Neural Networks xxx (xxxx) xxx

Contents lists available at ScienceDirect

Neural Networks

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

Bayesian deep matrix factorization network for multiple images denoising

Shuang Xu *, Chunxia Zhang, Jiangshe Zhang

School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, Shaanxi Province, 710049, China

ARTICLE INFO

Article history:
Received 15 May 2019
Received in revised form 2 November 2019
Accepted 22 December 2019
Available online xxxx

Keywords:
Matrix factorization
Bayesian neural networks
Variational Bayes

ABSTRACT

This paper aims at proposing a robust and fast low rank matrix factorization model for multiple images denoising. To this end, a novel model, Bayesian deep matrix factorization network (BDMF), is presented, where a deep neural network (DNN) is designed to model the low rank components and the model is optimized via stochastic gradient variational Bayes. By the virtue of deep learning and Bayesian modeling, BDMF makes significant improvement on synthetic experiments and real-world tasks (including shadow removal and hyperspectral image denoising), compared with existing state-of-the-art models.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

Low rank matrix factorization (LRMF) is a hot topic (Udell, Horn, Zadeh, & Boyd, 2016) in machine learning and it has been successfully applied to various tasks, such as structure-from-motion problems (Tomasi & Kanade, 1992), recommender systems (Bell, Koren, & Volinsky, 2009), image restoration (He, Zhang, Zhang, & Shen, 2016) and community detection (Yang & Leskovec, 2013), to name but a few. The basic idea of LRMF is to approximate an observed matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ by two low rank matrices. It can be formulated as the following issue, i.e.,

$$\min_{\boldsymbol{U}\in\mathbb{R}^{m\times k},\boldsymbol{V}\in\mathbb{R}^{n\times k}} d(\boldsymbol{X}, \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}) + \lambda p(\boldsymbol{U}, \boldsymbol{V}), \tag{1}$$

where $k \ll \min\{m, n\}$ denotes the rank of $\boldsymbol{U}$ and $\boldsymbol{V}$, $d(\cdot,\cdot)$ is a loss function, $p(\cdot,\cdot)$ is the penalty function with regard to two low rank matrices and $\lambda > 0$ is a penalty parameter to control the strength of penalization.

Generally speaking, the most popular loss $d(\cdot,\cdot)$ is L_2 norm, due to its good theoretical properties and ease of computation. When we are faced with real-world data, in practice, however, L_2 norm is so sensitive to outliers that it is far from being able to match up to our expectations. Thereafter, a growing number of researchers attempt to replace the L_2 norm loss function with a robust one. From the viewpoint of statistical modeling, L_2 norm loss corresponds to the situation in which the noise is distributed as a Gaussian distribution. To make the method robust to noise, we should assume that noise is sampled from a heavy-tailed distribution. For example, if we consider the following generative model, viz.,

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}} + \boldsymbol{E}, \tag{2}$$

and assume that the noise item $\boldsymbol{E}$ is governed by a Laplace distribution, the L_1 norm loss function can be derived (Eriksson & van den Hengel, 2010; Ke & Kanade, 2005).

Specifically, the noise in an image tends to be related with diverse factors and employing one heavy-tailed distribution is very likely to be inadequate. According to this fact, some researchers suggest utilizing the mixture of Gaussians (MoG) to model the noise in an image (Meng & De La Torre, 2013). This technique is able to automatically assign small weights to outliers. Moreover, some recent empirical studies demonstrate that the noise in a local region of the image is often correlated. The model will be further improved if some techniques such as Markov random field (Cao et al., 2015) or hierarchical Bayesian modeling (Chen, Cao, Zhao, Meng, & Xu, 2018) are employed to integrate this prior knowledge into the modeling process.

Although this pipeline has made great strides in many tasks, there are still some drawbacks of existing methods. First of all, most robust LRMF methods need to solve non-convex optimization problems, and current algorithms may make them get stuck into unexpected local optima. For example, the MoG based LRMF uses the expectation–maximization (EM) algorithm to infer parameters, which cannot guarantee the global optimum (Meng & De La Torre, 2013). Secondly, robust LRMF generally requires intensive computation and their running speed is extremely slow, especially for large-scale problems. Thirdly, it is found that some robust LRMF methods are so sensitive to hyper-parameters that they often perform badly if we do not carefully tune the hyper-parameters. Last but not least, the derivations of most robust

* Corresponding author.
E-mail address: shuangxu@stu.xjtu.edu.cn (S. Xu).

https://doi.org/10.1016/j.neunet.2019.12.023
0893-6080/© 2019 Elsevier Ltd. All rights reserved.

LRMF methods are very difficult and tedious. These shortcomings hinder us from rapidly exploring various applied problems.

Based on the above analysis, some novel techniques need to be developed in order to overcome the mentioned drawbacks or weaken their influence. Recently, deep learning based matrix factorization has emerged as a growing trend in research community of LRMF (Chatzis, 2017; Fan & Cheng, 2018; Trigeorgis, Bousmalis, Zafeiriou, & Schuller, 2017; Xue, Dai, Zhang, Huang, & Chen, 2017; Zhao, Ding, & Fu, 2017). The basic idea is to extend the shallow methods into multilayer models. Although they are still highly non-convex with regard to learnable parameters, the empirical studies show that deep LRMF methods perform better than traditional ones. Firstly, with the acceleration of graphical processing units, the training of deep LRMF methods is very fast. With the modern deep learning framework, deep LRMF methods are optimized by stochastic gradient descent and its variants, and to some degree the tedious derivations can be avoided. Secondly, it is shown that deep models outperform traditional ones in many tasks, including image clustering, matrix completion and so on. The reason mainly lies in that deep models can learn abstract features by means of nonlinear activation functions and hierarchical structure.

To the best of our knowledge, there is no deep learning based matrix factorization methods towards the multiple images denoising task. This paper develops a new LRMF model to rapidly explore the task and to weaken the influence of drawbacks of traditional methods. Our motivation is to replace the non-parametric solutions to $\boldsymbol{U}$ and $\boldsymbol{V}$ with two parametric estimators, so as to make the model more flexible. Hence, the new model employs two DNNs (rather than $\boldsymbol{U}$ and $\boldsymbol{V}$) to approximate the observed matrix. In formula, there is $\boldsymbol{X} \approx F(\boldsymbol{X}; \boldsymbol{w}^u)G^{\mathrm{T}}(\boldsymbol{X}; \boldsymbol{w}^v)$, where $\boldsymbol{w}^u$ and $\boldsymbol{w}^v$ denote the parameters of two DNNs. Essentially, we still consider a shallow LRMF. Although it cannot overcome all the drawbacks of traditional methods, it is found that this formulation significantly improves the performance. The experiments show that our model is less sensitive to hyperparameters, and that powerful low rank matrix generators enable our model to recover the clean images from the ones contaminated complicated noise. Additionally, to make the proposed model more robust to heavy noise, we exploit the Bayesian technique to infer unknown weights in the network.

In the rest of this paper, the related work is reviewed in Section 2 and preliminaries on variational Bayes are introduced in Section 3. Then, our model is presented in Section 4. In the next, experimental results are reported in Section 5. At last, we draw conclusions in Section 6.

2. Background and related work

In this section, we briefly introduce the task of multiple images denoising and then review related literatures on LRMF.

Multiple images denoising: It refers to image denoising with a collection of pictures $\{\boldsymbol{I}_i | i = 1, 2, \ldots, m\}$ taken for the same scene, where each of picture is corrupted less or more. Obviously, these images are located in a low-dimensional space. In other words, if the m images are reshaped into a matrix $\boldsymbol{X}$ whose ith row corresponds to pixels of the ith image, the rank of $\boldsymbol{X}$ is lower than m. Essentially, the multiple images denoising task can be typically solved by LRMF. According to (2), the denoising model can be interpreted as follows: there are k latent images $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$ and the observed images $\boldsymbol{X}$ are the linear mixture of latent ones, where the mixing coefficients are represented by $\boldsymbol{U}$. Owing to the low-rank property, the noise are removed by recovering $\boldsymbol{U}$ and $\boldsymbol{V}$.

