Closed-form Machine Unlearning for Matrix Factorization

Shuijing Zhang
Guangzhou Institute of Technology, Xidian University
shuijing@stu.xidian.edu.cn

Jian Lou*
Guangzhou Institute of Technology, Xidian University
jlou@xidian.edu.cn

Li Xiong
Department of Computer Science, Emory University
lxiong@emory.edu

Xiaoyu Zhang
School of Cyber Engineering, Xidian University
xiaoyuzhang@xidian.edu.cn

Jing Liu
Guangzhou Institute of Technology, Xidian University
neouma@mail.xidian.edu.cn

ABSTRACT

Matrix factorization (MF) is a fundamental model in data mining and machine learning, which finds wide applications in diverse application areas, including recommendation systems with user-item rating matrices, phenotype extraction from electronic health records, and spatial-temporal data analysis for check-in records. The "right to be forgotten" has become an indispensable privacy consideration due to the widely enforced data protection regulations, which allow personal users having contributed their data for model training to revoke their data through a data deletion request. Consequently, it gives rise to the emerging task of machine unlearning for the MF model, which removes the influence of the matrix rows/columns from the trained MF factors upon receiving the deletion requests from the data owners of these rows/columns. The central goal is to effectively remove the influence of the rows/columns to be forgotten, while avoiding the computationally prohibitive baseline approach of retraining from scratch. Existing machine unlearning methods are either designed for single-variable models and not compatible with MF that has two factors as coupled model variables, or require alternative updates that are not efficient enough. In this paper, we propose a closed-form machine unlearning method. In particular, we explicitly capture the implicit dependency between the two factors, which yields the total Hessian-based Newton step as the closed-form unlearning update. In addition, we further introduce a series of efficiency-enhancement strategies by exploiting the structural properties of the total Hessian. Extensive experiments on five real-world datasets from three application areas as well as synthetic datasets validate the efficiency, effectiveness, and utility of the proposed method.

CCS CONCEPTS

• **Security and privacy → Privacy protections**.

*Corresponding Author.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614811

KEYWORDS

Machine Unlearning; Matrix Factorization; Privacy-Preserving

ACM Reference Format:
Shuijing Zhang, Jian Lou, Li Xiong, Xiaoyu Zhang, and Jing Liu. 2023. Closed-form Machine Unlearning for Matrix Factorization. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614811

1 INTRODUCTION

Matrix Factorization (MF) is a fundamental model in machine learning and data mining. MF has been successfully applied to diverse areas of applications, such as recommendation systems [20, 32, 39], healthcare data analysis [13, 23–25, 33], and spatial-temporal data analysis [17, 19, 21, 41, 47]. MF trains its model by decomposing a data matrix $\mathbf{M}$ and generating two lower-dimensional factor matrices $\mathbf{A}, \mathbf{B}$ such that $\mathbf{M} \approx \mathbf{A}\mathbf{B}^\top$. In many applications, the data matrix $\mathbf{M}$ is collected from personal users, for example, product ratings in recommendation systems, electronic health records in healthcare data analysis, and location check-in records in spatial-temporal data analysis. Recently, the worldwide enacted personal data protection regulations enforce the "Right to be Forgotten", including the European Union's GDPR [8], Canada's proposed Consumer Privacy Protection Act (CPPA), and the California Consumer Privacy Act (CCPA). To comply with the regulations, machine unlearning [1, 3] becomes an emerging problem for machine learning models that rely on user data for training, including matrix factorization [22].

Machine unlearning research focuses on efficiently and effectively removing or scrubbing the influence of the data that needs to be forgotten. One baseline approach is the "retraining-from-scratch" that retrains the entire model from scratch on the remaining training dataset (i.e., the original training dataset excluding the data to be forgotten). Despite taking prohibitively high computations, it is considered the golden standard in terms of the effectiveness of removing the data influence. Current machine unlearning methods can be roughly divided into two categories according to whether dedicated to a specific type of model or broadly applicable to more general models. The former approaches are tailored to the specific properties of the target model, including unlearning from logistic regression [37], random forest [2], quantized K-means [9], and graph neural networks [5, 6]. The latter approaches are not limited to a specific model, including influence function-based methods

[10, 11, 14, 29, 35], reversing optimization path of the model training [36, 40], and SISA [1] and its variants [4, 42] that maintain a series of small models on multiple shards of the training dataset and retrain the small models only on the shard containing the data to be forgotten.

Recently, AltEraser [22] proposes a dedicated machine unlearning method tailored to the two-factor structure of the MF model. AltEraser proposes a two-stage iterative machine unlearning strategy. First, it fine-tunes on the remaining dataset by the original training algorithm for several iterations. Then, it further updates the two factors by influence function-based machine unlearning steps, again in an alternative fashion for several iterations. Although indicating that developing an MF-tailored approach is more promising than applying general machine unlearning methods, AltEraser still comes with several limitations that prevent it from achieving better efficiency and effectiveness. For example, the alternative influence function-based updates are not efficient enough since each iteration requires computing two Hessian matrices. In addition, the alternative updates do not provide clear stopping criteria. In the experiments of [22], the iteration number is heuristically set to 10, which makes it difficult to decide whether the data influence has been effectively removed at the given iteration.

1.1 Our Contributions

In this paper, we propose a **C**losed-form **M**achine **U**nlearning for **M**atrix **F**actorization (CMUMF) approach to effectively and efficiently remove data from trained MF factors. CMUMF builds on the approximate machine unlearning criteria that is derived from the optimality condition of the MF model retrained on the remaining data. However, straightforwardly relying on these criteria leads to a problem involving both factors, which seemingly again requires alternative machine unlearning updates. To deduce it to a single-factor problem, we introduce a set of auxiliary functions to explicitly capture the interdependence between the two factors. In particular, the derivative of the auxiliary function leads us to the total Hessian-based influence function, rather than the conventional Hessian-based influence function adopted in AltEraser, which is capable to capture more holistic influence from the two factors and grants sufficient data influence removal in a single machine unlearning update, as illustrated by Figure 1. To further enhance computational efficiency, we introduce a series of efficiency enhancement techniques when dealing with the total Hessian, including exploiting its sparsity, computing and storing in column-wise and block-wise style, saving from duplicated computations by making use of its symmetry. Extensive experiments on five real-world datasets from three application areas (i.e., user ratings from recommendation systems, check-in records from spatial-temporal analysis, and electronic health records from healthcare analysis), as well as synthetic datasets with three different sizes, validate the superiority of CMUMF over the state-of-the-art machine unlearning algorithms in terms of effectiveness, efficiency, and utility. To summarize, our main contributions are

- We propose a closed-form machine unlearning for matrix factorization approach, which suffices to remove the data influence by a single update due to the holistic influence captured by the total Hessian-based influence function.
- We introduce a series of efficiency enhancement techniques to further improve the computational and storage efficiency when dealing with the total Hessian.
- We conduct extensive experiments on five real-world datasets from three different application areas, as well as synthetic datasets with three different sizes to validate the effectiveness, efficiency, and utility of CMUMF.

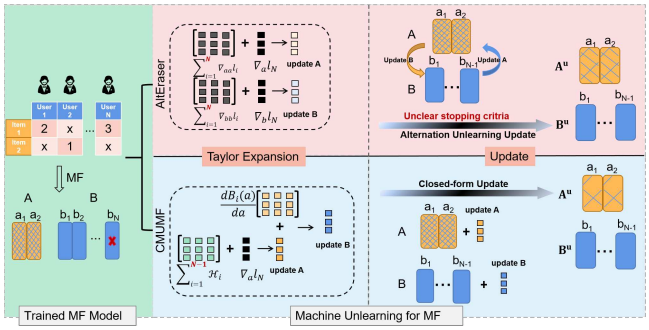

Figure 1: Comparison of AltEraser and CMUMF.

2 BACKGROUND AND PRELIMINARY

Table 1: Frequently used notations in this paper.

Symbol	Definition
$\mathbf{A}, \mathbf{B}$	Factor matrices
$\mathbf{W}$	Weight matrix
$\mathbf{M}$	Data matrix
$\mathbf{A}^u, \mathbf{B}^u$	Factor matrices after unlearning
$\mathbf{A}^\star, \mathbf{B}^\star$	Factor matrices after retrain
$\mathbf{A}^*, \mathbf{B}^*$	Factor matrices without data deletion
$\mathtt{vec}(\cdot)$	Column-first vectorization operator
Π	Projection matrix selecting nonzero entries of $\mathtt{vec}(\mathbf{W})$
$\mathbf{I}_d$	Identity matrix with size $d \times d$
$\nabla^2_{\mathbf{aa}}, \nabla^2_{\mathbf{ab}}, \nabla^2_{\mathbf{ba}}$	Partial Hessian matrices
$\|\cdot\|_F, \|\cdot\|_2$	Frobenius norm, ℓ_2 norm
$\odot, \otimes$	Hadamard product, Kronecker product
$\mathbf{A}_n, \mathbf{B}_n, \mathbf{W}_n, \mathbf{M}_n$	The n-th column of $\mathbf{A}, \mathbf{B}^\top, \mathbf{W}, \mathbf{M}$
$\mathbf{a}, \mathbf{b}, \mathbf{w}, \mathbf{m}$	The column-first vectorizations of $\mathbf{A}, \mathbf{B}, \mathbf{W}, \mathbf{M}$
$\mathbf{a}_n, \mathbf{b}_n, \mathbf{w}_n, \mathbf{m}_n$	The vectorizations of the n-th column of $\mathbf{A}, \mathbf{B}^\top, \mathbf{W}, \mathbf{M}$
$\mathbf{r}_n$	$\mathbf{w}_n \odot (\mathbf{A}\mathbf{b}_n - \mathbf{m}_n)$
$\widetilde{\mathbf{W}}$	$\Pi\mathtt{diag}(\mathtt{vec}(\mathbf{W}))$
$\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}$	$\widetilde{\mathbf{W}}(\mathbf{I} \otimes \mathbf{A}), \widetilde{\mathbf{W}}(\mathbf{B} \otimes \mathbf{I})$
$\mathbf{K}_{PR}$	The permutation matrix satifying $\mathbf{K}_{PR}\mathtt{vec}(\mathbf{A}) = \mathtt{vec}(\mathbf{A}^\top)$
$\mathcal{F}$	The objective function
f	The vectorizations of objective function
$\mathcal{L}_n$	The per-user objective function
l_n	The vectorizations of per-user objective function
$\mathcal{F}^{\backslash N}$	The objective function without user N
$f^{\backslash N}$	The vectorizations of objective function without user N
$\mathcal{B}_n$	The per-user Auxiliary functions
$\mathcal{H}$	The total hessian maxtix

The frequently used notations are summarized in Table 1.

