Efficient Covariance Estimation from Temporal Data

Hrayer Harutyunyan  
USC Information Sciences Institute  
Marina Del Rey, California  
hrayrh@isi.edu

Daniel Moyer  
USC Information Sciences Institute  
Marina Del Rey, California  
moyerd@usc.edu

Hrant Khachatrian  
YerevaNN  
Yerevan, Armenia  
hrant@yerevann.com

Greg Ver Steeg  
USC Information Sciences Institute  
Marina Del Rey, California  
gregv@isi.edu

Aram Galstyan  
USC Information Sciences Institute  
Marina Del Rey, California  
galstyan@isi.edu

ABSTRACT

Estimating the covariance structure of multivariate time series is a fundamental problem with a wide-range of real-world applications—from financial modeling to fMRI analysis. Despite significant recent advances, current state-of-the-art methods are still severely limited in terms of scalability, and do not work well in high-dimensional undersampled regimes. In this work we propose a novel method called Temporal Correlation Explanation, or T-CorEx, that (a) has linear time and memory complexity with respect to the number of variables, and can scale to very large temporal datasets that are not tractable with existing methods; (b) gives state-of-the-art results in highly undersampled regimes on both synthetic and real-world datasets; and (c) makes minimal assumptions about the character of the dynamics of the system. T-CorEx optimizes an information-theoretic objective function to learn a latent factor graphical model for each time period and employs two regularization techniques to induce temporal consistency of estimates. We perform extensive evaluation of T-CorEx using both synthetic and real-world data and demonstrate that it can be used for detecting sudden changes in the underlying covariance matrix, capturing transient correlations and analyzing extremely high-dimensional complex multivariate time series such as high-resolution fMRI data.

1 INTRODUCTION

Many complex systems in finance, biology, and social sciences can be modeled by time series data. One of the first steps in analyzing such time-varying complex systems is temporal covariance estimation—that is, estimation of the covariance matrix of the variables in different time periods. Such an analysis can be helpful for understanding relationships between components of the system, spotting trends, making predictions, detecting shifts in the underlying structure and for other tasks [1, 10, 11, 24].

There is an increasing need for efficient temporal covariance estimation methods that can work in high-dimensional undersampled regimes. In this regime, even static covariance estimation is a formidable problem [12]. Extending covariance estimation to the temporal case adds unique challenges. First, the samples from different time steps generally are not independent and identically distributed. Second, the dynamics of the system can be quite complex (e.g., financial time series, biological systems) and hard to model without strong assumptions. Furthermore, the number of time steps whose observations are relevant for estimating the covariance matrix for a particular time period is limited when the underlying covariance matrix changes quickly.

Current state-of-the-art temporal covariance estimation methods learn a sparse graphical model for each time period using variants of graphical lasso, while adding a regularization term for inducing temporal consistency of estimates [14, 31]. Unfortunately, these methods have at least cubic time complexity and quadratic memory complexity with respect to the number of variables, and do not scale to truly high-dimensional problems (e.g., the approaches described in Refs. [14, 31] do not scale beyond thousands of time series).

In this work we propose a novel temporal covariance estimation method, T-CorEx, that addresses the aforementioned challenges. T-CorEx is based on linear CorEx [29], which learns static latent variable graphical models by minimizing an information-theoretic objective function. Our method trains a linear CorEx for each time period and employs two regularization techniques to enforce temporal consistency of learned models. T-CorEx has linear time and memory complexity with respect to the number of variables and can be applied to temporal data with several orders of magnitude more variables than what existing methods can handle (e.g., it takes less than an hour on a moderate PC to estimate the covariance structure for time series with $10^7$ variables). The only assumption T-CorEx makes about the dynamics of the system is that on average, the underlying covariance matrix changes slowly with time.

We compare T-CorEx against other methods on both synthetic and real-world datasets. Our experiments show that T-CorEx yields state-of-the-art performance in highly undersampled regimes. More specifically, T-CorEx outperforms other methods in terms of describing existing correlations in the data (quantified by log-likelihood). We also apply T-CorEx to stock market and high-resolution functional magnetic resonance image (fMRI) data. For the former, we demonstrate that T-CorEx can detect transient correlations and sudden change-points of the underlying covariance matrix, not detectable with static methods. For the latter, we show that T-CorEx successfully scales for time series with 150K variables, and finds meaningful functional connectivity patterns.

We summarize our main contributions in this paper as follows:

- We introduce a new temporal covariance estimation method that has linear time and memory complexity with respect to the number of variables—which allows us to analyze systems that are several orders of magnitude larger than what was possible with previous methods.
To derive our method, we formulate the temporal covariance estimation problem for a sequence of multivariate Gaussian observations. We are given a sequence of observations \( \{D_t\}_{t=1}^T \), where \( D_t \in \mathbb{R}^{1 \times p} \) is a collection of \( s_t \) i.i.d. samples generated from a \( p \)-dimensional normal distribution with zero mean and covariance matrix \( \Sigma_t \in \mathbb{R}^{p \times p} \). The goal is to estimate the unknown \( \Sigma_t \) covariance matrices for each \( t \). Note that \( s_t \) can be 0 and can be different for different time steps. In particular, when \( s_t = 1, t = 1, 2, \ldots, T \), then \( \{D_t\}_{t=1}^T \) is a regular multivariate time series.

Without further assumptions, the problem is equivalent to having \( T \) independent static covariance estimation tasks. Usually covariance matrices of different time steps are related, and efficient temporal covariance estimation methods should exploit that relation. However, the relation can be quite complicated, making it hard to model without strong assumptions. To avoid making such assumptions, we only assume that on average, covariance matrices of neighboring time steps are close to each other with respect to some metric. Fig. 1 shows an example of a regularly sampled multivariate time series where the underlying covariance matrix exhibits different dynamics in different periods, but our assumption is met.

### 3 METHODS

Any static covariance estimation method \( M \) can be used for temporal covariance estimation by applying it in sliding window fashion. Namely, we divide the period \([1..T]\) into periods of size \( w \) and apply \( M \) on each period.\(^1\) There is no universal choice for \( w \), and different methods work best with different values of \( w \). Methods that work well with small values of \( w \) are usually better choices for temporal covariance estimation. However, these static estimators do not exploit the temporal aspect of the problem, and in general, produce worse estimates (see Section 4).

