Manuscript version: Published Version
The version presented in WRAP is the published version (Version of Record).

Persistent WRAP URL:
http://wrap.warwick.ac.uk/148000

How to cite:
The repository item page linked to above, will contain details on accessing citation guidance from the publisher.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher’s statement:
Please refer to the repository item page, publisher’s statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk
Probabilistic Sequential Matrix Factorization

Ömer Deniz Akyildiz\textsuperscript{*,1} 
Gerrit J.J. van den Burg\textsuperscript{*,1} 
Theodoros Damoulas\textsuperscript{1,2} 
Mark F.J. Steel\textsuperscript{2} 
\textsuperscript{1}The Alan Turing Institute, London, UK 
\textsuperscript{2}University of Warwick, Coventry, UK

Abstract

We introduce the probabilistic sequential matrix factorization (PSMF) method for factorizing time-varying and non-stationary datasets consisting of high-dimensional time-series. In particular, we consider nonlinear Gaussian state-space models where sequential approximate inference results in the factorization of a data matrix into a dictionary and time-varying coefficients with potentially nonlinear Markovian dependencies. The assumed Markovian structure on the coefficients enables us to encode temporal dependencies into a low-dimensional feature space. The proposed inference method is solely based on an approximate extended Kalman filtering scheme, which makes the resulting method particularly efficient. PSMF can account for temporal nonlinearities and, more importantly, can be used to calibrate and estimate generic differentiable nonlinear subspace models. We also introduce a robust version of PSMF, called rPSMF, which uses Student-t filters to handle model misspecification. We show that PSMF can be used in multiple contexts: modeling time series with a periodic subspace, robustifying changepoint detection methods, and imputing missing data in several high-dimensional time-series, such as measurements of pollutants across London.

\textsuperscript{*} Joint first authorship.

1 INTRODUCTION

The problem of $r$-rank factorization of a data matrix $Y \in \mathbb{R}^{d \times n}$ as

$$Y \approx CX$$  \hspace{1cm} (1)

with $C \in \mathbb{R}^{d \times r}$ the dictionary matrix and $X \in \mathbb{R}^{r \times n}$ the coefficients, has received significant attention in past decades in multimedia signal processing and machine learning under the umbrella term of matrix factorization (MF) (Lee and Seung, 1999, 2001; Mairal et al., 2010). The classical method for solving problems of the form (1) is nonnegative matrix factorization (NMF) (Lee and Seung, 1999), which is proposed for nonnegative data matrices and obtains nonnegative factors. NMF and similar methods (e.g., singular value decomposition (SVD)) have been the focus of intensive research (Lin, 2007; Berry et al., 2007; Ding et al., 2008; Cai et al., 2010; Févotte and Idier, 2011) and has found applications in several fields, such as document clustering (Shahmaz et al., 2006), audio analysis (Smaragdis and Brown, 2003; Ozerov and Févotte, 2009), and video analysis (Bucak and Günel, 2009). This work was extended to general MF problems for real-valued data and factors, which found many applications including collaborative filtering (Rennie and Srebro, 2005) and drug-target prediction (Zheng et al., 2013).

The problem in (1) was originally tackled from an optimization perspective, i.e., minimizing a cost $d(Y; CX)$ over $C$ and $X$ (Lee and Seung, 1999, 2001; Lin, 2007; Mairal et al., 2010). Another promising approach has been through a probabilistic model by defining priors on $C$ and $X$. This was explored by, e.g., Cemgil (2009) with a Poisson-based model solved using variational inference (which reproduces NMF when the cost function is the Kullback-Leibler divergence) and in, e.g., Mnih and Salakhutdinov (2008) and Salakhutdinov and Mnih (2008) with a Gaussian model for the real-valued case solved via Markov chain Monte Carlo (MCMC). Naturally, online versions of these methods have received significant attention as they enable scaling up to
larger datasets. On this front, a number of algorithms have been proposed that are based either on stochastic optimization (e.g., Bucak and Günsel, 2009; Maïral et al., 2010; Gemulla et al., 2011; Mensch et al., 2016) or on a probabilistic model for online inference (Wang et al., 2012; Paisley et al., 2014; Akyildiz and Míguez, 2019). However, these methods are generally for i.i.d. data and cannot exploit the case where the columns of Y possess time dependency.

The success of MF in the i.i.d. data case motivated the development of matrix factorization methods for time-dependent data. In this case, the problem can be formulated as inferring parameters of a dynamical system or a state-space model (SSM), with a linear observation model C. This problem is also known as system identification (Katayama, 2006). When the columns of X (i.e., the hidden signal) also evolve linearly the problem reduces to that of inferring parameters of a linear SSM. This can be solved by maximum-likelihood estimation (MLE) through, e.g., expectation-maximization (EM) either offline (Ghahramani and Hinton, 1996; Elliott and Krishnamurthy, 1999) or online (Cappé and Moulines, 2009; Cappé, 2011), or with gradient-based methods (Andrieu et al., 2005; Kantas et al., 2015). In this vein, Yildirim et al. (2012) address the NMF problem by introducing a SSM with a Poisson likelihood where the inference is carried out with sequential Monte Carlo (SMC). In a similar manner Sun et al. (2012) propose an SSM-based approach where the dictionary is estimated using the EM algorithm. Similar MLE-based nonnegative schemes that also use SSMs attracted significant attention (e.g., Mohammadiha et al., 2013, 2014).

