER achieved almost the minimum objective value in about 20-50 seconds in any case, while our ADMM algorithms took more than 100 seconds for convergence. Thus, if the rank reduction is not necessary, the computational efficiency of KELLER is favorable in practice; whereas, if the rank reduction is desirable, the ADMM ($\eta > 0$) would be a better option.

In this experiment, the mean CPU time (with standard deviation) per iteration of the ADMM ($\eta = 0.01$) was 0.82 ± 0.13 [sec], which was slightly heavier than 0.78 ± 0.09 [sec] of ADMM ($\eta = 0$) due to the additional computation of SVD for non-zero η . On the other hand, Fig. 13 shows that the ADMM ($\eta = 0.01$) converged earlier than the ADMM ($\eta = 0$) did. This implies that the ADMM ($\eta = 0.01$) needed fewer iterations to converge than that by the ADMM ($\eta = 0$). Figure 13 also suggests that with larger λ , the ADMM ($\eta = 0$) tended to converge earlier.

6 Demonstration with a real-world network

Here, we demonstrate the effect of rank reduction when estimating time-varying MNs for a real-world social network: the network of one hundred senators during the 109th Congress of the U.S. Senate exhibited in their recorded votes during this congress⁶. This dataset has been analyzed in previous studies [3, 1, 28]. We followed [3] in particular for preparing the data.

In this dataset, each binary instance \mathbf{y}^n means the record of a single roll-call vote by the 100 senators ($D = 100$) for the n -th bill. Each vote by the i -th senator was recorded as $y_i^n = -1$ for ‘‘nay’’ and $y_i^n = 1$ for ‘‘yea’’ with missing votes simply treated as ‘‘nay’’ [3]. The total number of the bills, i.e., the length of the time-series, was $N = 645$. See [3] for more details about the preprocessing.

⁵Matlab code was available on <http://cogito-b.ml.cmu.edu/keller/downloads.html>

⁶The summary of roll-call votes is available from the website (<http://www.senate.gov/>)

6.1 Comparison of predictive performance

First, we evaluated the time-varying networks obtained by the proposed and the baseline methods in terms of their abilities for predicting unobserved data, instead of their performance in recovering graph structures. This was done because the true dependency graph between the 100 senators is not available. To this end, estimation (training) was performed in each run based only on the training dataset consisting of 516 (4/5 of the total) randomly selected instances, and a score of predictive performance was computed for the test dataset consisting of the other 129 instances, using the obtained estimate $\hat{\Theta}$. The predictability score was specifically given by the test log-pseudolikelihood, computed by summing-up the log-pseudolikelihood $\sum_i \log p(y_i^n | \mathbf{y}_{\setminus i}^n; \hat{\Theta}^n)$ over the test instances. This evaluates the performance of conditional prediction on the outcome of each node given those of the other nodes.

During training, the values of the smoothing kernel at the test instances were explicitly replaced with zeros in order to effectively exclude them from the training dataset. We used the Epanechnikov kernel (see Fig. 2), and employed the stopping criterion described in Section 3.3.3. For comparison, we also trained and tested the static sparse MN (Section 2.3) in the same manner as above.

Figure 14 shows the test log-pseudolikelihood versus $\log_{10} \lambda$ for some kernel widths w . Each panel shows the results for four specific values of η at a particular w , and also shows the results for the baseline method of $\eta = 0$. It also indicates the best performance achieved by the static MN, which was obtained at the best value of λ that maximized the test log-pseudolikelihood (Fig. 15). The rank reduction with appropriate combinations of η and λ greatly improved the prediction performance especially when the kernel width was relatively small. The performance by the proposed method was at a high level already at $w = 30$, especially with good choices of η and λ (say, $\log_{10} \eta = -1.75$ and $\log_{10} \lambda \in [-5, -4.5]$). The larger kernel widths then further improved the performance, at least until the width reached to $w = 90$. On the other hand, the performance of the baseline method at $w = 30$, with optimized λ , was even lower than that of the static network; this low performance was improved as the kernel width increased.

The social network considered here is greatly expected to have a community structure because of the two large political parties (i.e., Democratic and Republican), probably with several (transient) sub-communities within or across them. It is also reasonably expected that the network does not change very frequently, because it likely reflects the political positions of the senators. Thus, the result shown here is reasonable because in such a case, the estimation can be greatly improved by rank reduction, especially when the kernel width is relatively small, as actually seen in the simulation studies in Section 5.

6.2 Effect of rank reduction in estimated network

We obtained an intuitive understanding of how the rank reduction affects the quality of estimated network by again analyzing the whole dataset of the 645 instances by the proposed method with $\log_{10} \eta = -1.75$ and by the baseline method with $\eta = 0$, both with the kernel width of $w = 70$. The value of λ was commonly set at $\log_{10} \lambda = -4.5$, so that the two methods had a comparable level of predictability, as seen in Fig. 14.

Figure 16 shows the time courses of 50 randomly selected edge weights. The proposed method clearly reduced undesirable fluctuations of estimated weights θ_{ij}^n over time, which were unfavorably observed in the baseline method. This additional smoothing effect was particularly due to the rank reduction, as the two methods used the same kernel width: The underlying dependency patterns \mathbf{a}^k effectively divide different edges into groups and allow each group to show a consistent time course described by its common coefficient s_k^n .

Figure 16 also implies that most of the edge weights at each time point took non-zero values,

in particular in the result by the proposed method. Actually, we observed that maximizing the predictability score in this dataset tended to produce rather dense dependency graphs. This is not strange by itself, because the senators’ votes likely reflect some typical political stances, depending especially on the political party to which they belong, so that their network should not consist of sparse local connections seen in social networks of friendship.

6.3 Implication for subsequent analysis

In practice, determining useful knowledge from hundreds of large-scale dependency graphs is not easy. This is especially the case in the context of data mining, where we often do not have much prior knowledge or solid hypotheses about the data. Subsequent analysis of these dependency graphs will then be important for summarizing relevant information in order to disclose embedded knowledge.

Here, we demonstrate a simple case of this type of a post-analysis, and show that the improved quality of the dependency graphs obtained by the proposed method is actually beneficial at this stage. Specifically, we consider partitioning a graph into several clusters, each represented by an influential node called an *exemplar*. This is the problem considered in [17] for which an effective algorithm, the affinity propagation clustering (APC) [17], has been proposed. We applied this APC to every instance of the dependency graph in order to examine the changes in the exemplars and the clusters over time. The similarity matrix between any pair of nodes to which the APC was applied was defined as a matrix where both (i, j) and (j, i) entries are given by the estimated edge weight $\hat{\theta}_{ij}^n$. A nice property of APC is that it does not suffer from the issue of local optima, and it automatically determines the number of clusters by specifying a positive “preference” parameter [17]. This parameter was set at 0.01 based on a preliminary analysis.

For comparison, the APC was performed on both the estimates by the proposed and the baseline methods, denoted as $\hat{\Theta}_\eta$ and $\hat{\Theta}_0$, respectively. The kernel width was set at $w = 70$, and the regularization coefficients were specifically set at $\log_{10} \lambda = -5$ and $\log_{10} \eta = -1.75$ for the proposed method, and $\log_{10} \lambda = -4.5$ (and $\eta = 0$) for the baseline method, so as to roughly maximize the test log-pseudolikelihood in each method (Fig. 14). Note that the smaller predictability score of the baseline method indicates that the estimate $\hat{\Theta}_0$ was more likely overfitted to the training dataset.

