Neurocomputing 266 (2017) 540–549

Contents lists available at ScienceDirect

Neurocomputing

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

# Deep learning based matrix completion

Jicong Fan*, Tommy Chow

*Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China*

ARTICLE INFO

*Article history:*
Received 9 January 2017
Revised 10 April 2017
Accepted 25 May 2017
Available online 2 June 2017

Communicated by Dr. Xin Luo

*Keywords:*
Matrix completion
AutoEncoder
deep learning
out-of-sample extension
image inpainting
collaborative filtering

ABSTRACT

Previous matrix completion methods are generally based on linear and shallow models where the given incomplete matrices are of low-rank and the data are assumed to be generated by linear latent variable models. In this paper, we first propose a novel method called AutoEncoder based matrix completion (AEMC). The main idea of AEMC is to utilize the partially observed data to learn and construct a nonlinear latent variable model in the form of AutoEncoder. The hidden layer of the AutoEncoder has much fewer units than the visible layers do. Meanwhile, the unknown entries of the data are recovered to fit the nonlinear latent variable model. Based on AEMC, we further propose a deep learning based matrix completion (DLMC) method. In DLMC, AEMC is used as a pre-training step for both the missing entries and network parameters; the hidden layer of AEMC is then used to learn stacked AutoEncoders (SAEs) with greedy layer-wise training; finally, fine-tuning is carried out on the deep network formed by AEMC and SAEs to obtain the missing entries of the data and the parameters of the network. In addition, we also provide out-of-sample extensions for AEMC and DLMC to recover online incomplete data. AEMC and DLMC are compared with state-of-the-art methods in the tasks of synthetic matrix completion, image inpainting, and collaborative filtering. The experimental results verify the effectiveness and superiority of the proposed methods.



## 1. Introduction

Matrix completion [1–5] is to recover or predict the missing or unknown entries of partially observed matrices. It has been widely applied to practical problems such as image inpainting [6,7], collaborative filtering [8–10], and classification [11]. Conventional matrix completion is also called low-rank matrix completion, in which the given incomplete matrix is assumed to be of low-rank. Low-rank matrix can be completed by matrix factorization [12–14], where the incomplete matrix is approximated with the multiplication of a thin matrix and a short matrix. In the matrix factorization based methods [15–18], the matrix rank should be given in advance. In [19], a method called low-rank matrix fitting algorithm (LMaFit) was proposed for factorization based matrix completion to dynamically and adaptively adjust the matrix rank. Matrix factorization based methods have low computational complexities because the major computation is the multiplication of a thin matrix and a short matrix in each iteration. However, they are non-convex and their performances are sensitive to the given or estimated rank.

As the missing entries of a low-rank matrix can be optimized to make the matrix have lowest rank, nuclear-norm minimization based methods [1,2,20] were proposed for matrix completion. Nuclear-norm is defined as the sum of the singular values of a matrix and is a convex relaxation for matrix rank. Nuclear-norm minimization based matrix completion can be solved through different approaches such as the singular value thresholding (SVT) algorithm [21], inexact augmented Lagrange multiplier (IALM) method [22], and alternating direction method (ADM) [13,20]. Recently, a few extensions or improvements for nuclear-norm were applied to matrix completion [6,23–25]. For example, in [6], truncated nuclear-norm (TNN) minimization was proposed for matrix completion. Truncated nuclear-norm is defined as the sum of the smallest few singular values and is a better approximation than nuclear-norm for matrix rank. The reason is that the largest few singular values usually contain important information and should be preserved; on the other hand, the small singular values should be minimized. Compared with matrix factorization based methods, nuclear-norm minimization related methods have quite higher computational complexities because of the singular value decomposition (SVD) in each iteration even if economy or truncated SVD are utilized. However, nuclear-norm minimization related methods are convex, accurate, and do not require accurately estimated rank.

The matrix factorization and nuclear-norm minimization based methods are linear methods because the low-rankness is based on

* Corresponding author.
*E-mail address:* fanj.c.rick@gmail.com (J. Fan).

http://dx.doi.org/10.1016/j.neucom.2017.05.074
0925-2312/

linear latent variable model [26]. Therefore, they are not effective in recovering incomplete matrix in which the data are from nonlinear latent variable model [27–29]. To handle the non-linear problem, in [30], restricted Boltzmann machines (RBMs) was proposed for collaborative filtering. More recently, AutoEncoder based collaborative filtering [31–33] were proposed and outperformed RBMs based collaborative filtering. In [31], for a recommendation system (MovieLens datasets), it was proposed to train a different AutoEncoder for each user; all AutoEncoders have the same number of hidden units but may have different number of input units set as the number of ratings given by the user. The method avoided the influence of the missing ratings but one has to train a large number of AutoEncoders. In [32] and [33], the missing entries were replaced by pre-defined constants and all the data were used to train a single AutoEncoder in which only the reconstruction errors of the observed entries were considered. The method was called AutoEncoder based collaborative filtering (AECF). However, in AECF, the influence of the biases introduced by the pre-defined constants cannot be ignored. Moreover, the performances are quite sensitive to the pre-defined constants. At last, in these works of collaborative filtering, RBM and AutoEncoder were never compared with nuclear-norm related methods and the performance differences are unknown.

In this paper, we propose a novel method called AutoEncoder based matrix completion (AEMC). In AEMC, the parameters of the AutoEncoder and the missing entries of the data matrix are simultaneously optimized to minimize the reconstruction errors of the observed entries. In the AutoEncoder to be learned for AEMC, the hidden units are much fewer than the visible units. Such a structure indicates that the given variables are redundant and can be compressed into (called encoding) and represented by (called decoding) a fewer number of hidden or latent variables. It also indicates that a subset of the input variables is able to reconstruct the hidden or latent variables, which are further able to generate the remaining input variables. That is why AEMC is able to recover the missing entries of a matrix in which the data are assumed to be given by a nonlinear latent variable model. As deep-structure neural networks could be more effective than shallow-structure neural networks [34–37], AEMC is integrated with stacked AutoEncoders (SAEs) to form a deep learning based matrix completion (DLMC). In DLMC, first, an AEMC is implemented; then the hidden variables of AEMC is used to train SAEs; finally, fine-tuning is carried out on the deep-structure AEMC. To make the offline-learned models applicable to online missing entry recovery, out-of-sample extensions for AEMC and DLMC are provided in this paper. We compare AEMC and DLMC with state-of-the-art methods of matrix factorization, nuclear-norm minimization, truncated nuclear-norm minimization, and AECF in the tasks of synthetic matrix completion, image inpainting, and collaborative filtering. The experimental results show that AEMC and DLMC are more effective than other methods.

The contributions of this paper are summarized as follows. First, AEMC, as a novel method of matrix completion is proposed to recover the missing entries of incomplete matrix in which the data are given by nonlinear latent variable model. Second, AEMC is extended to DLMC, which is a deep learning method for matrix completion and is able to outperform shallow methods of matrix completion. Finally, out-of-sample extensions for AEMC and DLMC are proposed to recover online incomplete data.

The remaining content of this paper are organized as follows. In Section 2, the previous works of matrix completion are introduced. Section 3 details our proposed methods AEMC and DLMC. Section 4 compares the proposed methods with other state-of-the-art methods in the tasks of synthetic matrix completion, image inpainting, and collaborative filtering. Section 5 draws a conclusion for this paper.

## 2. Previous work of matrix completion

In this section, several representative methods of matrix completion will be introduced. Given that a low-rank matrix $X \in \mathbb{R}^{m\times n}$ is partially observed, the observed entries and their positions are noted as $M$ and $\Omega$, respectively. Then $X_{i,j} = M_{i,j}$ for each pair $(i, j) \in \Omega$. $X$ can be completed by solving the following matrix factorization problem [9,12,16–18]:

$$\min_{X,L,R} \tfrac{1}{2}(\|L\|_F^2 + \|R\|_F^2),$$
$$s.t.\ X = LR,\ X_{i,j} = M_{i,j},\ (i,j) \in \Omega \tag{1}$$

where $L \in \mathbb{R}^{m\times r}$ is a thin matrix and $R \in \mathbb{R}^{r\times n}$ is a short matrix. The parameter $r$ should be determined beforehand and the optimal value is the rank of $X$. In practice, because $X$ is incomplete, it is difficult to know or estimate $r$. To cope with the problem, [19] proposed a low-rank matrix fitting (LMaFit) algorithm for matrix completion. In LMaFit, with the model of (1), the parameter $r$ is dynamically and adaptively adjusted. These matrix factorization based methods are nonconvex and sensitive to the parameter $r$ though they have low computational complexities.

