Multi-Task Differential Privacy Under Distribution Skew

Walid Krichene [1] **Prateek Jain** [1] **Shuang Song** [1] **Mukund Sundararajan** [2]
Abhradeep Thakurta [1] **Li Zhang** [1]

Abstract

We study the problem of multi-task learning under user-level differential privacy, in which n users contribute data to m tasks, each involving a subset of users. One important aspect of the problem, that can significantly impact quality, is the distribution skew among tasks. Tasks that have much fewer data samples than others are more susceptible to the noise added for privacy. It is natural to ask whether algorithms can adapt to this skew to improve the overall utility.

We give a systematic analysis of the problem, by studying how to optimally allocate a user's privacy budget among tasks. We propose a generic algorithm, based on an adaptive reweighting of the empirical loss, and show that in the presence of distribution skew, this gives a quantifiable improvement of excess empirical risk.

Experimental studies on recommendation problems that exhibit a long tail of small tasks, demonstrate that our methods significantly improve utility, achieving the state of the art on two standard benchmarks.

1. Introduction

Machine learning models trained on sensitive user data present the risk of leaking private user information (Dwork et al., 2007; Korolova, 2010; Calandrino et al., 2011; Shokri et al., 2017). Differential Privacy (DP) (Dwork et al., 2006) mitigates this risk, and has become a gold standard widely adopted in industry and government (Abowd, 2018; Wilson et al., 2020; Rogers et al., 2021; Amin et al., 2022).

In this work, we adopt the notion of user-level DP (Dwork & Roth, 2014; Kearns et al., 2014), which seeks to protect *all data samples* of a given user, a harder goal than protecting a single sample. User-level DP has been studied under different settings, including empirical risk minimization (Amin et al., 2019; Levy et al., 2021), mean estimation (Cummings et al., 2022), and matrix completion (Jain et al., 2018; Chien et al., 2021). The setting we study is that of multi-task learning, in which users contribute data to multiple tasks. This is for example the case in recommendation systems: given a set of n users and m items (pieces of content, such as movies or songs), the goal is to learn a representation for each item based on the users' preferences. Another example is multi-class classification, where the goal is to learn m classifiers (one per class). Notice that the training data of a given user may be relevant to only a small subset of tasks. This is certainly the case in recommendation, where users typically interact with a small subset of the catalog of available content. This can also be the case in classification if the number of classes is very large. The interaction pattern between users and tasks can be described using a bipartite graph $\Omega \subseteq [m] \times [n]$, such that user j contributes data to task i if and only if $(i, j) \in \Omega$. We will also denote by $\Omega_i = \{j : (i, j) \in \Omega\}$ the set of users contributing to task i. The goal is then to minimize the empirical loss summed over tasks,

$$L(\theta) = \sum_{i=1}^{m} \sum_{j \in \Omega_i} \ell(\theta_i; x_{ij}, y_{ij}), \tag{1}$$

where θ_i represents the model parameters for task i, and x_{ij}, y_{ij} are respectively the features and labels contributed by user j to task i. In general, we will assume that Ω, x, and y are all private.

In practice, the tasks can be imbalanced, i.e. the distribution of task sizes $|\Omega_i|$ can be heavily skewed. This is the case in recommendation systems, in which there is a long tail of items with orders of magnitude fewer training examples than average (Yin et al., 2012; Liu & Zheng, 2020). Classification tasks can also exhibit a similar skew among classes (Kubat & Matwin, 1997). Under differential privacy constraints, this skew makes tail tasks (those with less data) more susceptible to quality losses. This disparate impact of differential privacy is well documented, for example in classification with class imbalance (Bagdasaryan et al., 2019) or in DP language models (McMahan et al., 2018).

A natural question is whether DP algorithms can be made adaptive to such distribution skew. Some heuristics were

[1]Google Research [2]Google. Correspondence to: Walid Krichene <walidk@google.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

found to be useful in practice. For example, (Chien et al., 2021) propose a sampling heuristic, which limits the number of samples per user, while biasing the sampling towards tail tasks. This was found to improve performance in practice (compared to uniform sampling), though no analysis was provided on the effect of this biased sampling. Another heuristic was proposed by (Xu et al., 2021), which applies to DPSGD, and operates by increasing the clipping norm of gradients on tail tasks. Similarly, this was observed to improve performance (compared to uniform clipping).

In both cases, the intuition is that by allocating a larger proportion of a user's privacy budget to the tail (either via sampling or clipping), one can improve utility on the tail, and potentially on the overall objective (1). However, a formal analysis of biased sampling and clipping remains elusive, as it introduces a bias that is hard to analyze. We take a different approach, by introducing task-user weights w_{ij} to have a finer control over the budget allocation. The weights play a similar role to sampling or clipping: by assigning larger weights to a task, we improve utility of that task. The question is then how to optimally choose weights w to minimize excess risk on the overall objective (1), under privacy constraints (which translate into constraints involving w and Ω). Fortunately, this problem is amenable to formal analysis, and we derive optimal choices of weights, and corresponding error bounds, under different assumptions on the loss and the task-user graph Ω.

1.1. Contributions

1. We give a formal analysis of the private multi-task learning problem under distribution skew. We propose a generic method based on computing task-user weights, and applying those weights to the loss function (1), as a mechanism to control each user's privacy budget allocation among tasks. The method is generic in that it applies to several algorithms such as DPSGD (Bassily et al., 2014; Abadi et al., 2016) or Sufficient Statistics Perturbation (Foulds et al., 2016).
2. We derive utility bounds that explicitly show how the budget allocation trades-off privacy and utility (Theorems 3.8 and 3.12). By adapting the allocation to the task sizes, we obtain utility bounds (Theorem 4.3) with a guaranteed improvement compared to uniform allocation. The improvement increases with the degree of skew in the task distribution.
3. We conduct experiments on synthetic data and two standard recommendation benchmarks which exhibit a heavy distribution skew. Our methods significantly improve accuracy, even compared to strong baselines such as the tail sampling heuristic of (Chien et al., 2021). We also provide a detailed analysis of quality impact across tasks. We find that adaptive budget allocation visibly improves quality on tail tasks, with a limited impact on head tasks.

1.2. Related Work

A common technique in user-level differential privacy is to adaptively control the sensitivity with respect to the data of a user. This is typically done via clipping (Abadi et al., 2016), sampling (Kasiviswanathan et al., 2013; Amin et al., 2019), using privacy filters (Feldman & Zrnic, 2021), or using weights in the objective function (Proserpio et al., 2014; Epasto et al., 2020) to scale the contribution of high sensitivity users. Our approach is related, in that we adaptively control sensitivity. But while these methods focus on adapting to *users* (users with high sensitivity are assigned lower weights), we focus on adapting to *tasks* (within each user, easier tasks are assigned lower weights).

We mention in particular (Epasto et al., 2020), who use weights to bound user sensitivity, and analyze the optimal choice of weights under certain assumptions. Their approach is perhaps the most similar to ours, in that the choice of weights is phrased as an optimization problem. But our setting is inherently different: they consider the traditional ERM setting with a single task, while we consider the multi-task setting (with heterogeneous tasks). As a result, their solution assigns identical weights to all examples of a given user, while our solution is designed to allocate a user's budget among tasks, and generally assigns different weights to different examples of a user.

Turning to the multi-task setting, prior work on DP multi-task learning (Li et al., 2020; Hu et al., 2021) has focused on *task-level* privacy, where every user is identified with one task. We consider a more general setting, in which tasks do not necessarily coincide with users. In particular, under our model, it's important that a user can contribute to more than one task.

Another related problem is that of answering multiple linear or convex queries under DP, e.g. (Dwork et al., 2010; Hardt & Rothblum, 2010; Ullman, 2015). Though related, the two problems are different because the utility measure is inherently different: in the multi-query setting, the goal is to answer *all* queries accurately (i.e. a minimax objective), while in the multi-task setting, the utility is a sum across tasks, as in (1). Besides, algorithms in the multi-query convex setting (closest to ours) often have to maintain a distribution over a discretization of the data space, making them prohibitively expensive in continuous feature spaces, due to an exponential dependence on the dimension.

In practice, budget allocation among tasks is often done through uniform sampling, for example in DP SQL (Wilson et al., 2020), LinkedIn's Audience Engagement API (Rogers et al., 2021), and Plume (Amin et al., 2022). The techniques that do adapt to the task distribution (via sampling as in (Chien et al., 2021) or clipping as in (Xu et al., 2021)) tend to be heuristic in nature, and lack a rigorous utility analysis.

To the best of our knowledge, we present the first systematic analysis of multi-task, user-level DP under distribution skew. Our method operates directly on the loss function, which makes it applicable to standard algorithms such as DPSGD, and has a guaranteed improvement under distribution skew.

2. Preliminaries

2.1. Notation

Throughout the paper, we will denote by $\Omega \subseteq [m] \times [n]$ the task-user membership graph, where m, n are the number of tasks and users, respectively. We assume that $m \leq n$. For all i, let $\Omega_i := \{j : (i,j) \in \Omega\}$ be the set of users who contribute to task i, and for all j, let $\Omega^j := \{i : (i,j) \in \Omega\}$ be the set of tasks to which j contributes data. We will denote by $n_i = |\Omega_i|$, and by $n^j = |\Omega^j|$. Let $\|x\|$ be the L_2 norm of a vector x, and $\|X\|_F$ be the Frobenius norm of a matrix X. For a symmetric matrix A, let $A^\dagger$ denote the pseudo-inverse of A's projection on the PSD cone. For a closed convex set $\mathcal{C}$, let $\|\mathcal{C}\|$ denote its Euclidean diameter, and $\Pi_{\mathcal{C}}(\cdot)$ denote the Euclidean projection on $\mathcal{C}$. Let $\mathcal{N}^d$ denote the multivariate normal distribution (of zero mean and unit variance), and $\mathcal{N}^{d\times d}$ denote the distribution of symmetric $d \times d$ matrices whose upper triangle entries are i.i.d. normal. Finally, whenever we state that a bound holds with high probability (abbreviated as w.h.p.), we mean with probability $1 - 1/n^c$, where c is any positive constant, say 2.

2.2. User-Level Differential Privacy

Let $\mathcal{D}$ be a domain of data sets. Two data sets $D, D' \in \mathcal{D}$ are called neighboring data sets if one is obtained from the other by removing all of the data samples from a single user. Let $\mathcal{A} : \mathcal{D} \to \mathcal{S}$ be a randomized algorithm with output space $\mathcal{S}$.

Definition 2.1 (User-Level Differential Privacy (Dwork & Roth, 2014))**.** Algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private (DP) if for all neighboring data sets $D, D' \in \mathcal{D}$, and all measurable $S \in \mathcal{S}$,

$$\Pr\left(\mathcal{A}(D) \in S\right) \leq e^{\epsilon} \Pr\left(\mathcal{A}(D') \in S\right) + \delta.$$

Remark 2.2. The data of a user j consists of Ω^j (the set of tasks that user contributes to), together with the set of features and labels $\{(x_{ij}, y_{ij}) : i \in \Omega^j\}$. Notice that the user-task graph Ω itself is private. For example, in the recommendation setting, Ω represents *which* pieces of content each user interacted with, while the labels y_{ij} represent *how much* a user liked that piece of content; we want to protect both.

For simplicity of presentation, our results will be stated in terms of Rényi differential privacy (RDP). Translation from RDP to DP is standard, and can be done for example using (Mironov, 2017, Proposition 3).

Definition 2.3 (User-Level Rényi Differential Privacy (Mironov, 2017))**.** An algorithm $\mathcal{A}$ is (α, ρ)-Rényi differentially private (RDP) if for all neighboring data sets $D, D' \in \mathcal{D}$,

$$D_\alpha\left(\mathcal{A}(D)\right)||\mathcal{A}(D')) \leq \rho,$$

where D_α is the Rényi divergence of order α.

3. Budget Allocation via Weights

First, observe that the objective function (1) can be decomposed as

$$L(\theta) = \sum_{i=1}^{m} L_i(\theta_i); \quad L_i(\theta_i) = \sum_{j\in\Omega_i} \ell(\theta_i; x_{ij}, y_{ij}), \quad (2)$$

where $\theta_i \in \mathbb{R}^d$ are the model parameters for task i, and θ is a shorthand for $(\theta_1, \ldots, \theta_m)$. For all i, we will denote the non-private solution of task i by

$$\theta_i^* = \operatorname*{argmin}_{\theta_i} L_i(\theta_i).$$

The goal is to learn a private solution $\hat{\theta}$ under user-level DP. The quality of $\hat{\theta}$ will be measured in terms of the excess empirical risk,

$$\sum_{i=1}^{m} \mathbb{E}[L_i(\hat{\theta}_i)] - L_i(\theta_i^*),$$

where the expectation is over randomness in the algorithm.

Remark 3.1. Although the objective is decomposable as a sum of $L_i(\theta_i)$, the problems are coupled through the privacy constraint: since user j contributes data to all tasks in Ω^j, the *combined sensitivity* of these tasks with respect to user j's data should be bounded. One is then faced with the question of how to allocate the sensitivity among tasks. This will be achieved by using weights in the objective function.