Although MLE estimates are consistent in the infinite data limit for general SSMs (Douc et al., 2011), EM-based methods are prone to get stuck in local minima (Katayama, 2006) and provide only point estimates rather than full posterior distributions. As an alternative to the EM-based approaches, optimization-based methods were also explored (e.g., Boots et al., 2008; Karami et al., 2017; White et al., 2015) which again result in point estimates. If the transition model for the coefficients X exhibits nonlinear dynamics while the observation model is linear (by the nature of MF), the problem reduces to parameter estimation in nonlinear SSMs (Särkkä, 2013). The MLE approach is again prominent in this setting using EM or gradient methods (Kantas et al., 2015). However, when inference cannot be done analytically this results in the use of SMC (Doucet et al., 2000) or particle MCMC methods (Andrieu et al., 2010) (see Kantas et al. (2015) for an overview). Unfortunately, these methods suffer in the high-dimensional case (Bengtsson et al., 2008; Snyder et al., 2008) which makes Monte Carlo-based methods unsuitable for solving the MF problem. Optimization-based approaches that formulate a cost function with temporal regularizers have also been studied (e.g., Yu et al., 2016; Shi et al., 2016; Liu and Hauskrecht, 2016; Ayed et al., 2019).

An alternative to the MLE or optimization-based approaches is to follow a Bayesian approach where a prior distribution is constructed over the parameters of the SSM, see, e.g., Särkkä (2013). The goal is then to obtain the posterior distributions of the columns of X and of C. This is also of interest when priors are used as regularizers to enforce useful properties such as sparsity (Cemgil, 2009; Schmidt et al., 2009). In this context, an extension of the NMF-like decompositions to the dynamic setting was considered by Févotte et al. (2013), where the authors followed a maximum-a posteriori (MAP) approach. We refer to Févotte et al. (2018) for a literature review of temporal NMF methods. However, these methods are batch (offline) schemes and do not return a probability distribution over the dictionary or the coefficients. Joint posterior inference of C and X in a fully Bayesian setting is difficult as it usually requires sampling schemes (Salakhutdinov and Mnih, 2008). To the best of our knowledge, a fully Bayesian approach for sequential (online) inference for matrix factorization that also scales well with the problem dimension has not been proposed in the literature.

Contribution. In this work, we propose the probabilistic sequential matrix factorization (PSMF) method by framing our matrix factorization model as a nonlinear Gaussian SSM. Our formulation is fully probabilistic in the sense that we place a matrix-variate Gaussian prior on the dictionary and use a general Markov model for the evolution of the coefficients. We then derive a novel approximate inference procedure that is based on extended Kalman filtering (Kalman, 1960; McLean et al., 1962) and results in a fast and efficient scheme. Our method is derived using numerical approximations to the optimal inference scheme and leverages highly efficient filtering techniques.

In particular, we derive analytical approximations and do not require a sampling procedure to approximate the posterior distributions. The inference method we provide is explicit and the update rules can be easily implemented without further consideration on the practitioner’s side. We also provide a robust extension of our model, called rPSMF, for the case where the model is misspecified and derive a corresponding inference scheme that adopts Student’s t-filters (Girón and Rojano, 1994; Tronarp et al., 2019). Our methods can be easily tailored to the application at hand by modifying the subspace model, as the necessary derivatives can be easily computed through automatic differentiation.
This work is structured as follows. In Sec. 2 we introduce our probabilistic state-space model and the robust extension. Next we develop our tractable inference and estimation method in Sec. 3. In Sec. 4, our method is empirically evaluated in different scenarios such as learning structured subspaces, multivariate changepoint detection, and missing data imputation. Sec. 5 concludes the paper.

**Notation** We denote the \(d \times d\) identity matrix by \(I_d\) and write \(\mathcal{N}(x; \mu, \Sigma)\) for the Gaussian density over \(x\) with mean \(\mu\) and covariance matrix \(\Sigma\). Similarly, \(\mathcal{T}(x; \mu, \Sigma, \lambda)\) is the multivariate \(t\) distribution with mean \(\mu\), scale matrix \(\Sigma\), and \(\lambda\) degrees of freedom, and \(\mathcal{IG}(s; \alpha, \beta)\) is the inverse gamma distribution over \(s\) with shape and scale parameters \(\alpha\) and \(\beta\). Further, \(\mathcal{MN}(X; M, U, V)\) denotes the matrix-variate Gaussian with mean-matrix \(M\), row-covariance \(U\), and column-covariance \(V\). Sequences are written as \(x_{1:n} = \{x_1, \ldots, x_n\}\) and for a matrix \(Z\), \(z = \text{vec}(Z)\) denotes vectorization of \(Z\). Recall that if \(C \sim \mathcal{MN}(C; M, U, V)\), then \(c \sim \mathcal{N}(c; \text{vec}(M), V \otimes U)\) where \(c = \text{vec}(C)\) and \(\otimes\) the Kronecker product (Gupta and Nagar, 1999). With \(y_k\) and \(x_k\) we respectively denote the \(k\)-th column of the matrices \(Y\) and \(X\).

