Online high rank matrix completion

Jicong Fan, Madeleine Udell
Cornell University
Ithaca, NY 14853, USA
{jf577,udell}@cornell.edu

Abstract

Recent advances in matrix completion enable data imputation in full-rank matrices by exploiting low dimensional (nonlinear) latent structure. In this paper, we develop a new model for high rank matrix completion (HRMC), together with batch and online methods to fit the model and out-of-sample extension to complete new data. The method works by (implicitly) mapping the data into a high dimensional polynomial feature space using the kernel trick; importantly, the data occupies a low dimensional subspace in this feature space, even when the original data matrix is of full-rank. We introduce an explicit parametrization of this low dimensional subspace, and an online fitting procedure, to reduce computational complexity compared to the state of the art. The online method can also handle streaming or sequential data and adapt to non-stationary latent structure. We provide guidance on the sampling rate required these methods to succeed. Experimental results on synthetic data and motion capture data validate the performance of the proposed methods.

1. Introduction

In the past ten years, low rank matrix completion (LRMC) has been widely studied [4, 16, 22, 23, 20, 13, 3, 18, 9]. For instance, Candès and Recht [4] showed that any $n \times n$ incoherent matrices of rank r can be exactly recovered from $Cn^{1.2}r\log n$ uniformly randomly sampled entries with high probability through solving a convex problem of nuclear norm minimization (NNM). However, LRMC cannot recover high rank or full-rank matrices, even when the the data lies on a low dimensional (nonlinear) manifold. To address this problem, recently a few researchers have developed new high rank matrix completion (HRMC) methods [8, 17, 28] for data drawn from multiple subspaces [7, 6, 11] or nonlinear models [1, 24, 10]. These HRMC methods can outperform LRMC methods for many real problems such as subspace clustering with missing data, motion data recovery [6, 24], image inpainting, and classification [1, 10].

All the aforementioned LRMC and HRMC methods are offline methods. However, for many problems, we obtain one sample at a time and would like to update the model as each new sample arrives using online optimization. In addition, compared to offline methods, online methods [25, 21, 29] often have lower space and time complexities and can adapt to changes in the latent data structure. For these reasons, online matrix completion has recently gained increasing attention [2, 5, 15, 19].

2. Related work and our contribution

Online matrix completion. Sun and Luo [26] and Jin et al. [14] proposed to use stochastic gradient descent (SGD) to solve the low rank factorization (LRF) problem minimize $\sum_{(i,j)\in\Omega}\left(\boldsymbol{X}_{ij}-\boldsymbol{U}_{i:}\boldsymbol{V}_{j:}^\top\right)^2$ with variables $\boldsymbol{U}\in\mathbb{R}^{m\times r}$, $\boldsymbol{V}\in\mathbb{R}^{n\times r}$, where $\boldsymbol{X}\in\mathbb{R}^{m\times n}$and Ω denotes the locations of observed entries of $\boldsymbol{X}$. Specifically, given an entry $\boldsymbol{X}_{ij}$, the i-th row of $\boldsymbol{U}$ and j-th row of $\boldsymbol{V}$ are updated by gradient descent. Yun et al. [29] studied the streaming or online matrix completion problem when the columns of the matrix are presented sequentially. The GROUSE method proposed in [2] used incremental gradient descent on the Grassmannian manifold of subspaces to learn a low rank factorization from incomplete data online. These online methods have a common limitation: they cannot recover high rank matrices. Mairal et al. [21] also studied the online factorization problem with the goal of learning a dictionary for sparse coding: $\underset{\boldsymbol{D}\in\mathcal{C},\boldsymbol{\alpha}}{\text{minimize}}\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{D}\boldsymbol{\alpha}\|^2+\lambda\|\boldsymbol{\alpha}\|_1$. A sparse factorization based matrix completion algorithm was proposed in [11]. It is possible to recover a high rank matrix online by combining ideas from [21] with [11].

High rank matrix completion. Elhamifar [6] proposed to use group-sparse constraint to complete high rank matrix consisting of data drawn from union of low-dimensional subspaces. Alameda-Pineda et al. [1] proposed a nonlinear matrix completion method for classification. The method performs matrix completion on a matrix consisting of (nonlinear) feature-label pairs, where the unknown labels are re-

garded as missing entries. The method is inapplicable to general matrix completion problems in which the locations of all missing entries are not necessarily in a single block. Ongie et al. [24] assumed $\boldsymbol{X}$ is given by an algebraic variety and proposed a method called VMC to recover the missing entries of $\boldsymbol{X}$ through minimizing the rank of $\phi(\boldsymbol{X})$, where $\phi(\boldsymbol{X})$ is a feature matrix given by polynomial kernel. Fan and Chow [10] assumed the data are drawn from a nonlinear latent variable model and proposed a nonlinear matrix completion method (NLMC) that minimizes the rank of $\phi(\boldsymbol{X})$, where $\phi(\boldsymbol{X})$ is composed of high-dimensional nonlinear features induced by polynomial kernel or RBF kernel.

Challenges in HRMC. First, existing HRMC methods lack strong theoretical guarantee on the sample complexity required for recovery. For example, in VMC, the authors provide a lower bound of sampling rate (ρ_0, equation (6) of [24]) only for low-order polynomial kernel and ρ_0 involved an unknown parameter R owing to the algebraic variety assumption. In NLMC [10], the authors only provided a coarse lower bound of sampling rate, i.e. $\rho > O(d/m)$, where d is the dimension of latent variables. Second, existing HRMC methods are not scalable to large matrices. For example, VMC and NLMC require singular value decomposition on an $n \times n$ kernel matrix in every iteration. The method of [6] is also not efficient because of the sparse optimization on an $n \times n$ coefficients matrix. Third, existing HRMC methods have no out-of-sample extensions, which means they cannot efficiently complete new data. Last but not least, existing HRMC methods are offline methods and cannot handle online data.

Contributions. In this paper, we aim to address these challenges. We propose a novel high rank matrix completion method based on kernelized factorization (KFMC). KFMC is more efficient and accurate than state-of-the-art methods. Second, we propose an online version for KFMC, which can outperform online LRMC significantly. Third, we propose an out-of-sample extension for KFMC, which enables us to use the pre-learned high rank model to complete new data directly. Finally, we analyze the sampling rate required for KFMC to succeed.

3. Methodology

3.1. High rank matrices

We assume the columns of $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ are given by

$$\boldsymbol{x} = \mathbf{f}(\boldsymbol{s}) = [f_1(\boldsymbol{s}), f_2(\boldsymbol{s}), \cdots, f_m(\boldsymbol{s})]^\top, \tag{1}$$

where $\boldsymbol{s} \in \mathbb{R}^d$ ($d \ll m < n$) consists of uncorrelated variables and each $f_i : \mathbb{R}^d \to \mathbb{R}$, $i = 1, \ldots, m$, is a p-order polynomial with random coefficients. For example, when $d = 2$ and $p = 2$, for $i = 1, \ldots, m$, $x_i = \boldsymbol{c}_i^\top \bar{\boldsymbol{s}}$, where $\boldsymbol{c}_i \in \mathbb{R}^6$ and $\bar{\boldsymbol{s}} = [1, s_1, s_2, s_1^2, s_2^2, s_1 s_2]^\top$. Lemma 1 shows that $\boldsymbol{X}$ is of high rank when p is large.

Lemma 1. *Suppose the columns of $\boldsymbol{X}$ satisfy (1). Then with probability 1,* $\operatorname{rank}(\boldsymbol{X}) = \min\{m, n, \binom{d+p}{p}\}$.

Proof. Expand the polynomial f_i for each $i = 1, \ldots, m$ to write $x_i = f_i(\boldsymbol{s}) = \boldsymbol{c}_i^\top \bar{\boldsymbol{s}}$, where $\bar{\boldsymbol{s}} = \{s_1^{\mu_1} \cdots s_d^{\mu_d}\}_{|\mu|\le p}$ and $\boldsymbol{c}_i \in \mathbb{R}^{\binom{d+p}{p}}$. Each column of $\boldsymbol{X}$ satisfies $\boldsymbol{x} = \boldsymbol{C}\bar{\boldsymbol{s}}$, where $\boldsymbol{C} = [\boldsymbol{c}_1 \cdots \boldsymbol{c}_m] \in \mathbb{R}^{m\times\binom{d+p}{p}}$. The matrix $\boldsymbol{X}$ can be written as $\boldsymbol{X} = \boldsymbol{C}\bar{\boldsymbol{S}}$, where $\bar{\boldsymbol{S}} = (\bar{\boldsymbol{s}}_1^\top, \ldots, \bar{\boldsymbol{s}}_n^\top) \in \mathbb{R}^{\binom{d+p}{p}\times n}$. The variables $\boldsymbol{s}$ are uncorrelated and the coefficients $\boldsymbol{c}$ are random, so generically both $\boldsymbol{C}$ and $\bar{\boldsymbol{S}}$ are full rank. Hence $\operatorname{rank}(\boldsymbol{X}) = \min\{m, n, \binom{d+p}{p}\}$. □

In this paper, our goal is to recover $\boldsymbol{X}$ from a few randomly sampled entries we denote by $\{\boldsymbol{M}_{ij}\}_{(i,j)\in\Omega}$. When p is large, $\boldsymbol{X}$ is generically high rank and cannot be recovered by conventional LRMC methods.

