Signal Processing 130 (2017) 93–104

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Sparse Bayesian dictionary learning with a Gaussian hierarchical model[☆]

Linxiao Yang [a], Jun Fang [a,*], Hong Cheng [b], Hongbin Li [c]

[a] National Key Laboratory on Communications, University of Electronic Science and Technology of China, Chengdu 611731, China
[b] School of Automation, University of Electronic Science and Technology of China, Chengdu 611731, China
[c] Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA

ARTICLE INFO

Article history:
Received 8 July 2015
Received in revised form
14 June 2016
Accepted 16 June 2016
Available online 21 June 2016

Keywords:
Dictionary learning
Gaussian-inverse Gamma prior
Variational Bayesian
Gibbs sampling

ABSTRACT

We consider a dictionary learning problem aimed at designing a dictionary such that the signals admit a sparse or an approximate sparse representation over the learnt dictionary. The problem finds a variety of applications including image denoising, feature extraction, etc. In this paper, we propose a new hierarchical Bayesian model for dictionary learning, in which a Gaussian-inverse Gamma hierarchical prior is used to promote the sparsity of the representation. Suitable non-informative priors are also placed on the dictionary and the noise variance such that they can be reliably estimated from the data. Based on the hierarchical model, a variational Bayesian method and a Gibbs sampling method are developed for Bayesian inference. The proposed methods have the advantage that they do not require the knowledge of the noise variance a priori. Numerical results show that the proposed methods are able to learn the dictionary with an accuracy better than existing methods, particularly for the case where there is a limited number of training signals.

1. Introduction

Sparse representation has been of significant interest over the past few years. It has found a variety of applications in practice as many natural signals admit a sparse or approximately sparse representation in a certain basis [1–3]. In many applications such as image denoising and interpolation, signals often have a sparse representation over a pre-specified non-adaptive dictionary, e.g. discrete cosine/wavelet transform (DCT/DWT) bases. Nevertheless, recent research [4,5] has shown that the recovery, denoising and classification performance can be considerably improved by utilizing an adaptive dictionary that is learnt from training signals [5,6]. This has inspired studies on dictionary learning aimed to design overcompelete dictionaries that can better represent the signals. A number of algorithms, such as the K-singular value decomposition (K-SVD) [4], the method of optimal directions (MOD) [7], dictionary learning with the majorization method [8], and the simultaneous codeword optimization (SimCO) [9], were developed for overcomplete dictionary learning and sparse representation. Most algorithms formulate the dictionary learning as an optimization problem which is solved via a two-stage iterative process, namely, a sparse coding stage and a dictionary update stage. The main difference among these algorithms lies in the dictionary update stage. Specifically, the MOD method [7] updates the dictionary via solving a least square problem which admits a closed-form solution for dictionary update. The K-SVD algorithm [4], instead, updates the atoms of the dictionary in a sequential manner and while updating each atom, the atom is updated along with the nonzero entries in the corresponding row vector of the sparse matrix. The idea of sequential atom update was later extended to provide sequential update of multiple atoms each time [9], and recently generalized to parallel atom-updating in order to further accelerate the convergence of the iterative process [10]. These methods [4,7–10], although offering state-of-the-art performance, have several limitations. Specifically, they may require the knowledge of the sparsity level or the noise/residual variance for sparse coding (e.g. [4]), or this knowledge is needed for meticulously selecting some regularization parameters to properly control the tradeoff between the sparsity level and the data fitting error (e.g. [8,10]). In practice, however, the prior information about the noise variance and sparsity level is usually unavailable and an inaccurate estimation may result in substantial performance degradation. To mitigate these limitations, a nonparametric Bayesian dictionary learning method called beta-Bernoulli process factor analysis (BPFA) was recently developed in [11]. The proposed method can estimate the usage frequency of each atom, based on which the required number of atoms can be automatically inferred. Moreover, BPFA is also able to

[☆]This work was supported in part by the National Science Foundation of China under Grants 61428103, 61522104, and the National Science Foundation under Grant ECCS-1408182.

[*] Corresponding author.

E-mail addresses: JunFang@uestc.edu.cn, 201321190224@std.uestc.edu.cn (J. Fang), hcheng@uestc.edu.cn (H. Cheng), Hongbin.Li@stevens.edu (H. Li).

http://dx.doi.org/10.1016/j.sigpro.2016.06.016
0165-1684/© 2016 Elsevier B.V. All rights reserved.

automatically infer the noise variance from the test image. These merits are deemed an important advantage over other dictionary learning methods. For [11], the posterior distributions cannot be derived analytically, and a Gibbs sampler was used for Bayesian inference. We also note that a class of online dictionary learning algorithms were developed in [12–16]. Unlike the above batch-based algorithms [4,7,9,10] which use the whole set of training data for dictionary learning, online algorithms continuously update the dictionary using only one or a few (or a small amount of) training data, which enables them to handle very large data sets.

In this paper, we propose a new hierarchical Bayesian model for dictionary learning, in which a Gaussian-inverse Gamma hierarchical prior [17,18] is used to promote the sparsity of the representation. Suitable non-informative priors are also placed on the dictionary and the noise variance such that they can be reliably inferred from the data. Based on the hierarchical model, a variational Bayesian method [19–21] and a Gibbs sampling method [22] are developed for Bayesian inference. For both inference methods, there are two different ways to update the dictionary: we can either update the whole set of atoms in one iteration, or update the atoms in a sequential manner. When updating the dictionary as a whole, the proposed variational Bayesian method has a dictionary update formula similar to the MOD method. For the Gibbs sampler, a sequential update seems to be able to expedite the convergence rate and helps achieve additional performance gain. Simulation results show that the proposed Gibbs sampling algorithm has notable advantages over other state-of-the-art dictionary learning methods in a number of interesting scenarios.

Note that the Gaussian-inverse Gamma hierarchical prior used in our paper is quite different from the beta-Bernoulli (also referred to as the spike-and-slab) prior employed in [11]. These two priors have their respective merits and both are widely used to promote the sparsity of solutions. In particular, the use of the Gaussian-inverse Gamma prior for sparse Bayesian learning has achieved great success in the framework of compressed sensing, e.g. [23–26]. It is therefore interesting to examine the problem of dictionary learning with such a prior and see if an additional performance improvement can be achieved. Note that the sparsity-promoting prior model (i.e. the hierarchical Gaussian-inverse Gamma prior) employed in this paper was also used in the sparse PCA framework (e.g. [27]). Nevertheless, to our best knowledge, our paper presents a first attempt to use the hierarchical Gaussian-inverse Gamma prior model to solve the dictionary learning problem. Although dictionary learning is closely related to sparse PCA [27], they still are two different problems with very distinct objectives: dictionary learning tries to learn an overcomplete dictionary to sparsely represent the observed data, whereas the sparse PCA aims to find a few sparse principle components of the underlying data matrix. Also, although sharing some degree of similarity, the prior model used in our paper is not exactly the same as the prior model in [27]. As a consequence, the derivations, update rules, and choice of model parameters in our work are different from those in [27]. Our work also provides an interesting comparison between two different inference methods, namely, the variational Bayes and the Gibbs sampling, for dictionary learning.

The rest of the paper is organized as follows. In Section 2, we introduce a hierarchical prior model for dictionary learning. Based on this hierarchical model, a variational Bayesian method and a Gibbs sampler are developed in Sections 3 and 4 for Bayesian inference. Simulation results are provided in Section 5, followed by concluding remarks in Section 6.

2. Hierarchical model

Suppose we have L training signals $\{\boldsymbol{y}_l\}_{l=1}^{L}$, where $\boldsymbol{y}_l \in \mathbb{R}^M$. Dictionary learning aims at finding a common sparsifying dictionary $\boldsymbol{D} \in \mathbb{R}^{M\times N}$ such that these L training signals admit a sparse representation over the overcomplete dictionary $\boldsymbol{D}$, i.e.

$$\boldsymbol{y}_l = \boldsymbol{D}\boldsymbol{x}_l + \boldsymbol{w}_l \quad \forall\, l, \tag{1}$$

where $\boldsymbol{x}_l$ and $\boldsymbol{w}_l$ denote the sparse vector and the residual/noise vector, respectively. Define $\boldsymbol{Y} \triangleq [\boldsymbol{y}_1 \ \dots \ \boldsymbol{y}_L]$, $\boldsymbol{X} \triangleq [\boldsymbol{x}_1 \ \dots \ \boldsymbol{x}_L]$, and $\boldsymbol{W} \triangleq [\boldsymbol{w}_1 \ \dots \ \boldsymbol{w}_L]$. The model (1) can be re-expressed as

$$\boldsymbol{Y} = \boldsymbol{D}\boldsymbol{X} + \boldsymbol{W}. \tag{2}$$

Also, we write $\boldsymbol{D} \triangleq [\boldsymbol{d}_1 \ \dots \ \boldsymbol{d}_N]$, where each column of the dictionary, $\boldsymbol{d}_n$, is called an atom.

In the following, we develop a Bayesian framework for learning the overcomplete dictionary and sparse vectors. To promote sparse representations, we assign a two-layer hierarchical Gaussian-inverse Gamma prior to $\boldsymbol{X}$. The Gaussian-inverse Gamma prior is one of the most popular sparsity-promoting priors which has been widely used in compressed sensing [23,24,28]. Specifically, in the first layer, $\boldsymbol{X}$ is assigned a Gaussian prior distribution

$$p(\boldsymbol{X}|\boldsymbol{\alpha}) = \prod_{n=1}^{N}\prod_{l=1}^{L} p(x_{nl}) = \prod_{n=1}^{N}\prod_{l=1}^{L} \mathcal{N}(x_{nl}|0, \alpha_{nl}^{-1}), \tag{3}$$

where x_{nl} denotes the (n, l)th entry of $\boldsymbol{X}$, and $\boldsymbol{\alpha} \triangleq \{\alpha_{nl}\}$ are non-negative sparsity-controlling hyperparameters. The notation $\mathcal{N}(x_{nl}|0, \alpha_{nl}^{-1})$ denotes Gaussian distribution with zero mean and variance α_{nl}^{-1}. The second layer specifies Gamma distributions as hyperpriors over the hyperparameters $\{\alpha_{nl}\}$, i.e.

$$p(\boldsymbol{\alpha}) = \prod_{n=1}^{N}\prod_{l=1}^{L} \text{Gamma}(\alpha_{nl}; a, b) = \prod_{n=1}^{N}\prod_{l=1}^{L} \Gamma(a)^{-1} b^a \alpha_{nl}^{a-1} e^{-b\alpha_{nl}}, \tag{4}$$

where $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$ is the Gamma function. Here the notation $\text{Gamma}(\alpha_{nl}; a, b)$ denotes the Gamma distribution of α_{nl} with parameters a and b. To illustrate the sparsity-promoting property of the Gaussian-inverse Gamma prior, we integrate out the hyperparameter α_{nl} and obtain the marginal distribution of x_{nl}, which was shown to be a student-t distribution, i.e.

$$\begin{aligned} p(x_{nl}) &= \int p(x_{nl}|\alpha_{nl}) p(\alpha_n l; a, b) d\alpha_{nl} \\ &= \frac{b^a \Gamma(a+0.5)}{(2\pi)^{1/2}\Gamma(a)} \left(b + \frac{x_{nl}^2}{2}\right)^{-(a+0.5)}. \end{aligned} \tag{5}$$

When b is very small, say $b = 10^{-6}$, the student-t distribution can be reduced to

$$p(x_{nl}) \propto \left(\frac{1}{x_{nl}^2}\right)^{(a+0.5)}. \tag{6}$$

We can easily see that (6) is a sparsity-promoting prior. Fig. 1 plots the student-t distributions with different choices of a and b. We see that the distribution has a sharp peak around zero when b is sufficiently small. Also, a larger a results in a sharper peak, which implies that a larger a leads to a more sparsity-encouraging prior. In our paper, the parameters a and b are chosen to be a=0.5 and $b = 10^{-6}$.