Matrix completion can also be solved by rank minimization. However, direct rank minimization is NP-hard. As a convex relaxation of matrix rank, nuclear-norm can be applied to matrix completion by solving the following problem

$$\min_X \|X\|_*,\ s.t.\ X_{i,j} = M_{i,j},\ (i,j) \in \Omega \tag{2}$$

where $\|\cdot\|_*$ denotes the nuclear norm, the sum of the singular values ($\sigma$) of a given matrix, i.e., $\|X\|_* = \sum_{i=1}^{min(m,n)} \sigma_i(X)$. Problem (2) is convex and can be solved by techniques such as the singular value thresholding (SVT) algorithm [21], inexact augmented Lagrange multiplier (IALM) method [22], and alternating direction method (ADM) [20].

As known, the largest few singular values of a matrix often contain important information and should be preserved if possible. Therefore, in [6], the truncated nuclear norm (TNN), a better rank approximation than nuclear-norm, was applied to matrix completion. TNN is defined as the nuclear norm subtracted by the sum of the largest few singular values, i.e., $\|X\|_r = \|X\|_* - \sum_{i=1}^{r} \sigma_i(X) = \sum_{i=r+1}^{min(m,n)} \sigma_i(X)$. For matrix completion, [6] proposed to solve the following problem

$$\min_X \|X\|_r,\ s.t.\ X_{i,j} = M_{i,j},\ (i,j) \in \Omega \tag{3}$$

which can be reformulated as

$$\min_X \|X\|_* - Tr(U_l X V_l^T),$$
$$s.t.\ X_{i,j} = M_{i,j},\ (i,j) \in \Omega, \tag{4}$$

where $U_l$ ($V_l$) are the first $r$ columns of $U$ ($V$) given by the singular value decomposition of $X$ in the $l$th iteration, i.e., $X_l = USV^T$ [6]. The optimization of (4) can be solved by alternating direction method of multipliers (ADMM) [38].

Recently, neural networks were applied to collaborative filtering [30–33]. For example, in [32], AutoEncoder based collaborative filtering (AECF) was proposed. In AECF, first, the missing entries of the data were replaced with pre-defined constants; then the data were utilized to learn an AutoEncoder, for which the reconstruction errors of the observed entries were minimized; finally, the missing entries were set as the corresponding outputs of the network.

## 3. Deep learning based matrix completion (DLMC)

### 3.1. Matrix completion by AutoEncoder and deep learning

Assume that a set of variables or measurements are given by the following nonlinear latent variable model

$$x = f(z) + \epsilon, \tag{5}$$

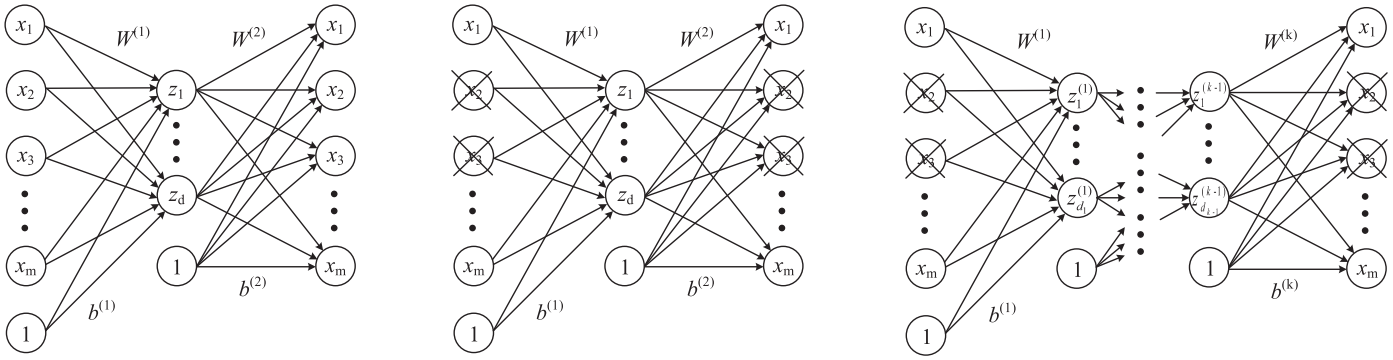


**Fig. 1.** Structures of AutoEncoder (left), AEMC (middle), and DLMC (right), in which $x$ is the vector of input variables, $z$ is the vector of latent variables, and a node with the symbol $\times$ is a missing entry.

where $x \in \mathbb{R}^m$ are the observations, $z \in \mathbb{R}^d$ are the latent variables, $d < m$, $f(\cdot)$ is a nonlinear map, and $\epsilon$ is an additive noise term. As $d < m$, $x$ is redundant. The nonlinear latent variable model is also the basis of manifold learning and nonlinear dimensionality reduction. Given a set of samples of $x$, a few methods such as kernel PCA (KPCA), Gaussian process latent variable (GPLVM) model, and AutoEncoder (AE) can be used to find the latent variables $z$ approximately:

$$z = f^{-1}(x), \tag{6}$$

where $f^{-1}(\cdot)$ is an approximation for the inverse of $f(\cdot)$. It is worth noting that KPCA and GPLVM are non-parameter methods, where $f(\cdot)$ and $f^{-1}(\cdot)$ cannot be given explicitly. On the contrary, AE can provide both $f(\cdot)$ and $f^{-1}(\cdot)$ explicitly. Given a data matrix $X \in \mathbb{R}^{m\times n}$ whose rows and columns are variables and samples, respectively, AE solves the following problem

$$\begin{aligned}\min_{\{W^{(1)},b^{(1)},W^{(2)},b^{(2)}\}} & \tfrac{1}{2n}\sum_{i=1}^{n}(\|x_i - g^{(2)}(W^{(2)}g^{(1)}(W^{(1)}x_i \\ & + b^{(1)}) + b^{(2)})\|^2) \\ & + \tfrac{\lambda}{2}(\|W^{(1)}\|_F^2 + \|W^{(2)}\|_F^2),\end{aligned} \tag{7}$$

where $g^{(1)}(\cdot)$, $g^{(2)}(\cdot)$ are the activation functions, $W^{(1)} \in \mathbb{R}^{d\times m}$, $W^{(2)} \in \mathbb{R}^{m\times d}$ are the weights matrices, $b^{(1)}$, $b^{(2)}$ are the vectors of bias terms, of the hidden layer and output layer, respectively. Hence, we have

$$\begin{aligned} z &= g^{(1)}(W^{(1)}x + b^{(1)}) \approx f^{-1}(x), \\ x &= g^{(2)}(W^{(2)}z + b^{(2)}) \approx f(z). \end{aligned} \tag{8}$$

The structure of AE is shown by the first chart in Fig. 1.

When the data matrix $X$ is partially observed, problem (7) is intractable. We denote the observed entries and their positions as $M$ and $\Omega$, i.e., $X_{i,j} = M_{i,j}$ for each pair $(i, j) \in \Omega$. We also assume that the observed entries are sampled uniformly at random and the sampling rate is $\rho$. Hence, the number of the observed entries is $|\Omega| = \rho mn$. It can be found that, in terms of the model in (5), if $\rho m < d$, $z$ cannot be determined by the observed entries, which indicates that the missing entries cannot be determined. Therefore, a necessary condition for a successful recovery for the missing entries is $\rho \geq d/min(m, n)$. To compute the parameters $\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}$ and recover the missing entries of $X$ simultaneously, we propose to solve the following problem:

$$\begin{aligned}\min_{\{\theta,X\}} & \tfrac{1}{2n}\sum_{i=1}^{n}(\|\Psi_i \odot (x_i - g^{(2)}(W^{(2)}g^{(1)}(W^{(1)}x_i \\ & + b^{(1)}) + b^{(2)}))\|^2) \\ & + \tfrac{\lambda}{2}(\|W^{(1)}\|_F^2 + \|W^{(2)}\|_F^2), \\ & s.t.\ X_{i,j} = M_{i,j},\ \ (i,j) \in \Omega.\end{aligned} \tag{9}$$