Remark. Throughout this paper, we use the terms "low rank" or "high rank" matrix to mean a matrix whose rank is low or high *relative* to its side length.

Let $\phi : \mathbb{R}^m \to \mathbb{R}^{\bar{m}}$ be a q-order polynomial feature map $\phi(\boldsymbol{x}) = \{x_1^{\mu_1} \cdots x_m^{\mu_m}\}_{|\mu|\le q}$. Here $\bar{m} = \binom{m+q}{q}$. Write $\phi(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \cdots, \phi(\boldsymbol{x}_n)]$ and consider its rank:

Theorem 1. *Suppose the columns of $\boldsymbol{X}$ satisfy (1). Then with probability 1,* $\operatorname{rank}(\phi(\boldsymbol{X})) = \min\{\bar{m}, n, \binom{d+pq}{pq}\}$.

Proof. Define the pq-order polynomial map $\psi(\boldsymbol{s}) := \phi(\boldsymbol{x}) = \phi(\mathbf{f}(\boldsymbol{s}))$. Expanding as above, write the vector $\phi(\boldsymbol{x}) = \boldsymbol{\Psi}\tilde{s}$ with $\boldsymbol{\Psi} \in \mathbb{R}^{\bar{m}\times\binom{d+pq}{pq}}$ and $\tilde{\boldsymbol{s}} \in \mathbb{R}^{\binom{d+pq}{pq}}$, and write the matrix $\phi(\boldsymbol{X}) = \boldsymbol{\Psi}\tilde{\boldsymbol{S}}$ with $\tilde{\boldsymbol{S}} = (\tilde{\boldsymbol{s}}_1^\top, \ldots, \tilde{\boldsymbol{s}}_n^\top) \in \mathbb{R}^{\binom{d+pq}{pq}\times n}$. As above, $\boldsymbol{\Psi}$ and $\tilde{\boldsymbol{S}}$ are generically full rank, so $\operatorname{rank}(\phi(\boldsymbol{X})) = \min\{\bar{m}, n, \binom{d+pq}{pq}\}$ with probability 1. □

While $\operatorname{rank}(\phi(\boldsymbol{X})) \ge \operatorname{rank}(\boldsymbol{X})$, Theorem 1 shows that $\phi(\boldsymbol{X})$ is generically low rank when d is small and n is large. For example, when $d = 2$, $m = 20$, $n = 200$, $p = 4$, and $q = 2$, generically $\frac{\operatorname{rank}(\boldsymbol{X})}{\min\{m,n\}} = 0.75$ while $\frac{\operatorname{rank}(\phi(\boldsymbol{X}))}{\min\{\bar{m},n\}} = 0.225$: $\boldsymbol{X}$ is high rank but $\phi(\boldsymbol{X})$ is low rank.

3.2. Kernelized factorization

To recover the missing entries of $\boldsymbol{X}$, we propose to solve

$$\begin{array}{ll}\text{minimize}_{\boldsymbol{X},\boldsymbol{A},\boldsymbol{Z}} & \frac{1}{2}\|\phi(\boldsymbol{X}) - \boldsymbol{A}\boldsymbol{Z}\|_F^2 \\ \text{subject to} & \boldsymbol{X}_{ij} = \boldsymbol{M}_{ij},\ (i,j) \in \Omega,\end{array} \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{\bar{m}\times r}$, $\boldsymbol{Z} \in \mathbb{R}^{r\times n}$, and $r = \binom{d+pq}{pq}$. The solution to (2) completes the entries of $\boldsymbol{X}$ using the natural low rank structure of $\phi(\boldsymbol{X})$. Problem (2) implicitly defines an estimate for f, $\hat{\mathbf{f}}(\boldsymbol{S}) := \boldsymbol{X} \approx \phi^{-1}(\boldsymbol{A}\boldsymbol{Z})$.

For numerical stability, we regularize $\boldsymbol{A}$ and $\boldsymbol{Z}$ and solve

$$\begin{aligned} &\underset{\boldsymbol{X},\boldsymbol{A},\boldsymbol{Z}}{\text{minimize}}\ \tfrac{1}{2}\|\phi(\boldsymbol{X})-\boldsymbol{A}\boldsymbol{Z}\|_F^2+\tfrac{\alpha}{2}\|\boldsymbol{A}\|_F^2+\tfrac{\beta}{2}\|\boldsymbol{Z}\|_F^2, \\ &\text{subject to } \boldsymbol{X}_{ij}=\boldsymbol{M}_{ij},\ (i,j)\in\Omega, \end{aligned} \tag{3}$$

where α and β are regularization parameters, instead of (2).

It is possible to solve (3) directly but the computational cost is quite high if m and q are large. The following lemma shows that there is no need to model $\boldsymbol{A}$ explicitly.

Lemma 2. *For any $\boldsymbol{X}$ generated by (1), there exist $\boldsymbol{D}\in\mathbb{R}^{m\times r}$ and $\boldsymbol{Z}\in\mathbb{R}^{r\times n}$ such that $\phi(\boldsymbol{X})=\phi(\boldsymbol{D})\boldsymbol{Z}$.*

Proof. Suppose $\boldsymbol{D}\in\mathbb{R}^{m\times r}$ are also generated by (1) (e.g. any r columns of $\boldsymbol{X}$), so $\phi(\boldsymbol{D})$ and $\phi(\boldsymbol{X})$ share their column space and (with probability 1) $\phi(\boldsymbol{D})$ is full rank. More precisely, $\phi(\boldsymbol{D})=\boldsymbol{B}\boldsymbol{C}_{\boldsymbol{D}}$ and $\phi(\boldsymbol{X})=\boldsymbol{B}\boldsymbol{C}_{\boldsymbol{X}}$, where $\boldsymbol{B}\in\mathbb{R}^{\bar{m}\times r}$ is a basis for the column space, and both $\boldsymbol{C}_{\boldsymbol{X}}\in\mathbb{R}^{r\times n}$ and $\boldsymbol{C}_{\boldsymbol{D}}\in\mathbb{R}^{r\times r}$ are full rank. Define $\boldsymbol{Z}=\boldsymbol{C}_{\boldsymbol{D}}^{-1}\boldsymbol{C}_{\boldsymbol{X}}$ and the result follows. □

Hence any solution of the following also solves (3):

$$\begin{aligned} &\underset{\boldsymbol{X},\boldsymbol{D},\boldsymbol{Z}}{\text{minimize}}\ \tfrac{1}{2}\|\phi(\boldsymbol{X})-\phi(\boldsymbol{D})\boldsymbol{Z}\|_F^2+\tfrac{\alpha}{2}\|\phi(\boldsymbol{D})\|_F^2+\tfrac{\beta}{2}\|\boldsymbol{Z}\|_F^2 \\ &\text{subject to } \boldsymbol{X}_{ij}=\boldsymbol{M}_{ij},\ (i,j)\in\Omega, \end{aligned} \tag{4}$$

where $\boldsymbol{D}\in\mathbb{R}^{m\times r}$ is much smaller than $\boldsymbol{A}\in\mathbb{R}^{\bar{m}\times r}$ of (3). Use the trace function Tr to rewrite the objective in (4) as

$$\begin{aligned} &\tfrac{1}{2}\mathrm{Tr}\left(\phi(\boldsymbol{X})^\top\phi(\boldsymbol{X})-2\phi(\boldsymbol{X})^\top\phi(\boldsymbol{D})\boldsymbol{Z}+\boldsymbol{Z}^\top\phi(\boldsymbol{D})^\top\phi(\boldsymbol{D})\boldsymbol{Z}\right) \\ &+\tfrac{\alpha}{2}\mathrm{Tr}\left(\phi(\boldsymbol{D})^\top\phi(\boldsymbol{D})\right)+\tfrac{\beta}{2}\|\boldsymbol{Z}\|_F^2. \end{aligned}$$

