Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction

Hsiang-Fu Yu
University of Texas at Austin
rofuyu@cs.utexas.edu

Nikhil Rao
Technicolor Research
nikhilrao86@gmail.com

Inderjit S. Dhillon
University of Texas at Austin
inderjit@cs.utexas.edu

Abstract

Time series prediction problems are becoming increasingly high-dimensional in modern applications, such as climatology and demand forecasting. For example, in the latter problem, the number of items for which demand needs to be forecast might be as large as 50,000. In addition, the data is generally noisy and full of missing values. Thus, modern applications require methods that are highly scalable, and can deal with noisy data in terms of corruptions or missing values. However, classical time series methods usually fall short of handling these issues. In this paper, we present a temporal regularized matrix factorization (TRMF) framework which supports data-driven temporal learning and forecasting. We develop novel regularization schemes and use scalable matrix factorization methods that are eminently suited for high-dimensional time series data that has many missing values. Our proposed TRMF is highly general, and subsumes many existing approaches for time series analysis. We make interesting connections to graph regularization methods in the context of learning the dependencies in an autoregressive framework. Experimental results show the superiority of TRMF in terms of scalability and prediction quality. In particular, TRMF is two orders of magnitude faster than other methods on a problem of dimension 50,000, and generates better forecasts on real-world datasets such as Wal-mart E-commerce datasets.

1 Introduction

Time series analysis is a central problem in many applications such as demand forecasting and climatology. Often, such applications require methods that are highly scalable to handle a very large number (n) of possibly inter-dependent one-dimensional time series and/or have a large time frame (T). For example, climatology applications involve data collected from possibly thousands of sensors, every hour (or less) over several years. Similarly, a store tracking its inventory would track thousands of items every day for multiple years. Not only is the scale of such problems huge, but they might also involve missing values, due to sensor malfunctions, occlusions or simple human errors. Thus, modern time series applications present two challenges to practitioners: scalability to handle large n and T and the flexibility to handle missing values.

Most approaches in the traditional time series literature such as autoregressive (AR) models or dynamic linear models (DLM)[7, 21] focus on low-dimensional time-series data and fall short of handling the two aforementioned issues. For example, an AR model of order L requires $O(TL^2n^4 + L^3n^6)$ time to estimate $O(Ln^2)$ parameters, which is prohibitive even for moderate values of n. Similarly, Kalman filter based DLM approaches need $O(kn^2T + k^3T)$ computation cost to update parameters, where k is the latent dimensionality, which is usually chosen to be larger than n in many situations [13]. As a specific example, the maximum likelihood estimator implementation in the widely used R-DLM package [12], which relies on a general optimization solver, cannot scale beyond

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

n in the tens. (See Appendix D for details). On the other hand, for models such as AR, the flexibility to handle missing values can also be very challenging even for one-dimensional time series [1], let alone the difficulty to handle high dimensional time series.

A natural way to model high-dimensional time series data is in the form of a matrix, with rows corresponding to each one-dimensional time series and columns corresponding to time points. In light of the observation that n time series are usually highly correlated with each other, there have been some attempts to apply low-rank matrix factorization (MF) or matrix completion (MC) techniques to analyze high-dimensional time series [2, 14, 16, 23, 26]. Unlike the AR and DLM models above, state-of-the-art MF methods *scale linearly* in n, and hence can handle large datasets. Let $Y \in \mathbb{R}^{n\times T}$ be the matrix for the observed n-dimensional time series with Y_{it} being the observation at the t-th time point of the i-th time series. Under the standard MF approach, Y_{it} is estimated by the inner product $\boldsymbol{f}_i^\top \boldsymbol{x}_t$, where $\boldsymbol{f}_i \in \mathbb{R}^k$ is a k-dimensional latent embedding for the i-th time series, and $\boldsymbol{x}_t \in \mathbb{R}^k$ is a k-dimensional latent temporal embedding for the t-th time point. We can stack the $\boldsymbol{x}_t$s into the columns into a matrix $X \in \mathbb{R}^{k\times T}$ and $\boldsymbol{f}_i^\top$ into the rows of $F \in \mathbb{R}^{n\times k}$ (Figure 1) to get $Y \approx FX$. We can then solve:

$$\min_{F,X} \sum_{(i,t)\in\Omega} \left(Y_{it} - \boldsymbol{f}_i^\top \boldsymbol{x}_t\right)^2 + \lambda_f \mathcal{R}_f(F) + \lambda_x \mathcal{R}_x(X), \tag{1}$$

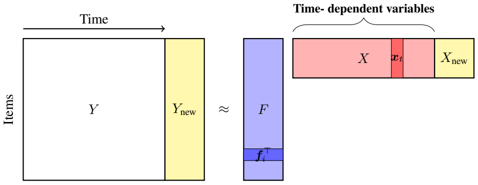

Figure 1: Matrix Factorization model for multiple time series. F captures features for each time series in the matrix Y, and X captures the latent and time-varying variables.

where Ω is the set of the observed entries. $\mathcal{R}_f(F)$, $\mathcal{R}_x(X)$ are regularizers for F and X, which usually play a role to avoid overfitting and/or to encourage some specific *temporal structures* among the embeddings. It is clear that the common choice of the regularizer $\mathcal{R}_x(X) = \|X\|_F$ is no longer appropriate for time series applications, as it does not take into account the ordering among the temporal embeddings $\{\boldsymbol{x}_t\}$. Most existing MF approaches [2, 14, 16, 23, 26] adapt graph-based approaches to handle temporal dependencies. Specifically, the dependencies are described by a weighted similarity graph and incorporated through a Laplacian regularizer [18]. However, graph-based regularization fails in cases where there are negative correlations between two time points. Furthermore, unlike scenarios where explicit graph information is available with the data (such as a social network or product co-purchasing graph for recommender systems), explicit temporal dependency structure is usually unavailable and has to be inferred or approximated, which causes practitioners to either perform a separate procedure to estimate the dependencies or consider very short-term dependencies with simple fixed weights. Moreover, existing MF approaches, while yielding good estimations for missing values in past points, are poor in terms of forecasting future values, which is the problem of interest in time series analysis.

In this paper, we propose a novel temporal regularized matrix factorization framework (TRMF) for high-dimensional time series analysis. In TRMF, we consider a principled approach to **describe** the structure of temporal dependencies among latent temporal embeddings $\{\boldsymbol{x}_t\}$ and design a temporal regularizer to **incorporate** this temporal structure into the standard MF formulation. Unlike most existing MF approaches, our TRMF method supports data-driven temporal dependency learning and also brings the ability to **forecast** future values to a matrix factorization approach. In addition, inherited from the property of MF approaches, TRMF can easily handle high-dimensional time series data even in the presence of many missing values. As a specific example, we demonstrate a novel autoregressive temporal regularizer which encourages AR structure among temporal embeddings $\{\boldsymbol{x}_t\}$. We also make connections between the proposed regularization framework and graph-based approaches [18], where even negative correlations can be accounted for. This connection not only leads to better understanding about the dependency structure incorporated by our framework but also brings the benefit of using off-the-shelf efficient solvers such as GRALS [15] directly to solve TRMF.

Paper Organization. In Section 2, we review the existing approaches and their limitations on data with temporal dependencies. We present the proposed TRMF framework in Section 3, and show that the method is highly general and can be used for a variety of time series applications. We introduce a novel AR temporal regularizer in Section 4, and make connections to graph-based regularization approaches. We demonstrate the superiority of the proposed approach via extensive experimental results in Section 5 and conclude the paper in Section 6.

2 Motivations: Existing Approaches and Limitations

2.1 Classical Time-Series Models

Models such as AR and DLM are not suitable for modern multiple high-dimensional time series data (i.e., both n and T are large) due to their inherent computational inefficiency (see Section 1). To avoid overfitting in AR models, there have been studies with various structured transition matrices such as low rank and sparse matrices [5, 10, 11]. The focus of this research has been on obtaining better statistical guarantees. The scalability issue of AR models remains open. On the other hand, it is also challenging for many classic time-series models to deal with data that has many missing values [1].

In many situations where the model parameters are either given or designed by practitioners, the Kalman filter approach is used to perform forecasting, while the Kalman smoothing approach is used to impute missing entries. When model parameters are unknown, EM algorithms are applied to estimate both the model parameters and latent embeddings for DLM [3, 8, 9, 17, 19]. As most EM approaches for DLM contain the Kalman filter as a building block, they cannot scale to very high dimensional time series data. Indeed, as shown in Section 5, the popular R package for DLM's does not scale beyond data with tens of dimensions.