In (9), $\odot$ denotes the Hadamard product. $\Psi_i$ is the $i$th column of a binary matrix $\Psi$, in which the entry $\Psi_{ij}$ is 1 if $(i, j) \in \Omega$ and 0 otherwise. Problem (9) can be reformulated as

$$\begin{aligned}\min_{\{\theta,X\}} & \tfrac{1}{2n}\|\Psi \odot (X - h(X,\theta))\|_F^2 + \Gamma(\theta), \\ & s.t.\ X_{i,j} = M_{i,j},\ \ (i,j) \in \Omega.\end{aligned} \tag{10}$$

where $\Gamma(\theta) = \frac{\lambda}{2}(\|W^{(1)}\|_F^2 + \|W^{(2)}\|_F^2)$. $h(\cdot)$ denotes a nonlinear function carried out in two steps: first, $X \in \mathbb{R}^{m\times n}$ is compressed into a matrix $Z \in \mathbb{R}^{d\times n}$ of latent variables; then, $Z$ is mapped back to reconstruct $X$. The second chart in Fig. 1 gives the network structure of AEMC. Because $X$ is partially observed, only the reconstruction errors of the observed entries are considered. According to (8), problem (10) is equivalent to

$$\begin{aligned}\min_{\{\theta,X\}} & \tfrac{1}{2n}\|\Psi \odot (X - f(Z))\|_F^2 + \Gamma(\theta), \\ & s.t.\ Z = f^{-1}(X),\ X_{i,j} = M_{i,j},\ \ (i,j) \in \Omega.\end{aligned} \tag{11}$$

Because $X$ is given by the model of (5), there exists an approximation for $f(\cdot)$ performing on the low-dimensional latent variables $Z$ that minimize the reconstruction errors for the observed entries of $X$. By solving (10), $f(\cdot)$ and $Z$ can be obtained. Therefore, the missing entries of $X$ are recovered as

$$X_{i,j} = [f(Z)]_{i,j},\ \ (i,j) \in \bar{\Omega}. \tag{12}$$

where $\bar{\Omega}$ denotes the positions of the missing entries of $X$.

As known, a multiple-hidden-layers neural network could be more effective than a single-hidden-layer neural network in approximating nonlinear functions. In other words, deep structures could outperform shallow structures [34–37]. Therefore, based on (9), we propose the following deep learning based matrix completion (DLMC):

$$\begin{aligned}\min_{\{\theta,X\}} & \tfrac{1}{2n}\sum_{i=1}^{n}(\|\Psi_i \odot (x_i - g^{(k)}(g^{(k-1)}(\ldots g^{(1)}(x_i,\theta^{(1)}) \\ & \ldots,\theta^{(k-1)}),\theta^{(k)}))\|^2) + \tfrac{\lambda}{2}\sum_{j=1}^{k}\|W^{(j)}\|_F^2, \\ & s.t.\ X_{i,j} = M_{i,j},\ \ (i,j) \in \Omega,\end{aligned} \tag{13}$$

where $\theta^{(j)} = \{W^{(j)}, b^{(j)}\}$, $j = 1, 2, \ldots, k$, and $k$ is the total number of mappings. The network structure of DLMC is shown by the third

chart in Fig. 1. It is troublesome to solve (13) directly because the solution is quite vulnerable to bad local minima. However, problem (13) can be formulated as stacked AutoEncoders and then solved by greedy layer-wise training [35,37]. Hence, $k$ is an even number; DLMC consists of $k/2$ AutoEncoders; the $i$th ($i \le k/2$) layer and $k-i+1$th layer are the encoding and decoding layers, respectively of the $i$th AutoEncoder. Similar with (9), problem (13) can also be rewritten as (10), where $h(X, \theta)$ denotes a function on $X$ carried out through multiple stages of nonlinear mapping. Moreover, problem (13) can also be written into the form of (11) that minimizes $\|\Psi \odot (X - f(Z))\|_F^2$, where

$$\begin{aligned} z &= g^{\left(\frac{k}{2}\right)}\left(W^{\left(\frac{k}{2}\right)}\left(\dots\left(g^{(1)}\left(W^{(1)}x + b^{(1)}\right)\dots\right)\right) + b^{\left(\frac{k}{2}\right)}\right) \approx f^{-1}(x), \\ x &= g^{(k)}\left(W^{(k)}\left(\dots\left(g^{\left(\frac{k}{2}+1\right)}\left(W^{\left(\frac{k}{2}+1\right)}z + b^{\left(\frac{k}{2}+1\right)}\right)\dots\right)\right) + b^{(k)}\right) \\ &\approx f(z). \end{aligned} \tag{14}$$

Therefore, in (13), $X$ is compressed into $Z$ through multiple stages of nonlinear mapping and $Z$ is then mapped back to reconstruct $X$ through multiple stages of nonlinear mapping. The missing entries of $X$ and parameters $\{\theta^{(j)}\}_{j=1}^{k}$ are optimized simultaneously to minimize the reconstruction errors for the observed entries of $X$. After $f(\cdot)$ and $Z$ are obtained, the missing entries of $X$ are recovered as (12). In practice, DLMC can be solved through the following steps. First, an AEMC is trained. Then the latent variables of AEMC are used to train a series of AutoEncoders sequentially. Finally, all AutoEncoders are stacked together as an incomplete-data deep-AutoEncoder and fine-tuning is performed to refine the parameters and missing entries of $X$. Compared with AEMC that approximates $f(\cdot)$ and $f^{-1}(\cdot)$ through single-layer networks, DLMC approximates $f(\cdot)$ and $f^{-1}(\cdot)$ through multi-layer networks. DLMC is expected to outperform AEMC and conventional methods.

In DLMC, the first step is to design the structure of the network, including the number of AutoEncoders, the number of hidden units of each AutoEncoder, and activation functions. When the number of AutoEncoder is one, DLMC reduces to AEMC. The parameter $d$, the optimal number of hidden units in AEMC or the smallest number of hidden units in DLMC, is the dimensionality of the latent variable $z$. But in practice, $d$ is unknown and also difficult to estimate. However, as mentioned previously, $d$ should be smaller than $p$, which is the number of observed entries in each data vector. Therefore, $d$ could be set as $d < (1-\delta)min(m,n)$, where $\delta$ is the missing rate. For the activation functions, sigmoid function and hyperbolic tangent function are widely-used choices [39]. In AEMC and DLMC, type of the activation functions for the last layer (output layer) should be determined according to the range of the data. Alternatively, the data should be transformed to match the output range of the activation functions. For example, in image inpainting task, the activation function could be sigmoid function because the input can be easily transformed into the interval of [0, 1].

### 3.2. Optimizations and out-of-sample extensions for AEMC and DLMC

The optimization problem of AEMC in (10) is nonconvex but the local minima could be found by widely-used nonlinear optimization techniques such as gradient descent method [39], nonlinear conjugate gradient method [40], BFGS, or Limited-memory BFGS (LBFGS) [41]. In general, these optimization techniques require only the gradient of the objective function. In (10), we denote the objective function as

$$J(X,\theta) = L(X,\theta) + \Gamma(\theta), \tag{15}$$

where $L(X,\theta) = \frac{1}{2n}\|\Psi \odot (X - h(X,\theta))\|_F^2$. $\partial L(X,\theta)/\partial\theta$ can be computed by back-propagation algorithm. $\partial L(X,\theta)/\partial X$ is given as

$$\frac{\partial L}{\partial X} = \frac{1}{n}(\Psi \odot (X - h(X,\theta)) + \omega(X,\theta), \tag{16}$$

where $\omega(X,\theta) = \frac{\partial L}{\partial(-h(X,\theta))}\frac{\partial(-h(X,\theta))}{\partial X}$. As the input to the first hidden layer is $V = W^{(1)}X + b^{(1)}$, we have

$$\begin{aligned} \omega(X,\theta) &= \frac{\partial L}{\partial(-h(X,\theta))}\frac{\partial(-h(X,\theta))}{\partial V}\frac{\partial V}{\partial X} \\ &= W^{(1)^T}\frac{\partial L}{\partial V}, \end{aligned} \tag{17}$$

where $\partial L/\partial V$ is actually an intermediate in computing $\partial L(X,\theta)/\partial\theta$. Then $\partial J/\partial X$ and $\partial J/\partial\theta$ can be obtained. In AEMC, by using the above formulations, $X$ and $\theta$ can be optimized directly. In DLMC, as mentioned in the previous section, $X$ and $\theta$ can be obtained by solving the following steps: AEMC→ SAEs→ DeepAEMC. In this paper, we propose to use nonlinear conjugate gradient method to solve the optimizations of AEMC and DLMC because its computational cost is often much lower than that of BFGS and L-BFGS.