Now we use the kernel trick to avoid explicitly computing the feature map ϕ. Define $k(\boldsymbol{x},\boldsymbol{y}):=\phi(\boldsymbol{x})^\top\phi(\boldsymbol{y})=\langle\phi(\boldsymbol{x}),\phi(\boldsymbol{y})\rangle$, so $\phi(\boldsymbol{X})^\top\phi(\boldsymbol{X})=\boldsymbol{K}_{\boldsymbol{XX}}$, $\phi(\boldsymbol{X})^\top\phi(\boldsymbol{D})=\boldsymbol{K}_{\boldsymbol{XD}}$, and $\phi(\boldsymbol{D})^\top\phi(\boldsymbol{D})=\boldsymbol{K}_{\boldsymbol{DD}}$, where $\boldsymbol{K}_{\boldsymbol{XX}}$, $\boldsymbol{K}_{\boldsymbol{XD}}$, and $\boldsymbol{K}_{\boldsymbol{DD}}$ are the corresponding kernel matrices. The most widely-used kernels are the polynomial kernel (Poly) and the radial basis function kernel (RBF)

$$\begin{aligned} &\text{Poly}:\ k(\boldsymbol{x},\boldsymbol{y})=(\boldsymbol{x}^\top\boldsymbol{y}+c)^q \\ &\text{RBF}:\ k(\boldsymbol{x},\boldsymbol{y})=\exp\left(-\tfrac{1}{\sigma^2}\|\boldsymbol{x}-\boldsymbol{y}\|^2\right), \end{aligned} \tag{5}$$

with hyperparameters c, q, and σ. The (implicit) feature maps $\phi(\boldsymbol{x})$ of Poly and RBF are the q-order and infinite-order polynomial maps respectively. Rewrite (4) to define kernelized factorizatiom matrix completion (KFMC)

$$\begin{aligned} &\underset{\boldsymbol{X},\boldsymbol{D},\boldsymbol{Z}}{\text{minimize}}\ \ell(\boldsymbol{Z},\boldsymbol{D},\boldsymbol{X}) \\ &\text{subject to } \boldsymbol{X}_{ij}=\boldsymbol{M}_{ij},\ (i,j)\in\Omega \end{aligned} \tag{KFMC}$$

where $\ell(\boldsymbol{Z},\boldsymbol{D},\boldsymbol{X})=\frac{1}{2}\mathrm{Tr}\left(\boldsymbol{K}_{\boldsymbol{XX}}-2\boldsymbol{K}_{\boldsymbol{XD}}\boldsymbol{Z}+\boldsymbol{Z}^\top\boldsymbol{K}_{\boldsymbol{DD}}\boldsymbol{Z}\right)+\frac{\alpha}{2}\mathrm{Tr}(\boldsymbol{K}_{\boldsymbol{DD}})+\frac{\beta}{2}\|\boldsymbol{Z}\|_F^2$. For the RBF kernel, $\mathrm{Tr}(\boldsymbol{K}_{\boldsymbol{DD}})\equiv r$ is a constant and can be dropped from the objective.

3.3. Optimization for KFMC

The optimization problem (KFMC) is nonconvex and has three blocks of variables. We propose using coordinate descent over these three blocks to find a stationary point.

Update $\boldsymbol{Z}$. To begin, complete entries of $\boldsymbol{X}$ arbitrarily and randomly initialize $\boldsymbol{D}$. Define the $r\times r$ identity $\boldsymbol{I}_r$. Fix $\boldsymbol{X}$ and $\boldsymbol{D}$ and update $\boldsymbol{Z}$ as

$$\begin{aligned} \boldsymbol{Z}&\leftarrow\arg\min_{\boldsymbol{Z}}\ell(\boldsymbol{Z},\boldsymbol{D},\boldsymbol{X}) \\ &=\arg\min_{\boldsymbol{Z}}-\mathrm{Tr}(\boldsymbol{K}_{\boldsymbol{XD}}\boldsymbol{Z})+\tfrac{1}{2}\mathrm{Tr}(\boldsymbol{Z}^\top\boldsymbol{K}_{\boldsymbol{DD}}\boldsymbol{Z})+\tfrac{\beta}{2}\|\boldsymbol{Z}\|_F^2 \\ &=(\boldsymbol{K}_{\boldsymbol{DD}}+\beta\boldsymbol{I}_r)^{-1}\boldsymbol{K}_{\boldsymbol{XD}}^\top, \end{aligned} \tag{6}$$

Update $\boldsymbol{D}$. There is no closed form solution for the minimization of $\ell(\boldsymbol{Z},\boldsymbol{D},\boldsymbol{X})$ with respect to $\boldsymbol{D}$ due to the kernel matrices. Instead, we propose the additive update $\boldsymbol{D}\leftarrow\boldsymbol{D}-\boldsymbol{\Delta}_{\boldsymbol{D}}$. We compute $\boldsymbol{\Delta}_{\boldsymbol{D}}$ using a relaxed Newton method, described below for the Poly and RBF kernels.

For the polynomial kernel, rewrite the terms in the objective in which $\boldsymbol{D}$ appears as

$$\begin{aligned} \ell(\boldsymbol{Z},\boldsymbol{D},\boldsymbol{X}):=&-\mathrm{Tr}((\boldsymbol{W}_1\odot(\boldsymbol{X}^T\boldsymbol{D}+c))\boldsymbol{Z}) \\ &+\tfrac{1}{2}\mathrm{Tr}(\boldsymbol{Z}^T(\boldsymbol{W}_2\odot(\boldsymbol{D}^T\boldsymbol{D}+c))\boldsymbol{Z}) \\ &+\tfrac{\alpha}{2}\mathrm{Tr}(\boldsymbol{W}_2\odot(\boldsymbol{D}^T\boldsymbol{D}+c)). \end{aligned} \tag{7}$$

defining $\boldsymbol{W}_1=\langle\boldsymbol{X}^\top\boldsymbol{D}+c\rangle^{q-1}$ and $\boldsymbol{W}_2=\langle\boldsymbol{D}^\top\boldsymbol{D}+c\rangle^{q-1}$, where $\odot$ is elementwise multiplication and $\langle\cdot\rangle^u$ denotes the element-wise u-power. Inspired by iteratively reweighted optimization, fix $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ to approximate the gradient and Hessian of $\ell(\boldsymbol{Z},\boldsymbol{D},\boldsymbol{X})$ with respect to $\boldsymbol{D}$ as

$$\begin{aligned} \boldsymbol{g}_{\boldsymbol{D}}&:=-\boldsymbol{X}(\boldsymbol{W}_1\odot\boldsymbol{Z}^\top)+\boldsymbol{D}((\boldsymbol{Z}\boldsymbol{Z}^\top+\alpha\boldsymbol{I}_r\odot\boldsymbol{W}_2)) \\ \boldsymbol{H}_{\boldsymbol{D}}&:=\boldsymbol{Z}\boldsymbol{Z}^\top\odot\boldsymbol{W}_2+\alpha\boldsymbol{W}_2\odot\boldsymbol{I}_r. \end{aligned}$$

$\boldsymbol{H}_{\boldsymbol{D}}$ is positive definite by the Schur product theorem. Now choose $\tau>1$ for numerical stability and define the update

$$\boldsymbol{\Delta}_{\boldsymbol{D}}:=\tfrac{1}{\tau}\boldsymbol{g}_{\boldsymbol{D}}\boldsymbol{H}_{\boldsymbol{D}}^{-1}. \tag{8}$$

The effectiveness of our update for $\boldsymbol{D}$ is guaranteed by the following lemma. (The proof of the lemma and discussion about the role of τ are in the supplementary material.)

Lemma 3. *The update (8) is a relaxed Newton's method and ensures sufficient decrease in the objective:*

$$\ell(\boldsymbol{Z},\boldsymbol{D}-\boldsymbol{\Delta}_{\boldsymbol{D}},\boldsymbol{X})-\ell(\boldsymbol{Z},\boldsymbol{D},\boldsymbol{X})\le-\tfrac{1}{2\tau}\mathrm{Tr}(\boldsymbol{g}_{\boldsymbol{D}}\boldsymbol{H}_{\boldsymbol{D}}^{-1}\boldsymbol{g}_{\boldsymbol{D}}^\top).$$

For the RBF kernel, the gradient is

$$\nabla_{\boldsymbol{D}}\ell=\tfrac{1}{\sigma^2}(\boldsymbol{X}\boldsymbol{Q}_1-\boldsymbol{D}\boldsymbol{\Gamma}_1)+\tfrac{2}{\sigma^2}(\boldsymbol{D}\boldsymbol{Q}_2-\boldsymbol{D}\boldsymbol{\Gamma}_2). \tag{9}$$

(Throughout, we abuse notation to write ℓ for $\ell(\boldsymbol{Z}, \boldsymbol{D}, \boldsymbol{X})$.) Here $\boldsymbol{Q}_1 = -\boldsymbol{Z}^\top \odot \boldsymbol{K}_{XD}$, $\boldsymbol{Q}_2 = (0.5\boldsymbol{Z}\boldsymbol{Z}^\top + 0.5\alpha \boldsymbol{I}_r) \odot \boldsymbol{K}_{DD}$, $\boldsymbol{\Gamma}_1 = \operatorname{diag}(\mathbf{1}_n^\top \boldsymbol{Q}_1)$, and $\boldsymbol{\Gamma}_2 = \operatorname{diag}(\mathbf{1}_r^\top \boldsymbol{Q}_2)$, where $\mathbf{1}_n \in \mathbb{R}^n$ and $\mathbf{1}_r \in \mathbb{R}^r$ are composed of 1s. The following lemma (proved in the supplementary material) indicates that $\boldsymbol{X}\boldsymbol{Q}_1$ in (9) is nearly a constant compared to $\boldsymbol{D}\boldsymbol{\Gamma}_1$, $\boldsymbol{D}\boldsymbol{Q}_2$, and $\boldsymbol{D}\boldsymbol{\Gamma}_2$, provided that σ and n are large enough:

Lemma 4. *$\|\boldsymbol{X}(\boldsymbol{Z}^\top \odot \boldsymbol{K}_{XD_1}) - \boldsymbol{X}(\boldsymbol{Z}^\top \odot \boldsymbol{K}_{XD_2})\|_F \leq \frac{c}{\sigma\sqrt{n}}\|\boldsymbol{X}\|_2 \|\boldsymbol{D}_1 - \boldsymbol{D}_2\|_F$, where c is a small constant.*

Therefore, we can compute an approximate Hessian neglecting $\boldsymbol{X}\boldsymbol{Q}_1$. As in (8), we define

$$\boldsymbol{\Delta}_D := \tfrac{1}{\tau}\nabla_D \ell(\tfrac{1}{\sigma^2}(2\boldsymbol{Q}_2 - \boldsymbol{\Gamma}_1 - 2\boldsymbol{\Gamma}_2))^{-1}. \tag{10}$$

Update $\boldsymbol{X}$. Finally, fixing $\boldsymbol{Z}$ and $\boldsymbol{D}$, we wish to minimize (KFMC) over $\boldsymbol{X}$, which again has no closed-form solution. Again, we suggest updating $\boldsymbol{X}$ using a relaxed Newton method $\boldsymbol{X} \leftarrow \boldsymbol{X} - \boldsymbol{\Delta}_X$. For the polynomial kernel,

$$\begin{aligned} \boldsymbol{g}_X &= \boldsymbol{X}(\boldsymbol{W}_3 \odot \boldsymbol{I}_n) - \boldsymbol{D}(\boldsymbol{W}_4^\top \odot \boldsymbol{Z}) \\ &= q\boldsymbol{X} \odot (\mathbf{1}_m \boldsymbol{w}^\top) - q\boldsymbol{D}(\boldsymbol{W}_4^\top \odot \boldsymbol{Z}), \end{aligned} \tag{11}$$

where $\boldsymbol{W}_3 = \langle \boldsymbol{X}^\top \boldsymbol{X} + c \rangle^{q-1}$, $\boldsymbol{W}_4 = \langle \boldsymbol{X}^\top \boldsymbol{D} + c \rangle^{q-1}$, $\mathbf{1}_m \in \mathbb{R}^m$ consists of 1s, and $\boldsymbol{w} \in \mathbb{R}^m$ consists of the diagonal entries of $\boldsymbol{W}_3$. As above, we define

$$\boldsymbol{\Delta}_X := \tfrac{1}{\tau}\boldsymbol{g}_X \odot (\mathbf{1}_m \boldsymbol{w}^{-T}). \tag{12}$$

When RBF kernel is used, we get

$$\nabla_X \ell = \tfrac{1}{\sigma^2}(\boldsymbol{D}\boldsymbol{Q}_3 - \boldsymbol{X}\boldsymbol{\Gamma}_3) + \tfrac{2}{\sigma^2}(\boldsymbol{X}\boldsymbol{Q}_4 - \boldsymbol{X}\boldsymbol{\Gamma}_4). \tag{13}$$

Here $\boldsymbol{Q}_3 = -\boldsymbol{Z} \odot \boldsymbol{K}_{XD}^\top$, $\boldsymbol{Q}_4 = 0.5\boldsymbol{I}_n \odot \boldsymbol{K}_{XX}$, $\boldsymbol{\Gamma}_3 = \operatorname{diag}(\mathbf{1}_r^\top \boldsymbol{Q}_3)$, and $\boldsymbol{\Gamma}_4 = \operatorname{diag}(\mathbf{1}_n^\top \boldsymbol{Q}_4)$. As in (10), define

$$\boldsymbol{\Delta}_X := \tfrac{1}{\tau}\nabla_X \ell(\tfrac{1}{\sigma^2}(2\boldsymbol{Q}_4 - \boldsymbol{\Gamma}_3 - 2\boldsymbol{\Gamma}_4))^{-1}. \tag{14}$$

Here the computational cost is not high in practice because the matrix to be inverted is diagonal.

We can also use a momentum update to accelerate the convergence of $\boldsymbol{D}$ and $\boldsymbol{X}$:

$$\begin{cases} \widehat{\boldsymbol{\Delta}}_D \leftarrow \eta\widehat{\boldsymbol{\Delta}}_D + \boldsymbol{\Delta}_D, \ \boldsymbol{D} \leftarrow \boldsymbol{D} - \widehat{\boldsymbol{\Delta}}_D \\ \widehat{\boldsymbol{\Delta}}_X \leftarrow \eta\widehat{\boldsymbol{\Delta}}_X + \boldsymbol{\Delta}_X, \ \boldsymbol{X} \leftarrow \boldsymbol{X} - \widehat{\boldsymbol{\Delta}}_X \end{cases} \tag{15}$$

where $0 < \eta < 1$ is a constant. The optimization method is summarized as Algorithm 1. The following lemma (with proof in the supplement) shows the method converges.

Lemma 5. *For sufficiently small η, Algorithm 1 converges to a stationary point.*

Algorithm 1 Offline KFMC

Input: $\boldsymbol{M}$, Ω, r, $k(\cdot,\cdot)$, α, β, $t_{\max}$, η

Initialize: $t = 0$, $\boldsymbol{X}$, $\boldsymbol{D} \sim \mathcal{N}(0,1)$, $\widehat{\boldsymbol{\Delta}}_D = \mathbf{0}$, $\widehat{\boldsymbol{\Delta}}_X = \mathbf{0}$
repeat
  $t \leftarrow t + 1$
  $\boldsymbol{Z} = (\boldsymbol{K}_{DD} + \beta \boldsymbol{I}_r)^{-1} \boldsymbol{K}_{XD}^\top$
  Compute $\boldsymbol{\Delta}_D$ using (8) or (10)
  $\widehat{\boldsymbol{\Delta}}_D = \eta\widehat{\boldsymbol{\Delta}}_D + \boldsymbol{\Delta}_D$
  $\boldsymbol{D} \leftarrow \boldsymbol{D} - \widehat{\boldsymbol{\Delta}}_D$
  Compute $\boldsymbol{\Delta}_X$ using (12) or (14)
  $\widehat{\boldsymbol{\Delta}}_X = \eta\widehat{\boldsymbol{\Delta}}_X + \boldsymbol{\Delta}_X$
  $\boldsymbol{X} \leftarrow \boldsymbol{X} - \widehat{\boldsymbol{\Delta}}_X$ and $\boldsymbol{X}_{ij} = \boldsymbol{M}_{ij}\ \forall (i,j) \in \Omega$
until converged or $t = t_{\max}$

Output: $\boldsymbol{X}$, $\boldsymbol{D}$

3.4. Online KFMC

Suppose we get an incomplete sample $\boldsymbol{x}$ at time t and need to update the model of matrix completion timely or solve the optimization online. In (4), we can put the constraint into the objective function directly and get the following equivalent problem

$$\underset{[\boldsymbol{X}]_{\bar{\Omega}}, \boldsymbol{D}, \boldsymbol{Z}}{\text{minimize}} \sum_{j=1}^{n} \tfrac{1}{2}\|\phi(\boldsymbol{x}_j) - \phi(\boldsymbol{D})\boldsymbol{z}_j\|^2 + \tfrac{\alpha}{2n}\|\phi(\boldsymbol{D})\|_F^2 + \tfrac{\beta}{2}\|\boldsymbol{z}_j\|^2, \tag{16}$$

where $[\boldsymbol{X}]_{\bar{\Omega}}$ denotes the unknown entries of $\boldsymbol{X}$. Denote

$$\begin{aligned} \ell([\boldsymbol{x}_j]_{\omega_j}, \boldsymbol{D}) := \min_{\boldsymbol{z}_j, [\boldsymbol{x}_j]_{\bar{\omega}_j}} & \tfrac{1}{2}\|\phi(\boldsymbol{x}_j) - \phi(\boldsymbol{D})\boldsymbol{z}_j\|^2 \\ & + \tfrac{\alpha}{2n}\|\phi(\boldsymbol{D})\|_F^2 + \tfrac{\beta}{2}\|\boldsymbol{z}_j\|^2, \end{aligned} \tag{17}$$

where $[\boldsymbol{x}_j]_{\omega_j}$ ($[\boldsymbol{x}_j]_{\bar{\omega}_j}$) denotes the observed (unknown) entries of $\boldsymbol{x}_j$ and ω_j ($\bar{\omega}_j$) denotes the corresponding locations. Then (16) minimizes the empirical cost function

$$g_n(\boldsymbol{D}) := \frac{1}{n}\sum_{j=1}^{n} \ell([\boldsymbol{x}_j]_{\omega_j}, \boldsymbol{D}). \tag{18}$$