One way to build better temporal covariance estimation methods is to apply a static covariance estimation method in sliding window fashion while also adding a regularization term that enforces temporal consistency of estimates \([14, 31]\). We use a similar approach to extend linear CorEx \([29]\) for temporal covariance estimation. There are several reasons why we choose to base our method on linear CorEx. First, linear CorEx has been shown to have low sample complexity. This makes it possible to select small window sizes, which allows the method to detect short-term variations. Second, the complexity of linear CorEx scales linearly with \( p \) allowing us to apply it to extremely high-dimensional data (where \( p \) is greater than \( 10^3 \)). Third, the objective of linear CorEx is differentiable, which allows us to use any variant of gradient descent to train it.

#### 3.1 Linear CorEx

To proceed further we need some details of linear CorEx and one definition.

**Definition 3.1.** A jointly Gaussian distribution with \( p \) observed variables \((X_1, X_2, \ldots, X_p)\) and \( m \) hidden variables \((Z_1, Z_2, \ldots, Z_m)\) is called a \((p, m)\) non-overlapping Gaussian latent factor (NGLF) model if the joint distribution \( p(x, z) \) factorizes in the following way:

\[
p(x, z) = \left( \prod_{j=1}^m p(z_j) \right) \left( \prod_{i=1}^p p(x_i | z_{\pi_i}) \right),
\]

where \( \pi_i \in \{1, 2, \ldots, m\}, i = 1, 2, \ldots, p \).

The factorization of an NGLF model defines a directed probabilistic graphical model shown in Fig. 2. In other words, the NGLF model corresponds to the product of one-factor Gaussian models. Linear CorEx is a latent factor model designed to uncover the structure of a multivariate Gaussian variable. It is a variant of the more general CorEx method \([32]\). For given \( p \) dimensional Gaussian random variable \( X = (X_1, X_2, \ldots, X_p) \), the algorithm finds an \( m \) dimensional Gaussian random variable \( Z = (Z_1, Z_2, \ldots, Z_m) \), such that the joint system \((X, Z)\) is close to a \((p, m)\) NGLF model. The authors show that a jointly Gaussian distribution \( p(X, Z) \) is an NGLF model if and only if \( TC(X | Z) + TC(Z) = 0 \) and \( \forall i, TC(X_i | Z) = 0 \). This makes it possible to learn a NGLF model by minimizing \( TC(X | Z) + TC(Z) \) under the constraint \( \forall i, TC(X_i | Z) = 0 \). Linear CorEx attempts to solve the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad TC(X | Z) + TC(Z) \\
\text{subject to} & \quad TC(Z | X_i) = 0, \ i = 1, \ldots, p.
\end{align*}
\]

Note that if \( Z_i | X \sim \mathcal{N}(W_i \cdot x, \eta_i^2) \), then the joint distribution \( p(X, Z) \) is Gaussian. Assuming \( \forall i, \mathbb{E}[X_i] = 0 \) and \( \mathbb{E}[X_i^2] = 1 \), after some

\(^1\) There are some subtleties such as deciding whether to overlap periods, or what to do with remaining time steps when \( w \) does not divide \( T \).
Another reason is that linear CorEx doesn’t find a model of data that time periods to have similar covariance matrix estimates). As a first attempt towards building time-varying linear CorEx, we try to solve the following optimization problem:

$$\min_{W_1, \ldots, W_T} \sum_{t=1}^{T} O(W_t, D_t) + \lambda \sum_{t=1}^{T-1} \Phi(W_{t+1} - W_t),$$

where $O(W_t, D_t)$ is the linear CorEx objective with parameters $W_t$ applied to data $D_t$, and $\Phi$ is the penalty function, which in this work is either $\ell_1$ or $\ell_2$ vector norm. The former is suitable for systems with sudden changes, while the latter is better suited for smoothly varying systems. We name the method of Eq. (4) T-CorEx-simple.

While T-CorEx-simple follows the general framework of building a temporal covariance estimation method from a static covariance estimation method, it is not powerful enough to get significantly better performance compared to linear CorEx applied in sliding window fashion without a regularization term (see Table 1). This happens because of some properties of linear CorEx. Namely, in linear CorEx, $\Sigma_t$ is a function of $W_t$ and $D_t$ (see Eq. 3). Therefore, even when the regularization coefficient $\lambda$ is infinity, making $W_1 = W_2 = \ldots = W_T$, still $\Sigma_t \neq \Sigma_{t+1}$ in general. Therefore, the model can never produce very close estimates of covariance matrices. If we choose to explicitly penalize the difference, $(\Sigma_t - \Sigma_{t+1})$ instead of the difference $(W_t - W_{t+1})$, in general, there will be no $W_1, W_2, \ldots, W_T$ that will make the new regularization term equal to zero.

To make the regularization effective, we note that $\Sigma_t$ depends on $D_t$ through $R_{j,i}$, which in turn depends on $\mathbb{E}[X_t Z_j]$ and $\mathbb{E}[Z_j^2]$, where the expectations are taken over samples of $D_t$ using parameters $W_t$. When $D_t$ contains a small number of samples, these quantities can be quite noisy, resulting in noisy estimates $\Sigma_t$. To reduce the noise without increasing the window size, we propose a simple remedy. To estimate $\mathbb{E}[X_t Z_j]$ and $\mathbb{E}[Z_j^2]$ for time period $t$, we use not only the samples of $D_t$, but also samples of other time periods $D_{t+\delta}$. Of course, samples belonging to time periods far from $t$ are less important than the samples belonging to time periods close to $t$. Since both $\mathbb{E}[X_t Z_j]$ and $\mathbb{E}[Z_j^2]$ are expectations over the sample distribution, we change sample weights/probabilities to capture the intuition above. We introduce a new hyper-parameter $\beta$ that defines the rate of decay of "importance" as we go far from the current time period $t$. Samples of time period $\tau$ have weight $\alpha_\tau(\tau) = \beta^{t-\tau}(0 < \beta < 1)$ when they are used to estimate $\mathbb{E}[X_t Z_j]$ and $\mathbb{E}[Z_j^2]$ for time period $t$. Below shows how we estimate $\mathbb{E}[Z_j^2]$ for time period $t$:

$$\mathbb{E}[Z_j^2] \leftarrow \frac{1}{\sum_{\tau=1}^{T} \alpha_\tau(\tau)s_{\tau}},$$

