Online Forecasting Matrix Factorization

San Gultekin and John Paisley

***Abstract*—We consider the problem of forecasting a high-dimensional time series that can be modeled as matrices where each column denotes a measurement and use low-rank matrix factorization for predicting future values or imputing missing ones. We define and analyze our problem in the online setting in which the data arrive as a stream and only a single pass is allowed. We present and analyze new matrix factorization techniques that can learn low-dimensional embeddings effectively in an online manner. Based on these embeddings, we derive a recursive minimum mean square error estimator based on an autoregressive model. Experiments with two real datasets of tens of millions of measurements show the benefits of the proposed approach.**

***Index Terms*—Online learning, time series, forecasting, matrix factorization, embeddings, estimation theory.**

I. Introduction

MUCH of signal processing is founded on problems of acquiring, storing, and analyzing data that constantly changes and presents new challenges. These challenges have been growing as some techniques become obsolete for handling the complexity and size of big data. Streaming data with missing values is one such challenge, for which matrix factorization techniques are ideally designed. For example, collaborative filtering [1], [2] has addressed the problem of recommendation [3], while related methods have addressed natural language processing [4], [5], image processing [6], finance [7], and power systems analysis [8]. Non-negative matrix factorization has also received significant attention for parts-based learning problems [9]–[11].

While matrix factorization has been a popular choice for many different problems, its applications to time series analysis has been relatively less developed. Modern time series can often be viewed as time-evolving high dimensional vectors with missing values. An entire time series can be treated as a sparse matrix, for which low-rank representations can be useful. For example, the recent work of [12] proposed a temporal regularized matrix factorization based on this observation. A key property of their solution is that the columns of one of the factor matrices is regularized by an AR process. The coefficients of this process are learned from the data, and can be used to forecast future values. The emphasis in [12] was on batch learning, but for many practical applications it might be impractical to load and process the entire batch, or the data itself may be arriving as a stream. This leads to the online learning setting, where the data is processed as a stream and no storage or multiple passes are allowed.

Online learning for time series prediction is an active area of research [13]–[15]. In particular, the recent work [16] considers online predictions with missing values. However, online forecasting of time series has not been considered from a matrix factorization perspective. In this paper we discuss this approach and propose novel techniques for forecasting high dimensional time series based on matrix factorization. A key observation in previous works of [13], [16] is that the textbook methods for time series analysis typically assume stationarity and/or Gaussianity of noise, which is often unrealistic. In this paper we make fewer assumptions about the data generating process.

The problem we consider in this paper falls into the category of dynamic matrix factorization, in which one finds time-varying factors for a time-varying observation matrix. One appealing approach to dynamic matrix factorization uses state-space models. Here, the columns of latent factor matrices follow a generative state-space model, and their values are inferred from the observations. This is equivalent to a non-convex version of the Kalman filter [17]. Several recent works consider dynamic state-space models in the batch setting [18]–[20]. On the other hand, [21] proposed a dynamic state-space model called the collaborative Kalman filter (CKF), which can estimate the states in an online manner. It is also worthwhile to note that many well established algorithms such as probabilistic matrix factorization [1] can also be extended to the online and dynamic settings. Another recent work [22] is concerned with providing theoretical guarantees in the dynamic setting.

Our problem setup is different from this previous literature in two senses: First, in the online case we no longer observe a dynamic matrix at each time point, but instead observe a single column of a high dimensional time-series. Therefore, this setting is ill-defined as a naive dynamic matrix factorization approach will always give a rank-1 approximation. Second, while previous work focuses on predicting missing entries of the current observation matrix, the forecasting problem is concerned with predicting future values, for which an additional extrapolation step is necessary.

Another line of work, known as latent subspace tracking, considers online estimation of a time-varying subspace. For example the GRASTA algorithm has been successfully applied for video processing [23] and uses an ADMM-based optimization to recover the time series, whereas the works ReProCS [24]–[26]

Manuscript received December 19, 2017; revised May 21, 2018 and October 31, 2018; accepted December 15, 2018. Date of publication December 27, 2018; date of current version January 14, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Namrata Vaswani. *(Corresponding author: San Gultekin.)*

The authors are with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: sg3108@columbia.edu; jpaisley@columbia.edu).

Digital Object Identifier 10.1109/TSP.2018.2889982

and Online Stochastic RPCA [27] decompose the observation as sum of a low rank term and sparse noise. Finally, we note that there is a significant body of work that considers the missing value and forecasting problems in other settings: [28] proposes a convex optimization framework for transition matrix estimation in vector-valued time series. [29] employs a time-series model to represent missing observations, while [30] handles them with an EM algorithm. The work [31] uses an AR process to impute missing values, and [32] considers the Kalman filtering problem with intermittent observations. The main benefit of of using matrix factorization is that, the low rank representation is a natural choice for high dimensional and sparse time series. And as shown in this paper, non-trivial low rank factorizations can be learned efficiently in the online setting.

We organize this paper as follows: Section II establishes the background for AR processes and the matrix factorization approach to time series analysis. Section III is concerned with introducing matrix factorization methods, which finds low-rank factorizations suitable for forecasting. Building on such factorization, Section IV shows how the coefficients of the AR process can be estimated in an optimal manner. Section V contains experiments with two real datasets with tens of millions of measurements; our experiments show that the proposed techniques are effective in practical situations. We conclude in Section VI.

II. Background and Motivation

This section provides background on time series and matrix factorization, and introduces a generative model which we subsequently develop. In this paper, we are interested in forecasting the future values of a high dimensional time series $\{\boldsymbol{x}_t\}_{t=1}^T$, where each $\boldsymbol{x}_t$ is an $M \times 1$ vector. At each time step t the value of $\boldsymbol{x}_t$ must be predicted before it is observed, denoted by $\widehat{\boldsymbol{x}}_t$, and after observation the model is updated according to a loss function. In this paper we let the time indices be discrete and equally spaced in time, t. Total number of samples is T and $[T] = \{1, 2, \ldots, T\}$.

In the well-known Box-Jenkins approach [33], given samples one constructs a signal model by finding (i) a trend, (ii) a seasonal component, and (iii) a noise component, where the latter is typically modeled by an autoregressive moving average (ARMA) model, which is a combination of the AR and MA models. The learned model can then be evaluated using appropriate statistical tests. One drawback of this approach is that finding a trend and seasonal component requires storage and processing of the entire data, which might be unsuitable due to storage or computation time requirements. In addition, this methodology is also unsuitable for streaming data.

For these reasons, we start from a generic vector AR process model, VAR(P), of form

$$\boldsymbol{x}_t = \theta_1 \boldsymbol{x}_{t-1} + \cdots + \theta_P \boldsymbol{x}_{t-P} + \boldsymbol{\eta}_{\boldsymbol{x},t}, \tag{1}$$

where $\boldsymbol{\eta}_{\boldsymbol{x},t}$ is zero mean white noise and P denotes the model order. The choice of P has a major impact on the accuracy of the model, as it captures the maximal lag for correlation. Let the parameters of this model be denoted by $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_P]^\top$. It is clear that, with scalar coefficients VAR(P) corresponds to M copies of an AR(P) model. In addition, when the polynomial $\psi^P - \theta_1 \psi^{P-1} - \cdots - \theta_P$ has roots inside the unit circle, the model is stationary [34]. For a given finite number of measurements, the parameters of the AR(P) model can be estimated by minimizing the mean square error

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}'} \mathbb{E}_\theta \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2, \tag{2}$$

where expectation is taken with respect to the prior $p(\boldsymbol{\theta})$. An advantage of this is that the optimum linear minimum mean squared error estimator (LMMSE) does not make any distribution assumptions on $p(\boldsymbol{\theta})$ or $p(\boldsymbol{\eta})$, and can be calculated in closed form given the first and second order statistics. Also, unlike least squares or the best linear unbiased estimator, LMMSE is guaranteed to exist.

Returning to matrix factorization, we first observe that a time series can be represented by an $M \times T$ matrix $\boldsymbol{X}$. If d denotes the rank of this matrix, then it is possible to find a $d \times M$ matrix $\boldsymbol{U}$ and a $d \times T$ matrix $\boldsymbol{V}$ such that $\boldsymbol{X} = \boldsymbol{U}^\top \boldsymbol{V}$ [35]. Note that such a factorization is not-unique and there are multiple ways to it such as singular value decomposition (SVD). Furthermore, when $\boldsymbol{X}$ represents a time series, such a factorization can be interpreted as follows: Since the matrix $\boldsymbol{V}$ is $d \times T$, it corresponds to a compression of the original $M \times T$ matrix $\boldsymbol{X}$. Therefore the matrix $\boldsymbol{V}$ is itself a time series, while the matrix $\boldsymbol{U}$ provides the combination coefficients to reconstruct $\boldsymbol{X}$ from $\boldsymbol{V}$. Based on this observation, [12] proposed a temporal regularized matrix factorization where the regularizer on the columns of $\boldsymbol{V}$ is in the form of an AR process. They showed that such a regularization has notable impact on performance.

Motivated by this, our goal is to learn the factorizations $\boldsymbol{U}$ and $\boldsymbol{V}$, along with the AR model of Eq. (2) in the online setting, where at each time instance we observe a single column of the data matrix, $\boldsymbol{x}_t$. While this is similar to previous work on online/dynamic matrix factorization [18], [21], one main issue sets it apart. In these papers, at each time an $M \times N$ matrix is observed with $N \gg 1$, while in our case the observation is simply $M \times 1$. This is illustrated in Fig. 1; in (a) we show the batch factorization of an $M \times T$ matrix $\boldsymbol{X}$ and (b) is the case where at each time a subset of the matrix entries are observed. However, when $\boldsymbol{X}$ is a time series matrix, at each time we observe a single column as shown in (c).

A problem with sequentially observing and dynamically factorizing vectors is that the latent rank is at most 1, whereas the batch problem in Fig. 1(a) will have a solution of rank d. Since the end goal here is to factorize the entire data with two evolving matrices, it is desirable to start from a rank-d representation and gradually update it. However, finding such factors naively gives poor performance (Fig. 4), therefore our task is to devise an effective way of achieving this. This can be done using specific penalties on the matrix $\boldsymbol{U}$, which yield feasible optimization problems, as discussed in the next section. One way to motivate our approach is to consider a probabilistic generative state-space representation for the data, as frequently used in Bayesian methods [36]. Previous research [20], [37] shows that generative model approach is indeed effective at capturing time evolution, in the context of finding user embeddings. Our

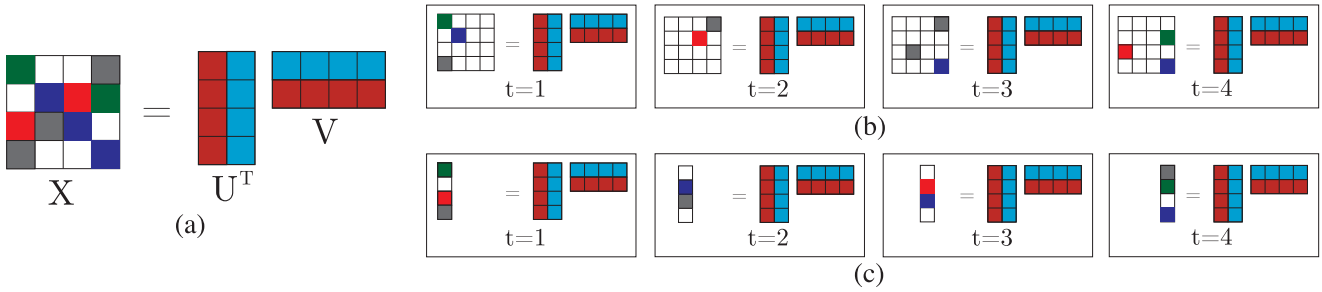

Fig. 1. Comparison of online matrix factorization schemes. (a) A matrix $\boldsymbol{X}$ is factorized in the batch setting, whereas in (b) at each time a subset of the matrix is observed. For illustrative purposes the observed rank is always greater than one. (c) Shows online matrix factorization, where without appropriate regularization (implied in what is shown) the rank cannot exceed one.

model is

$$\begin{aligned}\boldsymbol{U}_t &= \boldsymbol{U}_{t-1} + \boldsymbol{\eta}_{\boldsymbol{U},t}\\ \boldsymbol{v}_t &= \theta_1 \boldsymbol{v}_{t-1} + \cdots + \theta_P \boldsymbol{v}_{t-P} + \boldsymbol{\eta}_{\boldsymbol{v},t}\\ \boldsymbol{x}_t &= \boldsymbol{U}_t^\top \boldsymbol{v}_t + \boldsymbol{\eta}_{\boldsymbol{x},t},\end{aligned} \tag{3}$$

where $[\boldsymbol{\eta}_{\boldsymbol{U},T}]$, $[\boldsymbol{\eta}_{\boldsymbol{v},T}]$, and $[\boldsymbol{\eta}_{\boldsymbol{x},T}]$ are white noise sequences, independent of each other.