2.1 Matrix Factorization

MF decomposes the data matrix $\mathbf{M} \in \mathbb{R}^{P\times N}$ into two factor matrices $\mathbf{A} \in \mathbb{R}^{P\times R}$ and $\mathbf{B} \in \mathbb{R}^{N\times R}$, where R is the rank. As some entries of $\mathbf{M}$ can be missing in practice, the weight matrix $\mathbf{W} \in \{0,1\}^{P\times N}$ indicates whether the entry is observed (i.e., $\mathbf{W}_{ij} = 1$) or missing (i.e., $\mathbf{W}_{ij} = 0$). MF approximates $\mathbf{M} \approx \mathbf{A}\mathbf{B}^\top$ by solving the following problem

$$[\mathbf{A}^*, \mathbf{B}^*] = \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \mathcal{F}(\mathbf{A},\mathbf{B}) = \|\mathbf{W} \odot (\mathbf{A}\mathbf{B}^\top - \mathbf{M})\|_F^2 + \lambda(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2), \tag{1}$$

where λ is a regularization hyperparameter.

Since the data matrix is collected from personal users in many applications, without loss of generality, we assume different columns of $\mathbf{M}$ correspond to different users and the rows correspond to, e.g., purchased products or the rated items. Then $\mathbf{A}$ is the product/item factor and $\mathbf{B}$ is the user factor. Let $\mathbf{M}_1, \ldots, \mathbf{M}_N$ be each column of $\mathbf{M}$ and $\mathbf{B}_1^*, \ldots, \mathbf{B}_N^*$ be each row of $\mathbf{B}^*$. Eq.(1) can be equivalently described by the following per-user problem formulation,

$$[\mathbf{A}^*, \mathbf{B}_1^*, \ldots, \mathbf{B}_N^*] = \underset{\mathbf{A},\mathbf{B}_1,\ldots,\mathbf{B}_N}{\operatorname{argmin}} \sum_{n=1}^{N} \mathcal{L}_n(\mathbf{A},\mathbf{B}_n) + \lambda\|\mathbf{A}\|_F^2, \tag{2}$$

where $\mathcal{L}_n(\mathbf{A},\mathbf{B}_n)$ is the per-user loss function,

$$\mathcal{L}_n(\mathbf{A},\mathbf{B}_n) = \|\mathbf{W}_n \odot (\mathbf{A}\mathbf{B}_n^\top - \mathbf{M}_n)\|_F^2 + \lambda\|\mathbf{B}_n\|_F^2. \tag{3}$$

Vectorization Notations. In the following paper, we will work with the vectorization notations of MF, which will be convenient for the derivation of the machine unlearning algorithms. Then, the vectorized MF formulation is

$$\begin{aligned}\begin{bmatrix}\mathbf{a}^*\\ \mathbf{b}^*\end{bmatrix} &= \underset{\mathbf{a},\mathbf{b}}{\operatorname{argmin}} f(\mathbf{a},\mathbf{b};\mathbf{m},\mathbf{w}) = \sum_{n=1}^{N} l_n(\mathbf{a},\mathbf{b}_n;\mathbf{m}_n,\mathbf{w}_n) + \lambda\|\mathbf{a}\|_2^2\\ &= \sum_{n=1}^{N}\left(\|\mathbf{w}_n \odot (\mathbf{A}\mathbf{b}_n - \mathbf{m}_n)\|_2^2 + \lambda\|\mathbf{b}_n\|_2^2\right) + \lambda\|\mathbf{a}\|_2^2.\end{aligned} \tag{4}$$

2.2 Matrix Unlearning for Matrix Factorization

MF Problem Formulation after Data Forgotten. For representational simplicity, let the N-th column of $\mathbf{M}$ be the user data to be forgotten, Denote the optimal vectorized factors after forgetting the N-th column by $[\mathbf{a}^\star, \mathbf{b}^\star] = (\mathbf{a}^\star, \mathbf{b}_1^\star, \ldots, \mathbf{b}_{N-1}^\star)$ (please note that the $\mathbf{b}_N$ vector can be directly nullified after forgetting $\mathbf{m}_N$ and we slightly abuse the notation of $\mathbf{b}$), which corresponds to the following MF problem,

$$\begin{aligned}\begin{bmatrix}\mathbf{a}^\star\\ \mathbf{b}^\star\end{bmatrix} &= \underset{\mathbf{a},\mathbf{b}}{\operatorname{argmin}} f^{\setminus N}(\mathbf{a},\mathbf{b};\mathbf{m},\mathbf{w}) = \sum_{n=1}^{N-1} l_n(\mathbf{a},\mathbf{b}_n;\mathbf{m}_n,\mathbf{w}_n) + \lambda\|\mathbf{a}\|_2^2\\ &= \sum_{n=1}^{N-1}\left(\|\mathbf{w}_n \odot (\mathbf{A}\mathbf{b}_n - \mathbf{m}_n)\|_2^2 + \lambda\|\mathbf{b}_n\|_2^2\right) + \lambda\|\mathbf{a}\|_2^2.\end{aligned} \tag{5}$$

Approximate Machine Unlearning Criteria for MF. Before describing the machine unlearning criteria for MF, we recall the first-order optimality condition of eq.(4), where $[\mathbf{a}^*, \mathbf{b}^*]$ are the factors of the trained MF model satisfying:

$$\begin{cases}\nabla_{\mathbf{a}} f(\mathbf{a}^*,\mathbf{b}^*) = \nabla_{\mathbf{a}} \sum_{n=1}^{N} l_n(\mathbf{a}^*,\mathbf{b}^*) + 2\lambda\mathbf{a}^* &= \mathbf{0},\\ \nabla_{\mathbf{b}} f(\mathbf{a}^*,\mathbf{b}^*) = \nabla_{\mathbf{b}} \sum_{n=1}^{N} l_n(\mathbf{a}^*,\mathbf{b}^*) &= \mathbf{0}.\end{cases} \tag{6}$$

Similar to eq.(6), the optimality condition after data forgotten for $\mathbf{a}^\star, \mathbf{b}^\star$ in eq.(5) is:

$$\begin{cases}\nabla_{\mathbf{a}} f^{\setminus N}(\mathbf{a}^\star,\mathbf{b}^\star) = \nabla_{\mathbf{a}} \sum_{n=1}^{N-1} l_n(\mathbf{a}^\star,\mathbf{b}^\star) + 2\lambda\mathbf{a}^\star &= \mathbf{0},\\ \nabla_{\mathbf{b}} f^{\setminus N}(\mathbf{a}^\star,\mathbf{b}^\star) = \nabla_{\mathbf{b}} \sum_{n=1}^{N-1} l_n(\mathbf{a}^\star,\mathbf{b}^\star) &= \mathbf{0}.\end{cases} \tag{7}$$

Machine unlearning for MF seeks to efficiently obtain the updated factors $\mathbf{a}^u, \mathbf{b}^u$ that approximately satisfy the optimality condition in eq.(5):

$$\begin{cases}\nabla_{\mathbf{a}} f^{\setminus N}(\mathbf{a}^u,\mathbf{b}^u) = \nabla_{\mathbf{a}} \sum_{n=1}^{N-1} l_n(\mathbf{a}^u,\mathbf{b}^u) + 2\lambda\mathbf{a}^u &\approx \mathbf{0},\\ \nabla_{\mathbf{b}} f^{\setminus N}(\mathbf{a}^u,\mathbf{b}^u) = \nabla_{\mathbf{b}} \sum_{n=1}^{N-1} l_n(\mathbf{a}^u,\mathbf{b}^u) &\approx \mathbf{0}.\end{cases} \tag{8}$$

2.3 Unlearning Update for Matrix Unlearning

Considering the eq.(4), the AltEraser [22] do Taylor expansion on the objective function $f(\mathbf{a},\mathbf{b};\mathbf{m},\mathbf{w})$ about $\mathbf{a}$ and $\mathbf{b}$ respectively:

$$\begin{cases}f^{\setminus N}(\mathbf{a}^u,\mathbf{b}^u) &= f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*) + \nabla_{\mathbf{a}} f^{\setminus N}(\mathbf{a}^*,\mathbf{a}^*)(\mathbf{a}^u - \mathbf{b}^*) +\\ &\quad \tfrac{1}{2}(\mathbf{a}^u - \mathbf{a}^*)^\top \nabla_{\mathbf{aa}}^2 f^{\setminus N}(\mathbf{a}^*,\mathbf{a}^*)(\mathbf{a}^u - \mathbf{a}^*),\\ f^{\setminus N}(\mathbf{a}^u,\mathbf{a}^u) &= f^{\setminus N}(\mathbf{a}^*,\mathbf{a}^*) + \nabla_{\mathbf{b}} f^{\setminus N}(\mathbf{a}^*,\mathbf{a}^*)(\mathbf{b}^u - \mathbf{b}^*) +\\ &\quad \tfrac{1}{2}(\mathbf{b}^u - \mathbf{b}^*)^\top \nabla_{\mathbf{bb}}^2 f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*)(\mathbf{b}^u - \mathbf{b}^*).\end{cases} \tag{9}$$

Ignoring the quadratic term and applying the eq.(8), then the eq.(9) can be:

$$\begin{cases}\mathbf{a}^u = \mathbf{a}^* - [\nabla_{\mathbf{aa}}^2 f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*)]^{-1} \nabla_{\mathbf{a}} f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*),\\ \mathbf{b}^u = \mathbf{b}^* - [\nabla_{\mathbf{bb}}^2 f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*)]^{-1} \nabla_{\mathbf{b}} f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*).\end{cases} \tag{10}$$