The expected cost is

$$g(\boldsymbol{D}) := \mathbb{E}_{[\boldsymbol{x}]_\omega}\left[\ell([\boldsymbol{x}]_\omega, \boldsymbol{D})\right] = \lim_{n\to\infty} g_n(\boldsymbol{D}). \tag{19}$$

To approximately minimize (19) online, we propose the following optimization for a given incomplete sample $\boldsymbol{x}$

$$\begin{aligned} \underset{[\boldsymbol{x}]_{\bar{\omega}}, \boldsymbol{D}, \boldsymbol{z}}{\text{minimize}}\ \hat{\ell}(\boldsymbol{z}, [\boldsymbol{x}]_{\bar{\omega}}, \boldsymbol{D}) := & \tfrac{1}{2}\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{D})\boldsymbol{z}\|^2 \\ & + \tfrac{\alpha}{2}\|\phi(\boldsymbol{D})\|_F^2 + \tfrac{\beta}{2}\|\boldsymbol{z}\|^2. \end{aligned} \tag{20}$$

With randomly initialized $\boldsymbol{D}$, we first compute $\boldsymbol{z}$ and $[\boldsymbol{x}]_{\bar{\omega}}$ via alternately minimizing $\hat{\ell}(\boldsymbol{z}, [\boldsymbol{x}]_{\bar{\omega}}, \boldsymbol{D})$, which is equivalent to

$$\underset{[\boldsymbol{x}]_{\bar{\omega}}, \boldsymbol{z}}{\text{minimize}}\ \tfrac{1}{2}k_{\boldsymbol{x}\boldsymbol{x}} - \boldsymbol{k}_{\boldsymbol{x}D}\boldsymbol{z} + \tfrac{1}{2}\boldsymbol{z}^\top \boldsymbol{K}_{DD}\boldsymbol{z} + \tfrac{\beta}{2}\|\boldsymbol{z}\|^2. \tag{21}$$

Specifically, in each iteration, $\boldsymbol{z}$ is updated as

$$\boldsymbol{z} = (\boldsymbol{K}_{DD} + \beta \boldsymbol{I}_r)^{-1} \boldsymbol{k}_{x\boldsymbol{D}}^\top. \tag{22}$$

We propose to update $[\boldsymbol{x}]_{\bar{\omega}}$ by Newton's method, i.e., $[\boldsymbol{x}]_{\bar{\omega}} \leftarrow [\boldsymbol{x}]_{\bar{\omega}} - [\boldsymbol{\Delta}_x]_{\bar{\omega}}$. When polynomial kernel is used, we obtain

$$\nabla_x \hat{\ell} = w_1 \boldsymbol{x} - \boldsymbol{D}(\boldsymbol{w}_2^\top \odot \boldsymbol{z}) \tag{23}$$

where $w_1 = \langle \boldsymbol{x}^\top \boldsymbol{x} + c \rangle^{q-1}$, $\boldsymbol{w}_2 = \langle \boldsymbol{x}^\top \boldsymbol{D} + c \rangle^{q-1}$. Then

$$\boldsymbol{\Delta}_x = \tfrac{1}{\tau w_1} \nabla_x \hat{\ell}. \tag{24}$$

When RBF kernel is used, we have

$$\nabla_x \hat{\ell} = \tfrac{1}{\sigma^2}(\boldsymbol{D}\boldsymbol{q} - \gamma \boldsymbol{x}), \tag{25}$$

where $\boldsymbol{q} = -\boldsymbol{z} \odot \boldsymbol{k}_{x\boldsymbol{D}}^\top$ and $\gamma = \boldsymbol{1}_r^\top \boldsymbol{q}$. Then

$$\boldsymbol{\Delta}_x = \tfrac{\sigma^2}{\tau\gamma} \nabla_x \hat{\ell}. \tag{26}$$

The derivations of (24) and (26) are similar to those of (8) and (10). Then we repeat (22)–(26) until converged.

After $\boldsymbol{z}$ and $[\boldsymbol{x}]_{\bar{\omega}}$ are computed, we compute $\boldsymbol{D}$ via minimizing $\hat{\ell}(\boldsymbol{z}, [\boldsymbol{x}]_{\bar{\omega}}, \boldsymbol{D})$, which is equivalent to

$$\underset{\boldsymbol{D}}{\text{minimize}} \; -\boldsymbol{k}_{x\boldsymbol{D}}\boldsymbol{z} + \tfrac{1}{2}\boldsymbol{z}^\top \boldsymbol{K}_{DD}\boldsymbol{z} + \tfrac{\alpha}{2}\mathrm{Tr}(\boldsymbol{K}_{DD}). \tag{27}$$

We propose to use SGD to update $\boldsymbol{D}$, i.e., $\boldsymbol{D} \leftarrow \boldsymbol{D} - \boldsymbol{\Delta}_{\boldsymbol{D}}$. When polynomial kernel is used, we have

$$\nabla_{\boldsymbol{D}} \hat{\ell} = -\boldsymbol{x}(\boldsymbol{w}_1 \odot \boldsymbol{z}^\top) + \boldsymbol{D}(\boldsymbol{z}\boldsymbol{z}^\top \odot \boldsymbol{W}_2) + \alpha \boldsymbol{D}(\boldsymbol{W}_2 \odot \boldsymbol{I}_r)), \tag{28}$$

where $\boldsymbol{w}_1 = \langle \boldsymbol{x}^\top \boldsymbol{D} + c \rangle^{q-1}$ and $\boldsymbol{W}_2 = \langle \boldsymbol{D}^\top \boldsymbol{D} + c \rangle^{q-1}$. Then we have

$$\boldsymbol{\Delta}_{\boldsymbol{D}} = \tfrac{1}{\tau} \nabla_{\boldsymbol{D}} \hat{\ell} / \| \boldsymbol{z}\boldsymbol{z}^\top \odot \boldsymbol{W}_2 + \alpha \boldsymbol{W}_2 \odot \boldsymbol{I}_r \|_2. \tag{29}$$

Here we cannot use the method of (8) because $\boldsymbol{z}\boldsymbol{z}^\top$ is not as stable as $\boldsymbol{Z}\boldsymbol{Z}^\top$. In addition, the following lemma (proved in the supplementary material) ensures the effectiveness of updating $\boldsymbol{D}$:

Lemma 6. *Updating $\boldsymbol{D}$ as $\boldsymbol{D} - \boldsymbol{\Delta}_{\boldsymbol{D}}$ does not diverge and $\hat{\ell}(\boldsymbol{z}, [\boldsymbol{x}]_{\bar{\omega}}, \boldsymbol{D} - \boldsymbol{\Delta}_{\boldsymbol{D}}) - \hat{\ell}(\boldsymbol{z}, [\boldsymbol{x}]_{\bar{\omega}}, \boldsymbol{D}) \le -\frac{1}{2\tau\tau_0} \| \nabla_{\boldsymbol{D}} \hat{\ell} \|_F^2$ provided that $\tau > 1$, where $\tau_0 = \| \boldsymbol{z}\boldsymbol{z}^\top \odot \boldsymbol{W}_2 + \alpha \boldsymbol{W}_2 \odot \boldsymbol{I}_r \|_2$.*

When RBF kernel is used, the derivative is

$$\nabla_{\boldsymbol{D}} \hat{\ell} = \tfrac{1}{\sigma^2}(\boldsymbol{x}\boldsymbol{Q}_1 - \boldsymbol{D}\boldsymbol{\Gamma}_1) + \tfrac{2}{\sigma^2}(\boldsymbol{D}\boldsymbol{Q}_2 - \boldsymbol{D}\boldsymbol{\Gamma}_2), \tag{30}$$

where $\boldsymbol{Q}_1 = -\boldsymbol{z}^\top \odot \boldsymbol{k}_{X\boldsymbol{D}}$, $\boldsymbol{Q}_2 = (0.5\boldsymbol{z}\boldsymbol{z}^\top + 0.5\alpha \boldsymbol{I}_r) \odot \boldsymbol{K}_{DD}$, $\boldsymbol{\Gamma}_1 = \mathrm{diag}(\boldsymbol{Q}_1)$, and $\boldsymbol{\Gamma}_2 = \mathrm{diag}(\boldsymbol{1}_r^\top \boldsymbol{Q}_2)$. Similar to (29) and Lemma 6, we obtain

$$\boldsymbol{\Delta}_{\boldsymbol{D}} = \tfrac{1}{\tau} \nabla_{\boldsymbol{D}} \hat{\ell} / \| \tfrac{1}{\sigma^2}(2\boldsymbol{Q}_2 - \boldsymbol{\Gamma}_1 - 2\boldsymbol{\Gamma}_2) \|_2. \tag{31}$$

Similar to offline KFMC, we also use momentum to accelerate the optimization of online KFMC. The optimization steps are summarized in Algorithm 2. The error of online matrix completion can be reduced with multi-pass optimization or increasing the number of samples. In Algorithm 2, the sequence $\ell([\boldsymbol{x}_t]_{\omega_t}, \boldsymbol{D})$ defined in (17) may not decrease continuously because the incomplete sample $\boldsymbol{x}_t$ can introduce high uncertainty. However, the sequence $g_t(\boldsymbol{D})$, the empirical cost function defined in (18), is convergent because for $j = 1, \cdots, t$, $\ell([\boldsymbol{x}_j]_{\omega_j}, \boldsymbol{D})$ is minimized through optimizing $[\boldsymbol{x}_j]_{\bar{\omega}_j}$, $\boldsymbol{z}_j$, and $\boldsymbol{D}$.