Although matrix completion is usually an offline task, it is practical and important to recover online incomplete data samples. Once AEMC and DLMC have been accomplished with offline dataset, the trained networks can be utilized to complete online new data samples. Given a new incomplete data sample $x_{new} = [x_{new}^{obs}; x_{new}^{mis}]$, a straightforward approach is to solve the following problem

$$\min_{x_{new}^{mis}} \tfrac{1}{2}\|\Psi_{new} \odot ([x_{new}^{obs}; x_{new}^{mis}] - h([x_{new}^{obs}; x_{new}^{mis}], \theta))\|_F^2. \tag{18}$$

When the online data is small, (18) can be solved efficiently. Otherwise, there is no need to solve (18) iteratively. Because $x_{new}$ is from the same nonlinear latent variable model as the offline data, we have

$$\hat{x}_{new} = g^{(2)}(W^{(2)}z_{new} + b^{(2)}), \tag{19}$$

and

$$\begin{aligned} \hat{x}_{new}^{obs} &= g^{(2)}(W^{(2)^{obs}}z_{new} + b^{(2)^{obs}}), \\ \hat{x}_{new}^{mis} &= g^{(2)}(W^{(2)^{mis}}z_{new} + b^{(2)^{mis}}), \end{aligned} \tag{20}$$

where $\hat{x}_{new} = [\hat{x}_{new}^{obs}; \hat{x}_{new}^{mis}]$ is the actual output of the network. With (20), there exists a least-square solution of $z$ as

$$\hat{z}_{new} = Q(W^{(2)^{obs}})^T((g^{(2)})^{-1}(\hat{x}_{new}^{obs}) - b^{(2)^{obs}}), \tag{21}$$

where

$$Q = ((W^{(2)^{obs}})^T W^{(2)^{obs}})^{-1}. \tag{22}$$

Then we have

$$\hat{x}_{new}^{mis} = g^{(2)}(W^{(2)^{mis}}\hat{z}_{new} + b^{(2)^{mis}}). \tag{23}$$

The solution given by (21) minimizes the squared error

$$e_z = \|z_{new} - \hat{z}_{new}\|^2, \tag{24}$$

where

$$z_{new} = g^{(1)}(W^{(1)}[x_{new}^{obs}; x_{new}^{mis}] + b^{(1)}). \tag{25}$$

Given that the number of the observed elements of $x_{new}$ is $p$, then $W^{(2)^{obs}} \in \mathbb{R}^{p\times d}$. When $p > d$, $(W^{(2)obs})^T W^{(2)obs}$ is invertible and (21) exists. The estimation error for the missing entries is

$$\begin{aligned} e_{\hat{x}_{new}^{mis}} &= \|\hat{x}_{new}^{mis} - \hat{\hat{x}}_{new}^{mis}\|^2 \\ &= \|g^{(2)}(W^{(2)^{mis}}z_{new} + b^{(2)^{mis}}) \\ &\quad - g^{(2)}(W^{(2)^{mis}}\hat{z}_{new} + b^{(2)^{mis}})\|^2. \end{aligned} \tag{26}$$

Because the activation function $g^{(2)}(\cdot)$ is Lipschitz continuous, we have

**Table 1**
RMSE (%) of synthetic matrix completion.

| Missing rate | MF(LMaFit) | NNM(IALM) | TNNM(ADMM) | AECF | AEMC | DLMC |
|---|---|---|---|---|---|---|
| 10% | 29.41 | 27.98 | 26.52 | 8.93 | **5.78** | **4.06** |
| 20% | 30.13 | 29.06 | 28.18 | 11.54 | **6.67** | **4.73** |
| 30% | 30.78 | 31.67 | 30.32 | 14.32 | **7.27** | **5.08** |
| 40% | 31.16 | 33.72 | 32.08 | 16.83 | **8.49** | **5.69** |
| 50% | 32.92 | 36.45 | 33.12 | 17.49 | **10.26** | **7.03** |
| 60% | 34.09 | 39.21 | 35.64 | 22.03 | **14.14** | **8.94** |
| 70% | 36.15 | 43.83 | 40.16 | 27.25 | **17.32** | **13.51** |
| 80% | 46.44 | 52.63 | 51.16 | 48.27 | **26.53** | **25.31** |

$$\begin{aligned} e_{\hat{x}_{new}^{mis}} &\le L\|(W^{(2)^{mis}}(z_{new}-\hat{z}_{new})\|^2 \\ &\le L\|W^{(2)^{mis}}\|_2^2\|z_{new}-\hat{z}_{new}\|^2 \\ &= L\|W^{(2)^{mis}}\|_2^2 e_z = \eta e_z, \end{aligned} \tag{27}$$

where $L$ is the Lipschitz constant. For example, the Lipschitz constant of sigmoid function is $L=0.25$. It can be found that $e_{\hat{x}^{mis}}$ is bounded with $\eta e_z$. As $e_z$ has been minimized, $e_{\hat{x}_{new}^{mis}}$ is minimized. Moreover, as the conditional number of $W^{(2)obs}$ is $\kappa \propto d/p$ and $e_z \propto \kappa$, we have

$$e_z \propto d/p, \tag{28}$$

which means that the recovery errors of the missing entries are smaller when the dimensionality of the latent variable is smaller compared with the number of observed entries.

In terms of AEMC, referring to (21) and (22), we can compute $z_{new}$ as

$$z_{new} = Q(W^{(2)^{obs}})^T((g^{(2)})^{-1}(x_{new}^{obs}) - b^{(2)^{obs}}). \tag{29}$$

Then we can recover the missing entries of $x_{new}$ as

$$x_{new}^{mis} = g^{(2)}(W^{(2)^{mis}} z_{new} + b^{(2)^{mis}}). \tag{30}$$

In terms of DLMC, $W^{(2)}$ in the above two equations should be replaced with the $W$ for the output layer of the deep neural network learned by DLMC.

## 4. Experiments

In this section, the proposed methods AEMC and DLMC will be compared with four state-of-the-art methods of matrix completion in the tasks of synthetic matrix completion, image inpainting and collaborative filtering. The first compared method is the matrix factorization based method solved by LMaFit, which is noted by MF(LMaFit) [19]. The second one is the nuclear-norm minimization based method solved by IALM, which is noted by NNM(IALM) [22]. The third one is the truncated nuclear-norm minimization based method solved by ADMM, which is noted by TNNM(ADMM) [6]. The last one is the AECF method [32]. The optimizations of AECF, AEMC, and DLMC are solved by nonlinear conjugate gradient method. The performances are evaluated by the widely-used normalized mean absolute error (NMAE) [2,19], which is defined as

$$NMAE = \frac{1}{(X_{max}-X_{min})|\bar{\Omega}|}\sum_{(i,j)\in\bar{\Omega}}|\hat{X}_{ij}-X_{ij}|, \tag{31}$$

where $\hat{X}$ is the recovered matrix, $X$ is the true matrix, $\bar{\Omega}$ denotes the positions of the missing entries, $X_{max}$, $X_{min}$ are the maximum and minimum of $X$, respectively.

### 4.1. Matrix completion of synthetic data

To intuitively evaluate the performance of the proposed methods in handling nonlinear data, we produce a synthetic dataset for matrix completion by the following approach:

$$X = g(1.2(0.5g^2(AB) - g(AB) - 1)), \tag{32}$$

where $A \in \mathbb{R}^{m\times d}$ and $B \in \mathbb{R}^{d\times n}$ are random Gaussian matrices, $g(t) = 1.7159tanh(\frac{2}{3}t)$ is element-wise activation function. In this study, we set $m=200$, $n=500$, and $d=10$. As the data range of $X$ is unknown, the following relative mean square error (RMSE) is used to evaluate the performance of matrix completion

$$RMSE = \sum_{(i,j)\in\bar{\Omega}}(\hat{X}_{ij}-X_{ij})^2 / \sum_{(i,j)\in\bar{\Omega}}X_{ij}^2. \tag{33}$$

The parameters of all methods are carefully determined to give their best performances. The average results of 50 repeated trials are reported in Table 1. It is found that the proposed methods especially DLMC outperform other methods significantly when the missing rate increases from 10% to 80%. The result confirms that the proposed methods AEMC and DLMC are able to handling nonlinear data while conventional methods such as MF, NNM, and TNNM fail evidently.

### 4.2. Single-image inpainting

Image inpainting is an image restoration problem to remove noises, masks, or recover missing parts of images [6,42]. In the single-image inpainting task, each image forms a data matrix. We use two well-known images in the field of image processing, and they are the *Cameraman* (256 × 256 gray-scale) and *Lena* (256 × 256 × 3 RGB, unfolded to 256 × 768). Two types of masks are considered in this study. The first one is a random pixel mask for which 50% pixels are randomly removed. Another one is a text mask of English words. To make all methods perform their best, their parameters are carefully determined. Specifically, in MF(LMaFit), the rank is set as 25 because the auto-estimation in this problem often leads to bad performance. In TNNM(ADMM), the parameter $r$ is set as 10. In AECF, the weight-decay penalty $\lambda$ is set as 0.001; the number of hidden units is set as 50 for *Cameraman* and 75 for *Lena*; the missing entries of the inputs are predefined with the results of MF(LMaFit) otherwise the performances are quite bad. In AEMC, the weight-decay penalty $\lambda$ is set as 0.01; the number of hidden units is set as 50 for *Cameraman* and 75 for *Lena*. In DLMC, the weight-decay penalty $\lambda$ is set as 0.01; the network has three hidden layers; the numbers of hidden units are set as [100 50 100] for *Cameraman* and [100 75 100] for *Lena*.

