Private Learning with Public Features

Walid Krichene **Nicolas Mayoraz** **Steffen Rendle** **Shuang Song** **Abhradeep Thakurta** **Li Zhang**
Google

Abstract

We study a class of private learning problems in which the data is a join of *private and public features*. This is often the case in private personalization tasks such as recommendation or ad prediction, in which features related to individuals are sensitive, while features related to items (the movies or songs to be recommended, or the ads to be shown to users) are publicly available and do not require protection. A natural question is whether private algorithms can achieve higher utility in the presence of public features. We give a positive answer for multi-encoder models where one of the encoders operates on public features. We develop new algorithms that take advantage of this separation by only protecting certain sufficient statistics (instead of adding noise to the gradient). This method has a guaranteed utility improvement for linear regression, and importantly, achieves the state of the art on two standard private recommendation benchmarks, demonstrating the importance of methods that adapt to the private-public feature separation.

1 INTRODUCTION

Models trained on private user data risk leaking sensitive information (Korolova, 2010; Calandrino et al., 2011; Zhu et al., 2019), and Differential Privacy (DP) (Dwork et al., 2006) offers ways to quantify and limit this risk. One of the main challenges of DP training is that privacy guarantees come at the expense of losses in utility – the state of the art private models suffer significant quality losses compared to their non-private counterpart in many benchmarks. To mitigate these losses, a large body of work (e.g., Bassily et al. (2020); Yu et al. (2021a); Zhou et al. (2021); Golatkar et al. (2022); Li et al. (2022a); Amid et al. (2022); Bassily et al. (2023); Ganesh et al. (2023) among others) has explored methods to improve utility by leveraging public data sources. For example, it was shown that in image classification and certain language tasks, pre-training models on large public data can significantly improve utility of private models (Li et al., 2022c; De et al., 2022; Mehta et al., 2023; Ganesh et al., 2023).

The vast majority of these approaches assume access to *public training examples*. A different practical setting that remains under-explored–and which we propose to study–is having access to *public features*. This is often the case in personalization problems such as recommendation or ad prediction, where the same training example contains sensitive features about an individual, as well as features about an item that the individual interacts with (e.g., the recommended movie or song, or the advertised product). These item features are public and don't need privacy protection; for example, the director of a movie, the genre of a song, or the brand of a product are all public information. Formally, we assume access to a feature matrix X^{pub} that is public, and each training example contains *some row* of this matrix, together with private features and labels. Our goal is to design private algorithms adapted to this setting, and to study whether access to X^{pub} can improve privacy/utility trade-offs.

1.1 Contributions

1. We design new first-order algorithms for private learning with public features. The main algorithmic novelty is that instead of protecting gradients (as one would do in noisy gradient descent (Bassily et al., 2014b) and its variants), our algorithms work by computing and protecting certain sufficient statistics, that are then used to compute the final gradient. This has several practical advantages that we will discuss in detail; a notable one is that this allows us to preserve gradient sparsity, addressing a major practical issue of noisy gradient descent.
2. We give a utility analysis for linear regression (Theorem 3), where we compare the utility when some

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

features are public, to the utility when all features are private. We show that the excess empirical risk improves by a factor of $\sqrt{m/p}$ (m is the number of items, and p is the number of public features). In other words, the typical $\sqrt{p}$ scaling is replaced with a $\sqrt{m}$ scaling. To the best of our knowledge, this is the first analysis that shows a utility improvement under public features.

3. Empirically, we evaluate our methods on two standard private recommendation benchmarks (one classification task and one regression task from the MovieLens data sets (Harper and Konstan, 2016)). Our method achieves the DP state of the art on both benchmarks, demonstrating the importance of algorithms that adapt to the private-public feature structure. Besides improving utility, our method is also more computationally efficient, achieving better quality than DP-SGD at a fraction of the computational cost. We conclude with a discussion of limitations and open questions.