For many time series forecasting purposes an AR model is sufficient, as the past values may be the only inputs available. However, if additional predictors are given at any point in time, they can also be incorporated in the generative model. For instance, if an additional set of predictor vectors $\{\boldsymbol{w}_{t,1}, \ldots, \boldsymbol{w}_{t,R}\}$ are provided we can set

$$\boldsymbol{v}_t = \sum_{p=1}^{P} \theta_p \boldsymbol{v}_{t-p} + \sum_{r=1}^{R} \theta'_r \boldsymbol{w}_{t,r} + \boldsymbol{\eta}_{v,t}.$$

In this paper we only assume access to the time series.

To compare the state-space model in Eq. (3) to the related work: If we set $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \eta_{\boldsymbol{v},t}$, we recover the setting of matrix factorization. Using the previous value of $\boldsymbol{U}$ as regularizer, we have a feasible optimization problem and the factors can be estimated by either alternating least squares [3] or projected gradients for non-negative factorization [9]. Alternatively, the factors can also be estimated using subspace tracking or robust PCA; these methods bring an additional noise term to the model, which provides outlier robustness. In terms of model complexity and number of parameters, the main difference is that, in the above method we use an AR model for time series embeddings. While this introduces an additional AR order parameter to be set, as we show in the experiments it can lead to significant improvement in forecast accuracy, as multiple past values can be used to predict the future. Without this, the model would predict based only on the current factors. Therefore, when we have a time series where the cross-sections are correlated at multiple lags, our generative model can capture the dependencies and provide better forecasts.

III. Online Matrix Factorization

We present algorithms that fit an online AR model to the sequence $[\boldsymbol{v}_T] = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T\}$. To get a good fit, it is necessary to generate the vectors $\boldsymbol{v}_t$ is a proper manner. To illustrate this, for a given measurement vector if we find a factorization $\boldsymbol{x}_t = \boldsymbol{U}_t^\top \boldsymbol{v}_t$, for any orthogonal matrix $\boldsymbol{Q}$ and positive scaling constant a we get $\boldsymbol{x}_t = (a\boldsymbol{Q}\boldsymbol{U}_t)^\top (a^{-1}\boldsymbol{Q}\boldsymbol{v}_t)$. This scaling and rotation could have a significant effect on forecasting accuracy (Fig. 4). On the other hand, the matrix $\boldsymbol{U}_t$ is a slowly time-varying quantity which means we can constrain its variation. Accurate selection of the penalty on $\boldsymbol{U}_t$ has a dramatic effect on the generated $[\boldsymbol{v}_T]$, which then dictates the forecasting accuracy.

Another assumption we make is that the absolute value of observations are upper bounded by a finite number. Therefore, by scaling we can assume $\sup_{\boldsymbol{x} \in [\boldsymbol{x}_T]} \|\boldsymbol{x}\|_\infty = 1$. For the power data we will consider, this is dictated by the physical constraints of the network. For the traffic data we will consider, the measurements are in percentages.

A. Fixed Penalty Constraint

The first algorithm we present is based on a simple fixed penalty function on the norms of the factors. The batch version for this algorithm was previously considered in [1]. There, the cost function is

$$\begin{aligned}f(\boldsymbol{U}, \boldsymbol{v}) = &\sum_{m,n} (x_{m,n} - \boldsymbol{u}_m^\top \boldsymbol{v}_n)^2\\ &+ \rho_u \sum_m \|\boldsymbol{u}_m\|_2^2 + \sum_n \rho_v \|\boldsymbol{v}_n\|_2^2,\end{aligned} \tag{4}$$

which is equivalent to adding Gaussian priors to each column of $\boldsymbol{U}$ and $\boldsymbol{v}$. In [1] this is referred to as probabilistic matrix factorization (PMF). This non-convex, unconstrained problem can be optimized by coordinate descent

$$\begin{aligned}\boldsymbol{u}_m^{(i)} &\leftarrow \left(\rho_u \boldsymbol{I} + \sum_n \boldsymbol{v}_n^{(i)} \boldsymbol{v}_n^{(i)\top}\right)^{-1} \left(\sum_n x_{m,n} \boldsymbol{v}_n^{(i)}\right),\\ \boldsymbol{v}_n^{(i+1)} &\leftarrow \left(\rho_v \boldsymbol{I} + \sum_m \boldsymbol{u}_m^{(i)} \boldsymbol{u}_m^{(i)\top}\right)^{-1} \left(\sum_m x_{m,n} \boldsymbol{u}_m^{(i)}\right),\end{aligned} \tag{5}$$

found by matrix differentiation.

Turning to the online case, at each time a single column of $\boldsymbol{X}$ is observed. Using the model of Eq. (3), at time t we would like to minimize the following cost function

$$\begin{aligned}f(\boldsymbol{U}_t, \boldsymbol{v}_t) = &\|\boldsymbol{x}_t - \boldsymbol{U}_t^\top \boldsymbol{v}_t\|_2^2\\ &+ \rho_u \|\boldsymbol{U}_t - \overline{\boldsymbol{U}}\|_F^2 + \rho_v \|\boldsymbol{v}_t - \overline{\boldsymbol{v}}\|_2^2.\end{aligned} \tag{6}$$

Here and in subsequent sections we describe the matrix factorization algorithms for generic $\overline{\boldsymbol{U}}$ and $\overline{\boldsymbol{v}}$. From the generative model of Eq. (3) we can see that these should be set as $\overline{\boldsymbol{U}} = \boldsymbol{U}_{t-1}$ and $\overline{\boldsymbol{v}} = \sum_{p=1}^{P} \theta_l \boldsymbol{v}_{t-l}$ In Section IV we discuss how to estimate the hyperparameters in the equation of $\overline{\boldsymbol{v}}$.

Here, $\boldsymbol{U}_t$ is the submatrix of $\boldsymbol{U}$ corresponding to the columns with observation in $\boldsymbol{x}_t$. When there are missing observations, only a subset $\boldsymbol{U}$ gets updated. At a given time t, the number of observations is M_t, and $\boldsymbol{v}_t$ has d parameters. Typically $M_t > d$ and the update for $\boldsymbol{v}_t$ can be feasible even if $\rho_v = 0$; therefore ρ_v is a small set-and-forget constant that we include for numerical stability.[1] Therefore ρ_v can be set to a small constant for numerical stability. On the other hand, $\boldsymbol{U}_t$ contains $M_t d > M_t$ unknowns and the Gram matrix $\boldsymbol{U}_t \boldsymbol{U}_t^\top$ is not invertible. Therefore, $\rho_u > 0$ is necessary to make the problem feasible. Since both ρ_u and ρ_v are fixed at the beginning, we refer to Eq. (6) as a fixed penalty (FP) matrix factorization.

The objective of Eq. (6) finds the maximum a posteriori (MAP) solution. Here, we center the priors on the previous value of $\boldsymbol{U}$ and $\overline{\boldsymbol{v}} = \sum_{p=1}^{P} \theta_l \boldsymbol{v}_{t-l}$. We note that we will set $\rho_u \gg \rho_v$, which means FP will find a solution for which $\boldsymbol{U}_t$ is close to $\boldsymbol{U}_{t-1}$, i.e., $\boldsymbol{U}_t$ is slowly time-varying. This agrees with the interpretation that, in the batch case $\boldsymbol{U}$ is a fixed set of coefficients and $\boldsymbol{V}$ contains the compressed time series. Another caution here is that, setting ρ_v high would over-constrain the problem as both $\boldsymbol{U}_t$ and $\boldsymbol{v}_t$ would be forced to stay close to $\overline{\boldsymbol{U}}$ and $\overline{\boldsymbol{v}}$ while trying to minimize the approximation error to $\boldsymbol{x}_t$. The update equations for FP are

$$\begin{aligned} \boldsymbol{U}_t^{(i)} &\leftarrow \left(\rho_u \boldsymbol{I} + \boldsymbol{v}_t^{(i)} \boldsymbol{v}_t^{(i)\top}\right)^{-1} \left(\rho_u \overline{\boldsymbol{U}} + \boldsymbol{v}_t^{(i)} \boldsymbol{x}_t^\top\right) \\ \boldsymbol{v}_t^{(i)} &\leftarrow \left(\rho_v \boldsymbol{I} + \boldsymbol{U}_t^{(i)} \boldsymbol{U}_t^{(i)\top}\right)^{-1} \left(\rho_v \overline{\boldsymbol{v}} + \boldsymbol{U}_t^{(i)} \boldsymbol{x}_t\right). \end{aligned} \tag{7}$$

A key argument in Eq. (6) is that, the state equations of Eq. (3) addresses scaling and rotation issues through $\overline{\boldsymbol{U}}$ and $\overline{\boldsymbol{v}}$. A naive approach, which does not impose any temporal structure on the latent variables, constructs the alternative objective

$$f(\boldsymbol{U}_t, \boldsymbol{v}_t) = \|\boldsymbol{x}_t - \boldsymbol{U}_t^\top \boldsymbol{v}_t\|_2^2 + \rho_u \|\boldsymbol{U}_t\|_F^2 + \rho_v \|\boldsymbol{v}_t\|_2^2. \tag{8}$$

We also consider this alternative "naive" model in the experiments, to show that, in the absence of temporal regularization in Eq. (3), scaling and rotation cannot be prevented,[2] hindering the prediction quality. The FP matrix factorization is summarized in Algorithm 1.

B. Fixed Tolerance Constraint

The fixed penalty approach to matrix factorization suffers from several potential issues. While ρ_v can be set to a small number, setting ρ_u well has a major impact on performance. It is usually not clear *a priori* which values would yield good results, and often times this may require a large number of cross validations. Another drawback is that ρ_u is fixed for the entire data stream. This may not be desirable as changing the regularization level at different time points may improve performance. For these reasons it can be useful to allow for time varying, self-tunable regularization.

Algorithm 1: Fixed Penalty Matrix Factorization (FP).

Require: $\boldsymbol{x}_t, \mathcal{I}_t, \rho_u, \rho_v, \overline{\boldsymbol{U}}, \overline{\boldsymbol{v}}$, `max_ite`
Return: $\boldsymbol{U}_t, \boldsymbol{v}_t$
Re-assign $\overline{\boldsymbol{U}} \leftarrow \overline{\boldsymbol{U}}(:, \mathcal{I}_t)$
for $i = 1, \ldots,$ `max_ite` **do**
  $\boldsymbol{v}^{(i)} \leftarrow (\rho_v \boldsymbol{I} + \boldsymbol{U}^{(i-1)} \boldsymbol{U}^{(i-1)\top})^{-1} (\rho_v \overline{\boldsymbol{v}} + \boldsymbol{U}^{(i-1)} \boldsymbol{x}_t)$
  $\boldsymbol{U}^{(i)} \leftarrow (\rho_u \boldsymbol{I} + \boldsymbol{v}^{(i)} \boldsymbol{v}^{(i)\top})^{-1} (\rho_u \overline{\boldsymbol{U}} + \boldsymbol{v}^{(i)} \boldsymbol{x}_t^\top)$
end for
Update $\boldsymbol{U}_t(:, \mathcal{I}_t) \leftarrow \boldsymbol{U}$ and overwrite $\boldsymbol{U}_t(:, \mathcal{I}_t^c) \leftarrow \boldsymbol{U}_{t-1}(:, \mathcal{I}_t^c)$.
Update $\boldsymbol{v}_t \leftarrow \boldsymbol{v}$.
Note 1: $\boldsymbol{U}_t(:, \mathcal{I}_t)$ are those columns for which there is a corresponding observation at time t, indexed by $\mathcal{I}_t$. The remaining columns are $\boldsymbol{U}_t(:, \mathcal{I}_t^c)$..
Note 2: At time t, only the observed entries get updated.