L_1 norm based LRMF: Ke and Kanade (2005) developed an alternated linear/quadratic programming (ALP/AQP) method to solve L_1 norm based LRMF with missing entries. Eriksson and van den Hengel (2010) extended the Wiberg algorithm to this issue. Their method, Wiberg-L_1 is faster and more accurate than ALP/AQP. Then, Okutomi, Yan, Sugimoto, Liu, and Zheng (2012) proposed RegL1ALM which adds a nuclear-norm regularization and orthonormality constraint to catalyze convergence. In order to further improve running speed, inspired by coordinate decent algorithm, Meng, Xu, Zhang, and Zhao (2013) proposed cyclic weighted median (CWM) method which casts the original problem into a series of scalar optimization problems. Recently, Lin, Xu, and Zha (2018) developed LRMF-MM to solve the problem of L_1 loss plus the L_2-norm penalty placing on $\boldsymbol{U}$ and $\boldsymbol{V}$. Different from existing methods, LRMF-MM has theoretical guarantee on convergence.

Mixture distribution based LRMF: The main drawback of L_1 norm based LRMF is that the noise in real-world data is often not distributed as a Laplacian distribution. On account of MoG's universal approximation to any continuous distribution, Meng and De La Torre (2013) utilized MoG to model noise. Formally, it is assumed that $p(e) = \sum_{k=1}^{K} \pi_k \mathcal{N}(0, \sigma_k^2)$, where π_k denotes the proportion coefficient, and $\mathcal{N}(0, \sigma_k^2)$ is the Gaussian distribution with mean 0 and variance σ_k^2. In order to further improve robustness and adaptively fit intensely complicated noise, Cao et al. (2015) employed a mixture of exponential power (MoEP) distribution in place of MoG. The probability density function of the EP distribution is defined as

$$\mathrm{EP}(e|p, \sigma) = \frac{p}{2\sigma\Gamma(1/p)} \exp(-|\frac{e}{\sigma}|^p). \tag{3}$$

where p and σ control the shape and scale, respectively. It is obvious that the EP distribution coincides with Gaussian or Laplacian distribution if p is set to 2 or 1, respectively. Compared with L_1 or MoG based LRMF, this method is extremely flexible. In addition, to automatically select the number of mixture components (i.e., K), Cao et al. (2015) placed a sparse penalty on proportion coefficients $\pi_k (k = 1, 2, \ldots, K)$. Although MoEP based LRMF greatly improves the performance, it is very slow because it carries out cross-validation to determine a proper penalty strength. Furthermore, its performance highly depends on the shape parameters $p_k (k = 1, 2, \ldots, K)$. At last, MoG or MoEP based LRMF is often trapped in local minima since they are solved by the EM algorithm, which hinders them from achieving stable performance.

Robust PCA: Robust principle component analysis (PCA) considers a similar model, that is, $\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S} + \boldsymbol{E}$, where $\boldsymbol{L}, \boldsymbol{S}$ and $\boldsymbol{E}$ are low-rank matrix, sparse noise and Gaussian noise, respectively. The stable principle component pursuit (SPCP) proposed by Zhou, Li, Wright, Candès, and Ma (2010) casts this model into the following optimization problem,

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_{L_1} \quad \text{s.t.} \|\boldsymbol{X} - \boldsymbol{L} - \boldsymbol{S}\|_{L_2} < \epsilon, \tag{4}$$

where $\|\cdot\|_*$ denotes the nuclear norm. Recently, Guo, Liu, Xu, Xu, and Tao (2018) suggested to model the noise $\boldsymbol{E}$ by correntropy and proposed a more robust and faster algorithm called GoDec+.

Deep LRMF: Trigeorgis et al. (2017) designed a deep semi-NMF (DNMF) for image clustering. It is assumed that the data matrix $\boldsymbol{X}$ can be factorized into m layers. In the first $m-1$ layers, it aims to extract the low-dimensional representations $\boldsymbol{H}_i = \prod_{j=i+1}^{m} \boldsymbol{Z}_j \boldsymbol{H}_m, (i = 1, \ldots, m-1)$, where $\boldsymbol{H}_i$ is restricted to be non-negative for the purpose of a clustering interpretation. Then, in the last layer, the data is reconstructed by $\boldsymbol{X} = \boldsymbol{Z}_1 \boldsymbol{H}_1$. In conclusion, this model is expressed by $\boldsymbol{X} = \prod_{j=1}^{m} \boldsymbol{Z}_j \boldsymbol{H}_m$. At the same time, some activation functions can be inserted into this model to further improve the performance. Then, Zhao et al. (2017) generalized DNMF model for multi-view clustering by adding a Laplacian penalty. Fan and Cheng (2018) extended DNMF to cope

Please cite this article as: S. Xu, C. Zhang and J. Zhang, Bayesian deep matrix factorization network for multiple images denoising. Neural Networks (2020), https://doi.org/10.1016/j.neunet.2019.12.023.

with data completion tasks. Recently, some deep learning based matrix factorization methods are also designed for recommendation systems (Chatzis, 2017; Xue et al., 2017), which are closely related to the matrix completion problem.

3. Stochastic gradient variational Bayes

Because the concept of variational Bayes (VB) (Blei, Kucukelbir, & McAuliffe, 2017; Blundell, Cornebise, Kavukcuoglu, & Wierstra, 2015) is intensely used in the rest of this paper, VB is introduced in this section. For a Bayesian model, let $\boldsymbol{w}$ refer to all learnable parameters in this model, and a prior distribution $p(\boldsymbol{w})$ is imposed on it. Our aim is to obtain the posterior of $\boldsymbol{w}$ according to Bayes' rule $p(\boldsymbol{w} \mid \boldsymbol{X}) = p(\boldsymbol{X} \mid \boldsymbol{w})p(\boldsymbol{w})/p(\boldsymbol{X})$. On the plus side, Bayesian model is usually more accurate since the uncertainty of estimation of each parameter is measured by a distribution. On the minus side, unfortunately, $p(\boldsymbol{X})$ is a high-dimensional and computationally intensive integral. It is unable to acquire the analytical posterior distribution of $\boldsymbol{w}$. As a substitute, the posterior can be approximated by VB (Blei et al., 2017), the key idea of which is to minimize the KL-divergence between the variational distribution $q(\boldsymbol{w})$ and the posterior distribution $p(\boldsymbol{w} \mid \boldsymbol{X})$. After some derivations, it can be proved that this problem is equivalent to maximizing evidence lower bound (ELBO), i.e.,

$$\max_{q(\boldsymbol{w})} \left\{ \text{ELBO} = \mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})} \left[\log \frac{p(\boldsymbol{X} \mid \boldsymbol{w})p(\boldsymbol{w})}{q(\boldsymbol{w})} \right] \right\}, \tag{5}$$

where $q(\boldsymbol{w})$ denotes the variational distribution. For simple Bayesian models, there is an analytical solution for this optimization issue. Generally speaking, for complicated deep learning models, stochastic gradient variational Bayes (SGVB) technique is a better choice (Kingma & Welling, 2014). The key idea is to approximate the expectation by re-parameterization trick and Monte Carlo (MC) sampling. At first, with the re-parameterization trick, the randomness of $\boldsymbol{w}$ can be excluded by a differentiable function with regard to the auxiliary random variable $\boldsymbol{\epsilon}$, i.e.

$$\boldsymbol{w} = h(\boldsymbol{\epsilon}; \boldsymbol{\theta}), \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}), \tag{6}$$

where $\boldsymbol{\theta}$ is the parameter. Then, the objective function can be estimated by drawn samples of $\boldsymbol{\epsilon}$,

$$\text{ELBO} = \frac{1}{S}\sum_{s=1}^{S} \left[\log \frac{p(\boldsymbol{X} \mid h(\boldsymbol{\epsilon}^{(s)}; \boldsymbol{\theta}))p(h(\boldsymbol{\epsilon}^{(s)}; \boldsymbol{\theta}))}{q(h(\boldsymbol{\epsilon}^{(s)}; \boldsymbol{\theta}))} \right]. \tag{7}$$

In this fashion, the model can be optimized by stochastic gradient descent (SGD) and its generalizations.