Algorithm 2 Online KFMC

Input: Incomplete samples $\{[\boldsymbol{x}_1]_{\omega_1}, [\boldsymbol{x}_2]_{\omega_2}, \cdots, [\boldsymbol{x}_t]_{\omega_t}\}$, r, $k(\cdot,\cdot)$, α, β, n_{iter}, η, n_{pass}

Initialize: $\boldsymbol{D} \sim \mathcal{N}(0,1)$, $\widehat{\boldsymbol{\Delta}}_{\boldsymbol{D}} = \boldsymbol{0}$
for $u = 1$ to n_{pass} **do**
  for $j = 1$ to t **do**
    $l = 0$, $\widehat{\boldsymbol{\Delta}}_{\boldsymbol{X}} = \boldsymbol{0}$, $\boldsymbol{C} = (\boldsymbol{K}_{DD} + \beta \boldsymbol{I}_r)^{-1}$
    repeat
      $l \leftarrow l + 1$ and $\boldsymbol{z}_j = \boldsymbol{C}\boldsymbol{k}_{\boldsymbol{X}\boldsymbol{D}}^\top$
      Compute $\boldsymbol{\Delta}_x$ using (24) or (26)
      $\widehat{\boldsymbol{\Delta}}_x \leftarrow \eta \widehat{\boldsymbol{\Delta}}_x + \boldsymbol{\Delta}_x$
      $[\boldsymbol{x}_j]_{\bar{\omega}_j} \leftarrow [\boldsymbol{x}_j]_{\bar{\omega}_j} - [\widehat{\boldsymbol{\Delta}}_x]_{\bar{\omega}_j}$
    until converged or $l = n_{\text{iter}}$
    Compute $\boldsymbol{\Delta}_{\boldsymbol{D}}$ using (29) or (31)
    $\widehat{\boldsymbol{\Delta}}_{\boldsymbol{D}} \leftarrow \eta \widehat{\boldsymbol{\Delta}}_{\boldsymbol{D}} + \boldsymbol{\Delta}_{\boldsymbol{D}}$ and $\boldsymbol{D} \leftarrow \boldsymbol{D} - \widehat{\boldsymbol{\Delta}}_{\boldsymbol{D}}$
  end for
end for

Output: $\boldsymbol{X}_t = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t]$, $\boldsymbol{D}$

3.5. Out-of-sample extension of KFMC

The matrix $\boldsymbol{D}$ obtained from offline matrix completion (1) or online matrix completion (2) can be used to recover the missing entries of new data without updating the model. We can also compute $\boldsymbol{D}$ from complete training data: the corresponding algorithm is similar to Algorithms 1 and 2, but does not require the $\boldsymbol{X}$ update. We can complete a new (incomplete) sample $\boldsymbol{x}_{\text{new}}$ by solving

$$\underset{[\boldsymbol{x}_{\text{new}}]_{\bar{\omega}_{\text{new}}}, \boldsymbol{z}_{\text{new}}}{\text{minimize}} \; \tfrac{1}{2}\|\phi(\boldsymbol{x}_{\text{new}}) - \phi(\boldsymbol{D})\boldsymbol{z}_{\text{new}}\|^2 + \tfrac{\beta}{2}\|\boldsymbol{z}_{\text{new}}\|^2, \tag{32}$$

where $[\boldsymbol{x}_{\text{new}}]_{\bar{\omega}_{\text{new}}}$ denotes unknown entries of $\boldsymbol{x}_{\text{new}}$. This out-of-sample extension of KFMC is displayed as Algorithm 3.

3.6. Complexity analysis

Consider a high (even, full) rank matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ ($m \ll n$) given by (1). In the methods VMC and NLMC, and our KFMC, the largest object stored is the kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$. Hence their space complexities are all $O(n^2)$. In VMC and NLMC, the major computational step is to

Algorithm 3 Out-of-sample extension for KFMC

Input: $\boldsymbol{D}$ (computed from training data), $k(\cdot,\cdot)$, β, n_{iter}, η, new incomplete samples $\{[\boldsymbol{x}_1]_{\omega_1}, [\boldsymbol{x}_2]_{\omega_2}, \cdots, [\boldsymbol{x}_t]_{\omega_t}\}$

$\boldsymbol{C} = (\boldsymbol{K}_{DD} + \beta \boldsymbol{I}_r)^{-1}$
for $j = 1$ to t **do**
  $l = 0$, $\widehat{\boldsymbol{\Delta}}_x = \boldsymbol{0}$
  repeat
    $l \leftarrow l + 1$ and $\boldsymbol{z}_j = \boldsymbol{C}\boldsymbol{k}_{x\boldsymbol{D}}^\top$
    Compute $\boldsymbol{\Delta}_x$ using (24) or (26)
    $\widehat{\boldsymbol{\Delta}}_x \leftarrow \eta \widehat{\boldsymbol{\Delta}}_x + \boldsymbol{\Delta}_x$
    $[\boldsymbol{x}_j]_{\bar{\omega}_j} \leftarrow [\boldsymbol{x}_j]_{\bar{\omega}_j} - [\widehat{\boldsymbol{\Delta}}_x]_{\bar{\omega}_j}$
  until converged or $l = n_{\text{iter}}$
end for

Output: $\boldsymbol{X}_{\text{new}} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t]$

	Space complexity	Time complexity
VMC	$O(n^2)$	$O(n^3 + mn^2)$
NLMC	$O(n^2)$	$O(n^3 + mn^2)$
KFMC	$O(n^2)$	$O(mn^2 + rmn)$
OL-KFMC	$O(mr + r^2)$	$O(r^3)$
OSE-KFMC	$O(mr + r^2)$	$O(mr)$

Table 1: Time and space complexities ($\boldsymbol{X} \in \mathbb{R}^{m\times n}$, $m \ll n$)

compute $\boldsymbol{K}$ and its singular value decomposition at every iteration. Hence their time complexities are $O(mn^2 + n^3)$. In our KFMC, the major computational steps are to form $\boldsymbol{K}$, to invert the $r \times r$ matrix in (6), and to multiply an $m \times n$ and $n \times r$ matrix to compute the derivatives. Hence the time complexity is $O(mn^2 + r^3 + rmn) = O(mn^2 + rmn)$, since $n \gg r$.

Online KFMC does not store the kernel matrix $\boldsymbol{K}$. Instead, the largest objects stored are $\boldsymbol{D}$ and $\boldsymbol{K}_{DD}$. Hence the space complexity is $O(mr + r^2)$. The major computational step is to invert an $r \times r$ matrix (see Algorithm 2). Thus the time complexity is $O(r^3)$. In the out-of-sample extension, the largest objects stored are $\boldsymbol{D}$ and $\boldsymbol{C}$ (see Algorithm 3), so the space complexity is $O(mr + r^2)$. For each online sample, we only need to multiply $m \times r$ matrices with vectors. Hence the time complexity is just $O(mr)$.

This analysis are summarized in Table 1. We see that the space and time complexities of the proposed three approaches are much lower than those of VMC and NLMC.

3.7. Generalization for union of subspaces

KFMC can also handle data drawn from a union of subspaces. Suppose the columns of $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ are given by

$$\{\boldsymbol{x}^{\{k\}} = \mathbf{f}^{\{k\}}(\boldsymbol{s}^{\{k\}})\}_{k=1}^{u}, \tag{33}$$

where $\boldsymbol{s}^{\{k\}} \in \mathbb{R}^d$ ($d \ll m < n$) are random variables and $\mathbf{f}^{\{k\}} : \mathbb{R}^d \to \mathbb{R}^m$ are p-order polynomial functions for each $k = 1, \ldots, u$. For convenience, we write

$$\boldsymbol{X} = [\boldsymbol{X}^{\{1\}}, \boldsymbol{X}^{\{2\}}, \cdots, \boldsymbol{X}^{\{u\}}], \tag{34}$$

where the columns of each $\boldsymbol{X}^{\{k\}}$ are in the range of $\mathbf{f}^{\{k\}}$, though we do not know which subspace each column of $\boldsymbol{X}$ is drawn from. An argument similar to Lemma 1 shows

$$\text{rank}(\boldsymbol{X}) = \min\{m, n, u\tbinom{d+p}{p}\} \tag{35}$$

with probability 1, so $\boldsymbol{X}$ is very likely to be of high rank or full rank when u or p is large.