Figure 17 shows the temporal changes of cluster members when one of the six nodes (senators) was chosen by APC as an exemplar. The six exemplars displayed here were the most frequently-chosen ones (in all the 645 time points) based on $\hat{\Theta}_\eta$. The cluster members displayed here for each exemplar were those belonging to the exemplar at least five times, which was also determined based on $\hat{\Theta}_\eta$. The same exemplars and cluster members are also displayed for the result based on $\hat{\Theta}_0$ for the sake of comparison. As seen in this figure, the overall patterns of the clustering in $\hat{\Theta}_\eta$ and $\hat{\Theta}_0$, indicated by “Proposed” and “No nuclear norm” at the top of each block, respectively, were quite similar to each other. However, the APC recovered transient clusters more stably in “Proposed” than in “No nuclear norm.” For example, the proposed method indicated that Senator Sununu as the exemplar had a cluster with Senator Gregg and Senator Kyl from the time points from 1 to about 300. However, this stable cluster is not clearly seen in “No nuclear norm” as it is fragmented into many discontinuous segments. Thus, the rank reduction in the estimation stage can improve the quality of the result of subsequent analysis, from which one may draw clearer and more specific speculation into human relationships.

7 Discussion

We proposed a method for learning time-varying MNs based on the assumption that the rank of parameter matrix is relatively low, in addition to the sparsity and temporal smoothness of the network structure. The problem was formulated as a non-smooth convex optimization problem where the objective function (i.e., kernel-weighted log-pseudolikelihood) was jointly regularized with both ℓ_1 and nuclear norms. We proposed to solve this problem by using the ADMM, which led to a simple first-order optimization algorithm. The main contributions can be summarized as follows: 1) we proposed a novel approach to learning time-varying MNs with a new type of application of the joint ℓ_1 - and nuclear-norm regularization, and 2) we empirically showed that our proposed method can outperform existing methods in terms of the estimation performance as well as the predictability, as recapitulated below. Although we did not give any theoretical performance guarantee, the empirical results successfully demonstrated the applicability of the proposed approach to various situations.

In the simulation studies, we examined the effect of rank reduction on estimation of time-varying dependency graphs, in cases where 1) the network exhibits a relatively small number of dependency patterns (Secs. 5.2 and 5.4), resembling community structures or regime switching, and 2) the network *a priori* has no limited number of dependency patterns (Section 5.3). In case 1, the rank reduction was particularly effective when the network changes frequently, so that temporally-distant networks may have similar dependency, which has not been dealt with very well by local kernel smoothing; the effect was rather limited if the time-series was very short, but was still evident in some situations with appropriate kernel widths. In case 2, the effect of rank reduction became weaker in most situations as expected, while it was surprisingly helpful when the number of dependency patterns was limited *a posteriori*; namely, when the stationary regimes in the entire time-series were relatively small due to the stability of the network. The rank reduction is therefore useful for dealing with datasets with a small number of change points, even when the network neither exhibits community structure nor repeats any previous regimes.

We also demonstrated the effect of rank reduction with a real-world dataset of US Senate voting records (Section 6). The rank reduction improved the predictability of the obtained network in terms of the test pseudolikelihood, and also led to more interpretable results in subsequent analysis of the collection of estimated dependency graphs. In particular, we demonstrated the use of the APC for clustering the nodes (senators).

As seen in Section 6, the test pseudolikelihood can be used for choosing η jointly with λ , which evaluates the predictability of the model and, in principle, can avoid overfitting to the training dataset. The kernel width w may also be automatically selected in a similar manner, while the choice should also reflect our prior knowledge about the time scale of the phenomena of interest. Thus, in practice, we recommend interactive selection of w from a relatively small set of candidates around the time scale of interest. A joint search for w , λ and η could be conducted by an heuristic greedy scheme for ease of computation. For example, w , λ , and η may be sequentially determined with λ and η initialized by zeros. Alternatively, the BIC-like information criterion developed in the previous study [28] could also be modified to be suitable for the method proposed here, which will further reduce the computational burden.

Our present study successfully demonstrated the feasibility and empirical performance of the joint ℓ_1 - and nuclear-norm regularization method for learning time-varying networks through extensive simulation studies as well as a real-world data analysis. To further clarify the applicability of our approach beyond the situations therein, we will need theoretical analyses of its statistical performance, but such analyses are beyond the scope of the current study. Yet, recent theoretical studies [41, 35] of joint ℓ_1 - and nuclear-norm regularization give insights into our approach. In these studies, error bounds on the matrix estimation were theoretically given

in specific settings of matrix recovery [41] and multi-task regressions [35] both with quadratic loss functions. As seen in Section 4, our loss function is closely related to a collection of many logistic regressions, similar to the multi-task setting in [35]. Hence, although kernel smoothing might need a special treatment, the approach in these studies could be extended to the problem considered here by incorporating more general loss functions [41]. These error bounds basically imply that true low-rank matrices can be correctly estimated when the true matrix is actually sparse and of low rank and the regularization coefficients are set large enough compared to the true cardinality (i.e., number of nonzeros) and rank. We expect that similar results will qualitatively hold for our problem; detailed theoretical analyses will clarify the general conditions for the proposed method to succeed or to fail. We leave this topic as an open issue for future study.

We finally discuss possible future extensions and other promising applications of our proposed framework. First, we focused on the fundamental application to the Ising model, but the proposed approach is also applicable to other time-varying MNs possibly with slight modifications. For example, the GGM is a real-valued counterpart of the Ising model, where each edge is given a scalar-valued (edge-wise) parameter defining the analog graph structure. The use of the ℓ_1 -norm, which induces element-wise sparsity, is then directly applicable. However, for general MNs, each edge may be associated with a vector-valued parameter, or higher-order cliques may even be involved. In these cases, the simple ℓ_1 -norm regularization can be replaced by other advanced techniques of sparse estimation, such as the group ℓ_1 [54, 36] or the hierarchical sparsity [56, 27, 44]. The ADMM is then slightly modified so that Eq. (18b) is replaced with an appropriate operation corresponding to the alternative regularization term.

Second, kernel smoothing may be replaced by other techniques if one needs to precisely know the timing of abrupt network changes (see also Section 5.1). Some previous studies of time-varying MNs [1, 28, 29] actually used a technique called total variation smoothing to allow the network parameters to be piecewise constants. It additionally introduces regularization terms $\sum_{n=2}^N \|\boldsymbol{\theta}^n - \boldsymbol{\theta}^{n-1}\|_1$ into the objective function with a non-smoothed version of empirical loss, $(1/N) \sum_{n=1}^N l(\mathbf{y}^n, \boldsymbol{\theta}^n)$. The ℓ_1 -norm of parameter differences allows each element of the parameter to have jumps at a few time points, and thus it may be more suitable for situations like regime switching. The additional low-rank regularization in our method then further exploits the potential similarity between temporally distant pairs of networks, as well as the similarity between pairs of the time-series of network parameters, which cannot be achieved by solely the total variation smoothing. Note that ADMM then can be applied by including an additional updating step (e.g., of \mathbf{Z}_4) for the additional regularization terms. A more conventional smoothing regularizer, $\sum_{n=2}^N \|\boldsymbol{\theta}^n - \boldsymbol{\theta}^{n-1}\|_2^2$, is also available without complicating the algorithm (since it is differentiable), but it may not be conceptually very different from kernel smoothing as it introduces a similar smoothness into the network parameters.

Third, an interesting extension of the proposed time-varying MNs would be to use of the same technique for auto-regressive models, relaxing the basic assumption in the Ising model that \mathbf{y}^n is independent of all past observations $\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^{n-1}$ given $\boldsymbol{\theta}^n$. For example, we could assume that \mathbf{y}^n depends on the past τ observations (i.e., $\mathbf{y}^n \sim p(\mathbf{y}^n | \mathbf{y}^{n-1}, \dots, \mathbf{y}^{n-\tau}; \boldsymbol{\theta}^n)$). The definition of $\boldsymbol{\theta}^n$ in the present study could then be simply modified to incorporate additional terms of the lagged effects, such as $y_i^n y_j^{n-m}$ ($1 \leq m \leq \tau$). This only changes the number of rows in Θ , and hence the ADMM algorithm will be directly applicable.