4. Model formulation

In general, the traditional LRMF can be solved in the fashion of cyclic descent, namely, fixing one low rank component and updating the other one. In formula, there is

$$\boldsymbol{U}^t \leftarrow f(\boldsymbol{X}, \boldsymbol{V}^{t-1}) \tag{8}$$

and

$$\boldsymbol{V}^t \leftarrow g(\boldsymbol{X}, \boldsymbol{U}^t). \tag{9}$$

Here, t indexes the number of iterations. And the final estimate can be written as follows,

$$\boldsymbol{U} \leftarrow F(\boldsymbol{X}, \boldsymbol{V}^0) = f(\boldsymbol{X}, g(\boldsymbol{X}, \ldots, f(\boldsymbol{X}, \boldsymbol{V}^0))) \tag{10}$$

and

$$\boldsymbol{V} \leftarrow G(\boldsymbol{X}, \boldsymbol{V}^0) = g(\boldsymbol{X}, f(\boldsymbol{X}, \ldots, g(\boldsymbol{X}, f(\boldsymbol{X}, \boldsymbol{V}^0)))). \tag{11}$$

Since the update functions f and g are determined by loss and penalty, the performance of LRMF depends on how well the objective function is designed. Nevertheless, there are few adjustable parameters in f and g.

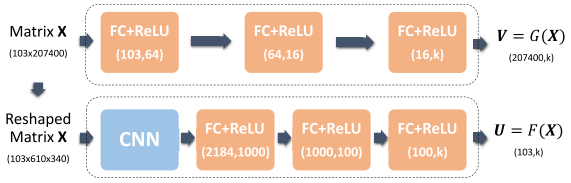

Fig. 1. The architecture of deep matrix factorization network.

Table 1
The CNN structure used in Fig. 1. Conv and AvgPool represent convolution and average pooling layers, respectively.

	Out channel	Kernel size	Stride	Padding
Conv	6	11	3	0
AvgPool	–	3	2	1
Conv	12	5	1	2
Conv	12	5	1	2
AvgPool	–	3	2	1
Conv	24	3	1	1
Conv	24	3	1	1
AvgPool	–	3	2	1
Conv	1	1	1	0
Conv	1	3	1	1
Conv	1	3	1	1
Conv	24	3	1	1
AvgPool	–	3	2	1

Our core idea is to employ two parametric estimators modeled by the non-linear function with hierarchical structure to approximate low rank components, instead of non-parametric solutions as shown in Eqs. (10) and (11). In formula, the deep matrix factorization (DMF) network can be written as,

$$\boldsymbol{X} = F(\boldsymbol{X}; \boldsymbol{w}^u)G^{\mathrm{T}}(\boldsymbol{X}; \boldsymbol{w}^v) + \boldsymbol{E}. \tag{12}$$

Here, the low rank matrix estimators F and G map the observed data from $\mathbb{R}^{m\times n}$ into $\mathbb{R}^{m\times k}$ and $\mathbb{R}^{n\times k}$, respectively, where $\boldsymbol{w}^u$ and $\boldsymbol{w}^v$ are learnable parameters. Hence, the estimation functions F and G in our model automatically vary on different datasets. In contrast, for traditional methods, F and G are fixed once the objective function is designed.

4.1. Network structure of DMF

DMF is designed for processing image data. For ease of illustration, a hyperspectral image (HSI) dataset called "Pavia University" (PU) is taken as an example. It contains 103 bands of HSI with size 610 × 340. Therefore, the size of the target matrix $\boldsymbol{X}$ is 103 × 207 400 and each row can be regarded as an image.

The overall structure of DMF network is given in Fig. 1. The framework consists of two parts, that is, F-branch and G-branch. They represent two low rank component generators, $F(\boldsymbol{X})$ and $G(\boldsymbol{X})$, respectively. In G-branch, it computes $\boldsymbol{V} \in \mathbb{R}^{207\,400\times k}$, whose columns can be regarded as k latent clean images. Therefore, in order to keep the structure of spatial space unchanged, three fully connected (FC) layers are used to integrate the information of each band (or say, image) and to gradually reduce rank. In the meantime, each FC layer is followed by a rectified linear unit (ReLU) for nonlinear activation. In F-branch, it computes mixing coefficients $\boldsymbol{U} \in \mathbb{R}^{103\times k}$. Here, the FC layer is not suggested due to the large number of parameters. Instead, we firstly reshape the input data $\boldsymbol{X}$ to make sure that it is with size 103 × 610 × 340 (namely, the number of bands × Height × Width), and then reshaped data passes through a CNN and 3 FC layers with ReLU. The CNN significantly reduces the number of parameters and

Please cite this article as: S. Xu, C. Zhang and J. Zhang, Bayesian deep matrix factorization network for multiple images denoising. Neural Networks (2020), https://doi.org/10.1016/j.neunet.2019.12.023.

makes a contraction in spatial space. Note that the configuration of CNN is displayed in Table 1. Finally, L_1 norm is employed as the loss function, viz. $\ell(\boldsymbol{w}) = \sum_{i=1}^{m}\sum_{j=1}^{n}|x_{ij} - \hat{x}_{ij}|$, where $\hat{x}_{ij}$ is the (i,j)th entry of $\hat{\boldsymbol{X}} = F(\boldsymbol{X})G(\boldsymbol{X})^{\mathrm{T}}$. This end-to-end model can be optimized by the current framework of deep learning. Our empirical studies show that SGD frequently raises the "nan" error. Thus, to learn the associated parameters, the adam algorithm is suggested in this paper.

4.2. Bayesian DMF

Now, we have built the architecture of DMF. Although DMF performs well in many cases, the empirical studies show that it may break down if data are corrupted by heavy noise. Many techniques, including variational dropout (Molchanov, Ashukha, & Vetrov, 2017) and Bayesian adversarial defense (Liu, Li, Wu, & Hsieh, 2018), have proved that the network performance can be improved if assuming parameters to be random variables. Therefore, in this subsection, to improve DMF's performance, we incorporate Bayesian inference into our model. For simplicity, it is hypothesized that the parameter w_j is governed by a Gaussian distribution, i.e.,

$$p(w_j) = \mathcal{N}(w_j \mid 0, s_0^2).$$

In addition, its variational distribution is assumed to be

$$q(w_j) = \mathcal{N}(w_j \mid \mu_j, \exp(2s_j)).$$

In Bayesian DMF (BDMF), the main purpose is to acquire w_j's variational mean and variance (viz. μ_j and $\exp(2s_j)$) rather than w_j itself. On account of the complexity of DNN, we have to employ SGVB to infer our model. According to Eq. (5), we ought to maximize ELBO with regard to μ_j and s_j. Therefore, Eq. (5) can be rewritten as

$$\min_{\boldsymbol{\mu},\boldsymbol{s}} \left\{\mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})}[-\log p(\boldsymbol{X} \mid \boldsymbol{w})] + \mathrm{KL}(q(\boldsymbol{w}) \parallel p(\boldsymbol{w}))\right\}. \tag{13}$$

Note that the second term is the Kullback–Leibler (KL) divergence between the variational and prior distributions, which plays the role of penalty. It has a closed-form expression, viz.