## 2 THE PROBABILISTIC MODEL

We first describe the SSM, which consists of observations \((y_k)_{k \geq 1} \in \mathbb{R}^d\), latent coefficients \((x_k)_{k \geq 0} \in \mathbb{R}^r\), and a latent dictionary matrix \(C \in \mathbb{R}^{d \times r}\), as follows

\[
\begin{align*}
    p(C) &= \mathcal{MN}(C; C_0, I_d, V_0), \\
    p(x_0) &= \mathcal{N}(x_0; \mu_0, P_0), \\
    p(y_k | x_{k-1}, C) &= \mathcal{N}(y_k; Cx_k, R_k),
\end{align*}
\]

(2) \(\quad\) (3) \(\quad\) (5)

Here, \(f_\theta : \mathbb{R}^r \times \Theta \rightarrow \mathbb{R}^r\) is a nonlinear mapping that defines the dynamics of the coefficients with \(\Theta \subset \mathbb{R}^{d_\Theta}\) the parameter space, and \((Q_k, R_k)_{k \geq 1}\) are respectively the noise covariances of the coefficient dynamics (4) and the observation model (5). The initial covariances of the coefficients and the dictionary are denoted by \(P_0\) and \(V_0\), respectively.

Intuitively, the model (2)–(5) is a dimensionality reduction model where the dynamical structure of the learned subspace is explicitly modeled via the transition density (4). This means that inferring \(C\) and \((x_k)_{k \geq 0}\) will lead to a probabilistic dimensionality reduction scheme where the dynamical structure in the data will manifest itself in the dynamics of the coefficients \((x_k)_{k \geq 0}\). One main difficulty for applying standard schemes in this case is that we assume \(C\) to be an unknown and random matrix, therefore, the (extended) Kalman filter cannot be applied directly for inference. To alleviate this problem, we formulate the prior in (2) with a Kronecker covariance structure, which enables us to update (conditional on \(x_k\)) the posterior distribution of \(C\) analytically (Akyildiz and Míguez, 2019).

### 2.1 The case of the misspecified model

In the model (2)–(5), when the practitioner does not have a good idea of how to set the hyperparameters or when they are misspecified, the resulting inference scheme may perform suboptimally. To remedy this situation and to demonstrate the flexibility of our framework, we additionally propose a robust version of our model by introducing an inverse-gamma-distributed scale variable \(s\), and the model

\[
\begin{align*}
    p(s) &= \mathcal{IG}(s; \lambda_0/2, \lambda_0/2), \\
    p(C | s) &= \mathcal{MN}(C; C_0, I_d, sV_0)), \\
    p(x_0 | s) &= \mathcal{N}(x_0; \mu_0, sP_0), \\
    p(y_k | x_{k-1}, s) &= \mathcal{N}(y_k; f_\theta(x_{k-1}), sQ_0), \\
    p(y_k | x_k, C, s) &= \mathcal{N}(y_k; Cx_k, sR_0),
\end{align*}
\]

(6) \(\quad\) (7) \(\quad\) (8) \(\quad\) (9) \(\quad\) (10)

Note that in this model only the initial noise covariances \(Q_0\) and \(R_0\) need to be specified, in contrast to the model in (2)–(5). By marginalizing out the scale variable \(s\) in the multivariate normal distributions we obtain multivariate \(t\) distributions (e.g., \(p(x_0) = \int \mathcal{N}(x_0; \mu_0, sP_0) \mathcal{IG}(s; \lambda_0/2, \lambda_0/2)\, ds = T(x_0; \mu_0, P_0, \lambda_0)\), see Bishop (2006)). This technique has previously been used for robust versions of the Kalman filter (Girón and Rojano, 1994; Basu and Das, 1994; Roth et al., 2017, 2013; Tronarp et al., 2019). We follow the approach of Tronarp et al. (2019) to update \(Q_k\) and \(R_k\) at every iteration. These updates to the noise covariances lead to robustness in light of model misspecification, as discussed in Tronarp et al. (2019).

## 3 INFERENCE AND ESTIMATION

Here we derive the algorithm for performing sequential inference in the model (2)–(5). Inference in the robust model (6)–(10) is largely analogous, but necessary modifications are given in Sec. 3.2.3. We first present the optimal inference recursions and then describe our approximate inference scheme.

### 3.1 Optimal sequential inference

We give the optimal inference recursions for our model when \(\theta\) is assumed to be fixed, and thus drop \(\theta\) for notational clarity (parameter estimation is revisited in Sec. 3.2.4). To define a recursive one-step ahead procedure we assume that we are given the filters \(p(x_{k-1}|y_{1:k-1})\) and \(p(c|y_{1:k-1})\) at time \(k - 1\).
\textbf{Prediction.} Using the model (2)–(5) we compute the predictive distribution as
\begin{equation}
p(x_k|y_{1:k-1}) = \int p(x_k|y_{1:k-1})p(x_k|x_{1:k-1}) \, dx_{k-1}.
\end{equation}
(11)
We note that given \(p(x_{k-1}|y_{1:k-1})\) this step is independent of the dictionary.

\textbf{Update.} Given this predictive distribution of \(x_k\), we can now define the update steps of the method. In contrast to the Kalman filter, we have two quantities to update: \(x_k\) and \(c\). We first define the incremental marginal likelihood as
\begin{equation}
p(y_k|y_{1:k-1}) = \int p(y_k|c, x_k)p(x_k|y_{1:k-1}) \, p(c|y_{1:k-1}) \, dx_k \, dc.
\end{equation}
(12)

Next, we define the optimal recursions for updating the dictionary \(C\) and coefficients \((x_k)_{k \geq 1}\).