Finally, other than analyzing networks of social interactions as successfully demonstrated with the US Senate data, the proposed method can also be useful in the context of analyzing biological phenomena (e.g., gene-regulatory networks), as considered in the previous study [46]. Another promising application of the time-varying MNs would be the analysis of multi-channel measurements of the brain activity, such as neuronal spike trains [45] or other brain imaging modalities like functional magnetic resonance imaging (fMRI) or magnetoencephalography

(MEG). Interestingly, time-varying functional connectivity (dependency) between cortical regions has recently been a very active research target in the neuroscience and brain imaging communities [24, 33]. We will apply the method proposed here to these emerging new fields in our future studies.

Acknowledgements This research was supported by a contract entitled “Novel and innovative R&D making use of brain structures” with the Ministry of Internal Affairs and Communications, Japan, a contract entitled “Brain mapping by integrated neurotechnologies for disease studies” with the Japan Agency for Medical Research and Development, and JSPS KAKENHI Grant Numbers 24300114, 25730155. A.H. was supported by the Academy of Finland, Centre-of-Excellence in Inverse Problems Research.

References

- [1] A. Ahmed and E. P. Xing. Recovering time-varying networks of dependencies in social and biological studies. *Proceedings of the National Academy of Sciences of the United States of America*, 106(29):11878–11883, 2009.
- [2] P. Bachmann and D. Precup. Improved estimation in time varying models. In *Proceedings of the 29th International Conference on Machine Learning (ICML’12)*, 2012.
- [3] O. Banerjee, L. E. Ghaoui, and A. d’Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.
- [4] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: Numerical methods*. Prentice-Hall, Inc., 1989.
- [5] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.
- [6] J. Besag. Efficiency of pseudo-likelihood estimation for simple Gaussian fields. *Biometrika*, 64:616–618, 1977.
- [7] 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(11):1–122, 2011.
- [8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [9] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.
- [10] C. M. Carvalho and M. West. Dynamic matrix-variate graphical models. *Bayesian Analysis*, 2(1):69–97, 2007.
- [11] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20:33–61, 1998.
- [12] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.
- [13] X. V. Doan and S. Vavasis. Finding approximately rank-one submatrices with the nuclear norm and ℓ_1 -norm. *SIAM Journal on Optimization*, 23(4):2502–2540, 2013.

- [14] J. Eckstein and D. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 5:293–318, 1992.
- [15] M. Fazel, H. Hindi, and S. Boyd. Rank minimization and applications in system theory. In *Proceedings American Control Conference*, pages 3273–3278, 2004.
- [16] M. Figueiredo and J. Bioucas-Dias. Restoration of Poissonian images using alternating direction optimization. *IEEE Transactions on Image Processing*, 19(12):3133–3145, 2010.
- [17] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.
- [18] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostat*, 9(3):432–441, 2008.
- [19] W. J. Fu. Penalized regressions: the Bridge versus the Lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.
- [20] J. Guo, E. Levina, G. Michailidis, and J. Zhu. Joint structure estimation of markov networks. Submitted, 2010. Available from <http://www.stat.lsa.umich.edu/~guojian/>.
- [21] B. He and X. Yuan. On the $o(1/n)$ convergence rate of the douglasrachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.
- [22] J. Hirayama, A. Hyvärinen, and S. Ishii. Sparse and low-rank estimation of time-varying markov networks with alternating direction method of multipliers. In *International Conference on Neural Information Processing (ICONIP’10), Lecture Notes in Computer Science*, volume 6443, pages 371–379, 2010.
- [23] H. Höfling and R. Tibshirani. Estimation of sparse binary pairwise Markov networks using pseudo-likelihoods. *Journal of Machine Learning Research*, 10:883–906, 2009.
- [24] R. M. Hutchison et al. Dynamic functional connectivity: promise, issues, and interpretations. *NeuroImage*, 80:360–378, 2013.
- [25] A. Hyvärinen. Consistency of pseudolikelihood estimation of fully visible Boltzmann machines. *Neural Computation*, 18(10):2283–2292, 2006.
- [26] Aapo Hyvärinen. Sparse code shrinkage: Denoising of nongaussian data by maximum likelihood estimation. *Neural Computation*, 11(7):1739–1768, 1999.
- [27] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.
- [28] M. Kolar, L. Song, A. Ahmed, and E. P. Xing. Estimating time-varying networks. *Annals of Applied Statistics*, 4(1):94–123, 2010.
- [29] M. Kolar, L. Song, and E. P. Xing. Sparsistent learning of varying-coefficient models with structural changes. In *Proceedings of the 23rd Neural Information Processing Systems (NIPS’09)*, 2009.
- [30] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, MA, 2009.

- [31] S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.
- [32] S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2007.
- [33] N. Leonardi et al. Principal components of functional connectivity: A new approach to study dynamic brain connectivity during rest. *NeuroImage*, 83:937–950, 2013.
- [34] C. Loader. *Local Regression and Likelihood*. Springer, 1999.
- [35] S. Mei, B. Cao, and J. Sun. Encoding low-rank and sparse structures simultaneously in multi-task learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.
- [36] L. Meier, S. van de Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society B*, 70:53–71, 2008.
- [37] N. Meinshausen, P. Bühlmann, and E. Zürich. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34:1436–1462, 2006.
- [38] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582, 2006.
- [39] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer Verlag, 1999.
- [40] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.
- [41] E. Richard, P.-A. Savalle, and N. Vayatis. Estimation of simultaneously sparse and low rank matrices. In *Proceedings of the 29th International Conference on Machine Learning (ICML’12)*, 2012.
- [42] G. Rocha, P. Zhao, and B. Yu. A path following algorithm for sparse pseudo-likelihood inverse covariance estimation (SPLICE). Technical Report 759, Statistics Department, UC Berkeley, 2008.
- [43] M. Schmidt. minFunc: unconstrained differentiable multivariate optimization in Matlab. <http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html>, 2005.
- [44] M. Schmidt and K. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 709–716, 2010.
- [45] H. Shimazaki, S. Amari, E. N. Brown, and S. Grün. State-space analysis of time-varying higher-order spike correlation for multiple neural spike train data. *PLoS Comput Biol*, 8(3):e1002385, 2012.
- [46] L. Song, M. Kolar, and E. P. Xing. KELLER: estimating time-varying interactions between genes. *Bioinformatics*, 25(12):i128–i136, 2009.
- [47] N. Srebro, J. Rennie, and T. Jaakkola. Maximum margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, 2005.
- [48] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

- [49] K. C. Toh and S. W. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific J. Optimization*, 6:615–640, 2010.
- [50] R. Tomioka and M. Sugiyama. Dual augmented lagrangian method for efficient sparse reconstruction. *IEEE Signal Processing Letters*, 16(12):1067–1070, 2009.
- [51] R. Tomioka, T. Suzuki, M. Sugiyama, and H. Kashima. A fast augmented lagrangian algorithm for learning low-rank matrices. In *Proc. of the 27th Annual International Conference on Machine Learning (ICML 2010)*, 2010.
- [52] M. J. Wainwright, P. Ravikumar, and J. D. Lafferty. High-dimensional graphical model selection using ℓ_1 -regularized logistic regression. In *Advances in Neural Information Processing Systems*, volume 19, pages 1465–1472, Cambridge, MA., 2007. MIT Press.
- [53] R. Yoshida, S. Imoto, and T. Higuchi. Estimating time-dependent gene networks from time series microarray data by dynamic linear models with markov switching. In *Proc IEEE Comput Syst Bioinform Conf.*, pages 289–298, 2005.
- [54] 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.
- [55] X. M. Yuan. Alternating direction methods for covariance selection models. *Journal of Scientific Computing*, 51(2):261–273, 2012.
- [56] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Ann. Statist.*, 37(6A):3468–3497, 2009.
- [57] K. Zhou, H. Zha, and L. Song. Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS’13)*, pages 641–649, 2013.
- [58] S. Zhou, J. Lafferty, and L. Wasserman. Time varying undirected graphs. *Machine Learning*, 80(2–3):295–319, 2010.