In addition, in order to prevent the entries in the dictionary from becoming infinitely large, we assume that the atoms of the dictionary $\{\boldsymbol{d}_n\}$ are mutually independent, and upon each atom we place a Gaussian prior, i.e.

$$p(\boldsymbol{D}) = \prod_{n=1}^{N} p(\boldsymbol{d}_n) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{d}_n|\boldsymbol{0}, \beta\boldsymbol{I}), \tag{7}$$

where β is a parameter whose choice will be discussed later. The noise $\{\boldsymbol{w}_l\}$ are assumed independent multivariate Gaussian noise with zero mean and covariance matrix $(1/\gamma)\boldsymbol{I}$, where the noise

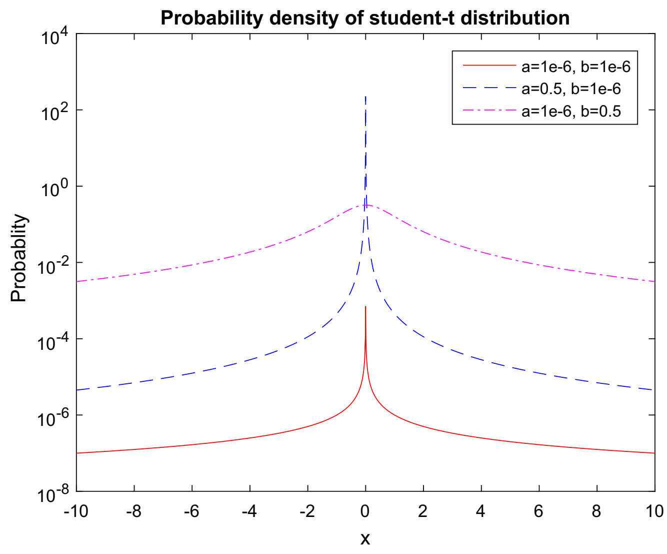

Fig. 1. Student-*t* distribution with different parameter settings.

variance $1/\gamma$ is assumed unknown a priori. To estimate the noise variance, we place a Gamma hyperprior over γ, i.e.

$$p(\gamma) = \text{Gamma}(\gamma; c, d) = \Gamma(c)^{-1} d^c \gamma^{c-1} e^{-d\gamma}, \tag{8}$$

where we set $c=0.5$ and $d = 10^{-6}$ to better suppress the data fitting error since a large c encourages a large value of γ, i.e. a small noise variance. The proposed hierarchical model provides a general framework for learning the overcomplete dictionary, the sparse codes, as well as the noise variance. In the following, we develop a variational Bayesian method and a Gibbs sampling method for Bayesian inference.

3. Variational inference

3.1. Review of the variational Bayesian methodology

Before proceeding, we firstly provide a brief review of the variational Bayesian methodology. In a probabilistic model, let $\boldsymbol{y}$ and $\boldsymbol{\theta}$ denote the observed data and the hidden variables, respectively. It is straightforward to show that the marginal probability of the observed data can be decomposed into two terms [22]:

$$\ln p(\boldsymbol{y}) = L(q) + \text{KL}(q \parallel p), \tag{9}$$

where

$$L(q) = \int q(\boldsymbol{\theta}) \ln \frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} \tag{10}$$

and

$$\text{KL}(q \parallel p) = -\int q(\boldsymbol{\theta}) \ln \frac{p(\boldsymbol{\theta}|\boldsymbol{y})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}, \tag{11}$$

where $q(\boldsymbol{\theta})$ is any probability density function, $\text{KL}(q \parallel p)$ is the Kullback–Leibler divergence [29] between $p(\boldsymbol{\theta}|\boldsymbol{y})$ and $q(\boldsymbol{\theta})$. Since $\text{KL}(q \parallel p) \geq 0$, it follows that $L(q)$ is a rigorous lower bound for $\ln p(\boldsymbol{y})$. Moreover, notice that the left-hand side of (9) is independent of $q(\boldsymbol{\theta})$. Therefore maximizing $L(q)$ is equivalent to minimizing $\text{KL}(q \parallel p)$, and thus the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$ can be approximated by $q(\boldsymbol{\theta})$ through maximizing $L(q)$.

The significance of the above transformation is that it circumvents the difficulty of computing the posterior probability $p(\boldsymbol{\theta}|\boldsymbol{y})$, when it is computationally intractable. For a suitable choice for the distribution $q(\boldsymbol{\theta})$, the quantity $L(q)$ may be more amiable to compute. Specifically, we could assume some specific parameterized functional form for $q(\boldsymbol{\theta})$ and then maximize $L(q)$ with respect to the parameters of the distribution. A particular form of $q(\boldsymbol{\theta})$ that has been widely used with great success is the factorized form over the component variables $\{\theta_i\}$ in $\boldsymbol{\theta}$ [19], i.e. $q(\boldsymbol{\theta}) = \prod_i q_i(\theta_i)$. We therefore can compute the posterior distribution approximation by finding $q(\boldsymbol{\theta})$ of the factorized form that maximizes the lower bound $L(q)$. The maximization can be conducted in an alternating fashion for each latent variable, which leads to [19]

$$q_i(\theta_i) = \frac{e^{\langle \ln p(\boldsymbol{y}, \boldsymbol{\theta}) \rangle_{k \neq i}}}{\int e^{\langle \ln p(\boldsymbol{y}, \boldsymbol{\theta}) \rangle_{k \neq i}} d\theta_i}, \tag{12}$$

where $\langle \cdot \rangle_{k \neq i}$ denotes the expectation with respect to the distributions $q_k(\theta_k)$ for all $k \neq i$. By taking the logarithm on both sides of (12), it can be equivalently written as

$$\ln q_i(\theta_i) = \langle \ln p(\boldsymbol{y}, \boldsymbol{\theta}) \rangle_{k \neq i} + \text{constant}. \tag{13}$$

3.2. Proposed variational Bayesian method

We now proceed to perform variational Bayesian inference for the proposed hierarchical model. Let $\boldsymbol{\theta} \triangleq \{\boldsymbol{X}, \boldsymbol{\alpha}, \boldsymbol{D}, \gamma\}$ denote all hidden variables. Our objective is to find the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$. Since $p(\boldsymbol{\theta}|\boldsymbol{y})$ is usually computationally intractable, we, following the idea of [19], approximate $p(\boldsymbol{\theta}|\boldsymbol{y})$ by $q(\boldsymbol{X}, \boldsymbol{\alpha}, \boldsymbol{D}, \gamma)$ which has a factorized form over the hidden variables $\{\boldsymbol{X}, \boldsymbol{\alpha}, \boldsymbol{D}, \gamma\}$, i.e.

$$q(\boldsymbol{X}, \boldsymbol{\alpha}, \boldsymbol{D}, \gamma) = q_x(\boldsymbol{X}) q_\alpha(\boldsymbol{\alpha}) q_d(\boldsymbol{D}) q_\gamma(\gamma). \tag{14}$$

It is noted that such a factorized form leads to a compact estimate of $p(\boldsymbol{\theta}|\boldsymbol{y})$ with the correct mean [22]. We then maximize $L(q)$ defined in (10) with respect to $q_x(\boldsymbol{X})$, $q_\alpha(\boldsymbol{\alpha})$, $q_d(\boldsymbol{D})$, and $q_\gamma(\gamma)$, i.e.

$$\max_{\{q_x(\boldsymbol{X}), q_\alpha(\boldsymbol{\alpha}), q_d(\boldsymbol{D}), q_\gamma(\gamma)\}} L(q). \tag{15}$$

As mentioned in the previous subsection, the maximization of $L(q)$ can be conducted in an alternating fashion for each latent variable, which leads to (details of the derivation can be found in [19])

$$\begin{aligned}
\ln q_x(\boldsymbol{X}) &= \langle \ln p(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{\alpha}, \gamma) \rangle_{q_d(\boldsymbol{D}) q_\alpha(\boldsymbol{\alpha}) q_\gamma(\gamma)} + \text{constant}, \\
\ln q_d(\boldsymbol{D}) &= \langle \ln p(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{\alpha}, \gamma) \rangle_{q_x(\boldsymbol{X}) q_\alpha(\boldsymbol{\alpha}) q_\gamma(\gamma)} + \text{constant}, \\
\ln q_\alpha(\boldsymbol{\alpha}) &= \langle \ln p(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{\alpha}, \gamma) \rangle_{q_x(\boldsymbol{X}) q_d(\boldsymbol{D}) q_\gamma(\gamma)} + \text{constant}, \\
\ln q_\gamma(\gamma) &= \langle \ln p(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{\alpha}, \gamma) \rangle_{q_x(\boldsymbol{X}) q_d(\boldsymbol{D}) q_\alpha(\boldsymbol{\alpha})} + \text{constant},
\end{aligned}$$

where $\langle \rangle_{q_1(\cdot) \ldots q_K(\cdot)}$ denotes the expectation with respect to (w.r.t.) the distributions $\{q_k(\cdot)\}_{k=1}^K$. In summary, the posterior distribution approximations are computed in an alternating fashion for each hidden variable, with the distribution of other variables fixed. Details of this Bayesian inference scheme are provided next.

(1) *Update of* $q_x(\boldsymbol{X})$: The calculation of $q_x(\boldsymbol{X})$ can be decomposed into a set of independent tasks, with each task computing the posterior distribution approximation for each column of $\boldsymbol{X}$, i.e. $q_x(\boldsymbol{x}_l)$. We have

$$\ln q_x(\boldsymbol{x}_l) \propto \langle \ln[p(\boldsymbol{y}_l|\boldsymbol{D}, \boldsymbol{x}_l, \gamma) p(\boldsymbol{x}_l|\boldsymbol{\alpha}_l)] \rangle_{q_d(\boldsymbol{D}) q_\alpha(\boldsymbol{\alpha}) q_\gamma(\gamma)}, \tag{16}$$

where $\boldsymbol{\alpha}_l \triangleq \{\alpha_{nl}\}_{n=1}^N$ are the sparsity-controlling hyperparameters associated with $\boldsymbol{x}_l$, $p(\boldsymbol{y}_l|\boldsymbol{D}, \boldsymbol{x}_l, \gamma)$ and $p(\boldsymbol{x}_l|\boldsymbol{\alpha}_l)$ are, respectively, given by

$$p(\boldsymbol{y}_l|\boldsymbol{D}, \boldsymbol{x}_l, \gamma) = \left(\frac{\gamma}{2\pi}\right)^{\frac{M}{2}} e^{-\frac{\gamma\|\boldsymbol{y}_l-\boldsymbol{D}\boldsymbol{x}_l\|_2^2}{2}},$$
$$p(\boldsymbol{x}_l|\boldsymbol{\alpha}_l) = \prod_{n=1}^{N} \mathcal{N}(x_{nl}|0, \alpha_{nl}^{-1}). \tag{17}$$

Substituting (17) into (16) and after some simplifications, it can be readily verified that $q_x(\boldsymbol{x}_l)$ follows a Gaussian distribution

$$q_x(\boldsymbol{x}_l) = \mathcal{N}(\boldsymbol{x}_l|\boldsymbol{\mu}_l^x, \boldsymbol{\Sigma}_l^x) \tag{18}$$

with its mean $\boldsymbol{\mu}_l^x$ and covariance matrix $\boldsymbol{\Sigma}_l^x$ given, respectively, as

$$\boldsymbol{\mu}_l^x = \langle\gamma\rangle\boldsymbol{\Sigma}_l^x\langle\boldsymbol{D}\rangle^T\boldsymbol{y}_l,$$
$$\boldsymbol{\Sigma}_l^x = \left(\langle\gamma\rangle\langle\boldsymbol{D}^T\boldsymbol{D}\rangle + \langle\boldsymbol{\Lambda}_l\rangle\right)^{-1}, \tag{19}$$

where $\langle\gamma\rangle$ denotes the expectation w.r.t. $q_\gamma(\gamma)$, $\langle\boldsymbol{D}\rangle$ and $\langle\boldsymbol{D}^T\boldsymbol{D}\rangle$ denote the expectation w.r.t. $q_d(\boldsymbol{D})$, and $\langle\boldsymbol{\Lambda}_l\rangle \triangleq \text{diag}(\langle\alpha_{1l}\rangle, \ldots, \langle\alpha_{Nl}\rangle)$, in which $\langle\alpha_{nl}\rangle$ represents the expectation w.r.t. $q_\alpha(\boldsymbol{\alpha})$. All these expectations are given by (30)–(35).