2.2 Existing Matrix Factorization Approaches for Data with Temporal Dependencies

In standard MF (1), the squared Frobenius norm $\mathcal{R}_x(X) = \|X\|_F^2 = \sum_{t=1}^{T} \|\boldsymbol{x}_t\|^2$ is usually the regularizer of choice for X. Because squared Frobenius norm assumes no dependencies among $\{\boldsymbol{x}_t\}$, standard MF formulation is *invariant to column permutation* and not applicable to data with temporal dependencies. Hence most existing temporal MF approaches turn to the framework of graph-based regularization [18] for temporally dependent $\{\boldsymbol{x}_t\}$, with a graph encoding the temporal dependencies. An exception is the work in [22], where the authors use specially designed regularizers to encourage a log-normal structure on the temporal coefficients.

Graph regularization for temporal dependencies: The framework of graph-based regularization is an approach to describe and incorporate general dependencies among variables. Let G be a graph over $\{\boldsymbol{x}_t\}$ and G_{ts} be the edge weight between the t-th node and s-th node. A popular regularizer to include as part of an objective function is the following:

$$\mathcal{R}_x(X) = \mathcal{G}(X \mid G, \eta) := \frac{1}{2} \sum_{t \sim s} G_{ts} \|\boldsymbol{x}_t - \boldsymbol{x}_s\|^2 + \frac{\eta}{2} \sum_{t} \|\boldsymbol{x}_t\|^2, \tag{2}$$

where $t \sim s$ denotes an edge between t-th node and s-th node, and the second summation term is used to guarantee strong convexity. A large G_{ts} will ensure that $\boldsymbol{x}_t$ and $\boldsymbol{x}_s$ are close to each other in Euclidean distance, when (2) is minimized. Note that to guarantee the convexity of $\mathcal{G}(X \mid G, \eta)$, we need $G_{ts} \geq 0$.

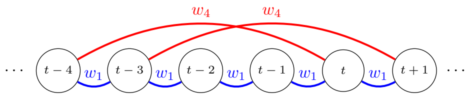

Figure 2: Graph-based regularization for temporal dependencies.

To apply graph-based regularizers to *temporal dependencies*, we need to specify the (repeating) dependency pattern by a lag set $\mathcal{L}$ and a weight vector $\boldsymbol{w}$ such that all the edges $t \sim s$ of distance l (i.e., $|s - t| = l$) share the same weight $G_{ts} = w_l$. See Figure 2 for an example with $\mathcal{L} = \{1, 4\}$. Given $\mathcal{L}$ and $\boldsymbol{w}$, the corresponding graph regularizer becomes

$$\mathcal{G}(X \mid G, \eta) = \frac{1}{2} \sum_{l \in \mathcal{L}} \sum_{t : t > l} w_l (\boldsymbol{x}_t - \boldsymbol{x}_{t-l})^2 + \frac{\eta}{2} \sum_{t} \|\boldsymbol{x}_t\|^2. \tag{3}$$

This direct use of graph-based approach, while intuitive, has two issues: **a)** there might be negatively correlated dependencies between two time points; **b)** unlike many applications where such regularizers are used, the explicit temporal dependency structure is usually not available and has to be inferred. As a result, most existing approaches consider only very simple temporal dependencies such as a small size of $\mathcal{L}$ (e.g., $\mathcal{L} = \{1\}$) and/or uniform weights (e.g., $w_l = 1, \ \forall l \in \mathcal{L}$). For example, a simple chain graph is considered to design the smoothing regularizer in TCF [23]. This leads to poor forecasting abilities of existing MF methods for large-scale time series applications.

2.3 Challenges to Learn Temporal Dependencies

One could try to learn the weights w_l automatically, by using the same regularizer as in (3) but with the weights unknown. This would lead to the following optimization problem:

$$\min_{F, X, \boldsymbol{w} \geq \boldsymbol{0}} \sum_{(i,t) \in \Omega} \left(Y_{it} - \boldsymbol{f}_i^\top \boldsymbol{x}_t \right)^2 + \lambda_f \mathcal{R}_f(F) + \frac{\lambda_x}{2} \sum_{l \in \mathcal{L}} \sum_{t : t - l > 0} w_l (\boldsymbol{x}_t - \boldsymbol{x}_{t-l})^2 + \frac{\lambda_x \eta}{2} \sum_{t} \|\boldsymbol{x}_t\|^2, \tag{4}$$

where $\boldsymbol{0}$ is the zero vector, and $\boldsymbol{w} \geq \boldsymbol{0}$ is the constraint imposed by graph regularization.

It is not hard to see that the above optimization yields the trivial all-zero solution for $\boldsymbol{w}^*$, meaning the objective function is minimized when no temporal dependencies exist! To avoid the all zero solution, one might want to impose a simplex constraint on $\boldsymbol{w}$ (i.e., $\sum_{l\in\mathcal{L}} w_l = 1$). Again, it is not hard to see that this will result in $\boldsymbol{w}^*$ being a 1-sparse vector, with w_{l^*} being 1, where $l^* = \arg\min_{l\in\mathcal{L}} \sum_{t:t>l} \|\boldsymbol{x}_t - \boldsymbol{x}_{t-l}\|^2$. Thus, looking to learn the weights automatically by simply plugging in the regularizer in the MF formulation is not a viable option.

3 Temporal Regularized Matrix Factorization

In order to resolve the limitations mentioned in Sections 2.2 and 2.3, we propose the **T**emporal **R**egularized **M**atrix **F**actorization (TRMF) framework, which is a novel approach to incorporate temporal dependencies into matrix factorization models. Unlike the aforementioned graph-based approaches, we propose to use well-studied time series models to describe temporal dependencies among $\{\boldsymbol{x}_t\}$ explicitly. Such models take the form:

$$\boldsymbol{x}_t = M_\Theta(\{\boldsymbol{x}_{t-l} : l \in \mathcal{L}\}) + \boldsymbol{\epsilon}_t, \tag{5}$$

where $\boldsymbol{\epsilon}_t$ is a Gaussian noise vector, and M_Θ is the time-series model parameterized by $\mathcal{L}$ and Θ. $\mathcal{L}$ is a set containing the lag indices l, denoting a dependency between t-th and $(t-l)$-th time points, while Θ captures the weighting information of temporal dependencies (such as the transition matrix in AR models). To incorporate the temporal dependency into the standard MF formulation (1), we propose to design a new regularizer $\mathcal{T}_{\mathrm{M}}(X \mid \Theta)$ which encourages the structure induced by M_Θ.

Taking a standard approach to model time series, we set $\mathcal{T}_{\mathrm{M}}(X \mid \Theta)$ be the negative log likelihood of observing a particular realization of the $\{\boldsymbol{x}_t\}$ for a given model M_Θ:

$$\mathcal{T}_{\mathrm{M}}(X \mid \Theta) = -\log \mathbb{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T \mid \Theta). \tag{6}$$

When Θ is given, we can use $\mathcal{R}_x(X) = \mathcal{T}_{\mathrm{M}}(X \mid \Theta)$ in the MF formulation (1) to encourage $\{\boldsymbol{x}_t\}$ to follow the temporal dependency induced by M_Θ. When the Θ is unknown, we can treat Θ as another set of variables and include another regularizer $\mathcal{R}_\theta(\Theta)$ into (1):

$$\min_{F,X,\Theta} \sum_{(i,t)\in\Omega} \left(Y_{it} - \boldsymbol{f}_i^\top \boldsymbol{x}_t\right)^2 + \lambda_f \mathcal{R}_f(F) + \lambda_x \mathcal{T}_{\mathrm{M}}(X \mid \Theta) + \lambda_\theta \mathcal{R}_\theta(\Theta), \tag{7}$$

which be solved by an alternating minimization procedure over F, X, and Θ.

Data-driven Temporal Dependency Learning in TRMF: Recall that in Section 2.3, we showed that directly using graph based regularizers to incorporate temporal dependencies leads to trivial solutions for the weights. TRMF circumvents this issue. When F and X are fixed, (7) is reduced to:

$$\min_{\Theta} \quad \lambda_x \mathcal{T}_{\mathrm{M}}(X \mid \Theta) + \lambda_\theta \mathcal{R}_\theta(\Theta), \tag{8}$$

which is a maximum-a-posterior (MAP) estimation problem (in the Bayesian sense) to estimate the best Θ for a given $\{\boldsymbol{x}_t\}$ under the M_Θ model. There are well-developed algorithms to solve (8) and obtain non-trivial Θ. Thus, unlike most existing temporal matrix factorization approaches where the strength of dependencies is fixed, Θ in TRMF can be learned automatically from data.