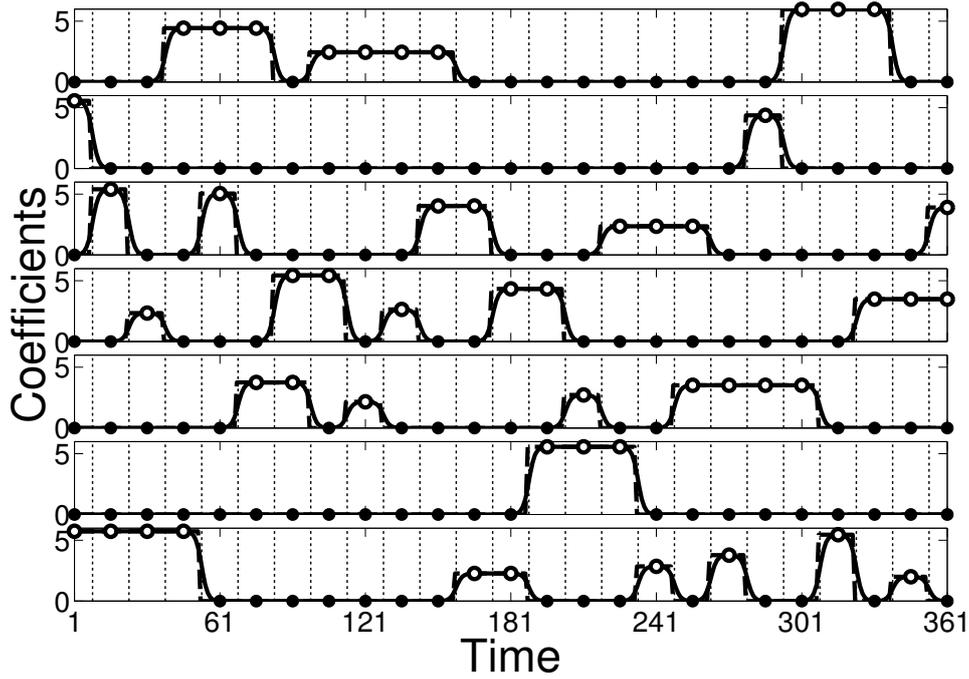
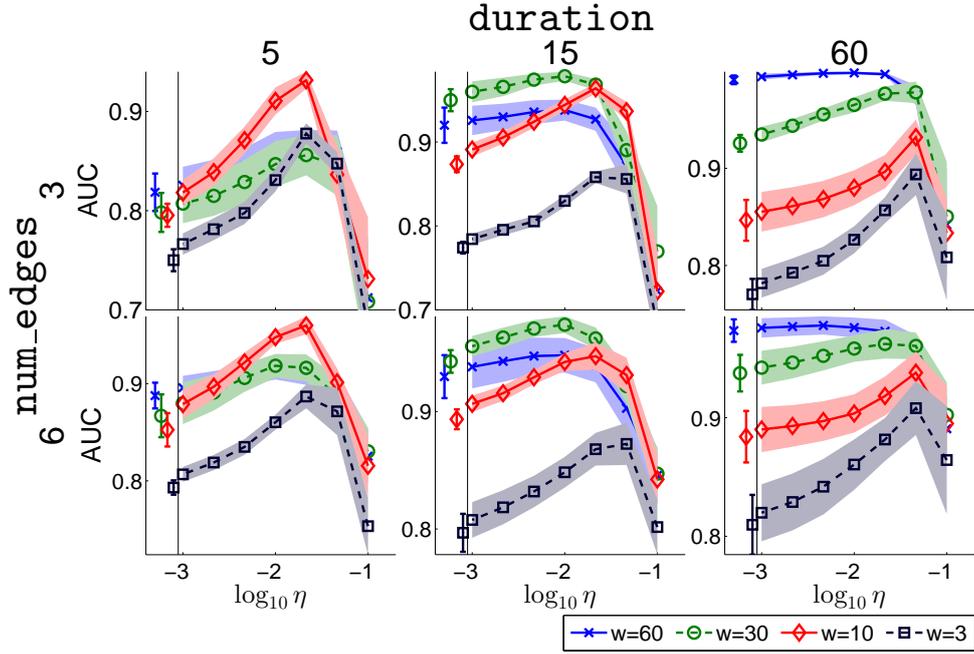
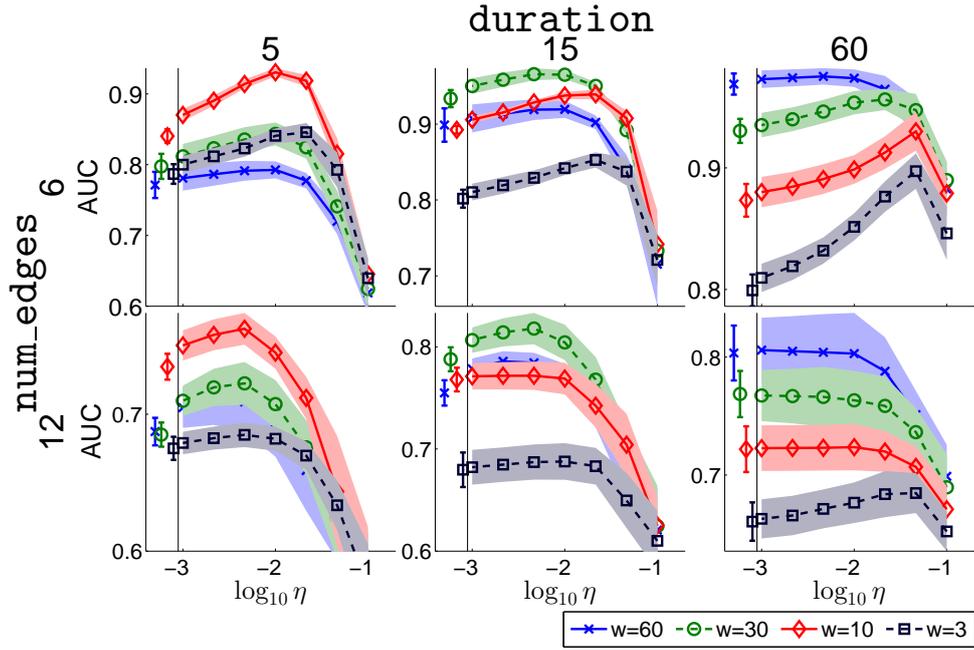


Figure 7: Simulation study II: Illustration of the scheme generating smoothly-varying basis coefficients s_i^n , from which the parameter vector is given by $\boldsymbol{\theta}^n = \sum_{k=1}^K s_k^n \mathbf{a}^k$ where \mathbf{a}^k is a randomly generated pattern vector corresponding to a 3-node clique (triangle) as in Fig. 3. This illustration corresponds to $K = 7$ (number of underlying patterns \mathbf{a}_k), `num_edges` = 6 (number of nonzero edges at each time point, i.e., pairs of coefficients s_i^n being nonzero at a time), `duration` = 15 (length of near-stationary blocks), and $N = 360$ (length of time series; the 361-th time point is included just for illustration). In each block segmented by vertical dotted lines, two selected coefficients are randomly given positive values which have been sampled uniformly in the range of $[2, 6]$ (white circle), while the rest are set to zero (black circle). The piecewise constant values (dashed line) are then smoothed by replacing the transition periods with sigmoidal curves (solid line). See the text for more details.