Owing to $\mathbf{a}^*, \mathbf{b}^*$ are the minimum point of the function $f(\mathbf{a},\mathbf{b};\mathbf{m},\mathbf{w})$,

$$\begin{cases}\nabla_{\mathbf{a}} f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*) + \nabla_{\mathbf{a}} l_N(\mathbf{a}^*,\mathbf{b}_N^*) = 0,\\ \nabla_{\mathbf{b}} f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*) + \nabla_{\mathbf{b}} l_N(\mathbf{a}^*,\mathbf{b}_N^*) = 0,\end{cases} \tag{11}$$

we can obtain Theorem 2.1 of AltEraser's algorithm below:

Theorem 2.1. *The approximate machine unlearning criteria in eq.(8) will be satisfied, when the unlearning updated factor* $\mathbf{a}^u$ *and* $\mathbf{b}^u$ *take the following forms,*

$$\mathbf{a}^u = \mathbf{a}^* + [\nabla_{\mathbf{aa}}^2 f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*)]^{-1} \nabla_{\mathbf{a}} l_N(\mathbf{a}^*,\mathbf{b}_N^*), \tag{12}$$

$$\mathbf{b}^u = \mathbf{b}^* + [\nabla_{\mathbf{bb}}^2 f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*)]^{-1} \nabla_{\mathbf{b}} l_N(\mathbf{a}^*,\mathbf{b}_N^*). \tag{13}$$

3 CLOSED-FORM MATRIX UNLEARNING FOR MATRIX FACTORIZATION

In order to obtain $\mathbf{a}^u, \mathbf{b}^u$ that satisfy the approximate machine unlearning criteria in eq.(8) given the trained MF model factors $\mathbf{a}^*, \mathbf{b}^*$, we take the first-order multi-variable Taylor expansion of $\nabla_{\mathbf{a}} f^{\setminus N}(\mathbf{a}^u,\mathbf{b}^u)$ around $\mathbf{a}^*, \mathbf{b}^*$ and choose $\mathbf{a}^u, \mathbf{b}^u$ to let the Taylor expansion be zero. That is, we take $\nabla_{\mathbf{a}} f^{\setminus N}(\mathbf{a}^u,\mathbf{b}^u) \approx \nabla_{\mathbf{a}} f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*) + \nabla_{\mathbf{aa}}^2 f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*)(\mathbf{a}^u - \mathbf{a}^*) + \nabla_{\mathbf{ab}}^2 f^{\setminus N}(\mathbf{a}^*,\mathbf{b}^*)(\mathbf{b}^u - \mathbf{b}^*)$, which apparently involves both factors $\mathbf{a}^u, \mathbf{b}^u$. To have a closed-form machine unlearning step, we explicitly model the interdependence between $\mathbf{a}^u$ and $\mathbf{b}^u$ by introducing a set of auxiliary functions $\mathbf{b}_n = \mathcal{B}_n(\mathbf{a})$ for each $n \in [N]$, based on which we can further have $\mathbf{b}_n^u - \mathbf{b}_n^* \approx \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}(\mathbf{a}^u - \mathbf{a}^*)$ by the first-order Taylor expansion. At this point, we arrive at a single factor representation (i.e., with respect to $\mathbf{a}$) of the approximate machine unlearning criteria, which gives us the desired closed-form machine unlearning update for $\mathbf{a}^u$. Finally, equipped with $\mathbf{a}^u$, we can easily get $\mathbf{b}^u$ according to the interdependence auxiliary functions, which is again a closed-form update.

3.1 Conceptual Machine Unlearning for MF

First, we derive the conceptual CMUMF algorithm without involving specific MF computations.

Conceptual Core Unlearning Update. Before deriving the core unlearning update, we introduce a set of auxiliary functions to model the interdependency between $\mathbf{a}$ and $\mathbf{b}_n$ for all $n \in [N]$: $\mathcal{B}_n(\mathbf{a}) = \text{argmin}_{\mathbf{b}_n} \sum_{n=1}^{N} l_n(\mathbf{a}, \mathbf{b}_n) + \lambda\|\mathbf{a}\|_2^2 = \text{argmin}_{\mathbf{b}_n} l_n(\mathbf{a}, \mathbf{b}_n)$. In particular, by substituting $\mathbf{a}^u$ and $\mathbf{a}^*$ in, we will respectively have $\mathcal{B}_n(\mathbf{a}^u)$ and $\mathcal{B}_n(\mathbf{a}^*)$. The specific formulation of $\mathcal{B}_n(\mathbf{a})$ and its derivative $\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}$ will be given in Sec.3.2.

The conceptual core unlearning update is summarized in Theorem 3.1 below.

Theorem 3.1. *The approximate machine unlearning criteria in eq.(8) will be satisfied, when the unlearning updated factors $\mathbf{a}^u$ and $\mathbf{b}^u$ take the following forms,*

$$\mathbf{a}^u = \mathbf{a}^* + \Delta_N, \quad \mathbf{b}_n^u = \mathbf{b}_n^* + \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Delta_N, \tag{14}$$

where Δ_N is the core unlearning update for forgetting the N-th column as follows $\Delta_N =$

$$\Big(2\lambda\mathbf{I} + \sum_{n=1}^{N-1}\Big[\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Big]\Big)^{-1} \cdot \nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*).$$

Proof. By the definition of $\nabla_{\mathbf{a}} f^{\backslash N}(\mathbf{a}^u, \mathbf{b}^u)$ in eq.(8), we have

$$\begin{aligned}
&\nabla_{\mathbf{a}} f^{\backslash N}(\mathbf{a}^u, \mathbf{b}^u) = 2\lambda\mathbf{a}^u + \nabla_{\mathbf{a}} \sum_{n=1}^{N-1} l_n(\mathbf{a}^u, \mathbf{b}_n^u) \\
&\approx 2\lambda\mathbf{a}^u + \sum_{n=1}^{N-1}\Big[\nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \\
&\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathbf{a}^u - \mathbf{a}^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathbf{b}_n^u - \mathbf{b}_n^*)\Big] \\
&= 2\lambda\mathbf{a}^u + \sum_{n=1}^{N-1}\Big[\nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \\
&\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathbf{a}^u - \mathbf{a}^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathcal{B}_n(\mathbf{a}^u) - \mathcal{B}_n(\mathbf{a}^*))\Big],
\end{aligned}$$

where the approximation is by multi-variable Taylor expansion and omitting the higher-order terms, the equation is by introducing the auxiliary functions $\mathcal{B}_n(\mathbf{a})$ at $\mathbf{a}^u$ and $\mathbf{a}^*$. Next, by first-order Taylor expansion and omitting the high-order terms, we have

$$\mathcal{B}_n(\mathbf{a}^u) - \mathcal{B}_n(\mathbf{a}^*) \approx \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Big|_{\mathbf{a}=\mathbf{a}^*}(\mathbf{a}^u - \mathbf{a}^*), \tag{15}$$

which we abbreviate by $\mathcal{B}_n(\mathbf{a}^u) - \mathcal{B}_n(\mathbf{a}^*) \approx \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}(\mathbf{a}^u - \mathbf{a}^*)$ for notational simplicity in the following. Thus, we further have

$$\begin{aligned}
&\nabla_{\mathbf{a}} f^{\backslash N}(\mathbf{a}^u, \mathbf{b}^u) \approx 2\lambda\mathbf{a}^u + \sum_{n=1}^{N-1}\Big[\nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \\
&\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathbf{a}^u - \mathbf{a}^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}(\mathbf{a}^u - \mathbf{a}^*)\Big].
\end{aligned} \tag{16}$$

By the optimality condition of $\mathbf{a}^*$ in eq.(6), we have $2\lambda\mathbf{a}^u + \sum_{n=1}^{N-1} l_n(\mathbf{a}^u, \mathbf{b}^u) = -\nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*) + 2\lambda(\mathbf{a}^u - \mathbf{a}^*)$, then

$$\begin{aligned}
&\nabla_{\mathbf{a}} f^{\backslash N}(\mathbf{a}^u, \mathbf{b}^u) \approx -\nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*) + 2\lambda(\mathbf{a}^u - \mathbf{a}^*) + \\
&\sum_{n=1}^{N-1}\Big[\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Big](\mathbf{a}^u - \mathbf{a}^*).
\end{aligned} \tag{17}$$

To satisfy the approximate unlearning criteria, we let the right-hand side of the above eq.(17) be zero, which gives

$$\begin{aligned}
&\mathbf{a}^u = \mathbf{a}^* + \Big(2\lambda\mathbf{I} + \sum_{n=1}^{N-1}\Big[\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \\
&\nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Big]\Big)^{-1}\nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*) = \mathbf{a}^* + \Delta_N.
\end{aligned} \tag{18}$$

In addition, for $n \in [N-1]$, let $\mathbf{b}_n^u = \mathbf{b}_n^* + \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Delta_N$, which gives $\mathbf{b}_n^u \approx \mathcal{B}_n(\mathbf{a}^*) + \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}(\mathbf{a}^u - \mathbf{a}^*)$. By the definition of $\mathbf{b}_n^u = \mathcal{B}_n(\mathbf{a}^u) = \text{argmin}_{\mathbf{b}_n} l_n(\mathbf{a}^u, \mathbf{b}_n)$, we have $\mathbf{b}_n^u \approx \mathcal{B}_n(\mathbf{a}^u)$, i.e., $\mathbf{b}_n^u$ is the approximate optimum of $\text{argmin}_{\mathbf{b}_n} l_n(\mathbf{a}^u, \mathbf{b}_n)$.

As a result, $\mathbf{a}^u$ and $\mathbf{b}^u$ satisfy the approximate machine unlearning criteria that $\nabla_{\mathbf{a}} f^{\backslash N}(\mathbf{a}^u, \mathbf{b}^u) = \nabla_{\mathbf{a}} \sum_{n=1}^{N-1} l_n(\mathbf{a}^u, \mathbf{b}^u) + 2\lambda\mathbf{a}^u \approx \mathbf{0}$ and $\sum_{n=1}^{N-1} \nabla_{\mathbf{b}} l_n(\mathbf{a}^u, \mathbf{b}^u) \approx 0$. □

Remark 1. It is worth mentioning that the term $\Big[\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Big]$ takes the form of total Hessian in optimization literature [46]. Also, Δ_N can be regarded as a total Hessian-based Full Newton step, which has been proven to have fast convergence for MF model training due to the rich curvature information provided by total Hessian [16]. On the contrary, [22] adopts the conventional Hessian-based (i.e., $\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*)$) Newton step, which misses the indirect Hessian part (i.e., $\nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}$) and has to run multiple alternative updates for machine unlearning.