We can generalize Theorem 1 to show $\text{rank}(\phi(\boldsymbol{X})) = \min\{\bar{m}, n, r\}$ with probability 1, where $\bar{m} = \binom{m+q}{q}$ and $r = u\binom{d+pq}{pq}$. Hence when d is small and n is large, $\phi(\boldsymbol{X})$ is low rank, so missing entries of $\boldsymbol{X}$ can still be recovered by the offline and online methods proposed in this paper. In particular, for data drawn from a union of linear subspaces ($p = 1$ and $u > 1$), generically $\text{rank}(\boldsymbol{X}) = \min(m, n, ud)$ while $\text{rank}(\phi(\boldsymbol{X})) = u\binom{d+q}{q}$.

3.8. On the sampling rate

Suppose $\boldsymbol{X}$ is generated by (33), and a proportion ρ_{KFMC} of its entries are observed. We provide some heuristics to help decide how many entries should be observed for completion with the polynomial and RBF kernels. Detailed calculations for (36) and (37) are deferred to the supplement.

To complete $\phi(\boldsymbol{X})$ uniquely using a q-order polynomial kernel, one rule of thumb is that the number of entries observed should be at least as large as the number of degrees of freedom in the matrix $\phi(\boldsymbol{X})$ [24]. Here, $\phi(\boldsymbol{X})$ is a $\bar{m} \times n$ matrix with rank $r = u\binom{d+pq}{pq}$, where $\bar{m} = \binom{m+q}{q}$. We count the degrees of freedom of matrices with this rank to argue sampling rate should satisfy

$$\rho_{\text{KFMC}} \geq \left(r/n + r/\bar{m} - r^2/n/\bar{m}\right)^{1/q}. \tag{36}$$

Equation (36) bounds the number of degrees of freedom of $\phi(\boldsymbol{X})$ by considering its rank and size. But of course $\phi(\boldsymbol{X})$ is a deterministic function of $\boldsymbol{X}$, which has many fewer degrees of freedom. Hence while (36) provides a good rule of thumb, we can still hope that lower sampling rates might produce sensible results.

For the RBF kernel, $q = \infty$, so the condition (36) is vacuous. However, the RBF kernel can be well approximated by a polynomial kernel and we have

$$\phi_i(\boldsymbol{x}) = \hat{\phi}_i(\boldsymbol{x}) + O(\sqrt{\tfrac{c^{q+1}}{(q+1)!}}), \tag{37}$$

where $\hat{\phi}_i(\boldsymbol{x})$ denotes the i-th feature of q-order polynomial kernel and $\phi_i(\boldsymbol{x})$ denotes the i-th feature of the RBF kernel. Hence exact recovery of $\hat{\phi}_i(\boldsymbol{x})$ implies approximate recovery of $\phi_i(\boldsymbol{x})$ with error $O(\sqrt{\frac{c^{q+1}}{(q+1)!}})$. This argument

provides the intuition that the RBF kernel should recover the low-order ($\leq q$) features of $\phi(\boldsymbol{x})$ with error $O(\sqrt{\frac{c^{q+1}}{(q+1)!}})$ provided that (36) holds. Of course, we can identify missing entries of $\boldsymbol{X}$ by considering the first block of the completed matrix $\phi(\boldsymbol{X})$.

In experiments, we observe that the RBF kernel often works better than polynomial kernel. We hypothesize two reasons for the effectiveness of the RBF kernel: 1) It captures the higher-order features in $\phi(\boldsymbol{x})$, which could be useful when n is very large 2) It is easier to analyze and to optimize, speeding convergence.

Low rank matrix completion methods can only uniquely complete a matrix given a sampling rate that satisfies

$$\rho_{\text{LRMC}} > \big((m+n)r_{\boldsymbol{X}} - r_{\boldsymbol{X}}^2\big)/(mn), \tag{38}$$

where $r_{\boldsymbol{X}} = \min\{m, n, u\binom{d+p}{p}\}$. This bound can be vacuous (larger than 1) if u or p are large. In contrast, ρ_{KFMC} given by (36) can still be smaller than 1 in the same regime, provided that n is large enough. For example, when $m = 20$, $d = 2$, $p = 2$, and $u = 3$, we have $\rho_{\text{LRMC}} > 0.91$. Let $q = 2$ and $n = 300$, we have $\rho_{\text{KFMC}} > 0.56$. If $p = 1$ and $u = 10$, we have $\rho_{\text{LRMC}} > 1$ and $\rho_{\text{KFMC}} > 0.64$. This calculation provides further intuition for how our methods can recover high rank matrices while classical low rank matrix completion methods fail.

3.9. Analytic functions and smooth functions

Hitherto, we have assumed that $\mathbf{f}$ is a finite order polynomial function. However, our methos also work when $\mathbf{f}$ is an analytic or smooth function. Analytic functions are well approximated by polynomials. Furthermore, smooth functions can be well approximated by polynomial functions at least on intervals. Hence for a smooth function $\mathbf{h} : \mathbb{R}^d \to \mathbb{R}^m$, we consider the generative model

$$\boldsymbol{x} = \mathbf{h}(\boldsymbol{s}) = \mathbf{f}(\boldsymbol{s}) + \boldsymbol{e} \tag{39}$$

where $\mathbf{f}$ is a p-order Taylor expansion of $\mathbf{h}$ and $\boldsymbol{e} \in \mathbb{R}^m$ denotes the residual, which scales as $\boldsymbol{e} \sim O(\frac{c}{(p+1)!})$ where c is the $p+1$th derivative of $\mathbf{h}$.

We see that the error $\boldsymbol{e}$ from our polynomial model decreases as p increases. To fit a model with larger p, the bound (36) suggests we need more samples n. We conjecture that for any smooth $\mathbf{h}$, it is possible to recover the missing entries with arbitrarily low error provided n is sufficiently large.

4. Experiments

4.1. Synthetic data

We generate the columns of $\boldsymbol{X}$ by $\boldsymbol{x} = \mathbf{f}(\boldsymbol{s})$ where $\boldsymbol{s} \sim \mathcal{U}(0,1)$ and $\mathbf{f} : \mathbb{R}^3 \to \mathbb{R}^{30}$ is a p-order polynomial mapping. The model can be reformulated as $\boldsymbol{x} = \boldsymbol{P}\boldsymbol{z}$, where $\boldsymbol{P} \in \mathbb{R}^{30\times(\binom{3+p}{p}-1)}$, $\boldsymbol{P} \sim \mathcal{N}(0,1)$, and $\boldsymbol{z}$ consists of the polynomial features of $\boldsymbol{s}$. Consider the following cases:

- *Single nonlinear subspace* Let $p = 3$, generate one $\boldsymbol{P}$ and 100 $\boldsymbol{s}$. Then the rank of $\boldsymbol{X} \in \mathbb{R}^{30\times 100}$ is 19.
- *Union of nonlinear subspaces* Let $p = 3$, generate three $\boldsymbol{P}$ and for each $\boldsymbol{P}$ generate 100 $\boldsymbol{s}$. Then the rank of $\boldsymbol{X} \in \mathbb{R}^{30\times 300}$ is 30.
- *Union of linear subspaces* Let $p = 1$, generate ten $\boldsymbol{P}$ and for each $\boldsymbol{P}$ generate 100 $\boldsymbol{s}$. Then the rank of $\boldsymbol{X} \in \mathbb{R}^{30\times 1000}$ is 30.

We randomly remove some portions of the entries of the matrices and use matrix completion to recover the missing entries. The performances are evaluated by the relative error defined as $RE = \|\widehat{\boldsymbol{X}} - \boldsymbol{X}\|_F / \|\boldsymbol{X}\|_F$ [6], where $\widehat{\boldsymbol{X}}$ denotes the recovered matrix. As shown in Figure 1, the recovery errors of LRMC methods, i.e. LRF [26] and NNM [4], are considerably high. In contrast, HRMC methods especially our KFMC have significantly lower recovery errors. In addition, our KFMC(Poly) and KFMC(RBF) are much more efficient than VMC [24] and NLMC [10], in which randomized SVD [12] has been performed.

Figure 2 shows the results of online matrix completion, in which OL-DLSR (dictionary learning and sparse representation) is an online matrix completion method we modified from [21] and [11] and detailed in our supplementary material. We see that our method OL-KFMC outperformed other methods significantly. Figures 3 shows the results of out-of-sample extension (OSE) of HRMC, in which our OSE-KFMC outperformed other methods. More details about the experiment/parameter settings and analysis are in the supplementary material.