To address this we consider the following problem

$$\begin{aligned} &\min_{\boldsymbol{U}_t, \boldsymbol{v}_t} \|\boldsymbol{U}_t - \overline{\boldsymbol{U}}\|_F^2 + \|\boldsymbol{v}_t - \overline{\boldsymbol{v}}\|_2^2 \\ &\text{s.t. } \|\boldsymbol{x}_t - \boldsymbol{U}_t^\top \boldsymbol{v}_t\|_2^2 \le \epsilon \end{aligned} \tag{9}$$

Instead of having ρ_u and ρ_v, we introduced ϵ. This new parameter forces the approximation error to remain below ϵ. Since this error bound is fixed at the beginning, we call this fixed tolerance (FT) matrix factorization. Here $\overline{\boldsymbol{U}}$ and $\overline{\boldsymbol{v}}$ are defined as in FP. Based on the model in Eq. (3), we can interpret the optimization problem of Eq. (9) as follows: FT aims finding the point estimates closest to the previous values that keep the error below ϵ. The objective function of FT can be optimized by coordinate descent.

1) Update for $\boldsymbol{U}_t$: For a fixed $\boldsymbol{v}_t$ the Lagrangian is

$$\mathcal{L}(\boldsymbol{U}_t, \lambda) = \|\boldsymbol{U}_t - \overline{\boldsymbol{U}}\|_F^2 + \lambda \|\boldsymbol{x}_t - \boldsymbol{U}_t^\top \boldsymbol{v}_t\|_2^2 - \lambda \epsilon, \tag{10}$$

which yields the following update for $\boldsymbol{U}_t$.

$$\boldsymbol{U}_t \leftarrow (\lambda^{-1} \boldsymbol{I} + \boldsymbol{v}_t \boldsymbol{v}_t^\top)^{-1} (\lambda^{-1} \overline{\boldsymbol{U}} + \boldsymbol{v}_t \boldsymbol{x}_t^\top). \tag{11}$$

This is equivalent to (7) when $\lambda = \rho_u^{-1}$. But since the Lagrange multiplier changes value with every update of $\boldsymbol{v}_t$ we now have a varying regularizer. We will discuss setting λ after the following update for $\boldsymbol{v}_t$.

2) Update for $\boldsymbol{v}_t$: For fixed $\boldsymbol{U}_t$ the Lagrangian and optimal update for $\boldsymbol{v}_t$ are

$$\mathcal{L}(\boldsymbol{v}_t, \lambda) = \|\boldsymbol{v}_t - \overline{\boldsymbol{v}}\|_2^2 + \lambda \|\boldsymbol{x}_t - \boldsymbol{U}_t^\top \boldsymbol{v}_t\|_2^2 - \lambda \epsilon, \tag{12}$$

$$\boldsymbol{v}_t \leftarrow (\lambda^{-1} \boldsymbol{I} + \boldsymbol{U}_t \boldsymbol{U}_t^\top)^{-1} (\lambda^{-1} \overline{\boldsymbol{v}} + \boldsymbol{U}_t \boldsymbol{x}_t). \tag{13}$$

3) Optimizing the Lagrange Multiplier λ: The main issue with this FT approach is the structure of the constraint set and its enforcement via the Lagrange multiplier λ. This is a quadratically constrained quadratic program (QCQP), for which there is no closed-form solution in general [39]. The optimization for

[1] Indeed, $\rho_v = 10^{-4}$ for all experiments in this paper.

[2] One alternative way to address this problem would utilize post-processing. In particular, the optimum rotation between two sets of points can be found by solving the Procrustes problem [38] however this would incur additional computation.

Algorithm 2: Fixed Tolerance Matrix Factorization (FT).

Require: $\boldsymbol{x}_t, \mathcal{I}_t, \epsilon, \rho_v, \overline{\boldsymbol{U}}, \overline{\boldsymbol{v}}$, `max_ite`
Return: $\boldsymbol{U}_t, \boldsymbol{v}_t$
Re-assign $\overline{\boldsymbol{U}} \leftarrow \overline{\boldsymbol{U}}(:, \mathcal{I}_t)$
for $i = 1, \ldots,$ `max_ite` **do**
  $\boldsymbol{v}^{(i)} \leftarrow (\rho_v \boldsymbol{I} + \boldsymbol{U}^{(i-1)} \boldsymbol{U}^{(i-1)\top})^{-1} (\rho_v \overline{\boldsymbol{v}} + \boldsymbol{U}^{(i-1)} \boldsymbol{x}_t)$
  Compute c_1, c_2, as in (14).
  $\lambda^\star \leftarrow \frac{\sqrt{c_1}}{\sqrt{\epsilon} c_2} - \frac{1}{c_2}$
  $\boldsymbol{U}^{(i)} \leftarrow (\boldsymbol{I} + \lambda^\star \boldsymbol{v}^{(i)} \boldsymbol{v}^{(i)\top})^{-1} (\overline{\boldsymbol{U}} + \lambda^\star \boldsymbol{v}^{(i)} \boldsymbol{x}_t^\top)$
end for
Update $\boldsymbol{U}_t(:, \mathcal{I}_t) \leftarrow \boldsymbol{U}$ and $\boldsymbol{U}_t(:, \mathcal{I}_t^c) \leftarrow \boldsymbol{U}_{t-1}(:, \mathcal{I}_t^c)$.
Update $\boldsymbol{v}_t \leftarrow \boldsymbol{v}$.

$\boldsymbol{U}_t$ given $\boldsymbol{v}_t$ can be shown to be convex, so off-the-shelf solvers could be employed to find the global optimum. However, using a convex solver at every time step is inefficient and defeats the purpose of scalable online learning.

In fact a closed form solution to $\boldsymbol{U}_t$ can be found. Defining

$$c_1 = \|\boldsymbol{x}_t - \overline{\boldsymbol{U}}^\top \boldsymbol{v}_t\|_2^2, \; c_2 = \|\boldsymbol{v}_t\|_2^2, \tag{14}$$

and setting the Lagrange multiplier to

$$\lambda^\star = \frac{\sqrt{c_1}}{c_2 \sqrt{\epsilon}} - \frac{1}{c_2} \tag{15}$$

the optimal update for $\boldsymbol{U}_t$ is

$$\boldsymbol{U}_t \leftarrow (\boldsymbol{I} + \lambda^\star \boldsymbol{v}_t \boldsymbol{v}_t^\top)^{-1} (\overline{\boldsymbol{U}} + \lambda^\star \boldsymbol{v}_t \boldsymbol{x}_t^\top). \tag{16}$$

We observe by the nature of (9), if $\epsilon > c_1$ then $\boldsymbol{U}_t = \overline{\boldsymbol{U}}$. We discuss this derivation in Appendix A.

A more challenging issue arises for the update of $\boldsymbol{v}_t$. For a given threshold ϵ it is not clear if we can find a $\boldsymbol{v}_t$ such that the constraint is satisfied. As an example, when the system of equations is over-determined, the smallest error we can achieve is the least squares error. When the system is underdetermined and the least squares error is greater than ϵ, the value of $\boldsymbol{v}_t$ from the previous iteration will still be the best (discussed in Appendix B). Unfortunately, the feasible set contains many such isolated points. Therefore if we seek the minimum norm solution for $\boldsymbol{v}_t$ in Eq. (9) it is likely that the optimization will terminate early resulting in poor performance.

To fix this we propose the following modification: Instead of finding the minimum-norm solution, we consider updating $\boldsymbol{v}_t$ using the Lagrangian in Eq. (12) for a fixed λ. This gives the same update equation as FP, which is given in Eq. (7). We discuss the relationship between our ridge approximation and the true solution in Appendix B.

We summarize FT summarized in Algorithm 2. The key difference between FT and FP is the computation of λ when updating of $\boldsymbol{U}_t$.

C. Zero Tolerance Constraint

We have discussed two different approaches to online matrix factorization, fixed penalty (FP) and fixed tolerance (FT). From the user perspective, the difference is in replacing one tunable parameter with another. We next discuss a parameter free option in which $\epsilon = 0$, which we refer to as zero tolerance (ZT) matrix factorization. Interpreting from the perspective of the model in Eq. (3), ZT estimates the latent factors $\boldsymbol{U}_t$ and $\boldsymbol{v}_t$ that are as close to the prior as possible, while allowing no approximation error on $\boldsymbol{x}_t$.

The optimization problem now becomes

$$\min_{\boldsymbol{U}_t, \boldsymbol{v}_t} \|\boldsymbol{U}_t - \overline{\boldsymbol{U}}\|_F^2 + \|\boldsymbol{v}_t - \overline{\boldsymbol{v}}\|_2^2 \tag{17}$$

$$\text{s.t. } \boldsymbol{U}_t^\top \boldsymbol{v}_t = \boldsymbol{x}_t. \tag{18}$$

This is related to nuclear norm minimization problems [40], [41]. In this scenario, we consider the factored form of the nuclear norm [42] and performed online optimization.

Considering optimizing $\boldsymbol{U}_t$ while $\boldsymbol{v}_t$ is fixed, as before the linear system $\boldsymbol{U}_t \boldsymbol{v}_t = \boldsymbol{x}_t$ is underdetermined for a variable $\boldsymbol{U}_t$. Eq. (17) suggests finding the solution with the least Frobenius norm. This generalizes the least norm problem that is considered for linear underdetermined systems to the matrix case. Since the system is underdetermined, the feasible set will contain infinitely many points. (This follows the same reasoning discussed in Appendix A.)

Optimizing $\boldsymbol{U}_t$ given $\boldsymbol{v}_t$ can be done with Lagrange multipliers. Following a rescaling, the Lagrangian is given by

$$\mathcal{L}(\boldsymbol{U}_t, \boldsymbol{\lambda}) = \frac{1}{2} \|\boldsymbol{U}_t - \boldsymbol{U}_{t-1}\|_F^2 + \boldsymbol{\lambda}^\top (\boldsymbol{x}_t - \boldsymbol{U}_t \boldsymbol{v}_t). \tag{19}$$

The stationarity conditions are

$$\begin{aligned} \nabla_{\boldsymbol{U}_t} \mathcal{L}(\boldsymbol{U}_t, \boldsymbol{\lambda}) = 0 &= \boldsymbol{U}_t - \boldsymbol{U}_{t-1} + \boldsymbol{v}_t \boldsymbol{\lambda}^\top, \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{U}_t, \boldsymbol{\lambda}) = 0 &= \boldsymbol{U}_{t-1}^\top \boldsymbol{v}_t - \boldsymbol{x}_t. \end{aligned} \tag{20}$$

The solution is then

$$\boldsymbol{\lambda} = \frac{\boldsymbol{U}_{t-1}^\top \boldsymbol{v}_t - \boldsymbol{x}_t}{\boldsymbol{v}_t^\top \boldsymbol{v}_t}, \; \boldsymbol{U}_t = \boldsymbol{U}_{t-1} - \boldsymbol{v}_t \boldsymbol{\lambda}^\top. \tag{21}$$

Here, though $\boldsymbol{\lambda}$ is changing over time, it can no longer be seen as the inverse regularizer of the FP term because ϵ is no longer a tunable parameter, but hard-coded to zero. This is advantageous in that the user does not have to cross-validate to find a good value for it. On the other hand, as we will show in the experiments, the $\epsilon = 0$ requirement can become too restrictive in some cases, which will then require a higher rank factorization.

The update for $\boldsymbol{v}_t$ suffers from the same problem discussed in the previous section. For the ZT constraint, consider when $\boldsymbol{U}_t^\top \boldsymbol{v}_t = \boldsymbol{x}_t$ is overdetermined for variable $\boldsymbol{v}_t$. Then the smallest error achievable will be given by the least squares solution, which satisfies $\epsilon_{\text{ls}} > 0$, so the feasible set is empty. However, since the the optimization is done in an alternating manner, this worst case does not occur in practice. In particular, at iteration $i-1$ we have found $(\boldsymbol{U}_{t-1}^{(i-1)}, \boldsymbol{v}_{t-1}^{(i-1)})$ such that $\boldsymbol{U}_{t-1}^{(i-1)\top} \boldsymbol{v}_{t-1}^{(i-1)} = \boldsymbol{x}_t$. This means, when we update for $\boldsymbol{v}_t^{(i)}$ for a fixed $\boldsymbol{U}_t^{(i-1)}$, the feasible set will contain at least $\boldsymbol{v}_t^{(i-1)}$. The problem is if this is the only point contained in the feasible set, the optimization will terminate early in the process. We again address this by replacing the least norm solution with the ℓ_2 regularized one, for which we re-introduce ρ_v as a small parameter.

Algorithm 3: Zero Tolerance Matrix Factorization (ZT).