We estimate $\mathbb{E}[X_t Z_j]$ for each time period similarly. For computational efficiency, at time period $t$ we do not consider time periods $\tau$ for which $\alpha_\tau(\tau) < 10^{-5}$. Summing up, we get the following optimization problem:

$$\min_{W_1, \ldots, W_T} \sum_{t=1}^{T} O(W_t, D_{1:t}) + \lambda \sum_{t=1}^{T-1} \Phi(W_{t+1} - W_t),$$

3.2 Time-Varying Linear CorEx

Hereafter, we assume that the data is already split into periods. We can represent the data as a sequence of collections of observations, $\{D_t\}_{t=1}^{T}$, where $D_t \in \mathbb{R}^{s_t \times p}$. As mentioned earlier, we train one linear CorEx for each time period and use regularization techniques to enforce temporal consistency of estimates (i.e., enforcing adjacent time periods to have similar covariance matrix estimates). As a first
where \(O(W_t, D_{1:T})\) is the linear CorEx objective with parameters \(W_t\) applied on data \(D_1, D_2, \ldots, D_T\), where samples of \(D_r\) have weight \(\alpha_r(t)\). We name the method of Eq. (5) T-CorEx.

Note that along the way we extended linear CorEx to work in cases where samples have weights assigned to them. This extended method can be helpful when different samples have different importance, reliability, or relevance.

### 3.3 Implementation

The authors of linear CorEx derive a quasi-Newton optimization algorithm to train their model. Unfortunately, we could not use a similar derivation for T-CorEx. Instead, we use gradient descent to optimize the objective of T-CorEx. More specifically, in all our experiments we use the Adam optimizer [18] with \(\alpha = 10^{-3}, \beta_1 = 0.9\) and \(\beta_2 = 0.999\). We noticed that optimizing the objective function of linear CorEx with gradient descent requires 2-4 times more iterations than with quasi-Newton, but it eventually converges to similar solutions in terms of objective value. We initialize the weights of T-CorEx with the weights of a linear CorEx trained on all samples of all time periods. Since training of T-CorEx involves many matrix multiplications, GPUs can be used for speeding up the training. In fact, when \(p \geq 10^3\), a single GPU can reduce the training time up to 10 times. T-CorEx is implemented in PyTorch [25]. The code can be found on GitHub.

### 3.4 Scalability

The time complexity of the linear CorEx at time period \(t\) is \(O(n_tmp)\), where \(p\) is the number of observed variables, \(m\) is the number of hidden variables, \(n_t\) is the number of samples used for estimating \(\mathbb{E}[X_t, Z_t]\) and \(\mathbb{E}[Z_t^2]\) for time period \(t\). Since we don’t consider time periods \(\tau\), for which \(\alpha_r(\tau) < 10^{-9}\), we get that \(\sum_{t=1}^{T} n_t = O(\sum_{t=1}^{T} s_t / \log \beta)\). Therefore, the time complexity of T-CorEx is \(O(nmp/\log \beta)\), where \(n = \sum_{t=1}^{T} s_t\) is the total number of samples. Explicitly computing covariance matrix using Eq. 3 has \(O(mp^2)\) complexity. However, one can avoid computing it explicitly, as the covariance estimates of T-CorEx are diagonal plus low-rank matrices. Multiplying such matrices with other matrices can be made faster using their low-rank plus diagonal decomposition. One can compute the determinant of such matrices in \(O(m^3 + m^2p)\) time using the generalization of the matrix determinant lemma. Furthermore, the inverse of such matrices has diagonal plus low-rank form and can be computed in \(O(m^3 + m^2p)\) time using the Woodbury matrix identity. The memory complexity of T-CorEx is \(O(m^T + np)\).

We want to emphasize that the proposed method has linear time and memory complexity w.r.t. the number of variables \(p\), assuming \(n, T\) and \(m\) don’t depend on \(p\). To our best knowledge, T-CorEx is the only temporal covariance estimation method that scales linearly with \(p\). Time-varying graphical lasso (TVGL) [14] and latent variable time-varying graphical lasso (LTGL) [31], which are the direct competitors of our method, have \(O(p^3)\) time complexity. Fig. 3 shows the scalability comparison of different temporal covariance estimation methods.

---

1https://github.com/harhro94/T-CorEx

---

### 4 EXPERIMENTS

We compare T-CorEx with other static and temporal covariance estimation methods on both synthetic and real-world datasets. The simplest baseline is the diagonal matrix with the sample variances of corresponding variables. Other static baselines are the Ledoit-Wolf (LW) method [20], oracle approximating shrinkage (OAS) method [7], factor analysis (FA), sparse PCA [22, 37], linear CorEx [29], graphical lasso (GLASSO) [13], and latent variable graphical lasso (LVGLASSO) [5]. The temporal baselines are time-varying graphical lasso (TVGL) [14] and latent variable time-varying graphical lasso (LTGL) [31]. In the experiments with synthetic data, we add two additional baselines: T-CorEx-no-reg (T-CorEx with \(\lambda \to 0\)) and T-CorEx-simple (T-CorEx with \(\beta \to 0\)) to show the importance of the regularization term and the importance of introducing sample weights respectively.

In all quantitative experiments we split the data into train, validation and test sets. We use the validation set to select hyperparameters of the baselines and we report the final scores on the test set. The metric we use in our experiments is the negative log-likelihood of estimates averaged over time periods. The grid of hyperparameters for each baseline can be found in the supplementary material.

#### 4.1 Synthetic Data

We design experiments with synthetic data to compare our method with other baselines in the case when the modeling assumptions of T-CorEx are met (i.e., the data of each period is generated from an NGLF model). We generate synthetic data for two scenarios. In the first scenario, the underlying covariance matrix is constant for the first half of the time periods. Then a sudden change happens after which the underlying covariance matrix remains constant for the remaining half of time periods. We call this scenario sudden change. In the second scenario, the underlying covariance matrix is slowly changing from \(\Sigma_1\) to \(\Sigma_2\). We call this scenario smooth change. The
with T-CorEx-no-reg, we conclude that even after introducing sam-
ple weights, having a regularization term that induces temporal
NGLF modeling assumptions are met. The advantage of T-CorEx
show that the proposed method gives state-of-the-art results when
train/validation/test splits. Again, the metric is time-averaged neg-
atives. The data is downloaded using the API of tradingeconomics.com.