$$\begin{aligned}
&\mathrm{KL}(q(\boldsymbol{w}) \parallel p(\boldsymbol{w})) \equiv \mathrm{KL}(\boldsymbol{\mu}, \boldsymbol{s})\\
=&\mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})}[\log \frac{q(\boldsymbol{w})}{p(\boldsymbol{w})}]\\
=&\sum_j \mathbb{E}_{w_j\sim q(w_j)}[\log \frac{q(w_j)}{p(w_j)}]\\
=&\sum_j \mathbb{E}_{w_j\sim q(w_j)}[\log \frac{\mathcal{N}(w_j \mid \mu_j, \exp(2s_j))}{\mathcal{N}(w_j \mid 0, s_0^2)}]\\
=&\sum_j \mathbb{E}_{w_j\sim q(w_j)}[\log s_0 + \frac{w_j^2}{2s_0^2} - \log\exp(s_j) - \frac{(w_j-\mu_j)^2}{2\exp(2s_j)}]\\
=&\sum_j \log\frac{s_0}{\exp(s_j)} + \frac{\exp(2s_j)+\mu_j^2}{2s_0^2} - 0.5.
\end{aligned} \tag{14}$$

As for the first term in Eq. (13), we assume the noise is governed by a Laplace distribution. Hence, it is easy to see that $-\log p(\boldsymbol{X} \mid \boldsymbol{w})$ is the reconstruction loss between $\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$ measured by L_1 norm. However, it is hard to obtain the analytical expression of $\mathbb{E}_{\boldsymbol{w}\sim q(\boldsymbol{w})}[-\log p(\boldsymbol{X} \mid \boldsymbol{w})]$. For the sake of adaptation to current deep learning framework, we re-parameterize w_j as,

$$w_j = \mu_j + \exp(s_j)\epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0,1). \tag{15}$$

By the virtue of reparameterization, randomness is excluded from the parameter w_j. When implementing forward propagation, we sample ϵ and compute the weight $\boldsymbol{w}$ according to Eq. (15), then data passes through network. Now the objective function can be reformulated as

$$\ell^{\mathrm{Bayes}}(\boldsymbol{\mu}, \boldsymbol{s}) = \frac{1}{mn}\left[\sum_{i=1}^{m}\sum_{j=1}^{n}|x_{ij} - \hat{x}_{ij}| + \mathrm{KL}(\boldsymbol{\mu}, \boldsymbol{s})\right]. \tag{16}$$

For this objective function, the standard framework of gradient descent is available. The gradients of $\boldsymbol{\mu}$ and $\boldsymbol{s}$ are given by

$$\begin{aligned}
\frac{\partial \ell^{\mathrm{Bayes}}}{\partial \boldsymbol{\mu}} &= \frac{\partial \ell^{\mathrm{Bayes}}}{\partial \boldsymbol{w}} \odot \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{\mu}} + \frac{\partial \ell^{\mathrm{Bayes}}}{\partial \boldsymbol{\mu}}\\
&= \frac{\partial \ell^{\mathrm{Bayes}}}{\partial \boldsymbol{w}} + \frac{1}{mn}\frac{\boldsymbol{\mu}}{s_0^2}
\end{aligned} \tag{17}$$

and

$$\begin{aligned}
\frac{\partial \ell^{\mathrm{Bayes}}}{\partial \boldsymbol{s}} &= \frac{\partial \ell^{\mathrm{Bayes}}}{\partial \boldsymbol{w}} \odot \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{s}} + \frac{\partial \ell^{\mathrm{Bayes}}}{\partial \boldsymbol{s}}\\
&= \frac{\partial \ell^{\mathrm{Bayes}}}{\partial \boldsymbol{w}} \odot \exp(\boldsymbol{s}) \odot \boldsymbol{\epsilon} + \frac{1}{mn}(-1 + \frac{\exp(2\boldsymbol{s})}{s_0^2}).
\end{aligned} \tag{18}$$

Here, $\odot$ denotes the element-wise product. Algorithm 1 summarizes the main steps of training of our model. Typically, epoch T is set to 250 because it is found that for most cases the loss curve is very flat when $T = 250$. It is worth pointing out that $\boldsymbol{\mu}$ in Algorithm 1 is initialized by Xavier strategy (Glorot & Bengio, 2010). The initialization of s_0 and $\boldsymbol{s}$ is discussed in Section 5.1. After training, the posterior of reconstructed matrix, $F(\boldsymbol{X})G(\boldsymbol{X})^{\mathrm{T}}$, should be computed. Obviously, it is an impossible task. A heuristic strategy is to make a MC estimation, that is, to sample net parameters H times and then to average the result $F_h(\boldsymbol{X})$ as well as $G_h(\boldsymbol{X})$ over H implements. To facilitate the understanding, Algorithm 2 lists the steps of this strategy.

Algorithm 1 Training of Bayesian deep matrix factorization

Input: $\boldsymbol{X}$
Initialize s_0, $\boldsymbol{\mu}$, $\boldsymbol{s}$ and maximum iteration (epoch) T.
for $t = 1, 2, \cdots, T$ **do**
  Sample ϵ and compute $\boldsymbol{w} = \boldsymbol{\mu} + \exp(\boldsymbol{s}) \odot \boldsymbol{\epsilon}$.
  $\hat{\boldsymbol{X}} \leftarrow$ forward(net,$\boldsymbol{X}$).
  $\ell^{\mathrm{Bayes}} \leftarrow$ loss($\boldsymbol{X}, \hat{\boldsymbol{X}}$) + KL($\boldsymbol{\mu}, \boldsymbol{s}$).
  grad $\leftarrow$ backward(ℓ^{Bayes}).
  $\boldsymbol{\mu}, \boldsymbol{s} \leftarrow$ update(net, grad).
end for

Algorithm 2 Reconstruction of Bayesian deep matrix factorization

Input: $\boldsymbol{X}$, net, H
for $h = 1, 2, \cdots, H$ **do**
  Sample ϵ and compute $\boldsymbol{w} = \boldsymbol{\mu} + \exp(\boldsymbol{s}) \odot \boldsymbol{\epsilon}$.
  $F_h(\boldsymbol{X}), G_h(\boldsymbol{X}) \leftarrow$ forward(net,$\boldsymbol{X}$).
end for
$\tilde{F} = \frac{1}{H}\sum_{h=1}^{H} F_h(\boldsymbol{X})$, $\tilde{G} = \frac{1}{H}\sum_{h=1}^{H} G_h(\boldsymbol{X})$.
$\hat{\boldsymbol{X}} = \tilde{F}\tilde{G}^{\mathrm{T}}$.

5. Experiments

The performance of DMF as well as BDMF is empirically studied by a series of experiments. Six methods, including CWM (Meng et al., 2013), RegL1ALM (Okutomi et al., 2012), L1-MM (Lin et al., 2018), MoG (Meng & De La Torre, 2013), GoDec+ (Guo et al., 2018) and DNMF (Trigeorgis et al., 2017), are selected for comparison. In some experiments, we also report the performance of a baseline model, that is, the shallow LRMF supervised by L_1 norm

Please cite this article as: S. Xu, C. Zhang and J. Zhang, Bayesian deep matrix factorization network for multiple images denoising. Neural Networks (2020), https://doi.org/10.1016/j.neunet.2019.12.023.

and optimized by the adam algorithm with regard to $\boldsymbol{U}$ and $\boldsymbol{V}$. All experiments were conducted on a computer with Intel Core i7-8750 CPU @ 2.20 GHz and GTX 1060 GPU. DMF, BDMF, DNMF and the baseline model were implemented by Python, while others were implemented by Matlab 2018b. Two metrics, structural similarity index (SSIM) and root-mean-square error (RMSE), were employed to evaluate the performance of all algorithms. Note that SSIM is widely used to compare the similarity between two images according to the luminance, contrast and structure. In formula, given two image patches $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$, SSIM is defined by

$$\text{SSIM}(\boldsymbol{A}_i, \boldsymbol{B}_i) = \frac{(2\mu_{\boldsymbol{A}_i}\mu_{\boldsymbol{B}_i} + C_1)(2\sigma_{\boldsymbol{A}_i\boldsymbol{B}_i} + C_2)}{(\mu^2_{\boldsymbol{A}_i} + \mu^2_{\boldsymbol{B}_i} + C_2)(\sigma^2_{\boldsymbol{A}_i} + \sigma^2_{\boldsymbol{B}_i} + C_2)},$$

where $\mu_{\boldsymbol{A}_i}$, $\sigma^2_{\boldsymbol{A}_i}$ and $\sigma_{\boldsymbol{A}_i\boldsymbol{B}_i}$ denote the mean, variance and covariance, respectively. The C_1 and C_2 are small constants for computation stability. At last, the final SSIM value of two images is averaged over all patches. The higher SSIM is, the more similar two images are.