Require: $\boldsymbol{x}_t, \mathcal{I}_t, \rho_v, \overline{\boldsymbol{U}}, \overline{\boldsymbol{v}}$, max_ite
Return: $\boldsymbol{U}_t, \boldsymbol{v}_t$
Re-assign $\overline{\boldsymbol{U}} \leftarrow \overline{\boldsymbol{U}}(:, \mathcal{I}_t)$
for $i = 1, \ldots,$ max_ite **do**
  $\boldsymbol{v}^{(i)} \leftarrow (\rho_v \boldsymbol{I} + \boldsymbol{U}^{(i-1)}\boldsymbol{U}^{(i-1)\top})^{-1}(\rho_v \overline{\boldsymbol{v}} + \boldsymbol{U}^{(i-1)}\boldsymbol{x}_t)$
  $\boldsymbol{\lambda} \leftarrow (\overline{\boldsymbol{U}}\boldsymbol{v}^{(i)} - \boldsymbol{x}_t)/(\boldsymbol{v}^{(i)\top}\boldsymbol{v}^{(i)})$
  $\boldsymbol{U}^{(i)} \leftarrow \overline{\boldsymbol{U}} - \boldsymbol{v}^{(i)}\boldsymbol{\lambda}^\top$
end for
Update $\boldsymbol{U}_t(:, \mathcal{I}_t) \leftarrow \boldsymbol{U}$ and $\boldsymbol{U}_t(:, \mathcal{I}_t^c) \leftarrow \boldsymbol{U}_{t-1}(:, \mathcal{I}_t^c)$.
Update $\boldsymbol{v}_t \leftarrow \boldsymbol{v}$.

The update is then

$$\boldsymbol{v}_t \leftarrow (\rho_v \boldsymbol{I} + \boldsymbol{U}_t\boldsymbol{U}_t^\top)^{-1}(\rho_v \overline{\boldsymbol{v}} + \boldsymbol{U}_t\boldsymbol{x}_t). \tag{22}$$

In summary, ZT is simply the special case of FT where we set $\epsilon = 0$. As ρ_v is a small constant, ZT is effectively a parameter-free matrix factorization method. ZT is summarized in Algorithm 3.

IV. Optimum Sequence Prediction

In Section III we presented three online matrix factorization approaches with smoothness penalties to constrain the evolving $\boldsymbol{U}$. As discussed in Section II, each column $\boldsymbol{v}_t$ in the product $\boldsymbol{x}_t \approx \boldsymbol{U}_t\boldsymbol{v}_t$ is also generated sequentially. When the columns of the original time series matrix $\boldsymbol{X}$ are correlated, it is natural to model a correlation structure in $\boldsymbol{V}$. For this reason, we use an AR model for the columns of $\boldsymbol{V}$,

$$\boldsymbol{v}_t = \underbrace{\theta_1 \boldsymbol{v}_{t-1} + \ldots + \theta_P \boldsymbol{v}_{t-P}}_{\overline{\boldsymbol{v}}} + \boldsymbol{\eta}_{\boldsymbol{v},t}. \tag{23}$$

As indicated, $\overline{\boldsymbol{v}}$ in the previous algorithms is an AR model.

Therefore, a further task is to find the coefficient vector $\boldsymbol{\theta}$ for this AR model. While there is no single answer, it is useful to have a flexible estimation method with and few assumptions. Thus, we adopt the LMMSE estimator since (i) it only needs first and second order statistics, and (ii) optimization is numerically stable, in contrast to, e.g., the best linear unbiased (BLU) estimator.

Below, we introduce the following notation: First note that (23) corresponds to $\boldsymbol{v}_t = \boldsymbol{P}_t\boldsymbol{\theta}$ where $\boldsymbol{P}_t = [\boldsymbol{v}_{t-1} \cdots \boldsymbol{v}_{t-P}]$ is a $d \times P$ *patch* matrix of the previous P columns. The collection of such matrices is obtained by vertical stacking, $\boldsymbol{P}^\top = [\boldsymbol{P}_1^\top \cdots \boldsymbol{P}_T^\top]$, which is a $Td \times P$ matrix. Stacking the observation vectors vertically, we obtain $\boldsymbol{p}^\top = [\boldsymbol{v}_1^\top \cdots \boldsymbol{v}_T^\top]$, a vector with Td elements. The vector $\boldsymbol{\eta}$ is defined similarly.[3]

Using this notation, for the set $[\boldsymbol{v}_T]$, we have the relation

$$\boldsymbol{P}\boldsymbol{\theta} + \boldsymbol{\eta} = \boldsymbol{p}, \tag{24}$$

which means each vector observation contains information about a latent vector $\boldsymbol{\theta}$. This is different from Kalman Filter [17], where the vector $\boldsymbol{\theta}$ itself is a time-varying latent variable. A good estimator should provide accurate values for $\boldsymbol{\theta}$, with minimal assumptions about the distributions of the random variables involved. To that aim, we let the noise distribution have the first- and second-order statistics

$$\mathbb{E}[\boldsymbol{\eta}_t] = \boldsymbol{0},\ \mathbb{E}[\boldsymbol{\eta}_{t_1}\boldsymbol{\eta}_{t_2}^\top] = \boldsymbol{\Sigma}_{\boldsymbol{\eta}}\delta(t_1, t_2), \tag{25}$$

where $\delta(t_1, t_2)$ is the Kronecker delta function. This is a white noise process with stationary covariance. For the parameter $\boldsymbol{\theta}$ there are two options: (i) it can be treated as an unknown deterministic parameter (classical inference) or (ii) it can be modeled as a random variable (Bayesian inference). We will choose (ii) and assume that $\boldsymbol{\theta}$ satisfies the following

$$\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{0},\ \mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^\top] = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}. \tag{26}$$

We also note that $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ are assumed independent. Based on the above, we restrict ourselves to the linear estimators of form $\widehat{\boldsymbol{\theta}} = \boldsymbol{W}\boldsymbol{p}$ with $\boldsymbol{W}$ is a $P \times dT$ weight matrix. We want to minimize the mean square error

$$\text{MSE} = \mathbb{E}\min_{\widehat{\boldsymbol{\theta}}} \|\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\|_2^2 \equiv \mathbb{E}\min_{\boldsymbol{W}} \|\boldsymbol{\theta} - \boldsymbol{W}\boldsymbol{p}\|_2^2, \tag{27}$$

for which we can write the expected MMSE as

$$\begin{aligned}\text{MSE}(\boldsymbol{W}) &= \mathbb{E}\left[(\boldsymbol{\theta} - \boldsymbol{W}\boldsymbol{p})^\top(\boldsymbol{\theta} - \boldsymbol{W}\boldsymbol{p})\right] \\ &= \text{tr}\{\boldsymbol{\Sigma}_{\boldsymbol{\theta}} + \boldsymbol{W}(\boldsymbol{P}\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\boldsymbol{P}^\top + \boldsymbol{\Sigma}_{\boldsymbol{\eta}})\boldsymbol{W}^\top - 2\boldsymbol{W}\boldsymbol{P}\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\},\end{aligned} \tag{28}$$

which shows the optimum estimator can be found matrix derivative to be

$$\boldsymbol{W} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\boldsymbol{P}^\top(\boldsymbol{\Sigma}_{\boldsymbol{\eta}} + \boldsymbol{P}\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\boldsymbol{P}^\top)^{-1}. \tag{29}$$

Using the matrix inversion lemma, we can equivalently write.

$$\begin{aligned}\boldsymbol{W} &= \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\boldsymbol{P}^\top\left[\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-1}\boldsymbol{P}[\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} + \boldsymbol{P}^\top\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-1}\boldsymbol{P}]^{-1}\boldsymbol{P}^\top\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-1}\right] \\ &= [\boldsymbol{P}^\top\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-1}\boldsymbol{P} + \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}]^{-1}\boldsymbol{P}^\top\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-1}.\end{aligned} \tag{30}$$

This last line results from algebraic manipulation. The LMMSE estimator is then given by

$$\widehat{\boldsymbol{\theta}} = [\boldsymbol{P}^\top\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-1}\boldsymbol{P} + \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}]^{-1}\boldsymbol{P}^\top\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-1}\boldsymbol{p}. \tag{31}$$

It follows from this functional form that the LMMSE estimator reduces to BLUE when a non-informative prior is chosen. This can effectively be written as $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \infty\boldsymbol{I}$. When $\boldsymbol{\Sigma}_{\boldsymbol{\eta}} = \boldsymbol{I}$, BLUE coincides with the ordinary least squares estimator, yielding the Gauss-Markov theorem [43]. For this paper we consider the case $\boldsymbol{\Sigma}_{\boldsymbol{\eta}} = \boldsymbol{I}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = r_0\boldsymbol{I}$ for a tunable parameter r_0.

The transition from (29) to (30) with $\boldsymbol{\Sigma}_{\boldsymbol{\eta}} = \boldsymbol{I}$ allows us to use matrix partitioning [35] and write (31) as

$$\widehat{\boldsymbol{\theta}} = \left[\sum_{t=1}^{T}\boldsymbol{P}_t^\top\boldsymbol{P}_t + \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}\right]^{-1}\left[\sum_{t=1}^{T}\boldsymbol{P}_t^\top\boldsymbol{v}_t\right]. \tag{32}$$

This shows we can compute terms recursively. Specifically, define $\boldsymbol{r}_{l,0} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}$ and $\boldsymbol{r}_{r,0} = 0$ and the recursions

$$\boldsymbol{r}_{l,t} = \boldsymbol{r}_{l,t-1} + \boldsymbol{P}_t^\top\boldsymbol{P}_t,\ \boldsymbol{r}_{r,t} = \boldsymbol{r}_{r,t-1} + \boldsymbol{P}_t^\top\boldsymbol{v}_t. \tag{33}$$

[3]We observe that at the beginning of Section II we assumed that the observations start at $T = 1$. To obtain a patch matrix which does not contain any zero-column, we should start constructing matrices $\boldsymbol{P}$ and $\boldsymbol{p}$ from the index $t = P + 1$. We omit this detail to simplify the equations. The final algorithm we present, however, addresses this.

Algorithm 4: Online Forecasting Matrix Factorization.

Require: $\boldsymbol{X}$, $\mathcal{I}$, d, r_0, ρ_u (FP), ϵ (FT), ρ_v, `max_ite`
Return: $\forall t$: $\widehat{\boldsymbol{x}}_t$, $\boldsymbol{U}_t$, $\boldsymbol{v}_t$, $\boldsymbol{\theta}_t$
Initialize $\boldsymbol{U}^{(0)} \leftarrow \texttt{rand}(M, d)$, $\boldsymbol{v}^{(0)} \leftarrow \texttt{rand}(d, 1)$.
Initialize $\boldsymbol{r}_{l,0} \leftarrow r_0 \boldsymbol{I}$, $\boldsymbol{r}_{r,0} \leftarrow 0$.
for $t = 1, \ldots, T$ **do**
    `// Forecast Step`
    $\overline{\boldsymbol{U}} \leftarrow \boldsymbol{U}_{t-1}$, $\overline{\boldsymbol{v}} \leftarrow \sum_{p=1}^{P} \theta_l \boldsymbol{v}_{t-l}$ †
    Forecast: $\widehat{\boldsymbol{x}}_t = \overline{\boldsymbol{U}}^\top \overline{\boldsymbol{v}}$
    `// Update Parameters`
    Use one of the following:
        • FP($\boldsymbol{X}_t$, $\mathcal{I}_t$, ρ_u, ρ_v, $\overline{\boldsymbol{U}}$, $\overline{\boldsymbol{v}}$, `max_ite`)
        • FT($\boldsymbol{X}_t$, $\mathcal{I}_t$, ϵ, ρ_v, $\overline{\boldsymbol{U}}$, $\overline{\boldsymbol{v}}$, `max_ite`)
        • ZT($\boldsymbol{X}_t$, $\mathcal{I}_t$, ρ_u, $\overline{\boldsymbol{U}}$, $\overline{\boldsymbol{v}}$, `max_ite`)
    if $t > P$ **then**
        $\boldsymbol{P}_t \leftarrow [\boldsymbol{v}_{t-1}, \ldots, \boldsymbol{v}_{t-P}]$
        $\boldsymbol{r}_{l,t} \leftarrow \boldsymbol{r}_{l,t-1} + \boldsymbol{P}_t^\top \boldsymbol{P}_t$
        $\boldsymbol{r}_{r,t} \leftarrow \boldsymbol{r}_{r,t-1} + \boldsymbol{P}_t^\top \boldsymbol{v}_t$
        $\boldsymbol{\theta}_t \leftarrow \boldsymbol{r}_{l,t}^{-1} \boldsymbol{r}_{r,t}$
    end if
end for
†If $t = 1$ set $\overline{\boldsymbol{U}} = \boldsymbol{0}$ and $\overline{\boldsymbol{v}} = \boldsymbol{0}$. If $1 < t < P$ set $\overline{\boldsymbol{v}} = \boldsymbol{v}_{t-1}$. Otherwise use the update in Line 7.