Time Series Analysis with TRMF: We can see that TRMF (7) lends itself seamlessly to handle a variety of commonly encountered tasks in analyzing data with temporal dependency:

- **Time-series Forecasting:** Once we have M_Θ for latent embeddings $\{\boldsymbol{x}_t : 1, \ldots, T\}$, we can use it to predict future latent embeddings $\{\boldsymbol{x}_t : t > T\}$ and have the ability to obtain non-trivial forecasting results for $\boldsymbol{y}_t = F\boldsymbol{x}_t$ for $t > T$.
- **Missing-value Imputation:** In some time-series applications, some entries in Y might be unobserved, for example, due to faulty sensors in electricity usage monitoring or occlusions in the case of motion recognition in video. We can use $\boldsymbol{f}_i^\top \boldsymbol{x}_t$ to impute these missing entries, much like standard matrix completion, and is useful in recommender systems [23] and sensor networks [26].

Extensions to Incorporate Extra Information: Like matrix factorization, TRMF (7) can be extended to incorporate additional information. For example, pairwise relationships between the time series can be incorporated using structural regularizers on F. Furthermore, when features are known for the time series, we can make use of interaction models such as those in [6, 24, 25]. Also, TRMF can be extended to tensors. More details on these extensions can be found in Appendix B.

4 A Novel Autoregressive Temporal Regularizer

In Section 3, we described the TRMF framework in a very general sense, with the regularizer $\mathcal{T}_{\mathrm{M}}(X \mid \Theta)$ incorporating dependencies specified by the time series model M_Θ. In this section, we specialize this to the case of AR models, which are parameterized by a lag set $\mathcal{L}$ and weights $\mathcal{W} = \left\{W^{(l)} \in \mathbb{R}^{k\times k} : l \in \mathcal{L}\right\}$. Assume that $\boldsymbol{x}_t$ is a noisy linear combination of some previous points; that is, $\boldsymbol{x}_t = \sum_{l\in\mathcal{L}} W^{(l)}\boldsymbol{x}_{t-l} + \boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t$ is a Gaussian noise vector. For simplicity, we assume that the $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \sigma^2 I_k)$, where I_k is the $k \times k$ identity matrix[1]. The temporal regularizer $\mathcal{T}_{\mathrm{M}}(X \mid \Theta)$ corresponding to this AR model can be written as:

$$\mathcal{T}_{\mathrm{AR}}(X \,|\mathcal{L}, \mathcal{W}, \eta) := \frac{1}{2}\sum_{t=m}^{T}\left\| \boldsymbol{x}_t - \sum_{l\in\mathcal{L}} W^{(l)}\boldsymbol{x}_{t-l}\right\|^2 + \frac{\eta}{2}\sum_t \|\boldsymbol{x}_t\|^2, \tag{9}$$

where $m := 1 + L$, $L := \max(\mathcal{L})$, and $\eta > 0$ to guarantee the strong convexity of (9).

TRMF allows us to learn the weights $\{W^{(l)}\}$ when they are unknown. Since each $W^{(l)} \in \mathbb{R}^{k\times k}$, there will be $|\mathcal{L}|k^2$ variables to learn, which may lead to overfitting. To prevent this and to yield more interpretable results, we consider diagonal $W^{(l)}$, reducing the number of parameters to $|\mathcal{L}|k$. To simplify notation, we use $\mathcal{W}$ to denote the $k \times L$ matrix where the l-th column constitutes the diagonal elements of $W^{(l)}$. Note that for $l \notin \mathcal{L}$, the l-th column of $\mathcal{W}$ is a zero vector. Let $\bar{\boldsymbol{x}}_r^\top = [\cdots, X_{rt}, \cdots]$ be the r-th row of X and $\bar{\boldsymbol{w}}_r^\top = [\cdots, \mathcal{W}_{rl}, \cdots]$ be the r-th row of $\mathcal{W}$. Then (9) can be written as $\mathcal{T}_{\mathrm{AR}}(X \,|\mathcal{L}, \mathcal{W}, \eta) = \sum_{r=1}^{k} \mathcal{T}_{\mathrm{AR}}(\bar{\boldsymbol{x}}_r \,|\mathcal{L}, \bar{\boldsymbol{w}}_r, \eta)$, where we define

$$\mathcal{T}_{\mathrm{AR}}(\bar{\boldsymbol{x}} \,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta) = \frac{1}{2}\sum_{t=m}^{T}\left(x_t - \sum_{l\in\mathcal{L}} w_l x_{t-l}\right)^2 + \frac{\eta}{2}\|\bar{\boldsymbol{x}}\|^2, \tag{10}$$

with x_t being the t-th element of $\bar{\boldsymbol{x}}$, and w_l being the l-th element of $\bar{\boldsymbol{w}}$.

Correlations among Multiple Time Series. Even when $\{W^l\}$ is diagonal, TRMF retains the power to capture the correlations among time series via the factors $\{\boldsymbol{f}_i\}$, since it has an effect only on the structure of latent embeddings $\{\boldsymbol{x}_t\}$. Indeed, as the i-th dimension of $\{\boldsymbol{y}_t\}$ is modeled by $\boldsymbol{f}_i^\top X$ in (7), the low rank F is a k dimensional latent embedding of multiple time series. This embedding captures correlations among multiple time series. Furthermore, $\{\boldsymbol{f}_i\}$ acts as time series features, which can be used to perform classification/clustering even in the presence of missing values.

Choice of Lag Index Set $\mathcal{L}$. Unlike most approaches mentioned in Section 2.2, the choice of $\mathcal{L}$ in TRMF is more flexible. Thus, TRMF can provide important advantages: First, because there is no need to specify the weight parameters $\mathcal{W}$, $\mathcal{L}$ can be chosen to be larger to account for long range dependencies, which also yields more accurate and robust forecasts. Second, the indices in $\mathcal{L}$ can be discontinuous so that one can easily embed *domain knowledge* about periodicity or seasonality. For example, one might consider $\mathcal{L} = \{1, 2, 3, 51, 52, 53\}$ for weekly data with a one year seasonality.

Connections to Graph Regularization. We now establish connections between $\mathcal{T}_{\mathrm{AR}}(\bar{\boldsymbol{x}} \,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta)$ and graph regularization (2) for matrix factorization. Let $\bar{\mathcal{L}} := \mathcal{L} \cup \{0\}$, $w_0 = -1$ so that (10) is

$$\mathcal{T}_{\mathrm{AR}}(\bar{\boldsymbol{x}} \,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta) = \frac{1}{2}\sum_{t=m}^{T}\left(\sum_{l\in\bar{\mathcal{L}}} w_l x_{t-l}\right)^2 + \frac{\eta}{2}\|\bar{\boldsymbol{x}}\|^2,$$

and let $\delta(d) := \left\{ l \in \bar{\mathcal{L}} : l - d \in \bar{\mathcal{L}} \right\}$. We then have the following result:

Theorem 1. *Given a lag index set $\mathcal{L}$, weight vector $\bar{\boldsymbol{w}} \in \mathbb{R}^L$, and $\bar{\boldsymbol{x}} \in \mathbb{R}^T$, there is a weighted signed graph G^{AR} with T nodes and a diagonal matrix $D \in \mathbb{R}^{T\times T}$ such that*

$$\mathcal{T}_{AR}(\bar{\boldsymbol{x}} \,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta) = \mathcal{G}\big(\bar{\boldsymbol{x}} \mid G^{AR}, \eta\big) + \frac{1}{2}\bar{\boldsymbol{x}}^\top D \bar{\boldsymbol{x}}, \tag{11}$$

where $\mathcal{G}\big(\bar{\boldsymbol{x}} \mid G^{AR}, \eta\big)$ is the graph regularization (2) *with $G = G^{AR}$. Furthermore, $\forall t$ and d*

$$G^{AR}_{t,t+d} = \begin{cases} \displaystyle\sum_{l\in\delta(d)} \sum_{m\le t+l\le T} -w_l w_{l-d} & \text{if } \delta(d) \neq \phi, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and } D_{tt} = \left(\sum_{l\in\bar{\mathcal{L}}} w_l\right)\left(\sum_{l\in\bar{\mathcal{L}}} w_l [m \le t + l \le T]\right)$$

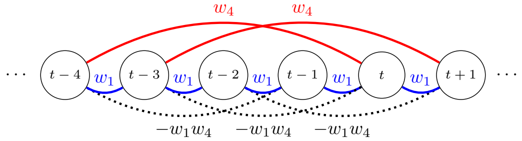

Figure 3: The graph structure induced by the AR temporal regularizer (10) with $\mathcal{L} = \{1, 4\}$.