\textbf{Dictionary Update:} Given \(p(y_k|y_{1:k-1})\), we can first update the dictionary as follows
\begin{equation}
p(c|y_{1:k}) = p(c|y_{1:k-1}) \frac{p(y_k|c, y_{1:k-1})}{p(y_k|y_{1:k-1})}
\end{equation}
(13)
where
\begin{equation}
p(y_k|c, y_{1:k-1}) = \int p(y_k|c, x_k)p(x_k|y_{1:k-1}) \, dx_k.
\end{equation}
(14)

\textbf{Coefficient Update:} We also update the coefficients at time \(k\) (independent of the dictionary) as:
\begin{equation}
p(x_k|y_{1:k}) = p(x_k|y_{1:k-1}) \frac{p(y_k|x_k, y_{1:k-1})}{p(y_k|y_{1:k-1})}
\end{equation}
(15)
where
\begin{equation}
p(y_k|x_k, y_{1:k-1}) = \int p(y_k|x_k, c)p(c|y_{1:k-1}) \, dc.
\end{equation}
(16)
Unfortunately, these exact recursions are intractable. In the next section, we make these steps tractable by introducing approximations and obtain an efficient and explicit inference algorithm.

3.2 Approximate sequential inference

We start by assuming a special structure on the model. First, we note that the matrix-Gaussian prior in (2) can be written as \(p(c) = \mathcal{N}(c; c_0, V_0 \otimes I_d)\). The Kronecker structure in the covariance will be key to obtain an approximate and tractable posterior distribution with the same covariance structure. To describe our inference scheme we assume that we are given \(p(c|y_{1:k-1}) = \mathcal{N}(c; c_{k-1}, V_{k-1} \otimes I_d)\) and \(p(x_{k-1}|y_{1:k-1}) = \mathcal{N}(x_{k-1}; \mu_{k-1}, P_{k-1})\). Departing from these two distributions it is not possible to exactly update \(p(c|y_{1:k})\) and \(p(x_k|y_{1:k})\). As we introduce several approximations we will denote approximate densities with the symbol \(\tilde{p}(\cdot)\) instead of \(p(\cdot)\) to indicate that the distribution is not exact.

3.2.1 Prediction

In the prediction step, we need to compute (11). This is analytically tractable for \(f_\theta(x) = Ax\). More specifically, when \(f_\theta(x) = Ax\), given \(p(x_{k-1}|y_{1:k-1}) = \mathcal{N}(x_{k-1}; \mu_{k-1}, P_{k-1})\), we obtain \(p(x_k|y_{1:k-1}) = \mathcal{N}(x_k; \mu_k, P_k)\) where \(\mu_k = A\mu_{k-1}\) and \(P_k = AP_{k-1}A^\top + Q_k\). However, if \(f_\theta(x)\) is a nonlinear function, no solution exists and the integral in (11) is intractable. In this case, we can use the well-known extended Kalman update (EKF). This update is based on the local linearization of the transition model (McLean et al., 1962; Anderson and Moore, 1979), which gives \(\tilde{p}(x_k|y_{1:k-1}) = \mathcal{N}(x_k; \mu_k, P_k)\) with \(\mu_k = f_\theta(\mu_{k-1})\) and \(P_k = F_kP_{k-1}F_k^\top + Q_k\) where \(F_k = \frac{\partial f_\theta(x)}{\partial x}|_{x=\mu_{k-1}}\) is a Jacobian matrix associated with \(f_\theta\). The unscented Kalman filter of Julier and Uhlmann (1997) can also be used in this step when it is not possible to compute \(F_k\) or when \(f_\theta\) is highly nonlinear. However, since \(f_\theta\) is a modelling choice (analogous to choosing a kernel function in Gaussian Processes) this scenario is unlikely in practice and the EKF will generally suffice.

3.2.2 Update

For the update step, we are interested in updating both \(x_k\) and \(C\). Given the approximate predictive distribution \(\tilde{p}(x_k|y_{1:k-1})\), we would like to obtain \(\tilde{p}(c|y_{1:k})\) and \(\tilde{p}(x_k|y_{1:k})\). We first describe the update rule for the dictionary \(C\), then derive the approximate posterior of \(x_k\). Given the prediction, update steps of \(C\) and \(x_k\) are independent to avoid the repeated use of the data point \(y_k\).

\textbf{Dictionary Update.} To obtain \(\tilde{p}(c|y_{1:k})\), we note the integral (14) can be computed as
\begin{equation}
p(y_k|c, y_{1:k-1}) = \mathcal{N}(y_k; C\tilde{\mu}_k, R_k + CP_kC^\top).
\end{equation}
(17)
This closed form is not helpful to us since this distribution plays the role of the likelihood in (13). Since both the mean and the covariance depend on \(C\), the update (13) is intractable. To solve this problem, we first replace \(CP_kC^\top \approx C_{k-1}P_kC_{k-1}^\top\) in (17). This enables a tractable update where the likelihood is of the form \(\mathcal{N}(y_k; C\tilde{\mu}_k, R_k + CP_{k-1}C_{k-1}^\top)\). Finally, we choose the Gaussian with a constant diagonal covariance that is closest in terms of KL-divergence and obtain (see, e.g.,
where \( \eta_k = \text{Tr}(R_k + C_{k-1} \hat{P}_k C_{k-1}^T)/d \). With this approximation the update for the new posterior \( \tilde{p}(c|y_{1:k-1}) \) can be computed analytically, given formally in the following proposition based on Akyildiz and Miguez (2019).