Then, at any given time t have $\widehat{\boldsymbol{\theta}}_t = \boldsymbol{r}_{l,t}^{-1} \boldsymbol{r}_{r,t}$. We now have a fully online algorithm for both factorizing the incoming data matrix and estimating the AR coefficients for the compressed time series.

We summarize the entire algorithm in Algorithm 4. At a given time, dropping the time index, let $s = \text{nnz}(\boldsymbol{x})$ be the number of observed entries, noting that $s > d$ and $P > d$ typically, the complexity of all three proposed algorithms is $\mathcal{O}(I_1 d^2 s + P^3)$ where I_1 is the number of iterations in the factorization.

V. Experiments

We test our proposed methodology using two time-series datasets downloaded from the UCI machine learning repository:

- Electricity:[4] Hourly power consumption (Mega Watts) of 370 customers between Jan. 1, 2012 to Jan. 1, 2015 in Portugal. This gives a matrix of 370 rows and 26,304 columns, with 9,732,480 entries.
- Traffic:[5] Hourly occupancy rates of 963 roads in Bay Area, California, recorded between Jan. 1, 2008 and Mar. 30, 2009. This matrix has 963 rows and 10,560 columns, giving 10,169,280 entries.

For both datasets there are no missing values. We generate missing data in two ways: (i) unstructured sparsity where at each time step the corresponding column of $\boldsymbol{X}$ is uniformly subsampled; (ii) structured sparsity where the sparsity for a row follows a geometric process with certain arrival/departure rates. The forecasting task is to predict the entries at given time step in the future.

We compare with several approaches:

- *Base:* This is a base estimator, which estimates the current value as the last observation. If the observation at previous time is missing, then it predicts the average of the last observed vector.
- *AR(P):* This is simply the AR model of (1), implemented on the vector observations. We learn the model in an online manner using the LMMSE estimator derived in Section IV.
- *PMF:* Probabilistic matrix factorization algorithm [1]. To extend PMF to the online setting, the FP cost function in Eq. (6) is used, but of course no AR structure is imposed.
- *CKF:* Collaborative Kalman Filter [21]. This is an online approach to matrix factorization problem, where the latent states follow a Brownian motion.
- *ORP:* Online robust PCA. The low dimensional time series v_t is estimated by the unconstrained principal component pursuit method described in [27].
- *GRA:* Grassmannian robust adaptive subspace tracking. This time v_t is estimated by the ADMM approach proposed in [23].
- *NMF:* Online non-negative matrix factorization. The implementation is similar to PMF, except that projection steps are added for the updates of U_t and v_t to ensure the factors are nonnegative.
- *Naive MF:* This is not a competitive algorithm; it corresponds to the model in (8).
- *FP-MF:* Fixed penalty matrix factorization (Sec. III-A).
- *FT-MF:* Fixed tolerance matrix factorization (Sec. III-B).
- *ZT-MF:* Zero tolerance matrix factorization (Sec. III-C).

Among these, the Base and AR(P) are not designed to handle missing or low rank data. Imputation based on the entire data is not possible in an online setting. We found the best-performing approach to be to impute the missing values at time t with the average of observed entries at that time, and use this value as the forecasts for time $t + 1$. While low rank methods are better at handling missing values, as our experiments show, there are also cases where this imputation strategy can be effective (see Fig. 5(b)).

In terms of the computational complexity, recall that all three of our proposed approaches have $\mathcal{O}(I_1 d^2 s + P^3)$ cost per time step. For the others we have: AR(P) is $\mathcal{O}(P^2 M)$; PMF, CKF, and NMF are $\mathcal{O}(I_1 d^2 s)$; ORP and GRA are $\mathcal{O}(I_1 I_2 d^2 s)$. Here, I_1 is the number of iterations run to update $\boldsymbol{U}_t$ and $\boldsymbol{v}_t$. In addition for ORP and GRA there is an inner loop for optimizing $\boldsymbol{v}$, using PCP and ADMM respectively; I_2 denotes the number of times this inner loop is done. Compared to other algorithms, the AR update step we introduced incurs a cost of P^3. Practically, $I_1 \approx P$ and $d^2 \geq P$, therefore the two terms $I_1 d^2 s$ and P^3 are comparable in value. Consequently, the computational complexity of the three proposed approaches is similar to competing methods. We use Matlab for implementation on a CPU, and note that the runtime for each algorithm considered is several minutes to process the entire data, for both datasets. Therefore, all approaches considered for real-time applications.

[4]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[5]https://archive.ics.uci.edu/ml/datasets/PEMS- SF

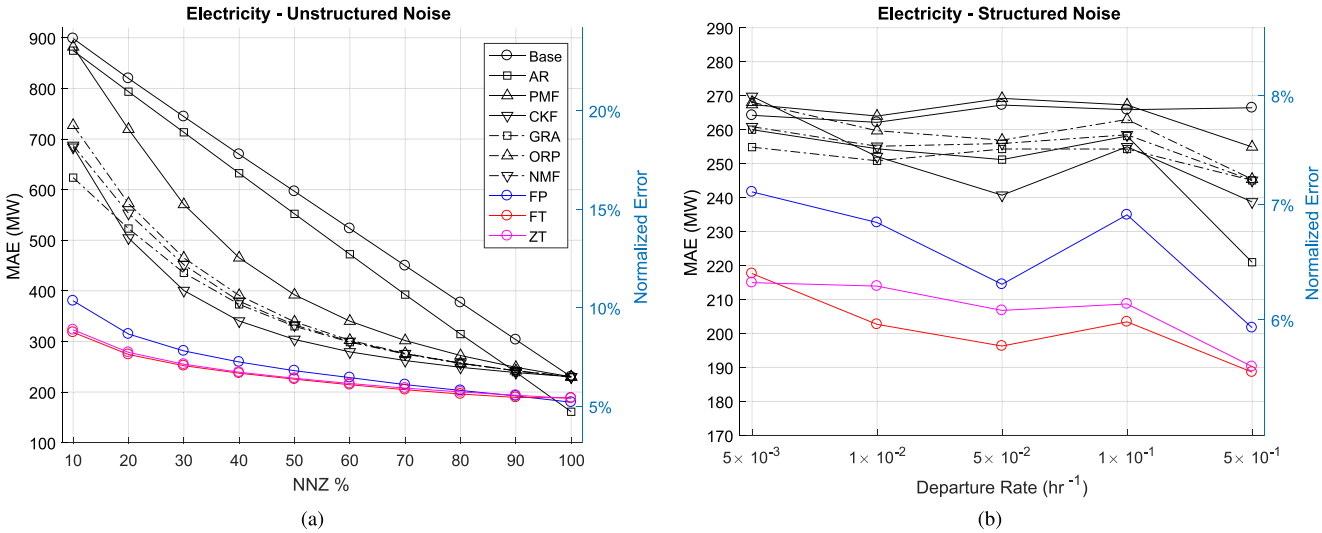

Fig. 2. Performance comparison of 10 predictors listed in the beginning of this section, for the electricity dataset. (a) The sparsity pattern is unstructured, and 20 sets of experiments are performed for 10 different levels. (b) The sparsity pattern is structured, and 20 sets of experiments are performed for 5 different departure rates.

For performance evaluation we use the mean absolute error (MAE) of forecasts one time-step ahead. For a time series with missing observations this is defined as:

$$\epsilon_{\text{MAE}} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{\ell(\boldsymbol{x}_t)}\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\|_1, \tag{34}$$

where $\boldsymbol{x}_t$ is the observation at time t, $\hat{\boldsymbol{x}}_t$ is its forecast, and $\ell(\boldsymbol{x}_t)$ is the number of observations.

A. Results on Electricity Data

For this set of experiments, the tunable parameters of each algorithm is set to:

- AR: $P = 24$, $r_0 = 1$
- PMF: $d = 5$, $\rho_u = 1$, $\rho_v = 10^{-4}$
- CKF:[6] $d = 5$, $\nu_d = 10^{-4}$, $\nu_x = 10^{-4}$
- Naive MF: $d = 5$, $\rho_u = 1$, $\rho_v = 10^{-4}$
- FP-MF: $d = 5$, $\rho_u = 1$, $\rho_v = 10^{-4}$, $P = 24$, $r_0 = 1$
- FT-MF: $d = 5$, $\epsilon = 0.05$, $\rho_v = 10^{-4}$, $P = 24$, $r_0 = 1$
- LN-MF: $d = 5$, $\rho_v = 10^{-4}$, $P = 24$, $r_0 = 1$
- `max_ite` $= 15$ for all algorithms.

These values were found by cross-validation. We observe that the parameters shared by different algorithms ended up with the same values, which indicates any difference in performance is due to the model structure, rather than parameter settings.

We first show results for one-step ahead prediction when the missingness pattern is *unstructured* (i.e., totally random). Unstructured sparsity is important in that it makes the learning environment adversarial and algorithms that are unfit for missing values are strongly affected. For our experiments we use 10 different sparsity levels, letting the percentage of observed entries vary from 10% to 100% in 10% increments. We abbreviate this as number of non-zeros (NNZ) as a percentage. For each NNZ level we assess the performance using mean absolute error(MAE) which is in MegaWatts (MW). We do this for all methods, averaging over 20 sets to ensure statistical significance.

In Fig. 2(a) we show the one-step ahead prediction performance of all algorithms. When there are no missing observations (NNZ $= 100\%$) the AR model has the best performance, but as NNZ decreases, the performance of AR quickly deteriorates and the three proposed algorithms give the best prediction. This transition already has taken place when NNZ $\leq 90\%$. As the sparsity increases, the base predictor and AR suffer the most. On the other hand, PMF and CKF perform better because they utilize the low-rank representation to impute. Finally, FP/FT/ZT utilize both low-rank and temporal regularization, yielding the best results. Their prediction suffers significantly less than other models as a function of increasing sparsity. FT and ZT perform better than FP, showing that adaptive regularization is indeed useful. We also note that, in general, while most of the online algorithms are concerned with finding factors sequentially, we incorporate an AR process to the generative model in Equation (3). This way, while the other online approaches can also find good embeddings sequentially, their predictive power is still limited as these models do not consider dependencies over multiple time lags.

We next experiment with structured sparsity patterns, which is not as adversarial as the previous case. Here, the missing values corresponds to the arrivals of a random process. We use a geometric distribution to generate arrival/departure points for missingness. This sparsity pattern could represent sensor failures or down times. A higher arrival rate indicates increased susceptibility to failure. For the electricity data we set the arrival rate to 0.05 and departure takes values in $\{0.005, 0.01, 0.05, 0.1, 0.5\}$. A higher departure rate means lower sparsity.

In Fig. 2(b) the prediction MAE is shown as a function of departure rates. An immediate observation is that, even if the

[6] ν_d and ν_x are the drift and measurement noise variance respectively [21].

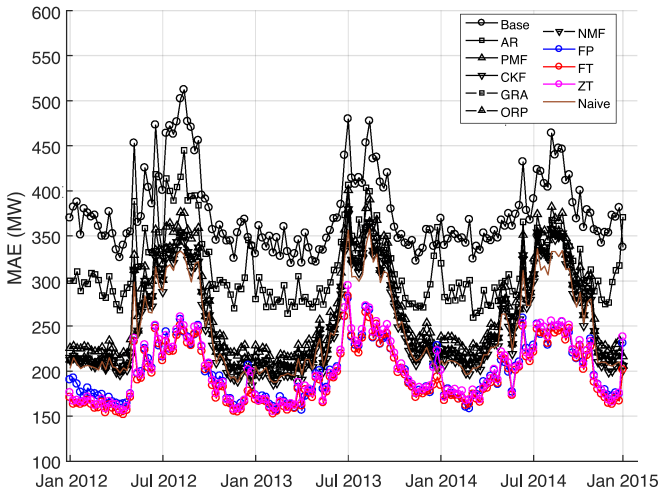

Fig. 3. Time-varying comparison of all models on electricity data with unstructured sparsity and NNZ $= 80\%$. While naive MF can forecast better than PMF, it is worse than FP/FT/ZT. Overall, FP/FT/ZT consistently outperform the other methods over time, which agrees with the results of Fig. 2.

sparsity is high (92% when departure rate is 0.005) none of the algorithms deteriorate as much as they do in Fig. 2(a). Once again, FT has the best performance, and the margin between FT, ZT and FP is more noticeable. On the other hand, the impute-predict scheme of the baseline predictor and the AR predictor also produce acceptable results.