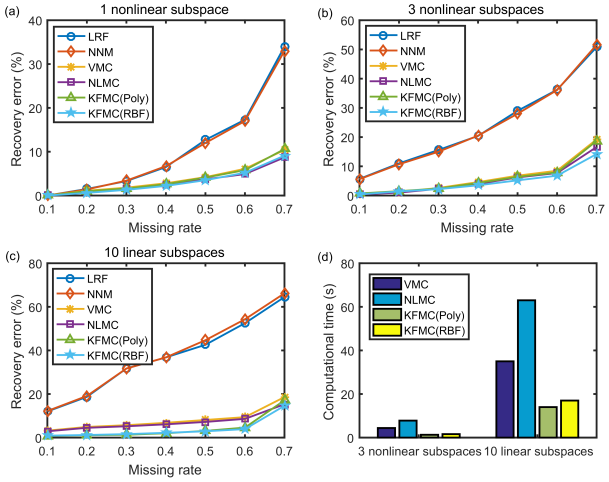

Figure 1: Offline matrix completion on synthetic data

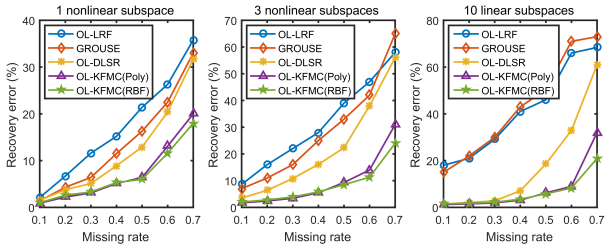

Figure 2: Online matrix completion on synthetic data

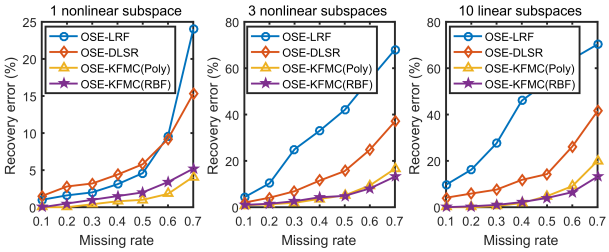

Figure 3: Out-of-sample extension of matrix completion on synthetic data

4.2. Hopkins155 data

Similar to [24], we consider the problem of subspace clustering on incomplete data, in which the missing data of Hopkins155 [27] were recovered by matrix completion and then SSC (sparse subspace clustering [7]) was performed. We consider two downsampled video sequences, *1R2RC* and *1R2TCR*, each of which consists of 6 frames. The average clustering errors [7] of 10 repeated trials are reported in Figure 4. Our method KFMC with RBF kernel is more accurate and efficient than VMC and NLMC.

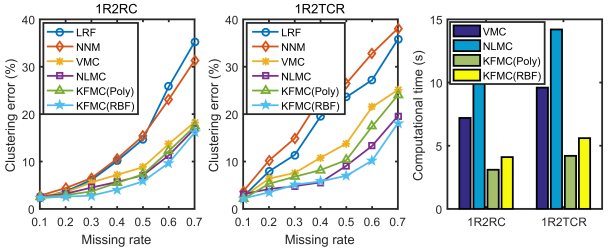

Figure 4: Subspace clustering on incomplete data

4.3. CMU motion capture data

We use matrix completion to recover the missing data of time-series trajectories of human motions (e.g. running and jumping). Similar to [6, 24], we use the trial #6 of subject #56 of the CMU motion capture dataset, which forms a high rank matrix [6]. We consider two cases of incomplete data, randomly missing and continuously missing. More details about the experimental settings are in the supplementary material. The average results of 10 repeated trials are reported in Figure 5. We see that HRMC methods outperformed LRMC methods while online methods outperformed offline methods. One reason is that the structure of the data changes with time (corresponding to different motions) and online methods can adapt to the changes. Comparing Figure 5 with the Figure 4 of [6], we find that VMC, NLMC, and our KFMC outperformed the method proposed in [6]. In addition, our OL-KFMC especially with RBF kernel is the most accurate one. Regarding the computational cost, there is no doubt that the linear methods including LRF, NNM, GROUSE, and DLSR are faster than other methods. Hence we only show the computational cost of the nonlinear methods in Table 2 for comparison (randomized SVD [12] has been performed in VMC and NLMC). Our KFMC is faster than VMC and NLMC while our OL-KFMC is at least 10 times faster than all methods.

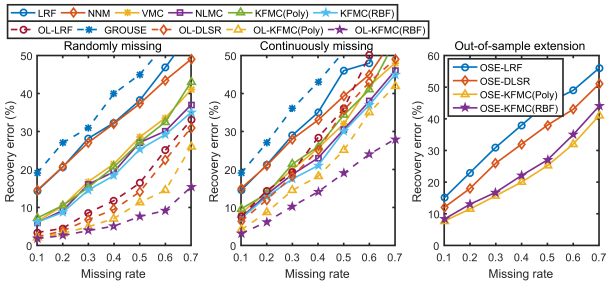

Figure 5: CMU motion capture data recovery

VMC	370	NLMC	610	KFMC$_{(Poly)}$	170
KFMC$_{(RBF)}$	190	OL-KFMC$_{(Poly)}$	16	OL-KFMC$_{(RBF)}$	19

Table 2: Time cost (second) on CMU motion capture data

5. Conclusion

In this paper, we proposed kernelized factorization matrix completion (KFMC), a new method for high rank matrix completion, together with an online version and an out-of-sample extension, which outperform state-of-the-art methods. Our numerics demonstrate the success of the method for motion data recovery. We believe our methods will also be useful for transductive learning (classification), vehicle/robot/chemistry sensor signal denoising, recommender systems, and biomedical data recovery.

Acknowledgement

This work was supported in part by DARPA Award FA8750-17-2-0101.

References

[1] Xavier Alameda-Pineda, Elisa Ricci, Yan Yan, and Nicu Sebe. Recognizing emotions from abstract paintings using non-linear matrix completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5240–5248, 2016. 1

[2] Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 704–711. IEEE, 2010. 1

[3] Srinadh Bhojanapalli and Prateek Jain. Universal matrix completion. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1881–1889. JMLR, 2014. 1

[4] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009. 1, 7

[5] Charanpal Dhanjal, Romaric Gaudel, and Stéphan Clémençon. Online matrix completion through nuclear norm regularisation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 623–631. SIAM, 2014. 1

[6] Ehsan Elhamifar. High-rank matrix completion and clustering under self-expressive models. In *Advances in Neural Information Processing Systems 29*, pages 73–81, 2016. 1, 2, 7, 8

[7] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013. 1, 8

[8] Brian Eriksson, Laura Balzano, and Robert Nowak. High-rank matrix completion. In *Artificial Intelligence and Statistics*, pages 373–381, 2012. 1

[9] Jicong Fan and Tommy W.S. Chow. Matrix completion by least-square, low-rank, and sparse self-representations. *Pattern Recognition*, 71:290 – 305, 2017. 1

[10] Jicong Fan and Tommy W.S. Chow. Non-linear matrix completion. *Pattern Recognition*, 77:378 – 394, 2018. 1, 2, 7

[11] J. Fan, M. Zhao, and T. W. S. Chow. Matrix completion via sparse factorization solved by accelerated proximal alternating linearized minimization. *IEEE Transactions on Big Data*, pages 1–1, 2018. 1, 7

[12] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. 7, 8

[13] Moritz Hardt. Understanding alternating minimization for matrix completion. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 651–660. IEEE, 2014. 1

[14] Chi Jin, Sham M Kakade, and Praneeth Netrapalli. Provable efficient online matrix completion via non-convex stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 4520–4528, 2016. 1

[15] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. Efficient thompson sampling for online matrix-factorization recommendation. In *Advances in neural information processing systems*, pages 1297–1305, 2015. 1

[16] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, June 2010. 1

[17] Chun-Guang Li and Rene Vidal. A structured sparse plus structured low-rank framework for subspace clustering and completion. *IEEE Transactions on Signal Processing*, 64(24):6557–6570, 2016. 1

[18] Guangcan Liu and Ping Li. Low-rank matrix completion in the presence of high coherence. *IEEE Transactions on Signal Processing*, 64(21):5623–5633, 2016. 1

[19] Brian Lois and Namrata Vaswani. Online matrix completion and online robust pca. In *Information Theory (ISIT), 2015 IEEE International Symposium on*, pages 1826–1830. IEEE, 2015. 1

[20] C. Lu, J. Tang, S. Yan, and Z. Lin. Generalized nonconvex nonsmooth low-rank minimization. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4130–4137, June 2014. 1

[21] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009. 1, 7

[22] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug):2287–2322, 2010. 1

[23] Feiping Nie, Heng Huang, and Chris Ding. Low-rank matrix recovery via efficient schatten p-norm minimization. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 655–661. AAAI Press, 2012. 1

[24] Greg Ongie, Rebecca Willett, Robert D. Nowak, and Laura Balzano. Algebraic variety models for high-rank matrix completion. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2691–2700. PMLR, 2017. 1, 2, 6, 7, 8

[25] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM, 2008. 1

[26] Ruoyu Sun and Zhi-Quan Luo. Guaranteed matrix completion via non-convex factorization. *IEEE Transactions on Information Theory*, 62(11):6535–6579, 2016. 1, 7

[27] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 8

[28] Congyuan Yang, Daniel Robinson, and Rene Vidal. Sparse subspace clustering with missing entries. In *International Conference on Machine Learning*, pages 2463–2472, 2015. 1

[29] Se-Young Yun, Marc Lelarge, and Alexandre Proutiere. Streaming, memory limited matrix completion with noise. *arXiv preprint arXiv:1504.03156*, 2015. 1