1.2 Related Work

Private learning with public data Several approaches have been proposed to augment private training with public data. Existing methods broadly fall into two categories. The first is to use the public data to modify the training procedure: a common theme is to project gradients on a low-dimensional manifold estimated using public data (Kairouz et al., 2021; Yu et al., 2021b; Zhou et al., 2021), or to augment the private loss with terms that depend on the public samples (e.g. a regularization term as in (Golatkar et al., 2022), or a Bregman divergence term as in (Amid et al., 2022)). The second approach is to pre-train a model on public data, then fine-tune it on private data. This approach has proven successful in domains where public data is plentiful, as in image classification (De et al., 2022; Mehta et al., 2023) and natural language (Li et al., 2022c). For a comprehensive survey, see (Cummings et al., 2023). These works assume access to a separate public data set (often from the same or similar distribution as the private data). Our setting is different, in that we assume that the training data include some private and some public features. In this sense, our approach is orthogonal to, and can be combined with the aforementioned methods.

Label DP Perhaps the closest setting to private training in the presence of public features is the notion of label DP (Chaudhuri and Hsu, 2011; Ghazi et al., 2023a, 2021). There, it is assumed that all features are public, and only labels are private. The setting of this paper bears some similarity with label DP, with two important distinctions: first, our setting aims to be more general, by allowing some but not all of the features to be public. Second, even when all features are public, our methods provide a stronger DP guarantee than label DP. This will be discussed in more detail in Remark 3, but in short, our methods protect *which row* of X^{pub} appears in a given training example (i.e., the correlation between the public features and the users), while label DP provides no such protection (it assumes that this is public knowledge).

Vertical Federated Learning The literature on Vertical Federated Learning (vFL) is concerned with training models on data sets which are vertically partitioned, i.e. different entities own different features (for example, entities could be hospitals owning data about the same individual). Some works have recently considered vFL with DP guarantees (Tran et al., 2023; Cohen-Addad et al., 2023; Li et al., 2023), although all assume that all features are sensitive, and none have considered the presence of public features, hence their algorithms are not applicable to our setting. Our presentation will focus on the centralized setting for simplicity. However, our algorithms have additional advantages in the federated setting, that will be discussed in Appendix F.

Private model personalization We will consider the class of multi-encoder models studied for example by Collins et al. (2021); Singhal et al. (2021); Jain et al. (2021); Shen et al. (2023). These models consist of learning a shared item encoder (that learns representations of items), together with a personalized user encoder (that learns individual preferences), as will be detailed in Section 2.2. Prior work has only considered the case where all features are private. We study the case where the item features are public, while user features and labels are private, and show that by designing algorithms that adapt to this separation, one can achieve better utility.

Finally, in recent work, Curmei et al. (2023) developed a method for private matrix completion with public features, a special case of our setting. They show strong empirical results (achieving the current SOTA on the DP MovieLens benchmarks), though their method is limited to matrix completion, and lacks utility guarantees. Our methods apply to a broader class of models, come with utility guarantees (albeit in a simplified case), and show substantial improvements on the same benchmarks.

1.3 Notation

Throughout the paper, m, n will denote the number of items and users respectively, while D denotes the number of training examples. Each user typi-

cally interacts with a small subset of items, so usually $m+n \ll D \ll mn$. For a vector x, we will write $\|x\|_2$ for the Euclidean norm of x. For a matrix X, we will denote by $\|X\|_F$ its Frobenius norm, and by $\|X\|_2$ its induced norm. We will denote by $\mathcal{N}^d$ a multivariate normal vector, and $\mathcal{N}^{d\times d}$ a symmetric matrix whose upper triangle entries are i.i.d. normal. Additional notation is summarized in Appendix A.

2 PROBLEM FORMULATION

2.1 DP with Public Features

First, we formalize the notion of private learning with public features. Let $X^{\text{pub}} \in \mathbb{R}^{m\times p}$ be a public feature matrix. To give a concrete example, one can think of rows of X^{pub} as representing items (e.g., movies to recommend, or ads to show to a user), and columns of X^{pub} as representing features, (e.g., all possible movie genres and cast, or all possible ad brand and price). Note that practical applications tend to have a large number of categorical features, so X^{pub} tends to be sparse.

We assume that each training example is of the form $(x^{\text{priv}}, y, x_j^{\text{pub}})$: it contains a private feature vector, a private label, together with some row x_j^{pub} of X^{pub}. For instance in a movie recommendation task, a training example might consist of a user's features x^{priv}, the features of a movie that this user watched (x_j^{pub}), and a rating y that the user assigned to the movie.

