ONLINE BAYESIAN LOW-RANK SUBSPACE LEARNING FROM PARTIAL OBSERVATIONS

P.V. Giampouras, A.A. Rontogiannis, K.E. Themelis, K.D. Koutroumbas

IAASARS, National Observatory of Athens, GR-15236, Penteli, Greece

ABSTRACT

Learning the underlying low-dimensional subspace from streaming incomplete high-dimensional observations data has attracted considerable attention lately. In this paper, we present a new computationally efficient Bayesian scheme for online low-rank subspace learning and matrix completion. The proposed scheme builds upon a properly defined hierarchical Bayesian model that explicitly imposes low rank to the latent subspace by assigning sparsity promoting Student-t priors to the columns of the subspace matrix. The new algorithm is fully automated and as corroborated by numerical simulations, provides higher estimation accuracy and a better estimate of the true subspace rank compared to state of the art methods.

Index Terms— Online low-rank subspace learning, matrix completion, variational Bayes, big data.

1. INTRODUCTION

Extraction of information from high-dimensional data has been the subject of many applications in signal processing and machine learning. However, despite their high-dimensionality, frequently data are known to 'live' in low-dimensional subspaces. Principal components analysis (PCA) has been for many years the workhorse technique for dimensionality reduction and low-rank linear subspace learning in a static environment. Nevertheless, under dynamically changing conditions or when processing of streaming data is required, classical PCA becomes computationally prohibitive. This problem has been traditionally solved using adaptive singular value decomposition (SVD) based subspace tracking techniques, e.g., [1].

Nowadays, with the advent of big data analytics, the need to learn and track the low-rank subspace of high-dimensional observations data in a computationally efficient manner, has become both imperative and challenging. In addition to this, for various reasons observations data may be only partially known or purposely undersampled, rendering conventional methods infeasible. In such cases, (online) subspace learning turns out to be commensurate to the so-called (online) low-rank matrix completion problem. Very recently, online algorithms have been developed, which have the ability to recursively estimate the low-rank subspace where data reside, using only limited observations [2], [3], [4]. These, in essence, are based on *deterministic* alternating optimization strategies, each minimizing a suitably formed cost function.

In this work, we present a new sequential algorithm for online subspace learning from partial observations, which, unlike previous works, emerges from a *Bayesian* standpoint. The proposed algorithm is derived by transforming an approximate Bayesian inference scheme, which originates from an appropriate hierarchical Bayesian model defined on the problem parameters. The main feature of the model is that it explicitly enforces low rank to the data latent subspace via a matrix column sparsity inducing mechanism, [5]. Compared to state-of-the-art related techniques, the new online scheme a) is fully automated as no parameter fine-tuning is required, b) is of similar or lower computational complexity and c) as verified by simulations, offers a lower average estimation error and a more accurate estimate of the true subspace rank.

Notation: Column vectors are represented as boldface lowercase letters, e.g. $\mathbf{x}$, and matrices as boldface uppercase letters, e.g. $\mathbf{X}$, while, unless otherwise explicitly stated, x_i is the ith element of $\mathbf{x}$ and x_{ij} the ijth element of $\mathbf{X}$. Moreover, $(\cdot)^T$ denotes transposition, $\mathbf{I}_k$ is the $k \times k$ identity matrix, $\|\cdot\|$ is the standard ℓ_2-norm, $\|\cdot\|_F$ stands for the Frobenius norm, $\odot$ denotes Hadamard entrywise product, $\mathcal{N}(\cdot)$ is the Gaussian distribution, $\mathcal{G}(\cdot)$ is the Gamma distribution, $\langle\cdot\rangle$ is the expectation operator, $\mathrm{diag}(\mathbf{x})$ denotes a diagonal matrix whose diagonal entries are the elements of $\mathbf{x}$, and $\mathrm{Trace}(\mathbf{X})$ is the trace of the square matrix $\mathbf{X}$.

2. PROBLEM STATEMENT

We assume that $K \times 1$ vectors of streaming observations are generated according to the linear regression model

$$\mathbf{y}(n) = \mathbf{W}(n)\mathbf{x}(n) + \mathbf{e}(n), \tag{1}$$

where n is the time index. In (1), the columns of the $K \times L$ matrix $\mathbf{W}(n)$ span a low-dimensional subspace ($L \ll K$), the $L \times 1$ vector $\mathbf{x}(n)$ is the representation of $\mathbf{y}(n)$ in this subspace and $\mathbf{e}(n) \sim \mathcal{N}(\mathbf{e}(n)|\mathbf{0}, \beta^{-1}\mathbf{I}_K)$, with β being the noise presicion. Note that the true rank of $\mathbf{W}(n)$ is $r < L$

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Pro- gram Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Programs: THALIS and ARISTEIA- HSI-MARS- 1413.