(a) $K = 7$



(b) $K = 14$

Figure 8: Simulation study II: AUC versus the regularization coefficient η of the nuclear norm. Each panel corresponds to a specific combination of `num_edges` (level of sparsity) and `duration` (level of stationarity) to generate the true network, indicated respectively on the left and on the top of the panels. The maximum rank of the true parameter matrix Θ was set at $K = 7$ (a) or $K = 14$ (b). In each panel, the isolated markers on the left of the vertical solid line show the average AUC over ten runs by the baseline method ($\eta = 0$) for different w 's, with the error bars indicating the standard deviation. On the right of the vertical solid line, the average AUC by the proposed method ($\eta > 0$) versus $\log_{10} \eta$ is plotted, with the standard deviation indicated by the shaded regions. The different markers correspond to different kernel widths shown in the legend at the bottom.

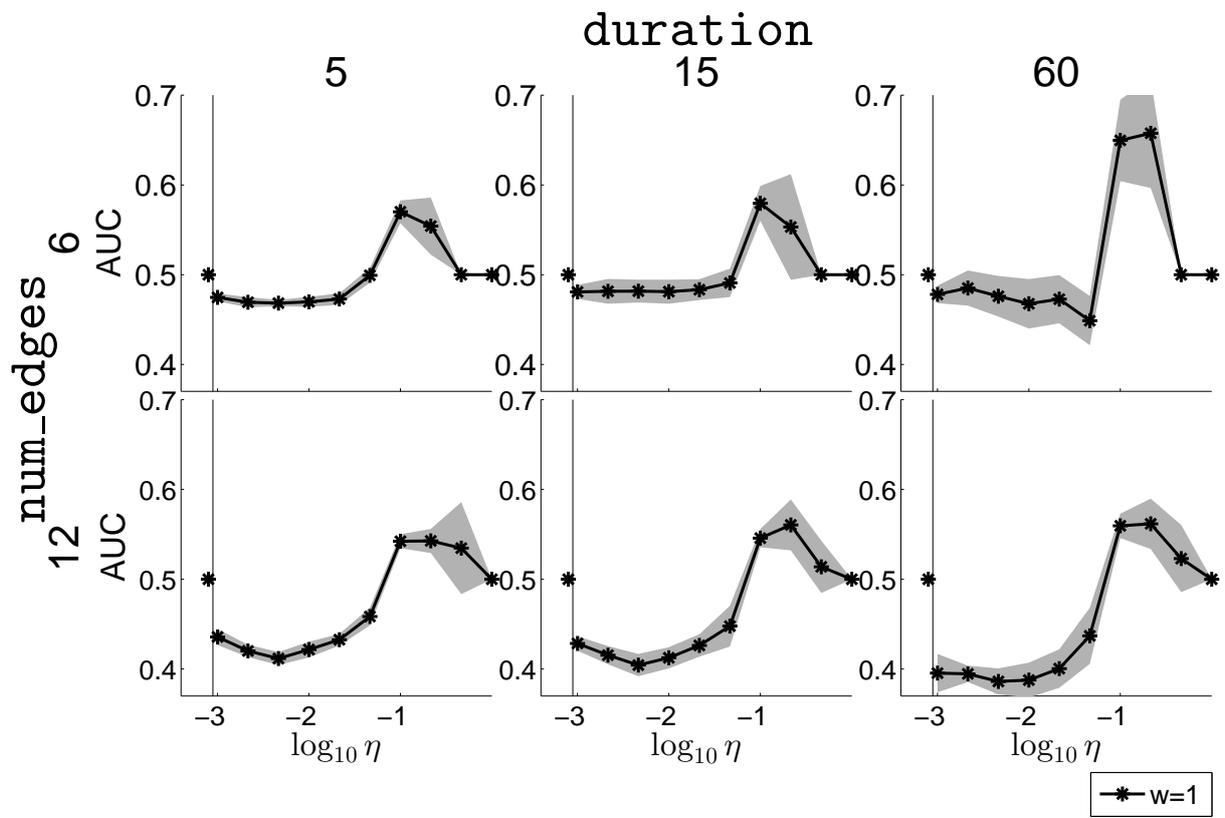


Figure 9: Simulation study II: AUC versus the regularization coefficient η of the nuclear norm, when the kernel smoothing was disabled, i.e., $w = 1$. The maximum rank was $K = 14$. See also the caption of Fig. 8.

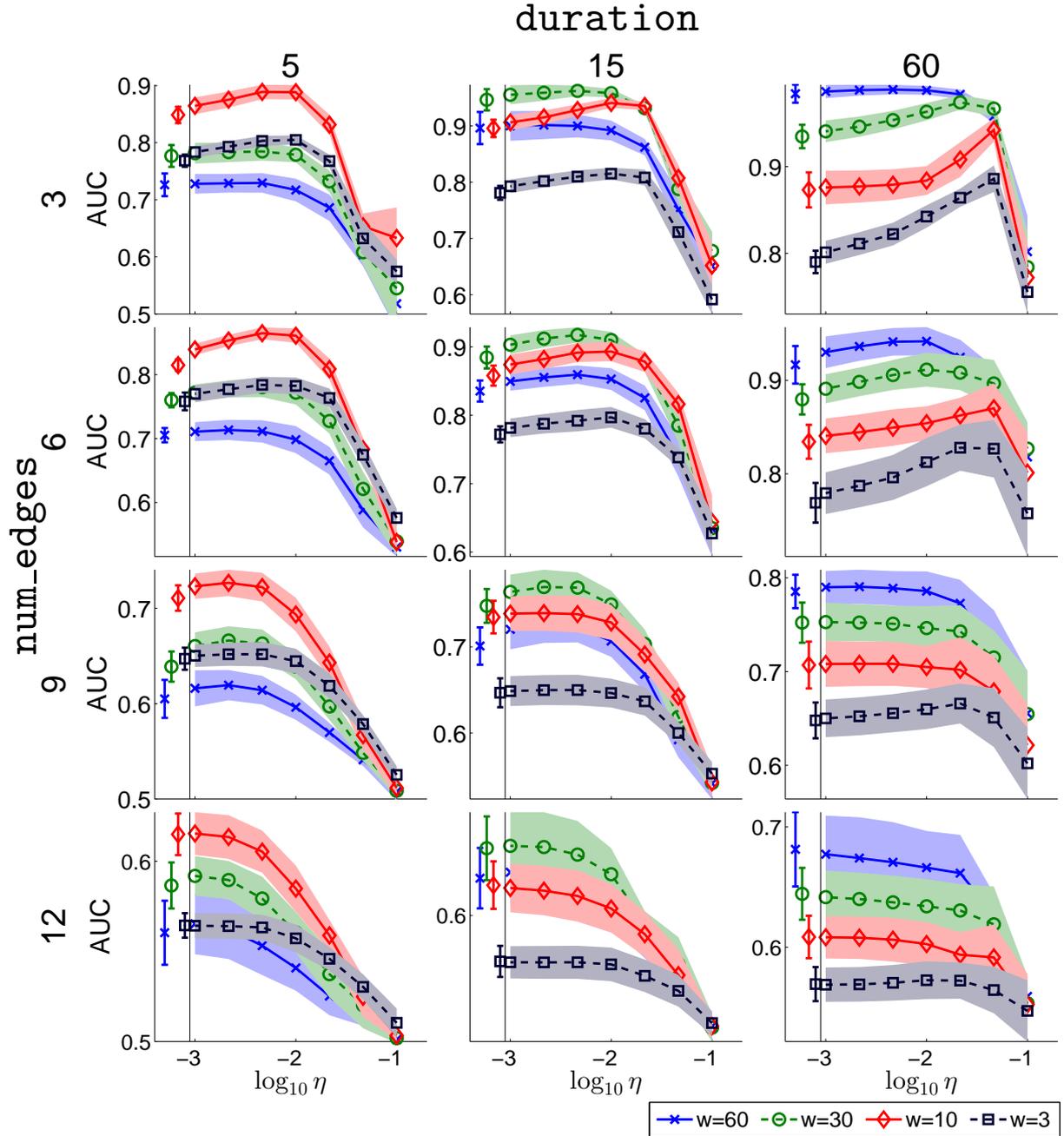


Figure 10: Simulation study III: AUC versus the regularization coefficient η of the nuclear norm. In this simulation, no explicit dependency patterns were used for generating data, with the weight of each edge separately generated by the scheme of Fig. 7. See also the caption of Fig. 8.