**Proposition 1.** Given \( \tilde{p}(c|y_{1:k-1}) = N(c; c_{k-1}, V_{k-1} \otimes I_d) \) and the likelihood \( \tilde{p}(y_k|c, y_{1:k-1}) = N(y_k; C \hat{u}_k, \eta_k \otimes I_d) \) the approximate posterior distribution is \( \tilde{p}(c|y_{1:k}) = N(c; c_k, V_k \otimes I_d) \), where \( c_k = \text{vec}(C_k) \) and the posterior column-covariance matrix \( V_k \) is given by

\[
V_k = V_{k-1} - \frac{V_{k-1} \hat{u}_k \hat{u}_k^T V_{k-1}}{\hat{u}_k^T V_{k-1} \hat{u}_k + \eta_k} \quad \text{for} \quad k \geq 1, \tag{19}
\]

and the posterior mean \( C_k \) of the dictionary \( C \) can be obtained in matrix-form as

\[
C_k = C_{k-1} + \frac{(y_k - C_{k-1} \hat{u}_k) \hat{u}_k^T V_{k-1}}{\hat{u}_k^T V_{k-1} \hat{u}_k + \eta_k} \quad \text{for} \quad k \geq 1. \tag{20}
\]

**Proof.** See Supp. B. \( \blacksquare \)

We note the main gain of this result is that we obtain matrix-variate update rules for the sufficient statistics of the posterior distribution. This is key to an efficient implementation of the method.

**Coefficient Update.** To update the posterior density of coefficients, we derive the approximation of \( p(y_k|y_{1:k-1}, x_k) \) by integrating out \( c \), as in (16). First, we have the following result.

**Proposition 2.** Given \( p(y_k|c, x_k) \) as in (5) and \( p(c|y_{1:k-1}) = N(c; c_{k-1}, V_{k-1} \otimes I_d) \), we obtain

\[
p(y_k|y_{1:k-1}, x_k) = N(y_k; C_{k-1} x_k, R_k + x_k^T V_{k-1} x_k \otimes I_d). \tag{21}
\]

**Proof.** See Supp. C. \( \blacksquare \)

We note that in practice this quantity will be approximate as, e.g., \( \tilde{p}(c|y_{1:k-1}) \) (and other quantities) will be approximate. However, the likelihood in (21) with its current form is not amenable to exact inference in (15), as it contains \( x_k \) in both mean and covariance. Therefore, we approximate (21) by

\[
\tilde{p}(y_k|y_{1:k-1}, x_k) = N(y_k; C_{k-1} x_k, \tilde{R}_k), \tag{22}
\]

where \( \tilde{R}_k = R_k + \hat{u}_k^T V_{k-1} \hat{u}_k \otimes I_d \). With this likelihood, we can obtain the approximate posterior using (15) by an application of the Kalman update (Anderson and Moore, 1979), as

\[
\tilde{p}(x_k|y_{1:k}) = N(x_k; \mu_k, P_k) \text{ with } \\
\mu_k = \hat{u}_k + \tilde{P}_k C_{k-1} (C_{k-1} \hat{P}_k C_{k-1}^T + R_k)^{-1} (y_k - C_{k-1} \hat{u}_k), \tag{23}
\]

\[
P_k = \tilde{P}_k - \tilde{P}_k C_{k-1} (C_{k-1} \hat{P}_k C_{k-1}^T + R_k)^{-1} C_{k-1} \hat{P}_k. \tag{24}
\]

Thus we see that the update equations for both the dictionary and the coefficients can be easily implemented by straightforward matrix operations.

**Remark 1.** When \( \tilde{R}_k \) is diagonal the Woodbury matrix identity (Woodbury, 1950) can be used to accelerate the computation of \( (C_{k-1} \hat{P}_k C_{k-1}^T + R_k)^{-1} \). We apply this technique in our experiments in Section 4.4.

### 3.2.3 Inference in the robust model

For the robust model in (6)–(10) inference and estimation proceeds analogously. We provide the full derivation in Supp. F. As a consequence of the multivariate t distribution the degrees of freedom in the update equations increase by \( d \) at every iteration, which we write as \( \lambda_k = \lambda_k - d \). Let \( \Delta^2_{t,k} = (y_k - C_{k-1} \hat{u}_k)^T (C_{k-1} \hat{P}_k C_{k-1}^T + R_k)^{-1} (y_k - C_{k-1} \hat{u}_k) \) and \( \omega_k = (\lambda_k - 1 + \Delta^2_{t,k})/\lambda_k \). Then the reparameterization of the scale variable introduced in Tronarp et al. (2019) results in multi-

\[
\omega_k = \omega_{k-1} s_{k-1}, \text{ as well as the updates } Q_k = \omega_k Q_{k-1} \text{ and } R_k = \omega_k R_{k-1} \text{ for the noise covariances}. \]

While the mean updates for the coefficients and the dictionary remain unchanged in the robust model, the update of the coefficient covariance \( P_k \) and the dictionary column-covariance \( V_k \) are affected. The Student’s t update for \( P_k \) results in multiplication of the right-hand side of (24) by \( \omega_k \). Now let \( \hat{P}_k = \hat{u}_k^T V_{k-1} \hat{u}_k + \eta_k \) and \( \Delta^2_{s,k} = ||y_k - C_{k-1} \hat{u}_k||^2/\hat{P}_k \). Analogously, the right-hand side of (19) is multiplied by a factor \( \varphi_k = (\lambda_k - 1 + \Delta^2_{s,k})/\lambda_k \). See Supp. F for full details.

### 3.2.4 Parameter estimation

To estimate the parameters of \( f_\theta \) in (4), we need to solve

\[
\theta^* \in \operatorname{argmax}_{\theta \in \Theta} \log p_\theta(y_{1:n}), \tag{25}
\]

using gradient-based schemes (Kantas et al., 2015). We first present an offline gradient ascent scheme for when the number of observations is relatively small and then introduce a recursive variant that can be used in a streaming setting.