See Appendix C.1 for a detailed proof. From Theorem 1, we see that $\delta(d)$ is non-empty if and only if there are edges between time points separated by d in G^{AR}. Thus, we can construct the dependency graph for $\mathcal{T}_{\mathrm{AR}}(\bar{\boldsymbol{x}} \,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta)$ by checking whether $\delta(d)$ is empty. Figure 3 demonstrates an example with $\mathcal{L} = \{1, 4\}$. We can see that besides edges of distance $d = 1$ and $d = 4$, there are also edges of distance $d = 3$ (dotted edges in Figure 3) because $4 - 3 \in \bar{\mathcal{L}}$ and $\delta(3) = \{4\}$.

[1] If the (known) covariance matrix is not identity, we can suitably modify the regularizer.

Table 1: Data statistics.

	synthetic	electricity	traffic	walmart-1	walmart-2
n	16	370	963	1,350	1,582
T	128	26,304	10,560	187	187
missing ratio	0%	0%	0%	55.3%	49.3%

Although Theorem 1 shows that AR-based regularizers are similar to the graph-based regularization framework, we note the following key differences:

- The graph G^{AR} in Theorem 1 contains ***both*** positive and negative edges. This implies that the AR temporal regularizer is able to support negative correlations, which the standard graph-based regularizer cannot. This can make $\mathcal{G}\big(\bar{\boldsymbol{x}} \mid G^{\text{AR}}, \eta\big)$ non-convex. The addition of the second term in (11), however, still leads to a convex regularizer $\mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}\,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta)$.
- Unlike (3) where there is freedom to specify a weight for each distance, in the graph G^{AR}, the weight values for the edges are ***more structured*** (e.g., the weight for $d = 3$ in Figure 3 is $-w_1 w_4$). Hence, minimization w.r.t. $w's$ is not trivial, and neither are the obtained solutions.

Plugging $\mathcal{T}_{\text{M}}(X \mid \Theta) = \mathcal{T}_{\text{AR}}(X\,|\mathcal{L}, \mathcal{W}, \eta)$ into (7), we obtain the following problem:

$$\min_{F,X,\mathcal{W}} \sum_{(i,t)\in\Omega} \big(Y_{it} - \boldsymbol{f}_i^\top \boldsymbol{x}_t\big)^2 + \lambda_f \mathcal{R}_f(F) + \sum_{r=1}^{k} \lambda_x \mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}_r\,|\mathcal{L}, \bar{\boldsymbol{w}}_r, \eta) + \lambda_w \mathcal{R}_w(\mathcal{W}), \tag{12}$$

where $\mathcal{R}_w(\mathcal{W})$ is a regularizer for $\mathcal{W}$. We will refer to (12) as TRMF-AR. We can apply alternating minimization to solve (12). In fact, solving for each variable reduces to well known methods, for which highly efficient algorithms exist:

Updates for F. When X and $\mathcal{W}$ are fixed, the subproblem of updating F is the same as updating F while X fixed in (1). Thus, fast algorithms such as alternating least squares or coordinate descent can be applied directly to find F, which costs $O(|\Omega|k^2)$ time.

Updates for X. We solve $\arg\min_X \sum_{(i,t)\in\Omega}\big(Y_{it} - \boldsymbol{f}_i^\top \boldsymbol{x}_t\big)^2 + \lambda_x \sum_{r=1}^{k} \mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}_r\,|\mathcal{L}, \bar{\boldsymbol{w}}_r, \eta)$. From Theorem 1, we see that $\mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}\,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta)$ shares the same form as the graph regularizer, and we can apply GRALS [15] to find X, which costs $O(|\mathcal{L}|Tk^2)$ time.

Updates for $\mathcal{W}$. How to update $\mathcal{W}$ while F and X fixed depends on the choice of $\mathcal{R}_w(\mathcal{W})$. There are many parameter estimation techniques developed for AR with various regularizers [11, 20]. For simplicity, we consider the squared Frobenius norm: $\mathcal{R}_w(\mathcal{W}) = \|\mathcal{W}\|_F^2$. As a result, each row of $\bar{\boldsymbol{w}}_r$ of $\mathcal{W}$ can be updated by solving the following one-dimensional autoregressive problem.

$$\arg\min_{\bar{\boldsymbol{w}}} \; \lambda_x \mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}_r\,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta) + \lambda_w \|\bar{\boldsymbol{w}}\|^2 \equiv \arg\min_{\bar{\boldsymbol{w}}} \sum_{t=m}^{T} \left(x_t - \sum_{l\in\mathcal{L}} w_l x_{t-l} \right)^2 + \frac{\lambda_w}{\lambda_x} \|\bar{\boldsymbol{w}}\|^2,$$

which is a simple $|\mathcal{L}|$ dimensional ridge regression problem with $T - m + 1$ instances, which can be solved efficiently by Cholesky factorization in $O(|\mathcal{L}|^3 + T|\mathcal{L}|^2)$ time

Note that since our method is highly modular, one can resort to *any* method to solve the optimization subproblems that arise for each module. Moreover, as mentioned in Section 3, TRMF can also be used with different regularization structures, making it highly adaptable.

4.1 Connections to Existing MF Approaches

TRMF-AR is a generalization of many existing MF approaches to handle data with temporal dependencies. Specifically, Temporal Collaborative Filtering [23] corresponds to $W^{(1)} = I_k$ on $\{\boldsymbol{x}_t\}$. The NMF method of [2] is an AR(L) model with $W^{(l)} = \alpha^{l-1}(1-\alpha)I_k$, $\forall l$, where α is pre-defined. The AR(1) model of [16, 26] has $W^{(1)} = I_n$ on $\{F\boldsymbol{x}_t\}$. Finally the DLM [7] is a latent AR(1) model with a general $W^{(1)}$, which can be estimated by EM algorithms.

4.2 Connections to Learning Gaussian Markov Random Fields

The Gaussian Markov Random Field (GMRF) is a general way to model multivariate data with dependencies. GMRF assumes that data are generated from a multivariate Gaussian distribution with a covariance matrix Σ which describes the dependencies among T dimensional variables i.e., $\bar{\boldsymbol{x}} \sim \mathcal{N}(0, \Sigma)$. If the unknown $\bar{\boldsymbol{x}}$ is assumed to be generated from this model, The negative log likelihood of the data can be written as $\bar{\boldsymbol{x}}^\top \Sigma^{-1} \bar{\boldsymbol{x}}$, ignoring the constants and where Σ^{-1} is the inverse covariance matrix of the Gaussian distribution. This prior can be incorporated into an empirical risk minimization framework as a regularizer. Furthermore, it is known that if $\left(\Sigma^{-1}\right)_{st} = 0$, x_t and x_s are conditionally independent, given the other variables. In Theorem 1 we established connections

Table 2: Forecasting results: ND/ NRMSE for each approach. Lower values are better. “-” indicates an unavailability due to scalability or an inability to handle missing values.

	Forecasting with Full Observation						
	Matrix Factorization Models			Time Series Models			
	TRMF-AR	SVD-AR(1)	TCF	AR(1)	DLM	R-DLM	Mean
synthetic	**0.373**/ **0.487**	0.444/ 0.872	1.000/ 1.424	0.928/ 1.401	0.936/ 1.391	0.996/ 1.420	1.000/ 1.424
electricity	0.255/ **1.397**	0.257/ 1.865	0.349/ 1.838	**0.219**/ 1.439	0.435/ 2.753	-/ -	1.410/ 4.528
traffic	**0.187**/ **0.423**	0.555/ 1.194	0.624/ 0.931	0.275/ 0.536	0.639/ 0.951	-/ -	0.560/ 0.826
	Forecasting with Missing Values						
walmart-1	**0.533**/ **1.958**	-/ -	0.540/2.231	-/ -	0.602/ 2.293	-/ -	1.239/3.103
walmart-2	**0.432**/ **1.065**	-/ -	0.446/1.124	-/ -	0.453/ 1.110	-/ -	1.097/2.088

to graph based regularizers, and that such methods can be seen as regularizing with the inverse covariance matrix for Gaussians [27]. We thus have the following result:

Corollary 1. *For any lag set $\mathcal{L}$, $\bar{\boldsymbol{w}}$, and $\eta > 0$, the inverse covariance matrix Σ_{AR}^{-1} of the GMRF model corresponding to the quadratic regularizer $\mathcal{R}_x(\bar{\boldsymbol{x}}) := \mathcal{T}_{AR}(\bar{\boldsymbol{x}}\,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta)$ shares the same off-diagonal non-zero pattern as G^{AR} defined in Theorem 1. Moreover, we have $\mathcal{T}_{AR}(\bar{\boldsymbol{x}}\,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta) = \bar{\boldsymbol{x}}^\top \Sigma_{AR}^{-1} \bar{\boldsymbol{x}}$.*