Conceptual CMUMF Algorithm. According to Theorem 3.1, the overall computational efficiency hinges on the core unlearning update Δ_N. We reformulate it as follows,

$$\begin{aligned}
&\sum_{n=1}^{N-1}\Big[\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Big] \\
&= \sum_{n=1}^{N}\Big[\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Big] \\
&- \Big[\nabla^2_{\mathbf{aa}} l_N(\mathbf{a}^*, \mathbf{b}_N^*) + \nabla^2_{\mathbf{ab}_N} l_N(\mathbf{a}^*, \mathbf{b}_N^*)\frac{d\mathcal{B}_N(\mathbf{a}^*)}{d\mathbf{a}}\Big],
\end{aligned} \tag{19}$$

where the first line of the right-hand-side is static, which can be computed right after training and before unlearning, while the second line is specified by the data forgotten request and needs to be evaluated at the unlearning time.

- Before unlearning: i) For each $n \in [N]$, compute $\mathcal{H}_n = \nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)\frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}$; ii) keep $\mathcal{H} = \sum_{n=1}^{N} \mathcal{H}_n$ in storage.
- Upon Unlearning: i) Upon receiving the data forgotten request from the N-th user, compute $\nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*)$ and $\mathcal{H}_N$; ii) Compute the core unlearning update Δ_N; iii) Obtain the unlearning updated factors $\mathbf{a}^u$ and $\mathbf{b}^u$.

3.2 Actual Machine Unlearning for MF

We present the actual CMUMF, which comes with a series of computational efficiency-enhancing strategies, including memorizing pre-computable Hessian, sparsity-aware computation, column-wise and block-wise Hessian computation and storage, and avoiding duplicated computation by exploiting the symmetry of Hessian.

Exploiting Sparsity and Additional Notations. For MF with missing observations, the Hessian-related matrices often have a very large number of zero elements. Thus, it suffices to compute and store only non-zero elements. To exploit such sparsity in Hessian, we have $\widetilde{\mathbf{W}} = \Pi \texttt{diag}(\mathbf{w}), \widetilde{\mathbf{W}}_n = \Pi_n \texttt{diag}(\mathbf{w}_n)$, $\widetilde{\mathbf{m}} = \widetilde{\mathbf{W}}\mathbf{m}, \widetilde{\mathbf{m}}_n = \widetilde{\mathbf{W}}_n\mathbf{m}_n$, $\widetilde{\mathbf{A}}_n = \widetilde{\mathbf{W}}_n\mathbf{A}$, $\widetilde{\mathbf{B}}_n = \widetilde{\mathbf{W}}_n(\mathbf{b}_n^\top \otimes \mathbf{I}_P)$, and $\mathbf{Z}_n = \mathbf{r}_n \otimes \mathbf{I}_R$. When the terms are evaluated at $\mathbf{A}^*$ and $\mathbf{B}^*$, we denote them by $(\cdot)^*$, e.g., $r_n^*, \mathbf{Z}_n^*$. Then, l_n can be equivalently denoted by

$$\begin{aligned} l_n(\mathbf{a};\mathbf{b}) &= \|\Pi_n \texttt{diag}(\mathbf{w}_n)(\mathbf{A}\mathbf{b}_n - \mathbf{m}_n)\|_2^2 + \lambda\|\mathbf{b}_n\|_2^2 \\ &= \|\widetilde{\mathbf{A}}_n\mathbf{b}_n - \widetilde{\mathbf{m}}\|_2^2 + \lambda\|\mathbf{b}_n\|_2^2. \end{aligned} \tag{20}$$

Column-wise Gradient and Hessian Computations. The gradient and partial Hessian matrices $\nabla_{\mathbf{a}} l_n(\mathbf{a}, \mathbf{b}_n)$, $\nabla_{\mathbf{a}} l_n(\mathbf{a}, \mathbf{b}_n)$ and $\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}, \mathbf{b}_n)$ for the MF model can be easily derived, which are summarized by Fact 3.1 below. In particular, Fact 3.1 shows that these components can be computed column-by-column (i.e., user-by-user).

Fact 3.1. *The gradient and partial Hessian $\nabla_{\mathbf{a}} l_n(\mathbf{a}, \mathbf{b}_n)$, $\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}, \mathbf{b}_n)$ and $\nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}, \mathbf{b}_n)$ for the MF model can be computed for each column $n \in [N]$ as follows,*

$$\nabla_{\mathbf{a}} l_n(\mathbf{a}, \mathbf{b}_n) = 2\widetilde{\mathbf{B}}_n^\top(\widetilde{\mathbf{A}}_n\mathbf{b}_n - \widetilde{\mathbf{m}}_n), \tag{21}$$

$$\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}, \mathbf{b}_n) = 2\widetilde{\mathbf{B}}_n^\top\widetilde{\mathbf{B}}_n, \tag{22}$$

$$\nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}, \mathbf{b}_n) = 2\widetilde{\mathbf{B}}_n^\top\widetilde{\mathbf{A}}_n + 2\mathbf{K}_{PR}^\top\mathbf{Z}_n. \tag{23}$$

Auxiliary Function and Its Derivative. Next, we present the explicit form of the auxiliary functions that capture the interdependency between $\mathbf{a}$ and $\mathbf{b}_n$ For each n, we have

$$\begin{aligned} \mathcal{B}_n(\mathbf{a}) &= \operatorname*{argmin}_{\mathbf{b}_n} \sum_{n=1}^{N} \left(\|\widetilde{\mathbf{A}}_n\mathbf{b}_n - \widetilde{\mathbf{m}}_n\|_2^2 + \lambda\|\mathbf{b}_n\|_2^2\right) + \lambda\|\mathbf{a}\|_2^2 \\ &= \operatorname*{argmin}_{\mathbf{b}_n} \|\widetilde{\mathbf{A}}_n\mathbf{b}_n - \widetilde{\mathbf{m}}_n\|_2^2 + \lambda\|\mathbf{b}_n\|_2^2 = (\widetilde{\mathbf{A}}_n^\top\widetilde{\mathbf{A}}_n + \lambda\mathbf{I})^{-1}(\widetilde{\mathbf{A}}_n)^\top\widetilde{\mathbf{m}}_n \\ &= \widetilde{\mathbf{A}}_n^{-\lambda}\widetilde{\mathbf{m}}_n, \ [\text{Define } \widetilde{\mathbf{A}}_n^{-\lambda} := (\widetilde{\mathbf{A}}_n^\top\widetilde{\mathbf{A}}_n + \lambda\mathbf{I})^{-1}(\widetilde{\mathbf{A}}_n)^\top], \end{aligned}$$

which indicates that the function $\mathcal{B}_n(\mathbf{a})$ of $\mathbf{a}$ depends only on the n-th user data of $\mathbf{w}_n, \mathbf{m}_n$. The next lemma provides the derivative of $\mathcal{B}_n(\mathbf{a})$ at $\mathbf{a}^*$.

Lemma 3.2. *The auxiliary function $\mathcal{B}_n(\mathbf{a})$ has derivative with respect to $\mathbf{a}$ at $\mathbf{a}^*$: $\frac{d\mathcal{B}_n(\mathbf{a})}{d\mathbf{a}}\Big|_{\mathbf{a}=\mathbf{a}^*} = -((\widetilde{\mathbf{A}}_n^*)^\top\widetilde{\mathbf{A}}_n^* + \lambda\mathbf{I})^{-1}\Big((\widetilde{\mathbf{A}}_n^*)^\top\widetilde{\mathbf{B}}_n^* + (\mathbf{Z}_n^*)^\top\mathbf{K}_{PR}\Big)$.*

Exploiting Symmetry of Total Hessian. Equipped with Fact 3.1 and Lemma 3.2 that have prepared the prerequisite components, each entry $\mathcal{H}_n$ of the total Hessian $\mathcal{H}$ becomes,

$$\begin{aligned} \mathcal{H}_n &= 2\big[\widetilde{\mathbf{B}}_n^{*\top}(\mathbf{I} - \widetilde{\mathbf{A}}_n^*\widetilde{\mathbf{A}}_n^{*-\lambda})\widetilde{\mathbf{B}}_n^*\big]_{\text{Term I}} \\ &- 2\big[\mathbf{K}_{PR}^\top\mathbf{Z}_n^*(\widetilde{\mathbf{A}}_n^{*\top}\widetilde{\mathbf{A}}_n^* + \lambda\mathbf{I})^{-1}\mathbf{Z}_n^{*\top}\mathbf{K}_{PR}\big]_{\text{Term II}} \\ &- 2\big[\widetilde{\mathbf{B}}_n^{*\top}(\widetilde{\mathbf{A}}_n^{*-\lambda})^\top\mathbf{Z}_n^{*\top}\mathbf{K}_{PR} + \mathbf{K}_{PR}^\top\mathbf{Z}_n^*\widetilde{\mathbf{A}}_n^{*-\lambda}\widetilde{\mathbf{B}}_n^*\big]_{\text{Term III}}. \end{aligned} \tag{24}$$

The overall efficiency of the machine unlearning largely hinges on the efficiency of the total Hessian computation. In addition to exploiting the sparsity, we exploit the symmetry of total Hessian to enhance efficiency.

For $(\widetilde{\mathbf{A}}_n^\top\widetilde{\mathbf{A}}_n + \lambda\mathbf{I})^{-1}$ that is shared by Term I-III in eq.(24), we introduce sparse Cholesky decomposition as follows

$$\texttt{chol}(\widetilde{\mathbf{A}}_n^{*\top}\widetilde{\mathbf{A}}_n^* + \lambda\mathbf{I}) = \mathbf{L}_n\mathbf{L}_n^\top, \tag{25}$$

where $\mathbf{L}_n$ is a lower triangle matrix. Then, $(\widetilde{\mathbf{A}}_n^\top\widetilde{\mathbf{A}}_n + \lambda\mathbf{I})^{-1} = (\mathbf{L}_n^{-1})^\top\mathbf{L}_n^{-1}$, which entails the symmetric proposition below.