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

and may be also time-varying. However, in order to facilitate our analysis, we use in (1) an overestimate L of the true rank, although one of our goals in this work is also the estimation of the true rank of $\mathbf{W}(n)$. The subspace matrix $\mathbf{W}(n)$ is defined in terms of its rows and columns as follows

$$\begin{aligned}\mathbf{W}(n) &= [\mathbf{w}_1(n), \mathbf{w}_2(n), \ldots, \mathbf{w}_K(n)]^T \\ &= [\boldsymbol{w}_1(n), \boldsymbol{w}_2(n), \ldots, \boldsymbol{w}_L(n)]\,. \end{aligned} \tag{2}$$

Note that $\mathbf{w}_k(n)$ is the column vector containing the elements of the kth row of $\mathbf{W}(n)$. By collecting observations up to time n as rows in the $n \times K$ observations matrix $\mathbf{Y}(n)$, yields

$$\mathbf{Y}(n) = [\mathbf{y}(1), \mathbf{y}(2), \cdots, \mathbf{y}(n)]^T = [\boldsymbol{y}_1(n), \boldsymbol{y}_2(n), \ldots, \boldsymbol{y}_K(n)], \tag{3}$$

and additionally

$$\mathbf{X}(n) = [\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(n)]^T = [\boldsymbol{x}_1(n), \boldsymbol{x}_2(n), \ldots, \boldsymbol{x}_L(n)]. \tag{4}$$

In this work, we assume that only a part of the observations in $\mathbf{Y}(n)$ are available. To model missing data, we define the time-increasing, binary $n \times K$ matrix $\boldsymbol{\Phi}(n) = [\boldsymbol{\phi}(1), \boldsymbol{\phi}(2), \ldots, \boldsymbol{\phi}(n)]^T = [\boldsymbol{\varphi}_1(n), \boldsymbol{\varphi}_2(n), \ldots, \boldsymbol{\varphi}_K(n)]$, having 1's at the positions where data are known, and 0's elsewhere. Therefore, the matrix $\mathbf{V}(n) = \boldsymbol{\Phi}(n) \odot \mathbf{Y}(n)$ contains only the available observations. Given $\mathbf{V}(n)$, our goal is to recursively estimate in time, a) the low-rank matrix $\mathbf{W}(n)$ and b) the projection $\mathbf{x}(n)$ of the current observations vector $\mathbf{y}(n)$ onto the column space of $\mathbf{W}(n)$. To achieve this goal, we may exploit the following exponentially weighted least squares (LS) cost function optimization problem,

$$\{\hat{\mathbf{W}}(n), \hat{\mathbf{X}}(n)\} = \underset{\mathbf{W}(n), \mathbf{X}(n)}{\operatorname{argmin}} \|\boldsymbol{\Phi}(n) \odot [\boldsymbol{\Lambda}^{1/2}(n)(\mathbf{Y}(n) - \mathbf{X}(n)\mathbf{W}^T(n))]\|_F^2, \tag{5}$$

where $\boldsymbol{\Lambda}(n) = \operatorname{diag}([\lambda^{n-1}, \lambda^{n-2}, \ldots, 1]^T)$ and λ is the usual forgetting factor that places more importance to recent observations ($0 \ll \lambda < 1$). Based on (5), a recursive LS type algorithm has been developed in [3], that alternates between computing the estimate $\hat{\mathbf{x}}(n)$ and updating the subspace $\hat{\mathbf{W}}(n)$. Also in [4], an alternating ridge-regression type adaptive algorithm is derived, after regularizing (5) with a Frobenius norm upper bound of the *low-rank promoting* nuclear norm of $\mathbf{X}(n)\mathbf{W}^T(n)$. In the current work a different approach is followed stemming from a Bayesian perspective. More specifically, $\mathbf{W}(n)$ is enforced to be low-rank by using the column sparsity inducing method proposed in [5] and explained in the next section.

3. LOW-RANK BAYESIAN MODELING

In this and the next sections, we temporarily drop the dependence of all quantities on the time index n and develop a batch iterative scheme for low-rank subspace estimation. First, taking into account a) the data generation model (1), b) the statistics of the noise and c) the exponentially weighted windowing of the data, we get the following likelihood function for the observations

$$p(\boldsymbol{\Phi} \odot \mathbf{Y} \mid \mathbf{X}, \mathbf{W}, \beta) = \prod_{j=1}^{n} \mathcal{N}(\boldsymbol{\phi}(j) \odot \mathbf{y}(j) \mid \boldsymbol{\phi}(j) \odot \mathbf{W}\mathbf{x}(j), \lambda^{n-j}\beta^{-1}\boldsymbol{\Phi}(j)), \tag{6}$$

where $\boldsymbol{\Phi}(j) = \operatorname{diag}(\boldsymbol{\phi}(j))$. Additionally, in order to impose low-rankness, we first recall that

$$\mathbf{X}\mathbf{W}^T = \sum_{l=1}^{L} \boldsymbol{x}_l \boldsymbol{w}_l^T, \tag{7}$$