Our general approach will be to apply a DP algorithm to a weighted version of the objective $L(\theta)$. Let $w := (w_{ij})_{(i,j)\in\Omega}$ be a collection of arbitrary positive weights, and define the *weighted multi-task objective*

$$L_w(\theta) = \sum_{i=1}^{m} \sum_{j\in\Omega_i} w_{ij} \ell(\theta_i; x_{ij}, y_{ij}). \quad (3)$$

We first give some intuition on the effect of the weights (a formal analysis will be given in the next section). Suppose $\nabla\ell$ is bounded, and that we compute $\hat{\theta}$ by applying noisy gradient descent (with fixed noise standard deviation) to $L_w(\theta)$. Then w will have the following effect on privacy and utility: increasing the weights of task i will generally improve its utility (since it increases the norm of task i's

Algorithm 1 Task-weighted SSP (Sufficient Statistic Perturbation)

1: **Inputs**: Task-user graph Ω, user features and labels $(x_{ij}, y_{ij})_{(i,j)\in\Omega}$, clipping parameters Γ_x, Γ_*, weights w, domain $\mathcal{C}$.
2: For all i, j, $x_{ij} \leftarrow \text{clip}(x_{ij}, \Gamma_x)$, $y_{ij} \leftarrow \text{clip}(y_{ij}, \Gamma_x\Gamma_*)$, where $\text{clip}(x, \Gamma) = x \min(1, \Gamma/\|x\|)$
3: **for** $1 \le i \le m$ **do**
4: Sample $\xi_i \sim \mathcal{N}^d$, $\Xi_i \sim \mathcal{N}^{d\times d}$
5: $\hat{A}_i \leftarrow \sum_{j\in\Omega_i} w_{ij}(x_{ij}x_{ij}^\top + \lambda I) + \Gamma_x^2 \Xi_i$
6: $\hat{b}_i \leftarrow \sum_{j\in\Omega_i} w_{ij} y_{ij} x_{ij} + \Gamma_x^2 \Gamma_* \xi_i$
7: $\hat{\theta}_i \leftarrow \hat{A}_i^\dagger \hat{b}_i$
8: **end for**
9: **Return** $\hat{\theta}$

gradients, hence increases its signal-to-noise ratio). Increasing the weights of user j will decrease her privacy (since it increases the norm of user j's gradients, hence increases the L_2 sensitivity w.r.t. her data). Utility depends on task weights $(w_{ij})_{j\in\Omega_i}$, while sensitivity/privacy depend on user weights $(w_{ij})_{i\in\Omega^j}$. By characterizing how sensitivity and utility precisely scale in terms of the weights w, we will be able to choose w that achieves the optimal trade-off.

Remark 3.2. Although we will apply a DP algorithm to the weighted objective L_w, utility will always be measured in terms of the original, unweighted objective L. The weights should be thought of as a tool to allocate users' privacy budget, rather than a change in the objective function.

3.1. Weighted Multi-Task Ridge Regression

We start by analyzing the case of multi-task ridge regression, optimized using the Sufficient Statistics Perturbation (SSP) method (Foulds et al., 2016; Wang, 2018). This will pave the way to the more general convex setting in the next section.

Suppose that each task is a ridge regression problem. The multi-task objective (2) becomes

$$L(\theta) = \sum_{i=1}^{m} \sum_{j\in\Omega_i} \left[(\theta_i \cdot x_{ij} - y_{ij})^2 + \lambda\|\theta_i\|^2\right], \quad (4)$$

where λ is a regularization constant. The exact solution of (4) is given by $\theta_i^* = A_i^{-1} b_i$ where

$$A_i = \sum_{j\in\Omega_i} (x_{ij}x_{ij}^\top + \lambda I), \quad b_i = \sum_{j\in\Omega_i} y_{ij} x_{ij}. \quad (5)$$

We propose a task-weighted version of SSP, described in Algorithm 1. The private solution $\hat{\theta}_i$ is obtained by forming *weighted and noisy* estimates $\hat{A}_i, \hat{b}_i$ of the sufficient statistics (5), then returning $\hat{\theta}_i = \hat{A}_i^\dagger \hat{b}_i$. Notice that if we use larger weights for task i, the *relative noise scale* in $\hat{A}_i, \hat{b}_i$ becomes smaller (see Lines 5-6), which improves the quality of the estimate, while also increasing the sensitivity. Once we analyze how the weights affect privacy and utility, we will be able to reason about an optimal choice of weights.

First, we state the privacy guarantee of Algorithm 1. All proofs are deferred to Appendix A.

Theorem 3.3 (Privacy guarantee of Algorithm 1)**.** *Let $(w_{ij})_{(i,j)\in\Omega}$ be a collection of task-user weights, and let $\beta = \max_{j\in[n]} \sum_{i\in\Omega^j} w_{ij}^2$. Then Algorithm 1 run with weights w is $(\alpha, \alpha\beta)$-RDP for all $\alpha > 1$.*

The result can be easily translated to traditional DP, for example, for all $\epsilon, \delta > 0$ with $\epsilon \le \log(1/\delta)$, if $\beta \le \frac{\epsilon^2}{8\log(1/\delta)}$, then the algorithm is (ϵ, δ)-DP.

Remark 3.4. The theorem can be interpreted as follows: each user j has a total budget β, that can be allocated among the tasks in Ω^j by choosing weights such that $\sum_{i\in\Omega^j} w_{ij}^2 \le \beta$. For this reason, we will refer to β as the *user RDP budget*.

Remark 3.5. Sampling a fixed number of tasks per user (as done in (Chien et al., 2021)) is a special case of this formulation, where w_{ij} is a constant if task i is sampled for user j, and 0 otherwise. Using general weights allows a finer trade-off between tasks, as we shall see next. There is also a practical advantage to using weights: sampling methods require choosing a maximum number of tasks to sample per user (a hyper-parameter), this number is used in the privacy accounting to determine user sensitivity. One needs to tune this hyper-parameter carefully–if it's too large, many users are below the threshold and we end up adding too much noise, and if it's too small, we might drop too many samples. When using weights, there is no such hyper-parameter (we keep all samples of all users), which simplifies parameter tuning.

For the utility analysis, we will make the following standard assumption.

Assumption 3.6. We assume that there exist $\Gamma_x, \Gamma_* > 0$ such that for all i, j, $\|x_{ij}\| \le \Gamma_x$, $\|\theta_i^*\| \le \Gamma_*$, and $\|y_{ij}\| \le \Gamma_x\Gamma_*$.

Remark 3.7. We state the result in the ridge regression case with bounded data to simplify the presentation. Similar results can be obtained under different assumptions, for instance when x_{ij} are i.i.d. Gaussian, in which case one can bound the minimum eigenvalue of the covariance matrices A_i without the need for regularization, see (Wang, 2018) for a detailed discussion.

Theorem 3.8 (Utility guarantee of Algorithm 1)**.** *Suppose Assumption 3.6 holds. Let $\omega \in \mathbb{R}^m$ be a vector of positive weights such that for all j, $\sum_{i\in\Omega^j} \omega_i^2 \le \beta$. Let $\hat{\theta}$ be the output of Algorithm 1 with weights*[1] *$w_{ij} = \omega_i \; \forall (i,j) \in \Omega$.*

[1]The choice of $w_{ij} = \omega_i$ (all examples of a task share the same weights) simplifies the proof, and is sufficient to improve utility under skew, see Section 4.

Algorithm 2 Task-weighted noisy gradient descent

Inputs: Task-user graph Ω, user features and labels x, y, domain $\mathcal{C}$, clipping parameter Γ, weights w, initial parameters $\hat{\theta}^{(0)}$, numer of steps T, learning rates $\eta_i^{(t)}$.
for $1 \le i \le m$ **do**
  for $1 \le t \le T$ **do**
    Sample $\xi_i^{(t)} \sim \mathcal{N}^d$
    $g_i^{(t)} \leftarrow \text{clip}(\sum_{j\in\Omega_i} w_{ij}\nabla_{\theta_i}\ell(\theta_i^{(t-1)}; x_{ij}, y_{ij}), \Gamma)$
    $\hat{\theta}_i^{(t)} \leftarrow \Pi_{\mathcal{C}}[\theta_i^{(t-1)} - \eta_i^{(t)}(g_i^{(t)} + \Gamma\sqrt{T/2}\cdot\xi_i^{(t)})]$
  end for
end for
Return $\hat{\theta}^{(T)} = (\hat{\theta}_1^{(T)}, \dots, \hat{\theta}_m^{(T)})$

Then Algorithm 1 is $(\alpha, \alpha\beta)$-RDP, for all $\alpha > 1$, and w.h.p.,

$$L(\hat{\theta}) - L(\theta^*) = \mathcal{O}\left(\frac{\Gamma_x^4\Gamma_*^2 d}{\lambda}\sum_{i=1}^m \frac{1}{\omega_i^2 n_i}\right). \tag{6}$$

Theorem 3.8 highlights that under the user-level privacy constraint ($\max_j \sum_{i\in\Omega^j}\omega_i^2 \le \beta$), there is a certain trade-off between tasks: by increasing ω_i we improve quality for task i, but this may require decreasing ω for other tasks to preserve a constant total privacy budget. This allows us to reason about how to choose weights that achieve the optimal trade-off under a given RDP budget β. Remarkably, this problem is convex:

$$\min_{\omega\in\mathbb{R}_+^d}\sum_{i=1}^m \frac{1}{\omega_i^2 n_i} \quad \text{s.t.} \sum_{i\in\Omega^j}\omega_i^2 \le \beta \quad \forall j \in \{1, \dots, n\}.$$

If the task-user graph Ω were public, one could solve the problem exactly. But since task-user membership often represents private information, we want to protect Ω. Our approach will be to compute an approximate solution. Before tackling this problem, we first derive similar weight-dependent privacy and utility bounds for noisy gradient descent.

3.2. Weighted Multi-Task Convex Minimization

We now consider the more general problem of minimizing empirical risk for convex losses in the multi-task setting. Our method applies noisy GD (Bassily et al., 2014) to the weighted objective (3). This is summarized in Algorithm 2. We will consider two standard settings, in which noisy GD is known to achieve nearly optimal empirical risk bounds:

Assumption 3.9 (Lipschitz convex on a bounded domain)**.** Assume that $\mathcal{C}$ is a bounded convex domain, and that there exists $\Gamma > 0$ such that $\ell(\theta; x, y)$ is a convex and Γ-Lipschitz function of $\theta \in \mathcal{C}$, uniformly in x, y.

Assumption 3.10 (Lipschitz, strongly convex)**.** Assume that there exist $\lambda, \Gamma > 0$ such that $\ell(\theta; x, y)$ is a λ-strongly convex and Γ-Lipschitz function of θ, uniformly in x, y.

Next, we derive privacy and utility bounds.

Theorem 3.11 (Privacy guarantee of Algorithm 2)**.** *Let $(w_{ij})_{(i,j)\in\Omega}$ be a collection of task-user weights, and let $\beta = \max_{j\in[n]}\sum_{i\in\Omega^j} w_{ij}^2$. Then Algorithm 2 is $(\alpha, \alpha\beta)$-RDP for all $\alpha > 1$.*

Theorem 3.12 (Utility guarantee of Algorithm 2)**.** *Let $\omega \in \mathbb{R}^m$ be a vector of positive weights such that for all j, $\sum_{i\in\Omega^j}\omega_i^2 \le \beta$. Let $\hat{\theta}$ be the output of Algorithm 2 with weights $w_{ij} = \omega_i$. Then Algorithm 2 is $(\alpha, \alpha\beta)$-RDP, for all $\alpha > 1$. Furthermore,*

(a) Under Assumption 3.9, if $T = \frac{2}{d}\frac{\sum_{i=1}^m n_i^2}{\sum_{i=1}^m 1/\omega_i^2}$ and $\eta_i^{(t)} = \frac{\|C\|}{\Gamma\sqrt{\omega_i^2 n_i^2 + Td/2}}\frac{1}{\sqrt{t}}$, then

$$\mathbb{E}[L(\theta)] - L(\theta^*) = \tilde{\mathcal{O}}\left(\|\mathcal{C}\|\Gamma\sqrt{md}\sqrt{\sum_{i=1}^m \frac{1}{\omega_i^2}}\right), \tag{7}$$

where $\tilde{\mathcal{O}}$ hides polylog factors in n_i.

(b) Under Assumption 3.10, if $T = \frac{2}{d}\frac{|\Omega|}{\sum_{i=1}^m 1/\omega_i^2 n_i}$, and $\eta_i^{(t)} = \frac{1}{\omega_i^2 n_i \lambda t}$, then

$$\mathbb{E}[L(\theta)] - L(\theta^*) = \tilde{\mathcal{O}}\left(\frac{\Gamma^2 d}{\lambda}\sum_{i=1}^m \frac{1}{\omega_i^2 n_i}\right). \tag{8}$$

We consider some special cases, as a consistency check.

Single task Suppose there is a single task ($m = 1$), then the RDP budget constraint is simply satisfied with $\omega_1^2 = \beta$. Plugging this into the bounds from the theorem, and dividing by the number of examples n (as our loss L is a summation, instead of average, over all examples), we get that the average empirical risk is bounded by $\tilde{\mathcal{O}}\left(\frac{\|\mathcal{C}\|\Gamma\sqrt{d}}{n\sqrt{\beta}}\right)$ and $\tilde{\mathcal{O}}\left(\frac{\Gamma^2 d}{\lambda n^2\beta}\right)$ in the convex and strongly convex case, respectively. This recovers the usual convex ERM bounds (Bassily et al., 2014) – with the correspondence $\beta = \mathcal{O}\left(\epsilon^2/\log(1/\delta)\right)$.

Complete bipartite Ω Suppose there are m tasks and each user participates in *all* tasks. Then the budget constraint becomes $m\omega_i^2 = \beta$ for all i. Plugging this into the bounds and dividing by mn (total number of examples), we get that the average empirical risk is $\tilde{\mathcal{O}}\left(\frac{\|\mathcal{C}\|\Gamma\sqrt{md}}{n\sqrt{\beta}}\right)$ and $\tilde{\mathcal{O}}\left(\frac{\Gamma^2 md}{\lambda n^2\beta}\right)$ in the convex and strongly convex case, respectively. Notice that this is equivalent to solving a single task in dimension md, and the resulting ERM bound is the same.