Proposition 3.3. *Term I and Term II can be equivalently calculated as three Kronecker products between symmetric matrices. For Term I, we have*

$$\widetilde{\mathbf{B}}_n^{*\top}\widetilde{\mathbf{B}}_n^* = (\mathbf{b}_n^*\mathbf{b}_n^{*\top}) \otimes (\widetilde{\mathbf{W}}_n^\top\widetilde{\mathbf{W}}_n); \tag{26}$$

$$\widetilde{\mathbf{B}}_n^{*\top}\widetilde{\mathbf{A}}_n^*\widetilde{\mathbf{A}}_n^{*-\lambda}\widetilde{\mathbf{B}}_n^* = (\mathbf{b}_n^*\mathbf{b}_n^{*\top}) \otimes (\widetilde{\mathbf{W}}_n^\top\widetilde{\mathbf{A}}_n(\mathbf{L}_n^{-1})^\top\mathbf{L}_n^{-1}\widetilde{\mathbf{A}}_n^\top\widetilde{\mathbf{W}}_n). \tag{27}$$

For Term II, we have

$$\begin{aligned} &\mathbf{K}_{PR}^\top\mathbf{Z}_n^*(\widetilde{\mathbf{A}}_n^{*\top}\widetilde{\mathbf{A}}_n^* + \lambda\mathbf{I})^{-1}\mathbf{Z}_n^{*\top}\mathbf{K}_{PR} \\ &= ((\mathbf{L}_n^{-1})^\top\mathbf{L}_n^{-1}) \otimes ((\mathbf{w}_n \odot \mathbf{r}_n^*)(\mathbf{w}_n \odot \mathbf{r}_n^*)^\top). \end{aligned} \tag{28}$$

Proposition 3.3 not only helps save duplicated computations of repeated terms (e.g., $\mathbf{b}_n^*\mathbf{b}_n^{*\top}$), but more importantly also converts the direct matrix products to Kronecker products between pairs of symmetric matrices, which has been validated to have a speed up by up to 4 times according to MF optimization literature [16]. Finally, Term III contains two symmetric parts (i.e., $\widetilde{\mathbf{B}}_n^{*\top}(\widetilde{\mathbf{A}}_n^{*-\lambda})^\top\mathbf{Z}_n^{*\top}\mathbf{K}_{PR} = (\mathbf{K}_{PR}^\top\mathbf{Z}_n^*\widetilde{\mathbf{A}}_n^{*-\lambda}\widetilde{\mathbf{B}}_n^*)^\top$), which also saves half the computation.

Actual CMUMF Algorithm. Equipped with Proposition 3.3 for computing eq.(19), we summarize the actual unlearning algorithm below.

- Before Unlearning: i) For each $n \in [N]$, compute $\mathcal{H}_n$ by

$$\begin{aligned} \mathcal{H}_n &= 2\big[(\mathbf{b}_n^*\mathbf{b}_n^{*\top}) \otimes (\widetilde{\mathbf{W}}_n^\top\widetilde{\mathbf{W}}_n) \\ &- (\mathbf{b}_n^*\mathbf{b}_n^{*\top}) \otimes (\widetilde{\mathbf{W}}_n^\top\widetilde{\mathbf{A}}_n(\mathbf{L}_n^{-1})^\top\mathbf{L}_n^{-1}\widetilde{\mathbf{A}}_n^\top\widetilde{\mathbf{W}}_n)\big] \\ &- 2\big[((\mathbf{L}_n^{-1})^\top\mathbf{L}_n^{-1}) \otimes ((\mathbf{w}_n \odot \mathbf{r}_n^*)(\mathbf{w}_n \odot \mathbf{r}_n^*)^\top\big] \\ &- 2\Big[\widetilde{\mathbf{B}}_n^{*\top}(\widetilde{\mathbf{A}}_n^{*-\lambda})^\top\mathbf{Z}_n^{*\top}\mathbf{K}_{PR} + \mathbf{K}_{PR}^\top\mathbf{Z}_n^*\widetilde{\mathbf{A}}_n^{*-\lambda}\widetilde{\mathbf{B}}_n\Big]. \end{aligned} \tag{29}$$

  ii) keep $\mathcal{H} = \sum_{n=1}^N \mathcal{H}_n$ in storage.
- Upon Unlearning: i) Upon receiving the data forgotten request from the N-th user, compute $\nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*) = 2\widetilde{\mathbf{B}}_N^{*\top}(\widetilde{\mathbf{A}}_N^*\mathbf{b}_N^* - \widetilde{\mathbf{m}}_N)$, and $\mathcal{H}_N$ by eq.(29) with $n = N$; ii) Compute the core unlearning update $\Delta_N = (2\lambda\mathbf{I} + \mathcal{H} - \mathcal{H}_N)^{-1}\nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*)$, which can be solved by Conjugate Gradient or Fixed-point method. iii) Obtain the unlearning updated factors $\mathbf{a}^u$ and $\mathbf{b}^u$ by eq.(14) in Theorem 3.1, correspondingly.

3.3 The Complete CMUMF Algorithm Description

We provide the complete algorithm description in Algorithm 1. The CMUMF algorithm consists of two stages: *Before Unlearning stage and Upon Unlearning stage*. In particular, Option I corresponds to single-column removal and Option II corresponds to batch-of-columns removal. The proof of batch-of-columns removal can be found in Appendix A.

Algorithm 1 Closed-form Machine Unlearning for Matrix Factorization (CMUMF)

Input: Matrix factorization model (with data matrix $\mathbf{M}$, weight matrix $\mathbf{W}$, and regularization parameter λ), trained factor $\mathbf{a}^*$, factor $\mathbf{b}^*$, delete request N or $[S]$.

Stage I. Before Unlearning:

1: Compute the total Hessian of each user $\mathcal{H}_n$, $n \in \{1, 2, ..., N\}$ by

$$\mathcal{H}_n = 2\Big[(\mathbf{b}_n^*\mathbf{b}_n^{*\top}) \otimes (\widetilde{\mathbf{W}}_n^\top \widetilde{\mathbf{W}}_n) - (\mathbf{b}_n^*\mathbf{b}_n^{*\top}) \otimes (\widetilde{\mathbf{W}}_n^\top \widetilde{\mathbf{A}}_n (\mathbf{L}_n^{-1})^\top \mathbf{L}_n^{-1} \widetilde{\mathbf{A}}_n^\top \widetilde{\mathbf{W}}_n)\Big]$$
$$- 2\Big[((\mathbf{L}_n^{-1})^\top \mathbf{L}_n^{-1}) \otimes ((\mathbf{w}_n \odot \mathbf{r}_n^*)(\mathbf{w}_n \odot \mathbf{r}_n^*)^\top\Big] - 2\Big[\widetilde{\mathbf{B}}_n^{*\top} (\widetilde{\mathbf{A}}_n^{*-\lambda})^\top \mathbf{Z}_n^{*\top} \mathbf{K}_{PR} + \mathbf{K}_{PR}^\top \mathbf{Z}_n^* \widetilde{\mathbf{A}}_n^{*-\lambda} \widetilde{\mathbf{B}}_n\Big];$$

2: Keep $\mathcal{H} = \sum_{n=1}^{N} \mathcal{H}_n$ in storage;

Stage II. Upon Unlearning:

3: Execute data forgotten requests:

- **Option I: Single Column Removal**
  - Compute the gradient of the revoked sample N: $\nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*) = 2\widetilde{\mathbf{B}}_N^{*\top}(\widetilde{\mathbf{A}}_N^* \mathbf{b}_N^* - \widetilde{\mathbf{m}}_N)$;
  - Compute the total Hessian of the revoked sample $\mathcal{H}_N$;
  - Compute the core unlearning update: $\Delta_N = \big(2\lambda\mathbf{I} + \mathcal{H} - \mathcal{H}_N\big)^{-1} \nabla_{\mathbf{a}} l_N(\mathbf{a}^*, \mathbf{b}_N^*)$ by conjugate gradient;
  - Update factor $\mathbf{a}$: $\mathbf{a}^u = \mathbf{a}^* + \Delta_N$;
  - Calculate the derivative of $\mathbf{b}$ with respect to $\mathbf{a}$: $\frac{d\mathcal{B}_n(\mathbf{a})}{d\mathbf{a}}\Big|_{\mathbf{a}=\mathbf{a}^*} = -((\mathbf{L}_n^{-1})^\top \mathbf{L}_n^{-1})\Big((\widetilde{\mathbf{A}}_n^*)^\top \widetilde{\mathbf{B}}_n^* + (\mathbf{Z}_n^*)^\top \mathbf{K}_{PR}\Big)$;
  - Update factor $\mathbf{b}$: for each $n \in [N-1]$, $\mathbf{b}_n^u = \mathbf{b}_n^* + \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}} \Delta_N$;
- **Option II: Batch Column Removal**
  - Compute the gradient of the batch revoked samples $[S]$: $\sum_{n\in[S]} \nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) = \sum_{n\in[S]} 2\widetilde{\mathbf{B}}_n^{*\top}(\widetilde{\mathbf{A}}_n^* \mathbf{b}_n^* - \widetilde{\mathbf{m}}_n)$;
  - Compute the Hessian of the batch revoked samples $\sum_{n\in[S]} \mathcal{H}_n$;
  - Compute the core unlearning update: $\Delta_{[S]} = \big(2\lambda\mathbf{I} + \mathcal{H} - \sum_{n\in[S]} \mathcal{H}_n\big)^{-1} \sum_{n\in[S]} \nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*)$ by conjugate gradient;
  - Update factor $\mathbf{a}$: $\mathbf{a}^u = \mathbf{a}^* + \Delta_{[S]}$;
  - Calculate the derivative of $\mathbf{b}_n$ with respect to $\mathbf{a}$ for $n \notin [S]$: $\frac{d\mathcal{B}_n(\mathbf{a})}{d\mathbf{a}}\Big|_{\mathbf{a}=\mathbf{a}^*} = -((\mathbf{L}_n^{-1})^\top \mathbf{L}_n^{-1})\Big((\widetilde{\mathbf{A}}_n^*)^\top \widetilde{\mathbf{B}}_n^* + (\mathbf{Z}_n^*)^\top \mathbf{K}_{PR}\Big)$;
  - Update factor $\mathbf{b}$: for each $n \in [N] \setminus [S]$, $\mathbf{b}_n^u = \mathbf{b}_n^* + \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}} \cdot \Delta_{[S]}$;

4: Resize vectors $\mathbf{a}^u, \mathbf{b}^u$ to matrices $\mathbf{A}^u, \mathbf{B}^u$.