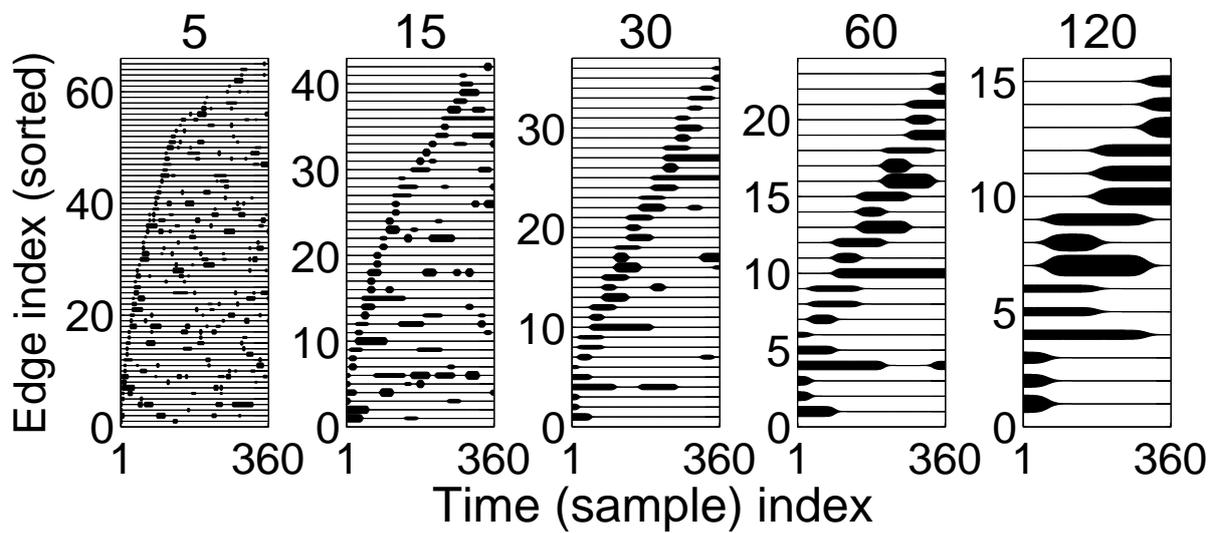
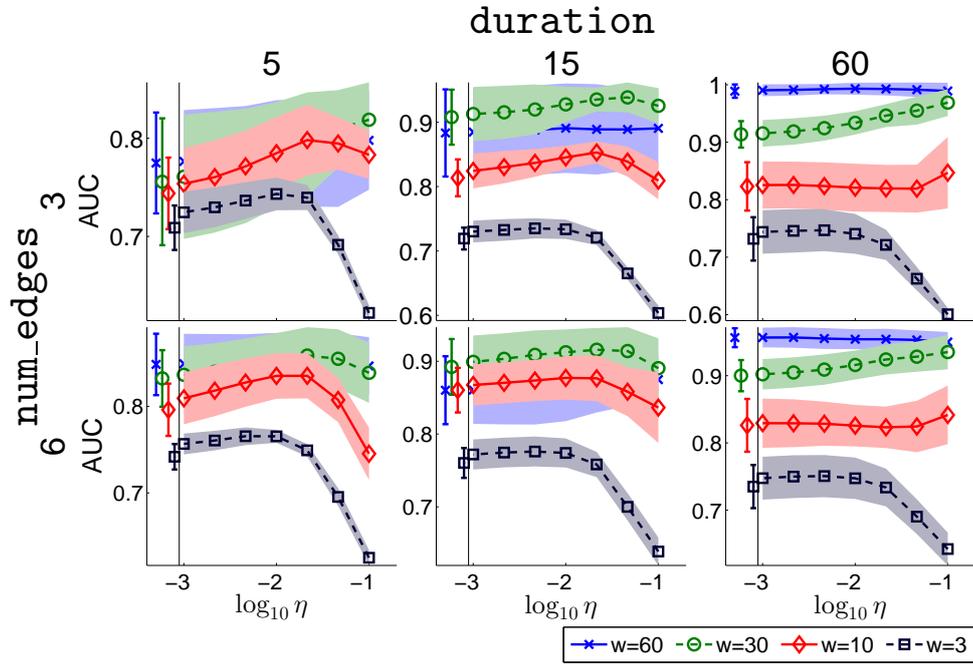
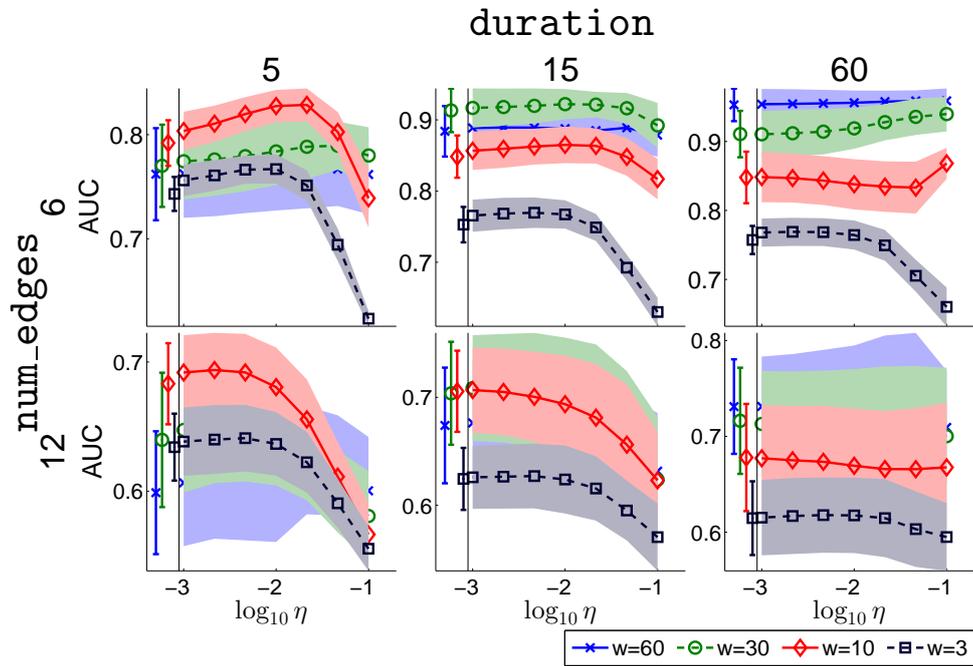


Figure 11: Simulation study III: Examples of time-varying edge weights in different levels of stationarity, **duration**, indicated by the number on the top of each panel. In each panel, the horizontal axis denotes the time steps and the vertical axis denotes the edge indices (sorted, so that an edge activated earlier with a nonzero weight had a smaller index). The varying thickness of each horizontal line indicates the relative magnitude of the corresponding edge weight at each time point. Only the edges whose weights became nonzero at least once are displayed.



(a) $K = 7$



(b) $K = 14$

Figure 12: Simulation study IV: AUC versus the regularization coefficient η of the nuclear norm, with a reduced sample size of $N = 60$. The simulation setting except for the sample size was the same as that of simulation study II. See also the caption of Fig. 8.