Figs. 2–5 show the original images, masked images, and examples of restored images given by the six methods. As can be seen, the recovered images given by MF(LMaFit) and AECF are more coarse than that given by other methods; the performances of DLMC are the best. The average NMAEs of 20 repeated trials are reported in Table 2, in which the smallest two NMAEs are highlighted in bold type. It can be found that, in general, in terms of effectiveness, the six methods can be sorted as DLMC> TNNM> AEMC> AECF> NNM> MF. Additionally, Fig. 6 shows NMAEs for *Lena* with random pixel masks of different missing rates (10% – 90%). As can be seen, AEMC can outperform TNNM when

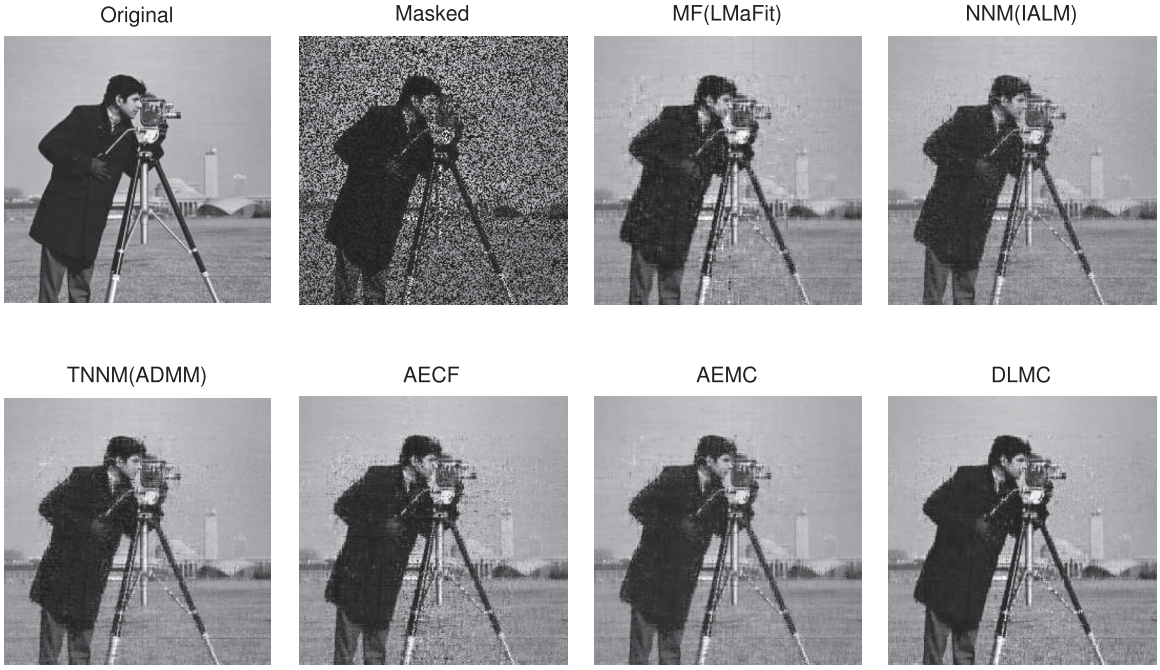



**Fig. 2.** Inpainting performance for *Cameraman* with random pixel mask.

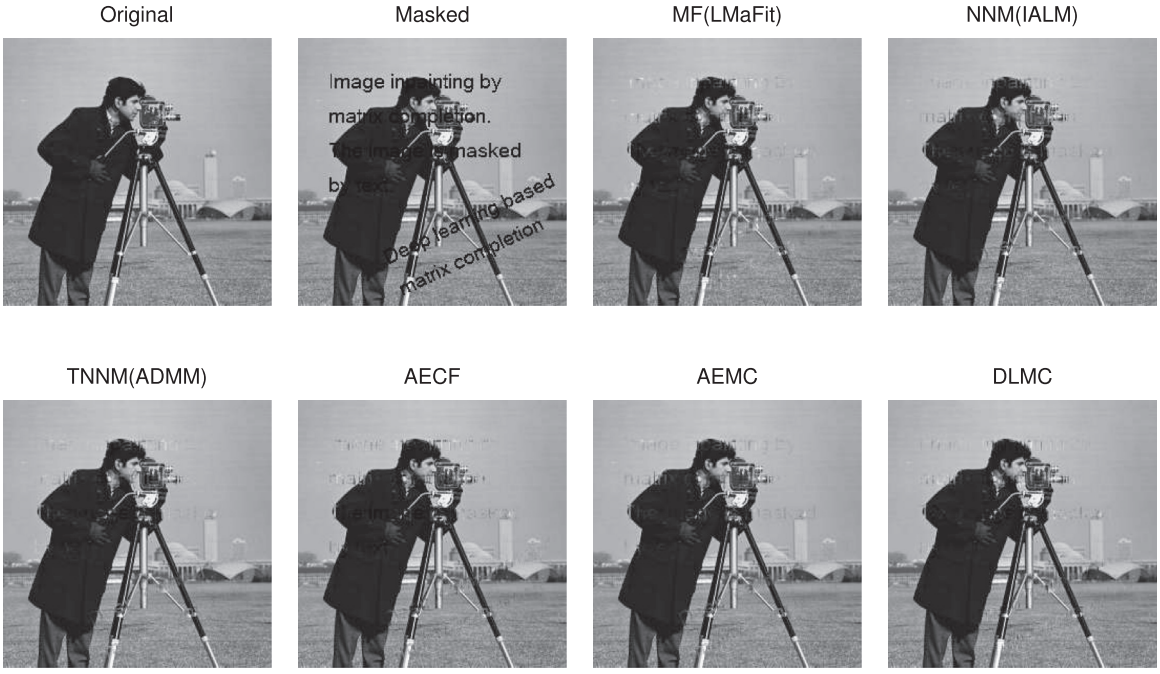



**Fig. 3.** Inpainting performance for *Cameraman* with text mask.

the missing rates are higher than 60%; DLMC outperforms other methods significantly.

### 4.3. Group-image inpainting

In this group-image inpainting task, each image forms a vector (column) of the matrix. The MNIST dataset of handwritten digits ($28 \times 28$) [43] is used in this study. Two types of masks are used for the images: random pixel (70%) mask and square block mask ($14 \times 14$). The offline inpainting is carried out on 1000 images. To get the best performance of each method, their parameters are set as follows. In MF(LMaFit), the rank is adjusted adaptively within [10,50]. In TNNM(ADMM), the parameter $r$ is set as 5. In AECF, the weight-decay penalty $\lambda$ is set as 0; the number of hidden units is set as 50. In AEMC, the weight-decay penalty $\lambda$ is set as $10^{-4}$; the number of hidden units is set as 50. In DLMC, the weight-decay penalty $\lambda$ is set as 0.01; the network has three hidden layers; the numbers of hidden units are set as [100 50 100].

Fig. 7 shows examples of original, pixel-masked, and the recovered digits. As can be seen, the recovery performances of the proposed methods AEMC and DLMC are much better than that of other methods, especially for digits $\{5-9\}$. Fig. 8 shows examples of recovering block-masked digits. As can be seen, AEMC and DLMC outperform other methods significantly, especially for