5.1. Synthetic experiments I

5.1.1. The hyper-parameter settings

At first, we study the performance of DMF and BDMF with different hyper-parameters. A built-in colored image of `skimage` package, chelsea, was used here. The original size of chelsea is 300×451. Firstly, we converted it into a gray-scale image of size 75×112 and normalized it to make sure its pixel value ranges from 0 to 1. Then, Gaussian noise $\mathcal{N}(0, \sigma^2)$ was added and this operation was implemented 30 times. Finally, we obtained 30 corrupted images, so the size of this synthetic data was 30×8400. Since the latent clean images are the same, the true rank of the observed data in this experiment is one. In the followings, $\sigma^2 = 0.1, 0.3, 0.5$. In order to avoid the influence of randomness, we implemented the experiments 100 times.

Fig. 2 demonstrates the performance of DMF for each σ^2 with different learning rates, ranging from 10^{-2} to 10^{-6}. It is found that the lowest RMSE values are reached when learning rates are around 10^{-3}, whereas the highest SSIM values are reached when learning rates are 10^{-2}. Intending to get more insights, Fig. 3 shows the loss curves with learning rates being 10^{-2} and 10^{-3}. Obviously, when the learning rate is 10^{-2}, the fluctuation of loss curve is higher. In conclusion, 10^{-3} is a better choice since it strikes the balance among RMSE, SSIM and loss.

Figs. 4 and 5 show the performance of BDMF with different s_0 and s_j. It is reported that the performance of BDMF is not sensitive to s_0 in terms of both RMSE and SSIM. In contrast, the initial variational standard deviation $\exp(s_j)$ strongly affects BDMF. From Eq. (15), it can be seen that larger s_j makes more noise embedded in network weight $\boldsymbol{w}$ and leads to more inaccurate estimation of gradient. Therefore, smaller s_j is suggested. According to these experiments, the parametric settings in next subsections are as follows: $s_0 = 0.08, s_j = \log(0.01)$, learning rate $= 10^{-3}$. Although there may be better configurations on specific datasets, the following experimental results indicate that there is no need to fine-tune these parameters, since our model with this configuration works very well on various datasets. In contrast, other models should carefully fine-tune their parameters on different datasets. In this sense, our model is less sensitive to parameters.

5.1.2. Comparison with baseline

In order to verify the effectiveness of our model, we compare it with the baseline model. An experiment was done on synthetic dataset, chelsea, with Gaussian noise and $\sigma^2 = 0.5$. DMF and the baseline model were optimized over 250 and 5000 epochs, respectively. As shown in Fig. 6, RMSE and SSIM values are plotted as a function of the number of epochs. It is found that our model rapidly achieves very low RMSE and high SSIM values, while the baseline model performs badly because of the slow convergence, and higher RMSE and lower SSIM values. To a certain degree, this result demonstrates the rationality of our idea (that is, employ two DNNs to approximate the low-rank components).

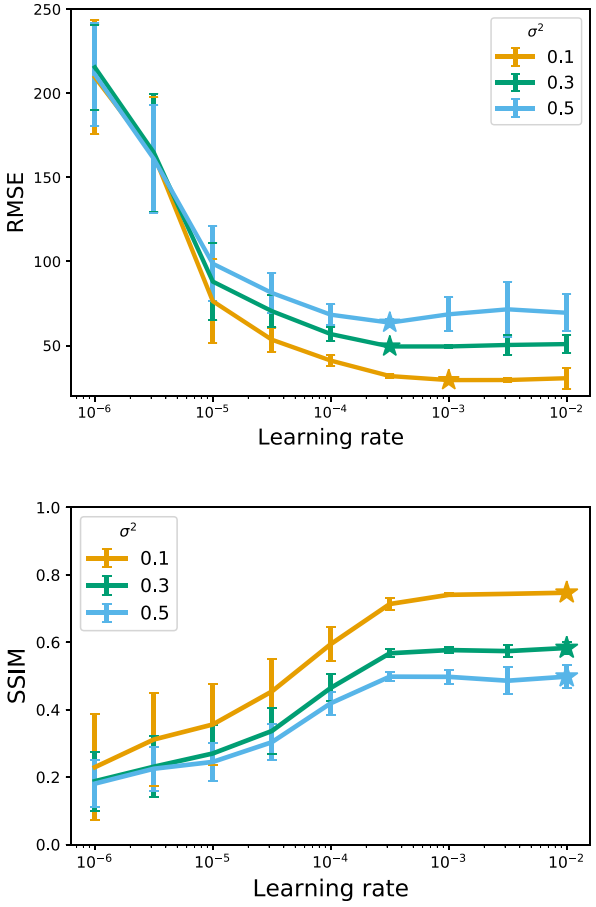

Fig. 2. The performance of DMF with different learning rates, where the best performance is marked with a star. The bar indicates the mean plus or minus one standard deviation.

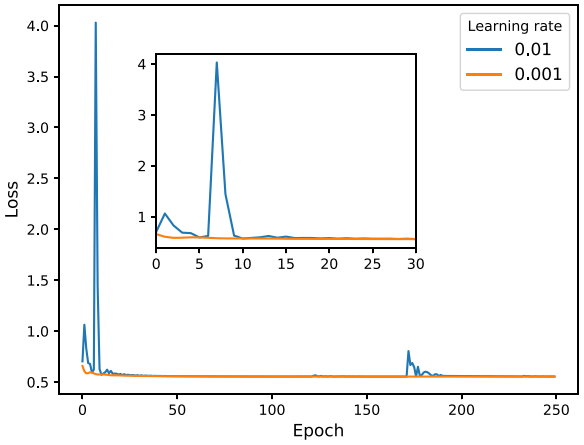

Fig. 3. The loss curve of DMF with different learning rates.

5.2. Synthetic experiments II

Here, we evaluate the performance on synthetic datasets of both small and large scales with various kinds of noise.

Please cite this article as: S. Xu, C. Zhang and J. Zhang, Bayesian deep matrix factorization network for multiple images denoising. Neural Networks (2020), https://doi.org/10.1016/j.neunet.2019.12.023.

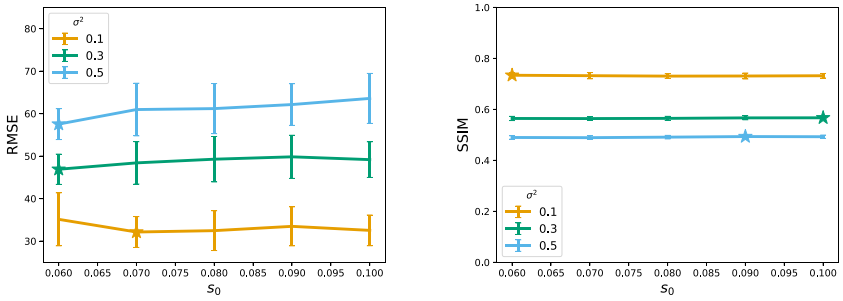

Fig. 4. The performance of BDMF with different s_0, where the best performance is marked with a star. The bar indicates the mean plus or minus one standard deviation.

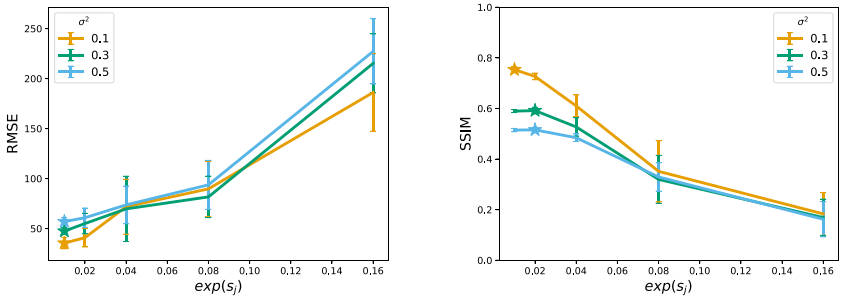

Fig. 5. The performance of BDMF with different s_j, where the best performance is marked with a star. The bar indicates the mean plus or minus one standard deviation.