| Method         | Training | Validation | Test  |
|----------------|----------|------------|-------|
| Linear CorEx   | 424.9 ± 40.1 | 384.7 ± 100.3 | 372.4 ± 54.6 |
| OAS            | 405.5 ± 34.2 | 376.8 ± 85.5 | 403.1 ± 51.9 |
| FA             | -         | -          | 900.3 ± 189.0 |
| Sparse PCA     | 507.7 ± 71.5 | 382.9 ± 66.7 | 301.1 ± 31.2 |
| Linear CorEx   | 458.1 ± 47.6 | 330.1 ± 31.9 | 279.7 ± 14.6 |
| GLASSO         | 473.3 ± 65.5 | 400.3 ± 54.8 | 304.8 ± 33.6 |
| LVGLASSO       | 430.6 ± 23.6 | 372.7 ± 59.0 | 289.3 ± 22.1 |
| TVGL           | 335.6 ± 24.8 | 298.3 ± 60.0 | 260.5 ± 32.6 |
| LTGL           | 329.3 ± 18.1 | 308.5 ± 49.1 | 252.0 ± 27.9 |
| T-CorEx-simple | 313.5 ± 16.4 | 269.5 ± 16.1 | 246.1 ± 12.9 |
| T-CorEx-no-reg | -         | -          | 244.0 ± 4.8 |

Table 2: Time-averaged negative log-likelihood of the estimates produced by different methods on stock market data, shown in mean ± standard deviation format. The means and standard deviations are computed for 10 random train/validation/test splits.

As all baselines used in this work need the time series to be divided into periods, we break the time series into non-overlapping periods, each containing w samples. We randomly partition samples of each period into train, validation and test sets containing 2w/3, w/6, w/6 samples respectively. After breaking time series into periods, we consider only the last 10 periods. Table 2 shows the results for w = 12, 24, 48, 96 averaged over 10 random train/validation/test splits. Again, the metric is time-averaged negative log-likelihood of the estimates under multivariate Gaussian distribution. We justify our choice of the metric with the fact that all baselines besides the Ledoit-Wolf method assume the data has Gaussian distribution.

For better interpretation of results, we suggest looking at the distribution of test scores for those 10 different runs (Fig. 4). First of all, many baselines have high standard deviations, but the figure shows that it is usually caused by some runs that produce very

full details of the data generation process for these two scenarios are presented in the supplementary material.

**Sudden Change.** To create data with a sudden change, we generate two different (p, m) NGLF models—\(p_1(x, z)\) and \(p_2(x, z)\). The data of the first five periods is generated from \(p_1(x, z)\), while the data of the next five periods is generated from \(p_2(x, z)\). We generate 16 training, 16 validation and 1000 testing samples for each period. The left multi-column of Table 1 shows the comparison of baselines on this type of data for \(m = 8, p = 128\) and varying values of \(s\).

**Smooth Change.** To create a synthetic dataset with smooth change, we generate two \((p, m)\) NGLF models, \(p^{(1)}(x, z)\) and \(p^{(2)}(x, z)\). We start from \(p^{(1)}(x, z)\) and smoothly change the model into \(p^{(T)}(x, z)\), so that for each \(t = 2, 3, \ldots, T - 1\) time period the joint distribution remains NGLF. Let \(p^{(t)}(x, z)\) denote the NGLF model at time period \(t\). We generate 16 training, 16 validation and 1000 testing samples from \(p^{(t)}(x, z)\). The right multi-column of Table 1 shows the comparison of baselines on this type of data for \(m = 8, p = 128\) and varying values of \(s\).

The results of sudden change and smooth change experiments show that the proposed method gives state-of-the-art results when NGLF modeling assumptions are met. The advantage of T-CorEx is significant when the number of samples is small. Comparing T-CorEx-simple with linear CorEx, we see that indeed, the regulariza-
tion term of T-CorEx-simple is not enough to get the best performance. Furthermore, the results show that introducing sample weights improves the results significantly. Comparing T-CorEx with T-CorEx-no-reg, we conclude that even after introducing sample weights, having a regularization term that induces temporal consistency improves the results.

### 4.2 Stock Market Data

Next, we evaluate our method on a stock prices dataset, where the modeling assumptions are not met (i.e. the data of each time step does not correspond to an NGLF model). We take the daily sampled historical data of stock prices of S&P 500 index from January, 2000 to January, 2016.\(^3\) For simplicity, we keep the stocks that are present for at least 95% of the period. The resulting dataset contains daily prices for 391 stocks. We compute the daily log-returns, standardize each variable, and add isotropic Gaussian noise with \(10^{-4}\) variance.

| Method         | Sudden Change | Smooth Change |
|----------------|---------------|---------------|
| Ground Truth   | 196.0 ± 196.0 | 196.0 ± 196.0 |
| Diagonal       | 279.6 ± 266.7 | 254.9 ± 277.4 |
| LW             | 286.2 ± 266.7 | 227.5 ± 287.3 |
| OAS            | 281.5 ± 266.3 | 227.5 ± 279.9 |
| FA             | 524.4 ± 236.9 | 201.3 ± 267.5 |
| Sparse PCA     | 270.5 ± 232.9 | 200.7 ± 274.7 |
| Linear CorEx   | 312.3 ± 221.5 | 197.7 ± 233.0 |
| GLASSO         | 266.5 ± 242.0 | 205.5 ± 280.1 |
| LVGLASSO       | 271.6 ± 245.5 | 210.2 ± 276.6 |
| TVGL           | 237.5 ± 224.5 | 203.7 ± 259.0 |
| LTGL           | 248.6 ± 230.0 | 204.7 ± 265.0 |
| T-CorEx-simple | 228.0 ± 213.8 | 197.7 ± 250.6 |
| T-CorEx-no-reg | 275.8 ± 217.9 | 199.6 ± 243.3 |

Table 1: Time-averaged negative log-likelihood of the estimates produced by different methods on synthetic data (\(m = 8, p = 128\)).

\(^3\)The data is downloaded using the API of tradingeconomics.com.
bad estimates. The results suggest that T-CorEx is doing significantly better when the number of samples is small ($w = 12, 24$). For $w = 48$, we see that TVGL, LTGL and T-CorEx perform similarly, even though T-CorEx has slightly better mean because it has fewer outliers with very poor performance. At $w = 96$, LTGL starts to produce better estimates. Notably, T-CorEx always has the smallest standard deviation. We hypothesize that the assumption of T-CorEx of data distribution being approximately NGLF is more restrictive than the sparsity assumption of TVGL and LTGL. Therefore, the variance of T-CorEx is relatively smaller. On the other hand, the assumptions of T-CorEx do not introduce large biases. In fact, we see that T-CorEx has the smallest bias when the number of samples is small, and has comparable bias otherwise. Interestingly, as we increase $w$, LW and OAS become worse. This happens because these methods start to shrink the sample covariance matrix less as the number of samples increases.