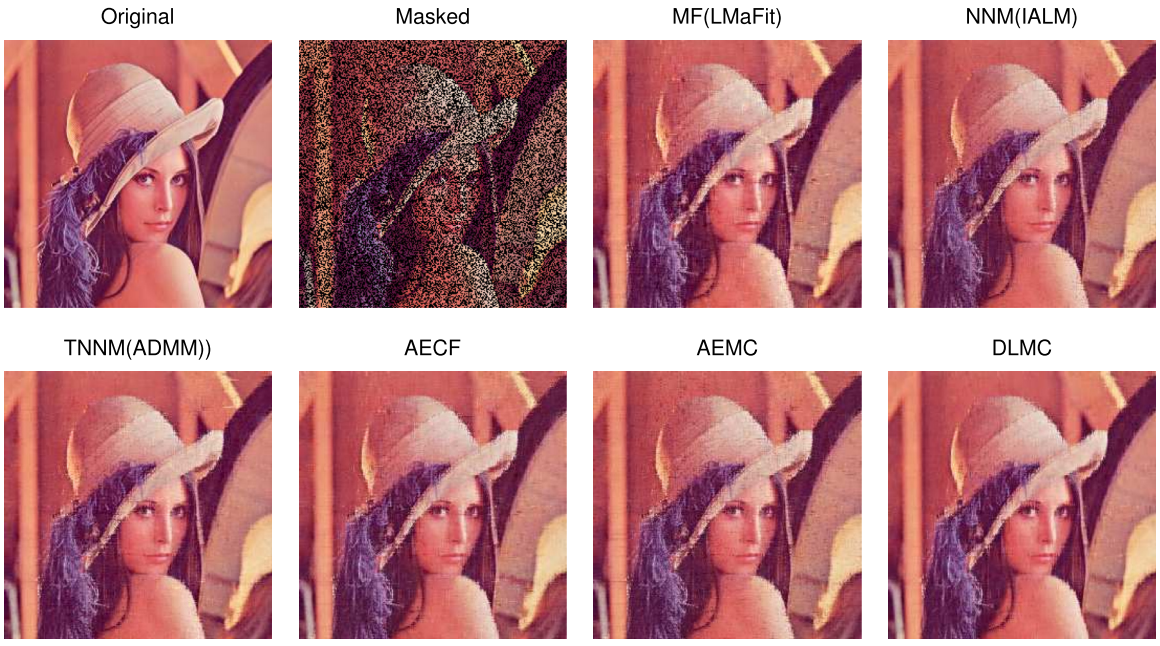



**Fig. 4.** Inpainting performance for *Lena* with random pixel mask.

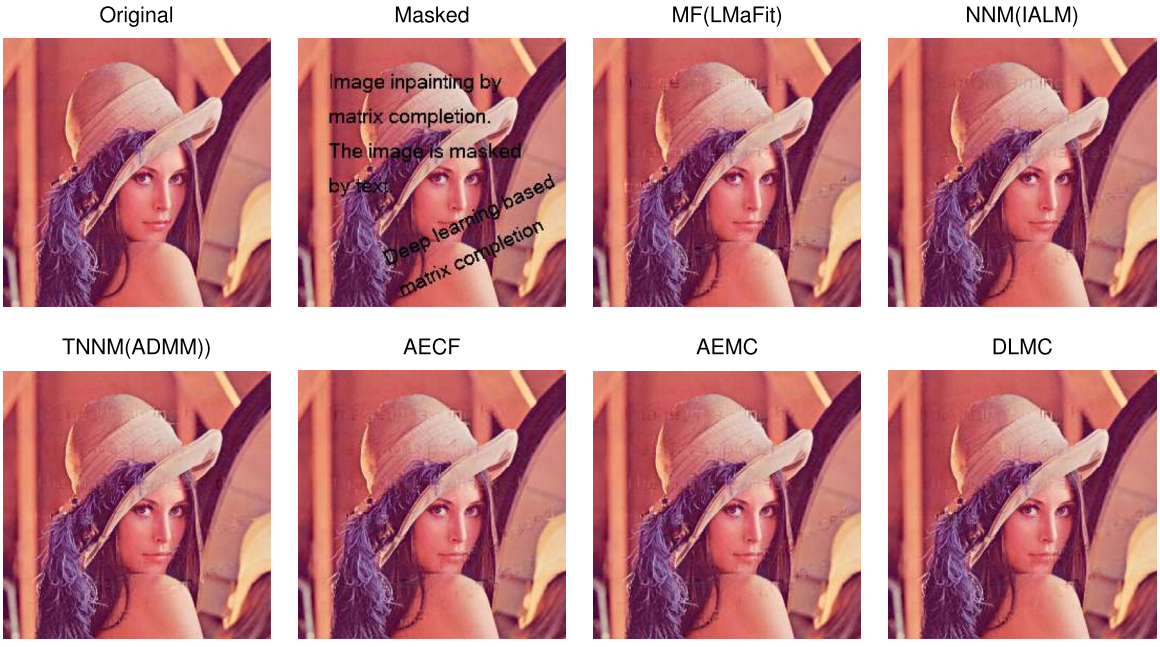



**Fig. 5.** Inpainting performance for *Lena* with text mask.

**Table 2**
NMAE (%) of single-image inpainting.

| Image | Mask type | MF(LMaFit) | NNM(IALM) | TNNM(ADMM) | AECF | AEMC | DLMC |
|---|---|---|---|---|---|---|---|
| Cameraman | Random | 6.04 | 5.35 | 5.08 | 5.21 | **5.03** | **4.63** |
| | Text | 8.96 | 7.83 | **7.45** | 7.71 | 7.59 | **6.83** |
| Lena | Random | 5.38 | 4.91 | 4.59 | 5.06 | **4.42** | **4.18** |
| | Text | 6.38 | 5.45 | **5.14** | 6.12 | 5.28 | **5.04** |

digits $\{4-8\}$. The models of these methods learned offline are then used for online inpainting on 10,000 images. The average NMAEs of 20 repeated trials are reported in Table 3. It can be found that, in both offline stage and online stage, the recovery errors of AEMC and DLMC are significantly lower than that of other methods.

### 4.4. Collaborative filtering

In this study, three datasets of collaborative filtering are used. The first one is the Jester-joke dataset-1 [44], in which the data are from 24,983 users who have rated 36 or more of 100 jokes. The rating values are within $[-10,10]$. We use the ratings of 5000 users

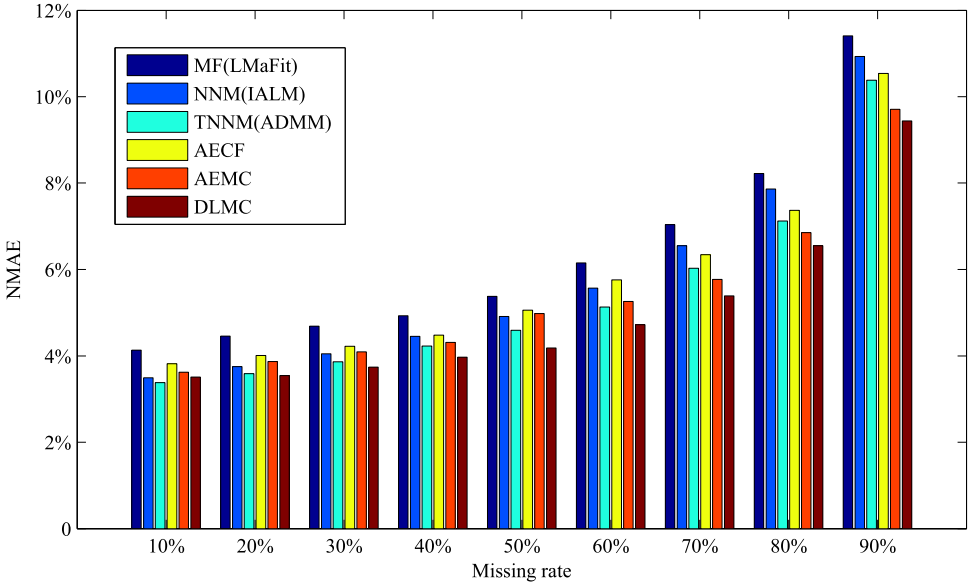



**Fig. 6.** NMAE for *Lena* with random pixel masks of different missing rates.

**Table 3**
NMAE (%) of group-image inpainting on MNIST data.

| Mask type | Stage | MF(LMaFit) | NNM(IALM) | TNNM(ADMM) | AECF | AEMC | DLMC |
|---|---|---|---|---|---|---|---|
| Random | Offline | 11.08 | 9.82 | 9.45 | 8.73 | **8.18** | **6.77** |
| | Online | 11.05 | 10.35 | 10.28 | 10.31 | **8.59** | **7.25** |
| Block | Offline | 18.22 | 15.72 | 15.21 | 14.75 | **13.81** | **13.26** |
| | Online | 23.24 | 21.79 | 21.08 | 15.51 | **14.25** | **13.78** |

**Fig. 7.** Inpainting performance for MNIST images with random pixel mask: row-1:original, row-2:masked, row-3:MF(LMaFit), row-4:NNM(IALM), row-5:TNNM(ADMM), row-6:AECF, row-7:AEMC, row-8:DLMC.