Remark 1. *One important detail is that we want to protect* which *row of the public matrix appears in each training example. In the above scenario, even though the features of a given movie are public, we want to protect which movies were seen by a user, as this is sensitive information. We formalize this by saying that X^{pub} is public, but the row index j is private.*

Formally, we will define the private dataset as follows

$$\mathcal{D} = \{(x_i^{\text{priv}}, y_i, j_i, k_i) \in \mathbb{R}^q \times \mathbb{R} \times [m] \times [n]\}_{i\in\{1,\dots,D\}}$$

where q is the dimension of private features, m is the number of items, and n is the number of users. Here j_i represents an item index (it indexes into X^{pub}), and k_i represents a user index (its only purpose is to allow for user-level privacy accounting).

Given the dataset definition above, we adhere to the usual notion of (approximate) differential privacy:

Definition 1. *A randomized algorithm $\mathcal{A}$ with output space $\mathcal{S}$ is said to be (ϵ, δ)-DP, if for all neighboring data sets $\mathcal{D}, \mathcal{D}'$, and all measurable $S \subset \mathcal{S}$,*

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

In example-level privacy, $\mathcal{D}, \mathcal{D}'$ are said to be neighboring data sets if one is obtained from the other by removing a single example. In user-level privacy, they are said to be neighboring if one is obtained from the other by removing all examples belonging to a user.

Remark 2 (Interpretation of DP with Public Features)**.** *Note that we use the standard DP definition. The only new assumption we make is that the private data includes an index $j_i \in [m]$, used to lookup a feature vector from a public matrix X^{pub}. The fact that X^{pub} is public means in particular that it doesn't change with the dataset $\mathcal{D}$.*

Remark 3 (Comparison to Label DP)**.** *Similar to our setting, label DP (Chaudhuri and Hsu, 2011; Ghazi et al., 2021), also defines a privacy guarantee where only part of the data needs protection. There are two distinctions with our setting: first, label DP assumes all features to be public, while DP with public features (our setting) allows for some features to be private. Second, even in the special case where all features are public, our setting provides a stronger notion of privacy. This is due to the definition of neighboring data sets: in label DP, neighboring data sets are only allowed to differ in a single label, and feature vectors are not allowed to change. As a consequence, participation in the process is not protected. In our setting, participation is protected (since completely removing the example yields a valid neighboring dataset). Another consequence is that label DP provides no protection of* which *public item appears in an example (since the feature vector is not allowed to change), while our setting does (the index j_i can change).*

Remark 4 (Comparison to Attribute DP)**.** *Finally, the notion of Attribute DP (Ghazi et al., 2023b) has been recently introduced to express per-attribute privacy. The guarantee replaces the constant ϵ in the usual definition with $\epsilon\|x - x'\|_0$. Here x, x' are the two examples that differ in the two neighboring data sets ($\mathcal{D}'$ is obtained from $\mathcal{D}$ by removing x and adding x'). The definition is meant to capture that the privacy guarantee depends on the number of attributes that change (the larger the change, the worse the guarantee). Attribute privacy does not directly model public features. One way to handle public features is to require that x, x' only differ in x^{priv}. This yields a definition similar to label DP (the public feature vector is not allowed to change), thus similar to Remark 3, it would protect neither participation, nor the mapping from example to public items.*

2.2 Multi-Encoder Models and Alternating Minimization

We consider multi-encoder models, studied for example by Agarwal and Chen (2009); Jain et al. (2021);

Shen et al. (2023), and commonly used in practice, in advertising (Agarwal and Chen, 2009), recommender systems (Covington et al., 2016; Volkovs et al., 2017), and similarity learning (Chopra et al., 2005; Schroff et al., 2015; Dong and Shen, 2018).

In our case, given a training example $(x^{\text{priv}}, x^{\text{pub}})$, the model's prediction is given by

$$f_{\theta_u,\theta_v}(x^{\text{priv}}, x^{\text{pub}}) = u_{\theta_u}(x^{\text{priv}}) \cdot v_{\theta_v}(x^{\text{pub}}), \quad (1)$$

where $v_{\theta_v} : \mathbb{R}^p \to \mathbb{R}^d$ is an encoder that maps the public features to an embedding in $\mathbb{R}^d$, $u_{\theta_u} : \mathbb{R}^q \to \mathbb{R}^d$ is a second encoder that maps the private features to another embedding in $\mathbb{R}^d$, and the model's prediction is the dot product of the two embeddings. Finally, $\theta_v \in \mathbb{R}^{d_v}, \theta_u \in \mathbb{R}^{d_u}$ are the parameters of the two encoders. We will refer to $v_{\theta_v}(\cdot)$ as the public encoder (since it operates on public features) and to $u_{\theta_u}(\cdot)$ as the private encoder. This architecture is well-suited to our problem, as it creates a separation between the private and public features, and as we shall see, our algorithms will take advantage of this separation.