**Qualitative Analysis.** With quantitative analysis we showed that the estimates of T-CorEx are much more accurate than those of linear CorEx applied in sliding window fashion. Next, we want to see what are the qualitative differences of those better estimates for stock market data. For this purpose we plot the entries of inverse correlation matrix that have absolute value greater than some threshold. This can be interpreted as plotting the adjacency matrix of a Markov random field. First, we train a linear CorEx on the whole period, ignoring the temporal aspect (the left part of Fig. 5). This shows how the system looks like “on average.” We see that most of the edges are within sectors. Then we train the proposed model with window size equal to two weeks and plot its estimate for a random time period $t_0$ (the middle part of Fig. 5). First, We see that the sectors are not as densely connected. Second, T-CorEx captures some dependencies that are not present for the whole period. The opposite is also true—two variables can be directly connected on average, but be disconnected for some period of time (e.g., some connections between sectors “Materials” and “Energy”). The advantage of T-CorEx lies in the ability of detecting correlations that occur only for a short period of time. Methods requiring large number of samples (large window size) cannot detect such correlations. To finalize our analysis, we fit linear CorEx on $s$ samples around the same random time period $t_0$. When $s$ is too small, the estimates are too noisy. When $s$ is too large the estimates are less related to the true statistics of time period $t_0$. The right part of Fig. 5 shows the estimate of linear CorEx for the best value of $s$ ($s = 128$). The estimate of linear CorEx is qualitatively different from that of T-CorEx and gives a significantly worse score, indicating the inherent problems of applying a static covariance estimation method in sliding window fashion.

**Change Point Detection.** Estimated covariance/precision matrices can be used to detect the shifts in the underlying system [14, 31]. One simple way to detect changes in the underlying structure is to look at the Frobenius norm of the difference of inverse correlation matrices of neighboring time periods. Fig. 6 shows that from 2000 to 2016, T-CorEx detects all but one of the major US stock market events. Interestingly, for some events major changes are visible up to two months earlier. However, the algorithm cannot be used for early detection because both the regularization term and the temporal smoothing exploit data of future time steps. This is also true for TVGL and LTGL methods. Fig. 6 also shows the same analysis for sliding window linear CorEx with optimal window size. The peaks of linear CorEx align much worse with the ground truth events. Furthermore, the estimates of linear CorEx are temporally less consistent. As expected, linear CorEx does not detect changes that occur in very short periods, such as the changes related to the September 11 attacks and the 2010 flash crash.

---

In our experiments the threshold is set to 0.025.
4.3 High-Resolution fMRI Data

To demonstrate the scalability and usefulness of the proposed method, we apply it to high-dimensional functional magnetic resonance images (fMRI). The most common measurement in fMRI is blood oxygen level-dependent (BOLD) contrast, which measures blood flow changes in biological tissues ("activation"). In a typical fMRI session, which lasts 3-10 minutes, hundreds of high-resolution brain images are captured, each having 100K-600K volumetric pixels (voxels). Correlation analysis is widely used to study functional connections between brain regions [27]. While in general these analyses are conducted assuming static covariance, recently time-varying covariance ("dynamic functional connectivity") has received more attention [6, 21]. This latter case is exactly the use case of T-CorEx.

---

The events are listed in https://en.wikipedia.org/w/index.php?title=List_of_stock_market_crashes_and_bear_markets&oldid=859807280.
While the induced shift is clearly synthetic, this experiment shows T-CorEx on the resulting data. The orange plot of Fig. 7 shows the Frobenius norm of differences of inverse correlation matrices of neighboring time periods for a T-CorEx trained on the middle 12 time periods after swapping its first 6 and last 6 time periods.

We demonstrate the feasibility of T-CorEx in fMRI analysis by inducing an artificial change point in an otherwise stable time series. While the induced shift is clearly synthetic, this experiment shows possible value for the fMRI community in detecting natural change points and/or using T-CorEx for more nuanced domain-specific analyses, demonstrating that T-CorEx can scale to the 100K+ variable regime. First, we fit T-CorEx on a resting-state fMRI session from the MyConnectome project [26]. The session has 518 time-point volumes, each having 148262 voxels. We divide the whole period into 20 non-overlapping periods, ignoring the first 18 time-points.

The blue curve in Fig. 7 shows the Frobenius norm of differences of inverse correlation matrices of neighboring time periods. We see relatively large variability in the beginning and in the end of the session. Next, we consider the middle 12 periods (i.e., removing 4 periods from both sides). We swap the first 6 and the last 6 periods of those periods, creating an artificial change in the middle, and retrain T-CorEx on the resulting data. The orange plot of Fig. 7 shows the Frobenius norm of differences of inverse correlation matrices of neighboring time periods for this case. T-CorEx detects the shift we created, while other methods do not scale to this regime.

T-CorEx can also provide secondary analyses by grouping correlated regions. We assign each voxel to the latent factor that has the largest mutual information with it, forming groups by each factor. This produces \( m = 50 \) clusters of voxels for each time period. Fig. 8 shows three clusters from time period 12, identified using NeuroSynth [34]. Clusters displayed in (a) and (b) correspond to standard anatomic regions [4], namely both left and right Brodmann area 40, as well as left Brodmann areas 44/45 (Broca’s Area), which is known to be a highly lateralized (asymmetric) region. Cluster (c) displays distributed activation in the left hemisphere. We found that cluster (a) is present in all time periods; (b) is present starting at time period 3. These two clusters exhibit small variations over time. Cluster (c) is present starting at time period 11, but varies more compared to the other two clusters.

5 RELATED WORK

Many works have addressed the problem of high-dimensional covariance estimation [12]. One major direction of estimating high-dimensional covariance matrices is introducing sparsity constraints. Sparsity constraints are often used to reduce the effective number of parameters of the model, to regularize the model or to incorporate our beliefs. Sparse covariance matrix estimation for multivariate Gaussian random variables is investigated in [3]. Most often sparsity constraints are imposed on the inverse covariance matrix (precision matrix). The precision matrix encodes the conditional independences between pairs of variables. Learning a sparse precision matrix corresponds to learning a sparse graphical model [2, 19]. The graphical lasso method [13] does sparse inverse covariance matrix estimation for multivariate Gaussian random variables. The problem of network inference using graphical lasso is well studied [13, 23, 28, 35].