A detailed proof is in Appendix C.2. As a result, our proposed AR-based regularizer is equivalent to imposing a Gaussian prior on $\bar{\boldsymbol{x}}$ with a structured inverse covariance described by the matrix G^{AR} defined in Theorem 1. Moreover, the step to learn $\mathcal{W}$ has a natural interpretation: the lag set $\mathcal{L}$ imposes the non-zero pattern of the graphical model on the data, and then we solve a simple least squares problem to learn the weights corresponding to the edges. As an application of Theorem 1 from [15] and Corollary 1, when $\mathcal{R}_f(F) = \|F\|_F^2$,we can relate $\mathcal{T}_{\text{AR}}$ to a weighted nuclear norm:

$$\|ZB\|_* = \frac{1}{2} \inf_{F,X:Z=FX} \|F\|_F^2 + \sum_r \mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}_r\,|\mathcal{L}, \bar{\boldsymbol{w}}, \eta), \tag{13}$$

where $B = US^{1/2}$ and $\Sigma_{\text{AR}}^{-1} = USU^\top$ is the eigen-decomposition of Σ_{AR}^{-1}. (13) enables us to apply the results from [15] to obtain guarantees for the use of AR temporal regularizer when $\mathcal{W}$ is given. For simplicity, we assume $\bar{\boldsymbol{w}}_r = \bar{\boldsymbol{w}}$, $\forall r$ and consider a relaxed convex formulation for (12) as follows:

$$\hat{Z} = \arg\min_{Z \in \mathcal{C}} \frac{1}{N} \sum_{(i,j)\in\Omega} (Y_{ij} - Z_{ij})^2 + \lambda_z \|ZB\|_*, \tag{14}$$

where $N = |\Omega|$, and $\mathcal{C}$ is a set of matrices with low *spikiness*. Full details are provided in Appendix C.3. As an application of Theorem 2 from [15], we have the following corollary.

Corollary 2. *Let $Z^\star = FX$ be the ground truth $n \times T$ time series matrix of rank k. Let Y be the matrix with $N = |\Omega|$ randomly observed entries corrupted with additive Gaussian noise with variance σ^2. Then if $\lambda_z \geq C_1 \sqrt{\frac{(n+T)\log(n+T)}{N}}$, with high probability for the $\hat{Z}$ obtained by* (14),

$$\left\| Z^\star - \hat{Z} \right\|_F \leq C_2 \alpha^2 \max(1, \sigma^2) \frac{k(n+T)\log(n+T)}{N} + O(\alpha^2/N),$$

where C_1,C_2 are positive constants, and α depends on the product $Z^\star B$.

See Appendix C.3 for details. From the results in Table 3, we observe superior performance of TRMF-AR over standard MF, indicating that $\bar{\boldsymbol{w}}$ learnt from our data-driven approach (12) does aid in recovering the missing entries for time series. We would like to point out that establishing a theoretical guarantee for TRMF with $\mathcal{W}$ is unknown remains a challenging research direction.

5 Experimental Results

We consider five datasets (Table 1). For synthetic, we first randomly generate $F \in \mathbb{R}^{16\times 4}$ and generate $\{\boldsymbol{x}_t\}$ following an AR process with $\mathcal{L} = \{1, 8\}$. Then Y is obtained by $\boldsymbol{y}_t = F\boldsymbol{x}_t + \boldsymbol{\epsilon}_t$ where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, 0.1I)$. The data sets electricity and traffic are obtained from the UCI repository, while walmart-1 and walmart-2 are two propriety datasets from Walmart E-commerce containing weekly sale information. Due to reasons such as out-of-stock, 55.3% and 49.3% of entries are missing respectively. To evaluate the prediction performance, we consider the normalized deviation (ND) and normalized RMSE (NRMSE). See details for the description for each dataset and the formal definition for each criterion in Appendix A.

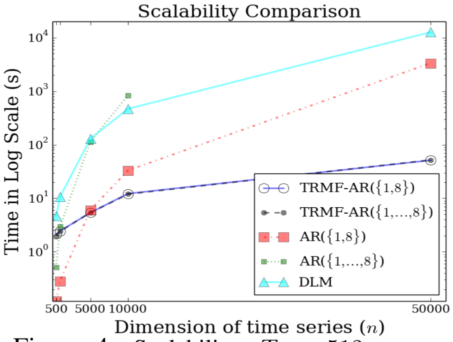

Figure 4: Scalability: $T = 512$. $n \in \{500, 1000, \ldots, 50000\}$. AR($\{1, \ldots, 8\}$) cannot finish in 1 day.

Table 3: Missing value imputation results: ND/ NRMSE for each approach. Note that TRMF outperforms all competing methods in almost all cases.

	$\frac{\|\Omega\|}{n\times T}$	Matrix Factorization Models			Time Series Models	
		TRMF-AR	TCF	MF	DLM	Mean
synthetic	20%	**0.467/ 0.661**	0.713/ 1.030	0.688/ 1.064	0.933/ 1.382	1.002/ 1.474
	30%	**0.336/ 0.455**	0.629/ 0.961	0.595/ 0.926	0.913/ 1.324	1.004/ 1.445
	40%	**0.231/ 0.306**	0.495/ 0.771	0.374/ 0.548	0.834/ 1.259	1.002/ 1.479
	50%	**0.201/ 0.270**	0.289/ 0.464	0.317/ 0.477	0.772/ 1.186	1.001/ 1.498
electricity	20%	**0.245/ 2.395**	0.255/ 2.427	0.362/ 2.903	0.462/ 4.777	1.333/ 6.031
	30%	**0.235/ 2.415**	0.245/ 2.436	0.355/ 2.766	0.410/ 6.605	1.320/ 6.050
	40%	0.231/ 2.429	0.242/ 2.457	0.348/ 2.697	**0.196/ 2.151**	1.322/ 6.030
	50%	0.223/ 2.434	0.233/ 2.459	0.319/ 2.623	**0.158/ 1.590**	1.320/ 6.109
traffic	20%	**0.190/ 0.427**	0.208/ 0.448	0.310/ 0.604	0.353/ 0.603	0.578/ 0.857
	30%	**0.186/ 0.419**	0.199/ 0.432	0.299/ 0.581	0.286/ 0.518	0.578/ 0.856
	40%	**0.185/ 0.416**	0.198/ 0.428	0.292/ 0.568	0.251/ 0.476	0.578/ 0.857
	50%	**0.184/ 0.415**	0.193/ 0.422	0.251/ 0.510	0.224/ 0.447	0.578/ 0.857

Methods/Implementations Compared:

- TRMF-AR: The proposed formulation (12) with $\mathcal{R}_w(\mathcal{W}) = \|\mathcal{W}\|_F^2$. For $\mathcal{L}$, we use $\{1, 2, \ldots, 8\}$ for synthetic, $\{1, \ldots, 24\} \cup \{7 \times 24, \ldots, 8 \times 24 - 1\}$ for electricity and traffic, and $\{1, \ldots, 10\} \cup \{50, \ldots, 56\}$ for walmart-1 and walmart-2 to capture seasonality.
- SVD-AR(1): The rank-k approximation of $Y = USV^\top$ is first obtained by SVD. After setting $F = US$ and $X = V^\top$, a k-dimensional AR(1) is learned on X for forecasting.
- TCF: Matrix factorization with the simple temporal regularizer proposed in [23].
- AR(1): n-dimensional AR(1) model.[2]
- DLM: two implementations: the widely used R-DLM package [12] and the code provided in [8].
- Mean: The baseline, which predicts everything to be the mean of the observed portion of Y.

For each method and data set, we perform a grid search over various parameters (such as k, λ values) following a rolling validation approach described in [11].

Scalability: Figure 4 shows that traditional time-series approaches such as AR or DLM suffer from the scalability issue for large n, while TRMF-AR scales much better with n. Specifically, for $n = 50,000$, TRMF is 2 orders of magnitude faster than competing AR/DLM methods. Note that the results for R-DLM are not available because the R package cannot scale beyond n in the tens (See Appendix D for more details.). Furthermore, the `dlmMLE` routine in R-DLM uses a general optimization solver, which is orders of magnitude slower than the implementation provided in [8].

5.1 Forecasting

Forecasting with Full Observations. We first compare various methods on the task of forecasting values in the test set, given fully observed training data. For synthetic, we consider one-point ahead forecasting task and use the last ten time points as the test periods. For electricity and traffic, we consider the 24-hour ahead forecasting task and use last seven days as the test periods. From Table 2, we can see that TRMF-AR outperforms all the other methods on both metrics considered.

Forecasting with Missing Values. We next compare the methods on the task of forecasting in the presence of missing values in the data. We use the Walmart datasets here, and consider 6-week ahead forecasting and use last 54 weeks as the test periods. Note that SVD-AR(1) and AR(1) cannot handle missing values. The second part of Table 2 shows that we again outperform other methods.