4. Optimizing Weights Under Distribution Skew

Equipped with the utility bounds of the previous section, we can ask what choice of weights minimizes the error bound under a given privacy budget. Observe that all utility bounds (6), (7), (8) are increasing in the quantity $\sum_{i=1}^m \frac{1}{\omega_i^2 n_i^\gamma}$, where $\gamma = 0$ in the convex case, and $\gamma = 1$ in the strongly convex case (for both SSP and noisy GD). This amounts to solving a constrained optimization problem of the form

$$\min_{\omega \in \mathbb{R}_+^m} \sum_{i=1}^m \frac{1}{\omega_i^2 n_i^\gamma} \quad \text{s.t.} \sum_{i \in \Omega^j} \omega_i^2 \leq \beta, \quad \forall j \in \{1, \dots, n\}. \tag{9}$$

In general, the task-user graph Ω is private (see Remark 2.2). We cannot solve (9) exactly, but under distributional assumptions on Ω, we can solve it approximately.

In modeling the distribution of Ω, we take inspiration from the closely related problem of matrix completion, where a standard assumption is that elements of Ω are sampled uniformly at random, see e.g. (Jain et al., 2013) – in other words, Ω is an Erdős-Rényi random graph. However, making such assumption implies that the task sizes $n_i = |\Omega_i|$ concentrate around their mean, and this fails to capture the task-skew problem we set out to solve. Instead, we will relax the assumption by removing uniformity of tasks while keeping uniformity of users. Specifically,

Assumption 4.1. Assume that Ω is obtained by sampling, for each task i, n_i users independently and uniformly from $\{1, \dots, n\}$. Furthermore, assume that the task sizes n_i are publicly known, and that $n_i \geq 1$ for all i.

In practice, both users and tasks can be heterogeneous. In that sense, Assumption 4.1 is a stylized model, that is meant to primarily capture the task long tails. We leave extensions to the user heterogeneous case to future work.

The assumption that n_i are publicly known is relatively mild: in some applications, this information is available, for example in video recommendation, the number of views of each video is often public. A similar assumption was made by (Epasto et al., 2020), where the number of examples contributed by each user is assumed to be public. When n_i are not available, the solution can be based on private estimates of n_i; we derive in Appendix B the extension to this case. In experiments, we will always privately estimate n_i and account for the privacy cost of doing so.

A consequence of Assumption 4.1 is that the quantity $B_j := \sum_{i \in \Omega^j} \omega_i^2$ (that appears in the inequality constraint in equation (9)) becomes a random variable, that is concentrated around its mean, which is equal to $\sum_{i=1}^m \frac{n_i}{n} \omega_i^2$. More precisely, we will show that, w.h.p., all B_j are bounded above by $c(n) \cdot \sum_{i=1}^m \frac{n_i}{n} \omega_i^2$ for an appropriate $c(n)$. Then we can replace the n constraints in (9) by a single constraint (for the privacy guarantee, the constraints will be enforced with probability 1, as we will discuss below; but for the purpose of optimizing the utility bound, it suffices that the constraints hold w.h.p.). The problem becomes:

$$\min_{\omega \in \mathbb{R}_+^m} \sum_{i=1}^m \frac{1}{\omega_i^2 n_i^\gamma} \quad \text{s.t. } c(n) \sum_{i=1}^m \frac{n_i}{n} \omega_i^2 \leq \beta, \tag{10}$$

which we can solve in closed form as follows. Let $\bar{\beta} = \frac{\beta}{c(n)}$. The Lagrangian is $\sum_{i=1}^m \frac{1}{\omega_i^2 n_i^\gamma} + \lambda(\sum_{i=1}^m n_i \omega_i^2 - n\bar{\beta})$ (λ is a Lagrange multiplier), and the KKT conditions yield: $\forall i,\ -\frac{1}{\omega_i^3 n_i^\gamma} + \lambda n_i \omega_i = 0,\ \sum_{i=1}^m n_i \omega_i^2 = n\bar{\beta}$, which simplifies to

$$\omega_i^* = \frac{n_i^{-(\gamma+1)/4}}{\sqrt{\sum_{i'=1}^m n_{i'}^{(1-\gamma)/2} / n\bar{\beta}}}. \tag{11}$$

With ω_i in this form and $c(n) = c \log n$, it can be shown (see Lemma A.4) that w.h.p., $\max_j \sum_{i \in \Omega^j} \omega_i^2 \leq \beta$. But since we need the privacy guarantee to always hold, we simply add the following clipping step: let w^* be the task-user weights defined as

$$w_{ij}^* = \omega_i^* \min\left(1, \sqrt{\frac{\beta}{\sum_{i' \in \Omega_j} {\omega_{i'}^*}^2}}\right) \quad \forall i, j \in \Omega. \tag{12}$$

In other words, for each user j, if the total sensitivity $\sum_{i' \in \Omega_j} {\omega_{i'}^*}^2$ exceeds β, we scale down the weights of this user to meet the budget. This step ensures that the privacy guarantee is exact. But for utility analysis, we know that w.h.p., $w_{ij}^* = \omega_i$ (since the constraint is satisfied w.h.p.), so utility bounds will hold w.h.p.

Remark 4.2 (Weight clipping preserves privacy). Observe that from eq. (12), the clipped weights of user j only depend on ω^* (which is released differentially privately) and on Ω^j; it does not depend on data of other users j'. This means that weight clipping can be done separately for each user, much like gradient clipping in DPSGD. Furthermore, the clipped weights are never publicly released, the weight clipping is internal to each algorithm.

The next theorem provides the utility guarantee under weights w^*.

Theorem 4.3 (Privacy-utility trade-off of Algorithms 1 and 2 under adaptive weights)**.** *Suppose Assumption 4.1 holds.*

(a) Under Assumption 3.6, let $\hat{\theta}$ be the output of Algorithm 1 run with weights w^* (eq. (11)-(12) with $\gamma = 1$). Then w.h.p.,

$$L(\hat{\theta}) - L(\theta^*) = \tilde{\mathcal{O}}\left(\frac{\Gamma_x^4 \Gamma_*^2 d m^2}{n \lambda \beta}\right). \tag{13}$$

(b) Under Assumption 3.9, if Algorithm 2 is run with weights w^* (eq. (11)-(12) with $\gamma = 0$) and parameters listed in Theorem 3.12(a), then

$$\tilde{\mathbb{E}}[L(\theta)] - L(\theta^*) = \tilde{\mathcal{O}}\left(\|\mathcal{C}\|\Gamma\sqrt{\frac{md}{n\beta}}\sum_{i=1}^{m} n_i^{1/2}\right), \tag{14}$$

where $\tilde{\mathbb{E}}$ denotes expectation conditioned on a high probability event.

(c) Under Assumption 3.10, if Algorithm 2 is run with w^* (eq. (11)-(12) with $\gamma = 1$) and parameters listed in Theorem 3.12(b), then

$$\tilde{\mathbb{E}}[L(\theta)] - L(\theta^*) = \tilde{\mathcal{O}}\left(\frac{\Gamma^2 d m^2}{n\lambda\beta}\right). \tag{15}$$

In all three cases, the algorithm is $(\alpha, \alpha\beta)$-RDP for all $\alpha > 1$.

To understand the effect of adapting to task skew, we compare these bounds to the case when we use uniform weights. With uniform weights, the privacy constraint yields $\omega_i^{\text{uniform}} = (n\bar{\beta}/\sum_{i'=1}^{m} n_{i'})^{1/2}, \quad \forall i$. Taking the ratio between the utility bound for ω^{uniform} and the utility bound for ω^*, we obtain the following: In the Lipschitz bounded case, the relative improvement is $R_0(n_1, \ldots, n_m) = \frac{(m\sum_{i=1}^{m} n_i)^{1/2}}{\sum_{i=1}^{m} n_i^{1/2}}$. In the strongly convex case (both for SSP and noisy GD), the relative improvement is $R_1(n_1, \ldots, n_m) = \frac{\sum_{i=1}^{m}\frac{1}{n_i}\sum_{i=1}^{m} n_i}{m^2}$. In particular, R_0, R_1 are lower bounded by 1 (by Cauchy-Schwarz), and equal to 1 when the task distribution is uniform (all n_i are equal). In both cases, the improvement can be arbitrary large. To see this, consider the extreme case when $n_1 = m^{1+\nu}$ and n_i is a constant for all other $m-1$ tasks ($\nu > 0$). A simple calculation shows that in this case, the leading term as $m \to \infty$ is $R_0 \approx m^{\nu/2}$ and $R_1 \approx m^{\nu}$. Both are unbounded in m.

Finally, we give a qualitative comment on the optimal weights (11). While it's clear that increasing the weight of a task improves its quality, it was not clear, a priori, that increasing weights on tail tasks benefits the *overall objective* (1) (since tail tasks represent fewer terms in the sum). The analysis shows that this is indeed the case: the optimal trade-off (see eq. (11)) is obtained when $\omega_i \propto n_i^{-(1+\gamma)/4}$, i.e. larger weights are assigned to smaller tasks. This can be explained by a certain diminishing returns effect that depends on the task size: from the optimization problem (10), the marginal utility benefit of increasing ω_i is smaller for larger tasks (due to the n_i^{γ} term in the objective). At the same time, the marginal privacy cost of increasing ω_i is higher for larger tasks (due to the n_i term in the constraint).

5. Empirical Evaluation

To evaluate our methods, we run experiments[2] on large-scale recommendation benchmarks on the MovieLens data sets (Harper & Konstan, 2016). As mentioned in the introduction, recommendation problems are known to exhibit a long tail of content with much fewer training data than average, and this is the main practical issue we seek to address. We also run experiments on synthetic data, reported in Appendix C.

In our evaluation, we investigate the following questions:

1. Whether on realistic data sets, our adaptive weight methods can improve the overall utility by shifting weights towards the tail, as the analysis indicates. In particular, Theorem 4.3 suggests that (under distributional assumptions on Ω), the optimal task weights are of the form $\omega_i \propto n_i^{-(1+\gamma)/4}$. We experiment with weights of the form $\omega_i \propto n_i^{-\mu}$ for different values of $\mu \in [0, 1]$.
2. Do we observe similar improvements for different algorithms? Our analysis applies both to SSP (Algorithm 1) and noisy GD (Algorithm 2). We run the experiments with both algorithms.
3. Beyond improvements in the average metrics, what is the extent of quality impact across the head and tail of the task distribution?

Experimental setup Each of the MovieLens data sets consists of a sparse matrix of ratings given by users to movies. The first benchmark, from Lee et al. (2013), is a rating prediction task on MovieLens 10 Million (abbreviated as ML10M), where the quality is evaluated using the RMSE of the predicted ratings. The second benchmark, from (Liang et al., 2018), is a top-k item recommendation task on MovieLens 20 Million (abbreviated as ML20M), where the model is used to recommend k movies to each user, and the quality is measured using Recall@k. Figure 7 in the appendix shows the movie distribution skew in each data set. For example, in ML20M, the top 10% movies account for 86% of the training data.

In both cases, we train a DP matrix factorization model. We identify movies to tasks and apply our algorithms to learning the movie embedding representations, for details, see Appendix D.

The current state of the art on these benchmarks is the DP alternating least squares (DPALS) method (Chien et al., 2021), which we use as a baseline. We will compare to DPALS both with uniform sampling (sample a fixed number of movies per user, uniformly at random) and with tail-biased sampling (sort all movies by increasing counts, then for each

[2]The code is available at the following repository: https://github.com/google-research/google-research/tree/master/dp_alternating_minimization

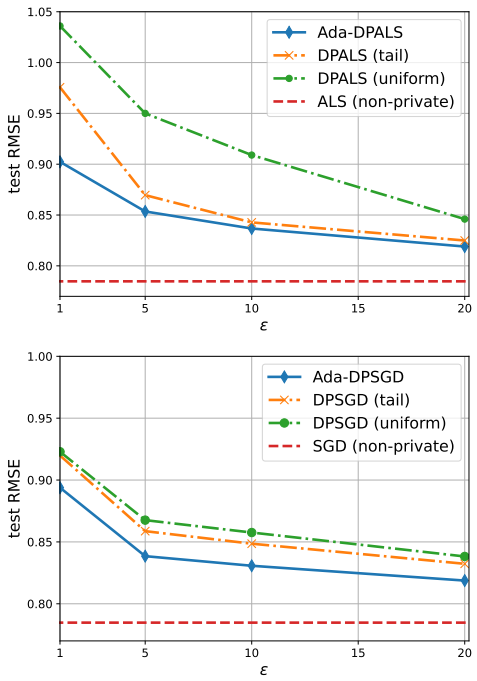


Figure 1. Privacy-utility trade-off on ML10M, using uniform sampling, tail-biased sampling, and adaptive weights (our method), applied to DPALS (top) and DPSGD (bottom). The utility is measured using RMSE (lower is better).

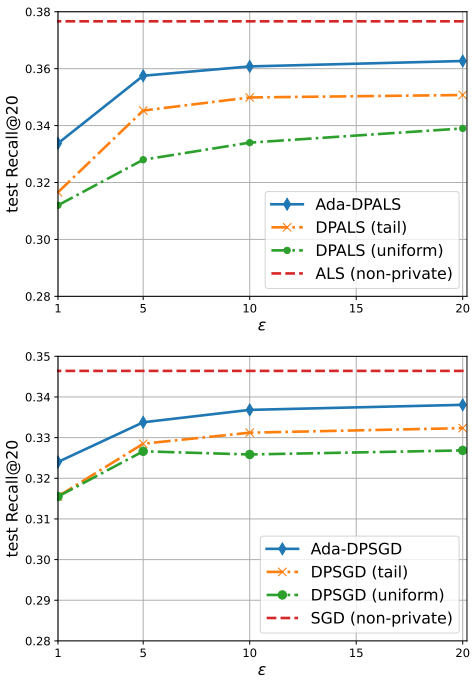


Figure 2. Privacy-utility trade-off on ML20M, using uniform sampling, tail-biased sampling, and adaptive weights (our method), applied to DPALS (top) and DPSGD (bottom). The utility is measured using Recall@20 (higher is better).

user, keep the k first movies). The latter was a heuristic specifically designed by (Chien et al., 2021) to address the movie distribution skew, and is the current SOTA.