In many real-world applications, there are latent factors influencing the system. Modeling those factors usually leads to better estimates of the covariance matrix of the observed variables. Factor analysis and probabilistic PCA [30] are latent factor models that can be used for covariance estimation. However, they fail in undersampled regimes. Sparse PCA [22, 37] remedies this problem. Covariance estimation can also be done by learning graphical models with latent factors [5, 8, 9]. The latent variable graphical lasso method [5] learns a sparse graphical model with latent factors. Linear CorEx [29] is another latent factor model and is central to this work. Some details of linear CorEx are provided in Subsection 3.1.

Many works extended a particular covariance estimation method for temporal covariance estimation. Sparse PCA has been adapted for high-dimensional multivariate vector autoregressive time series [33]. The time-varying graphical lasso method [14] extends graphical lasso. It breaks down the time series into periods and applies graphical lasso on each period, while also adding a regularization term for inducing temporal consistency of estimates. In a similar fashion, latent variable time-varying graphical lasso [31] extends latent variable graphical lasso. T-CorEx takes the same approach, but has a couple of differences. Instead of forcing sparsity, T-CorEx attempts to learn models closer to NGLF. Additionally, unlike T-CorEx, none of the mentioned methods use weighted samples for estimating their parameters.

The time and memory complexity of a method are crucial in high-dimensional settings. All non-CorEx temporal covariance estimation methods listed here have at least quadratic time complexity with respect to the number of observed variables, \( p \). In fact, most of them have \( \Omega(p^3) \) time complexity as they require computing the inverse and/or SVD of a \( p \times p \) matrix. Most methods store \( p \times p \) matrices explicitly. All these methods become inapplicable for systems having more than \( 10^4 \) variables. There are successful attempts to build faster static covariance/precision matrix estimation methods [15–17, 36]. To our knowledge, our proposed method is the only temporal covariance estimation method that has linear time and memory complexity with respect to \( p \).

We use the processed resting-state fMRI of session 014 (dataset version: 1.0.4). We then do spatial smoothing using a Gaussian filter with fwhm=8mm.

Note, although the correlation matrices are extremely large, we are able to compute the inverses and norms of differences, since estimates of T-CorEx are diagonal plus low-rank matrices.
6 CONCLUSION AND FUTURE WORK

We developed a novel temporal covariance estimation method called T-CorEx. The proposed method has linear time and memory complexity with respect to the number of observed variables and makes minimal assumptions about the dynamics of the system. We evaluated our method on both synthetic and real-world datasets, showing state-of-the-art results in highly undersampled regimes. We also studied the range of possible applications of T-CorEx.

In future research we aim to simplify the hyperparameter selection of T-CorEx. An interesting direction of research is inferring the sample weights $\alpha_t(\tau)$ automatically. A successful implementation of this will increase the modeling capacity of T-CorEx. Another interesting extension of the model is adapting it to work in the online regime. In fact, exploiting samples of future time steps does not allow us to use the method for early detection and forecasting.

ACKNOWLEDGMENTS

The authors thank Federico Tomasi for his valuable help on training latent variable time-varying graphical lasso, and Neal Lawton for insightful discussions. This research is based upon work supported in part by DARPA, via W911NF-16-1-0575, and the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2016-16041100004. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. Additionally, H. Harutyunyan is supported by an Annenberg fellowship.

REFERENCES

[1] Amr Ahmed and Eric P Xing. 2009. Recovering time-varying networks of dependencies in social and biological studies. Proceedings of the National Academy of Sciences 106, 29 (2009), 11878–11883.
[2] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d’Aspremont. 2008. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. J. Mach. Learn. Res. 9 (June 2008), 485–516.
[3] Jacob Bien and Robert J. Tibshirani. 2011. Sparse estimation of a covariance matrix. Biometrika 98, 4 (2011), 807–820.
[4] Korbinian Brodmann. 1909. Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellverbandes. Barth.
[5] Venkat Chandrasekaran, Pablo A. Parrilo, and Alan S. Willsky. 2012. Latent variable graphical model selection via convex optimization. Ann. Statist. 40, 4 (08 2012), 1935–1967.
[6] Cate Chang and Gary H. Glover. 2010. Time–frequency dynamics of resting-state brain connectivity measured with fMRI. Neuroimage 50, 1 (2010), 81 – 98.
[7] Y. Chen, A. Wiesel, Y. C. Eldar, and A. O. Hero. 2010. Shrinkage Algorithms for MMSE Covariance Estimation. IEEE Transactions on Signal Processing 58, 10 (Oct 2010), 5016–5029.
[8] Myung Jin Choi, Venkat Chandrasekaran, and Alan S Willsky. 2010. Gaussian multiresolution models: Exploiting sparse Markov and covariance structure. IEEE Transactions on Signal Processing 58, 3 (2010), 1022–1024.
[9] Myung Jin Choi, Vincent YF Tan, Animashree Anandkumar, and Alan S Willsky. 2011. Learning latent tree graphical models. Journal of Machine Learning Research 12, May (2011), 1771–1812.
[10] Gianluca De Nard, Olivier Ledoit, Michael Wolf, et al. 2018. Factor models for portfolio selection in large dimensions: the good, the better, and the ugly. Technical Report. Department of Economics-University of Zurich.
[11] Robert F. Engle, Olivier Ledoit, and Michael Wolf. 2017. Large Dynamic Covariance Matrices. Journal of Business & Economic Statistics 0, 0 (2017), 1–13.
[12] Jianqing Fan, Yuan Liao, and Han Liu. 2016. An overview of the estimation of large covariance and precision matrices. The Econometrics Journal 19, 1 (2016), C1–C32.
[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. Biostatistics 9, 3 (2008), 432–441.
[14] David Hallac, Youngsu Park, Stephen Boyd, and Jure Leskovec. 2017. Network Inference via the Time-Varying Graphical Lasso. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’17). ACM, New York, NY, USA, 205–213.
[15] Jean Honorio and Tommi Jaakkola. 2013. Inverse Covariance Estimation for High-dimensional Data in Linear Time and Space: Spectral Methods for Riced and Sparse Models. In Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI’13). AUAI Press, Arlington, Virginia, United States, 291–300.
[16] Cho-Jui Hsieh, Mätys A. Sustik, Inderjit S. Dhillon, and Pradeep Ravikumar. 2014. QIC. Quadratic Approximation for Sparse Inverse Covariance Estimation. Journal of Machine Learning Research 15 (2014), 2911–2947.
[17] Cho-Jui Hsieh, Mätys A Sustik, Inderjit S Dhillon, Pradeep K Ravikumar, and Russell Foulds. 2013. BIG & QIC: Sparse inverse covariance estimation for a million variables. In Advances in Neural Information Processing Systems. 3165–3173.
[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
[19] Steffen L Lauritzen. 1996. Graphical models. Vol. 17. Clarendon Press.
[20] Olivier Ledoit and Michael Wolf. 2004. A well-conditioned estimator for large-dimensional covariance matrices. Journal of Multivariate Analysis 88, 2 (2004), 365 – 411.
[21] Daniel J Lurie, Daniel Kessler, Danielle S Bassett, Richard F Betzel, Michael Breakspear, Shella Keilholz, Aaron Kucyi, Raphaël Liègeois, Martin A Lindquist, Anthony R McIntosh, and et al. 2018. On the nature of resting IMRI and time-varying functional connectivity. https://doi.org/10.31234/osf.io/xtze
[22] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2009. Online Dictionary Learning for Sparse Coding. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML ’09). ACM, New York, NY, USA, 689–696.
[23] Nicolai Meinshausen and Peter Bühlmann. 2006. High-Dimensional Graphs and Variable Selection with the Lasso. The Annals of Statistics 34, 3 (2006), 1436–1462.
[24] Riccardo Pio Monti, Peter Hellyer, David Sharp, Robert Leech, Christoforos Anagnostopoulos, and Giovanni Montana. 2014. Estimating time-varying brain connectivity networks from functional MRI time series. Neuroimage 103 (2014), 427 – 443.
[25] Adam Pazske, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In NIPS-W.
[26] Russell A Poldrack, Timothy O Laumann, Oluwasanmi Koyejo, Brenda Gregory, Ashleigh Hower, Mei-Yen Chen, Krzysztof J Gorgolewski, Jeffrey Luci, Sung Jun Joo, Ryan L Boyd, et al. 2015. Long-term neural and physiological phenotyping of a single human. Nature communications 6 (2015), 8885.
[27] Maria Giulia Preti, Thomas AW Bolton, and Dimitri Van De Ville. 2017. The dynamic functional connectome: State-of-the-art and perspectives. Neuroimage 160 (2017), 41 – 54. Functional Architecture of the Brain.
Supplementary Material: Efficient Covariance Estimation from Temporal Data