5.2 Missing Value Imputation

We next consider the case of imputing missing values in the data. As in [9], we assume that blocks of data are missing, corresponding to sensor malfunctions for example, over a length of time. To create data with missing entries, we first fixed the percentage of data that we were interested in observing, and then uniformly at random occluded blocks of a predetermined length (2 for synthetic data and 5 for the real datasets). The goal was to predict the occluded values. Table 3 shows that TRMF outperforms the methods we compared to on almost all cases.

6 Conclusions

We propose a novel temporal regularized matrix factorization framework (TRMF) for high-dimensional time series problems with missing values. TRMF not only models temporal dependency among the data points, but also supports data-driven dependency learning. TRMF generalizes several well-known methods and yields superior performance when compared to other state-of-the-art methods on real-world datasets.

Acknowledgements: This research was supported by NSF grants (CCF-1320746, IIS-1546459, and CCF-1564000) and gifts from Walmart Labs and Adobe. We thank Abhay Jha for the help on Walmart experiments.

[2] In Appendix A, we also show a baseline which applies an independent AR model to each dimension.

References

[1] O. Anava, E. Hazan, and A. Zeevi. Online time series prediction with missing data. In *Proceedings of the International Conference on Machine Learning*, pages 2191–2199, 2015.

[2] Z. Chen and A. Cichocki. Nonnegative matrix factorization with temporal smoothness and/or spatial decorrelation constraints. *Laboratory for Advanced Brain Signal Processing, RIKEN, Tech. Rep*, 68, 2005.

[3] Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. Technical report, Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science, 1996.

[4] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., 2nd edition, 1994.

[5] F. Han and H. Liu. Transition matrix estimation in high dimensional time series. In *Proceedings of the International Conference on Machine Learning*, pages 172–180, 2013.

[6] P. Jain and I. S. Dhillon. Provable inductive matrix completion. *arXiv preprint arXiv:1306.0626*, 2013.

[7] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[8] L. Li and B. A. Prakash. Time series clustering: Complex is simpler! In *Proceedings of the International Conference on Machine Learning*, pages 185–192, 2011.

[9] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos. DynaMMo: Mining and summarization of coevolving sequences with missing values. In *ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 507–516. ACM, 2009.

[10] I. Melnyk and A. Banerjee. Estimating structured vector autoregressive model. In *Proceedings of the Thirty Third International Conference on Machine Learning (ICML)*, 2016.

[11] W. B. Nicholson, D. S. Matteson, and J. Bien. Structured regularization for large vector autoregressions. Technical report, Technical Report, University of Cornell, 2014.

[12] G. Petris. An r package for dynamic linear models. *Journal of Statistical Software*, 36(12):1–16, 2010.

[13] G. Petris, S. Petrone, and P. Campagnoli. *Dynamic Linear Models with R*. Use R! Springer, 2009.

[14] S. Rallapalli, L. Qiu, Y. Zhang, and Y.-C. Chen. Exploiting temporal stability and low-rank structure for localization in mobile networks. In *International Conference on Mobile Computing and Networking*, MobiCom '10, pages 161–172. ACM, 2010.

[15] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in Neural Information Processing Systems 27*, 2015.

[16] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and internet traffic matrices (extended version). *IEEE/ACM Transactions on Networking*, 20(3):662–676, June 2012.

[17] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of time series analysis*, 3(4):253–264, 1982.

[18] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer, 2003.

[19] J. Z. Sun, K. R. Varshney, and K. Subbian. Dynamic matrix factorization: A state space approach. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 1897–1900. IEEE, 2012.

[20] H. Wang, G. Li, and C.-L. Tsai. Regression coefficient and autoregressive order shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(1):63–78, 2007.

[21] M. West and J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer Series in Statistics. Springer, 2013.

[22] K. Wilson, B. Raj, and P. Smaragdis. Regularized non-negative matrix factorization with temporal dependencies for speech denoising. In *Interspeech*, pages 411–414, 2008.

[23] L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *SIAM International Conference on Data Mining*, pages 223–234, 2010.

[24] M. Xu, R. Jin, and Z.-H. Zhou. Speedup matrix completion with side information: Application to multi-label learning. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2301–2309, 2013.

[25] H.-F. Yu, P. Jain, P. Kar, and I. S. Dhillon. Large-scale multi-label learning with missing labels. In *Proceedings of the International Conference on Machine Learning*, pages 593–601, 2014.

[26] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and internet traffic matrices. *SIGCOMM Comput. Commun. Rev.*, 39(4):267–278, Aug. 2009. ISSN 0146-4833.

[27] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*, volume 12, pages 403–414. SIAM, 2012.

A More Details about Experiments

A.1 Datasets and Evaluation Criteria

Datasets:

- synthetic: a small synthetic dataset with $n = 16, T = 128$. We generate $\left\{\boldsymbol{x}_t \in \mathbb{R}^4 : t = 1, \ldots, 128\right\}$ from the autoregressive process with a lag index set $\mathcal{L} = \{1, 8\}$, randomly generated $\left\{W^{(l)}\right\}$, and an additive white Gaussian noise of $\sigma = 0.1$. We then randomly generate a matrix $F \in \mathbb{R}^{16\times 4}$ and obtain $\boldsymbol{y}_t = F\boldsymbol{x}_t + \boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, 0.1)$.
- electricity[3]: the electricity usage in kW recorded every 15 minutes, for $n = 370$ clients. We convert the data to reflect hourly consumption, by aggregating blocks of 4 columns, to obtain $T = 26,304$. Teh coefficient of variation for electricity is 6.0341.
- traffic[4]: A collection of 15 months of daily data from the California Department of Transportation. The data describes the occupancy rate, between 0 and 1, of different car lanes of San Francisco bay area freeways. The data was sampled every 10 minutes, and we again aggregate the columns to obtain hourly traffic data to finally get $n = 963$, $T = 10,560$. The coefficient of variation for traffic is 0.8565.
- walmart-1 & walmart-2: two propriety datasets from Walmart E-commerce contain weekly sale information of 1,350 and 1,582 items for 187 weeks, respectively. The time-series of sales for each item start and end at different time points; for modeling purposes we assume one start and end timestamp by padding each series with missing values. This along with some other missing values due to out-of-stock reasons lead to 55.3% and 49.3% of entries being missing.

Evaluation Criteria: We compute the normalized deviation (ND) and normalized RMSE (NRMSE).

$$\text{Normalized deviation (ND):} \quad \left(\frac{1}{|\Omega_{test}|}\sum_{(i,t)\in\Omega_{test}}\left|\hat{Y}_{it} - Y_{it}\right|\right)\Bigg/\left(\frac{1}{|\Omega_{test}|}\sum_{(i,t)\in\Omega_{test}}|Y_{ij}|\right)$$

$$\text{Normalized RMSE (NRMSE):} \quad \sqrt{\frac{1}{|\Omega_{test}|}\sum_{(i,t)\in\Omega_{test}}\left(\hat{Y}_{it} - Y_{it}\right)^2}\Bigg/\left(\frac{1}{|\Omega_{test}|}\sum_{(i,t)\in\Omega_{test}}|Y_{ij}|\right)$$

For each method and data set, we perform the grid search over various parameters (such as k, λ values) following a rolling validation approach described in [11]. We search $k \in \{2, 4, 8\}$ for synthetic and $\in \{20, 40\}$ for other datasets. For TRMF-AR, SVD-AR(1), TCF, and AR(1), we search $\lambda \in \{50, 5, 0.5, 0.05\}$

Results with n independent AR models. One naive way for AR models to solve the scalability issue to handle n-dimension time series is assuming each dimension is independent from each other. Thus, one can apply an one-dimensional AR model to each dimension. Although this approach is embarrassingly parallel, it ignores the correlations among time series that we can take into account. Note that this approach still cannot solve the difficulty of AR models to handle missing values. Here, we include the forecasting results of this independent AR approach in Table 4. For synthetic, we learn an independent AR(1) for each dimension, while for electricity and traffic, we learn an independent AR($\{1, 24\}$) for each dimension. We can see that on synthetic and traffic, TRMF-AR still outperforms this independent AR approach.

Table 4: Forecasting Comparison.

	TRMF-AR	AR(1)	Independent AR	Mean
synthetic	**0.373**/ **0.487**	0.928/ 1.401	0.973/ 1.388	1.000/ 1.424
electricity	0.255/ 1.397	0.219/ 1.439	**0.206**/**1.220**	1.410/ 4.528
traffic	**0.187**/ **0.423**	0.275/ 0.536	0.294/ 0.591	0.560/ 0.826

[3] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014.

[4] https://archive.ics.uci.edu/ml/datasets/PEMS-SF.