(2) *Update of* $q_d(\boldsymbol{D})$: The approximate posterior $q_d(\boldsymbol{D})$ can be obtained as

$$\begin{aligned}\ln q_d(\boldsymbol{D}) &\propto \langle \ln[p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{D}, \gamma)p(\boldsymbol{D})]\rangle_{q_x(\boldsymbol{X})q_\gamma(\gamma)}\\ &\propto \left\langle -\gamma \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2 - \beta^{-1}\sum_{n=1}^{N}\boldsymbol{d}_n^T\boldsymbol{d}_n\right\rangle\\ &= \langle -\gamma\,\text{tr}\{(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X})(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X})^T\} - \beta^{-1}\,\text{tr}\{\boldsymbol{D}\boldsymbol{D}^T\}\rangle\\ &\propto \langle \text{tr}\{\boldsymbol{D}(\gamma\boldsymbol{X}\boldsymbol{X}^T + \beta^{-1}\boldsymbol{I})\boldsymbol{D}^T - 2\gamma\boldsymbol{Y}\boldsymbol{X}^T\boldsymbol{D}^T\}\rangle\\ &= \text{tr}\{\boldsymbol{D}(\langle\gamma\rangle\langle\boldsymbol{X}\boldsymbol{X}^T\rangle + \beta^{-1}\boldsymbol{I})\boldsymbol{D}^T - 2\langle\gamma\rangle\boldsymbol{Y}\langle\boldsymbol{X}\rangle^T\boldsymbol{D}^T\},\end{aligned} \tag{20}$$

where for simplicity, we have dropped the subscripts of the $\langle\cdot\rangle$ operator. Define

$$\boldsymbol{A} \triangleq (\langle\gamma\rangle\langle\boldsymbol{X}\boldsymbol{X}^T\rangle + \beta^{-1}\boldsymbol{I})^{-1},$$
$$\boldsymbol{B} \triangleq \langle\gamma\rangle\boldsymbol{Y}\langle\boldsymbol{X}\rangle^T,$$

The posterior $q_d(\boldsymbol{D})$ can be further expressed as

$$\ln q_d(\boldsymbol{D}) \propto \text{tr}\{\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{D}^T - 2\boldsymbol{B}\boldsymbol{D}^T\} = \sum_{m=1}^{M}(\boldsymbol{d}_{m\cdot}\boldsymbol{A}^{-1}\boldsymbol{d}_{m\cdot}^T - 2\boldsymbol{b}_{m\cdot}\boldsymbol{d}_{m\cdot}^T), \tag{21}$$

where $\boldsymbol{b}_{m\cdot}$ and $\boldsymbol{d}_{m\cdot}$ represents the mth row of $\boldsymbol{B}$ and $\boldsymbol{D}$, respectively. It can be easily seen from (21) that the posterior distribution $q_d(\boldsymbol{D})$ has independent rows and each row follows a Gaussian distribution with its mean and covariance matrix given by $\boldsymbol{b}_{m\cdot}\boldsymbol{A}$ and $\boldsymbol{A}$, respectively, i.e.

$$q_d(\boldsymbol{D}) = \prod_{m=1}^{M} p(\boldsymbol{d}_{m\cdot}) = \prod_{m=1}^{M} \mathcal{N}(\boldsymbol{b}_{m\cdot}\boldsymbol{A}, \boldsymbol{A}). \tag{22}$$

(3) *Update of* $q_\alpha(\boldsymbol{\alpha})$: The variational optimization of $q_\alpha(\boldsymbol{\alpha})$ yields

$$\begin{aligned}\ln q_\alpha(\boldsymbol{\alpha}) &\propto \langle \ln p(\boldsymbol{X}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})\rangle_{q_x(\boldsymbol{X})} = \sum_{n=1}^{N}\sum_{l=1}^{L}\langle \ln p(x_{nl}|\alpha_{nl})p(\alpha_{nl}; a, b)\rangle\\ &\propto \sum_{n=1}^{N}\sum_{l=1}^{L}\left\{\left(a-\frac{1}{2}\right)\ln\alpha_{nl} - \left(b+\frac{\langle x_{nl}^2\rangle}{2}\right)\alpha_{nl}\right\}.\end{aligned} \tag{23}$$

Thus $q_\alpha(\boldsymbol{\alpha})$ has a form of a product of Gamma distributions

$$q_\alpha(\boldsymbol{\alpha}) = \prod_{n=1}^{N}\prod_{l=1}^{L}\text{Gamma}(\alpha_{nl}; \tilde{a}, \tilde{b}_{nl}), \tag{24}$$

in which the parameters $\tilde{a}$ and $\tilde{b}_{nl}$ are, respectively, given as

$$\tilde{a} = a + \frac{1}{2}, \quad \tilde{b}_{nl} = b + \frac{1}{2}\langle x_{nl}^2\rangle. \tag{25}$$

(4) *Update of* $q_\gamma(\gamma)$: The variational optimization of $q_\gamma(\gamma)$ yields

$$\begin{aligned}\ln q_\gamma(\gamma) &\propto \langle \ln p(\boldsymbol{Y}|\boldsymbol{D}, \boldsymbol{X}, \gamma)p(\gamma)\rangle_{q_d(\boldsymbol{D})q_x(\boldsymbol{X})} \propto \left\langle \ln \prod_{l=1}^{L} p(\boldsymbol{y}_l|\boldsymbol{D}, \boldsymbol{x}_l, \gamma)p(\gamma)\right\rangle\\ &\propto \left\langle \frac{ML}{2}\ln\gamma - \frac{\gamma}{2}\sum_{l=1}^{L}(\boldsymbol{y}_l-\boldsymbol{D}\boldsymbol{x}_l)^T(\boldsymbol{y}_l-\boldsymbol{D}\boldsymbol{x}_l) + (c-1)\ln\gamma - d\gamma\right\rangle\\ &= \left(\frac{ML}{2}+c-1\right)\ln\gamma - \left(\frac{1}{2}\langle\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2\rangle + d\right)\gamma.\end{aligned} \tag{26}$$

Therefore $q_\gamma(\gamma)$ follows a Gamma distribution

$$q_\gamma(\gamma) = \text{Gamma}(\gamma|\tilde{c}, \tilde{d}) \tag{27}$$

with the parameters $\tilde{c}$ and $\tilde{d}$ given, respectively, by

$$\tilde{c} = \frac{ML}{2} + c,$$
$$\tilde{d} = d + \frac{1}{2}\langle\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2, \rangle \tag{28}$$

where

$$\begin{aligned}\langle\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2\rangle &= \langle \text{tr}\{(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X})^T(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X})\}\rangle\\ &= \|\boldsymbol{Y}-\langle\boldsymbol{D}\rangle\langle\boldsymbol{X}\rangle\|_F^2 + \text{tr}\{\langle\boldsymbol{D}^T\boldsymbol{D}\rangle\langle\boldsymbol{X}\boldsymbol{X}^T\rangle\}\\ &\quad - \text{tr}\{\langle\boldsymbol{D}^T\rangle\langle\boldsymbol{D}\rangle\langle\boldsymbol{X}\rangle\langle\boldsymbol{X}^T\rangle\}.\end{aligned} \tag{29}$$

In summary, the variational Bayesian inference involves updates of the approximate posterior distributions for hidden variables $\boldsymbol{X}$, $\boldsymbol{D}$, $\boldsymbol{\alpha}$, and γ. Some of the expectations and moments used during the update are summarized as

$$\langle x_{nl}^2\rangle = (\boldsymbol{\mu}_l^x[n])^2 + \boldsymbol{\Sigma}_l^x[n, n], \tag{30}$$

$$\langle\boldsymbol{X}\boldsymbol{X}^T\rangle = \langle\boldsymbol{X}\rangle\langle\boldsymbol{X}\rangle^T + \sum_{l=1}^{L}\boldsymbol{\Sigma}_l^x, \tag{31}$$

$$\langle\boldsymbol{D}\rangle = \boldsymbol{B}\boldsymbol{A}, \tag{32}$$

$$\langle\boldsymbol{D}^T\boldsymbol{D}\rangle = \langle\boldsymbol{D}\rangle^T\langle\boldsymbol{D}\rangle + M\langle\boldsymbol{A}\rangle, \tag{33}$$

$$\langle\alpha_{nl}\rangle = \tilde{a}/\tilde{b}_{nl}, \tag{34}$$

$$\langle\gamma\rangle = \tilde{c}/\tilde{d}. \tag{35}$$

where in (30), $\boldsymbol{\mu}_l^x[n]$ denotes the nth entry of $\boldsymbol{\mu}_l^x$, $\boldsymbol{\Sigma}_l^x[n, n]$ represents the nth diagonal element of $\boldsymbol{\Sigma}_l^x$, and (32) follows from (22). The estimate of the dictionary $\boldsymbol{D}$ can be chosen as the expectation of its posterior distribution, i.e. $\langle\boldsymbol{D}\rangle$. For clarity, we summarize our algorithm as Algorithm 1.

Algorithm 1. Sparse Bayesian dictionary learning – a variational Bayesian algorithm.

```
Input: $\boldsymbol{Y}$, $\beta$ and $N$
Output: $\boldsymbol{D}$, $\boldsymbol{X}$, $\boldsymbol{\alpha}$ and $\gamma$
1: Initialize $\langle\boldsymbol{D}\rangle$ with random matrix, $\langle\boldsymbol{A}\rangle$ with identity matrix,
   $\langle\boldsymbol{\alpha}\rangle$ with positive matrix and $\gamma$ with positive scalar.
2: while the convergence criterion (60) is not reached do
3:   for $l=1$ to $L$ do
4:     Calculate $q_x(\boldsymbol{x}_l)$ using (18) with $q_d(\boldsymbol{D})$, $q_\alpha(\boldsymbol{\alpha})$ and $q_\gamma(\gamma)$
       fixed.
5:   end for
6:   Update $q_d(\boldsymbol{D})$ using (22) with $q_x(\boldsymbol{X})$, $q_\alpha(\boldsymbol{\alpha})$, and $q_\gamma(\gamma)$ fixed.
```

```
7:      Update $q_\alpha(\boldsymbol{\alpha})$ using (24) with $q_x(\boldsymbol{X})$, $q_d(\boldsymbol{D})$ and $q_\gamma(\gamma)$ fixed.
8:      Update $q_\gamma(\gamma)$ using (27) with $q_x(\boldsymbol{X})$, $q_d(\boldsymbol{D})$ and $q_\alpha(\boldsymbol{\alpha})$ fixed.
9:   end while
10:  set the mean of $q_d(\boldsymbol{D})$, $q_x(\boldsymbol{X})$, $q_\alpha(\boldsymbol{\alpha})$ and $q_\gamma(\gamma)$ as the estimate
     of $\boldsymbol{D}$, $\boldsymbol{X}$, $\boldsymbol{\alpha}$ and $\gamma$, respectively.
```

Remark 1. We discuss the choice of the parameter β which defines the variance of the dictionary atoms. We might like to set β equal to $1/m$ such that the norm of each atom has unit variance. Our experiment results, however, suggest that a very large value of β, e.g. 10^8, leads to better performance. In fact, choosing an infinitely large β implies placing strictly non-informative priors over the atoms $\{\boldsymbol{d}_n\}$, in which case the update of the dictionary is simplified as

$$\langle \boldsymbol{D}\rangle = \boldsymbol{BA} = \boldsymbol{Y}\langle \boldsymbol{X}^T\rangle\langle \boldsymbol{XX}^T\rangle^{-1}. \tag{36}$$

This update formula is similar to the formula used for dictionary update in the MOD method, except that the point estimates $\boldsymbol{X}$ and $\boldsymbol{XX}^T$ are now replaced by the posterior mean $\langle \boldsymbol{X}\rangle$ and $\langle \boldsymbol{XX}^T\rangle$, respectively.