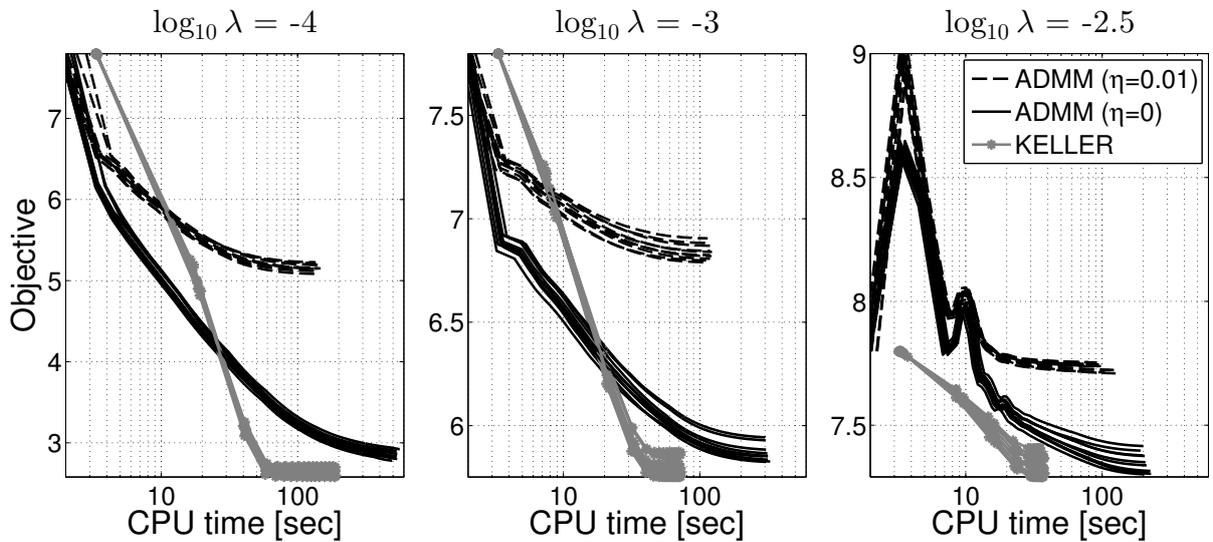


Figure 13: Comparison of computation time between the ADMM algorithms ($\eta = 0$ and 0.01) and the KELLER [46, 28] based on ℓ_1 -regularized (weighted) logistic regression. Solid ($\eta = 0$) and dashed ($\eta = 0.01$) lines without markers indicate the objective value versus the elapsed CPU time (log scale) at every iteration of the ADMM algorithm for ten different runs (for different datasets of $N = 360$, $K = 14$, `num_edges` = 6 and `duration` = 5; see Section 5.2 for details of the data). Note that the baseline ($\eta = 0$) and KELLER minimize the same objective function, while the low-rank ADMM includes the additional term $\eta \|\Theta\|_*$, with η being specifically set as 0.01. The three panels correspond to different settings of λ indicated on the top. The solid line with markers indicates the objective value versus the total CPU time of a single run of KELLER (i.e., to solve the ND logistic regression problems). Each line is plotted by varying termination conditions in the logistic regression on the same dataset.

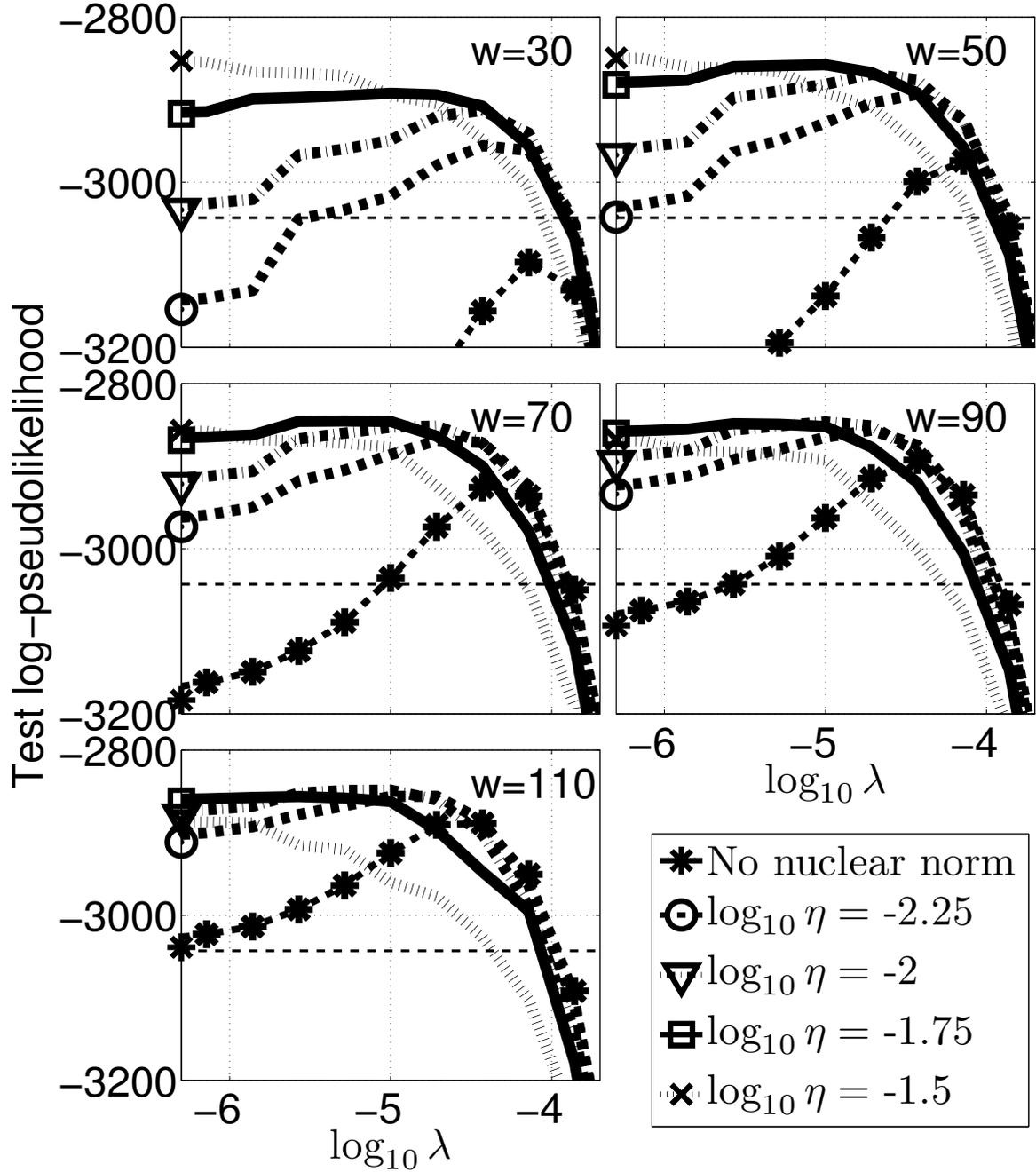


Figure 14: US Senate data: Test log-pseudolikelihood was improved by nuclear-norm regularization. Each panel shows the results obtained using the proposed method with four η 's (the four thick lines for which markers are only shown on the vertical axis, where η is the regularization coefficient of the nuclear norm) and that by the method without the nuclear-norm regularization (the dashed line with the marker '*'), for a specific kernel width w . In each panel, the vertical and horizontal axes denote the test log-pseudolikelihood and $\log_{10} \lambda$ (λ is the regularization coefficient of the ℓ_1 norm), respectively. The markers on the vertical axis indicate the results obtained with $\lambda = 0$, and the horizontal dashed line indicates the best result by the static MN, which is the maximum test log-pseudolikelihood shown in Fig. 15.

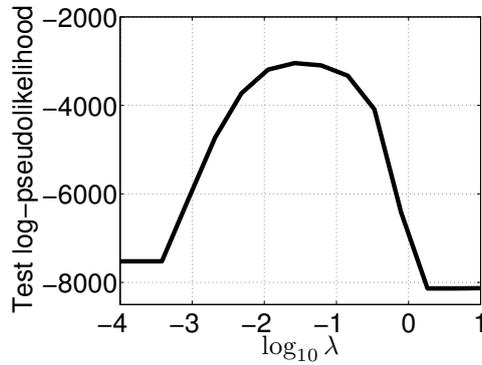


Figure 15: US Senate data: Test log-pseudolikelihood of static MNs in which the parameter vector θ was common for all the time steps. The horizontal axis denotes $\log_{10} \lambda$, where λ is the regularization coefficient for the ℓ_1 norm.