B Extensions to Incorporate Extra Information

In the same vein as matrix factorization approaches, TRMF (7) can be extended to incorporate additional information:

- **Known features for time series:** In many applications, one is given additional features along with the observed time series. Specifically, given a set of feature vectors $\{\boldsymbol{a}_i \in \mathbb{R}^d\}$ for each row of Y, we can look to solve
$$\begin{aligned}\min_{F,X,\Theta} \quad & \sum_{(i,t)\in\Omega} \left(Y_{it} - \boldsymbol{a}_i^\top F \boldsymbol{x}_t\right)^2 + \lambda_f \mathcal{R}_f(F) \\ & + \lambda_x \mathcal{T}_{\mathrm{M}}(X \mid \Theta) + \lambda_\theta \mathcal{R}_\theta(\Theta). \end{aligned} \tag{15}$$
That is, the observation Y_{it} is posited to be a bilinear function of the feature vector $\boldsymbol{a}_i$ and the latent vector $\boldsymbol{x}_t$. Such an inductive framework has two advantages: we can generalize TRMF to a new time series without any observations up to time T (i.e., a new row i' of Y without any observations). As long as the feature vector $\boldsymbol{a}_{i'}$ is available, the model learned by TRMF can be used to estimate $Y_{i't} = \boldsymbol{a}_{i'}^\top F \boldsymbol{x}_t,\ \forall t$. Furthermore, prediction can be significantly sped up when $d \ll n$, since the dimension of F is reduced from $n \times k$ to $d \times k$. Such methods for standard multi-label learning and matrix completion have been previously considered in [6, 24, 25].
- **Graph information among time series:** Often, separate features for the time series are not known, but other relational information is available. When a graph that encodes pairwise interactions among multiple time series is available, one can incorporate this graph in our framework using the graph regularization approach (2). Such cases are common in inventory and sales tracking, where sales of one item is related to sales of other items. Given a graph G^f describing the relationship among multiple time series, we can formulate a graph regularized problem:
$$\begin{aligned}\min_{F,X,\Theta} \quad & \sum_{(i,t)\in\Omega} \left(Y_{it} - \boldsymbol{f}_i^\top \boldsymbol{x}_t\right)^2 + \lambda_f \mathcal{G}\left(F \mid G^f, \eta\right) \\ & + \lambda_x \mathcal{T}_{\mathrm{M}}(X \mid \Theta) + \lambda_\theta \mathcal{R}_\theta(\Theta), \end{aligned} \tag{16}$$
where $\mathcal{G}\left(F \mid G^f, \eta\right)$ is the graph regularizer defined in (2) capturing pairwise interactions between time series. Graph regularized matrix completion methods have been previously considered in [15, 27].
- **Temporal-regularized tensor factorization:** Naturally, TRMF can be easily extended to analyze temporal collaborative filtering applications [19, 23], where the targeted data is a tensor with certain modes evolving over time. For example, consider $\mathcal{Y} \in \mathbb{R}^{m\times n\times T}$ be a 3-way tensor with Y_{ijt} encoding the rating of the i-th user for the j-th item at time point t. We can consider the following temporal regularization tensor factorization (TRTF) with $\mathcal{T}_{\mathrm{M}}(X \mid \Theta)$ as follows:
$$\begin{aligned}\min_{P,Q,X,\Theta} \quad & \sum_{(i,j,t)\in\Omega} \left(Y_{ijt} - \langle \boldsymbol{p}_i, \boldsymbol{q}_j, \boldsymbol{x}_t\rangle\right)^2 + \lambda_p \mathcal{R}_p(P) \\ & + \mathcal{R}_q(Q) + \mathcal{T}_{\mathrm{M}}(X \mid \Theta) + \mathcal{R}_\theta(\Theta), \end{aligned} \tag{17}$$
where $P = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_m]^\top \in \mathbb{R}^{m\times k}$ and $Q = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_n]^\top \in \mathbb{R}^{n\times k}$ are the latent embeddings for the m users and n items, respectively, and with some abuse of notation, we define $\langle \boldsymbol{p}_i, \boldsymbol{q}_j, \boldsymbol{x}_t\rangle = \sum_{r=1}^k p_{ir} q_{jr} x_{tr}$.

C Proofs

C.1 Proof of Theorem 1

Proof. In this proof, we use the notations and summation manipulation techniques introduced by Knuth [4]. To prove (11), it suffices to prove that

$$\sum_{m\le t\le T} \left(\sum_{l\in\bar{\mathcal{L}}} w_l x_{t-l}\right)^2 = \sum_{1\le t\le T} \sum_{1\le d\le L} G^{\mathrm{AR}}_{t,t+d}\left(x_t - x_{t+d}\right)^2 + \bar{\boldsymbol{x}}^\top D \bar{\boldsymbol{x}}. \tag{18}$$

The LHS of the (18) can be expanded and regrouped as follows.

$$
\begin{aligned}
&\sum_{m\le t\le T}\left(\sum_{l\in\bar{\mathcal{L}}} w_l x_{t-l}\right)^2 \\
&= \sum_{m\le t\le T}\left(\sum_{l\in\bar{\mathcal{L}}} w_l^2 x_{t-l}^2 + \sum_{1\le d\le L}\sum_{l\in\delta(d)} 2w_l w_{l-d} x_{t-l} x_{t-l+d}\right) \\
&= \sum_{m\le t\le T}\left(\sum_{l\in\bar{\mathcal{L}}} w_l^2 x_{t-l}^2 + \sum_{1\le d\le L}\sum_{l\in\delta(d)} \Big(-w_l w_{l-d}\big(x_{t-l} - x_{t-l+d}\big)^2 + w_l w_{l-d}\big(x_{t-l}^2 + x_{t-l+d}^2\big)\Big)\right) \\
&= \underbrace{\sum_{m\le t\le T}\sum_{1\le d\le L}\sum_{l\in\delta(d)} -w_l w_{l-d}\big(x_{t-l} - x_{t-l+d}\big)^2}_{\mathcal{G}(\bar{\boldsymbol{x}})} + \underbrace{\sum_{m\le t\le T}\left(\sum_{l\in\bar{\mathcal{L}}} w_l^2 x_{t-l}^2 + \sum_{1\le d\le L}\sum_{l\in\delta(d)} w_l w_{l-d}\big(x_{t-l}^2 + x_{t-l+d}^2\big)\right)}_{\mathcal{D}(\bar{\boldsymbol{x}})}
\end{aligned}
$$

Let's look at the first term $\mathcal{G}(\bar{\boldsymbol{x}})$:

$$
\begin{aligned}
\mathcal{G}(\bar{\boldsymbol{x}}) &= \sum_{1\le d\le L}\sum_{l\in\delta(d)}\sum_{m\le t\le T} -w_l w_{l-d}\big(x_{t-l} - x_{t-l+d}\big)^2 \\
&= \sum_{1\le d\le L}\sum_{l\in\delta(d)}\sum_{m-l\le t\le T-l} -w_l w_{l-d}\big(x_t - x_{t+d}\big)^2 \\
&= \sum_{1\le d\le L}\sum_{l\in\delta(d)}\sum_{1\le t\le T} -w_l w_{l-d}\big(x_t - x_{t+d}\big)^2 [m-l\le t\le T-l] \\
&= \sum_{1\le t\le T}\sum_{1\le d\le L}\left(\sum_{l\in\delta(d)} -w_l w_{l-d}[m-l\le t\le T-l]\right)\big(x_t - x_{t+d}\big)^2 \\
&= \sum_{1\le t\le T}\sum_{1\le d\le L}\underbrace{\left(\sum_{\substack{l\in\delta(d)\\ m\le t+l\le T}} -w_l w_{l-d}\right)}_{G_{t,t+d}}\big(x_t - x_{t+d}\big)^2,
\end{aligned}
$$

where we can see that $\mathcal{G}(\bar{\boldsymbol{x}})$ is equivalent to the first term of RHS of (18).