Remark 5 (Personalized model interpretation)**.** *One interpretation of this model (see Jain et al. (2021); Shen et al. (2023)), is that the prediction task is decomposed into training a shared* item encoder *(v_{θ_v}) which learns a user-independent representation of items, together with a* personalized classifier *(u_{θ_u}) which captures each user's preferences. This point of view is also common in the collaborative filtering literature (Hu et al., 2008).*

The goal is to minimize the empirical risk

$$\sum_{i=1}^{D} \ell\left(u_{\theta_u}(x_i^{\text{priv}}) \cdot v_{\theta_v}(x_{j_i}^{\text{pub}}), y_i\right), \quad (2)$$

where ℓ is a loss function. A popular approach to solve this problem is the Alternating Minimization (AM) procedure described in Algorithm 1, where one alternates between optimizing one encoder (while the other is frozen) and vice-versa. The recent works of Chien et al. (2021); Jain et al. (2021); Collins et al. (2021); Singhal et al. (2021); Shen et al. (2023) all fall in this family of algorithms, and they only differ in how the encoders are optimized (i.e. in the sub-routines `UserUpdate` and `ItemUpdate`). None of them, however, considers the problem of training with public features. Our focus will be to design an algorithm for optimizing the *public encoder*. For the private encoder, any DP training procedure such as DP-SGD can be used.

Algorithm 1: Alternating Minimization

Inputs: Training data $\mathcal{D} = \{(x_i^{\text{priv}}, y_i, j_i, k_i)\}_{i\in\{1,\dots,D\}}$, number of outer steps S^{outer}
for $1 \le s \le S^{outer}$ **do**
  $\theta_u \leftarrow$ `UserUpdate`$(\theta_u, \theta_v, \mathcal{D})$
  $\theta_v \leftarrow$ `ItemUpdate`$(\theta_u, \theta_v, \mathcal{D})$
return θ_u, θ_v

3 PRIVATE TRAINING OF THE PUBLIC ENCODER

In this section, we design and analyze algorithms for optimizing the public encoder under a quadratic loss, i.e., $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$. We discuss in Appendix D the more general case where ℓ is convex.

For ease of notation, we will denote by $u_i = u_{\theta_u}(x_i^{\text{priv}})$. We can then write the loss (2) in a more concise form:

$$\mathcal{L}(\theta_v) = \frac{1}{2}\sum_{i=1}^{D}\left(u_i \cdot v_{\theta_v}(x_{j_i}^{\text{pub}}) - y_i\right)^2. \quad (3)$$

Note that since the private encoder is frozen, we only keep the explicit dependence on the public encoder parameters θ_v.

3.1 Gradient Decomposition under Public Features

The main observation is that we can factorize the gradient into terms that only depend on public features, and terms that only depend on private data. Let $\Omega_j = \{i \in [D] : j_i = j\}$ (in other words, Ω_j is the set of examples which contain item j). Then we have the following decomposition:

Proposition 1. *The gradient of the loss (3) is equal to:*

$$\nabla\mathcal{L}(\theta_v) = \sum_{j=1}^{m} \frac{\partial v_{\theta_v}(x_j^{pub})}{\partial \theta_v}[A_j v_{\theta_v}(x_j^{pub}) - b_j], \quad (4)$$

where, for all j,

$$A_j = \sum_{i\in\Omega_j} u_i u_i^\top, \qquad b_j = \sum_{i\in\Omega_j} y_i u_i. \quad (5)$$

Observe that in (4), both of the terms $v_{\theta_v}(x_j^{\text{pub}}) \in \mathbb{R}^d$ and $\partial v_{\theta_v}(x_j^{\text{pub}})/\partial\theta_v \in \mathbb{R}^{d_v \times d}$ only depend on public features (these terms correspond respectively to the public encoder's forward pass, and its Jacobian), while the terms $A_j \in \mathbb{R}^{d\times d}, b_j \in \mathbb{R}^d$ depend on private data (y_i is a private label, and $u_i = u_{\theta_u}(x_i^{\text{priv}})$ is a function of the private features). We will refer to A_j, b_j as