A DETAILS OF GENERATING NGLF DATA
To describe a \((p, m)\) NGLF model, it is enough to specify 6 quantities: \(\mathbb{E}[Z_j], \mathbb{E}[Z_j^2], \mathbb{E}[X], \mathbb{E}[X^2], \pi_j\) - the parent of \(X_j\), and \(\rho_i\) - the correlation of \(X_i\) and its parent. Note the moments of \(Z_j\) don’t affect the marginal distribution of \(X\), so we set \(\mathbb{E}[Z_j] = 0\) and \(\mathbb{E}[Z_j^2] = 1\). We also set \(\mathbb{E}[X] = 0\). Summing up, we need only \(\pi_i, \rho_i\) and \(\sigma_j = \sqrt{\mathbb{E}[X_j^2]}\). In our experiments, we have \(\pi_i \sim \text{Uniform}[1, 2, \ldots, m]\), \(\sigma_j \sim U[1/4, 4]\). To define the correlations \(\rho_i\), for each \(X_i\) we first sample the signal to noise ratio (snr) of \(X_i|Z\) uniformly from \([0, 5]\) and then set the correlation between \(X_i\) and \(Z\), \(\rho_i = \text{sgn}(\xi)\sqrt{\text{snr}}\), with \(\xi \sim N(0, 1)\). This way we control the average signal to noise ratio similar to the experiments done in [29]. An example of covariance matrix corresponding to an NGLF model is shown in Fig. 9.

As stated in the main text, for creating a synthetic dataset with smooth change, we generate 2 \((p, m)\) NGLF models, \(\rho^{(i)}(x, z)\) and \(\rho^{(T)}(x, z)\). Let the former be characterized by \(\pi^{(i)}, \rho^{(i)}, \sigma^{(i)}\), and the latter be characterized by \(\pi^{(T)}, \rho^{(T)}, \sigma^{(T)}\). We start from \(\rho^{(i)}(x, z)\) and smoothly change the model into \(\rho^{(T)}(x, z)\), so that for each \(t = 2, 3, \ldots, T - 1\) time period the joint distribution remains NGLF. We define the parameters of \(\rho^{(t)}(t = 2, 3, \ldots, T - 1)\) in the following way:

\[
\rho^{(t)} = \alpha t \rho^{(1)} + (1 - \alpha t) \rho^{(T)} , \\
\sigma^{(t)} = \alpha t \sigma^{(1)} + (1 - \alpha t) \sigma^{(T)} ,
\]

with \(\alpha_t = (T - t)/(T - 1)\). To define \(\pi^{(t)}\), we randomly select when each variable will change its parent from \(\pi^{(1)}\) to \(\pi^{(T)}\). Formally, we sample \(\tau_i \sim \text{Uniform}[2, 3, \ldots, T]\) and set \(\pi^{(t)}_i = \pi^{(1)}_i\) if \(t < \tau_i\), otherwise we set \(\pi^{(t)}_i = \pi^{(T)}_i\).

B IMPLEMENTATION DETAILS
In this section we discuss the implementations of the baselines. We use scikit-learn implementations of LW, OAS, FA, and Sparse PCA methods. We use the original implementations of linear CorEx, TVGL, and LTGL. For GLASSO, we tried the scikit-learn implementation, the QUIC method (http://www.cs.utexas.edu/~sustik/QUIC/), and TVGL with \(\beta = 0\). In our experiment, the TVGL implementation was always better. Therefore, we selected the latter implementation. For LVGLASSO, we used the implementation available in the REGALN repository (https://github.com/fdtomas/regain), which also contains the original implementation of LTGL.