**Iterative estimation.** When the number of data points is limited, it is possible to employ an iterative
Algorithm 1 Iterative and recursive PSMF

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>Initialize $\gamma$, $\theta_0$, $C_0$, $V_0$, $P_0$, $(Q)<em>{k\geq 1}$, $(R)</em>{k\geq 1}$.</td>
</tr>
<tr>
<td>2:</td>
<td>for $i \geq 1$ do \textit{\textcopyright} iterative version</td>
</tr>
<tr>
<td>3:</td>
<td>$C_0 = C_n$, $\mu_0 = \mu_n$, $P_0 = P_n$, $V_0 = V_n$.</td>
</tr>
<tr>
<td>4:</td>
<td>for $1 \leq k \leq n$ do</td>
</tr>
<tr>
<td>5:</td>
<td>Compute predictive mean of $x_k$: $\bar{\mu}<em>k = f</em>{\theta_{i-1}}(\mu_{k-1})$ or $\bar{\mu}<em>k = f</em>{\theta_{i-1}}(\mu_{k-1})$</td>
</tr>
<tr>
<td>6:</td>
<td>Compute predictive covariance of $x_k$: $P_k = F_kP_{k-1}F_k^\top + Q_k$, with $F_k = \frac{\partial f(\theta)}{\partial x}</td>
</tr>
<tr>
<td>7:</td>
<td>Update dictionary mean $C_k$ using (20)</td>
</tr>
<tr>
<td>8:</td>
<td>Update dictionary covariance $V_k$ with (19)</td>
</tr>
<tr>
<td>9:</td>
<td>Update coefficient mean $\mu_k$ using (23)</td>
</tr>
<tr>
<td>10:</td>
<td>Update coefficient covariance $P_k$ with (24)</td>
</tr>
<tr>
<td>11:</td>
<td>Update parameters with (27) \textit{\textcopyright} iterative version</td>
</tr>
<tr>
<td>12:</td>
<td>Update parameters with (26) \textit{\textcopyright} iterative version</td>
</tr>
</tbody>
</table>

procedure using multiple passes over data by implementing

$$\theta_i = \theta_{i-1} + \gamma \nabla \log \tilde{p}_0(y_{1:n})|_{\theta=\theta_{i-1}},$$

(26)

at the $i$'th iteration. We refer to this approach as iterative PSMF. Since computing $\nabla \log p_0(y_{1:n})$ is not possible due to the intractability, we propose to use an approximation $\nabla \log \tilde{p}_0(y_{1:n}) = \sum_{k=1}^n \nabla \log \tilde{p}_0(y_k|y_{1:k-1})$ that can be computed during forward filtering and removes the need to store all gradients. We remark that it is possible to obtain two approximations of the incremental marginal likelihood $p_0(y_k|y_{1:k-1})$ by either integrating out $c$ in (18) or $x_k$ in (22). However, the resulting quantities are closely related and we choose the former path for computational reasons. We refer to Sec. 3.2.5 for the derivation of the approximate log-marginal likelihood $\log \tilde{p}_0(y_k|y_{1:k-1})$.

Recursive estimation. For long sequences, it is inefficient to perform (26). Instead, the parameter can be updated online during filtering by fixing $\theta = \theta_{k-1}$ and updating

$$\theta_k = \theta_{k-1} + \gamma \nabla \log \tilde{p}_0(y_k|y_{1:k-1})|_{\theta=\theta_{k-1}}.$$  

(27)

We call this approach recursive PSMF. This is an approximate recursive MLE procedure for SSMs (Kantas et al., 2015). This procedure has guarantees for finite-state space HMMs, but its convergence for general SSMs is an open problem (Kantas et al., 2015). The description of iterative and recursive PSMF is given in Algorithm 1.

Remark 2. The gradient steps in (26) and (27) can be replaced by modern optimizers to improve convergence, such as Adam (Kingma and Ba, 2015). We take advantage of this in Section 4.1 below.

Figure 1: Fitting rPSMF on synthetic data with $t$-distributed noise. Observed time series (blue) with unobserved future data (yellow) and the reconstruction from the model (red).

3.2.5 Approximating the marginal likelihood

Consider the likelihood (18) which equals $\tilde{p}_0(y_k|y_{1:k-1}, c) = \mathcal{N}(y_k; Cf_0(\mu_{k-1}), \eta_k \otimes I_d)$. Given $\tilde{p}(c|y_{1:k-1}) = \mathcal{N}(c; C_{k-1}V_{k-1} \otimes I_d)$, the negative log-likelihood is given by (see Supp. D)

$$-\log \tilde{p}_0(y_k|y_{1:k-1}) \leq \frac{d}{2} \log \left( \|f_0(\mu_{k-1})\|^2 + \eta_k \right) + \frac{1}{\eta_k} \left( \|y_k - C_{k-1}f_0(\mu_{k-1})\|^2 \right) - \frac{1}{2} \eta_k + \|f_0(\mu_{k-1})\|^2$$

(28)

where $\leq$ denotes equality up to some constants that are independent of $\theta$, hence irrelevant for the optimization. Eq. (28) can be seen as an optimization objective that arises from our model. We can compute the gradients of (28) using automatic differentiation for generic coefficient dynamics $f_0$.