4 EXPERIMENTS

4.1 Experimental Setup

Datasets. We consider five real datasets from three different application areas as well as synthetic datasets. The observation ratio and size of the datasets are summarized in Table 2.

i). Synthetic data: Synthetic data is generated as follows: The data matrix is generated by $\mathbf{M} = \mathbf{A}\mathbf{B}^\top + \mathbf{C}$, where the elements of $\mathbf{A} \in \mathbb{R}^{P\times R}, \mathbf{B} \in \mathbb{R}^{P\times R}$ are sampled i.i.d. from the standard normal distribution $\mathcal{N}(0, 1)$, and elements of $\mathbf{C}$ are sampled from $\mathcal{N}(0, 0.1)$. The ground-truth matrix is $\mathbf{G} = \mathbf{A}\mathbf{B}^\top$. We consider three different sizes of $\mathbf{M}$, including $1k \times 1k$, $2k \times 2k$, $3k \times 3k$.

ii). Recommendation systems data: We use two common datasets in the field of recommendation systems: The first one is Jester [12], which is a joke-rating dataset containing more than 20,000 users' ratings of 36 or more jokes. The rating of the Jester dataset continuously ranges from -10 to 10; The second one is Movielens100k (ML100k) [15]. It contains 100,000 ratings from 1000 users on 1700 movies, in which users rate the movies they have seen, with a score of 1 to 5.

iii). Check-in data: We use the Check-in dataset collected from Foursquare by [43–45]. It includes 18,201 and 11,874 users' check-in records at different POIs (point of interest) in New York City and Tokyo respectively. We choose the top 500 POIs and top 5000 users with the highest number of visiting records in New York City. The Check-in dataset contains an interaction matrix representing user-venue combination with size 5206×509.

iv). Electronic health record (EHR) data: We use the MIMIC-III [18] dataset, which is a free and open Intensive Care Units (ICUs) research database. It has important applications in phenotypic extraction [26–28]. Following[13], we select the top 500 frequently observed medications to form an interaction matrix representing patient-medication combination with size is 5000×500.

Table 2: Dataset Properties

Name	Synthetic	Jester1	Jester2	ML100k	Check-in	MIMIC3
Observation ratio	0.1000	0.5780	0.5800	0.0504	0.0205	0.0797
Size	$1k \times 1k$, $2k \times 2k$, $3k \times 3k$	100×24983	100×23500	1682×943	509×5206	500×5000

Training Algorithms. For each of the datasets, we obtain the trained MF model by six different training algorithms of matrix factorization , including ALS[30], PF[30, 38], BALM[7], SGD with Momentum (Mom-SGD)[31, 34], RMSProp[34], and CGD.

It should also be noted that, real-world data do not have as good properties as synthetic datasets, so it is difficult for some MF training algorithms to find the optimal solution. For example, the alternation approaches (i.e., ALS, PF, BALM) are used for matrix factorization on Jester1 and Jester2 datasets and perform well. However, they have poor performance on more sparse datasets. So we use the gradient descent training algorithms to perform matrix factorization on Movielens100k, Check-in, and MIMIC-III datasets. At last, each experiment runs five times, and then the results are averaged.

Compared Machine Unlearning Methods. The four compared machine unlearning methods include:

- Retraining-from-scratch: The most straightforward machine unlearning method, which retrains the model from scratch on the remaining data. It is computationally expensive but provides

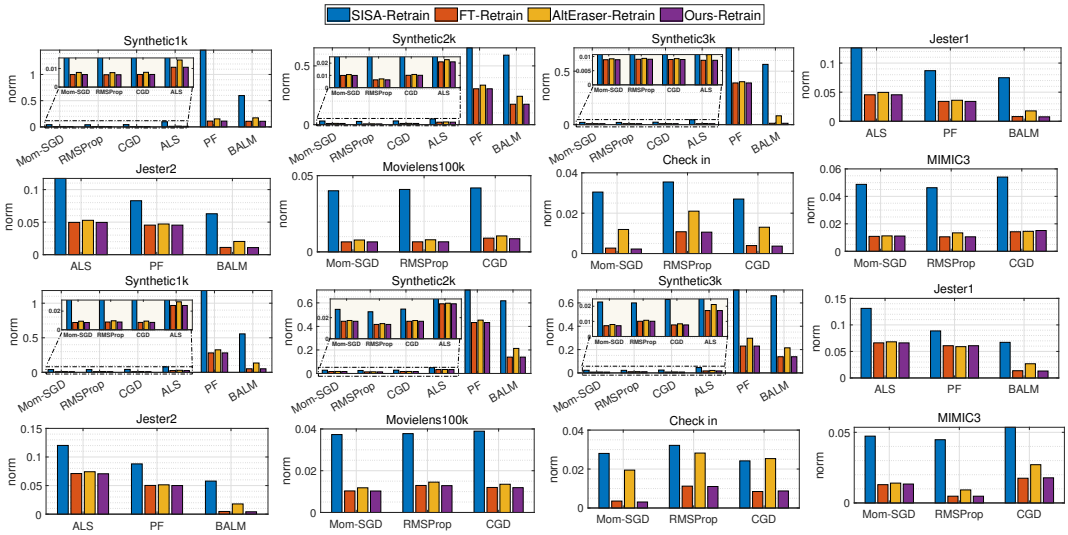

Figure 2: Effectiveness results: Normalized norm differences between factors obtained by the compared machine unlearning algorithms and the factors obtained by Retrain. Row 1, 2 for data removal percentage=5% and row 3, 4 for percentage=10%.

a golden standard for evaluating the effectiveness of the more efficient unlearning algorithms.

- SISA [1]: It randomly splits the training data into shards, then aggregate the results of all submodels.
- Fine-Tuning (FT): The model is initially trained on dataset $\mathcal{D}$, and then continues to be trained (fine-tuned) on the updated dataset $\mathcal{D}$ with the revoked sample removed.
- AltEraser [22]: A dedicated machine unlearning method for MF, which initializes model parameters with warm-start (i.e., Fine-Tuning) and then uses alternative unlearning updates based on the conventional Hessian-based influence function.

Evaluation Methodology. For real-world datasets that already come with a large ratio of missing entries, we select 80% as the training set and the remaining 20% as the test. For synthetic datasets that have all entries available, we randomly select 10% of the data to train and 90% to test (i.e., mimic the missing entries in practice).

We consider three types of evaluation metrics. 1) Effectiveness: we compare the normalized norm difference between the factors obtained by compared machine unlearning algorithms $(\mathbf{A}^u, \mathbf{B}^u)$ and the factors obtained by the retraining-from-scratch model $(\mathbf{A}^\star, \mathbf{B}^\star)$ which is considered as the golden standard: $\left\|\mathbf{A}^u - \mathbf{A}^\star\right\|_F / \texttt{size}(\mathbf{A}^u)$ and $\left\|\mathbf{B}^u - \mathbf{B}^\star\right\|_F / \texttt{size}(\mathbf{B}^u)$. 2) Efficiency: we compare the CPU time of the compared machine unlearning algorithms; 3) Utility: we compare the utility based on Mean Square Error (MSE)-based metrics: RMSE for real datasets and NMSE for synthetic dataset.

4.2 Experiment Results

4.2.1 Effectiveness Comparison of Machine Unlearning.
The effectiveness comparison results are shown in Figure 2.

Synthetic Data. The performances of CMUMF and FT are the same and outperform other methods. In addition, we can see that the norm difference values of SISA-Retrain on the six MF training algorithms gradually decrease as the size of the synthetic dataset increases, which indicates that the SISA model may work better on larger datasets.

Real-world Data. Our method is comparable to or slightly outperforms FT. Meanwhile, AltEraser does not consider the correlation between factor matrices, so it has a relatively large norm difference with Retrain, which is particularly obvious on real-world datasets. SISA has significantly poor performances.

From the case of 5% deletion and 10% deletion, we can see that the larger the number of deletions, the larger the norm value between all unlearning algorithms and Retrain models. This is expected, as the approximation of the efficient unlearning algorithms deteriorates as the number of deletions increases. However, this situation is more obvious on AltEraser, especially on datasets Check-in and MIMIC-III.

4.2.2 Efficiency Comparison of Machine Unlearning.
The efficiency comparison results are shown in Figure 3.

Synthetic Data. Our algorithm does not always consume the least time on the three datasets. It can be seen that on the three MF training algorithms, such as CGD, ALS and PF, the time spent by CMUMF increases gradually with the increase of the dataset, which is due to the fact that the time complexity of CMUMF depends only on the size of the factors and not on the type of MF training algorithm. In addition, it also can be seen that SISA training time is not always less than Retrain. As for AltEraser, the running time is less in most cases, but when the number of deletions is 10%, the running time on RMSProp is much longer than that on Retrain. The algorithm iteration time depends on whether it is easy to use the factor matrix obtained by the original MF algorithm to find the optimal solution after deleting some data.