Now, we consider the second term $\mathcal{D}(\bar{\boldsymbol{x}})$:

$$
\begin{aligned}
\mathcal{D}(\bar{\boldsymbol{x}}) &= \sum_{m\le t\le T}\left(\sum_{l\in\bar{\mathcal{L}}} w_l^2 x_{t-l}^2 + \sum_{1\le d\le L}\sum_{l\in\delta(d)} w_l w_{l-d}\big(x_{t-l}^2 + x_{t-l+d}^2\big)\right) \\
&= \underbrace{\sum_{m\le t\le T}\sum_{l\in\bar{\mathcal{L}}} w_l^2 x_{t-l}^2}_{\mathcal{D}_1(\bar{\boldsymbol{x}})} + \underbrace{\sum_{m\le t\le T}\sum_{1\le d\le L}\sum_{l\in\delta(d)} w_l w_{l-d} x_{t-l}^2}_{\mathcal{D}_2(\bar{\boldsymbol{x}})} + \underbrace{\sum_{m\le t\le T}\sum_{1\le d\le L}\sum_{l\in\delta(d)} w_l w_{l-d} x_{t-l+d}^2}_{\mathcal{D}_3(\bar{\boldsymbol{x}})}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{D}_1(\bar{\boldsymbol{x}}) &= \sum_{l\in\bar{\mathcal{L}}}\sum_{m\le t\le T} w_l^2 x_{t-l}^2 = \sum_{l\in\bar{\mathcal{L}}}\sum_{m-l\le t\le T-l} w_l^2 x_t^2 = \sum_{1\le t\le T}\left(\sum_{l\in\bar{\mathcal{L}}} w_l^2[m\le t+l\le T]\right)x_t^2\\
&= \sum_{1\le t\le T}\left(\sum_{l,l'\in\bar{\mathcal{L}}} w_l w_{l'}[m\le t+l\le T][l'=l]\right)x_t^2\\
\mathcal{D}_2(\bar{\boldsymbol{x}}) &= \sum_{m\le t\le T}\sum_{1\le d\le L}\sum_{l\in\delta(d)} w_l w_{l-d} x_{t-l}^2 = \sum_{1\le t\le T}\left(\sum_{1\le d\le L}\sum_{l\in\delta(d)} w_l w_{l-d}[m\le t+l\le T]\right)x_t^2\\
&= \sum_{1\le t\le T}\left(\sum_{l,l'\in\bar{\mathcal{L}}} w_l w_{l'}[m\le t+l\le T][l'<l]\right)x_t^2\\
\mathcal{D}_3(\bar{\boldsymbol{x}}) &= \sum_{m\le t\le T}\sum_{1\le d\le L}\sum_{l\in\delta(d)} w_l w_{l-d} x_{t-l+d}^2 = \sum_{1\le t\le T}\left(\sum_{1\le d\le L}\sum_{l\in\delta(d)} w_l w_{l-d}[m\le t+l-d\le T]\right)x_t^2\\
&= \sum_{1\le t\le T}\left(\sum_{l',l\in\bar{\mathcal{L}}} w_l w_{l'}[m\le t+l\le T][l'>l]\right)x_t^2
\end{aligned}
$$

Let $D\in R^{T\times T}$ be a diagonal matrix with D_{tt} be the coefficient associated with x_t^2 in $\mathcal{D}(\bar{\boldsymbol{x}})$. Combining the results of $\mathcal{D}_1(\bar{\boldsymbol{x}}),\mathcal{D}_2(\bar{\boldsymbol{x}})$, and $\mathcal{D}_3(\bar{\boldsymbol{x}})$, D_t can be written as follows.

$$
D_{tt} = \left(\sum_{l\in\bar{\mathcal{L}}} w_l\right)\left(\sum_{l\in\bar{\mathcal{L}}} w_l[m\le t+l\le T]\right)\quad \forall t.
$$

It is clear that $\mathcal{D}(\bar{\boldsymbol{x}}) = \bar{\boldsymbol{x}}^\top D\bar{\boldsymbol{x}}$. Note that $\forall t=m,\ldots,T-L$, $D_{tt}=\left(\sum_{l\in\bar{\mathcal{L}}} w_l\right)^2$. □

C.2 Proof of Corollary 1

Proof. It is well known that graph regularization can be written in the quadratic form [18] as follows.

$$
\frac{1}{2}\sum_{t\sim s} G_{ts}(x_t-x_s)^2 = \bar{\boldsymbol{x}}^\top \mathbf{Lap}(G)\bar{\boldsymbol{x}},
$$

where $\mathbf{Lap}(G)$ is the $T\times T$ graph Laplacian for G defined as:

$$
\mathbf{Lap}(G)_{ts} = \begin{cases} \sum_j G_{tj}, & t=s\\ -G_{ts}, & t\ne s \text{ and there is an edge } t\sim s\\ 0, & \text{otherwise.}\end{cases}
$$

Based on the above fact and the results from Theorem 1, we obtain the quadratic form for $\mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}\,|\mathcal{L},\bar{\boldsymbol{w}},\eta)$ as follows.

$$
\mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}\,|\mathcal{L},\bar{\boldsymbol{w}},\eta) = \frac{1}{2}\bar{\boldsymbol{x}}^\top\left(\mathbf{Lap}\big(G^{\text{AR}}\big) + \underbrace{D+\eta I}_{\text{diagonal}}\right)\bar{\boldsymbol{x}}.
$$

Because $D+\eta I$ is diagonal, the non-zero pattern of the off-diagonal entries of the inverse covariance Σ_{AR}^{-1} for $\mathcal{T}_{\text{AR}}(\bar{\boldsymbol{x}}\,|\mathcal{L},\bar{\boldsymbol{w}},\eta)$ is determined by $\mathbf{Lap}\big(G^{\text{AR}}\big)$ which shares the same non-zero pattern as G^{AR}. □

C.3 Details for Corollary 2

We use results developed in [15] to arrive at our result. Assume we are given the matrix B. We first define the following quantities:

$$
\alpha := \sqrt{nT}\frac{\|Z^\star B\|_\infty}{\|Z^\star B\|_F} \qquad \beta := \frac{\|Z^\star B\|_*}{\|Z^\star B\|_F} \tag{19}
$$

where the $\|\cdot\|_\infty$ norm is taken element-wise. Note that, the above quantities capture the "simplicity" of the matrix Z. For example, a small value of α implies that the matrix ZB is not overly spiky, meaning that the entries of the matrix are well spread out in magnitude. Next, define the set

$$\mathcal{C} := \left\{ Z : \alpha\beta \le C\sqrt{\frac{N}{\log(n+T)}} \right\}, \tag{20}$$

with C being a constant that depends on α.

Finally, we assume the following observation model: for each $i, j \in \Omega$, suppose we observe

$$Y_{ij} = Z^{\star}_{ij} + \frac{\sigma}{\sqrt{nT}}\eta_{ij} \quad \eta_{ij} \sim \mathcal{N}(0,1)$$

Then, we can see that the setup is identical to that considered in [15] with the difference being that there is no graph present that relates the rows of $Z^{\star}$. Hence, setting $A = I$ in Theorem 1 in the aforementioned paper yields our result.

D Details: Scalability Issue of R-DLM package

In this section, we show the source code demonstrating that R-DLM fails to handle high-dimensional time series even with $n = 32$. Interested readers can run the following R code to see that the `dlmMLE()` function in R-DLM is able to run on a 16-dimensional time series. However, when we increase the dimension to 32, `dlmMLE()` crashes the entire R program.

```
library(dlm)
builderFactory <- function(n,k) {
    n = n;
    k = k;
    init = c(rep(0,k), rep(0.1,3),0.1*rnorm(n*k), 0.1*rnorm(k*k))
    build = function(x) {
        m0 = x[1:k]
        C0 = (abs(x[k+1]))*diag(k)
        V  = (abs(x[k+2]))*diag(n)
        W  = (abs(x[k+3]))*diag(k)
        FF = matrix(nrow=n,ncol=k, data=x[(k+3+1):(k+3+n*k)])
        GG = matrix(nrow=k,ncol=k, data=x[(k+3+n*k+1):(k+3+n*k+k*k)])
        return (dlm( m0=m0, C0=C0, FF=FF, GG=GG, V=V, W=W))
    }
    return (list(n=n,k=k,init=init,build=build))
}

Rdlm_train <- function(Y, k, maxit) {
    if(missing(maxit)) { maxit=10 }

    if(ncol(Y)==3) {
        Ymat = matrix(nrow=max(Y(,1)),ncol=max(Y(,2)))
        Ymat[cbind(Y(,1),Y(,2))] = Y(,3)
    } else {
        Ymat = Y;
    }
    n = nrow(Ymat)
    TT = ncol(Ymat)
    dlm_builder = builderFactory(n, k)
    mle = dlmMLE(Ymat,dlm_builder$init,build=dlm_builder$build,
                 control=list(maxit=10))
    dlm = dlm_builder$build(mle$par)
    dlm_filt = dlmFilter(Ymat,dlm)
    return (dlm_filt)
}

tmp = t(as.matrix(Nile));
tmp=rbind(tmp,tmp); tmp=rbind(tmp,tmp);
tmp=rbind(tmp,tmp); tmp=rbind(tmp,tmp);

print(nrow(tmp))
```

```
Rdlm_train(tmp,4);
print('works')

tmp=rbind(tmp,tmp);
print(nrow(tmp))
Rdlm_train(tmp,4);
```