C HYPERPARAMETER GRIDS
In this section we describe the hyperparameter grids we used in our quantitative experiments. Tables 3, 4, and 5 describe the grids of hyperparameters we tried in the sudden change, smooth change, and stock market experiments respectively. Note that the excluded baselines (beside T-CorEx-simple and T-CorEx-no-reg) have no hyperparameters. In all our experiments we made sure that increasing the “max_iter” parameter does not improve the results of any baseline. Also, we made sure that the best values of hyperparameters are not the corner values. In the sudden and smooth change experiments we know the ground truth value of \(m\) (the number of latent factors). Therefore, we do not search the best value of \(m\) for FA, sparse PCA, linear CorEx, and T-CorEx baselines. This way we compare the modeling performances of our baselines, rather than how they depend on the choices of their hyperparameters. In T-CorEx, “l1” and “l2” hyperparameters are mutually exclusive, meaning that if one is set to a non-zero value, the other one will be zero. In LVGLASSO and LTGL, we set \(\rho = 1/\sqrt{s}\), where \(s\) is the
| Method        | Hyperparameters                                                                 |
|--------------|---------------------------------------------------------------------------------|
| Sparse PCA   | alpha: [0.1, 0.3, 1.0, 3.0, 10.0, 30.0] max_iter: 500                           |
| Linear CorEx | max_iter: 500                                                                    |
| GLASSO       | lamb: [0.01, 0.03, 0.1, 0.3, 1.0, 3.0] max_iter: 500                           |
| LVGLASSO     | alpha: [0.03, 0.1, 0.3, 1.0, 3.0, 10.0] tau: [1.0, 3.0, 10.0, 30.0, 100.0, 300.0] rho: $1/\sqrt{3}$ max_iter: 500 |
| TVGL         | lamb: [0.01, 0.03, 0.1, 0.3, 1.0, 3.0] max iter: 500                           |
|             | beta: [0.03, 0.1, 0.3, 1.0, 3.0] indexOfPenalty: [1, 2, 3] max_iter: 500       |
| LTGL         | alpha: [0.3, 1.0, 3.0, 10.0] tau: [10.0, 30.0, 100.0, 300.0, 1e3] beta: [1.0, 3.0, 10.0, 30.0, 100.0] psi: [1, 2, Laplacian] phi: [1, 2, Laplacian] rho: $1/\sqrt{3}$ max_iter: 500 |
| T-CorEx      | l1: [0.0, 0.03, 0.1, 0.3, 1.0, 3.0, 10.0] l2: [0, 0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0, 300.0] gamma: [1e-9, 0.1, 0.3, 0.4, 0.5, 0.6, 0.7] max_iter: 500 |

Table 3: Hyperparameter grids in the sudden change experiment.

number of training samples in each period. We noticed that these methods are not sensitive to this hyperparameter. We found this choice in their code repository.

| Method        | Hyperparameters                                                                 |
|--------------|---------------------------------------------------------------------------------|
| Sparse PCA   | n_components: [16, 32, 64, 128] max_iter: 500                                   |
| Linear CorEx | max_iter: 500                                                                    |
| GLASSO       | lamb: [0.01, 0.03, 0.1, 0.3, 1.0, 3.0] tau: [1.0, 3.0, 10.0, 30.0, 100.0, 300.0] rho: $1/\sqrt{3}$ max_iter: 500 |
| LVGLASSO     | alpha: [0.03, 0.1, 0.3, 1.0, 3.0, 10.0] beta: [0.03, 0.1, 0.3, 1.0, 3.0, 10.0] indexOfPenalty: [1, 2, 3] max_iter: 500 |
| TVGL         | lamb: [0.01, 0.03, 0.1, 0.3, 1.0, 3.0] max iter: 500                           |
|             | beta: [0.03, 0.1, 0.3, 1.0, 3.0] indexOfPenalty: [1, 2, 3] max_iter: 500       |
| LTGL         | alpha: [0.3, 1.0, 3.0, 10.0] tau: [10.0, 30.0, 100.0, 300.0, 1e3] beta: [1.0, 3.0, 10.0, 30.0, 100.0] psi: [1, 2, Laplacian] phi: [1, 2, Laplacian] rho: $1/\sqrt{3}$ max_iter: 500 |
| T-CorEx      | l1: [0.0, 0.03, 0.1, 0.3, 1.0, 3.0, 10.0] l2: [0, 0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0, 300.0] gamma: [1e-9, 0.1, 0.3, 0.4, 0.5, 0.6, 0.7] max_iter: 500 |

Table 4: Hyperparameter grids in the smooth change experiment.

| Method        | Hyperparameters                                                                 |
|--------------|---------------------------------------------------------------------------------|
| FA           | n_components: [16, 32, 64, 128] max_iter: 500                                   |
| Sparse PCA   | n_components: [0.1, 0.3, 1.0, 3.0, 10.0, 30.0] max_iter: 500                   |
| Linear CorEx | n_hidden: [16, 32, 64, 128] max_iter: 500                                       |
| GLASSO       | lamb: [0.01, 0.03, 0.1, 0.3, 1.0, 3.0] tau: [1.0, 3.0, 10.0, 30.0, 100.0, 300.0] rho: $1/\sqrt{3}$ max_iter: 500 |
| LVGLASSO     | alpha: [0.03, 0.1, 0.3, 1.0, 3.0, 10.0] beta: [10.0, 30.0, 100.0, 300.0, 1e3] indexOfPenalty: [1, 2, 3] max_iter: 500 |
| TVGL         | lamb: [0.01, 0.03, 0.1, 0.3, 1.0, 3.0] max iter: 500                           |
|             | beta: [0.03, 0.1, 0.3, 1.0, 3.0] indexOfPenalty: [1, 2, 3] max_iter: 500       |
| LTGL         | alpha: [0.3, 1.0, 3.0, 10.0] tau: [10.0, 30.0, 100.0, 300.0, 1e3] beta: [1.0, 3.0, 10.0, 30.0, 100.0] psi: [1, 2, Laplacian] phi: [1, 2, Laplacian] rho: $1/\sqrt{3}$ max_iter: 500 |
| T-CorEx      | l1: [0.0, 0.03, 0.1, 0.3, 1.0, 3.0, 10.0] l2: [0, 0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0, 300.0] gamma: [1e-9, 0.1, 0.3, 0.4, 0.5, 0.6, 0.7] max_iter: 500 |

Table 5: Hyperparameter grids in the stock market experiment.