In the above algorithm, atoms are updated in a parallel way. By assuming posterior independence among atoms $\{\boldsymbol{d}_n\}$, our method can also be readily adapted to provide sequential update of the atoms, i.e. it updates one atom at a time while fixing the rest atoms in the dictionary. The mean field approximation, in this case, can be expressed as

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \approx q(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{D}, \gamma) = q_x(\boldsymbol{x})q_\alpha(\boldsymbol{\alpha})\prod_{n=1}^{N} q_{d_n}(\boldsymbol{d}_n)q_\gamma(\gamma). \tag{37}$$

The posterior distribution $q_{d_n}(\boldsymbol{d}_n)$ can then be computed by maximizing $L(q)$ while keeping the distributions of other hidden variables fixed, which leads to

$$\begin{aligned}
\ln q_{d_n}(\boldsymbol{d}_n) &\propto \langle \ln p(\boldsymbol{Y}, \boldsymbol{X}, \{\boldsymbol{d}_k\}, \boldsymbol{\alpha}, \gamma)\rangle_{q_x(\boldsymbol{X})\prod_{k\neq n}^{N} q_{d_k}(\boldsymbol{d}_k)q_\alpha(\boldsymbol{\alpha})q_\gamma(\gamma)} \\
&\propto \langle \ln p(\boldsymbol{Y}|\boldsymbol{X}, \{\boldsymbol{d}_k\}, \gamma)p(\boldsymbol{d}_n)\rangle_{q_x(\boldsymbol{X})\prod_{k\neq n}^{N} q_{d_k}(\boldsymbol{d}_k)q_\gamma(\gamma)} \\
&\overset{(a)}{\propto} \langle \ln p(\boldsymbol{Y}^{-n}|\boldsymbol{d}_n, \boldsymbol{x}_{n\cdot}, \gamma)p(\boldsymbol{d}_n)\rangle_{q_x(\boldsymbol{X})\prod_{k\neq n}^{N} q_{d_k}(\boldsymbol{d}_k)q_\gamma(\gamma)} \\
&\overset{(b)}{\propto} \frac{1}{2}\langle \gamma \operatorname{tr}\{(\boldsymbol{Y}^{-n} - \boldsymbol{d}_n\boldsymbol{x}_{n\cdot})(\boldsymbol{Y}^{-n} - \boldsymbol{d}_n\boldsymbol{x}_{n\cdot})^T\} + \beta^{-1}\boldsymbol{d}_n^T\boldsymbol{d}_n\rangle \\
&= \frac{1}{2}\left[\boldsymbol{d}_n^T(\langle\gamma\rangle\langle \boldsymbol{x}_{n\cdot}\boldsymbol{x}_{n\cdot}^T\rangle + \beta^{-1})^{-1}\boldsymbol{d}_n - 2\boldsymbol{d}_n\langle \boldsymbol{Y}^{-n}\rangle\langle \boldsymbol{x}_{n\cdot}^T\rangle\right],
\end{aligned} \tag{38}$$

where in (a), we define

$$\boldsymbol{Y}^{-n} \triangleq \boldsymbol{Y} - \boldsymbol{D}^{-n}\boldsymbol{X}, \tag{39}$$

in which $\boldsymbol{D}^{-n}$ is generated by $\boldsymbol{D}$ with the nth column of $\boldsymbol{D}$ replaced by a zero vector, and $\boldsymbol{x}_{n\cdot}$ denotes the nth row of $\boldsymbol{X}$, (b) comes from the fact that $\boldsymbol{Y}^{-n} - \boldsymbol{d}_n\boldsymbol{x}_{n\cdot} = \boldsymbol{W}$ and thus we have

$$p(\boldsymbol{Y}^{-n}|\boldsymbol{d}_n, \boldsymbol{x}_{n\cdot}, \gamma) = \frac{\gamma^{\frac{ML}{2}}}{2\pi}\exp\left(-\frac{1}{2}\gamma \,\|\, \boldsymbol{Y}^{-n} - \boldsymbol{d}_n\boldsymbol{x}_{n\cdot}\,\|_F^2\right). \tag{40}$$

From (38), it can be seen that $\boldsymbol{d}_n$ follows a Gaussian distribution

$$q_{d_n}(\boldsymbol{d}_n) = \mathcal{N}(\boldsymbol{d}_n|\boldsymbol{\mu}_n^d, \boldsymbol{\Sigma}_n^d), \tag{41}$$

with the mean and the covariance matrix given, respectively, by

$$\begin{aligned}
\boldsymbol{\mu}_n^d &= \boldsymbol{\Sigma}_n^d\langle \boldsymbol{Y}^{-n}\rangle\langle \boldsymbol{x}_{n\cdot}^T\rangle, \\
\boldsymbol{\Sigma}_n^d &= (\langle\gamma\rangle\langle \boldsymbol{x}_{n}\boldsymbol{x}_{n\cdot}^T\rangle + \beta^{-1})^{-1}\boldsymbol{I},
\end{aligned} \tag{42}$$

where $\langle \boldsymbol{x}_{n}\boldsymbol{x}_{n\cdot}^T\rangle$ is the nth diagonal element of $\langle \boldsymbol{XX}^T\rangle$, and $\langle \boldsymbol{Y}^{-n}\rangle = \boldsymbol{Y} - \langle \boldsymbol{D}^{-n}\rangle\langle \boldsymbol{X}\rangle$. Our proposed algorithm therefore can be readily extended to a columnwise update procedure by replacing the update of $q_d(\boldsymbol{D})$ with the sequential update of $q_{d_n}(\boldsymbol{d}_n)$, $\forall\, n$. The results of the sequential update of $q_{d_n}(\boldsymbol{d}_n)$, $\forall\, n$ are not presented in our paper.

4. Gibbs sampler

Gibbs sampling is an alternative to the variational Bayes method for Bayesian inference. In particular, different from the variational Bayes which provides a locally optimal, exact analytical solution to an approximation of the posterior, Monte Carlo techniques such as Gibbs sampling provide a numerical approximation to the exact posterior of hidden variables using a set of samples. It has been observed that the Gibbs sampler may provide better performance than the variational Bayesian inference in some cases [30].

Let $\boldsymbol{\theta} \triangleq \{\boldsymbol{X}, \boldsymbol{\alpha}, \boldsymbol{D}, \gamma\}$ denote all hidden variables in our hierarchical model. We aim to find the posterior distribution of $\boldsymbol{\theta}$ given the observed data $\boldsymbol{Y}$:

$$p(\boldsymbol{\theta}|\boldsymbol{Y}) \propto p(\boldsymbol{Y}|\boldsymbol{D}, \boldsymbol{X}, \gamma)p(\boldsymbol{D})p(\boldsymbol{X}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})p(\gamma). \tag{43}$$

To provide an approximation to the posterior distribution of the hidden variables, the Gibbs sampler generates an instance from the distribution of each hidden variable in turn, conditional on the current values of the other hidden variables. It can be shown (see, for example, [31]) that the sequence of samples constitutes a Markov chain, and the stationary distribution of that Markov chain is just the sought-after joint distribution. Specifically, the sequential sampling procedure of the Gibbs sampler is given as follows:

- Sampling $\boldsymbol{X}$ according to its conditional marginal distribution $p(\boldsymbol{X}|\boldsymbol{Y}, \boldsymbol{D}^{(t)}, \boldsymbol{\alpha}^{(t)}, \gamma^{(t)})$.
- Sampling $\boldsymbol{D}$ according to its conditional marginal distribution $p(\boldsymbol{D}|\boldsymbol{Y}, \boldsymbol{X}^{(t+1)}, \boldsymbol{\alpha}^{(t)}, \gamma^{(t)})$.
- Sampling $\boldsymbol{\alpha}$ according to its conditional marginal distribution $p(\boldsymbol{\alpha}|\boldsymbol{Y}, \boldsymbol{D}^{(t+1)}, \boldsymbol{X}^{(t+1)}, \gamma^{(t)})$.
- Sampling γ according to its conditional marginal distribution $p(\gamma|\boldsymbol{Y}, \boldsymbol{D}^{(t+1)}, \boldsymbol{X}^{(t+1)}, \boldsymbol{\alpha}^{(t+1)})$.

Note that the above sampling scheme is also referred to as a blocked Gibbs sampler [22] because it groups two or more variables together and samples from their joint distribution conditioned on all other variables, rather than sampling from each one individually. Details of this sampling scheme are provided next. For simplicity, the notation $p(\boldsymbol{z}|-)$ is used in the following to denote the distribution of variable $\boldsymbol{z}$ conditioned on all other variables:

(1) *Sampling* $\boldsymbol{X}$: Samples of $\boldsymbol{X}$ can be obtained by independently sampling each column of $\boldsymbol{X}$, i.e. $\boldsymbol{x}_l$. The conditional marginal distribution of $\boldsymbol{x}_l$ is given as

$$p(\boldsymbol{x}_l|-) \propto p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{D}, \gamma)p(\boldsymbol{x}_l|\boldsymbol{\alpha}_l) \propto p(\boldsymbol{y}_l|\boldsymbol{D}, \boldsymbol{x}_l, \gamma)p(\boldsymbol{x}_l|\boldsymbol{\alpha}_l). \tag{44}$$

Recalling (17), it can be easily verified that $p(\boldsymbol{x}_l|-)$ follows a Gaussian distribution

$$p(\boldsymbol{x}_l|-) = \mathcal{N}(\boldsymbol{\mu}_l^x, \boldsymbol{\Sigma}_l^x) \tag{45}$$

with its mean $\boldsymbol{\mu}_l^x$ and covariance matrix $\boldsymbol{\Sigma}_l^x$ given by

$$\boldsymbol{\mu}_l^x = \gamma\boldsymbol{\Sigma}_l^x\boldsymbol{D}^T\boldsymbol{y}_l, \tag{46}$$

$$\boldsymbol{\Sigma}_l^x = (\gamma\boldsymbol{D}^T\boldsymbol{D} + \boldsymbol{\Lambda}_l)^{-1}, \tag{47}$$

where $\boldsymbol{\Lambda}_l \triangleq \text{diag}(\alpha_{1l}, \ldots, \alpha_{Nl})$.

(2) *Sampling* $\boldsymbol{D}$: There are two different ways to sample the dictionary: we can sample the whole set of atoms simultaneously, or sample the atoms in a successive way. We note that successive sampling way is essentially a Gauss–Seidel algorithm. Thus we can expect that sampling the atoms of the dictionary in a sequential manner converges fast than sampling the whole atoms simultaneously. Here, in order to expedite the convergence of the Gibbs sampler, we sample the atoms of the dictionary in a sequential manner. The conditional distribution of $\boldsymbol{d}_n$ can be written as

$$p(\boldsymbol{d}_n| -) \propto p(\boldsymbol{d}_n)p(\boldsymbol{Y}|\boldsymbol{D}, \boldsymbol{X}, \gamma) \propto p(\boldsymbol{d}_n)p(\boldsymbol{Y}^{-n}|\boldsymbol{d}_n, \boldsymbol{x}_{n\cdot}, \gamma), \tag{48}$$

where $\boldsymbol{Y}^{-n}$ is defined in (39). Recalling (40), we can show that the conditional distribution of $\boldsymbol{d}_n$ follows a Gaussian distribution

$$p(\boldsymbol{d}_n| -) = \mathcal{N}(\boldsymbol{\mu}_n^d, \boldsymbol{\Sigma}_n^d), \tag{49}$$

with its mean and covariance matrix given by

$$\boldsymbol{\mu}_n^d = \gamma \boldsymbol{\Sigma}_n^d \boldsymbol{Y}^{-n} \boldsymbol{x}_{n\cdot}^T, \tag{50}$$

$$\boldsymbol{\Sigma}_n^d = (\gamma \boldsymbol{x}_{n\cdot} \boldsymbol{x}_{n\cdot}^T + \beta^{-1})^{-1} \boldsymbol{I}. \tag{51}$$

(3) *Sampling* $\boldsymbol{\alpha}$: The log-conditional distribution of α_{nl} can be computed as

$$\ln p(\alpha_{nl}| -) \propto \ln p(\alpha_{nl}; a, b)p(x_{nl}|\alpha_{nl}) \propto \left(a - \frac{1}{2}\right)\ln \alpha_{nl} - \left(b + \frac{x_{nl}^2}{2}\right). \tag{52}$$

It is easy to verify that α_{nl} still follows a Gamma distribution

$$p(\alpha_{nl}| -) = \text{Gamma}(\hat{a}, \hat{b}_{nl}) \tag{53}$$

with the parameters $\hat{a}$ and $\hat{b}_{nl}$ given as

$$\hat{a} = a + \frac{1}{2}, \tag{54}$$

$$\hat{b}_{nl} = b + \frac{1}{2}x_{nl}^2. \tag{55}$$

(4) *Sampling* γ: The log-conditional distribution of γ is given by

$$\begin{aligned} \ln p(\gamma| -) &\propto \ln p(\boldsymbol{Y}|\boldsymbol{D}, \boldsymbol{X}, \gamma)p(\gamma) \propto \ln \prod_{l=1}^{L} p(\boldsymbol{y}_l|\boldsymbol{D}, \boldsymbol{x}_l, \gamma)p(\gamma) \\ &= \left(\frac{ML}{2} + c - 1\right)\ln \gamma - \left(\frac{1}{2} \| \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} \|_F^2 + d\right)\gamma, \end{aligned} \tag{56}$$

from which we can arrive at

$$p(\gamma| -) = \text{Gamma}(\hat{c}, \hat{d}), \tag{57}$$

where

$$\hat{c} = a + \frac{ML}{2}, \tag{58}$$

$$\hat{d} = d + \frac{1}{2} \| \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} \|_F^2. \tag{59}$$

So far we have derived the conditional marginal distributions for hidden variables $\{\boldsymbol{D}, \boldsymbol{X}, \boldsymbol{\alpha}, \gamma\}$. Gibbs sampler successively generates the samples of these variables according to their conditional distributions. After a burn-in period [32], the generated samples can be viewed as samples drawn from the posterior distribution $p(\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{\alpha}, \gamma|\boldsymbol{Y})$. With those samples, all the variables can be estimated by averaging the last few samples of the Gibbs sampler. For clarity, we now summarize the Gibbs sampling algorithm as Algorithm 2.