Real-world Data. The experimental results show that CMUMF takes the least time among all algorithms except on the Movielens100k dataset. In addition, it can be found that due to different algorithms used in matrix factorization and different datasets, the

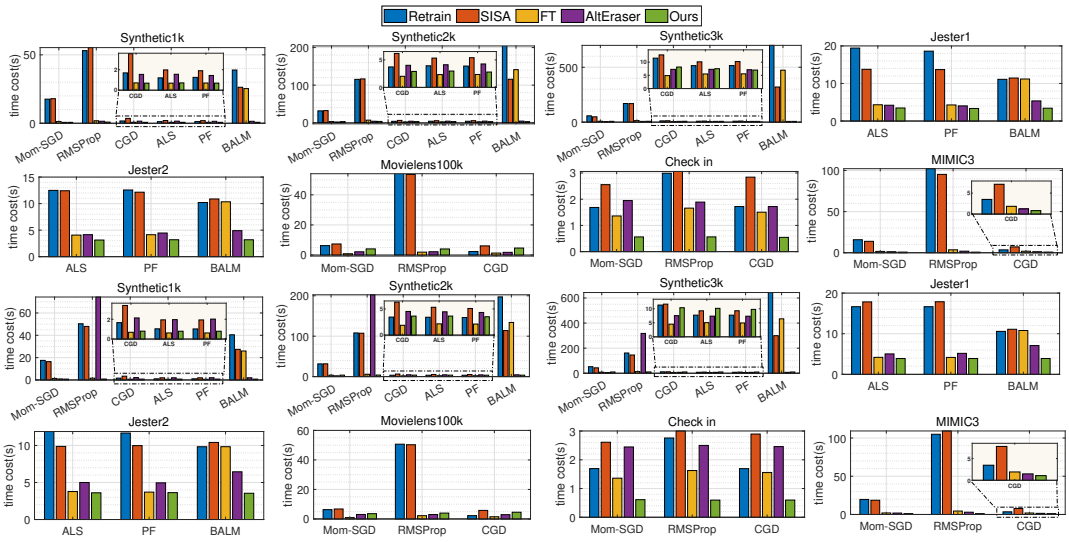

Figure 3: Efficiency results: CPU time (seconds) of the compared machine unlearning algorithms. Row 1, 2 for data removal percentage=5% and row 3, 4 for data removal percentage=10%.

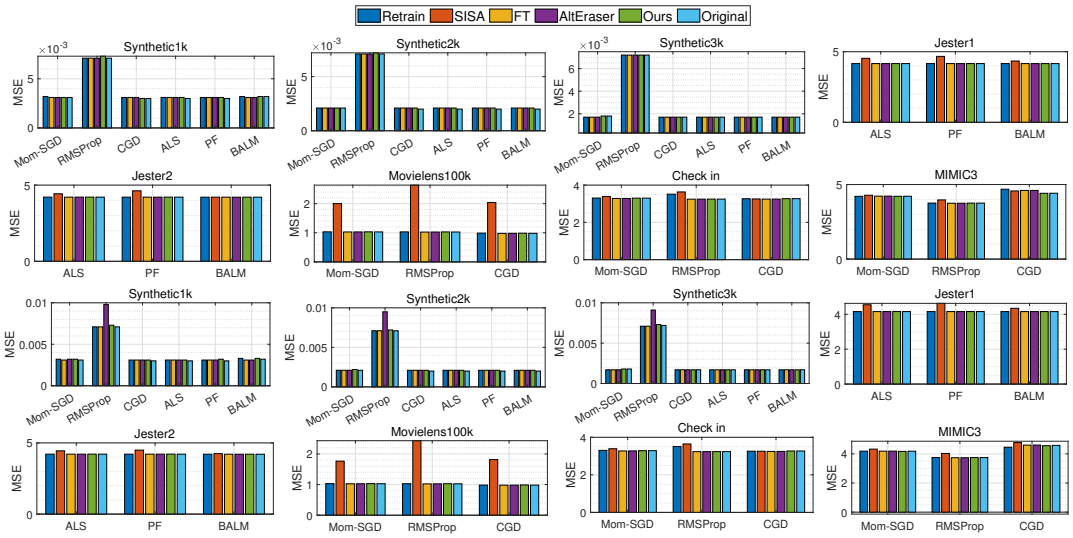

Figure 4: Utility results: MSE of the compared machine unlearning algorithms. Row 1, 2 for data removal percentage=5% and row 3, 4 for data removal percentage=10%.

time taken by the algorithm to find the optimal solution will be affected, which contributes to the fact that Retrain, FT and SISA algorithms are not stable in time. Compared with these algorithms, CMUMF always remains stable in time on the same dataset. As with the synthetic dataset, the SISA model is not always effective in terms of time overhead reduction on the five datasets. Because AltEraser is an iterative algorithm, it takes more time than FT in some cases on real-world datasets.

4.2.3 Utility Comparison of Machine Unlearning.
The utility comparison results are shown in Figure 4.

Synthetic Data. On the three synthetic data, the SISA model has a much larger MSE on all the six MF training algorithms, so we did not present its MSE. It can be seen that, except for SISA, other unlearning algorithms will not reduce the performance of the original model in most cases. It is worth noting that SISA not only has a large time overhead, but also greatly reduces the accuracy of the original matrix factorization model, which necessitates the need to develop a dedicated machine unlearning approach when the target model has unique properties.

Real-world Data. The MSE of our unlearning algorithm is similar to the original model and also retrain and FT algorithms. Unlike

synthetic datasets, SISA does not perform poorly in all MF algorithms. For example, the performance of SISA is comparable to that of retrain for the CGD algorithm on the Check-in dataset and the BALM algorithm on Jester2 dataset. Another striking observation that emerges from the comparison is that SISA performs particularly poorly on the ML100k dataset. By looking at Table 2 we can find that the ML100k and synthetic datasets are similar in size, while the other four datasets are larger, which indirectly indicates that SISA is not suitable for small datasets. Nevertheless, SISA still performs less well than CMUMF on the other four larger datasets.

When the number of deletions is 5% and 10%, except SISA and AltEraser, other unlearning algorithms do not damage the performance of the original model. We obviously noticed that AltEraser seriously damaged the performance of the original model when the MF algorithm was not convergent enough and the number of deletions was 10% (i.e., the RMSProp algorithm on synthetic data).

4.2.4 Effect of data removal percentage.
We study the effect of data removal percentage on CMUMF algorithm. Figure 5 is the time cost of CMUMF on three synthetic datasets when different numbers of data are deleted. On the three datasets, the time overhead increases with the number of deletions because the number of nonzeros in the Hessian matrix to be computed increases. Also, the time overhead is related to the size of the dataset, the larger the dataset, the higher the time overhead.

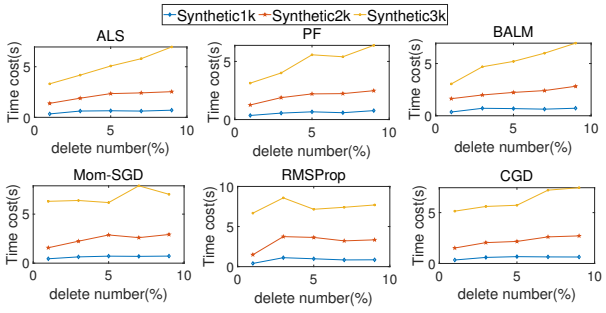

Figure 5: Time curve of CMUMF with the percentage of deletions with different MF training algorithms.

5 CONCLUSION

We proposed CMUMF which obtained a closed-form solution for matrix factorization machine unlearning problem for the first time. More importantly, we focus on the correlation between factor matrices in matrix factorization, so that user information can be deleted more thoroughly. In order to reduce the amount of computation, we introduced a series of efficiency enhancement strategies. Empirical results evaluated on five real-world datasets from three application areas validates the efficiency, effectiveness, and utility of CMUMF.

6 ACKNOWLEDGEMENT

This research has been funded in part by Guangdong Basic and Applied Basic Research Foundation (2021A151511073, 2022A1515011297, 202201011236), Guangdong High-level Innovation Research Institution Project (2021B0909050008), Guangzhou Key Research and Development Program (202206030003), National Science Foundation of China (NSFC) 62206207 and 62102300, National Science Foundation (NSF) CNS-2124104, CNS-2125530, National Institute of Health (NIH) R01ES033241.

A APPENDICES: RANDOM BATCH REMOVAL

In this part, we show that the single-column removal setting can be easily generalized to the random batch removal setting. Let the set of indices of the columns to be forgotten by $[S]$. The approximate machine unlearning criteria after removing $[S]$:

$$\begin{cases} \nabla_{\mathbf{a}} f^{\setminus[S]}(\mathbf{a}^u, \mathbf{b}^u) = \nabla_{\mathbf{a}} \sum_{n\notin[S]} l_n(\mathbf{a}^u, \mathbf{b}^u) + 2\lambda \mathbf{a}^u & \approx \mathbf{0}, \\ \nabla_{\mathbf{b}} f^{\setminus[S]}(\mathbf{a}^u, \mathbf{b}^u) = \nabla_{\mathbf{b}} \sum_{n\notin[S]} l_n(\mathbf{a}^u, \mathbf{b}^u) & \approx \mathbf{0}, \end{cases} \tag{30}$$

Theorem A.1 summarizes the core update of the batch unlearning.

Theorem A.1. *The approximate machine unlearning criteria in eq.(??) will be satisfied, when the unlearning updated factors* $\mathbf{a}^u$ *and* $\mathbf{b}^u$ *take the following forms,*

$$\mathbf{a}^u = \mathbf{a}^* + \Delta_{[S]}, \quad \mathbf{b}_n^u = \mathbf{b}_n^* + \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Delta_{[S]}, \tag{31}$$

where Δ_N *is the core unlearning update:* $\Delta_N = \left(2\lambda\mathbf{I} + \sum_{n\notin[S]} \left[\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*) \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\right]\right)^{-1} \cdot \sum_{n\in[S]} \nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*)$.