In Section III we mentioned that rotation and scaling of factor matrices have an important impact on performance. The main reason is, using $\overline{\boldsymbol{U}}$ in regularization encourages smooth variation. The alternative regularization in Eq. (8) does not have this feature, as the penalty term on $\boldsymbol{U}_t$ is centered around zero matrix. Since this constraint does not encourage smoothness, it is expected to do worse. We provide evidence for this in Fig. 3. Here, while the AR model still lets the naive factorization to forecast better than PMF, it is clearly inferior to our methods. This plot also shows that, FP/FT/ZT not only outperform their competitors; but they do so consistently over time. Here once again, mean absolute error (MAE) is computed and plotted over time; in particular, each point in the plot corresponds to a one-week block, which is given by $\epsilon_{\text{MAE}}^{\text{week}} = \frac{1}{T_{\text{week}}} \sum_{t=1}^{T_{\text{week}}} \frac{1}{\ell(\boldsymbol{x}_t)} \|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\|_1$. This plot also gives more information about the electricity data itself. In particular, we observe that all algorithms have higher forecast error during summer times, which indicates electric usage during this season is harder to predict in advance. (This data is collected in Portugal.)

Dimensionality and AR order are the two most important parameters which determine how the matrix factorization forecasting performs. We examine performance as a function of these two parameters in Fig. 4.

In Fig. 4(a) we plot the performance as a function of latent dimensionality for unstructured noise with 80% observed entries. Here we show results for PMF as well as FP, FT, and ZT. First note that a dimension of one gives worst results for all. This shows the optimum rank is indeed greater than one, and matrix factorization is a suitable approach. The choice $d = 10$ produces best results, although we have used $d = 5$ for our other

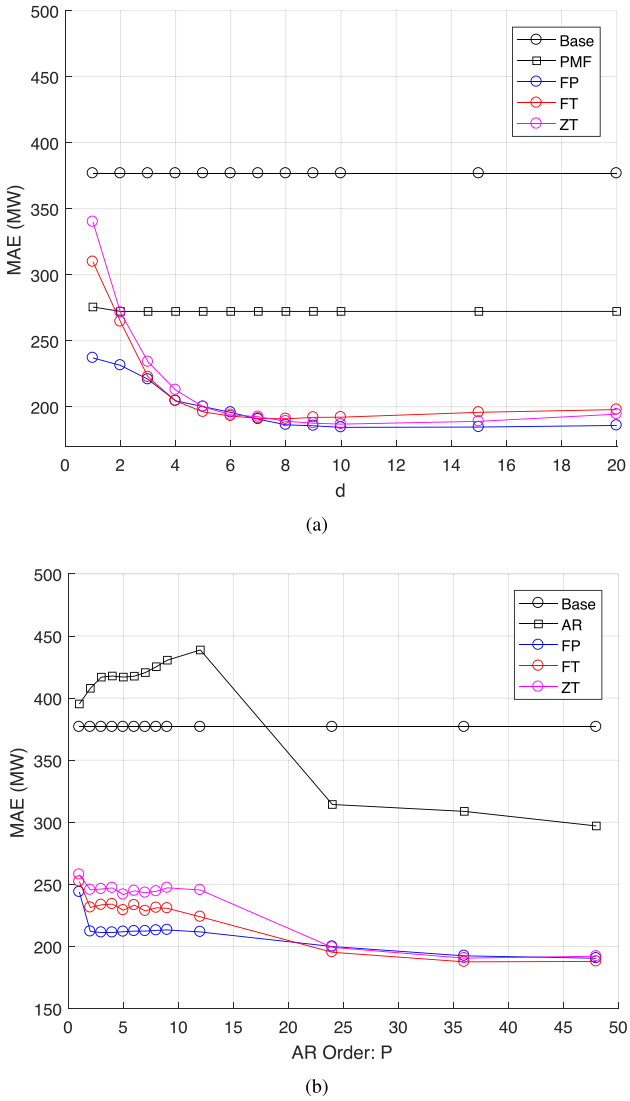

Fig. 4. Plot of prediction performances as a function of (a) rank and (b) AR order. Both plots obtained for electricity dataset with unstructured sparsity and NNZ $= 80\%$.

experiments, which still produce reliable results with the added benefit of higher compression. When the dimensionality is set low, both FT and ZT perform worse because, as dimension decreases, the fixed or zero tolerance constraint becomes more restrictive, which hurts performance. The degradation for ZT is greater than FT, which is expected since it is a zero error constraint. Therefore, when d needs to be low, we can use FT instead of ZT with $\epsilon > 0$ to provide better factorization as it provides slackness.

In Fig. 4(b) we show results as a function of AR order where $P \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 24, 36, 48\}$. We note a jump in performance as P moves from 12 to 24. This makes sense because $P = 24$ (a 24 hour period) indicates a daily periodicity for power consumption. Another observation is, in the case of missing data, the AR model gives unreliable estimates for lower orders, which suggests a correct choice of model order is important when imputing missing values.

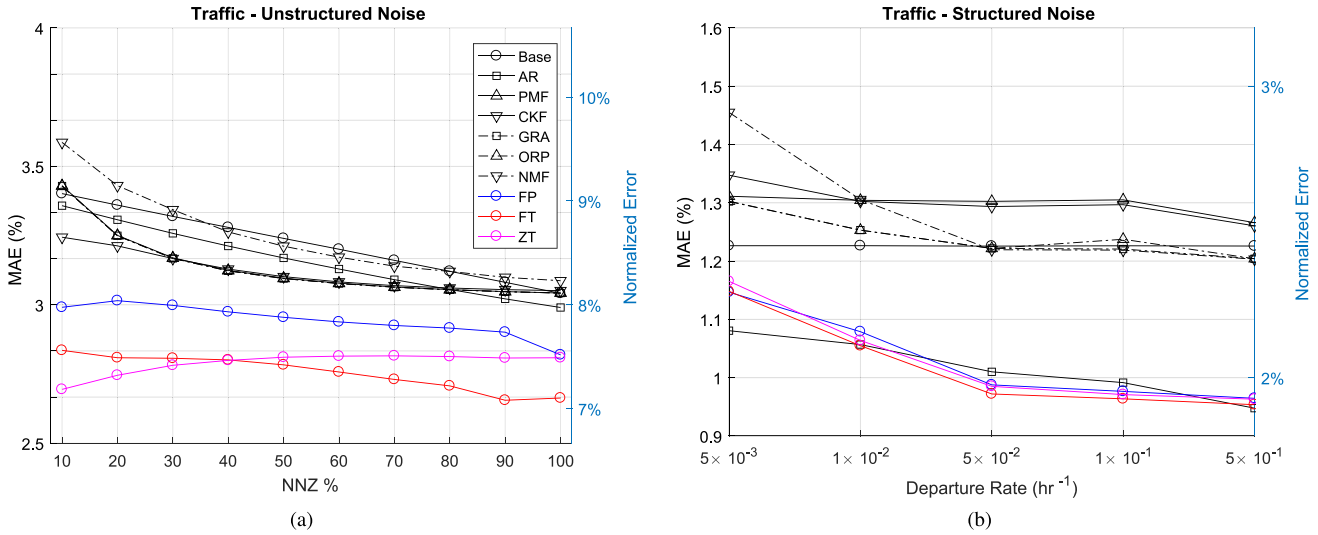

Fig. 5. Performance comparison of 10 predictors listed in the beginning of this section, for the traffic dataset. (a) The sparsity pattern is unstructured, and 20 sets of experiments are performed for 10 different levels. (b) The sparsity pattern is structured, and 20 sets of experiments are performed for 5 different departure rates.

B. Results on Traffic Data

For the traffic dataset, the parameter settings are:

- AR: $P = 24$, $r_0 = 1$
- PMF: $d = 20$, $\rho_u = 10^{-1}$, $\rho_v = 10^{-4}$
- CKF: $d = 20$, $\nu_d = 10^{-6}$, $\nu_x = 10^{-6}$
- Naive MF: $d = 20$, $\rho_u = 10^{-1}$, $\rho_v = 10^{-4}$
- FP-MF: $d = 20$, $\rho_u = 10^{-1}$, $\rho_v = 10^{-4}$, $P = 24$, $r_0 = 1$
- FT-MF: $d = 20$, $\epsilon = 0.05$, $\rho_v = 10^{-4}$, $P = 24$, $r_0 = 1$
- LN-MF: $d = 20$, $\rho_v = 10^{-4}$, $P = 24$, $r_0 = 1$
- $\texttt{max\_ite} = 15$ for all algorithms.

In particular, we increase d to 20 because of the increased dimensionality of the input data.

Once again, we consider structured and unstructured sparsity. The sparsity levels, arrival/departure rates, and the number of test sets are identical to what was used previously. In Fig. 5(a) the results for unstructured sparsity is shown. In this case, once again the best results are given by the proposed methods. On the other hand, Base and AR do not deteriorate as severely as fo the electricity data in Fig. 2(a). Also, PMF and CKF are no longer competitive on this data. In Fig. 5(b) we consider structured sparsity. This case is more unique from those previously considered. First, even for highly sparse inputs the performance of the base estimator does not deteriorate. Since Fig. 5(a) already shows that the sparsity does not have a very strong effect in adversarial case, the results for structured noise are not surprising. Since this is true for the base predictor, the AR predictor remains competitive as well. In fact, here the fill step is good enough to alleviate the missing data problem, so even if the data is sparse, the AR predictor can provide accurate forecasts. If we instead filled missing entries with zeros, this apparent advantage of AR disappears. Nevertheless, the difference between AR and FP/FT/ZT is small.

Similar to electricity data, we analyze the prediction performance as a function of rank and AR order in Fig. 6. In Fig. 6(a) we consider the effect of latent dimensionality. Unlike the electricity data, choosing $d = 1, 2$ resulted in unstable performance for FT and ZT because for this data choosing such a low rank is inappropriate. We therefore sweep $d \in \{5, 10, 15, 20, 30, 40\}$ and observe once $d \geq 10$ all factorizations produce consistent results. Once again we note that ZT is more susceptible to error compared to FT when dimension is low, as the zero tolerance constraint is more restrictive. In Fig. 6(b) we show the effect of AR order. Here the results are similar to the electricity data; setting $P = 24$ yields good results for FP, FT, and ZT. Once again, a one day periodicity is reasonable, since traffic intensity has a daily pattern, e.g. rush hours in the morning and evening.

VI. Forecasts on Individual Time Series

In this section we provide supplemental plots where we show the forecast values compared against the actual time series. Since the original series contains $\sim 10^4$ samples we use a sampling rate of 200 for electricity and 100 for traffic.

In Figs. 7 and 8 we show the prediction accuracy on two individual time series of the electricity data. These correspond to customers 142 and 309. Each plot is a 3×3 grid. The columns correspond to three filters AR(P), CKF, and FT; and the rows are in increasing (decreasing) sparsity (number of non-zero (NNZ) percentage). We show normalized plots to make comparisons across different series easier. The results do not change for the unnormalized case; everything is simply multiplied by a constant number. We use unstructured noise as there are bigger differences between filters as a function of NNZ for this case. For series number 142 we see that AR has the worse deterioration, losing track for NNZ $= 20\%$. CKF, being a low-rank method, has good performance for low sparsity as well, but we can see overshooting at peaks for NNZ $= 60\%$ and NNZ $= 20\%$. Finally, FT has good predictions for all sparsity levels, without such overshooting. For series number 309, once again AR deteriorates the most. CKF does better although the deterioration is particularly visible for NNZ $= 60\%$ and NNZ $= 20\%$. Again FT has best performance.