We also experiment with DPSGD on the same benchmarks, and compare uniform sampling, tail-biased sampling, and adaptive weights.

When computing adaptive weights for our methods, we first compute private estimates $\hat{n}_i$ of the counts (which we include in the privacy accounting), then define task weights following eq. (11)-(12), but allowing a wider range of exponents. More precisely, we replace eq. (11) with

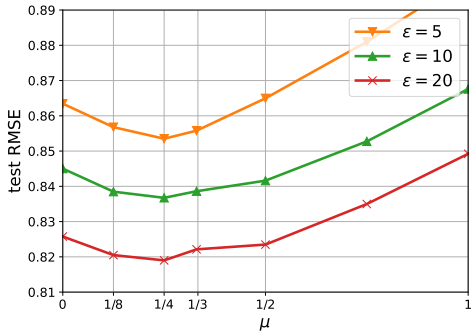
$$\omega_i^* = \frac{\hat{n}_i^{-\mu}}{\sqrt{\sum_{i'=1}^m \hat{n}_{i'}^{-2\mu+1}/n\bar{\beta}}}, \tag{16}$$

where μ is a hyper-parameter. In the analysis, the optimal choice of weights corresponds to $\mu = 1/4$ in the convex case (eq. (11) with $\gamma = 0$), and $\mu = 1/2$ in strongly convex case (eq.(11) with $\gamma = 1$). We experiment with different values of $\mu \in [0, 1]$ to evaluate the effect of shifting the weight distribution towards the tail.

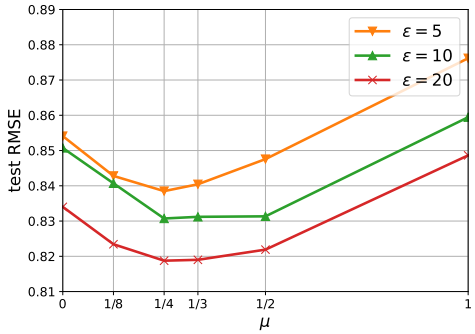
Effect of adaptive weights on privacy-utility trade-off We first evaluate the privacy/utility trade-off. The results are reported in Figure 1 (for ML10M) and Figure 2 (for ML20M), where we also include non-private baselines for reference. Our adaptive weights methods achieve the state of the art on both benchmarks. We see improvements when applying adaptive weights both to DPSGD and DPALS. The extent of improvement varies depending on the benchmark. In particular, the improvement is remarkable on the ML20M benchmark across all values of ϵ; using adaptive weights significantly narrows the gap between the previous SOTA and the non-private baseline.

To illustrate the effect of the exponent μ, we report, in Figure 3, the performance on ML10M for different values of μ. The best performance is typically achieved when μ is between $\frac{1}{4}$ and $\frac{1}{2}$, a range consistent with the analysis. For larger values of μ (for example $\mu = 1$), there is a clear degradation of quality, likely due to assigning too little weight to head tasks. The trend is similar on ML20M, see Figure 9 in the appendix.

Impact on head and tail tasks To better understand the extent of quality impact on head/tail movies, we report the same metrics, sliced by movie counts. We sort the movies i by increasing counts n_i, then group them into equally sized

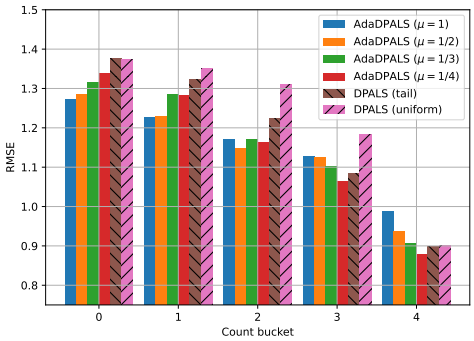


(a) DPALS on ML10M

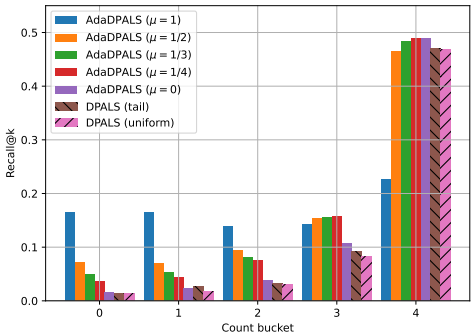


(b) DPSGD on ML10M

Figure 3. Effect of the weight exponent μ (see eq. (16)) on the ML10M benchmark, using DPALS (top) and DPSGD (bottom).



(a) RMSE on ML10M ($\epsilon = 1$).



(b) Recall on ML20M ($\epsilon = 1$).

Figure 4. RMSE and Recall metrics for $\epsilon = 1$, sliced by movie frequency. Each bucket contains an equal number of movies. Buckets are ordered by increasing movie counts.

buckets, and compute average metrics[3] on each bucket. The results are reported in Figure 4 for $\epsilon = 1$, and similar plots are provided in Appendix D for other values of ϵ.

The results show a clear trade-off between head and tail tasks. On buckets 0 and 1 (tail), it is generally the case that the larger μ is, the better the quality, while on bucket 4, the opposite trend can be observed. This is consistent with the intuition that as μ increases, more budget is assigned to the tail, which tends to shift the trade-off in favor of the tail.

Even compared to the previous SOTA (tail-sampling), the improvements on all but the top bucket are significant. Although the tail sampling heuristic was designed to improve tail quality, the experiments indicate that adaptive weights are much more effective. Furthermore, the parameter μ allows more control over this trade-off.

Finally, to give a concrete illustration of these improvements, we inspect the quality of recommendations on a few sample queries, reported in Appendix D. We find that there is a visible improvement on tail recommendations in models trained using our method.

[3]Since recall is naturally lower on tail tasks, we report Recall@k with larger k for tail buckets: we use $k = 20$ for the first bucket, 40 for the second, 60 for the third, and so on. This allows for more readable plots.

6. Conclusion

To address the practical problem of long-tailed data distributions, we formally analyze the question of budget allocation among tasks under user-level DP. Our method is based on computing weights that adapt to the task distribution. We quantify utility improvements under optimal weights, in a range of settings. Importantly, the method achieves significant improvements on benchmarks, and allows finer control over quality trade-offs between head and tail tasks.

To compute optimal weights, our analysis relied on certain distributional assumptions on the task-user graph Ω, and although this allowed us to model the task distribution skew, relaxing these assumptions is a promising direction. In particular, it may be possible to directly compute a privacy-preserving solution of problem (9) in its general form, using techniques similar to the constraint-private LPs studied by (Hsu et al., 2014).

Acknowledgements

We are grateful to Harsh Mehta, Steffen Rendle, John Anderson, and the anonymous reviewers for insightful discussions.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

Abowd, J. M. The u.s. census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pp. 2867, New York, NY, USA, 2018. Association for Computing Machinery.

Amin, K., Kulesza, A., Munoz, A., and Vassilvtiskii, S. Bounding user contributions: A bias-variance trade-off in differential privacy. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 263–271. PMLR, 09–15 Jun 2019.

Amin, K., Gillenwater, J., Joseph, M., Kulesza, A., and Vassilvitskii, S. Plume: Differential privacy at scale. *CoRR*, abs/2201.11603, 2022.

Bagdasaryan, E., Poursaeed, O., and Shmatikov, V. Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Bassily, R., Smith, A., and Thakurta, A. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 464–473, 2014.

Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., and Lamere, P. The million song dataset. In *In Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

Boucheron, S., Lugosi, G., and Massart, P. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

Brodersen, K. H., Ong, C. S., Stephan, K. E., and Buhmann, J. M. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pp. 3121–3124, 2010.

Calandrino, J. A., Kilzer, A., Narayanan, A., Felten, E. W., and Shmatikov, V. "you might also like:" privacy risks of collaborative filtering. In *2011 IEEE symposium on security and privacy*, pp. 231–246. IEEE, 2011.

Chien, S., Jain, P., Krichene, W., Rendle, S., Song, S., Thakurta, A., and Zhang, L. Private alternating least squares: Practical private matrix completion with tighter rates. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Cummings, R., Feldman, V., McMillan, A., and Talwar, K. Mean estimation with user-level privacy under data heterogeneity. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pp. 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

Dwork, C., McSherry, F., and Talwar, K. The price of privacy and the limits of lp decoding. In *Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, pp. 85–94, 2007.

Dwork, C., Rothblum, G. N., and Vadhan, S. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 51–60, 2010.

Epasto, A., Mahdian, M., Mao, J., Mirrokni, V., and Ren, L. Smoothly bounding user contributions in differential privacy. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13999–14010. Curran Associates, Inc., 2020.

Feldman, V. and Zrnic, T. Individual privacy accounting via a rényi filter. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021.

Foulds, J., Geumlek, J., Welling, M., and Chaudhuri, K. On the theory and practice of privacy-preserving bayesian data analysis. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, pp. 192–201, Arlington, Virginia, USA, 2016. AUAI Press.

Hardt, M. and Rothblum, G. N. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 61–70, 2010.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *Acm Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

Hsu, J., Roth, A., Roughgarden, T., and Ullman, J. Privately solving linear programs. In *Automata, Languages, and Programming*, pp. 612–624, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

Hu, S., Wu, Z. S., and Smith, V. Private multi-task learning: Formulation and applications to federated learning. *CoRR*, abs/2108.12978, 2021.

Huang, C., Li, Y., Loy, C. C., and Tang, X. Learning deep representation for imbalanced classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5375–5384, 2016.

Jain, P., Netrapalli, P., and Sanghavi, S. Low-rank matrix completion using alternating minimization. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pp. 665–674, New York, NY, USA, 2013. Association for Computing Machinery.

Jain, P., Thakkar, O. D., and Thakurta, A. Differentially private matrix completion revisited. In *International Conference on Machine Learning*, pp. 2215–2224. PMLR, 2018.

Jain, P., Rush, J., Smith, A., Song, S., and Guha Thakurta, A. Differentially private model personalization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29723–29735. Curran Associates, Inc., 2021.

Kasiviswanathan, S. P., Nissim, K., Raskhodnikova, S., and Smith, A. Analyzing graphs with node differential privacy. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pp. 457–476, Berlin, Heidelberg, 2013. Springer-Verlag.

Kearns, M., Pai, M., Roth, A., and Ullman, J. Mechanism design in large games: Incentives and privacy. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pp. 403–410, 2014.

Korolova, A. Privacy violations using microtargeted ads: A case study. In *2010 IEEE International Conference on Data Mining Workshops*, pp. 474–482. IEEE, 2010.

Kubat, M. and Matwin, S. Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*, pp. 179–186, 1997.

Lee, J., Kim, S., Lebanon, G., and Singer, Y. Local low-rank matrix approximation. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. II–82–II–90, 2013.

Levy, D., Sun, Z., Amin, K., Kale, S., Kulesza, A., Mohri, M., and Suresh, A. T. Learning with user-level privacy. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12466–12479, 2021.

Li, J., Khodak, M., Caldas, S., and Talwalkar, A. Differentially private meta-learning. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. Variational autoencoders for collaborative filtering. WWW '18, pp. 689–698, 2018.

Liu, S. and Zheng, Y. *Long-Tail Session-Based Recommendation*, pp. 509–514. Association for Computing Machinery, New York, NY, USA, 2020.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*, 2018.

Mironov, I. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 263–275. IEEE, 2017.

Proserpio, D., Goldberg, S., and McSherry, F. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proc. VLDB Endow.*, 7(8):637–648, apr 2014.

Rogers, R., Subramaniam, S., Peng, S., Durfee, D., Lee, S., Kancha, S. K., Sahay, S., and Ahammad, P. Linkedin's audience engagements api: A privacy preserving data analytics system at scale. *Journal of Privacy and Confidentiality*, 11(3), Dec. 2021.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017.

Ullman, J. Private multiplicative weights beyond linear queries. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '15, pp. 303–312, New York, NY, USA, 2015. Association for Computing Machinery.

Wang, Y. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. In Globerson, A. and Silva, R. (eds.), *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 93–103. AUAI Press, 2018.

Wang, Y.-X., Ramanan, D., and Hebert, M. Learning to model the tail. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Wilson, R. J., Zhang, C. Y., Lam, W., Desfontaines, D., Simmons-Marengo, D., and Gipson, B. Differentially private sql with bounded user contribution. In *Privacy Enhancing Technologies Symposium (PETS)*, 2020.

Xu, D., Du, W., and Wu, X. Removing disparate impact on model accuracy in differentially private stochastic gradient descent. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 1924–1932, New York, NY, USA, 2021. Association for Computing Machinery.

Yin, H., Cui, B., Li, J., Yao, J., and Chen, C. Challenging the long tail recommendation. *Proc. VLDB Endow.*, 5(9): 896–907, may 2012.

Appendix

The proofs of the main results are provided in Appendix A. Appendix B discusses the case when task counts are not public. We report additional experiments on synthetic data (Appendix C) and real data (Appendix D).