Algorithm 2: SSP-i: Sufficient Statistics Perturbation with Independent Noise

1 **Inputs**: Public features X^{pub}, training data $\mathcal{D} = \{(x_i^{\text{priv}}, y_i, j_i, k_i)\}_{i\in\{1,\dots,D\}}$, optional weights[1] $\{w_i\}$, number of steps T, clipping parameters Γ_y, Γ_u, noise standard deviation σ, learning rate η_t, initial parameters θ_v^0, θ_u^0.
2 Let $\bar{u}_i = \text{Clip}(u_{\theta_u}(x_i^{\text{priv}}), \Gamma_u)$, $\bar{y}_i = \text{Clip}(y_i, \Gamma_y)$
3 **for** $1 \le j \le m$ **do**
4 $\quad A_j \leftarrow \sum_{i\in\Omega_j} w_i \bar{u}_i \bar{u}_i^\top$
5 $\quad b_j \leftarrow \sum_{i\in\Omega_j} w_i \bar{y}_i \bar{u}_i$
6 **for** $0 \le t \le T-1$ **do**
7 $\quad$ **for** $1 \le j \le m$ **do**
8 $\quad\quad \hat{A}_j^t = A_j + \sigma\Gamma_u^2 \mathcal{N}^{d\times d}$
9 $\quad\quad \hat{b}_j^t = b_j + \sigma\Gamma_y\Gamma_u \mathcal{N}^d$
10 $\quad \hat{G}^t \leftarrow \sum_{j=1}^m \frac{\partial v_{\theta_v^t}(x_j^{\text{pub}})}{\partial \theta_v} [\hat{A}_j^t v_{\theta_v^t}(x_j^{\text{pub}}) - \hat{b}_j^t]$
11 $\quad \theta_v^{t+1} \leftarrow \theta_v^t - \eta_t \hat{G}^t$
12 **return** θ_v^T

Algorithm 3: SSP-c: Sufficient Statistics Perturbation with Correlated Noise

1 **Inputs**: Public features X^{pub}, training data $\mathcal{D} = \{(x_i^{\text{priv}}, y_i, j_i, k_i)\}_{i\in\{1,\dots,D\}}$, optional weights $\{w_i\}$, number of steps T, clipping parameters Γ_y, Γ_u, noise standard deviation σ, learning rate η_t, initial parameters θ_v^0, θ_u^0.
2 Let $\bar{u}_i = \text{Clip}(u_{\theta_u}(x_i^{\text{priv}}), \Gamma_u)$, $\bar{y}_i = \text{Clip}(y_i, \Gamma_y)$
3 **for** $1 \le j \le m$ **do**
4 $\quad \hat{A}_j \leftarrow \sum_{i\in\Omega_j} w_i \bar{u}_i \bar{u}_i^\top + \sigma\Gamma_u^2 \mathcal{N}^{d\times d}$
5 $\quad \hat{b}_j \leftarrow \sum_{i\in\Omega_j} w_i \bar{y}_i \bar{u}_i + \sigma\Gamma_y\Gamma_u \mathcal{N}^d$.
6 **for** $0 \le t \le T-1$ **do**
7 $\quad \hat{G}^t \leftarrow \sum_{j=1}^m \frac{\partial v_{\theta_v^t}(x_j^{\text{pub}})}{\partial \theta_v} [\hat{A}_j v_{\theta_v^t}(x_j^{\text{pub}}) - \hat{b}_j]$
8 $\quad \theta_v^{t+1} \leftarrow \theta_v^t - \eta_t \hat{G}^t$
9 **return** θ_v^T

the sufficient statistics for item j. A_j can be viewed as a partial private covariance matrix (partial in the sense that it's the covariance of just the examples that involve the j-th row of the public matrix).

3.2 Sufficient Statistics Perturbation Algorithms

The gradient decomposition in Proposition 1 suggests that one can achieve privacy protection by adding noise to the terms A_j, b_j, instead of the final gradient. We propose two algorithms based on this approach, see Algorithms 2 and 3.

Both operate by first computing the sufficient statistics for all items, then taking T steps of gradient descent, where the gradient is computed based on the noised statistics. They differ in how the noise is added (the differences are highlighted in blue). The first algorithm (SSP-i) samples independent noise at each iteration, while the second (SSP-c) only adds noise once, and reuses the noisy statistics across all iterations. Note that to achieve the same privacy, the noise standard deviations σ needed in the two algorithms are different.