with each outer-product of the respective columns of $\mathbf{X}$ and $\mathbf{W}$ in the right hand side of (7) adding one to the rank of the matrix $\mathbf{X}\mathbf{W}^T$. Herein, we adopt the approach proposed in [5], properly adjusted to our likelihood function. According to [5], sparsity is imposed jointly to the columns of $\mathbf{X}$ and $\mathbf{W}$ via appropriate hierarchical Bayesian modelling. More specifically, at the first level of the hierarchy the following Gaussian priors are assigned to the columns of $\mathbf{X}$ and $\mathbf{W}$,

$$p(\mathbf{X} \mid \mathbf{s}, \beta) = \prod_{l=1}^{L} \mathcal{N}(\boldsymbol{x}_l \mid \mathbf{0}, \beta^{-1}s_l^{-1}\boldsymbol{\Lambda}), \tag{8}$$

$$p(\mathbf{W} \mid \mathbf{s}, \beta) = \prod_{l=1}^{L} \mathcal{N}(\boldsymbol{w}_l \mid \mathbf{0}, \beta^{-1}s_l^{-1}\mathbf{I}_K), \tag{9}$$

where $\mathbf{s} = [s_1, s_2, \ldots, s_L]^T$. Note that the lth columns of $\mathbf{X}, \mathbf{W}$ share a common *sparsity promoting parameter* s_l, for $l = 1, 2, \ldots, L$. When performing Bayesian inference, some of the s_l's will take very large values, thus annihilating the corresponding columns of $\mathbf{X}$ and $\mathbf{W}$ and reducing the rank. To this end, at the second level of the hierarchy these parameters are assumed to follow a conjugate Gamma distribution,

$$p(\mathbf{s}) = \prod_{l=1}^{L} \mathcal{G}(s_l; \varsigma_l, \delta_l). \tag{10}$$

Similar to sparse Bayesian learning [6], by integrating out $\mathbf{s}$ from (8) and (9) leads to heavy-tailed Student-t marginal priors for the columns of $\mathbf{X}$ and $\mathbf{W}$, respectively. To complete the Bayesian model, a conjugate Gamma distribution is assigned to the noise precision β,

$$p(\beta) = \mathcal{G}(\beta; \kappa, \theta). \tag{11}$$

Based on the previously described Bayesian model, an efficient approximate Bayesian inference method is derived in the next section for low-rank subspace learning and matrix completion.

4. BATCH VARIATIONAL BAYES INFERENCE

To perform Bayesian inference, we need the joint posterior distribution $p(\mathbf{X}, \mathbf{W}, \mathbf{s}, \beta \mid \mathbf{Y})$, which, by applying the Bayes

rule, is expressed as

$$p(\mathbf{X},\mathbf{W},\mathbf{s},\beta \mid \mathbf{Y}) = \frac{p(\mathbf{Y},\mathbf{X},\mathbf{W},\mathbf{s},\beta)}{\int p(\mathbf{Y},\mathbf{X},\mathbf{W},\mathbf{s},\beta) d\mathbf{X} d\mathbf{W} d\mathbf{s} d\beta}. \quad (12)$$

Unfortunately, due to the complexity of our Bayesian model, $p(\mathbf{X},\mathbf{W},\mathbf{s},\beta \mid \mathbf{Y})$ can not be computed in closed-form from (12) and thus we have to properly approximate it. As in [5], [7], we adopt the approximation method termed variational Bayes (VB) inference. According to VB, $p(\mathbf{X},\mathbf{W},\mathbf{s},\beta \mid \mathbf{Y})$ is approximated by its closest distribution in the space of *factorized distributions* with respect to the Kullback-Leibler (KL) divergence criterion [7]. In this work we resort to the following factorization,

$$q(\mathbf{X},\mathbf{W},\mathbf{s},\beta) = q(\beta)\prod_{j=1}^{n} q(\mathbf{x}(j)) \prod_{k=1}^{K}\prod_{l=1}^{L} q(w_{kl}) \prod_{l=1}^{L} q(s_l). \quad (13)$$

Note that in contrast to [5], the approximating distribution $q(\mathbf{X},\mathbf{W},\mathbf{s},\beta)$ is assumed to be fully factorized with respect to the elements of $\mathbf{W}$. This modification is important, as it will pave the way for the development of a computationally efficient online algorithm in the next section. Now, by minimizing the KL divergence between $p(\mathbf{X},\mathbf{W},\mathbf{s},\beta \mid \mathbf{Y})$ and $q(\mathbf{X},\mathbf{W},\mathbf{s},\beta)$, the individual distribution factors $q(\cdot)$'s in the right hand side of (13) can be computed in closed-form [7].