4 EXPERIMENTS

We evaluate our method in several experiments. In the first two experiments, we show how PSMF can simultaneously learn the dictionary and the parameters of a nonlinear subspace model on synthetic and real data. The third experiment illustrates how our method can be beneficial for change point detection. Finally, our fourth experiment highlights how our method outperforms other MF methods for missing value imputation (modifications to handle missing data in our method are given in Supp. E). Additionally, in Supp. H and Supp. I we present experiments on the convergence of our method and recursive parameter estimation, respectively. Code to reproduce our experiments is available in an online repository.1

---

1See: [https://github.com/alan-turing-institute/rPSMF](https://github.com/alan-turing-institute/rPSMF)
4.1 A synthetic nonlinear periodic subspace

To demonstrate the ability of the algorithm to learn a dictionary and a structured subspace jointly, we choose $x_0 = \mu_0$ and $P_0 = 0$ and use $x_k = f_{\theta}(x_{k-1}) = \cos(2\pi \theta_1 k + x_{k-1})$, where $\theta \in \mathbb{R}_+^6$ and $Q_k = 0$ for all $k \geq 1$. This defines a deterministic subspace with highly periodic structure. We choose $d = 20$ and $r = 6$ and generate the data from the model with $\theta^* = 10^{-3} \cdot [1, 2, 3, 4, 5, 6]$. We explore both Gaussian and $t$-distributed measurement noise (the latter using 3 degrees of freedom) and both PMSF and rPSMF. We initialize $C_0$ randomly and draw $\theta_0$ from a uniform distribution on $[0, 0.1]^6$. We set $V_0 = v_0 \otimes I_r$ with $v_0 = 0.1$ and use $\lambda_0 = 1.8$ for rPSMF. We furthermore use iterative parameter estimation using the Adam optimizer (Kingma and Ba, 2015) with standard parameterization, and re-initialize $V_0$, $R_0$, and $Q_0$ at every (outer) iteration (see Supp. G.1 for details). The generated data can be seen in Fig. 1. The task is thus to identify the correct subspace structure by estimating $\theta$ as well as to learn the dictionary matrix $C$.

Fig. 1 shows a run with $t$-distributed noise fitted by rPSMF. We can see that even though the data exhibits clear outliers the model successfully learns both the underlying generative model and its parameters. Expanded results for PSMF are available in Supp. G.1.

4.2 Forecasting real-world data using a nonlinear subspace

To illustrate the advantage of specifying an appropriate subspace model, we explore weather-related features in a dataset on air quality in Beijing (Liang et al., 2015), obtained from the UCI repository (Bache and Lichman, 2013). To simplify the problem the dataset is sampled at every 100 steps, resulting in $n = 439$ observations and $d = 3$ variables (dew point, temperature, and atmospheric pressure). We compare PSMF using a random walk subspace model, $x_k = f(x_{k-1}) = x_{k-1}$, against a periodic subspace model $x_k = f_{\theta}(x_{k-1}) = \theta_1 \sin(2\pi \theta_2 k + \theta_3 x_{k-1}) + \theta_4 \cos(2\pi \theta_3 k + \theta_5 x_{k-1})$. In both settings we use $r = 1$, run iterative PSMF with 100 iterations, and withhold 20% of the data for prediction. Fig. 2 illustrates the benefit of an appropriate subspace model on forecasting performance and confirms that PSMF can recover nonlinear subspace dynamics in real-world datasets.

4.3 Learning representations for robust multivariate changepoint detection

We generate time series with $d = 20$ dimensions where only 3 exhibit a structural change. In addition to standard Gaussian noise we contaminate 5% of the entries on average using heavy-tailed $t$-distributed noise with degrees of freedom varying from 1.5 to 1.9. To learn the structural changes and be robust against the heavy-tailed noise, we design a smooth subspace model $(x_i(t))_{t \geq 0}$ for $i = 1, \ldots, r$ in continuous-time using a Gaussian process (GP) prior, $x_i(t) \sim \mathcal{GP}(0, k_{\nu}(t, t'))$, with Matérn-3/2 kernel with $\nu = 3/2$ (Williams and Rasmussen, 2006). This particular GP admits a state-space representation amenable to filtering (Hartikainen and Särkkä, 2010) as it can be recast (Särkkä et al., 2013) as the stochastic differential equation (SDE):

$$\frac{dx_i(t)}{dt} = \begin{bmatrix} 0 & 1 \\ -\kappa^2 & -2\kappa \end{bmatrix} x_i(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_i(t) \quad (29)$$

where $x_i(t) = [x_i(t), dx_i(t)/dt]$ and $\kappa = \sqrt{2\sigma^2}\ell$. We choose $\sigma^2 = 0.1$ and $\ell = 0.1$ and discretize equation (29) with the step-size $\gamma = 0.001$. We discretize the SDEs for $i = 1, \ldots, r$ and construct a joint state which leads to a linear dynamical system in $2r$ dimensions for which we can run PSMF. The details of the discretization and the corresponding PSMF model are given in Supp. G.2, along with an illustration of the learned GP features.
We first employ PELT (Killick et al., 2012) as an example changepoint detection method directly on the time-series to create a baseline. Then, we estimate a smooth GP subspace with PSMF and run PELT on that subspace (i.e., the columns of X). We additionally compare against a multivariate implementation of Bayesian online CPD (MBOCPD, Adams and MacKay, 2007). The results in Table 1 clearly show the improved performance and robustness of PSMF.

4.4 Missing Value Imputation

Finally, we test our method on imputation of missing values in time-series data. We consider data from various domains, including three series of air pollutants measured across London2, a gas-sensor dataset by Burgués et al. (2018) obtained from the UCI repository (Bache and Lichman, 2013), and five years of daily closing prices of stocks in the S&P500 index.3 The air pollution series contain hourly measurements between 2018-06-01 and 2018-12-01 (n = 4393) and consist of NO2 measured at d = 83 sites, PM10 from d = 74 sites, and PM25 measured at d = 26 sites. For the gas sensor dataset (Gas) we have n = 295, 719 and d = 19 and for the S&P500 dataset we have n = 1259 and d = 505.