A. Proofs

A.1. Theorem 3.3 (Privacy Guarantee of Algorithm 1)

Proof. The result is an application of the Gaussian mechanism. The procedure computes, for $i \in [m]$, the estimates $\hat{A}_i$ and $\hat{b}_i$ (Lines 5-6), given as follows

$$\hat{A}_i = \bar{A}_i + \Gamma_x^2 \Xi_i, \quad \bar{A}_i = \sum_{j \in \Omega_i} w_{ij}(x_{ij}x_{ij}^\top + \lambda I)$$
$$\hat{b}_i = \bar{b}_i + \Gamma_x \Gamma_y \xi_i, \quad \bar{b}_i = \sum_{j \in \Omega_i} w_{ij} y_{ij} x_{ij}$$

where $\Xi_i \sim \mathcal{N}^{d\times d}$, $\xi_i \sim \mathcal{N}^d$. Let $\bar{A}$ be the matrix obtained by stacking $(\bar{A}_i)_{i\in[n]}$. If $\bar{A}'$ is the same matrix obtained without user j's data, then

$$\|\bar{A} - \bar{A}'\|_F^2 = \sum_{i \in \Omega^j} \|w_{ij}x_{ij}x_{ij}^\top\|_F^2 \leq \sum_{i \in \Omega^j} w_{ij}^2 \Gamma_x^4 \leq \beta \Gamma_x^4$$

where we use the fact that for all i, $\|x_{ij}\| \leq \Gamma_x$ and $\sum_{j\in\Omega_i} w_{ij}^2 \leq \beta$. Since we add Gaussian noise with variance Γ_x^4, releasing $\hat{A}$ is $(\alpha, \alpha\frac{\beta}{2})$-RDP (Mironov, 2017).

Similarly, if $\bar{b}$ is obtained by stacking $(\bar{b}_i)_{i\in[m]}$, and $\bar{b}'$ is the same vector without user j's data, then $\|\bar{b} - \bar{b}'\|^2 \leq \sum_{i\in\Omega^j} \|w_{ij}y_{ij}x_{ij}\|^2 \leq \sum_{j\in\Omega_i} w_{ij}^2 \Gamma_y^2 \Gamma_x^2 \leq \beta \Gamma_y^2 \Gamma_x^2$, and releasing $\hat{b}$ is $(\alpha, \alpha\frac{\beta}{2})$-RDP. By simple RDP composition, the process is $(\alpha, \alpha\beta)$-RDP. □

A.2. Theorem 3.8 (Utility Guarantee of Algorithm 1)

Proof. Since by assumption, the weights satisfy $\sum_{i\in\Omega_j} \omega_i^2 \leq \beta$, the RDP guarantee is an immediate consequence of Theorem 3.3.

To prove the utility bound, recall that the total loss is a sum over tasks

$$L(\theta) = \sum_{i=1}^{n} \|A_i\theta_i - b_i\|_F^2 + n_i\lambda\|\theta_i\|^2. \tag{17}$$

We will bound the excess risk of each term, using the following result. For a proof, see, e.g. (Wang, 2018, Appendix B.1).

Lemma A.1. *Suppose Assumption 3.6 holds. Consider the linear regression problem $L(\theta_i) = \|A_i\theta_i - b_i\|_F^2$, let θ_i^* be its solution, and $\hat{\theta}_i$ be the SSP estimate obtained by replacing A_i and b_i with their noisy estimates $\hat{A}_i = A_i + \sigma\Gamma_x^2\Xi, \hat{b}_i = b_i + \sigma\Gamma_x^2\Gamma_*\xi$ where $\Xi \sim \mathcal{N}^{d\times d}, \xi \sim \mathcal{N}^d$. Then*

$$L_i(\hat{\theta}_i) - L_i(\theta_i^*) = \mathcal{O}\left(\frac{d^2\Gamma_x^2\Gamma_*^2}{\alpha n_i}\sigma^2\right),$$

where $\alpha = \frac{\lambda_{\min}(A_i^\top A_i)d}{n_i\Gamma_x^2} = \frac{\lambda d}{\Gamma_x^2}$.

Fix a task i. Since by assumption all weights w_{ij} are equal to ω_i, Lines 5-6 become $\hat{A}_i = \omega_i A_i + \Gamma_x^2\Xi_i$ and $\hat{b}_i = \omega_i b_i + \Gamma_x^2\Gamma_*\xi_i$, and $\hat{\theta}_i$ is the solution to $\hat{A}_i\theta_i = \hat{b}_i$. This corresponds to the SSP algorithm, applied to the loss L_i, with noise variances $\sigma^2 = \frac{1}{\omega_i^2}$. By Lemma A.1, we have

$$L_i(\hat{\theta}_i) - L_i(\theta_i^*) = \mathcal{O}\left(\frac{d^2\Gamma_x^2\Gamma_*^2}{\alpha}\frac{1}{n_i\omega_i^2}\right) \tag{18}$$

We conclude by summing (18) over $i \in [m]$. □

A.3. Theorem 3.11 (Privacy Guarantee of Algorithm 2)

Proof. The result is an application of the Gaussian mechanism. At each step t of Algorithm 2, the procedure computes, for $i \in [m]$, a noisy estimate of the gradient $g_i^{(t)} = \sum_{j \in \Omega_i} g_i^{(t)}$ (Line 5). Let $g^{(t)} \in \mathbb{R}^{md}$ be the vector obtained by stacking $g_i^{(t)}$ for all i. And let $g'^{(t)}$ be the same vector obtained without user j's data. Then

$$\begin{aligned}\|g'^{(t)} - g^{(t)}\|_2^2 &= \sum_{i \in \Omega^j} \|w_{ij} \nabla \ell(\theta_i; x_{ij}, y_{ij})\|_2^2 \\ &\leq \Gamma^2 \sum_{i \in \Omega^j} w_{ij^2} \\ &\leq \beta \Gamma^2.\end{aligned}$$

where we use the assumption that $\sum_{i \in \Omega^j} w_{ij^2} \leq \beta$. Since we add Gaussian noise with variance $\Gamma^2 T/2$, the procedure is $(\alpha, \alpha\beta/T)$-RDP. Finally, by composition over T steps, the algorithm is (α, β)-RDP. □

A.4. Theorem 3.12 (Utility Guarantee of Algorithm 2)

We will make use of the following standard lemmas (for example Lemma 2.5 and 2.6 in (Bassily et al., 2014)). Let f be a convex function defined on domain $\mathcal{C}$, let $\theta^* = \operatorname{argmin}_{\theta \in \mathcal{C}} f(\theta)$. Consider the SGD algorithm with learning η_t.

$$\theta^{(t+1)} = \Pi_{\mathcal{C}}[\theta^{(t)} - \eta_t g^{(t)}].$$

Assume that there exists G such that for all t, $\mathbb{E}[g^{(t)}] = \nabla \ell(\theta^{(t)})$ and $\mathbb{E}[\|g^{(t)}\|^2] \leq G^2$,

Lemma A.2 (Lipschitz case)**.** *Let $\eta^{(t)} = \frac{\|\mathcal{C}\|}{G\sqrt{t}}$. Then for all $T \geq 1$,*

$$\mathbb{E}[f(\theta^{(t)})] - f(\theta^*) = \mathcal{O}\left(\frac{\|C\| G \log T}{\sqrt{T}}\right).$$

Lemma A.3 (Strongly convex case)**.** *Assume that f is λ strongly convex and let $\eta^{(t)} = \frac{1}{\lambda t}$. Then for all $T \geq 1$,*

$$\mathbb{E}[f(\theta^{(t)})] - f(\theta^*) = \mathcal{O}\left(\frac{G^2 \log T}{\lambda T}\right).$$

proof of Theorem 3.12. First, since $w_{ij} = \omega_i$, the gradient $g_i^{(t)}$ in Line 5 of the algorithm becomes

$$g_i^{(t)} = \omega_i \nabla L_i(\theta_i^{(t-1)}).$$

Let $\hat{g}_i^{(t)} = g_i^{(t)} + \Gamma\sqrt{T/2}\xi_i^{(t)}$. Then $\mathbb{E}[\hat{g}_i^{(t)}] = g_i^{(t)}$ and

$$\begin{aligned}\mathbb{E}[\|\hat{g}_i^{(t)}\|^2] &= \mathbb{E}[\|g_i^{(t)}\|^2] + \mathbb{E}[\|\Gamma\sqrt{T/2}\xi_i^{(t)}\|^2] \\ &\leq \Gamma^2[\omega_i^2 n_i^2 + Td/2].\end{aligned}$$

In the first line we use independence of $\xi_i^{(t)}$ and $g_i^{(t)}$. In the second line we use that the variance of a multivariate normal is d, and the fact that L_i has Lipschitz constant $n_i\Gamma$ (since it is the sum of n_i terms, each being Γ-Lipschitz).

Define

$$G_i^2 = \Gamma^2[\omega_i^2 n_i^2 + Td/2].$$

First, consider the Lipschitz bounded case. Applying Lemma A.2 to $f = \omega_i L_i$, $G = G_i$, and $\eta_i = \frac{\|C\|}{G_i\sqrt{t}}$, we have for all T

$$\omega_i \, \mathbb{E}[L_i(\theta_i) - L_i(\theta_i^*)] = \mathcal{O}\left(\frac{\|C\| G_i \log T}{\sqrt{T}}\right).$$

Multiplying by $\frac{1}{\omega_i}$ and summing over tasks $i \in \{1, \dots, m\}$, we have

$$\mathbb{E}[L(\theta)] - L(\theta^*) = \mathcal{O}\left(\frac{\|\mathcal{C}\|\Gamma \log T}{\sqrt{T}} \sum_{i=1}^{m} \sqrt{n_i^2 + \frac{Td}{2\omega_i^2}}\right)$$
$$= \mathcal{O}\left(\|\mathcal{C}\|\Gamma \log T \sqrt{m} \sqrt{\frac{\sum_{i=1}^m n_i^2}{T} + \sum_{i=1}^{m} \frac{d}{2\omega_i^2}}\right)$$

where we used $\sum_{i=1}^m \sqrt{a_i} \leq \sqrt{m \sum_{i=1}^m a_i}$ (by concavity). Finally, setting T to equate the terms under the square root, we get $T = \frac{2}{d} \frac{\sum_{i=1}^m n_i^2}{\sum_{i=1}^m 1/\omega_i^2}$, and with this choice of T,

$$\mathbb{E}[L(\theta)] - L(\theta^*) = \tilde{\mathcal{O}}\left(\|\mathcal{C}\|\Gamma\sqrt{md} \sqrt{\sum_{i=1}^{m} \frac{1}{\omega_i^2}}\right),$$

as desired.

We now turn to the strongly convex case. Applying Lemma A.3 to $f = \omega_i L_i$, $G = G_i$ (same as above), strong convexity constant $\omega_i n_i \lambda$, and $\eta_i = \frac{1}{\omega_i n_i \lambda t}$, we have for all T,

$$\omega_i \, \mathbb{E}[L_i(\theta_i) - L_i(\theta_i^*)] = \mathcal{O}\left(\frac{G_i^2 \log T}{n_i \omega_i \lambda T}\right).$$

Multiplying by $\frac{1}{\omega_i}$ and summing over tasks, we get

$$\mathbb{E}[L(\theta)] - L(\theta^*) = \mathcal{O}\left(\frac{\Gamma^2 \log T}{\lambda}\left(\frac{1}{T}\sum_{i=1}^{m} n_i + \frac{d}{2}\sum_{i=1}^{m} \frac{1}{\omega_i^2 n_i}\right)\right).$$

Setting T to equate the last two sums, we get $T = \frac{2}{d} \frac{\sum_{i=1}^m n_i}{\sum_{i=1}^m 1/\omega_i^2 n_i}$ and with this choice of T,

$$\mathbb{E}[L(\theta)] - L(\theta^*) = \tilde{\mathcal{O}}\left(\frac{\Gamma^2 d}{\lambda} \sum_{i=1}^{m} \frac{1}{\omega_i^2 n_i}\right),$$

as desired. □

A.5. Theorem 4.3 (Privacy-Utility Trade-off Under Adaptive Weights)

To prove the theorem, we will use the following concentration result:

Lemma A.4 (Concentration bound on the privacy budget)**.** *Suppose that Assumption 4.1 holds and let $B > 0$ be given. Let $n_i = |\Omega_i|$ and $\omega_i = n_i^{-(1+\gamma)/4} / \sqrt{\sum_{i'} n_{i'}^{(1-\gamma)/2} / nB}$. Then there exists $c > 0$ (that does not depend on B) such that, with high probability, for all i, $\sum_{i \in \Omega^j} \omega_i^2 \leq Bc \log n$.*

Proof. Fix a user j. We seek to bound $\sum_{i \in \Omega^j} \omega_i^2$. Let $p_{ij} = P((i,j) \in \Omega)$. Recall that by Assumption 4.1, $p_{ij} = n_i/n$.

For each i, define X_i as the random variable which takes value $\omega_i^2 = nBn_i^{-(1+\gamma)/2} / \sum_{i'} n_{i'}^{(1-\gamma)/2}$ with probability n_i/n and 0 otherwise. Then bounding $\sum_{i \in \Omega^j} \omega_i^2$ is equivalent to bounding $\sum_{i=1}^m X_i$.