Specifically, it turns out easily that the posterior distribution $q(\mathbf{x}(j))$ of the given row $\mathbf{x}(j)$ of $\mathbf{X}$ is Gaussian,

$$q(\mathbf{x}(j)) = \mathcal{N}(\mathbf{x}(j) \mid \langle \mathbf{x}(j) \rangle, \mathbf{\Sigma}_{\mathbf{x}(j)}), \quad (14)$$

with mean and covariance matrix

$$\langle \mathbf{x}(j) \rangle = \langle \beta \rangle \mathbf{\Sigma}_{\mathbf{x}(j)} \langle \mathbf{W} \rangle^T \left(\boldsymbol{\phi}(j) \odot \mathbf{y}(j) \right), \quad (15)$$

$$\mathbf{\Sigma}_{\mathbf{x}(j)} = \langle \beta \rangle^{-1} \left(\langle \mathbf{W}^T \mathbf{\Phi}(j) \mathbf{W} \rangle + \langle \mathbf{S} \rangle \right)^{-1}. \quad (16)$$

where $\mathbf{S} = \mathrm{diag}(\mathbf{s})$. Moreover, $\langle \mathbf{W}^T \mathbf{\Phi}(j) \mathbf{W} \rangle$ equals

$$\langle \mathbf{W}^T \mathbf{\Phi}(j) \mathbf{W} \rangle = \langle \mathbf{W}^T \rangle \mathbf{\Phi}(j) \langle \mathbf{W} \rangle + \sum_{k=1}^{K} \phi_{jk} \mathbf{\Sigma}_{\mathbf{w}_k} \quad (17)$$

with $\mathbf{\Sigma}_{\mathbf{w}_k}$ the $L \times L$ covariance matrix of $\mathbf{w}_k$, which is diagonal due to the posterior independence assumed in (13).

Working as above, we find that $q(w_{kl})$ is the following Gaussian distribution

$$q(w_{kl}) = \mathcal{N}(w_{kl} \mid \langle w_{kl} \rangle, \sigma^2_{w_{kl}}), \quad (18)$$

where,

$$\langle w_{kl} \rangle = \langle \beta \rangle \sigma^2_{w_{kl}} \left(\langle \boldsymbol{x}_l^T \rangle \mathbf{\Lambda} \boldsymbol{\Phi}_k \boldsymbol{y}_k - \langle \boldsymbol{x}_l^T \mathbf{\Lambda} \boldsymbol{\Phi}_k \mathbf{X}_{\neg l} \rangle \langle \mathbf{w}_{k\neg l} \rangle \right) \quad (19)$$

$$\sigma^2_{w_{kl}} = \langle \beta \rangle^{-1} \left(\langle \boldsymbol{x}_l^T \mathbf{\Lambda} \boldsymbol{\Phi}_k \boldsymbol{x}_l \rangle + \langle s_l \rangle \right)^{-1}. \quad (20)$$

In (19), $\mathbf{X}_{\neg l}$ and $\mathbf{w}_{k\neg l}$ result from $\mathbf{X}$ and $\mathbf{w}_k$ after removing their lth column and lth entry, respectively and $\boldsymbol{\Phi}_k = \mathrm{diag}(\boldsymbol{\varphi}_k)$. In addition,

$$\langle \boldsymbol{x}_l^T \mathbf{\Lambda} \boldsymbol{\Phi}_k \mathbf{X}_{\neg l} \rangle = \langle \boldsymbol{x}_l^T \rangle \mathbf{\Lambda} \boldsymbol{\Phi}_k \langle \mathbf{X}_{\neg l} \rangle + \sum_{j=1}^{n} \lambda^{n-j} \phi_{jk} \boldsymbol{\sigma}_{\mathbf{x}(j)_{\neg l}} \quad (21)$$

$$\langle \boldsymbol{x}_l^T \mathbf{\Lambda} \boldsymbol{\Phi}_k \boldsymbol{x}_l \rangle = \langle \boldsymbol{x}_l^T \rangle \mathbf{\Lambda} \boldsymbol{\Phi}_k \langle \boldsymbol{x}_l \rangle + \sum_{j=1}^{n} \lambda^{n-j} \phi_{jk} \sigma^2_{x_{jl}} \quad (22)$$

where $\boldsymbol{\sigma}_{\mathbf{x}(j)_{\neg l}}$ represents the lth column of $\mathbf{\Sigma}_{\mathbf{x}(j)}$ after removing its lth element $\sigma^2_{x_{jl}}$. As far as the posterior of s_l is concerned, we get

$$q(s_l) = \mathcal{G}(s_l \mid \hat{\varsigma}_l, \hat{\delta}_l) \quad (23)$$

where, $\hat{\varsigma} = \varsigma_l + (K+n)/2$ and $\hat{\delta} = \delta_l + \Big(\langle \beta \rangle (\langle \boldsymbol{x}_l^T \mathbf{\Lambda} \boldsymbol{x}_l \rangle + \langle \boldsymbol{w}_l^T \boldsymbol{w}_l \rangle) \Big)/2$. The posterior mean of s_l is thus given by,

$$\langle s_l \rangle = \frac{2\varsigma_l + K + n}{2\delta_l + \langle \beta \rangle \left(\langle \boldsymbol{x}_l^T \mathbf{\Lambda} \boldsymbol{x}_l \rangle + \langle \boldsymbol{w}_l^T \boldsymbol{w}_l \rangle \right)}, \quad (24)$$