Algorithm 2. Sparse Bayesian dictionary learning – a Gibbs sampling algorithm

```
Input: $\boldsymbol{Y}$, $\beta$, $N$ and $T_{\text{burn-in}}$
Output: $\boldsymbol{D}$, $\boldsymbol{X}$, $\boldsymbol{\alpha}$ and $\gamma$
1:  Initialize $\boldsymbol{D}$ with random matrix, $\boldsymbol{\alpha}$ with positive matrix
    and $\beta$ with positive scalar.
2:  for $t=1$ to $t_{\max}$ do
3:    for $l=1$ to $L$ do
4:      Sample $\boldsymbol{x}_l$ using (45).
5:    end for
6:    for $n=1$ to $N$ do
7:      Sample $\boldsymbol{d}_n$ using (49).
8:    end for
9:    Sample $\boldsymbol{\alpha}$ using (53).
10:   Sample $\gamma$ using (57)
11:   if $t > T_{\text{burn-in}}$ then
12:     add $\boldsymbol{X}^{(t)}$ to the $\boldsymbol{X}$ samples set, $\boldsymbol{D}^{(t)}$ to the $\boldsymbol{D}$ samples set,
    $\boldsymbol{\alpha}^{(t)}$ to the $\boldsymbol{\alpha}$ samples set, $\gamma^{(t)}$ to the $\gamma$ samples set.
13:   end if
14: end for
15: Set $\boldsymbol{X}$, $\boldsymbol{D}$, $\boldsymbol{\alpha}$ and $\gamma$ to the average of the samples in the $\boldsymbol{X}$, $\boldsymbol{D}$,
    $\boldsymbol{\alpha}$ and $\gamma$ samples set, respectively.
```

5. Simulation results

We now carry out experiments to illustrate the performance of our proposed sparse Bayesian dictionary learning (SBDL) methods, which are, respectively, referred to as SBDL-VB and SBDL-Gibbs. Throughout our experiments, the parameters for our proposed method are set equal to $a=0.5$, $b = 10^{-6}$, $c=0.5$ and $d = 10^{-6}$. The parameter β is set to $\beta = 10^8$ for the SBDL-VB. Note that the SBDL-Gibbs is insensitive to the choice β and here we simply choose $\beta=1$. In the presence of noise, our proposed methods may yield an approximately sparse solution $\boldsymbol{X}$ which contains many small nonzero values. To obtain an exact sparse $\boldsymbol{X}$, the orthogonal matching pursuit (OMP) method can be used to perform the sparse coding after the dictionary is estimated via our proposed methods. The termination condition of OMP is set such that the residual is smaller than $C\sigma\sqrt{M}$, where σ is the noise standard deviation, M denotes the dimension of the training signal $\boldsymbol{y}_l$, and C is a factor. We set C to 1 for synthetic data and 1.15 for image denoising applications, as suggested by [5]. We compare our proposed methods with several existing state-of-the-art dictionary learning methods, namely, the K-SVD algorithm [4], the atom parallel-updating (APrU-DL) method [10], and the beta-Bernoulli process factor analysis (BPFA) method [11]. Both synthetic data and real data are used to test the performance of respective algorithms. For the SBDL-VB, we continue the iterative process until the difference between the estimated dictionaries of successive iterations is negligible, i.e.

$$\frac{\| \boldsymbol{D}^{(t+1)} - \boldsymbol{D}^{(t)} \|_F}{\| \boldsymbol{Y} \|_F} < 10^{-3}. \tag{60}$$

For the SBDL-Gibbs, the number of iterations is set to 300, i.e. $t_{\max} = 300$, and the estimate of the dictionary is simply chosen to be the last sample of the Gibbs sampler. Our experimental results shows that the number of $t_{\max} = 300$ iterations is sufficient for the Gibbs sampler to achieve a decent result. For a fair comparison, the competing algorithms including K-SVD, APrU-DL, and BPFA are executed with sufficient numbers of iterations to achieve their best performance.

5.1. Synthetic data

We generate a dictionary $\boldsymbol{D}$ of size 20×50, with each entry independently drawn from a normal distribution. Columns of $\boldsymbol{D}$ are then normalized to have unit norm. The training signals $\{\boldsymbol{y}_l\}_{l=1}^{L}$ are produced based on $\boldsymbol{D}$, where each signal $\boldsymbol{y}_l$ is a linear combination of K_l randomly selected atoms and the weighting coefficients are i.i.d. normal random variables. Two different cases are considered. First, all training samples are generated with the same number of atoms, i.e. $K_l = K, \forall\, l$, and K is assumed exactly known to the K-SVD method. The other case is that K_l varies from 3 to 6 for different l according to a uniform distribution. In this case, the K-SVD assumes that the sparsity level equals to 6 during the sparse coding stage. The observation noise is assumed multivariate Gaussian with zero mean and covariance matrix $\sigma^2\boldsymbol{I}$. Note that the APrU-DL (with FISTA) method requires to set two regularization parameters λ and λ_s to control the tradeoff between the sparsity and the data fitting error. The selection of these two parameters is always a tricky issue and an inappropriate choice may lead to considerable performance degradation. To show this, we use the following two different choices: $\{\lambda, \lambda_s\} = \{0.2, 0.15\}$ and $\{\lambda, \lambda_s\} = \{0.4, 0.4\}$, in which the former set of values is carefully selected by testing several pairs $\{\lambda, \lambda_s\}$ and choosing the best one, and the latter set of values slightly deviates from the former set of values. We use APrU-DL1 to denote the APrU-DL method which uses the former choice, and APrU-DL2 to denote the APrU-DL method which uses the latter one.

The recovery success rate is used to evaluate the dictionary learning performance. The success rate is computed as the ratio of the number of successfully recovered atoms to the total number of atoms. To calculate the success rate, the following steps are repeated N times: (1) For the ith atom $\boldsymbol{d}_i$, calculate the distance between this atom and each of the estimated atoms:

$$1 - \frac{|\boldsymbol{d}_i^T\hat{\boldsymbol{d}}_j|}{\|\boldsymbol{d}_i\|\|\hat{\boldsymbol{d}}_j\|}, \tag{61}$$

where $\hat{\boldsymbol{d}}_i$ denotes the jth estimated atom. (2) If the distance between this atom and one of the estimated atoms is less than a specified value, say, 0.1, then this atom is considered successfully recovered, and the associated estimated atom is excluded from the subsequent matching. Table 1 shows the average recovery success rates of respective algorithms, where we set L=1000 and L=2000, respectively, and the signal-to-noise ratio (SNR) varies from 10 to 30 dB. Results are averaged over 50 independent trials. From Table 1, we can see that:

- The proposed SBDL-Gibbs method achieves the highest recovery success rates in most cases. The proposed SBDL-VB method, although not as well as the SBDL-Gibbs, still provides quite competitive performance and presents a clear performance advantage over the K-SVD and APrU-DL methods when the number of training signals is limited, e.g. L=1000. In particular, both the SBDL-Gibbs and the SBDL-VB outperform the BPFA method by a big margin, despite all three methods were developed in a Bayesian framework.
- In the low SNR regime, e.g. SNR = 10 dB, the K-SVD method suffers from a significant performance loss when there is a discrepancy between the presumed sparsity level and the groundtruth (see the case where K_l varies but the presumed sparsity level is fixed to 6).
- The APrU-DL method is sensitive to the choice of the regularization parameters. It provides superior recovery performance when the regularization parameters are properly selected. Nevertheless, as we can see from Table 1, the APrU-DL method incurs a considerable performance degradation when the parameters deviate from their optimal choice, and there is no general guideline suggesting how to choose appropriate values for these regularization parameters.

Table 1
Recovery success rates.

L	SNR	Algorithm	K=3	K=4	K=5	Var. K
1000	10	K-SVD	80.52	36.36	2.52	0.80
		BPFA	76.76	57.12	22.56	43.32
		APrU-DL1	85.64	**64.40**	**33.44**	**53.68**
		APrU-DL2	48.20	17.48	4.68	12.52
		SBDL-VB	86.00	63.84	16.28	47.48
		SBDL-Gibbs	**91.52**	62.48	6.32	41.80
	20	K-SVD	93.20	93.44	92.08	84.68
		BPFA	87.96	92.00	94.08	93.58
		APrU-DL1	94.04	93.32	87.76	93.48
		APrU-DL2	72.48	40.32	14.15	33.04
		SBDL-VB	97.28	95.96	92.32	94.48
		SBDL-Gibbs	**99.64**	**99.16**	**97.52**	**99.12**
	30	K-SVD	94.24	94.32	93.92	86.64
		BPFA	87.04	91.20	94.56	92.60
		APrU-DL1	94.24	94.92	88.16	93.96
		APrU-DL2	73.40	43.16	17.16	34.36
		SBDL-VB	96.60	96.16	92.32	95.48
		SBDL-Gibbs	**99.60**	**99.16**	**98.64**	**99.00**
2000	10	K-SVD	91.00	88.88	50.56	25.32
		BPFA	91.16	92.28	86.44	90.84
		APrU-DL1	97.00	94.88	**86.24**	**95.44**
		APrU-DL2	84.84	68.36	42.28	64.04
		SBDL-VB	92.92	81.80	55.68	77.16
		SBDL-Gibbs	**98.56**	**95.72**	80.20	93.88
	20	K-SVD	95.64	96.68	95.16	94.00
		BPFA	89.68	91.76	94.72	93.04
		APrU-DL1	95.40	96.48	95.80	96.56
		APrU-DL2	85.32	82.44	64.48	79.84
		SBDL-VB	97.64	96.56	92.12	95.04
		SBDL-Gibbs	**99.48**	**99.56**	**98.92**	**99.16**
	30	K-SVD	95.88	96.92	96.96	93.36
		BPFA	87.24	91.80	95.92	93.94
		APrU-DL1	94.28	95.00	96.80	95.64
		APrU-DL2	86.32	82.40	66.08	80.52
		SBDL-VB	96.88	96.96	92.96	94.96
		SBDL-Gibbs	**99.40**	**99.16**	**99.52**	**99.32**

To better illustrate the performance of those methods, we also evaluate them with the following three metrics, namely, the achievable sparsity level (ASL), the root mean square error (RMSE) and the average run time. The ASL is defined as

$$\text{ASL} = \frac{\text{nnz}(\hat{\boldsymbol{X}})}{L}, \tag{62}$$

where $\text{nnz}(\hat{\boldsymbol{X}})$ denotes the number of the nonzero entries of $\hat{\boldsymbol{X}}$. The RMSE is defined as

$$\text{RMSE} = \frac{\|\tilde{\boldsymbol{Y}} - \hat{\boldsymbol{D}}\hat{\boldsymbol{X}}\|_F}{\|\tilde{\boldsymbol{Y}}\|_F}, \tag{63}$$

where $\tilde{\boldsymbol{Y}}$ denotes noise-free training samples and $\hat{\boldsymbol{D}}$ denotes the estimated dictionary. Table 3 shows the ASLs, RMSEs and average run times of respective algorithms, where we set L=1000 and K=4. Form Table 3, we see that the SBDL-Gibbs achieves best denoising performance. For the ASLs, we see that since the true sparsity level is known to the K-SVD method, it achieves the sparest representations for cases SNR = 10 dB and 20 dB. We also see that as the SNR increases, most methods are able to provide sparser representations and the proposed SBDL-Gibbs method even achieves a representation that is sparser than the groundtruth for SNR = 30 dB. In addition, we note that our proposed methods incur a higher computational complexity as compared