Remark 6. *Reusing the noisy statistics in SSP-c is only possible due to the fact that A_j, b_j do not depend on the iterate θ_v^t (see eq.(5)), so they don't need to be recomputed as θ_v^t changes. One consequence is that the scale of the noise in SSP-c is independent of the number of steps T (as will be apparent in the privacy guarantee). However, this makes the noise in the gradient estimates correlated across iterations, which makes utility analysis difficult. For this reason, our utility analysis will be for SSP-i (the independent noise version). Empirically we evaluate both algorithms, and we find that SSP-c performs better.*

To give some intuition how this approach improves upon noisy GD, observe that the gradient is a vector in $\mathbb{R}^{d_v}$ (where d_v is the number of parameters of the encoder, which can be very large), while the right factor in the decomposition (4) (the term $A_j v_{\theta_v}(x_j^{\text{pub}}) + b_j$) is a vector in $\mathbb{R}^d$, where d is the *output dimension* of the encoder, and typically much smaller than d_v. Protecting the lower dimensional object is more efficient.

Preserving Gradient Sparsity Another practical advantage of adding noise to the sufficient statistics is that it allows us to preserve gradient sparsity. Sparse gradients are very common in models that use a large number of categorical features, which is the case in most personalization models (Cheng et al., 2016). Typically, only few features are active for a given item (and inactive features have 0 gradients). This sparsity will manifest in the Jacobian term in eq.(4), indeed, if the encoder is of the form $v(x_j^{\text{pub}}) = \nu(\theta_{in}^\top x_j^{\text{pub}})$ (θ_{in} represents an input embedding layer, and ν is the rest of the encoder), then its Jacobian w.r.t. θ_{in} is of the form $x_j^{\text{pub}} \rho_j^\top$ for some vector ρ_j, meaning that for any feature that is not active ($x_{jl}^{\text{pub}} = 0$) the corresponding l-th row in the Jacobian (and hence in the gradient) is also 0. See Appendix E for an illustration of the sparsity in our experiments.

[1] The optional weights w_i (used as weights in the sufficient statistics, see Lines 4-5) are useful in practice for user-level privacy: for example, by reducing the weights of a user who has many examples, this allows us to control the worst-case user sensitivity, see Appendix E.1.4. Theorem 2 gives a precise statement on how these weights affect the DP guarantee.

In noisy gradient descent and its variants, since noise is added to the final gradient, this destroys its sparsity– this was identified as one of the challenges of applying DP-SGD in practice (Zhang et al., 2021). In our SSP algorithms, since no noise is added to the Jacobian, its sparsity (and that of the gradient) are preserved.

3.3 Privacy Guarantees

We now state the privacy guarantee for both algorithms. All proofs are deferred to the appendix.

Theorem 1 (Example-level Privacy Guarantee)**.** *Let $w_i = 1$ for all i. Let $\epsilon, \delta > 0$ with $\epsilon < \log 1/\delta$. Then Algorithm 2 (resp. Algorithm 3) with standard deviation $\sigma = \frac{\sqrt{8T \log 1/\delta}}{\epsilon}$ (resp. $\sigma = \frac{\sqrt{8 \log 1/\delta}}{\epsilon}$) is (ϵ, δ)-DP.*

The main difference between the two algorithms is that noise scales with $\sqrt{T}$ in the independent noise version.

To state the user-level guarantee, it will be useful to define the set $\Omega^k = \{i : k_i = k\}$ (in other words, these are the indices of examples that belong to user k).

Theorem 2 (User-level Privacy Guarantee)**.** *Let $\bar{w}^2 = \max_k \sum_{i \in \Omega^k} w_i^2$. Let $\epsilon, \delta > 0$ with $\epsilon < \log 1/\delta$. Then Algorithm 2 (resp. Algorithm 3) with noise standard deviation $\sigma = \frac{\bar{w}\sqrt{8T \log 1/\delta}}{\epsilon}$ (resp. $\sigma = \frac{\bar{w}\sqrt{8 \log 1/\delta}}{\epsilon}$) is (ϵ, δ)-DP.*