with $\langle \boldsymbol{x}_l^T \mathbf{\Lambda} \boldsymbol{x}_l \rangle = \langle \boldsymbol{x}_l^T \rangle \mathbf{\Lambda} \langle \boldsymbol{x}_l \rangle + \sum_{j=1}^{n} \lambda^{n-j} \sigma^2_{x_{jl}}$ and $\langle \boldsymbol{w}_l^T \boldsymbol{w}_l \rangle = \langle \boldsymbol{w}_l^T \rangle \langle \boldsymbol{w}_l \rangle + \sum_{k=1}^{K} \sigma^2_{w_{kl}}$. Likewise, the approximate posterior distribution of β is a Gamma distribution i.e.,

$$q(\beta) = \mathcal{G}(\beta \mid \hat{\kappa}, \hat{\theta}) \quad (25)$$

with $\hat{\theta} = \sum_{k=1}^{K} \left(\langle \| \mathbf{\Lambda}^{\frac{1}{2}} \boldsymbol{\Phi}_k \left(\boldsymbol{y}_k - \mathbf{X} \mathbf{w}_k \right) \|^2 \rangle + \langle {\mathbf{w}_k}^T \mathbf{S} \mathbf{w}_k \rangle \right)$ $+ \sum_{l=1}^{L} \langle s_l \rangle \langle \boldsymbol{x}_l^T \mathbf{\Lambda} \boldsymbol{x}_l \rangle + 2\theta$, $\hat{\kappa} = \kappa + \Big(n(K+L) + KL \Big)/2$ and $\langle \beta \rangle = \hat{\kappa}/\hat{\theta}$, where

$$\begin{aligned} \langle \| \mathbf{\Lambda}^{\frac{1}{2}} \boldsymbol{\Phi}_k \left(\boldsymbol{y}_k - \mathbf{X} \mathbf{w}_k \right) \|^2 \rangle &= \| \mathbf{\Lambda}^{\frac{1}{2}} \boldsymbol{\Phi}_k \left(\boldsymbol{y}_k - \langle \mathbf{X} \rangle \langle \mathbf{w}_k \rangle \right) \|^2 \\ &+ \mathrm{Trace}(\langle \mathbf{X} \rangle^T \mathbf{\Lambda} \boldsymbol{\Phi}_k \langle \mathbf{X} \rangle \mathbf{\Sigma}_{\mathbf{w}_k}) + \langle \mathbf{w}_k^T \rangle \sum_{j=1}^{n} \phi_{jk} \lambda^{n-j} \mathbf{\Sigma}_{\mathbf{x}(j)} \langle \mathbf{w}_k \rangle \\ &+ \mathrm{Trace}(\mathbf{\Sigma}_{\mathbf{w}_k} \sum_{j=1}^{n} \phi_{jk} \lambda^{n-j} \mathbf{\Sigma}_{\mathbf{x}(j)}) \end{aligned} \quad (26)$$

$$\langle \mathbf{w}_k^T \mathbf{S} \mathbf{w}_k \rangle = \langle \mathbf{w}_k^T \rangle \langle \mathbf{S} \rangle \langle \mathbf{w}_k \rangle + \sum_{l=1}^{L} s_l \sigma^2_{w_{kl}} \quad (27)$$

From the previous analysis, it can easily be seen that the expectations of the model parameters $\langle w_{kl} \rangle, \langle \mathbf{x}(j) \rangle, \langle s_l \rangle$ and $\langle \beta \rangle$ are mutually dependent. Thus, a cyclic VB iterative scheme can be defined among them that provides an estimate of the low-rank subspace matrix $\mathbf{W}$ in terms of $\langle w_{kl} \rangle$ after convergence (which is theoretically established). Such a scheme, however, cannot be used for online processing, since, in such a case, the sizes of the involved matrices $\mathbf{Y}, \mathbf{X}$ increase with time. In the next section we show how this batch iterative scheme can be properly transformed, so as to be able to efficiently process streaming data.

5. ONLINE VB SUBSPACE ESTIMATION

In this section a new VB algorithm is derived for online subspace estimation and matrix completion. To this end, in the

following we restore the time index n and for notational convenience the expectation operator $\langle\cdot\rangle$ is omitted. In the online scenario it is assumed that a stream of high-dimensional incomplete data vectors $\mathbf{y}(n)$ is received and we process them sequentially in order to estimate the low-rank subspace $\hat{\mathbf{W}}(n)$ where they reside, as well as their projection $\hat{\mathbf{x}}(n)$ on it. First, from (15) for $j = n$ we get,

$$\hat{\mathbf{x}}(n) = \beta(n-1)\boldsymbol{\Sigma}_{\hat{\mathbf{x}}}(n)\hat{\mathbf{W}}^T(n-1)\left(\boldsymbol{\phi}(n)\odot\mathbf{y}(n)\right), \quad (28)$$

$$\boldsymbol{\Sigma}_{\hat{\mathbf{x}}}(n) = \beta^{-1}(n-1)\Big(\hat{\mathbf{W}}^T(n-1)\boldsymbol{\Phi}(n)\hat{\mathbf{W}}(n-1) + \sum_{k=1}^{K}\phi_k(n)\boldsymbol{\Sigma}_{\hat{\mathbf{w}}_k}(n-1) + \mathbf{S}(n-1)\Big)^{-1}. \quad (29)$$