Table 2
PSNR.

r	σ	Algorithm	Boat	Cameraman	Couple
2	15	K-SVD	29.2802	31.4638	31.4068
		BPFA	29.5446	31.1759	31.2875
		APrU-DL	29.5718	**31.7662**	**31.5304**
		SBDL-VB	29.3557	31.1741	31.0691
		SBDL-Gibbs	**29.5881**	31.6978	31.4473
	25	K-SVD	26.9308	28.6211	28.6949
		BPFA	27.0726	28.4483	28.5825
		APrU-DL	26.8998	28.7069	28.5378
		SBDL-VB	26.6959	28.1587	28.4240
		SBDL-Gibbs	**27.1570**	**28.8380**	**28.8431**
	50	K-SVD	22.9499	23.9898	24.3532
		BPFA	23.4165	22.8861	24.5719
		APrU-DL	22.7274	23.5888	24.1901
		SBDL-VB	23.0861	23.3194	24.3299
		SBDL-Gibbs	**23.4651**	**24.1899**	**24.7870**
4	15	K-SVD	29.2585	31.3553	31.3513
		BPFA	29.4459	31.1063	31.1639
		APrU-DL	29.4554	**31.5541**	31.4276
		SBDL-VB	29.3217	31.0739	31.1359
		SBDL-Gibbs	**29.5376**	31.4931	**31.5443**
	25	K-SVD	26.6756	28.4350	28.5580
		BPFA	26.8726	28.3599	28.3287
		APrU-DL	26.7240	28.4447	28.4097
		SBDL-VB	26.5977	28.0960	28.3715
		SBDL-Gibbs	**27.0077**	**28.5539**	**28.7889**
	50	K-SVD	22.7708	23.2908	24.2388
		BPFA	23.1653	23.5942	24.2039
		APrU-DL	22.6036	23.3086	24.1107
		SBDL-VB	23.0404	23.3315	24.4163
		SBDL-Gibbs	**23.2525**	**23.8610**	**24.6326**

Table 3
ASL, RMSE and average run time.

SNR (dB)	Algorithm	ASL	RMSE	Run time
10	K-SVD	**4**	0.3064	**0.9778**
	BPFA	9.4243	0.3092	27.3396
	APrU-DL1	9.2787	**0.3086**	19.1625
	APrU-DL2	15.5645	0.3113	12.1774
	SBDL-VB	9.1543	0.3090	91.6423
	SBDL-Gibbs	9.3625	**0.3086**	154.9286
20	K-SVD	**4**	0.1344	**0.9563**
	BPFA	5.1772	0.0862	29.3492
	APrU-DL1	5.2479	0.0975	18.4018
	APrU-DL2	5.7878	0.1158	11.6770
	SBDL-VB	4.6019	0.0792	55.6192
	SBDL-Gibbs	4.3463	**0.0766**	154.6508
30	K-SVD	4	0.1127	**0.9757**
	BPFA	5.7770	0.0683	33.6997
	APrU-DL1	4.3184	0.0772	18.4779
	APrU-DL2	4.5205	0.0904	11.7725
	SBDL-VB	4.0543	0.577	58.5767
	SBDL-Gibbs	**3.8007**	**0.0536**	155.2510

with other methods, which is a major drawback of our proposed algorithms. This is because our proposed algorithms require conducting a matrix inverse operation when updating $\boldsymbol{X}$. The computational complexity required for this matrix inversion is of order $O(N^3)$. To reduce the computational complexity of our methods, the approximate message passing (AMP) or generalized approximate message passing (GAMP) technique [33,34] may be employed to circumvent the matrix inverse operation. This is an important topic worthy of our future investigation. We also note that the lower computational complexity of the APrU-DL method makes it possible to select proper regularization parameters by trying different sets of parameters.

Table 4
SSIM/ASL.

r	σ	Algorithm	Boat	Cameraman	Couple
2	15	K-SVD	0.8233/4.06	0.8959/2.50	0.8517/1.93
		BPFA	0.8345/4.72	0.8911/4.12	0.8528/3.55
		APrU-DL	0.8267/3.68	0.8975/2.30	0.8529/1.98
		SBDL-VB	0.8300/4.89	0.8922/3.53	0.8457/2.77
		SBDL-Gibbs	**0.8377**/4.01	**0.8990**/2.45	**0.8579**/2.18
	25	K-SVD	0.7313/1.81	0.8332/1.16	0.7710/0.94
		BPFA	0.7497/2.23	0.8295/1.90	0.7735/1.74
		APrU-DL	0.7299/1.83	0.8312/1.21	0.7640/1.04
		SBDL-VB	0.7357/2.44	0.8244/1.84	0.7641/1.30
		SBDL-Gibbs	**0.7533**/1.80	**0.8374**/1.18	**0.7813**/1.01
	50	K-SVD	0.5540/0.39	0.7110/0.30	0.5918/0.23
		BPFA	0.5817/0.48	0.7067/0.42	0.6062/0.34
		APrU-DL	0.5438/0.44	0.7010/0.36	0.5841/0.24
		SBDL-VB	0.5649/0.50	0.6928/0.48	0.5906/0.28
		SBDL-Gibbs	**0.5845**/0.40	**0.7167**/0.32	**0.6170**/0.23

5.2. Application to image denoising

We now demonstrate the results by applying the above methods to image denoising. Suppose images are corrupted by white Gaussian noise with zero mean and variance σ^2. We partition a noise-corrupted image into a number of overlapping patches (of size 8×8 pixels) obtained with one pixel shifting. Note that in our simulations, not all patches are selected for training, but only those patches whose top-left pixels are located at $[r \times i, r \times j]$ for any $i, j = 0, \ldots, \lfloor (Q-8)/r \rfloor$ are selected, where Q denotes the dimension of the $Q \times Q$ image, and r is chosen to be $r = \{2, 4\}$, respectively. The selected patches are then vectorized to generate the training signal $\{\boldsymbol{y}_l\}$. Also, in our experiments, we assume that the noise variance is perfectly known a priori by the K-SVD method. For the APrU-DL method, the regularization parameters λ and λ_s are carefully chosen to be λ=25 and $\lambda_s = 30$. After the training by respective algorithms, the trained dictionary is then used for denoising. The denoising process involves a sparse coding of all patches (including those used for training and those not) of size 8×8 pixels from the noisy image which is performed using OMP. The final estimate of each pixel is obtained by averaging the associated pixel from each of the denoised overlapping patches in which this pixel is included.

We consider three different images for image denoising, namely, “boat” (256×256), “cameraman” (256×256), and “couple” (512×512). Table 2 shows the peak signal to noise ratio (PSNR) results obtained for different images by respective algorithms, where the noise standard deviation is set to $\sigma = \{15, 25, 50\}$, respectively, and the dictionary to be inferred is assumed of size 64×256. The PSNR is defined as

$$\mathrm{PSNR} = 20\log_{10}\left(\frac{255 \times Q^2}{\| \hat{\boldsymbol{U}} - \boldsymbol{U} \|_F}\right),$$

where $\hat{\boldsymbol{U}}$ and $\boldsymbol{U}$ denote the denoised image and the original noise-free image, respectively. From Table 2, we see that the results of all methods are very close to each other in general. The proposed SBDL-Gibbs achieves a slightly higher PSNR than other methods in most cases, particularly when fewer number of signals is used for training. This result again demonstrates the superiority of the proposed method. The denoising performance is also evaluated by two other metrics, namely, a structural similarity (SSIM) index [35] and the ASL. The results are shown in Table 4. From Table 4, we see that all the algorithms provide similar achievable sparsity levels and SSIM indexes. The proposed SBDL-Gibbs achieves a slightly higher SSIM than other methods in most cases. In Figs. 2–6, we plot the noise-corrupted images “cameraman” and “couple”, and images denoised by respective algorithms. To show the details of

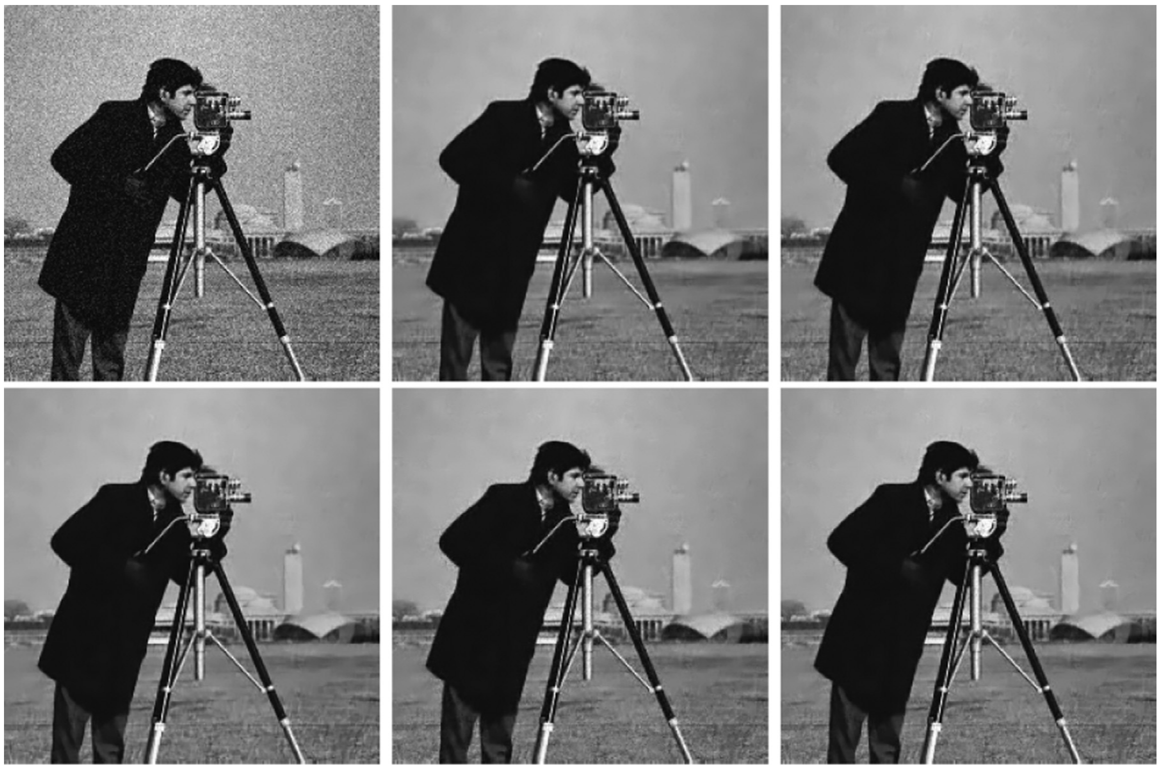

Fig. 2. Denoising results ($\sigma=15$ and $r=4$) for the image "cameraman". From left to right: the corrupted image, the denoised image by KSVD (PSNR=31.3553 dB, SSIM=0.8938), BPFA (PSNR=31.1063 dB, SSIM=0.8906), APrU-DL (PSNR=31.5541 dB, SSIM=0.8952), SBDL-VB (PSNR=31.0739 dB, SSIM=0.8913), and SBDL-Gibbs (PSNR=31.4931 dB, SSIM=0.8968).