We have

$$\mathrm{E}[X_i] = Bn_i^{(1-\gamma)/2} / \sum_{i'} n_{i'}^{(1-\gamma)/2},$$
$$\mathrm{Var}[X_i] \leq \mathrm{E}[X_i^2] = (n_i/n) \cdot \left(nBn_i^{-(1+\gamma)/2} / \sum_{i'} n_{i'}^{(1-\gamma)/2}\right)^2$$
$$= nB^2 n_i^{-\gamma} / \left(\sum_{i'} n_{i'}^{(1-\gamma)/2}\right)^2 \leq B^2/n,$$

where the last inequality is by the fact that $\forall i,\ n_i \geq 1$ (see Assumption 4.1). Hence we have that

$$\sum_{i=1}^{m} \mathrm{E}[X_i] = B\,, \quad \sum_{i=1}^{m} \mathrm{Var}[X_i] \leq (m/n)B^2 \leq B^2,$$

where we use the assumption that $n \geq m$. Furthermore, we have $\omega_i^2 \leq B$ since $\forall j,\ n_i \geq 1$. By the standard Bernstein's inequality ((Boucheron et al., 2013) §2.8), there exists $c > 0$ (that does not depend on B or n_i) such that:

$$\Pr\left(\sum_{i=1}^{m} X_i \geq Bc\log n\right) \leq 1/n^3\,.$$

The high probability bound follows by taking the union over all the i's. □

Proof of Theorem 4.3. Let ω^*, w^* be the weights given in eq. (11)-(12), respectively (ω^* are the optimal weights, and w^* are their clipped version). We apply Lemma A.4 with $B = \bar{\beta} = \beta/c\log n$ and hence obtain that with high probability, for all i, $\sum_{j\in\Omega_i} {\omega_i^*}^2 \leq \beta$. In other words, w.h.p. clipping does not occur and $w_{ij}^* = \omega_i$ for all $(i,j) \in \Omega$. Conditioned on this high probability event, we can apply the general utility bounds in Theorems 3.8 and 3.12, with $\omega_i = \omega_i^*$. This yields the desired bounds. □

A.6. Related Work on Class-Imbalanced Learning

The question of distribution skew has been studied in the non-private setting, and several techniques have been proposed to address class imbalance, either using re-sampling or re-weighting (Brodersen et al., 2010; Huang et al., 2016; Wang et al., 2017; Cui et al., 2019). The main question in this line of work is that if one measures performance using a *balanced* metric (i.e. one that assigns equal importance to the different classes, rather than following the skewed data distribution), how should one modify the training process to improve quality? This question is orthogonal to our approach. Our study shows that regardless of the loss being optimized (i.e., even if one cares about the empirical, not the balanced loss), learning can be improved through a more careful allocation of the privacy budget. It should be possible to obtain a similar improvement in the balanced case, in which the objective function assigns equal weights to each task (for example by defining $L_i(\theta_i)$ in eq. (2) to be a mean instead of a sum).

B. Utility Analysis Under Approximate Counts

In Section 4, the optimal choice of weights ω^* assumes knowledge of the counts n_i. If the counts n_i are not public, we can use DP estimates $\hat{n}_i$, and use them to solve the problem. Since differentially private counting has been studied extensively, we only state the utility bound in terms of the accuracy of the noisy counts – the final privacy bound can be done through standard composition. For any constant $s > 0$, we call a counting procedure $\mathcal{M}$ s-accurate, if w.h.p., the counts $\hat{n}_i$ produced by $\mathcal{M}$ satisfy that $|\hat{n}_i - n_i| \leq s$ for all i.

We will give an analysis of Algorithm 1 with approximate counts. The general idea is to apply the algorithm with weights $\hat{\omega}$ of the same form as the optimal weights ω^*, but where the exact counts are replaced with estimated counts. Suppose we are given s-accurate count estimates $\hat{n}_i$. Define

$$\hat{\omega}_i = \frac{1}{(\hat{n}_i + s)^{1/2}\sqrt{m/n\bar{\beta}}}, \tag{19}$$

This corresponds to eq. (11), but with n_i replaced by $\hat{n}_i + s$ (it will become clear below why we need to slightly over-estimate the counts).

As in the exact case, we need to clip the weights, so that the privacy guarantee always holds. Define the clipped version $\hat{w}$ as

$$\hat{w}_{ij} = \hat{\omega}_i \min\left(1, \frac{\beta^{1/2}}{(\sum_{i'\in\Omega^j} \hat{\omega}_{i'}^2)^{1/2}}\right) \quad \forall (i,j) \in \Omega. \tag{20}$$

Remark B.1. The input to the algorithm are the unclipped weights $\hat{\omega}$, which are computed differentially privately. The clipped weights are not released as part of the procedure. They are simply used as a scaling factor in the computation of $\hat{A}_i, \hat{b}_i$ (Lines 5-6).

Theorem B.2 (Privacy-utility trade-off of Algorithm 1 with estimated counts)**.** *Suppose Assumptions 3.6 and 4.1 hold. Let $\hat{n}_i \in \mathbb{R}^m$ be count estimates from an s-accurate procedure for some $s > 0$. Define $\hat{\omega}, \hat{w}$ as in eq. (19)-(20). Let $\hat{\theta}$ be the output of Algorithm 1 run with weights $\hat{w}$. Then Algorithm 1 is $(\alpha, \alpha\beta)$-RDP for all $\alpha > 1$, and w.h.p.,*

$$L(\hat{\theta}) - L(\theta^*) = \tilde{\mathcal{O}}\left(\frac{\Gamma_x^4 \Gamma_*^2 dm}{n\lambda\beta} \sum_{i=1}^m \frac{\hat{n}_i + s}{n_i}\right). \tag{21}$$

Proof of Theorem B.2. We first show that w.h.p., the approximate weights $\hat{\omega}_i$ *under-estimate* the exact weights ω_i^*. This will allow us to argue that w.h.p., clipping will not occur.

First, define the function $g : \mathbb{R}^d \to \mathbb{R}^d$,

$$g_i(\eta) = \frac{1}{\eta_i^{1/2}\sqrt{m/n\bar{\beta}}}.$$

Observe that whenever $\eta \leq \eta'$ (coordinate-wise), $g(\eta) \geq g(\eta')$ (coordinate-wise).

Let η^* be the vector of true counts ($\eta_i^* = n_i$), and $\hat{\eta}$ be the vector or approximate counts $\hat{\eta}_i = n_i$. We have that $\omega^* = g(\eta^*)$ and $\hat{\omega} = g(\hat{\eta} + s)$. But since by assumption, the counts estimates are s-accurate, then w.h.p., $|\hat{n}_i - n_i^*| \leq s$, thus $\eta^* \leq \hat{\eta} + s$ and

$$\hat{\omega} = g(\hat{\eta} + s) \leq \omega(\eta^*) = \omega^*,$$

that is, the estimated weights $\hat{\omega}$ under-estimate the optimal weights ω^* w.h.p. (this is precisely why, in the definition of $\hat{\omega}$, we used the adjusted counts $\hat{\eta} + s$ instead of $\hat{\eta}$).

Next, $\hat{\omega} \leq \omega^*$ implies that

$$\sum_{i \in \Omega^j} \hat{\omega}_i^2 \leq \sum_{i \in \Omega^j} {\omega_i^*}^2. \tag{22}$$

We also know, from Lemma A.4 (applied with $B = \bar{\beta}$), that w.h.p., $\sum_{i \in \Omega^j} {\omega_i^*}^2 \leq c\bar{\beta}\log n = \beta$ for all j. Combining this with (22), we have that w.h.p., $\hat{\omega}$ satisfies the RDP budget constraint, therefore $w_{ij} = \hat{\omega}_i^2$ (clipping does not occur since the budget constraint is satisfied).

Applying Theorem 3.8 with weights $\hat{\omega}$ yields the desired result. □

We compare the utility bounds under adaptive weights with exact counts (13), adaptive weights with approximate counts (21), and uniform weights. The bounds are, respectively, $\tilde{\mathcal{O}}\left(\frac{\Gamma_x^4\Gamma_*^2 dm^2}{n\lambda\beta}\right)$, $\tilde{\mathcal{O}}\left(\frac{\Gamma_x^4\Gamma_*^2 dm}{n\lambda\beta}\sum_{i=1}^m \frac{\hat{n}_i+s}{n_i}\right)$, and $\tilde{\mathcal{O}}\left(\frac{\Gamma_x^4\Gamma_*^2 d}{n\lambda\beta}\sum_{i=1}^m n_i \sum_{i=1}^m \frac{1}{n_i}\right)$.

The cost of using approximation counts (compared to exact counts) is a relative increase by the factor

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^m \frac{\hat{n}_i + s}{n_i} &\leq \frac{1}{m}\sum_{i=1}^m \frac{n_i + 2s}{n_i} \\ &= 1 + \frac{2}{m}\sum_{i=1}^m \frac{s}{n_i}.\end{aligned}$$

where we used that w.h.p., $\hat{n}_i \leq n_i + s$. The last term is the *average relative error* of count estimates. If the estimates are accurate on average, we don't expect to see a large utility loss due to using approximate counts.

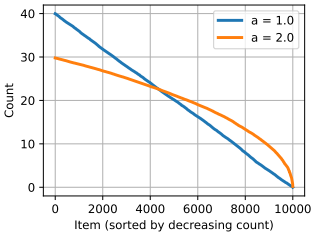
C. Synthetic Experiments

We conduct experiments following the setup in Section 3. Specifically, we consider $m = 100$ linear regression tasks of dimension $d = 5$ and data from $n = 10,000$ users. We assume that each task has an optimal solution θ_i, and each user j has a vector u_j, such that $x_{ij} = u_j$, and $y_{ij} = \langle u_i, \theta_j \rangle + \mathcal{N}(0, \sigma_F^2)$, with σ_F representing some inherent data noise. We generate u_j and θ_i from a Gaussian distribution $\mathcal{N}^d$ and projected to the unit ball, and set $\sigma_F = 10^{-3}$.

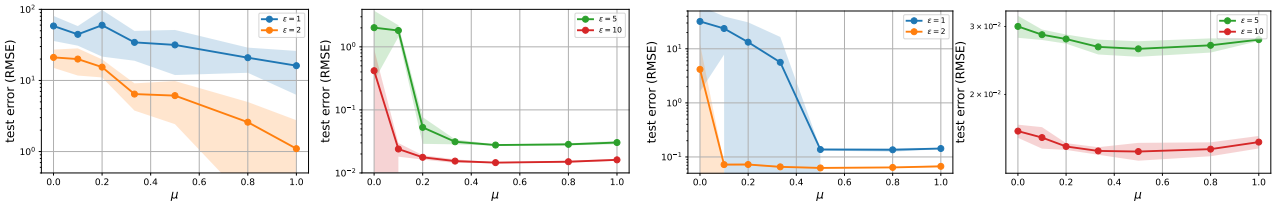
To model a skewed distribution of tasks, we first sample, for each task i, a value $q_i \in [0, 1]$ following a power law distribution with parameter $a = 1, 2$ (the pdf of the distribution is $f(x) \propto ax^{a-1}$), representing two level of skewness. We

then normalize q_is such that they sum up to 20. Then we construct Ω by sampling each (i, j) with probability q_i. This way, in expectation, each user contributes to 20 tasks, and $q_i n$ users contribute to task i. The distribution of $q_i n$ is plotted in Figure 5a. We partition the data into training and test sets following an 80-20 random split.

We run AdaDPALS and AdaDPSGD with weights set to $\omega_i \propto n_i^{-\mu}$ for varying μ, where n_i is the number of users contributing to task i. The weights are normalized per user. Setting $\mu = 0$ corresponds to the standard unweighted objective function. Since the purpose of the experiment is to validate the analysis and illustrate the algorithm in an stylized setting, we run the algorithm using the exact n_is instead of estimating them privately. In Figure 5b-5e, we plot the RMSE for different values of μ and ϵ, for both algorithms and skewness. We set $\delta = 10^{-5}$.

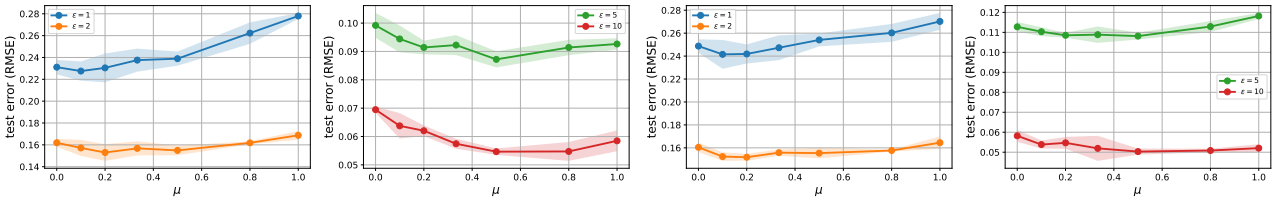


(a) n_i in the synthetic data.



(b) DPMultiRegression. Skewness $a = 1$.

(c) DPMultiRegression. Skewness $a = 2$.



(d) DPSGD. Skewness $a = 1$.

(e) DPSGD. Skewness $a = 2$.

Figure 5. RMSE vs. μ on synthetic data. $\delta = 10^{-5}$.

We observe that with uniform weights ($\mu = 0$), the quality of the estimate can be quite poor, especially at lower values of ϵ. The quality improves significantly as we increase μ. Theoretical analysis (Equation (11)) suggests $\mu = 1/2$ as the optimal choice, yet the empirically optimal μ can vary case by case. This may be due to the fact that the analysis makes no assumptions about the feature and label distribution, while in the experiment, the data is sampled from a linear model; we leave further analysis into designing better weighting strategies for *easier* data for future work.

D. Additional Experiments on MovieLens and Million Song Data

In this section, we report additional experiments on MovieLens and Million Song Data (Bertin-Mahieux et al., 2011) data sets. The code used to run the experiments is made available at the following URL: **https://github.com/google-research/google-research/tree/master/dp_alternating_minimization**.

We report additional experiments on the larger Million Song Data benchmark (abbreviated as MSD), see Figures 6-8. Due to the larger size of the data, we were only able to tune models of smaller size (embedding dimension of up to 32), but we expect the trends reported here to persist in larger dimension.

Table 1. Statistics of the MovieLens and Million Song data sets.

	ML10M	ML20M	MSD
n (number of users)	69,878	136,677	571,355
m (number of items)	10,677	20,108	41,140
$\|\Omega\|$ (number of observations)	10M	9.99M	33.63M

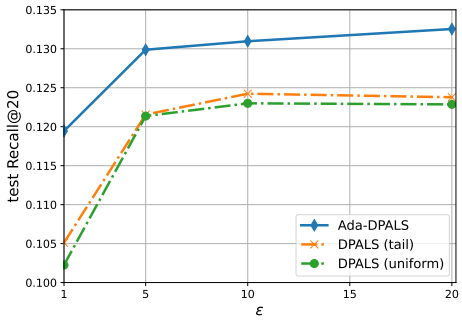


Figure 6. Privacy-utility trade-off on the Million Song Data benchmark.