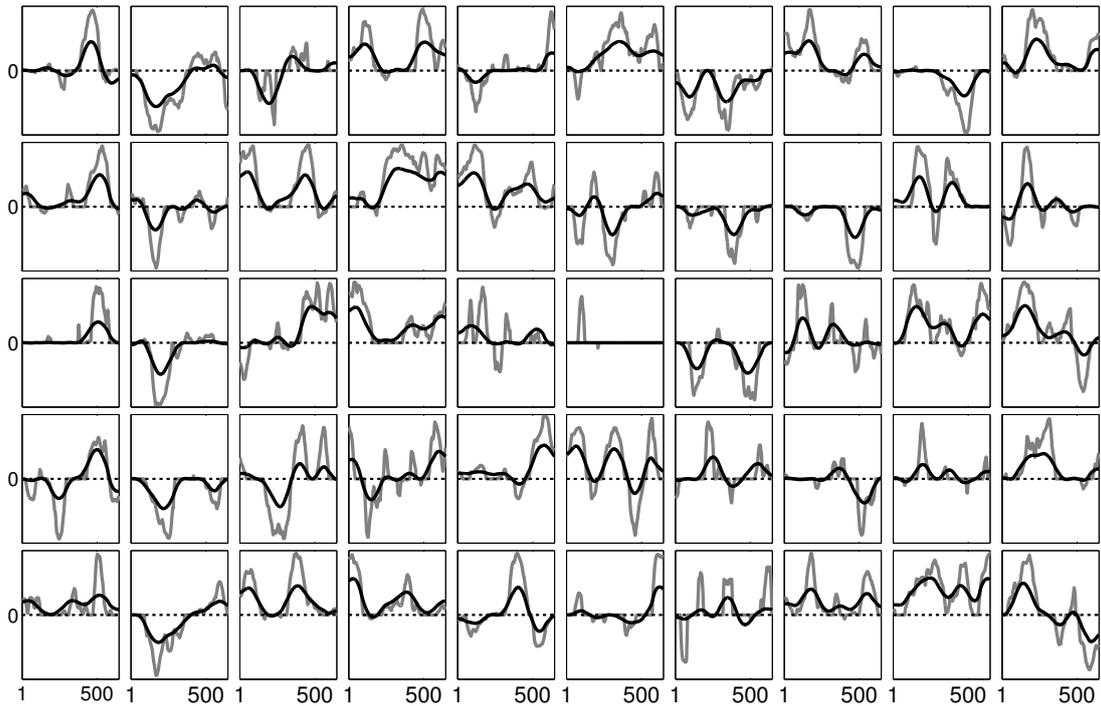


Figure 16: US Senate data: Time courses of 50 randomly-selected edge weights with (black line) and without (gray line) the nuclear-norm regularization. Vertical and horizontal axes in each panel show the edge weights (rescaled in each panel) and the time (from 1 to 645), respectively. The ordering of these panels (going downward from the top left) was determined so that similar time courses (solid line) are close to each other.

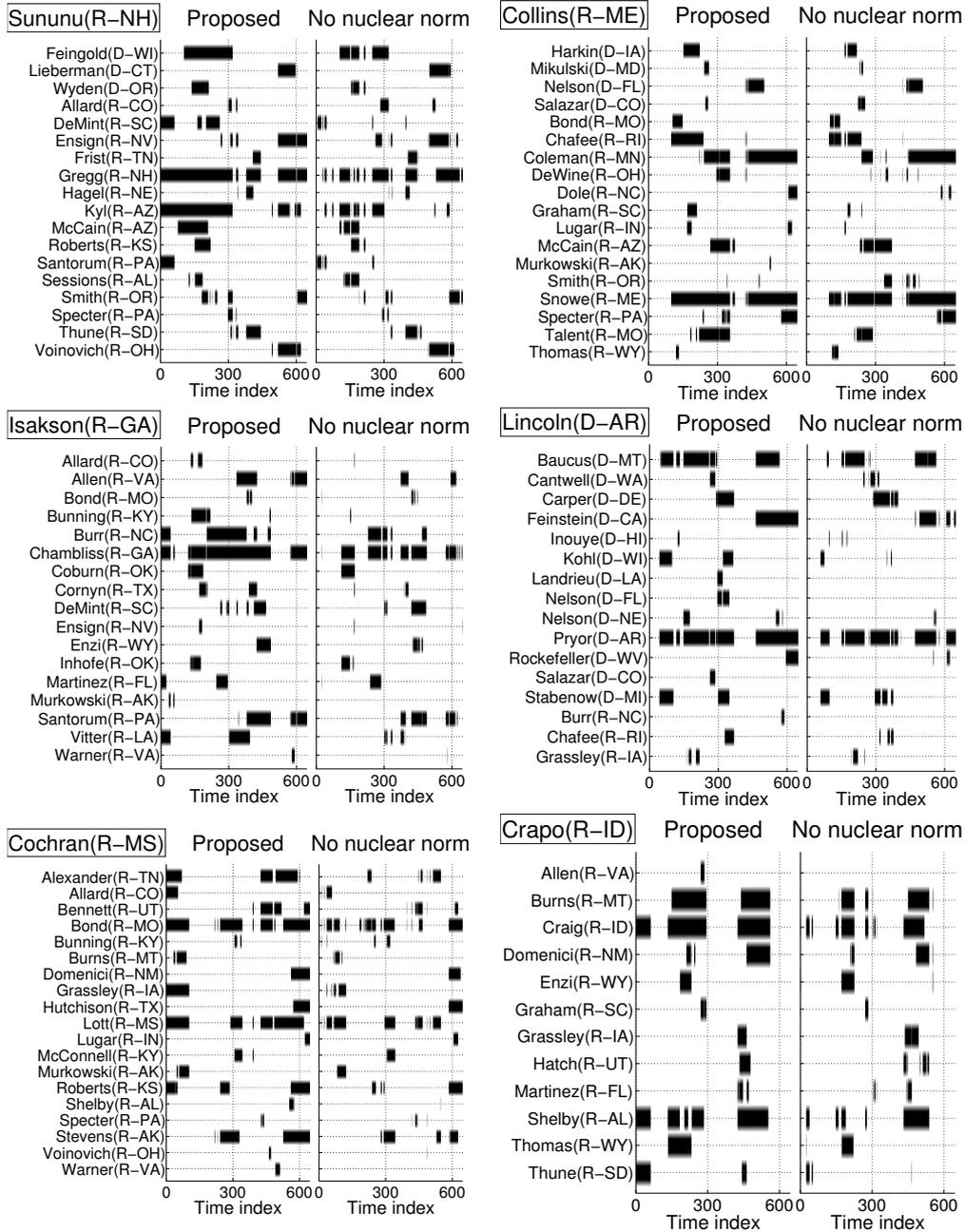


Figure 17: US Senate data: Temporal changes in clustering of nodes (senators) over the 645 time points found by the affinity propagation clustering (APC) [17]. The APC partitions every instance of the dependency graph into several clusters each represented by an influential node called an *exemplar*. Each of the six blocks shows the changes of the assignment of cluster members listed at the left side to a particular exemplar node indicated at the top left in a box. The annotation after the name of each senator refers to the senator’s political party (D: Democrat, R: Republican, I: Independent) and state (as a two-letter abbreviation). The “Proposed” and “No nuclear norm” indicate that the APC was applied to the estimate $\hat{\Theta}_\eta$ by the proposed method and on $\hat{\Theta}_0$ by the baseline method ($\eta = 0$), respectively, both at $w = 70$; the regularization coefficients η and λ were chosen so as to roughly maximize the test log-pseudolikelihood of Fig. 15 (see text). Each black horizontal bar indicates a time period during which the cluster member was actually assigned to the exemplar by the APC.