Additionally, we define the following fixed-size quantities that implicitly appear in the formulas of the previous section, for $k = 1, 2, \ldots, K$

$$\mathbf{P}_k(n) = \hat{\mathbf{X}}^T(n)\boldsymbol{\Lambda}(n)\boldsymbol{\Phi}_k(n)\hat{\mathbf{X}}(n) + \sum_{j=1}^{n}\phi_k(j)\lambda^{n-j}\boldsymbol{\Sigma}_{\hat{\mathbf{x}}}(j), \quad (30)$$

$$d_k(n) = \boldsymbol{y}_k^T(n)\boldsymbol{\Lambda}(n)\boldsymbol{\Phi}_k(n)\boldsymbol{y}_k(n), \quad (31)$$

$$\mathbf{z}_k(n) = \hat{\mathbf{X}}^T(n)\boldsymbol{\Lambda}(n)\boldsymbol{\Phi}_k(n)\boldsymbol{y}_k(n), \quad (32)$$

$$\mathbf{Q}(n) = \hat{\mathbf{X}}^T(n)\boldsymbol{\Lambda}(n)\hat{\mathbf{X}}(n) + \sum_{j=1}^{n}\lambda^{n-j}\boldsymbol{\Sigma}_{\hat{\mathbf{x}}}(j), \quad (33)$$

$$\mathbf{R}_k(n) = \mathbf{P}_k(n) + \mathbf{S}(n-1). \quad (34)$$

In the previous expressions and with a slight abuse in notation, we have replaced $\boldsymbol{\Sigma}_{\hat{\mathbf{x}}(j)}$ and ϕ_{jk} with $\boldsymbol{\Sigma}_{\hat{\mathbf{x}}}(j)$ and $\phi_k(j)$. It should be further noted that Eqs. (30), (31), (32) and (33) can be updated recursively as follows

$$\mathbf{P}_k(n) = \lambda\mathbf{P}_k(n-1) + \phi_k(n)\left(\boldsymbol{\Sigma}_{\hat{\mathbf{x}}}(n) + \hat{\mathbf{x}}(n)\hat{\mathbf{x}}^T(n)\right), \quad (35)$$

$$d_k(n) = \lambda d_k(n-1) + \phi_k(n)y_k^2(n), \quad (36)$$

$$\mathbf{z}_k(n) = \lambda\mathbf{z}_k(n-1) + \phi_k(n)\hat{\mathbf{x}}(n)y_k(n), \quad (37)$$

$$\mathbf{Q}(n) = \lambda\mathbf{Q}(n-1) + \boldsymbol{\Sigma}_{\hat{\mathbf{x}}}(n) + \hat{\mathbf{x}}(n)\hat{\mathbf{x}}^T(n). \quad (38)$$

These quantities can then be used to produce all remaining parameter estimates of the algorithm. Starting with the estimates of the entries of $\hat{\mathbf{W}}(n)$, (19) and (20) yield

$$\hat{w}_{kl}(n) = \beta(n)\sigma^2_{\hat{w}_{kl}}(n)\left(z_{k,l}(n) - \mathbf{r}^T_{k,\neg l}(n)\hat{\mathbf{w}}_{k\neg l}(n)\right), \quad (39)$$

$$\sigma^2_{\hat{w}_{kl}}(n) = \beta^{-1}(n)r^{-1}_{k,ll}(n), \quad (40)$$

where $r_{k,ll}(n)$ is the lth diagonal element of $\mathbf{R}_k(n)$, $z_{k,l}(n)$ is the lth entry of $\mathbf{z}_k(n)$, $\mathbf{r}^T_{k,\neg l}(n)$ is the lth row of $\mathbf{R}_k(n)$ after removing its lth element $r_{k,ll}(n)$ and

$$\hat{\mathbf{w}}_{k\neg l}(n) = [\hat{w}_{k1}(n), \hat{w}_{k2}(n), \ldots, \hat{w}_{kl-1}(n), \hat{w}_{kl+1}(n-1), \ldots, \hat{w}_{kL}(n-1)]. \quad (41)$$

From (39) and (41), it should be noticed that $\hat{w}_{kl}(n)$ is updated based on the most recent available estimates of the remaining elements of $\hat{\mathbf{w}}_k(n)$ via a coordinate descent type rule. Next, from (24), the column sparsity promoting parameters s_l's are time updated as follows

$$s_l(n) = \Big(2\varsigma_l + \frac{1}{1-\lambda} + K\Big)\Big/\Big(2\delta_l + \beta(n)\big(q_{ll}(n) + \hat{\boldsymbol{w}}_l^T(n)\hat{\boldsymbol{w}}_l(n) + \sum_{k=1}^{K}\sigma^2_{\hat{w}_{kl}}(n)\big)\Big), \quad (42)$$

where $q_{ll}(n)$ is the lth diagonal element of $\mathbf{Q}(n)$ and n has been replaced by the size of the effective time window$(1-\lambda)^{-1}$, as in [7]. Finally, working as in [7], we can derive the following adaptation formula that gives the posterior mean of the noise precision β at time n,