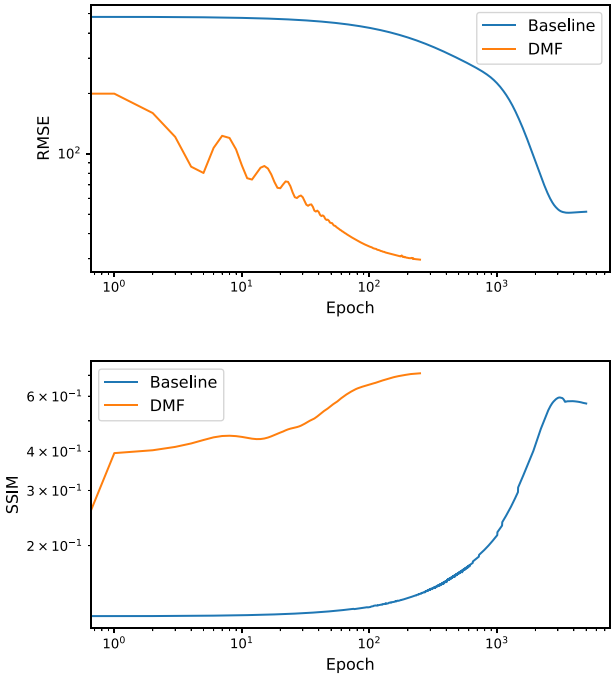

Fig. 6. The RMSE and SSIM curves.

5.2.1. Small scale dataset

The synthetic dataset chelsea described in previous subsection was used here. But four kinds of severe noise were considered, that is,

- Gaussian noise with standard deviation $\sigma = 0.5$;
- Laplace noise with scale parameter $b = 0.5$;
- Student's t noise with degree of freedoms $df = 5$;
- Mixture noise $0.5\mathcal{N}(0, 1) + 0.25\mathcal{N}(0, 2^2) + 0.25t(3)$.

Note the rank in DNMF was set to be 2, because its code requires $k \geq 2$. The results are reported in Table 2. Some conclusions can be drawn. (1) Both DMF and BDMF are very competitive with other methods. When data is corrupted by Gaussian noise, MoG and DNMF perform best, and DMF and BDMF take the second place. When data is corrupted the heavy-tailed noise, DMF and BDMF exhibit great advantage over others. (2) Although DNMF is a deep learning based method, it leads to good results in the cases of Gaussian and Mixture noise, while it performs even worse than traditional counterparts in other cases. It indicates that DNMF is instable for this task. (3) For traditional methods, it is found that, to some degree, they can remove noise, but the image structure may be ignored. For example, in the case of Gaussian noise, both traditional and deep learning based methods achieve similar RMSE values, but SSIM values of traditional methods are significantly lower than those of deep learning based ones.

5.2.2. Large scale dataset

In order to study the behavior on large scale dataset, a HSI dataset, PU, was employed in this part. The HSI can be viewed as a high-dimensional tensor, each pixel of which contains a continuous spectrum. In general, a very wide range of wavelengths is covered by HSIs and the spectrum is divided into many bands. For PU, there were 103 bands, and the image of each band was with size 610 × 340, so the final matrix was with size 103 × 207 400 (Band × Pixel). Since little noise corrupts PU, we thus refer the original dataset to ground truth. The same kinds of noise in the previous experiment were used to corrupt the ground truth. We did not implement L1-MM, because it was very slow on this dataset. Note that the same material has similar reflectance with similar wavelengths. Therefore, a HSI lies in a low-dimensional space. Here the rank was set to 4 for each algorithm as suggested by Cao et al. (2015).

Table 3 shows the results on the PU dataset. Not surprisingly, our methods are still the second to none and they reach lowest RMSE and highest SSIM values. However, with the mixture noise, DMF reaches relative higher SSIM value but worse RMSE value.

Please cite this article as: S. Xu, C. Zhang and J. Zhang, Bayesian deep matrix factorization network for multiple images denoising. Neural Networks (2020), https://doi.org/10.1016/j.neunet.2019.12.023.

Table 2
Result of synthetic experiments on the small scale dataset. The best and second best results are highlighted by bold and italic typeface, respectively.

	DMF	BDMF	CWM	RegL1ALM	L1-MM	MoG	GoDec+	DNMF
	Gaussian ($\sigma = 0.5$)							
RMSE	38.6867	42.9120	51.2163	49.1873	50.6483	**38.2023**	45.7578	*38.1953*
SSIM	**0.6636**	0.6151	0.4951	0.5084	0.4976	0.5946	0.5909	*0.6480*
	Laplace ($b = 0.5$)							
RMSE	57.3552	56.4178	55.1060	**53.5579**	53.9626	*53.8683*	237.0836	73.7276
SSIM	*0.5250*	**0.5262**	0.4575	0.4667	0.4641	0.4737	0.0540	0.4793
	Student's t ($df = 5$)							
RMSE	*99.1273*	**81.9081**	123.9953	121.5712	121.6975	108.2192	214.3883	128.4346
SSIM	**0.3593**	*0.3362*	0.1697	0.1719	0.1699	0.1992	0.0948	0.3049
	Mixture noise							
RMSE	*117.3937*	**101.5579**	158.6162	155.4506	157.2325	125.7987	209.2298	140.7827
SSIM	**0.3925**	*0.3782*	0.1519	0.1531	0.1530	0.1943	0.1176	0.3154

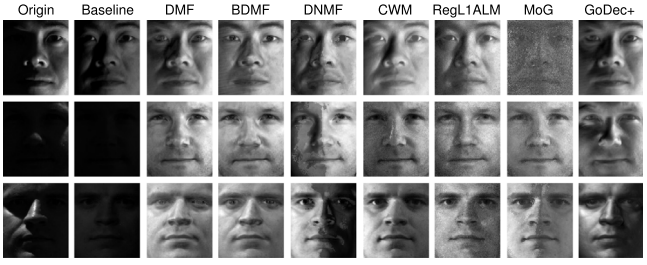

Fig. 7. Result on Yale Face dataset.

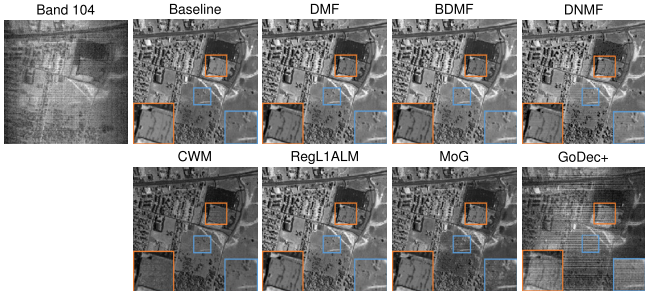

Fig. 8. Restoration result of band 104 in Urban dataset.

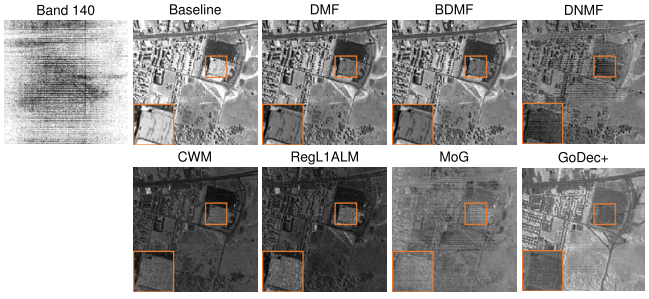

Fig. 9. Restoration result of band 140 in Urban dataset.