D.1. Detailed Experimental Setting

We follow the same experimental setting as (Jain et al., 2018; Chien et al., 2021; Jain et al., 2021). When reporting (ϵ, δ) DP guarantees, we use values of $\epsilon \in [1, 20]$, and take $\delta = 10^{-5}$ for ML10M and $\delta = 1/n$ for ML20M and MSD. All hyper-parameter values are specified in the source code.

We solve a private matrix completion problem, where the training data is a partially observed rating matrix $(y_{ij})_{(i,j)\in\Omega}$ (where y_{ij} is the rating given by user j to item i), and the goal is to compute a low-rank approximation $Y \approx UV^\top$, by minimizing the following objective function:

$$L(U, V) = \sum_{(i,j)\in\Omega} (\langle u_i, v_j\rangle - y_{ij})^2 + \lambda|u_i|^2 + \lambda|v_j|^2.$$

The matrix $U \in \mathbb{R}^{n\times d}$ represents item embeddings, and the matrix $V \in \mathbb{R}^{m\times d}$ represents user embeddings.

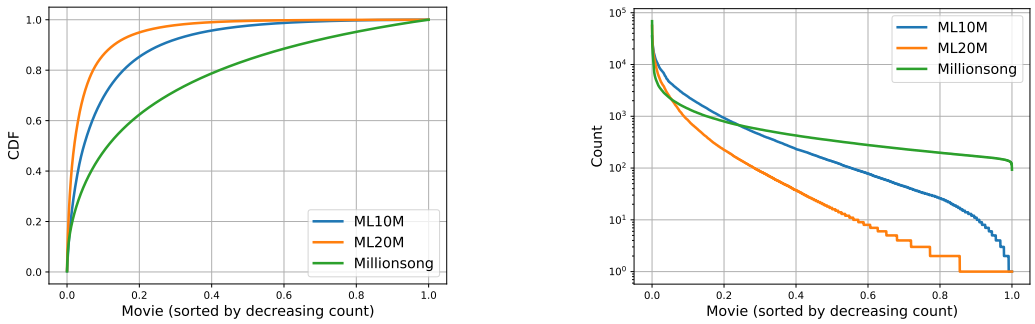
Algorithms We consider a family of alternating minimization algorithms studied by (Chien et al., 2021; Jain et al., 2021), in which we will use our algorithm as a sub-routine. This is summarized in Algorithm 3.

Algorithm 3 Alternating Minimization for Private Matrix Completion

1: **Inputs**: Training data $\{y_{ij}\}_{(i,j)\in\Omega}$, number of steps T, initial item matrix $\hat{U}^{(0)}$, pre-processing RDP budget β_0, training RDP budget β.
2: Pre-process the training data (with RDP budget β_0).
3: **for** $1 \le t \le T$ **do**
4: $\quad\hat{V}^{(t)} \leftarrow \mathrm{argmin}_V\, L(\hat{U}^{(t-1)}, V)$
5: $\quad$Compute a differentially private solution $\hat{U}^{(t)}$ of $\min_U L(U, \hat{V}^{(t)})$ (with RDP budget β)
6: **end for**
7: **Return** $\hat{U}^{(T)}$

Algorithm 3 describes a family of algorithms, that includes DPALS (Chien et al., 2021), and the AltMin algorithm of (Jain et al., 2021). It starts by pre-processing the training data (this includes centering the data, and computing private counts to be used for sampling or for computing adaptive weights). Then, it alternates between updating V and updating U. Updating V is done by computing an exact least squares solution, while updating U is done differentially privately.

Remark D.1. The algorithm only outputs the item embedding matrix $\hat{U}$. This is indeed sufficient for the recommendation task: given an item matrix $\hat{U}$ (learned differentially privately), to generate predictions for user j, one can compute the user's



(a) Cumulative distribution function

(b) Sorted task counts

Figure 7. Task distribution skew in MovieLens and Million Song Data.

embedding $v_j^* = \text{argmin}_{v \in \mathbb{R}^d} \sum_{i \in \Omega^j} (\langle \hat{u}_i, v \rangle - y_{ij})^2 + \lambda |v|^2$, then complete the j-th column by computing $\hat{U} v_j^*$. This minimization problem only depends on the published matrix $\hat{U}$ and user j's data, therefore can be done in isolation for each user (for example on the user's own device), without incurring additional privacy cost. This is sometimes referred to as the billboard model of differential privacy, see for example (Jain et al., 2021).

Notice that updating the item embeddings U (Line 5 in Algorithm 3) consists in solving the following problem differentially privately:

$$\min_U \sum_{i=1}^m \sum_{j \in \Omega_i} (\langle u_i, v_j \rangle - y_{ij})^2 + \lambda |u_i|^2.$$

This corresponds to our multi-task problem (1), where we identify each item to one task, with parameters $\theta_i = u_i$, features $x_{ij} = v_j$, and labels y_{ij}. Then we can either apply Algorithm 1 (SSP with adaptive weights), or Algorithm 2 (DPSGD with adaptive weights) to compute the item update. The baselines we compare to are summarized in Table 2.

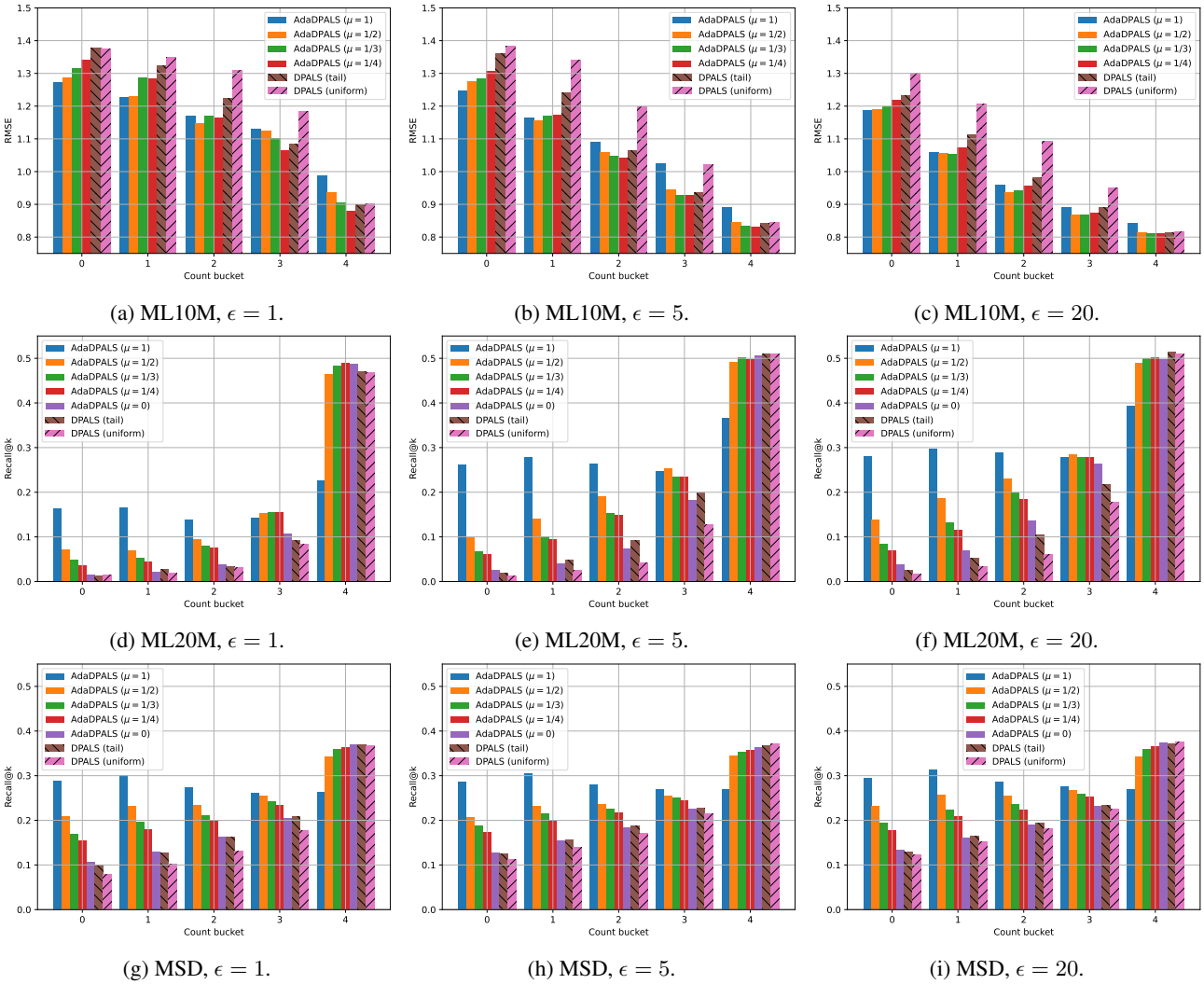
Table 2. Algorithms used in experiments.

Method	Sub-routine used to update U (Line 5 in Algorithm 3)	Budget allocation method
DPALS (uniform)	SSP	Uniform sampling
DPALS (tail)	SSP	Tail-biased sampling of (Chien et al., 2021)
AdaDPALS	Algorithm 1	Adaptive weights (eq. (11)-(12))
DPSGD (uniform)	DPSGD	Uniform sampling
DPSGD (tail)	DPSGD	Tail-biased sampling of (Chien et al., 2021)
AdaDPSGD	Algorithm 2	Adaptive weights (eq. (11)-(12))

We compare to two algorithms: the DPALS method (Chien et al., 2021) which is the current SOTA on these benchmarks, and the DPSGD algorithm (which we found to perform well in the full-batch regime, at the cost of higher run times). In each case, we compare three methods to perform budget allocation: using uniform sampling, the tail-biased sampling heuristic of (Chien et al., 2021), and our adaptive weights method. Note that tail sampling is *already adaptive to the task skew*: it estimates task counts and uses them to select the tasks to sample for each user.

Both for tail-sampling and adaptive weights, movie counts are estimated privately, and we account for the privacy cost of doing so. The proportion of RDP budget spent on estimating counts (out of the total RDP budget) is 20% for $\epsilon = 20$, 14% for $\epsilon = 5$ and 12% for $\epsilon = 1$. This was tuned on the DPALS baseline (with tail sampling).

Metrics The quality metrics that the benchmarks use are defined as follows: Let Ω^{test} be the set of test ratings. Then for a given factorization U, V, $\text{RMSE}(U, V) = \sqrt{\frac{\sum_{(i,j) \in \Omega^{\text{test}}} (\langle u_i, v_j \rangle - y_{ij})^2}{|\Omega^{\text{test}}|}}$. Recall is defined as follows. For a given user i, let ${\Omega^j}^{\text{test}}$ be the set of movies rated by the user j. If we denote by $\hat{\Omega}^j$ the set of top k predictions for user j, then the recall is defined as $\text{Recall@k} = \frac{1}{n} \sum_{j=1}^n \frac{|{\Omega^j}^{\text{test}} \cap \hat{\Omega}^j|}{\min(k, |{\Omega^j}^{\text{test}}|)}$.



(a) ML10M, $\epsilon = 1$. (b) ML10M, $\epsilon = 5$. (c) ML10M, $\epsilon = 20$.

(d) ML20M, $\epsilon = 1$. (e) ML20M, $\epsilon = 5$. (f) ML20M, $\epsilon = 20$.

(g) MSD, $\epsilon = 1$. (h) MSD, $\epsilon = 5$. (i) MSD, $\epsilon = 20$.

Figure 8. Metrics sliced by movie frequency (i.e. frequency), on the ML10M, ML20M, and MSD benchmark, using the DPALS method. Each bucket contains an equal number of movies. Buckets are ordered by increasing frequency.

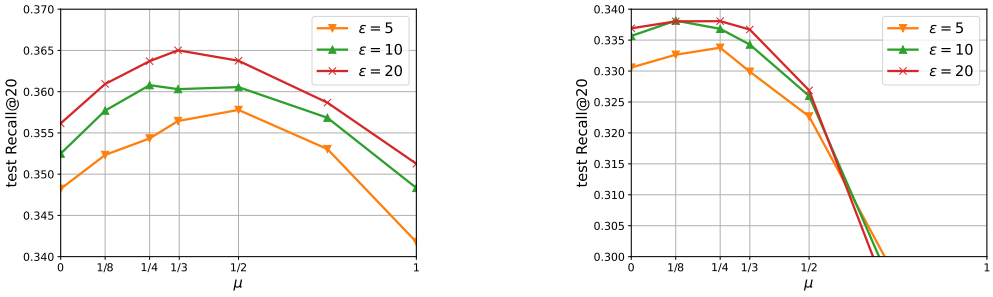


Figure 9. Comparison of the adaptive weights method with different values of μ on ML20M, when applied to DPALS (left) and DPSGD (right).

D.2. Quality Impact on Head/Tail Movies

In Figure 8, we report sliced RMSE (on ML10M) and Recall (on ML20M and MSD), for different values of ϵ.

The following trend can be observed on all benchmarks, across all values of ϵ. DPALS with tail sampling improves upon uniform sampling, especially on the tail buckets. Our method (AdaDPALS) further improves upon tail sampling. The improvement is quite significant for lower values of ϵ, and for tail buckets. For example, on ML10M with $\epsilon = 1$ (Figure 8a) we observe a large gap in RMSE, *across all values of* μ; the improvement is at least 21.6% on bucket 0, at least 23.7% on bucket 1, at least 22.8% on bucket 3, and at least 8.4% on bucket 4. The gap narrows as ϵ increases, which is consistent with the global privacy/utility trade-off plots in Figure 1.