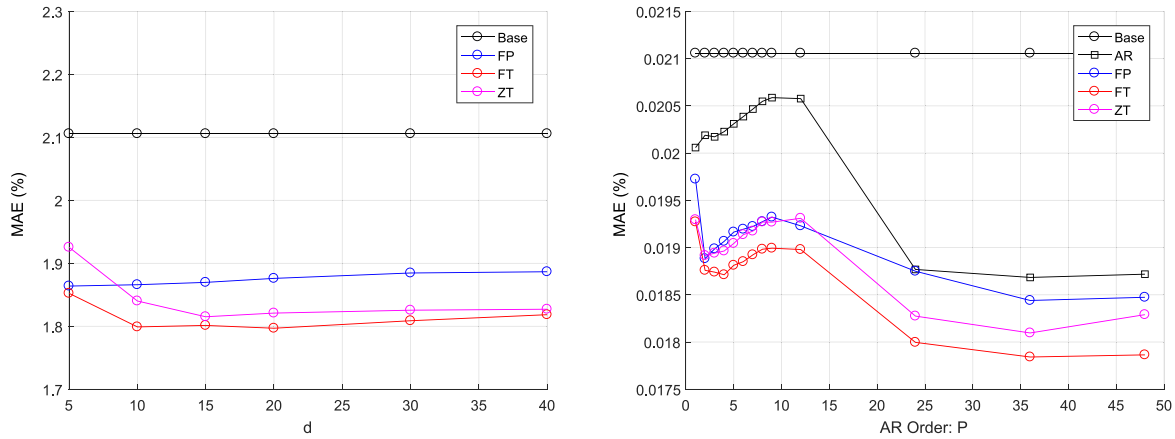

Fig. 6. Plot of prediction performances as a function of (a) rank and (b) AR order. Both plots obtained for traffic dataset with unstructured sparsity and NNZ = 50%.

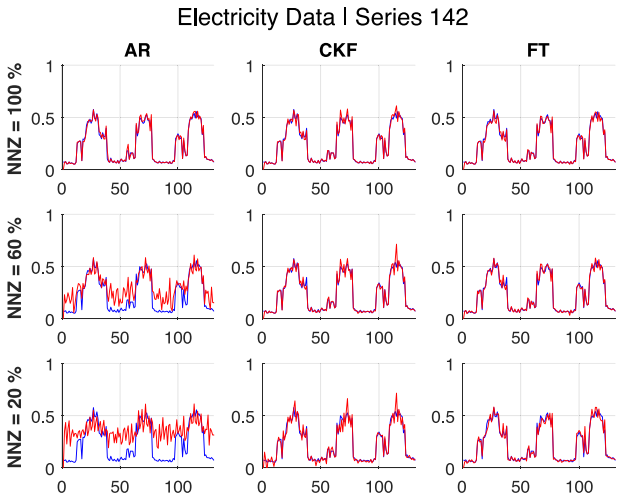

Fig. 7. Plot of time series no. 142 of Electricity data (blue) vs. forecasts (red) of AR, CKF, and FT for three sparsity levels, expressed in NNZ percentage.

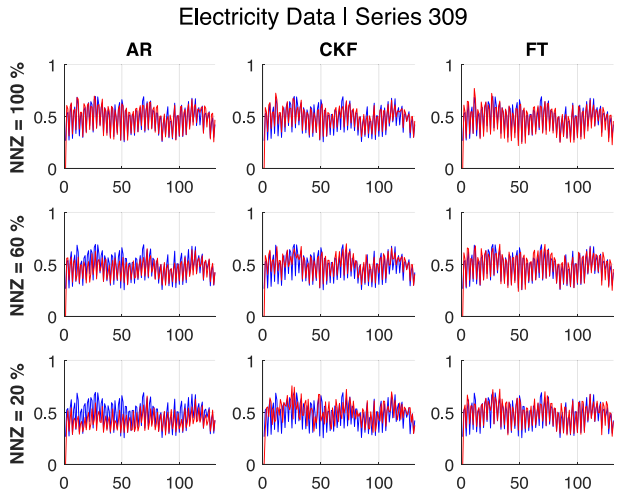

Fig. 8. Plot of time series no. 309 of Electricity data (blue) vs. forecasts (red) of AR, CKF, and FT for three sparsity levels, expressed in NNZ percentage.

Fig. 9. Plot of time series no. 290 of Traffic data (blue) vs. forecasts (red) of CKF, ORP, and LN for three sparsity levels, expressed in NNZ percentage.

Fig. 10. Plot of time series no. 704 of Traffic data (blue) vs. forecasts (red) of CKF, ORP, and LN for three sparsity levels, expressed in NNZ percentage.

In Figs. 7 and 8 we show the prediction accuracy on two individual time series of the traffic data. Here individual series correspond to occupancy rate of different roads. Once again unstructured noise is used and we compare CKF, ORP, and ZT. For both plots we see that where CKF and ORP has overshoots or larger errors, ZT performs better. However, compared to electricity data the differences between the three methods are smaller, and deterioration as a function of NNZ is also less severe.

VII. Conclusion

We have considered the problem of forecasting values of high dimensional time series. A high dimensional time series can be treated as a matrix, where each column denotes a cross-section at a particular time. And when missing values are present, a low rank matrix factorization can be used as a building block for a forecasting that also imputes missing values. Based on this idea, we proposed three methods which can perform matrix factorization in the online/streaming setting. These approaches differ by the type of regularization imposed, and an key conclusion is that time varying regularization can be achieved through a constrained optimization problem.

The matrix factorization component provides a low dimensional representation, which are then used to learn an AR model on next values in this low dimensional space. This in turn forms the basis of forecasting. We derived the optimum LMMSE estimator to find these AR coefficients that only requires the first and second order statistics of the noise terms. Finally, we considered two real datasets on electric power demand and traffic, and showed that when missing values are present in the data our methods will provide more reliable forecasts. Future developments of this technique could include the problem where entire columns of data are missing, which correspond system-wide blackouts, and also alternative factorization approaches that do not require any explicit regularization.

Appendix A
Fixed Tolerance Update: $\boldsymbol{U}_t$

At any given iteration-i of the E-step in Algorithm 2, for a fixed $\boldsymbol{v}_t^{(i)}$, we want to find

$$\boldsymbol{U}_t^{(i)} = \arg\min_{\boldsymbol{U}} \|\boldsymbol{U} - \overline{\boldsymbol{U}}\|_F^2 \text{ s.t. } \|\boldsymbol{x}_t - \boldsymbol{U}^\top \boldsymbol{v}_t^{(i)}\|_2^2 \leq \epsilon. \quad (35)$$

Note the indexing, as $\boldsymbol{U}_t^{(i)}$ is computed after $\boldsymbol{v}_t^{(i)}$, as the latter appears before the former in the loop in Algorithm 2. To avoid clutter we will drop the time and iteration indices. We will also use $\overline{\boldsymbol{U}} = \boldsymbol{U}_{t-1}$ to distinguish the time indexing. From Eq. (35) it is seen that the problem is strongly convex. To show this, let $\mathcal{I}_t$ denote the index of observed $\boldsymbol{x}_t$ entries, then

$$\begin{aligned} &\min_{\{\boldsymbol{u}_i\}_{i\in\mathcal{I}_t}} \sum_{i\in\mathcal{I}_t} \|\boldsymbol{u}_i - \overline{\boldsymbol{u}}_i\|_2^2 \text{ s.t. } \sum_{i\in\mathcal{I}_t} (\boldsymbol{x}_i - \boldsymbol{u}_i^\top \boldsymbol{v})^2 \leq \epsilon \\ \equiv &\min_{\{\boldsymbol{u}_i\}_{i\in\mathcal{I}_t}} \sum_{i\in\mathcal{I}_t} \boldsymbol{u}_i^\top \boldsymbol{I} \boldsymbol{u}_i - 2\boldsymbol{u}_i^\top \overline{\boldsymbol{u}}_i + \overline{\boldsymbol{u}}_i^\top \overline{\boldsymbol{u}}_i \\ &\text{s.t. } \sum_{i\in\mathcal{I}_t} \boldsymbol{u}_i^\top \boldsymbol{v}\boldsymbol{v}^\top \boldsymbol{u}_i - 2x_i \boldsymbol{u}_i^\top \boldsymbol{v} + x_i^2 \end{aligned} \quad (36)$$

and note that both $\boldsymbol{I}$ and $\boldsymbol{v}\boldsymbol{v}^\top$ are positive semidefinite. Moreover the feasible set is always nonempty. In fact, for any given $\boldsymbol{v}_t^{(i)}$, the feasible set will have infinitely many elements. To see this, let $\epsilon = 0$ and note that this gives M_t linear equations in dM_t unknowns. Since all of these solutions lie within the feasible set, the result follows. As a consequence, Slater's condition is satisfied and strong duality holds for this problem and we can characterize the solution via Karush Kuhn Tucker (KKT) conditions [39]. For this problem these conditions read as

1) $\nabla_{\boldsymbol{U}^\star} \mathcal{L} = 0$
2) $\|\boldsymbol{x} - \boldsymbol{U}^{\star\top} \boldsymbol{v}\|_2^2 \leq \epsilon$
3) $\lambda^\star \geq 0$
4) $\lambda^\star [\|\boldsymbol{x} - \boldsymbol{U}^{\star\top} \boldsymbol{v}\|_2^2 - \epsilon] = 0$

Using the first condition, which needs to hold for primal feasibility, it is easily shown that the solution satisfies

$$\boldsymbol{U} = [\lambda^{-1}\boldsymbol{I} + \boldsymbol{v}\boldsymbol{v}^\top]^{-1}(\lambda^{-1}\overline{\boldsymbol{U}} + \boldsymbol{v}\boldsymbol{x}^\top),$$

which can be re-written, using Sherman-Morrison identity, as

$$\begin{aligned} \boldsymbol{U} = \overline{\boldsymbol{U}} + \lambda \boldsymbol{v}\boldsymbol{x}^\top - \frac{\lambda}{1+\lambda \boldsymbol{v}^\top \boldsymbol{v}} (\boldsymbol{v}\boldsymbol{v}^\top)\overline{\boldsymbol{U}} \\ - \frac{\lambda^2}{1+\lambda \boldsymbol{v}^\top \boldsymbol{v}} (\boldsymbol{v}\boldsymbol{v}^\top)(\boldsymbol{v}\boldsymbol{x}^\top) \end{aligned} \quad (37)$$

This alternative expression is useful, as the inverse term containing λ disappears. Now, the third and fourth KKT conditions imply

$$\|\boldsymbol{x} - \boldsymbol{U}^\top \boldsymbol{v}\|_2^2 = \epsilon. \quad (38)$$

This also satisfy condition number two. This means, the solution of the optimization problem has an approximation error that is equal to the maximum tolerance ϵ. As $\boldsymbol{U}$ is always updated after $\boldsymbol{v}$ this means the training error of our algorithm at *each* time step will be $\sqrt{\epsilon}$. This also suggests, we can find the value of Lagrange multiplier by plugging Eq. (37) into Eq. (38). Now let the constants c_1–c_4 be as in Eq. (14). We need to calculate the equation

$$\boldsymbol{v}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{v} - 2\boldsymbol{x}^\top \boldsymbol{U}^\top \boldsymbol{v} - (\epsilon - c_3) = 0.$$

The first two terms evaluate as

$$\begin{aligned} \boldsymbol{v}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{v} &= c_4 + \lambda c_1 c_2 - \frac{\lambda}{1+\lambda c_2} c_2 c_4 - \frac{\lambda^2}{1+\lambda c_2} c_1 c_2^2 \\ &+ \lambda c_1 c_2 + \lambda^2 c_2^2 c_3 - \frac{\lambda^2}{1+\lambda c_2} c_1 c_2^2 - \frac{\lambda^3}{1+\lambda c_2} c_2^3 c_3 \\ &- \frac{\lambda}{1+\lambda c_2} c_2 c_4 - \frac{\lambda^2}{1+\lambda c_2} c_1 c_2^2 + \frac{\lambda^2}{(1+\lambda c_2)^2} c_2^2 c_4 \\ &+ \frac{\lambda^3}{(1+\lambda c_2)^2} c_1 c_2^3 - \frac{\lambda^2}{1+\lambda c_2} c_1 c_2^2 - \frac{\lambda^3}{1+\lambda c_2} c_2^3 c_3 \\ &+ \frac{\lambda^3}{(1+\lambda c_2)^2} c_1 c_2^3 + \frac{\lambda^4}{(1+\lambda c_2)^2} c_2^4 c_3 \end{aligned} \quad (39)$$

$$-2\boldsymbol{x}^\top \boldsymbol{U}^\top \boldsymbol{v} = -\frac{2c_1}{1+\lambda c_2} - \frac{2\lambda c_2 c_3}{1+\lambda c_2} \quad (40)$$

Multiplying with the denominator term $(1+\lambda c_2)^2$ and expanding the terms, the final expression is simply a second order

polynomial

$$-\epsilon c_2^2\lambda^2 - 2\epsilon c_2\lambda + (c_3 + c_4 - 2c_1 - \epsilon). \tag{41}$$

The roots are then given by the formula

$$-\frac{1}{c_2} \pm \frac{1}{\sqrt{\epsilon}c_2}\sqrt{c_3 + c_4 - 2c_1} \tag{42}$$

Since $c_2 = \|\boldsymbol{v}\|_2^2 > 0$ the first term is negative. Then the third KKT condition implies, the second term above must be greater than the first term, and thus the polynomial always has one positive and one negative root. The update for the Lagrange multiplier in Eq. (15) now follows; note that in that equation we only used two variables defined in Eq. (14) for conciseness.