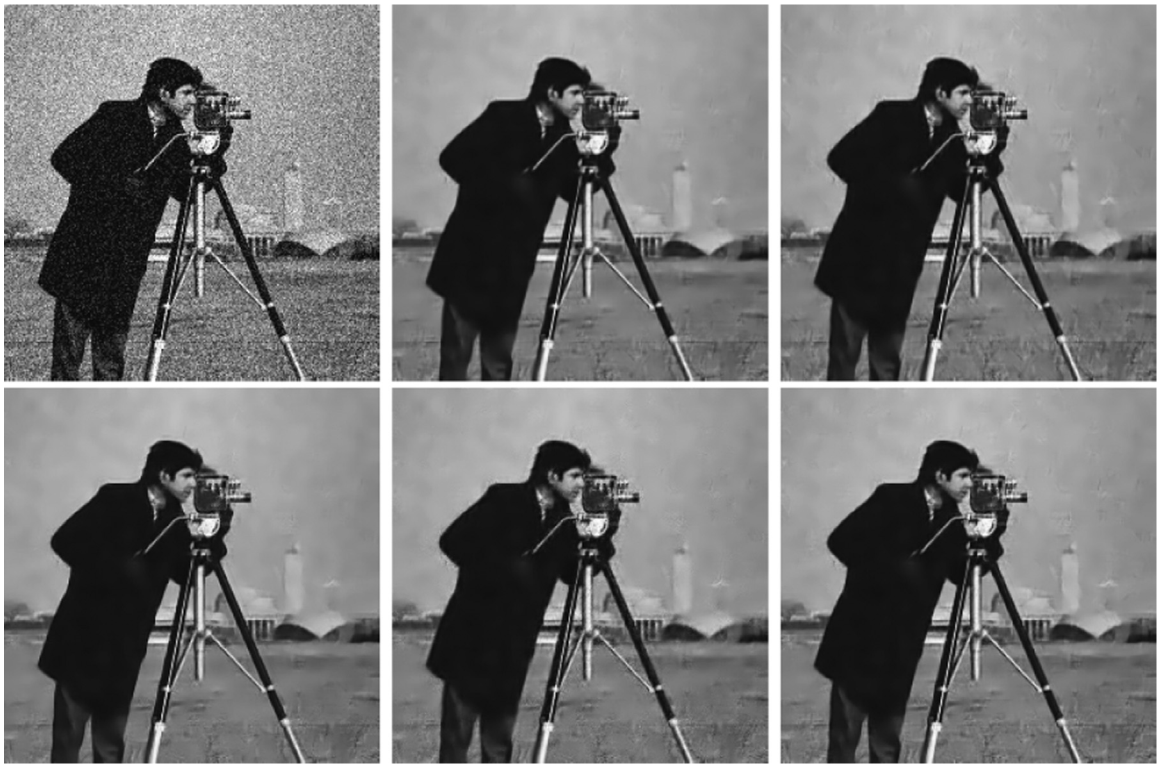

Fig. 3. Denoising results ($\sigma=25$ and $r=2$) for the image "cameraman". From left to right: the corrupted image, the denoised image by KSVD (PSNR=28.6211 dB, SSIM=0.8332), BPFA (PSNR=28.4483 dB, SSIM=0.8295), APrU-DL (PSNR=28.7069 dB, SSIM=0.8312), SBDL-VB (PSNR=28.1587 dB, SSIM=0.8244), and SBDL-Gibbs (PSNR=28.8380 dB, SSIM=0.8374).

the denoised images, a small area of the denoised image "couple" is zoomed in Fig. 7. The area that is magnified is of size 100×100 and located at the top left corner of door in the image. Fig. 8 plots the dictionaries estimated by respective methods from the image "couple", where $\sigma=50$ and $r=2$. We see that the APrU-DL and SBDL-VB methods provide noisy dictionaries as well as slightly

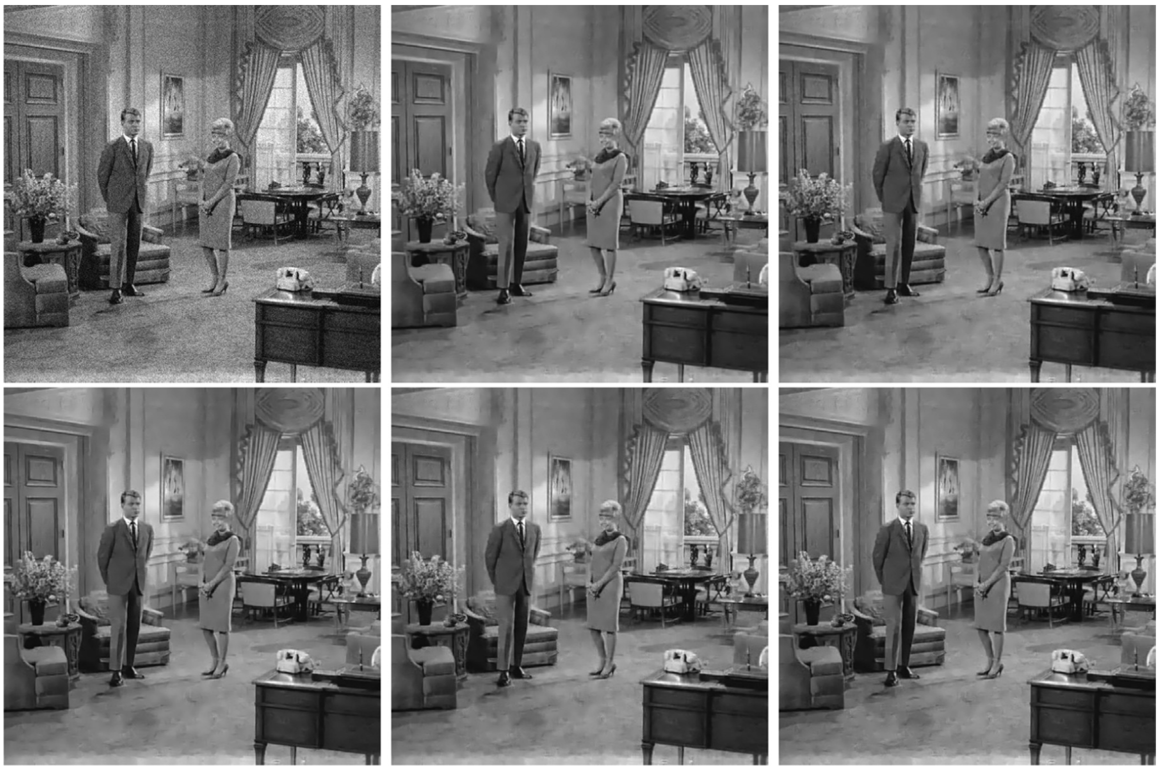

Fig. 4. Denoising results ($\sigma=15$ and $r=4$) for the image "couple". From left to right: the corrupted image, the denoised image by KSVD (PSNR=31.3513 dB, SSIM=0.8504), BPFA (PSNR=31.1639 dB, SSIM=0.8481), APrU-DL (PSNR=31.4276 dB, SSIM=0.8511), SBDL-VB (PSNR=31.1359 dB, SSIM=0.8482), and SBDL-Gibbs (PSNR=31.5443 dB, SSIM=0.8584).

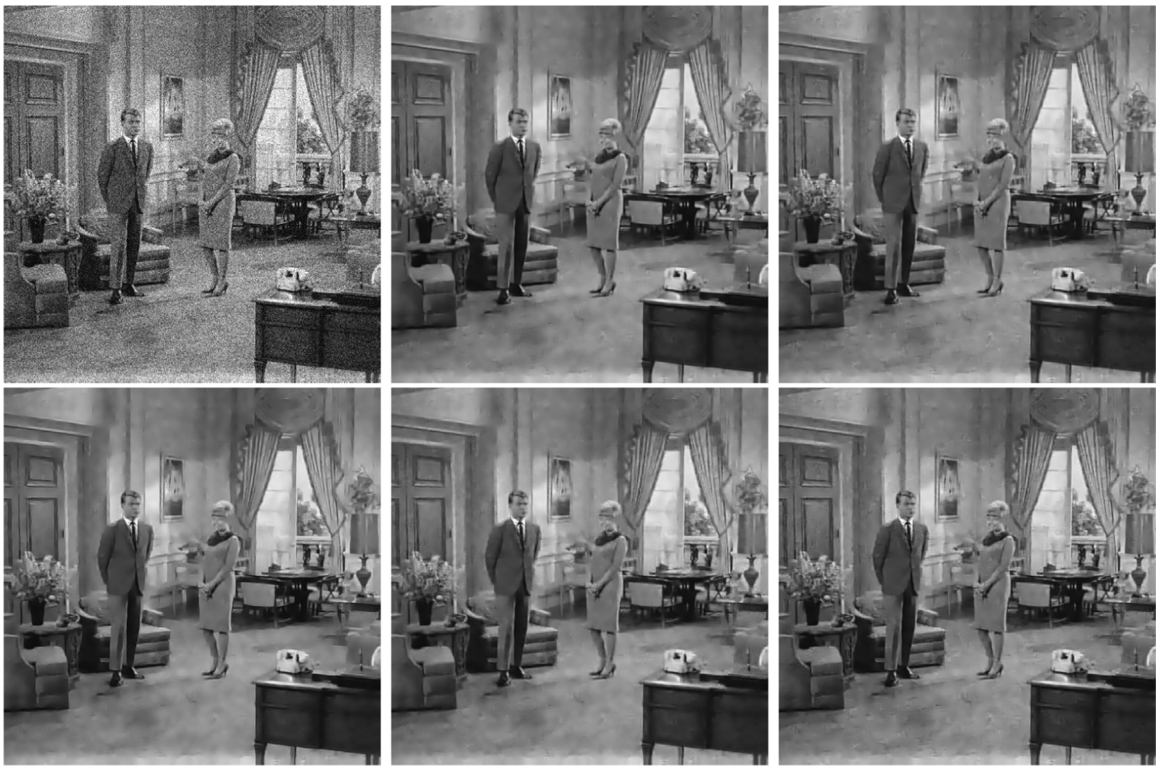

Fig. 5. Denoising results ($\sigma=25$ and $r=2$) for the image "couple". From left to right: the corrupted image, the denoised image by KSVD (PSNR=28.6949 dB,SSIM=0.7710), BPFA (PSNR=28.5825 dB, SSIM=0.7735), APrU-DL (PSNR=28.5378 dB, SSIM=0.7640), SBDL-VB (PSNR=28.4240 dB, SSIM=0.7641), and SBDL-Gibbs (PSNR=28.8431 dB, SSIM=0.7813).

dim denoised images compared with other methods.

We note that the proposed SBDL-Gibbs achieves better image denoising performance than the BPFA method. The BPFA is a state-of-the-art method which provides superior performance for image denoising, as illustrated by many experiments. Nevertheless, in [11], the BPFA uses all patches of a test image, based on which the

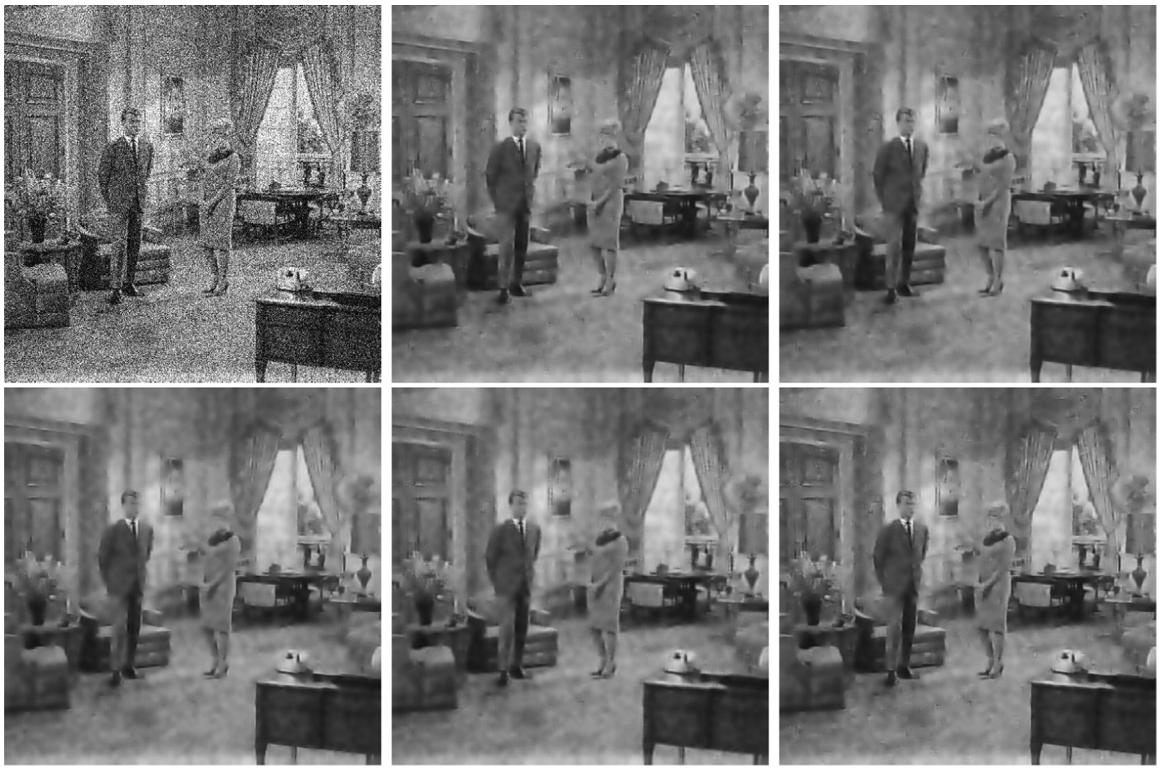

Fig. 6. Denoising results ($\sigma=50$ and $r=2$) for the image “couple”. From left to right: the corrupted image, the denoised image by KSVD (PSNR=24.3532 dB, SSIM=0.5918), BPFA (PSNR=24.5719 dB, SSIM=0.6062), APrU-DL (PSNR=24.1901 dB, SSIM=0.5841), SBDL-VB (PSNR=24.3299 dB, SSIM=0.5906), and SBDL-Gibbs (PSNR=24.7870 dB, SSIM=0.6170).