The exponent μ controls the trade-off between head and tail tasks: recall that the weights are defined as $\omega_i \propto 1/\hat{n}_i^{\mu}$ where $\hat{n}_i$ are the count estimates. A larger value of μ induces larger weights (and hence better quality) on the tail. This is visible on both benchmarks and across all values of ϵ: on lower buckets 0 and 1, better performance is obtained for larger values of μ, while the trend is reversed for the top bucket.

When comparing performance on the overall objective, we find that the best performance is typically achieved when $\mu = 1/4$, see Figures 3 and 9. When applied to DPALS performance remains high for a range of $\mu \in [1/4, 1/2]$. When applied to DPSGD, performance seems more sensitive to μ, and the best performance is achieved for $\mu = 1/4$.

D.3. Qualitative Evaluation on ML20M

To give a qualitative evaluation of the improvements achieved by our method, we inspect a few example queries. Though anecdotal, these examples give a perhaps more concrete illustration of some of the quality impact that our method can have, especially on tail recommendations. We will compare the following models: the ALS non-private baseline (same model in Figure 1-b), and two private models with $\epsilon = 1$: the DPALS method with tail-biased sampling, and Ada-DPALS (with adaptive weights) with $\mu = 1/3$ (we found that values of $\mu \in [1/4, 1/2]$ are qualitatively similar).

We evaluate the models by displaying the nearest neighbor movies to a given query movie, where the similarity between movies is defined by the learned movie embedding matrix $\hat{V}$ (the similarity between two movies i_1 and i_2 is $\langle \hat{v}_{i_1}, \hat{v}_{i_2} \rangle$. We select a few examples in Table 3; for additional examples, the models can be trained and queried interactively using the provided code. We select examples with varying levels of frequency (shown in the last column), to illustrate how privacy may affect quality differently depending on item frequency.

The first query is *The Shawshank Redemption*, the most frequent movie in the data set. We see a large overlap of the top nearest neighbors according to all three models. In other words, privacy has little impact on this item. A similar observation can be made for other popular items.

The second query is *Pinocchio*, a Disney animated movie from 1940. The nearest neighbors according to the non-private baseline (ALS) are other Disney movies from neighboring decades. The DPALS results are noisy: some of the top neighbors are Disney movies, but the fifth and sixth neighbors seem unrelated (Action and Drama movies). The AdaDPALS model returns more relevant results, all of the neighbors being Disney movies.

The third example is *Harry Potter and the Half-Blood Prince*, an Adventure/Fantasy movie. The nearest neighbors from the ALS baseline are mostly other Harry Potter movies. The DPALS model misses the Harry Potter neighbors, and instead returns mostly action/adventure movies with varying degrees of relevance. The AdaDPALS model recovers several of Harry Potter neighbors.

The next example is *Nausicaä of the Valley of the Wind*, a Japanese animated movie from Studio Ghibli, released in 1984. The nearest neighbors according to the ALS model are similar movies from Studio Ghibli. The neighbors returned by DPALS are much more noisy: The first result (Spirited Away) is a Studio Ghibli movie and is the most relevant in the list. Other neighbors in the list are arguably unrelated to the query. AdaDPALS returns much more relevant results, all neighbors are Japanese animation movies, and five out of six are from Studio Ghibli.

The last example is *Interstellar*, a Sci-Fi movie released in 2014. The ALS neighbors are other popular movies released around the same time (2013-2014) with a bias towards Action/Sci-Fi. Both private models (DPALS and AdaDPALS) return mixed results. Some results are relevant (Action/Sci-Fi movies) but others are arguably much less relevant, for example the third DPALS neighbor is a Hitchcock movie from 1945. AdaDPALS neighbors appear slightly better overall, in particular it returns two movies from the same director.

As can be seen from these examples, AdaDPALS generally returns better quality results compared to DPALS. If we compare both models to the non-private baseline (ALS), the overlap between AdaDPALS and ALS is generally larger then the overlap between DPALS and ALS. To quantify this statement, we generate, for each movie, the top-20 nearest neighbors according

to each model, then compute the percentage overlap with ALS. The results are reported in Figure 10.

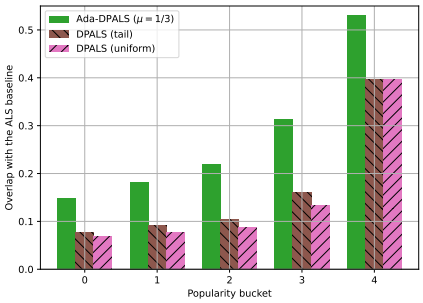


Figure 10. Percentage overlap between the top-20 nearest neighbors according to private models ($\epsilon = 1$) with the top-20 nearest neighbors according to the non-private baseline (ALS). Each bucket contains an equal number of movies (ordered by increasing frequency).

Model	Movie title	Genres	Frequency (in training set)
	Shawshank Redemption, The (1994)	Crime—Drama	47518
ALS	Usual Suspects, The (1995)	Crime—Mystery—Thriller	33834
	Silence of the Lambs, The (1991)	Crime—Horror—Thriller	42738
	Pulp Fiction (1994)	Comedy—Crime—Drama—Thriller	44626
	Schindler's List (1993)	Drama—War	35334
	Apollo 13 (1995)	Adventure—Drama—IMAX	26192
	Forrest Gump (1994)	Comedy—Drama—Romance—War	40422
DPALS	Silence of the Lambs, The (1991)	Crime—Horror—Thriller	42738
	Usual Suspects, The (1995)	Crime—Mystery—Thriller	33834
	Schindler's List (1993)	Drama—War	35334
	Pulp Fiction (1994)	Comedy—Crime—Drama—Thriller	44626
	Forrest Gump (1994)	Comedy—Drama—Romance—War	40422
	Braveheart (1995)	Action—Drama—War	32735
AdaDPALS	Silence of the Lambs, The (1991)	Crime—Horror—Thriller	42738
	Pulp Fiction (1994)	Comedy—Crime—Drama—Thriller	44626
	Usual Suspects, The (1995)	Crime—Mystery—Thriller	33834
	Forrest Gump (1994)	Comedy—Drama—Romance—War	40422
	Schindler's List (1993)	Drama—War	35334
	Braveheart (1995)	Action—Drama—War	32735
	Pinocchio (1940)	Animation—Children—Fantasy—Musical	5120
ALS	Snow White and the Seven Dwarfs (1937)	Animation—Children—Drama—Fantasy—Musical	7865
	Dumbo (1941)	Animation—Children—Drama—Musical	3580
	Cinderella (1950)	Animation—Children—Fantasy—Musical—Romance	3957
	Aristocats, The (1970)	Animation—Children	2669
	Fantasia (1940)	Animation—Children—Fantasy—Musical	6135
	Alice in Wonderland (1951)	Adventure—Animation—Children—Fantasy—Musical	3487
DPALS	Snow White and the Seven Dwarfs (1937)	Animation—Children—Drama—Fantasy—Musical	7865
	Jumanji (1995)	Adventure—Children—Fantasy	6203
	Beauty and the Beast (1991)	Animation—Children—Fantasy—Musical—Romance—IMAX	16391
	Aladdin (1992)	Adventure—Animation—Children—Comedy—Musical	19912
	Assassins (1995)	Action—Crime—Thriller	1146
	Miracle on 34th Street (1947)	Comedy—Drama	2445
AdaDPALS	Snow White and the Seven Dwarfs (1937)	Animation—Children—Drama—Fantasy—Musical	7865
	Fantasia (1940)	Animation—Children—Fantasy—Musical	6135
	Pocahontas (1995)	Animation—Children—Drama—Musical—Romance	2815
	Sword in the Stone, The (1963)	Animation—Children—Fantasy—Musical	2217
	Dumbo (1941)	Animation—Children—Drama—Musical	3580
	Jungle Book, The (1994)	Adventure—Children—Romance	2765

Table 3. Nearest neighbors according to ALS (non-private), DPALS, and AdaDPALS ($\mu = 1/3$).

Model	Movie title	Genres	Frequency
	Harry Potter and the Half-Blood Prince (2009)	Adventure—Fantasy—Mystery—Romance—IMAX	2176
ALS	Harry Potter and the Deathly Hallows: Part 1 (2010)	Action—Adventure—Fantasy—IMAX	2099
	Harry Potter and the Order of the Phoenix (2007)	Adventure—Drama—Fantasy—IMAX	2896
	Harry Potter and the Deathly Hallows: Part 2 (2011)	Action—Adventure—Drama—Fantasy—Mystery—IMAX	2265
	Sherlock Holmes (2009)	Action—Crime—Mystery—Thriller	2877
	Harry Potter and the Goblet of Fire (2005)	Adventure—Fantasy—Thriller—IMAX	4773
	Tangled (2010)	Animation—Children—Comedy—Fantasy—Musical—Romance—IMAX	1259
DPALS	Animatrix, The (2003)	Action—Animation—Drama—Sci-Fi	1216
	Ratatouille (2007)	Animation—Children—Drama	4728
	Avengers, The (2012)	Action—Adventure—Sci-Fi—IMAX	2770
	Tangled (2010)	Animation—Children—Comedy—Fantasy—Musical—Romance—IMAX	1259
	Slumdog Millionaire (2008)	Crime—Drama—Romance	5415
	Sherlock Holmes: A Game of Shadows (2011)	Action—Adventure—Comedy—Crime—Mystery—Thriller	1166
AdaDPALS	Harry Potter and the Deathly Hallows: Part 2 (2011)	Action—Adventure—Drama—Fantasy—Mystery—IMAX	2265
	Harry Potter and the Prisoner of Azkaban (2004)	Adventure—Fantasy—IMAX	6433
	Ratatouille (2007)	Animation—Children—Drama	4728
	Sherlock Holmes: A Game of Shadows (2011)	Action—Adventure—Comedy—Crime—Mystery—Thriller	1166
	Harry Potter and the Order of the Phoenix (2007)	Adventure—Drama—Fantasy—IMAX	2896
	Avatar (2009)	Action—Adventure—Sci-Fi—IMAX	4960
	Nausicaä of the Valley of the Wind (Kaze no tani no Naushika) (1984)	Adventure—Animation—Drama—Fantasy—Sci-Fi	2151
ALS	Laputa: Castle in the Sky (Tenkû no shiro Rapyuta) (1986)	Action—Adventure—Animation—Children—Fantasy—Sci-Fi	2227
	My Neighbor Totoro (Tonari no Totoro) (1988)	Animation—Children—Drama—Fantasy	3593
	Kiki's Delivery Service (Majo no takkyûbin) (1989)	Adventure—Animation—Children—Drama—Fantasy	1421
	Porco Rosso (Crimson Pig) (Kurenai no buta) (1992)	Adventure—Animation—Comedy—Fantasy—Romance	1022
	Grave of the Fireflies (Hotaru no haka) (1988)	Animation—Drama—War	2026
	Howl's Moving Castle (Hauru no ugoku shiro) (2004)	Adventure—Animation—Fantasy—Romance	3503
DPALS	Spirited Away (Sen to Chihiro no kamikakushi) (2001)	Adventure—Animation—Fantasy	9161
	Terminal, The (2004)	Comedy—Drama—Romance	1963
	Finding Neverland (2004)	Drama	3371
	Ring, The (2002)	Horror—Mystery—Thriller	3535
	Kill Bill: Vol. 1 (2003)	Action—Crime—Thriller	12467
	Man Who Wasn't There, The (2001)	Crime—Drama	2157
AdaDPALS	Spirited Away (Sen to Chihiro no kamikakushi) (2001)	Adventure—Animation—Fantasy	9161
	My Neighbor Totoro (Tonari no Totoro) (1988)	Animation—Children—Drama—Fantasy	3593
	Princess Mononoke (Mononoke-hime) (1997)	Action—Adventure—Animation—Drama—Fantasy	6101
	Ghost in the Shell (Kôkaku kidôtai) (1995)	Animation—Sci-Fi	4070
	Howl's Moving Castle (Hauru no ugoku shiro) (2004)	Adventure—Animation—Fantasy—Romance	3503
	Laputa: Castle in the Sky (Tenkû no shiro Rapyuta) (1986)	Action—Adventure—Animation—Children—Fantasy—Sci-Fi	2227
	Interstellar (2014)	Sci-Fi—IMAX	1048
ALS	Gone Girl (2014)	Drama—Thriller	847
	Edge of Tomorrow (2014)	Action—Sci-Fi—IMAX	881
	Gravity (2013)	Action—Sci-Fi—IMAX	1248
	Guardians of the Galaxy (2014)	Action—Adventure—Sci-Fi	1072
	Wolf of Wall Street, The (2013)	Comedy—Crime—Drama	1060
	Grand Budapest Hotel, The (2014)	Comedy—Drama	1339
DPALS	Day After Tomorrow, The (2004)	Action—Adventure—Drama—Sci-Fi—Thriller	1233
	Alive (1993)	Drama	994
	Spellbound (1945)	Mystery—Romance—Thriller	1365
	Source Code (2011)	Action—Drama—Mystery—Sci-Fi—Thriller	1595
	Fast and the Furious, The (2001)	Action—Crime—Thriller	1406
	Ice Storm, The (1997)	Drama	3020
AdaDPALS	Django Unchained (2012)	Action—Drama—Western	2692
	Inception (2010)	Action—Crime—Drama—Mystery—Sci-Fi—Thriller—IMAX	9147
	Dark Knight Rises, The (2012)	Action—Adventure—Crime—IMAX	2848
	Intouchables (2011)	Comedy—Drama	1803
	Source Code (2011)	Action—Drama—Mystery—Sci-Fi—Thriller	1595
	Avatar (2009)	Action—Adventure—Sci-Fi—IMAX	4960

Table 4. Nearest neighbors according to ALS (non-private), DPALS, and AdaDPALS ($\mu = 1/3$).