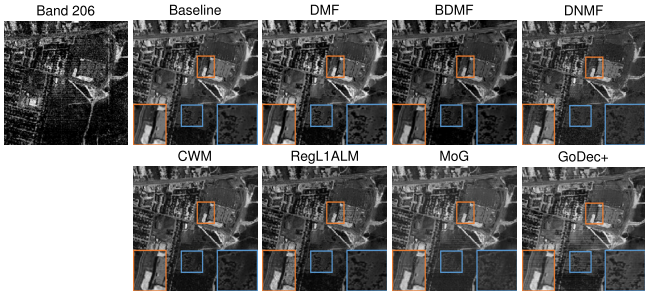

Fig. 10. Restoration result of band 206 in Urban dataset.

The reason may be that the mixture noise is intense. Without Bayesian hierarchical modeling on unknown parameters, DMF is not able to reconstruct the original matrix well. In addition, the behavior of DNMF is similar to traditional methods.

5.3. Shadow removal experiments

In this part, the Yale Face B Database (Georghiades, Belhumeur, & Kriegman, 2001) was employed to test the effectiveness of algorithms. There were 28 individuals, each of which had 64 images of size 192 × 168 under different illumination conditions. The purpose of this experiment is to remove the shadow in faces. For each individual, the size of target matrix is 64 × 32 256 and we set the rank to 4, as suggested by Meng and De La Torre (2013). We normalized data into [0,1]. For our networks, we initialized $s_j = \log(0.01)$, and set $s_0 = 0.08$ and the learning rate $= 10^{-3}$. The network structure can be seen in supplementary material. Three sets of typical images shown in Fig. 7 are randomly selected for comparison. It is easy to draw the conclusion that BDMF outperforms others, while DMF and GoDec+ can be ranked in the second and third places, respectively. Although CWM, RegL1ALM and MoG are able to remove the shadow of most area, they fail to provide good local details and the images generated by them contain visible noise. As for the baseline model, it totally fails to remove shadows.

5.4. HSI restoration experiments

Two HSI datasets, Urban and Terrain, were used in this experiment and there were 210 bands, each of which is of size 307 × 307 and 500 × 307, respectively. So, the target matrix is of size 210 × 94 249 and 210 × 153 500, respectively. We normalized data into [0, 1]. For our networks, we initialized $s_j = \log(0.01)$, and set $s_0 = 0.08$ and learning rate $= 10^{-3}$. The network structure can be seen in supplementary material. It is worthy to point out that dozens of bands were seriously polluted by the atmosphere and water absorption and can provide little useful information. In many papers, these seriously polluted bands are removed. But we remain them because it is a great challenge for algorithms to restore HSI with the existence of these polluted bands.

Figs. 8–12 display the reconstructed images of 3 bands in Urban and 2 bands in Terrain. For ease of comparison, some areas

Please cite this article as: S. Xu, C. Zhang and J. Zhang, Bayesian deep matrix factorization network for multiple images denoising. Neural Networks (2020), https://doi.org/10.1016/j.neunet.2019.12.023.

Table 3
Result of synthetic experiments on the large scale dataset. The best and second best results are highlighted by bold and italic typeface, respectively.

	DMF	BDMF	CWM	RegL1ALM	MoG	GoDec+	DNMF
	Gaussian ($\sigma = 0.5$)						
RMSE	*582.4957*	**499.1978**	838.1386	821.8644	645.4867	849.1218	636.6319
SSIM	*0.3540*	**0.4453**	0.1087	0.1360	0.1989	0.0998	0.2683
	Laplace ($b = 0.5$)						
RMSE	*859.3998*	**549.8610**	1095.0284	1110.4855	1098.0726	1505.7046	1019.4919
SSIM	*0.2191*	**0.2634**	0.0682	0.0867	0.1099	0.0121	0.2189
	Student's t ($df = 5$)						
RMSE	*916.4521*	**587.7235**	1237.3738	1248.5170	1047.9662	1405.4827	1094.9933
SSIM	*0.2390*	**0.2736**	0.0511	0.0594	0.0909	0.0157	0.1937
	Mixture noise						
RMSE	1667.9638	**645.2501**	1578.5254	1555.8727	*1280.9166*	1575.6985	1437.4592
SSIM	0.2296	*0.3094*	0.0301	0.0549	0.0929	0.0146	**0.3961**

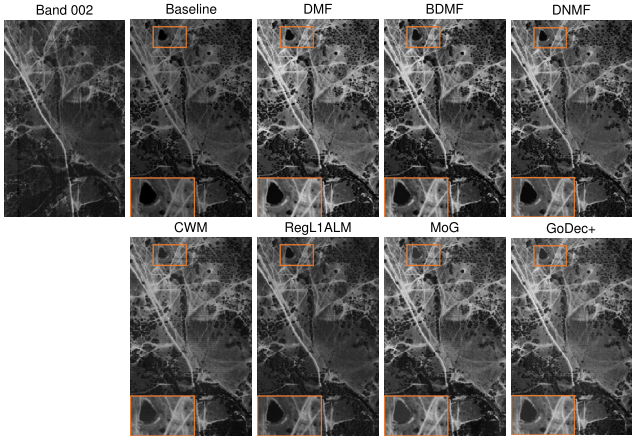

Fig. 11. Restoration result of band 002 in Terrain dataset.

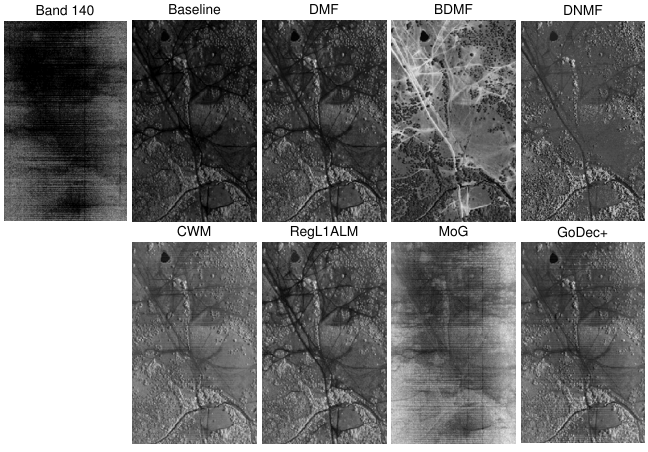

Fig. 12. Restoration result of band 140 in Terrain dataset.

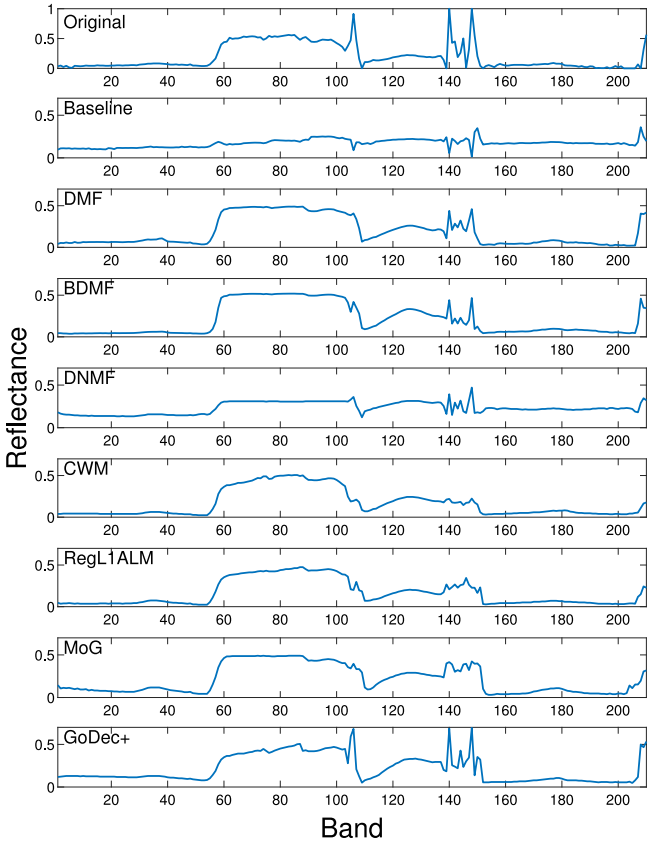

Fig. 13. Spectrum of pixel (281, 137) in Urban dataset.

are amplified. Compared with face modeling experiment, this HSI restoration task is far more difficult, since HSI is corrupted by various factors, including stripe, deadline, the atmosphere and water absorption. The baseline model, DMF and BDMF achieve the best reconstruction on both datasets. Not only are stripe and deadline noises removed, but also the spatial information is remained. It is found that the traditional methods usually fail to remove all of stripe or deadline noise. For example, as shown in Fig. 9, band 140 in Urban is intensely contaminated by stripe and Gaussian noise. DNMF, GoDec+ and MoG have few effects in stripe noise removal. CWM and RegL1ALM do better than the three methods, since they are able to eliminate most of stripe noise. Nonetheless, the amplified area marked by the orange box indicates that CWM and RegL1ALM fail to remove the Gaussian noise. Therefore, the results of CWM and RegL1ALM are not satisfactory. This may be caused by that fact that these stripes and deadlines exist at the same location in most of contaminated bands, which are treated as low rank clean images (rather than noise) by the traditional methods.