$$\beta(n) = \left(2\kappa + \left(\frac{K+L}{1-\lambda} + KL\right)\right)\Big/\Big(2\theta + \sum_{k=1}^{K}\Big(d_k(n) - \mathbf{z}_k^T(n)\hat{\mathbf{w}}_k(n) + \boldsymbol{\sigma}^T_{\hat{\mathbf{w}}_k}(n)\mathbf{r}_k(n)\Big) + \sum_{l=1}^{L}s_l(n)q_{ll}(n)\Big), \quad (43)$$

where $\boldsymbol{\sigma}_{\hat{\mathbf{w}}_k}(n)$ and $\mathbf{r}_k(n)$ contain the diagonal elements of $\boldsymbol{\Sigma}_{\hat{\mathbf{w}}_k}(n)$ and $\mathbf{R}_k(n)$, respectively.

Based on the previous analysis, the new online variational Bayes subspace learning (OVBSL) algorithm is summarized in Table 1. The estimation of the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}}(n)$ from (29), makes the computational complexity of OVBSL $\mathcal{O}(|\boldsymbol{\phi}(n)|L^2)$ (where $|\boldsymbol{\phi}(n)|$ denotes the number of observed entries at time n), which is similar to that of GROUSE and PETRELS algorithms presented in [2] and [3], respectively, and lower compared to that of Algorithm 1 of [4] which is $\mathcal{O}(KL^3)$. By fixing all hyperparameters $\kappa, \theta, \varsigma_l, \delta_l$ to very small values (as is commonly done in classical sparse Bayesian learning [6]), the proposed algorithm turns out to be fully automated and independent of any free parameters.

Table 1. The OVBSL algorithm

Initialize: $\mathbf{W}(0), \mathbf{S}(0), \boldsymbol{\Sigma}_{\mathbf{w}_k}(0), k = 1, 2, \ldots, K$
Set $\mathbf{R}_k(0) = \mathbf{0}, \mathbf{P}_k(0) = \mathbf{0}, \mathbf{z}_k(0) = \mathbf{0}, d_k(0) = 0, k = 1, 2, \ldots, K$
Set $\mathbf{Q}(0) = \mathbf{0}, \kappa = 10^{-6}, \theta = 10^{-6}, \lambda$
Set $\varsigma_l = 10^{-6}, \delta_l = 10^{-6}, l = 1, 2, \ldots, L$
for $n = 1, 2, \ldots$
 1: Compute $\hat{\boldsymbol{\Sigma}}_{\mathbf{x}}(n)$ from (29)
 2: Compute $\hat{\mathbf{x}}(n)$ from (28)
 for $k = 1, 2, \ldots, K$
  3: Update $\mathbf{P}_k(n)$ from (35)
  4: Update $d_k(n)$ from (36)
  5: Update $\mathbf{z}_k(n)$ from (37)
  6: Compute $\mathbf{R}_k(n)$ from (34)
  for $l = 1, 2, \ldots, L$
   8: Compute $\sigma^2_{w_{kl}}(n)$ from (40)
   9: Compute $\hat{w}_{kl}(n)$ from (39)
  end for
 end for
 7: Update $\mathbf{Q}(n)$ from (38)
 for $l = 1, 2, \ldots, L$
  10: Compute $s_l(n)$ from (42)
 end for
  11: Compute $\beta(n)$ from (43)
end for

6. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we evaluate the effectiveness of the proposed OVBSL algorithm by conducting two experiments on simulated data. To this end, we produce low-dimensional subspace matrices $\mathbf{W}_{true} \in \mathbb{R}^{400\times r}$ (hence $K = 400$) with independent and identically distributed (i.i.d) $\mathcal{N}(0, \frac{1}{K})$ entries. Next, we generate 30000 projection coefficient vectors, whose entries are Gaussian distributed, i.e., $\mathbf{x}_{true}(n) \sim \mathcal{N}(\mathbf{x}_{true}(n) \mid \mathbf{0}, \mathbf{I}_r)$. The observation vectors $\mathbf{y}(n)$ are generated according to (1), where the noise precision is set to $\beta = 10^3$ while the forgetting factor λ is set to 0.99. In our experiments, the proposed OVBSL algorithm is compared to three state-of-the-art schemes namely, GROUSE, [2], PETRELS, [3] and Algorithm 1 of [4]. The step size of GROUSE as well as the low-rankness parameter of Algorithm 1 of [4], are set to 0.1. Let $\mathbf{a}(n) = \mathbf{W}_{true}(n)\mathbf{x}_{true}(n)$. We consider two metrics, namely, the running average estimation error (RAEE), $\text{RAEE}(n) = \frac{1}{n}\sum_{j=1}^{n} \frac{\|\hat{\mathbf{a}}(j)-\mathbf{a}(j)\|}{\|\mathbf{a}(j)\|}$, where $\hat{\mathbf{a}}(j) = \hat{\mathbf{W}}(j)\hat{\mathbf{x}}(j)$, and the normalized subspace reconstruction error (NSRE), $\text{NSRE} = \|\mathcal{P}_{\hat{\mathbf{W}}^{\perp}(n)}\mathbf{W}_{true}(n)\|_F^2/\|\mathbf{W}_{true}(n)\|_F^2$, where $\mathcal{P}_{\hat{\mathbf{W}}^{\perp}(n)}$ is the projection operator onto the orthogonal complement of $\hat{\mathbf{W}}(n)$.