We run PSMF with the discretized GP subspace model with r = 10. We note that the goal is to obtain a representation that is helpful for changepoint detection. We first employ PELT (Killick et al., 2012) as an example changepoint detection method directly on the time-series to create a baseline. Then, we estimate a smooth GP subspace with PSMF and run PELT on that subspace (i.e., the columns of X). We additionally compare against a multivariate implementation of Bayesian online CPD (MBOCPD, Adams and MacKay, 2007). The results in Table 1 clearly show the improved performance and robustness of PSMF.

The air pollution datasets contain a high number of missing values due to sensor failures and maintenance, and the stock price dataset contains missing values due to stocks being added to the index.

To test the accuracy of imputations, we randomly remove segments of length 20 and thereby construct datasets with 30% missing data. We compare our methods against four baselines. The first is an MLE approach to online probabilistic matrix factorization (Yildirim et al., 2012; Sun et al., 2012; Févotte et al., 2013) where we construct an SSM where C is constant, denoted as MLE-SMF. The second is temporal matrix factorization (TMF) which is an adaptation of the optimisation-based method of Yu et al. (2016). We also add two popular offline methods that can only operate on the entire data matrix at once: PMF (Mnih and Salakhutdinov, 2008) and BPMF (Salakhutdinov and Mnih, 2008).

We assume the subspace model to be a random walk, fθ(x) = x, thus avoiding the parameter estimation problem, and we use the final estimates of C and X for data imputation. We formulate TMF with the weight matrix set to identity for tractability. We set r = 10 for all methods and datasets. For PMF and MLE-SMF we set Rk := R = ρ ⊗ I, with ρ = 10, P0 = I, Qk := Q = q ⊗ I, with q = 0.1. For rPSMF we use R0 = R and Q0 = Q and set λ0 = 1.8. For PMF and rPSMF we let V0 = v0 ⊗ I, where v0 = 2. All methods are run for two iterations (epochs) over the data to limit run-times, and we repeat the experiments 100 times with different initializations and missing data patterns. In Table 2 we see that PSMF and rPSMF attain lower RMSEs compared to all other methods and that they are competitive in terms of running time. The advantage of our method is especially noticeable on the NO2, SP500, and Gas datasets.

We can also measure the proportion of missing values that lie within a 2σ coverage interval of the approximate posterior distribution. Table 3 shows how our

<table>
<thead>
<tr>
<th>NO2</th>
<th>PM10</th>
<th>PM25</th>
<th>S&amp;P500</th>
<th>Gas</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSMF</td>
<td>0.76</td>
<td>0.76</td>
<td>0.92</td>
<td>0.83</td>
</tr>
<tr>
<td>rPSMF</td>
<td><strong>0.85</strong></td>
<td><strong>0.89</strong></td>
<td>0.87</td>
<td><strong>0.83</strong></td>
</tr>
<tr>
<td>MLE-SMF</td>
<td>0.43</td>
<td>0.56</td>
<td>0.80</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Table 3: Average coverage proportion of the missing data by the 2σ uncertainty bars of the posterior predictive estimates, averaged over 100 repetitions.

Table 2: Imputation error and runtime on several datasets using 30% missing values, averaged over 100 random repetitions. An asterisk marks offline methods.
method improves over the uncertainty quantification of MLE-SMF (the other methods do not provide a posterior distribution). This illustrates the added value of the matrix-variate prior on $C$, as well as our inference scheme. Note that rPSMF obtains a higher coverage percentage than PSMF on three of the datasets, which is due to the sequential updating of the noise covariances. Additional results with 20% and 40% missing data are available in Supp. G.3.

5 CONCLUSION

We have recast the problem of probabilistic dimensionality reduction for time-series as a joint state filtering and parameter estimation problem in a state-space model. Our model is fully probabilistic and we provide a tractable sequential inference algorithm to run the method with linear computational complexity with respect to the number of data points. Our algorithm is purely recursive and can be used in streaming settings. In particular, the batch algorithms that we compare to (such as PMF and BPMF) would incur the full runtime costs when a new sample arrives. This would likely be prohibitive in practice (e.g. BPMF takes 90 seconds for the Gas dataset) and would increase significantly with the dataset size. By contrast, our method only requires incremental computation to process a new sample (e.g. on the order of milliseconds for the Gas dataset).

We have also extended our initial model into a robust version to handle model misspecification and datasets contaminated with outliers. The robust version of our method has been shown to be advantageous in light of model misspecification.

The state-space formulation of the problem opens many directions for future research such as (i) the use of general models for $f_0$ or non-Gaussian likelihoods, (ii) the exploration of the use of switching SSMs, and (iii) the integration of more advanced inference techniques such as ensemble Kalman filters or Monte Carlo-based methods for nonlinear and non-Gaussian generalizations of our model.

Acknowledgments

Ö. D. A. is supported by the Lloyd’s Register Foundation Data Centric Engineering Programme and EPSRC Programme Grant EP/R034710/1. The work of GvdB is supported in part by EPSRC grant EP/N510129/1 to the Alan Turing Institute. T. D. acknowledges support from EPSRC EP/T004134/1, UKRI Turing AI Fellowship EP/V02678X/1, and Lloyd’s Register Foundation programme on Data Centric Engineering through the London Air Quality project.

References