Except for the visual comparison, to obtain more insights on the algorithms' behavior, Fig. 13 shows the quantitative comparison, viz., the spectral signatures of pixel (281, 137) in Urban dataset. It is the curve of pixel value versus band index. It is clear that there is severe fluctuation in the original spectrum curve. All restoration algorithms can smooth the curve less or more. However, it is found that, compared with DMF and BDMF, other methods tend to break the spectrum pattern. For example, the most significant fluctuation exists from band 140 to 150. DMF

Please cite this article as: S. Xu, C. Zhang and J. Zhang, Bayesian deep matrix factorization network for multiple images denoising. Neural Networks (2020), https://doi.org/10.1016/j.neunet.2019.12.023.

Table 4
Running time (in seconds) on real datasets.

	DMF	BDMF	CWM	RegL1ALM	MoG	GoDec+	DNMF	Baseline
Yale	6.1447	13.0601	60.5882	14.1753	878.6743	0.0957	24.4324	16.1479
Urban	25.7118	26.8361	632.0379	130.6271	794.4574	1.0150	118.5582	49.2113
Terrain	37.5131	40.8246	1023.3879	210.1247	1473.3918	1.8561	199.8209	201.0589

and BDMF are able to simultaneously smooth it and remain the fluctuation pattern, while others cannot achieve this end. In other words, the traditional methods, baseline model and DNMF will result in spectral distortion, although they can remove part of noise in the image space. This phenomenon also indicates our methods are superior to traditional counterparts.

5.5. Comparison of execution time

In this subsection, we compare the execution time of each algorithm on three real datasets. DMF, BDMF and DNMF are trained with GPU, while others are optimized with CPU. It seems to be unfair for CPU based methods, but we still report the computation time as an auxiliary metric. As shown in Table 4, GoDec+ is fastest, and DMF as well as BDMF can be ranked in the second place. The baseline model and DNMF are slower than our methods, but are faster than CWM, RegL1ALM and MoG.

6. Conclusions

In this paper, we present DMF and BDMF trying to overcome the shortcomings of popular counterparts in image restoration task, including lack of robustness and sensitivity to hyper-parameters. DMF and BDMF enable us to rapidly explore related image processing problems. The synthetic, face modeling and HSI restoration experiments show that DMF and BDMF are superior to the state-of-art methods.

Acknowledgments

The authors would like to thank the editor, the associate editor and anonymous reviewers for their useful suggestions which greatly helped to improve the paper. The research of S. Xu is supported by the Fundamental Research Funds for the Central Universities [Grant Number xzy022019059]. The research of C.X. Zhang is supported by the National Key Research and Development Program of China [Grant Number 2018AAA0102201] and the National Natural Science Foundation of China [Grant Number 11671317]. The research of J.S. Zhang is supported by the National Key Research and Development Program of China [Grant Number 2018YFC0809001], and the National Natural Science Foundation of China [Grant Numbers 61976174].

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neunet.2019.12.023.

References

Bell, R., Koren, Y., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*, 30–37.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. In *ICML* (pp. 1613–1622).

Cao, X., Chen, Y., Zhao, Q., Meng, D., Wang, Y., Wang, D., et al. (2015). Low-rank matrix factorization under general mixture noise distributions. In *ICCV* (pp. 1493–1501).

Chatzis, S. P. (2017). Deep Bayesian matrix factorization. In *PAKDD* (pp. 453–464).

Chen, Y., Cao, X., Zhao, Q., Meng, D., & Xu, Z. (2018). Denoising hyperspectral image with non-i.i.d. noise structure. *IEEE Transactions on Cybernetics*, *48*(3), 1054–1066.

Eriksson, A., & van den Hengel, A. (2010). Efficient computation of robust low-rank matrix approximations in the presence of missing data using the L_1 norm. In *CVPR* (pp. 771–778).

Fan, J., & Cheng, J. (2018). Matrix completion by deep matrix factorization. *Neural Networks*, *98*, 34–41.

Georghiades, A., Belhumeur, P., & Kriegman, D. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 643–660.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *ICML* (pp. 249–256).

Guo, K., Liu, L., Xu, X., Xu, D., & Tao, D. (2018). GoDec+: Fast and robust low-rank matrix decomposition based on maximum correntropy. *IEEE Transactions on Neural Networks and Learning System*, *29*(6), 2323–2336.

He, W., Zhang, H., Zhang, L., & Shen, H. (2016). Total-variation-regularized low-rank matrix factorization for hyperspectral image restoration. *IEEE Transactions on Geoscience and Remote*, *54*(1), 178–188.

Ke, Q., & Kanade, T. (2005). Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *CVPR* (pp. 739–746).

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *International conference on learning representations, ICLR Banff, AB, Canada, April 14–16*.

Lin, Z., Xu, C., & Zha, H. (2018). Robust matrix factorization by majorization minimization. *IEEE TPAMI*, *40*(1), 208–220.

Liu, X., Li, Y., Wu, C., & Hsieh, C. (2018). Adv-BNN: Improved adversarial defense through robust Bayesian neural network. ArXiv abs/1810.01279.

Meng, D., & De La Torre, F. (2013). Robust matrix factorization with unknown noise. In *ICCV* (pp. 1337–1344).

Meng, D., Xu, Z., Zhang, L., & Zhao, J. (2013). A cyclic weighted median method for L_1 low-rank matrix factorization with missing entries. In *AAAI* (pp. 704–710).

Molchanov, D., Ashukha, A., & Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. In *ICML* (pp. 2498–2507).

Okutomi, M., Yan, S., Sugimoto, S., Liu, G., & Zheng, Y. (2012). Practical low-rank matrix approximation under robust L_1-norm. In *CVPR* (pp. 1410–1417).

Tomasi, C., & Kanade, T. (1992). Shape and motion from image streams under orthography: a factorization method. *IJCV*, *9*(2), 137–154.

Trigeorgis, G., Bousmalis, K., Zafeiriou, S., & Schuller, B. W. (2017). A deep matrix factorization method for learning attribute representations. *IEEE TPAMI*, *39*(3), 417–429.

Udell, M., Horn, C., Zadeh, R., & Boyd, S. (2016). Generalized low rank models. *Foundations and Trends in Machine Learning*, *9*(1), 1–118.

Xue, H.-J., Dai, X., Zhang, J., Huang, S., & Chen, J. (2017). Deep matrix factorization models for recommender systems. In *IJCAI* (pp. 3203–3209).

Yang, J., & Leskovec, J. (2013). Overlapping community detection at scale: A nonnegative matrix factorization approach. In *WSDM* (pp. 587–596).

Zhao, H., Ding, Z., & Fu, Y. (2017). Multi-view clustering via deep matrix factorization. In *AAAI* (pp. 2921–2927).

Zhou, Z., Li, X., Wright, J., Candès, E., & Ma, Y. (2010). Stable principal component pursuit. In *IEEE international symposium on information theory* (pp. 1518–1522).