Proof. Similar to the single-column removal case, we have

$$\begin{aligned} &\nabla_{\mathbf{a}} f^{\setminus[S]}(\mathbf{a}^u, \mathbf{b}^u) = 2\lambda\mathbf{a}^u + \nabla_{\mathbf{a}} \sum_{n\notin[S]} l_n(\mathbf{a}^u, \mathbf{b}_n^u) \\ &\approx 2\lambda\mathbf{a}^u + \sum_{n\notin[S]} \Big[\nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathbf{a}^u - \mathbf{a}^*) \\ &+ \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathbf{b}_n^u - \mathbf{b}_n^*)\Big] \\ &= 2\lambda\mathbf{a}^u + \sum_{n\notin[S]} \Big[\nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathbf{a}^u - \mathbf{a}^*) \\ &+ \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*)(\mathcal{B}_n(\mathbf{a}^u) - \mathcal{B}_n(\mathbf{a}^*))\Big], \end{aligned} \tag{32}$$

where the approximation is by multi-variable Taylor expansion and omitting the higher-order terms, the equation is by introducing the auxiliary functions $\mathcal{B}_n(\mathbf{a})$ at $\mathbf{a}^u$ and $\mathbf{a}^*$. By first-order Taylor expansion and the optimality condition of $\mathbf{a}^*$ in eq.(6), we have

$$\begin{aligned} \nabla_{\mathbf{a}} f^{\setminus[S]}(\mathbf{a}^u, \mathbf{b}^u) &\approx -\sum_{n\in[S]} \nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + 2\lambda(\mathbf{a}^u - \mathbf{a}^*) \\ &+ \sum_{n\notin[S]} \left[\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*) \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\right](\mathbf{a}^u - \mathbf{a}^*). \end{aligned} \tag{33}$$

To satisfy the approximate unlearning criteria, we let the right-hand side of the above eq.(33) be zero, which gives

$$\begin{aligned} \mathbf{a}^u = \mathbf{a}^* + \Big(2\lambda\mathbf{I} + \sum_{n\notin[S]} \Big[&\nabla^2_{\mathbf{aa}} l_n(\mathbf{a}^*, \mathbf{b}_n^*) + \\ &\nabla^2_{\mathbf{ab}_n} l_n(\mathbf{a}^*, \mathbf{b}_n^*) \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Big]\Big)^{-1} \sum_{n\in[S]} \nabla_{\mathbf{a}} l_n(\mathbf{a}^*, \mathbf{b}_N^*) = \mathbf{a}^* + \Delta_{[S]}. \end{aligned} \tag{34}$$

In addition, for $n \in [N] \setminus [S]$, let $\mathbf{b}_n^u = \mathbf{b}_n^* + \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}\Delta_{[S]}$, which gives $\mathbf{b}_n^u \approx \mathcal{B}_n(\mathbf{a}^*) + \frac{d\mathcal{B}_n(\mathbf{a}^*)}{d\mathbf{a}}(\mathbf{a}^u - \mathbf{a}^*)$. By the definition of $\mathbf{b}_n^u = \mathcal{B}_n(\mathbf{a}^u) = \operatorname{argmin}_{\mathbf{b}_n} l_n(\mathbf{a}^u, \mathbf{b}_n)$, we have $\mathbf{b}_n^u \approx \mathcal{B}_n(\mathbf{a}^u)$, i.e., $\mathbf{b}_n^u$ is the approximate optimum of $\operatorname{argmin}_{\mathbf{b}_n} l_n(\mathbf{a}^u, \mathbf{b}_n)$.

As a result, $\mathbf{a}^u$ and $\mathbf{b}^u$ satisfy the approximate machine unlearning criteria that $\nabla_{\mathbf{a}} f^{\setminus[S]}(\mathbf{a}^u, \mathbf{b}^u) = \nabla_{\mathbf{a}} \sum_{n\notin[S]} l_n(\mathbf{a}^u, \mathbf{b}^u) + 2\lambda\mathbf{a}^u \approx \mathbf{0}$ and $\sum_{n\notin[S]} \nabla_{\mathbf{b}} l_n(\mathbf{a}^u, \mathbf{b}^u) \approx 0$. □

REFERENCES

[1] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 141–159.

[2] Jonathan Brophy and Daniel Lowd. 2021. Machine unlearning for random forests. In *International Conference on Machine Learning*. PMLR, 1092–1104.

[3] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy (SP)*. IEEE, 463–480.

[4] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. 2022. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*. 2768–2777.

[5] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 499–513.

[6] Eli Chien, Chao Pan, and Olgica Milenkovic. 2022. Certified graph unlearning. *arXiv preprint arXiv:2206.09140* (2022).

[7] Alessio Del Bue, Joao Xavier, Lourdes Agapito, and Marco Paladini. 2012. Bilinear Modeling via Augmented Lagrange Multipliers (BALM). *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 8 (2012), 1496–1508. https://doi.org/10.1109/TPAMI.2011.238

[8] General Data Protection Regulation 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016.

[9] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. 2019. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems* 32 (2019).

[10] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. 2021. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 792–801.

[11] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision*. Springer, 383–398.

[12] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. 2001. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval* 4, 2 (2001), 133–151.

[13] Suriya Gunasekar, Joyce C Ho, Joydeep Ghosh, Stephanie Kreml, Abel N Kho, Joshua C Denny, Bradley A Malin, and Jimeng Sun. 2016. Phenotyping using Structured Collective Matrix Factorization of Multi–source EHR Data. *arXiv preprint arXiv:1609.04466* (2016).

[14] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2019. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030* (2019).

[15] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[16] Je Hyeong Hong and Andrew Fitzgibbon. 2015. Secrets of matrix factorization: Approximations, numerics, manifold optimization and random restarts. In *Proceedings of the IEEE International Conference on Computer Vision*. 4130–4138.

[17] Haoji Hu, Haowen Lin, and Yao-Yi Chiang. 2022. Clustering Human Mobility with Multiple Spaces. In *IEEE International Conference on Big Data, Big Data 2022, Osaka, Japan, December 17-20, 2022*, Shusaku Tsumoto, Yukio Ohsawa, Lei Chen, Dirk Van den Poel, Xiaohua Hu, Yoichi Motomura, Takuya Takagi, Lingfei Wu, Ying Xie, Akihiro Abe, and Vijay Raghavan (Eds.). IEEE, 575–584.

[18] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.

[19] Antonios Karatzoglou, Stefan Christian Lamp, and Michael Beigl. 2017. Matrix factorization on semantic trajectories for predicting future semantic locations. In *2017 IEEE 13th international conference on wireless and mobile computing, networking and communications (WiMob)*. IEEE, 1–7.

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[21] Haowen Lin and Yao-Yi Chiang. 2017. SRC: automatic extraction of phrase-level map labels from historical maps. *ACM SIGSPATIAL Special* 9, 3 (2017), 14–15.

[22] Wenyan Liu, Juncheng Wan, Xiaoling Wang, Weinan Zhang, Dell Zhang, and Hang Li. 2022. Forgetting Fast in Recommender Systems. *arXiv preprint arXiv:2208.06875* (2022).

[23] Yuan Luo and Chengsheng Mao. 2020. ScanMap: supervised confounding aware non-negative matrix factorization for polygenic risk modeling. In *Machine learning for healthcare conference*. PMLR, 27–45.

[24] Yuan Luo, Chengsheng Mao, Yiben Yang, Fei Wang, Faraz S Ahmad, Donna Arnett, Marguerite R Irvin, and Sanjiv J Shah. 2018. Integrating hypertension phenotype and genotype with hybrid non-negative matrix factorization. In *Machine Learning for Healthcare Conference*. PMLR, 102–118.

[25] Jing Ma, Qiuchen Zhang, Jian Lou, Joyce C Ho, Li Xiong, and Xiaoqian Jiang. 2019. Privacy-preserving tensor factorization for collaborative health data analysis. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1291–1300.

[26] Jing Ma, Qiuchen Zhang, Jian Lou, Joyce C Ho, Li Xiong, and Xiaoqian Jiang. 2019. Privacy-preserving tensor factorization for collaborative health data analysis. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1291–1300.

[27] Jing Ma, Qiuchen Zhang, Jian Lou, Li Xiong, Sivasubramanium Bhavani, and Joyce C Ho. 2021. Communication efficient tensor factorization for decentralized healthcare networks. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1216–1221.

[28] Jing Ma, Qiuchen Zhang, Jian Lou, Li Xiong, and Joyce C Ho. 2021. Communication efficient federated generalized tensor factorization for collaborative health data analytics. In *Proceedings of the Web Conference 2021*. 171–182.

[29] Ronak Mehta, Sourav Pal, Vikas Singh, and Sathya N Ravi. 2022. Deep Unlearning via Randomized Conditionally Independent Hessians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10422–10431.

[30] Aeron Buchanan Morgan. 2004. Investigation into matrix factorization when elements are unknown. *Vis. Geometry Group, Dept. Eng. Sci., Univ. Oxford, Oxford, UK, Tech. Rep* (2004).

[31] Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks* 12, 1 (1999), 145–151.

[32] Xun Ran, Yong Wang, Leo Yu Zhang, and Jun Ma. 2022. A differentially private nonnegative matrix factorization for recommender system. *Information Sciences* 592 (2022), 21–35.

[33] Yifei Ren, Jian Lou, Li Xiong, and Joyce C Ho. 2020. Robust irregular tensor factorization and completion for temporal health data analysis. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1295–1304.

[34] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[35] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. 2021. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems* 34 (2021), 18075–18086.

[36] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. 2022. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 303–319.

[37] Cheng-Hao Tsai, Chieh-Yen Lin, and Chih-Jen Lin. 2014. Incremental and decremental training for linear classification. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 343–352.

[38] René Vidal and Richard Hartley. 2004. Motion segmentation with missing data using powerfactorization and gpca. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 2. IEEE, II–II.

[39] Yaqing Wang, Quanming Yao, and James Kwok. 2021. A Scalable, Adaptive and Sound Nonconvex Regularizer for Low-rank Matrix Learning. In *Proceedings of the Web Conference 2021*. 1798–1808.

[40] Yinjun Wu, Edgar Dobriban, and Susan Davidson. 2020. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*. PMLR, 10355–10366.

[41] Kun Xie, Lele Wang, Xin Wang, Gaogang Xie, Guangxing Zhang, Dongliang Xie, and Jigang Wen. 2015. Sequential and adaptive sampling for matrix completion in network monitoring systems. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2443–2451.

[42] Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. 2022. ARCANE: An Efficient Architecture for Exact Machine Unlearning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 4006–4013. https://doi.org/10.24963/ijcai.2022/556 Main Track.

[43] Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu. 2015. NationTelescope: Monitoring and visualizing large-scale collective behavior in LBSNs. *Journal of Network and Computer Applications* 55 (2015), 170–180.

[44] Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2016. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 3 (2016), 1–23.

[45] Dingqi Yang, Daqing Zhang, Bingqing Qu, and Philippe Cudré-Mauroux. 2016. PrivCheck: Privacy-preserving check-in data publishing for personalized location based services. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 545–556.

[46] Guojun Zhang, Kaiwen Wu, Pascal Poupart, and Yaoliang Yu. 2020. Newton-type methods for minimax optimization. *arXiv preprint arXiv:2006.14592* (2020).

[47] Yin Zhang, Matthew Roughan, Walter Willinger, and Lili Qiu. 2009. Spatiotemporal compressive sensing and internet traffic matrices. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. 267–278.