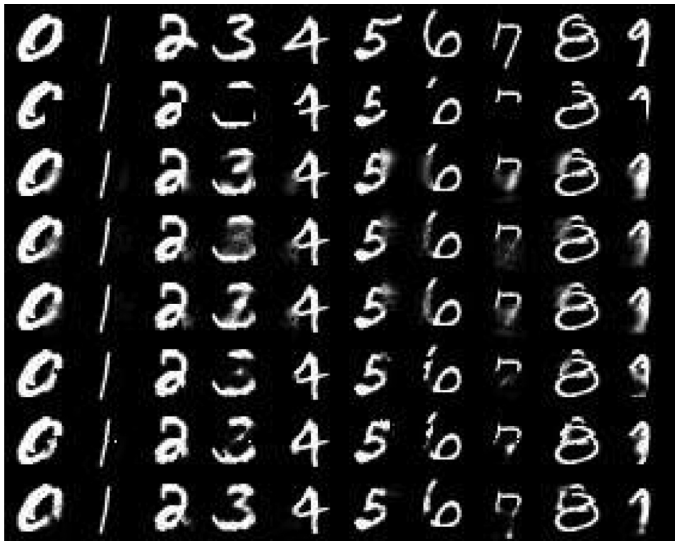


**Fig. 8.** Inpainting performance for MNIST images with block mask: row-1:original, row-2:masked, row-3:MF(LMaFit), row-4:NNM(IALM), row-5:TNNM(ADMM), row-6:AECF, row-7:AEMC, row-8:DLMC.

to perform offline missing entry recovery and the ratings of other 5000 users to perform online missing entry recovery. The second dataset is the MovieLens 100k [45], which consists of 100,000 ratings (1–5) from 943 users on 1682 movies. Each user has rated at least 20 movies. The third dataset is the MovieLens 1M [45], which consists of 1 million ratings (1–5) from 6040 users on 3260 movies. For the MovieLens 100k dataset, we remove the items rated by no more than 10 users. For the MovieLens 100k (1M) dataset, the data of the first 700 (2000) users are used for offline missing entry recovery and the data of the remaining 243 (4040) users are used for online missing entry recovery. It is worth noting that the three datasets are originally incomplete. In the Jester-joke data, about 72% of the entries are known. In the MovieLens datasets, only about 9% of the entries are known. Therefore, the missing rates in this study are to the known entries, not all entries. We randomly remove 30% and 50% of the known entries of the three datasets to test the performances of the proposed methods.

**Table 4**
NMAE (%) of collaborative filtering on Jester-joke 1 data.

| Missing rate | Stage | MF(LMaFit) | NNM(IALM) | TNNM(ADMM) | AECF | AEMC | DLMC |
|---|---|---|---|---|---|---|---|
| 0.3 | Offline | 16.45 | 16.43 | 16.11 | 16.29 | **16.17** | **16.03** |
| | Online | 15.89 | 15.87 | 15.88 | 16.08 | **15.74** | **15.53** |
| 0.5 | Offline | 16.65 | 16.91 | 16.53 | 16.64 | **16.38** | **16.31** |
| | Online | 16.28 | 16.21 | 16.24 | 16.32 | **16.09** | **15.97** |

**Table 5**
NMAE (%) of collaborative filtering on MovieLens 100k data.

| Missing rate | Stage | MF(LMaFit) | NNM(IALM) | TNNM(ADMM) | AECF | AEMC | DLMC |
|---|---|---|---|---|---|---|---|
| 0.3 | Offline | 19.25 | 19.69 | 19.78 | 19.31 | **18.87** | **18.39** |
| | Online | 19.14 | 19.06 | 19.22 | 19.05 | **18.52** | **18.21** |
| 0.5 | Offline | 20.26 | 20.22 | 20.45 | 19.52 | **19.06** | **18.75** |
| | Online | 20.78 | 20.91 | 21.17 | 19.41 | **19.14** | **18.71** |

**Table 6**
NMAE (%) of collaborative filtering on MovieLens 1M data.

| Missing rate | Stage | MF(LMaFit) | NNM(IALM) | TNNM(ADMM) | AECF | AEMC | DLMC |
|---|---|---|---|---|---|---|---|
| 0.3 | Offline | 18.61 | 18.51 | 18.34 | 18.49 | **18.17** | **18.03** |
| | Online | 18.14 | 18.17 | 18.27 | 18.11 | **17.95** | **17.82** |
| 0.5 | Offline | 18.63 | 18.95 | 18.79 | 18.73 | **18.36** | **18.31** |
| | Online | 18.48 | 18.49 | 18.55 | 18.51 | **18.22** | **18.15** |

**Table 7**
Computational time (second) in the tasks of image inpainting and collaborative filtering.

| Data | Size | MF(LMaFit) | NNM(IALM) | TNNM(ADMM) | AECF | AEMC | DLMC |
|---|---|---|---|---|---|---|---|
| *ameraman* | $256 \times 256$ | 3 | 13 | 32 | 7 | 12 | 33 |
| *Lena* | $256 \times 768$ | 4 | 18 | 28 | 22 | 39 | 95 |
| MNIST | $784 \times 1000$ | 12 | 407 | 453 | 98 | 157 | 372 |
| Jester-joke 1 | $100 \times 5000$ | 16 | 67 | 78 | 41 | 81 | 176 |
| MovieLens 100k | $700 \times 1682$ | 23 | 516 | 638 | 98 | 181 | 436 |
| MovieLens 1M | $2000 \times 3260$ | 198 | 10,800 | 11,200 | 690 | 1428 | 3177 |

The parameters of the considered methods are carefully determined as follows to give their best performances. In MF(LMaFit), the rank is adjusted adaptively within [1,30] for Jester-joke data and [5,50] for MovieLens data. In TNNM(ADMM), the parameter $r$ is set as 1 for Jester-joke data and 3 for MovieLens data. In AECF, the weight-decay penalty $\lambda$ is set as $10^{-3}$ for both datasets; the number of hidden units is set as 5 for Jester-joke data and 40 for MovieLens data. In AEMC, the weight-decay penalty $\lambda$ is set as $10^{-3}$ for both datasets; the number of hidden units is set as 5 for Jester-joke data and 40 for MovieLens data. In DLMC, the weight-decay penalty $\lambda$ is set as 0.01 for Jester-joke data and 0.03 for MovieLens data; the network has three hidden layers; the numbers of hidden units are set as [10 5 10] for Jester-joke data and [60 40 60] for MovieLens data. The average results of 20 repeated trials are reported in Tables 4–6. As can be seen, the proposed AEMC and DLMC significantly outperform other methods in all cases.

### 4.5. Analysis of computational complexity

The computational complexity of AEMC and DLMC are dominated by the computation of gradient. In each iteration of AEMC, the flop count of computing the gradient is about $8nmd$. In the pre-training and fine-tuning stages of DLMC, the flop count of computing the gradient is about $8nd_{j-1}d_j$, where $1 \le j \le k/2$ and $d \le d_j < d_{j-1} \le m$; $d_j$ and $d_{j-1}$ denote the numbers of latent variables and visible variables of the $j$th AutoEncoder; then the total flop count is about $16n\sum_{j=1}^{k/2} d_{j-1}d_j$. Because of the singular value decomposition, the flop count in each iteration of NNM and TNNM is about $14mn^2 + 8m^3$ if $m < n$. The flop count in each iteration of MF is about $4nmd$. As $d < <m$, MF is significantly faster than other methods. Compared with MF, NNM, and TNNM, AECF, AEMC, and DLMC often require more iterations to converge. However, AEMC and DLMC can be more efficient than NNM and TNNM when $m$ and $n$ are large. The approximate space complexity of these methods are as follows: MF/AECF/AEMC $O(mn + md + nd)$; NNM/TNNM $O(mn + mm + nn)$; DLMC $O(mn + \sum_{j=1}^{k/2}(d_{j-1}d_j + nd_j))$. It is found that the space complexities of NNM, TNNM and DLMC are higher than that of other methods.

All the methods were programmed in MATLAB and run on a computer with Intel Core 2 Quad CPU (3 GHz) and 4 GB RAM. The computational time of all methods on *Cameraman* (50% random mask), *Lena* (50% random mask), MNIST (70% random mask), Jester-joke 1 (30% missing rate), and MovieLens 100*k*/1*M* (30% missing rate) are reported in Table 7. MF is always the fastest one because it only involves matrix multiplication in each iteration. For the MNIST data and MovieLens data, NNM and TNNM cost more time than other methods do because SVD on large matrices is quite time-consuming. The time complexity of DLMC is at least two times of that of AEMC because DLMC has more layers and involves pre-training and fine-tuning. The results in Table 7 confirms the analysis of computational complexity made in the last paragraph. Although AEMC and DLMC have much higher computational complexity than MF does, the matrix completion accuracies of AEMC and DLMC are significantly higher than that of MF, NNM, TNNM, and AECF.