Appendix B
Fixed Tolerance Update: $\boldsymbol{v}_t$

Similar to Appendix A we study iteration-i of the E-step in Algorithm 2; but this time for a fixed $\boldsymbol{U}_t^{(i)}$, we want to find

$$\boldsymbol{v}_t^{(i+1)} = \arg\min_{\boldsymbol{v}} \|\boldsymbol{v} - \boldsymbol{v}_{t-1}\|_2^2 \text{ s.t. } \|\boldsymbol{x}_t - \boldsymbol{U}_t^{(i)}\boldsymbol{v}_t\|_2^2 \leq \epsilon. \tag{43}$$

Unlike the case for $\boldsymbol{U}_t$, it is not clear if the solution set is nonempty. In particular, note that in Appendix A, we showed that no matter how $\boldsymbol{v}_t$ is chosen, the solution set always has infinitely many elements. For $\boldsymbol{v}_t$ this is not true in general. To see this, consider the case where $\boldsymbol{U}_t^\top \boldsymbol{v} = \boldsymbol{x}$ is an over constrained system of linear equations. The minimum error achievable in this case is given by the least squares solution as $\epsilon_{\text{ls}} = \boldsymbol{x}^\top[\boldsymbol{I} - \boldsymbol{U}_t^\top(\boldsymbol{U}_t^\top\boldsymbol{U}_t)^{-1}\boldsymbol{U}_t]\boldsymbol{x}$. When $\epsilon_{\text{ls}} > \epsilon$ there are no solutions. However, if $\boldsymbol{U}_t$ is updated before $\boldsymbol{v}_t$, then it is guaranteed that the there is at least a single solution; because for the alternating optimization in Eq. (43), we are given that $\|\boldsymbol{U}_t^{(i)} - \boldsymbol{v}_t^{(i-1)}\|_2^2 < \epsilon$. As a result, $\boldsymbol{v}_t^{(i-1)}$ is guaranteed to be in the feasible set. Therefore the solution set is nonempty and since the problem is strongly convex, strong duality holds once again due to Slater's condition. The KKT conditions mirror the previous one:

1) $\nabla_{\boldsymbol{v}^\star}\mathcal{L} = 0$
2) $\|\boldsymbol{x} - \boldsymbol{U}^\top\boldsymbol{v}^\star\|_2^2 \leq \epsilon$
3) $\lambda^\star \geq 0$
4) $\lambda^\star[\|\boldsymbol{x} - \boldsymbol{U}^\top\boldsymbol{v}^\star\|_2^2 - \epsilon] = 0$

Letting $\overline{\boldsymbol{v}} = \boldsymbol{v}_{t-1}$ the first condition yields

$$\boldsymbol{v} = [\lambda^{-1}\boldsymbol{I} + \boldsymbol{U}\boldsymbol{U}^\top]^{-1}[\lambda^{-1}\overline{\boldsymbol{v}} + \boldsymbol{U}\boldsymbol{x}]. \tag{44}$$

This time, we cannot apply Sherman-Morrison identity, as $\boldsymbol{U}$ is typically not rank-1. Instead, note that $\boldsymbol{U}\boldsymbol{U}^\top$ is in the positive semidefinite cone of symmetric matrices, and admits an eigendecomposition with nonnegative eigenvalues. Let's denote this by $\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{Q}\boldsymbol{\Psi}\boldsymbol{Q}^\top$, and define the constant vectors $\boldsymbol{c}_1 = \boldsymbol{Q}^\top\boldsymbol{U}\boldsymbol{x}$ and $\boldsymbol{c}_2 = \boldsymbol{Q}^\top\overline{\boldsymbol{v}}$. Then it is straightforward to verify that

$$\boldsymbol{v} = \boldsymbol{Q}[\rho\boldsymbol{I} + \boldsymbol{\Psi}]^{-1}\boldsymbol{c}_1 + \boldsymbol{Q}\rho[\rho\boldsymbol{I} + \boldsymbol{\Psi}]^{-1}\boldsymbol{c}_2 \tag{45}$$

where we defined $\rho = 1/\lambda$; note the choice of notation here as ρ is the regularizer, analogous to the one in FP matrix factorization. Once again, from the fourth KKT condition we want to solve the equation

$$\boldsymbol{v}^\top\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{v} - 2\boldsymbol{x}^\top\boldsymbol{U}^\top\boldsymbol{v} - (\epsilon - c_3) = 0,$$

where the first two terms are evaluated as

$$\begin{aligned}\boldsymbol{v}^\top\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{v} &= \sum_{i=1}^d \frac{\psi_i}{(\rho+\psi_i)^2}c_{1,i}^2 + \sum_{i=1}^d \frac{\rho^2\psi_i}{(\rho+\psi_i)^2}c_{2,i}^2 \\ &\quad + 2\frac{\rho\psi_i}{(\rho+\psi_i)^2}c_{1,i}c_{2,i} \\ -2\boldsymbol{x}^\top\boldsymbol{U}^\top\boldsymbol{v} &= -2\sum_{i=1}^d \frac{1}{\rho+\psi_i}c_{1,i}^2 - 2\sum_{i=1}^d \frac{\rho}{\rho+\psi_i}c_{1,i}c_{2,i}\end{aligned} \tag{46}$$

Substituting these into the equation and multiplying with the denominator term $\prod_{j=1}^d(\rho+\psi_i)^2$ we get the following result.

Remark: The FT update for $\boldsymbol{v}_t$ and the ridge regression update in Eq. (13) are the same when λ is selected as the root of the following polynomial yielding smallest $\|\boldsymbol{v}_t\|_2^2$

$$\begin{aligned}p(\lambda) &= \sum_{i=1}^d(-2\rho-\psi_i)c_{1,i}^2\prod_{j\neq i}(\rho+\psi_j)^2 + \sum_{i=1}^d\rho^2\psi_i c_{2,i}^2 \\ &\quad\times\prod_{j\neq i}(\rho+\psi_j)^2 - 2\rho^2\sum_{i=1}^d c_{1,i}c_{2,i}\prod_{j\neq i}(\rho+\psi_j)^2 \\ &\quad - (\epsilon - \boldsymbol{x}^\top\boldsymbol{x})\prod_{j=1}^d(\rho+\psi_j)^2\end{aligned} \tag{47}$$

References

[1] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.

[2] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov Chain Monte Carlo," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 880–887.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[4] M. Falahatgar, M. I. Ohannessian, and A. Orlitsky, "Near-optimal smoothing of structured conditional probability matrices," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4860–4868.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.

[6] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, 2010.

[7] A. Y. Aravkin, K. R. Varshney, and D. M. Malioutov, "A robust nonlinear Kalman smoothing approach for dynamic matrix factorization," IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, Tech. Rep., 2015.

[8] J. Mei, Y. De Castro, Y. Goude, and G. Hebrail, "Nonnegative matrix factorization for time series recovery from a few temporal aggregates," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, vol. 70, pp. 2382–2390.

[9] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 556–562.

[10] A. Lefevre, F. Bach, and C. Fevotte, "Online algorithms for nonnegative matrix factorization with the Itakura-Saito divergence," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2011, pp. 313–316.

[11] R. Zhao, V. Y. F. Tan, and H. Xu, "Online nonnegative matrix factorization with general divergences," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2016, vol. 54, pp. 37–45.

[12] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 847–855.

[13] O. Anava, E. Hazan, S. Mannor, and O. Shamir, "Online learning for time series prediction," in *Proc. Conf. Learn. Theory*, 2013, pp. 172–184.
[14] C. Liu, S. C. H. Hoi, P. Zhao, and J. Sun, "Online ARIMA algorithms for time series prediction," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1867–1873.
[15] W. M. Koolen, A. Malek, P. L. Bartlett, and A.-Y. Yasin, "Minimax time series prediction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2557–2565.
[16] O. Anava, E. Hazan, and A. Zeevi, "Online time series prediction with missing data," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, vol. 37, pp. 2191–2199.
[17] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME–J. Basic Eng.*, vol. 82, no. Ser. D, pp. 35–45, 1960.
[18] J. Sun, D. Parthasarathy, and K. Varshney, "Collaborative Kalman filtering for dynamic matrix factorization," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3499–3509, Jul. 12014.
[19] N. Mohammadiha, P. Smaragdis, G. Panahandeh, and S. Doclo, "A state-space approach to dynamic nonnegative matrix factorization," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 949–959, Feb. 2015.
[20] A. Y. Aravkin, K. R. Varshney, and L. Yang, "Dynamic matrix factorization with social influence," in *Proc. IEEE 26th Int. Workshop Mach. Learn. Signal Process.*, 2016, pp. 1–6.
[21] S. Gultekin and J. Paisley, "A collaborative Kalman filter for time-evolving dyadic processes," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 140–149.
[22] L. Xu and M. A. Davenport, "Dynamic matrix recovery from incomplete observations under an exact low-rank constraint," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3585–3593.
[23] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1568–1575.
[24] H. Guo, C. Qiu, and N. Vaswani, "An online algorithm for separating sparse and low-dimensional signal sequences from their sum," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4284–4297, Aug. 2014.
[25] C. Qiu, N. Vaswani, B. Lois, and L. Hogben, "Recursive robust PCA or recursive sparse recovery in large but structured noise," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 5007–5039, Aug. 2014.
[26] J. Zhan and N. Vaswani, "Robust PCA with partial subspace knowledge," in *Proc. IEEE Int. Symp. Inf. Theory*, 2014, pp. 2192–2196.
[27] J. Feng, H. Xu, and S. Yan, "Online robust PCA via stochastic optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 404–412.
[28] F. Han and H. Liu, "Transition matrix estimation in high dimensional time series," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, vol. 28, pp. 172–180.
[29] W. Dunsmuir and P. R. Robinson, "Estimation of time series models in the presence of missing data," *J. Amer. Statist. Assoc.*, vol. 76, no. 375, pp. 560–568, Sep. 1981.
[30] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *J. Time Ser. Anal.*, vol. 3, no. 4, pp. 253–264, 1982.
[31] M. K. Choong, M. Charbit, and H. Yan, "Autoregressive-model-based missing value estimation for DNA microarray time series data," *IEEE Trans. Inf. Technol. Biomed.*, vol. 13, no. 1, pp. 131–137, Jan. 2009.
[32] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE Trans. Autom. Control* vol. 49, no. 9, pp. 1453–1464, Sep. 2004.
[33] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*. New York, NY, USA: Springer, 2016.
[34] J. D. Hamilton, *Time Series Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 1994.
[35] J. E. Gentle, *Matrix Algebra: Theory, Computations, and Applications in Statistics*. New York, NY, USA: Springer, 2007.
[36] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.
[37] A. Y. Aravkin, J. V. Burke, and G. Pillonetto, "Optimization viewpoint on Kalman smoothing with applications to robust and sparse estimation," in *Compressed Sensing & Sparse Filtering*. New York, NY, USA: Springer, 2014.
[38] R. A. Horn and C. B. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[39] S. Boyd and L. Vanderberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[40] E. J. Candes and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, 2009, Art. no. 717.
[41] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2053–2080, May 2010.
[42] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, no. 3, pp. 471–501, 2010.
[43] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2009.

San Gultekin received the B.S. and M.S. degrees from the Electrical and Electronics Engineering Department, Bilkent University, Ankara, Turkey. He is currently a Ph.D. candidate and Armstrong Fellow in the Electrical Engineering Department, Columbia University, New York, NY, USA. His research interests are in statistical signal processing, machine learning, large-scale data analysis, and time-series analysis.

John Paisley received the B.S., M.S., and Ph.D. degrees in electrical engineering from Duke University, Durham, NC, USA. He was a Postdoctoral researcher with the Computer Science Departments at University of California, Berkeley and Princeton University. He is currently an Associate Professor with the Department of Electrical Engineering, Columbia University, New York, NY, USA, and also a member of the Data Science Institute, Columbia University. His current research is machine learning, focusing on models and inference techniques for text and image processing applications.