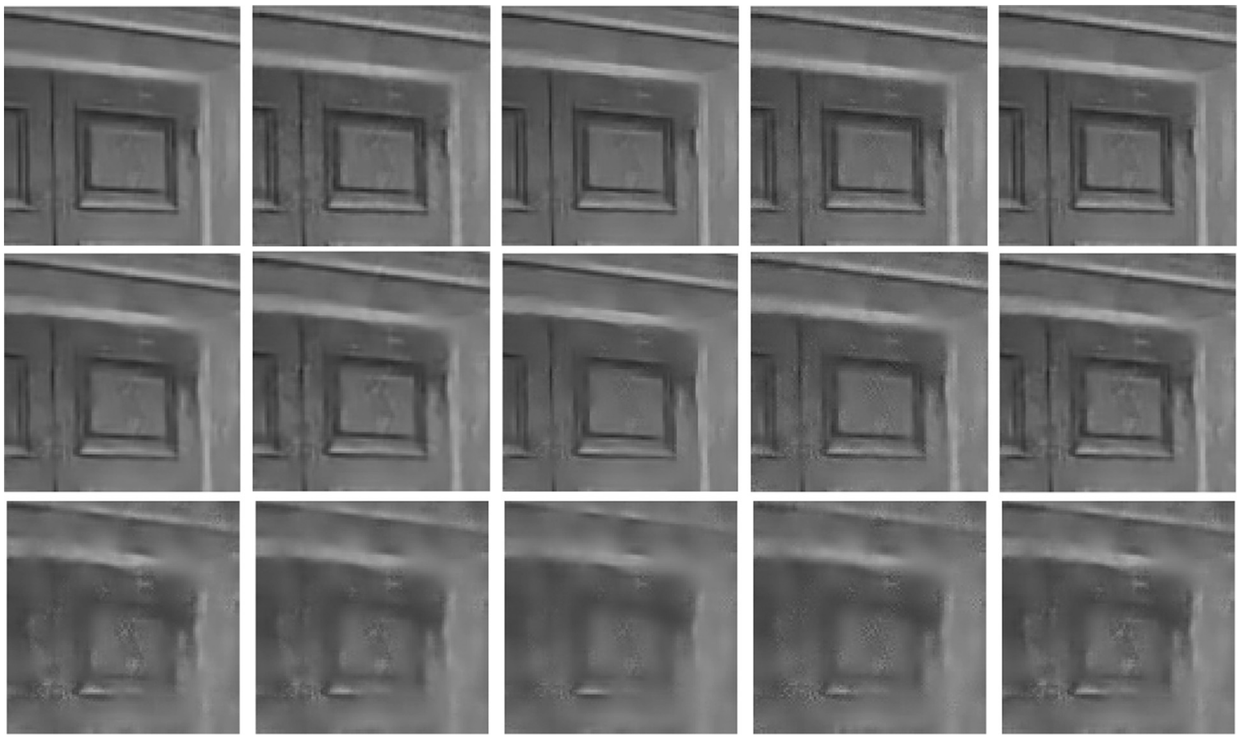

Fig. 7. Zoomed in part of the denoised images “couple”. From left to right: zoom in of the image denoised by KSVD, BPFA, APrU-DL, SBDL-VB, SBDL-Gibbs. Top row: $\sigma=15$ and $r=4$. Middle row: $\sigma=25$ and $r=2$. Bottom row: $\sigma=50$ and $r=2$.

clustering effects of adjacent patches are utilized to improve the performance. In our experiments, we, following [4], only use a subset of all patches, instead of all patches, for dictionary learning. Specifically, one-half (corresponding to $r=2$) and one-fourth (corresponding to $r=4$) of all patches are used. Since only a portion of patches are used for training, the benefit brought by the clustering effect may be limited, which possibly is the reason why the BPFA method is not as good as our proposed method.

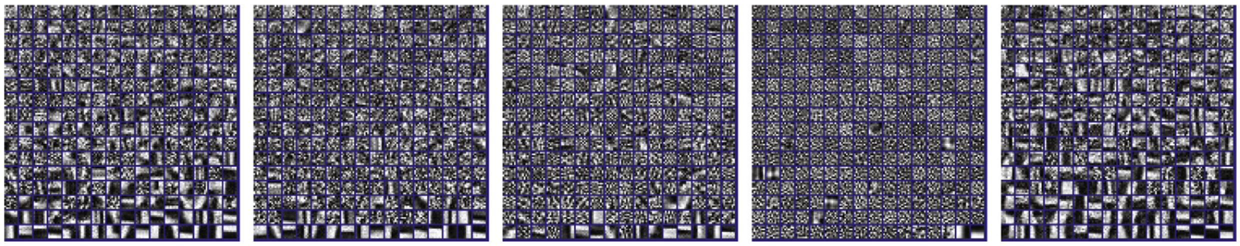

Fig. 8. Dictionaries estimated by respective methods ($\sigma=50$ and $r=2$) from the image "couple". From left to right: the dictionary trained by KSVD, BPFA, APrU-DL, SBDL-VB, and SBDL-Gibbs.

6. Conclusions

We developed a new Bayesian hierarchical model for learning overcomplete dictionaries based on a set of training data. This new framework extends the conventional sparse Bayesian learning framework to deal with the dictionary learning problem. Specifically, a Gaussian-inverse Gamma hierarchical prior is used to promote the sparsity of the representation. Suitable priors are also placed on the dictionary and the noise variance such that they can be inferred from the data. We developed a variational Bayesian method and a Gibbs sampler for Bayesian inference. Unlike some of previous methods, the proposed methods do not need to assume knowledge of the noise variance a priori, and can infer the noise variance automatically from the data. The performance of the proposed methods is evaluated using synthetic data. Numerical results show that the proposed methods are able to learn the dictionary with an accuracy notably better than the existing methods, particularly for the case where there is a limited number of training signals. The proposed methods are also applied to image denoising, where superior denoising results are achieved even compared with other state-of-the-art algorithms. Our proposed hierarchical model is also flexible to incorporate additional prior information to enhance the dictionary learning performance.

References

[1] E. Candés, T. Tao, Decoding by linear programming, IEEE Trans. Inf. Theory 12 (December) (2005) 4203–4215.
[2] J.M. Duarte-Carvajalino, G. Sapiro, Learning to sense sparse signals: simultaneous sensing matrix and sparsifying dictionary optimization, IEEE Trans. Image Process. 18 (July (7)) (2009) 1395–1408.
[3] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, Y. Ma, Robust face recognition vis sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (February (2)) (2009) 210–227.
[4] M. Aharon, M. Elad, A. Bruckstein, K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation, IEEE Trans. Signal Process. 54 (November (11)) (2006) 4311–4322.
[5] M. Elad, M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, IEEE Trans. Image Process. 15 (December (12)) (2006) 3736–3745.
[6] J. Mairal, F. Bach, J. Ponce, Task-driven dictionary learning, IEEE Trans. Pattern Anal. Mach. Intell. 34 (April (4)) (2012) 791–804.
[7] K. Engan, S.O. Aase, J.H. Hakon-Husoy, Method of optimal directions for frame design, in: IEEE International Conference on Acoustics, Speech and Signal Processing Phoenix, AZ, March 15–19, 1999.
[8] M. Yaghoobi, T. Blumensath, M.E. Davies, Dictionary learning for sparse approximations with the majorization method, IEEE Trans. Signal Process. 57 (June (6)) (2009) 2178–2191.
[9] W. Dai, T. Xu, W. Wang, Simultaneous codeword optimization (SimCO) for dictionary update and learning, IEEE Trans. Signal Process. 60 (December (12)) (2012) 6340–6353.
[10] M. Sadeghi, M. Babaie-Zadeh, C. Jutten, Learning overcomplete dictionaries based on atom-by-atom updating, IEEE Trans. Signal Process. 62 (February (4)) (2014) 883–891.
[11] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, L. Carin, Nonparametric Bayesian dictionary learning for analysis of noisy and incomplete images, IEEE Trans. Image Process. 21 (January (1)) (2012) 130–144.
[12] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online learning for matrix factorization and sparse coding, J. Mach. Learn. Res. 11 (2010) 19–60.
[13] K. Skretting, K. Engan, Recursive least squares dictionary learning algorithm, IEEE Trans. Signal Process. 58 (April (4)) (2010) 2121–2130.
[14] K. Labusch, E. Barth, T. Martinetz, Robust and fast learning of sparse codes with stochastic gradient descent, IEEE J. Sel. Top. Signal Process. 5 (5) (2011) 1048–1060.
[15] A. Koppel, G. Warnell, E. Stump, A. Ribeiro, D4l: Decentralized dynamic discriminative dictionary learning, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Hamburg, Germany, 2015, pp. 2966–2973.
[16] H. Raja, W.U. Bajwa, Cloud K-SVD: a collaborative dictionary learning algorithm for big, distributed data, IEEE Trans. Signal Process. 64 (January (1)) (2016) 173–188.
[17] M. Tipping, Sparse Bayesian learning and the relevance vector machine, J. Mach. Learn. Res. 1 (2001) 211–244.
[18] Z. Zhang, S. Wang, D. Liu, M.I. Jordan, EP-GIG priors and applications in Bayesian sparse learning, J. Mach. Learn. Res. 13 (1) (2012) 2031–2061.
[19] D.G. Tzikas, A.C. Likas, N.P. Galatsanos, The variational approximation for Bayesian inference, IEEE Signal Process. Mag. (November) (2008) 131–146.
[20] R.E. Turner, M. Sahani, Two problems with variational expectation maximisation for time-series models, in: Workshop on Inference and Estimation in Probabilistic Time-Series Models, vol. 2, no. 3, 2008.
[21] T.S. Jaakkola, Y. Qi, Parameter expanded variational Bayesian methods, in: Advances in Neural Information Processing Systems 2006, pp. 1097–1104.
[22] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2007.
[23] D.P. Wipf, B.D. Rao, An empirical Bayesian strategy for solving the simultaneous sparse approximation problem, IEEE Trans. Signal Process. 55 (July (7)) (2007) 3704–3716.
[24] S. Ji, Y. Xue, L. Carin, Bayesian compressive sensing, IEEE Trans. Signal Process. 56 (June (6)) (2008) 2346–2356.
[25] L. Wang, L. Zhao, G. Bi, C. Wan, Hierarchical sparse signal recovery by variational Bayesian inference, IEEE Signal Process. Lett. 21 (1) (2014) 110–113.
[26] L. Zhao, G. Bi, L. Wang, H. Zhang, An improved auto-calibration algorithm based on sparse Bayesian learning framework, IEEE Signal Process. Lett. 20 (9) (2013) 889–892.
[27] Y. Guan, J.G. Dy, Sparse probabilistic principal component analysis, in: International Conference on Artificial Intelligence and Statistics, 2009, pp. 185–192.
[28] J. Fang, Y. Shen, H. Li, P. Wang, Pattern-coupled sparse Bayesian learning for recovery of block-sparse signals, IEEE Trans. Signal Process. 63 (January (2)) (2015) 360–372.
[29] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. (1951) 79–86.
[30] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.
[31] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, D.B. Rubin, Bayesian Data Analysis, 3rd edition, Chapman and Hall/CRC, Boca Raton, FL, USA, 2013.
[32] B. Walsh, Markov Chain Monte Carlo and Gibbs Sampling, 2004.
[33] D.L. Donoho, A. Maleki, A. Montanari, Message passing algorithms for compressed sensing: I. Motivation and construction, in: Proceedings of Information Theory Workshop, Cairo, Egypt, January 2010.
[34] S. Rangan, Generalized approximate message passing for estimation with random linear mixing, in: Proceedings of IEEE International Symposium on Information Theory (ISIT), Saint Petersburg, Russia, August 2011.
[35] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.