## 5. Conclusion

In this paper, to solve the problem of matrix completion on data given by nonlinear latent variable model, AutoEncoder based matrix completion and deep learning based matrix completion were proposed. In addition, out-of-sample extensions of the methods

were also proposed to recover online incomplete data. Comparative studies were carried out in the tasks of synthetic matrix completion, single-image impainting, group-image inpainting, and collaborative filtering. Compared with the state-of-the-art methods, the proposed two methods, especially the deep learning based matrix completion, are able to provide significantly higher accuracies in matrix completion tasks.

## References


[1] E.J. Candes, B. Recht, Exact matrix completion via convex optimization, Found. Comput.Math. 9 (6) (2009) 717–772, doi:10.1007/s10208-009-9045-5.

[2] K.-C. Toh, S. Yun, An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems, Pac. J. Optim. 6 (3) (2010) 615–640.

[3] H. Wang, R. Zhao, Y. Cen, Rank adaptive atomic decomposition for low-rank matrix completion and its application on image recovery, Neurocomputing 145 (2014) 374–380.

[4] H. Ji, C. Liu, Z. Shen, Y. Xu, Robust video denoising using low rank matrix completion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, IEEE, 2010, pp. 1791–1798.

[5] J. Fan, T.W. Chow, Sparse subspace clustering for data with missing entries and high-rank matrix completion, Neural Netw. 93 (2017) 36–44. https://doi.org/10.1016/j.neunet.2017.04.005.

[6] Y. Hu, D. Zhang, J. Ye, X. Li, X. He, Fast and accurate matrix completion via truncated nuclear norm regularization, IEEE Trans. Pattern Anal. Mach. Intell. 35 (9) (2013) 2117–2130. http://doi.ieeecomputersociety.org/10.1109/TPAMI.2012.271.

[7] D.-Q. Chen, Y. Zhou, Inexact alternating direction method based on proximity projection operator for image inpainting in wavelet domain, Neurocomputing 189 (2016) 145–159.

[8] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Adv. Artif. Intell. 2009 (2009) 4:2–4:2, doi:10.1155/2009/421425.

[9] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009).

[10] G. Takács, I. Pilászy, B. Németh, D. Tikk, Scalable collaborative filtering approaches for large recommender systems, J. Mach. Learn. Res. 10 (2009) 623–656.

[11] Y. Luo, T. Liu, D. Tao, C. Xu, Multiview matrix completion for multilabel image classification, IEEE Trans. Image Process. 24 (8) (2015) 2355–2368, doi:10.1109/TIP.2015.2421309.

[12] N. Srebro, Learning with matrix factorizations, 2004 (Ph.D. thesis), Cambridge, MA, USA. AAI0807530.

[13] Y. Shen, Z. Wen, Y. Zhang, Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization, Optim. Methods Softw. 29 (2) (2014) 239–263, doi:10.1080/10556788.2012.700713.

[14] X. Luo, M. Zhou, Y. Xia, Q. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, IEEE Trans. Industr. Inform. 10 (2) (2014) 1273–1284.

[15] D. Meng, F. De La Torre, Robust matrix factorization with unknown noise, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1337–1344.

[16] X. Luo, M. Zhou, S. Li, Y. Xia, Z. You, Q. Zhu, H. Leung, An efficient second-order approach to factorize sparse matrices in recommender systems, IEEE Trans. Industr. Inform. 11 (4) (2015) 946–956.

[17] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, Q. Zhu, A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE Trans. Neural Netw. Learn. Syst. 27 (3) (2016) 579–592.

[18] X. Cao, Q. Zhao, D. Meng, Y. Chen, Z. Xu, Robust low-rank matrix factorization under general mixture noise distributions, IEEE Trans. Image Process. 25 (10) (2016) 4677–4690, doi:10.1109/TIP.2016.2593343.

[19] Z. Wen, W. Yin, Y. Zhang, Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, Math. Program. Comput. 4 (4) (2012) 333–361, doi:10.1007/s12532-012-0044-1.

[20] C. Chen, B. He, X. Yuan, Matrix completion via an alternating direction method, IMA J. Numer. Anal. 32 (1) (2011) 227–245, doi:10.1093/imanum/drq039.

[21] J.-F. Cai, E.J. Candés, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. Optim. 20 (4) (2010) 1956–1982, doi:10.1137/080738970.

[22] Z. Lin, M. Chen, Y. Ma, The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices, arXiv:1009.5055v3 [math.OC] (2010).

[23] F. Nie, H. Huang, C. Ding, Low-rank matrix recovery via efficient schatten p-norm minimization, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, in: AAAI'12, AAAI Press, 2012, pp. 655–661.

[24] C. Lu, C. Zhu, C. Xu, S. Yan, Z. Lin, Generalized singular value thresholding, Twenty-Ninth AAAI Conference on Artificial Intelligence (2015).

[25] C. Lu, J. Tang, S. Yan, Z. Lin, Generalized nonconvex nonsmooth low-rank minimization, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 4130–4137, doi:10.1109/CVPR.2014.526.

[26] E.J. Candes, X. Li, Y. Ma, J. Wright, Robust principal component analysis? J. ACM 58 (3) (2011) 1–37, doi:10.1145/1970392.1970395.

[27] B. Schlkopf, A. Smola, K.-R. Mller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319, doi:10.1162/089976698300017467.

[28] N. Lawrence, Probabilistic non-linear principal component analysis with gaussian process latent variable models, J. Mach. Learn. Res. 6 (2005) 1783–1816.

[29] X. Alameda-Pineda, E. Ricci, Y. Yan, N. Sebe, Recognizing emotions from abstract paintings using non-linear matrix completion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5240–5248.

[30] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann machines for collaborative filtering, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 791–798.

[31] Y. Ouyang, W. Liu, W. Rong, Z. Xiong, Autoencoder-based collaborative filtering, in: Proceedings of the International Conference on Neural Information Processing, Springer, 2014, pp. 284–291.

[32] S. Sedhain, A.K. Menon, S. Sanner, L. Xie, Autorec: autoencoders meet collaborative filtering, in: Proceedings of the 24th International Conference on World Wide Web, ACM, 2015, pp. 111–112.

[33] F. Strub, J. Mary, Collaborative filtering with stacked denoising autoencoders and sparse inputs, in: Proceedings of the NIPS Workshop on Machine Learning for eCommerce, 2015.

[34] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[35] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al., Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153.

[36] Y. Bengio, Learning deep architectures for ai, Found. Trends® Mach. Learn. 2 (1) (2009) 1–127.

[37] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.

[38] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (1) (2011) 1–122, doi:10.1561/2200000016.

[39] Y. LeCun, L. Bottou, G.B. Orr, K.R. Müller, Efficient BackProp, Springer, Berlin, Heidelberg, pp. 9–50. 10.1007/3-540-49430-8_2

[40] D.G. Luenberger, Introduction to linear and nonlinear programming, 28, Addison-Wesley Reading, MA, 1973.

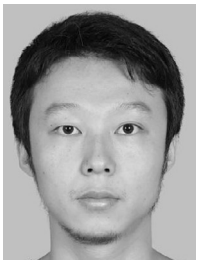
[41] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1–3) (1989) 503–528.

[42] Y. Shen, J. Li, Z. Zhu, W. Cao, Y. Song, Image reconstruction algorithm from compressed sensing measurements by dictionary learning, Neurocomputing 151, Part 3 (2015) 1153–1162.

[43] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324, doi:10.1109/5.726791.

[44] Jester jokes datasets, http://eigentaste.berkeley.edu/dataset/.

[45] Movielens 100k dataset, http://grouplens.org/datasets/movielens/100k/.

**Jicong Fan** received his B.E. and M.E. degrees in Automation and Control Science & Engineering, from Beijing University of Chemical Technology, Beijing, PR, China, in 2010 and 2013, respectively. From 2013 to 2015, he was a research assistant at the University of Hong Kong. Currently, he is working toward the Ph.D. degree at the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong S.A.R. His research interests include data mining and machine learning.

**Tommy W. S. Chow** received the B.Sc (First Hons.) and Ph.D. degrees from the University of Sunderland, Sunderland, UK. He joined the City University of Hong Kong, Hong Kong, as a Lecturer in 1988. He is currently a Professor in the Electronic Engineering Department. His research interests are in the area of Machine learning including Supervised and unsupervised learning, Data mining, Pattern recognition and fault diagnostic. He worked for NEI Reyrolle Technology at Hebburn, England developing digital simulator for transient network analyser. He then worked on a research project involving high current density current collection system for superconducting direct current machines, in collaboration with the Ministry of Defense (Navy) at Bath, England and the International Research and Development at Newcastle upon Tyne. He has authored or coauthored of over 200 technical papers in international journals, 5 book chapters, and over 60 technical papers in international conference proceedings.