In the first experiment, we compare the robustness of the considered algorithms for two different ratios π (percentage of missing elements), namely $\pi = \{0.25, 0.75\}$. The true rank r of $\mathbf{W}_{true}$ is set to 5. In all tested algorithms, an overestimate, $L = 10$, of the true rank r was used. Fig. 1 shows the obtained RAEE curves for all four algorithms. It is easily observed that the proposed algorithm outperforms its rivals, for both ratio values. As shown in the figure, without the knowledge of the true rank, GROUSE is trapped into local minima, while PETRELS diverges for $\pi = 0.75$. In contrast, OVBSL and Algorithm 1 of [4] present robustness and relatively low RAEE values. Note that for $\pi = 0.25$ an abrupt change is purposely induced in our model at $n = 10000$, by fully updating the subspace matrix without changing its rank. Although this change causes a sudden increase in the RAEE of OVBSL and Algorithm 1 of [4], both algorithms are able to track the subspace change in subsequent iterations.

The second experiment examines the ability of OVBSL and Algorithm 1 of [4] in detecting the true rank of the subspace matrix $\mathbf{W}_{true}$. In this regard, we fix π to 0.25, and generate four $\mathbf{W}_{true}$'s with ranks $r = 6, 8, 10, 12$. In both tested algorithms, the rank of the subspace matrix is intialized to $L = 15$. Table 2 shows the rank $\hat{r}$ of $\hat{\mathbf{W}}$ and the NSRE provided by the two algorithms after 30000 iterations. Interestingly, the proposed OVBSL algorithm detects the true rank in all cases, in contrast to its competitor, which is shown to end up always with an overestimate. As far the NSRE is concerned, OVBSL scores better results in all examined cases. Therefore, it seems that, besides its lower computational complexity, OVBSL also exhibits a better estimation performance. This is due to the fact that OVBSL entails a sparsity promoting mechanism that explicitly imposes low-rankness by zeroing whole columns of the subspace matrix.

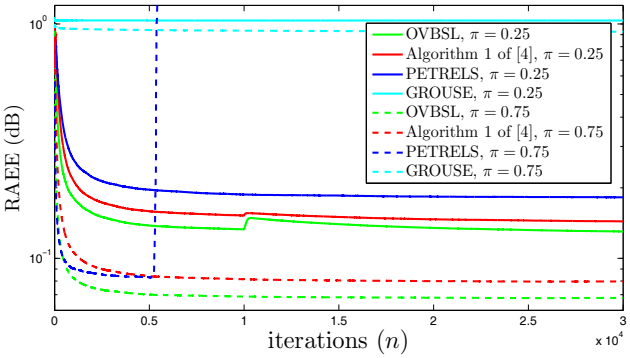

Fig. 1. RAEE of subspace tracking algorithms for $\pi = 0.25$ and $\pi = 0.75$.

Algorithm	$r = 6$		$r = 8$		$r = 10$		$r = 12$	
	$\hat{r}$	NSRE	$\hat{r}$	NSRE	$\hat{r}$	NSRE	$\hat{r}$	NSRE
OVBSL	6	0.0843	8	0.0850	10	0.0893	12	0.0909
Algorithm 1 of [4]	15	0.0889	15	0.0900	15	0.0940	15	0.0943

Table 2. Estimated rank $\hat{r}$ and NSRE of OVBSL and Algorithm 1 of [4].

REFERENCES

[1] P. Yang, "Projection approximation subspace tracking," *IEEE Trans. Sign. Process.*, vol. 43, no. 1, pp. 95–107, Jan. 1995.

[2] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Allerton Conference*, Sept. 2010.

[3] Y. Chi, Y.C. Eldar, and R. Calderbank, "PETRELS: parallel subspace estimation and tracking by recursive least squares from partial observations," *IEEE Trans. Sign. Process.*, vol. 61, no. 23, pp. 5947–5959, Dec. 2013.

[4] M. Mardani, G. Mateos, and G.B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Trans. Sign. Process.*, vol. 63, no. 10, pp. 2663–2677, May 2015.

[5] S.D. Babacan, M. Luessi, R. Molina, and A.K. Katsaggelos, "Sparse Bayesian methods for low-rank matrix estimation," *IEEE Trans. Sign. Process.*, vol. 60, no. 8, pp. 3964–3977, Aug. 2012.

[6] M.E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.

[7] K.E. Themelis, A.A. Rontogiannis, and K.D. Koutroumbas, "A variational Bayes framework for sparse adaptive estimation," *IEEE Trans. Sign. Process.*, vol. 62, no. 18, pp. 4723–4736, Sept. 2014.