3.4 Utility Analysis for Linear Encoders

Our goal in this section is to identify settings in which access to public features improves utility compared to the same problem where all features are private. We will consider linear encoders, i.e. $v_{\theta_v}(x_j^{\text{pub}}) = \theta_v^\top x_j^{\text{pub}}$ where $\theta_v \in \mathbb{R}^{p \times d}$. The problem is to minimize $\mathcal{L}(\theta_v) = \frac{1}{2}\sum_{i=1}^{D} ({x_{j_i}^{\text{pub}}}^\top \theta_v u_i - y_i)^2$, and the gradient factorization in Proposition 1 simplifies to

$$\nabla \mathcal{L}(\theta_v) = \sum_{j=1}^{m} x_j^{\text{pub}} [A_j \theta_v^\top x_j^{\text{pub}} - b_j]^\top \in \mathbb{R}^{p \times d}.$$

Assumption 1. *In this section, we assume that there exist $\Gamma_x, \Gamma_y, \Gamma_u$ such that for all j, $\|x_j^{pub}\|_2 \le \Gamma_x$, and for all i, $\|u_i\|_2 \le \Gamma_u$ and $|y_i| \le \Gamma_y$. Furthermore, we will optimize the problem over a bounded set $\Theta = \{\theta \in \mathbb{R}^{p \times d} : \max_j \|\theta^\top x_j^{pub}\|_2 \le \Gamma_y/\Gamma_u\}$.*

The boundedness assumptions on y, u are standard in private linear regression (Wang, 2018). The definition of the feasible set Θ simply requires that the model's output $(x_j^{\text{pub}} \theta^\top u_i)$ be of the same scale as the labels– this is often enforced in practice by normalizing the output of the encoder. This condition is for convenience (so that certain constants simplify).

We have the following bound on excess empirical risk:

Theorem 3 (Utility Guarantee of Algorithm 3)**.** *Suppose Assumption 1 holds. Let $\Gamma = \Gamma_x \Gamma_y \Gamma_u$. Let $\theta_v^* = \text{argmin}_{\theta_v \in \Theta} \mathcal{L}(\theta_v)$, and let $\hat{\theta}_v$ be the output of projected SSP-i run with $\sigma = \rho\sqrt{T}$, for $T = \frac{D^2}{md\rho^2}$ steps and with learning rates $\eta_t = \frac{|\Theta|}{\Gamma D\sqrt{8t}}$. Then*

$$\mathbb{E}[\mathcal{L}(\hat{\theta}_v)] - \mathcal{L}(\theta_v^*) = \tilde{\mathcal{O}}\left(|\Theta|\Gamma\rho\sqrt{md}\right),$$

where $\tilde{O}$ hides poly-log factors. Furthermore, setting $\rho = \frac{\sqrt{8 \log 1/\delta}}{\epsilon}$ guarantees that the algorithm is (ϵ, δ)-DP.

To highlight the implications of Theorem 3, we compare the bound to (projected) noisy gradient descent (Bassily et al., 2014b), where at each iteration

$$\theta_v^{t+1} = \Pi_\Theta \left[\theta_v^t - \eta \sum_{i=1}^{D} \text{Clip}(g_i(\theta_v^t), \Gamma) + \sigma\Gamma\mathcal{N}^{p \times d}\right],$$

where $g_i(\theta_v) = x_{j_i}^{\text{pub}}[u_i u_i^\top \theta_v^\top x_j^{\text{pub}} - y_i u_i]^\top$ (note that under Assumption 1, we have the bound $\|g_i\|_2 \le \Gamma = \Gamma_x\Gamma_u\Gamma_y$, hence we use Γ as the clipping constant). A standard analysis (Bassily et al., 2014b) shows that the excess empirical risk for noisy GD is bounded by $\tilde{\mathcal{O}}(|\Theta|\Gamma\rho\sqrt{pd})$. The main difference in Theorem 3 is that the dimension dependence $\sqrt{pd}$ is replaced with $\sqrt{md}$. Recall that d is the output dimension of the encoder, m is the number of public items, and p is the dimension of public features. For data with many categorical features, p can be orders of magnitude larger than m, especially when one uses feature crosses. In such cases, the bound of Theorem 3 suggests potentially large utility improvements.

Remark 7 (Reduced dimension dependence)**.** *We give another interpretation of Theorem 3. Note that in the setting of Theorem 3, the gradient of an example i is $x_{j_i}^{pub}\rho_i^\top \in \mathbb{R}^{p \times d}$ (where $\rho_i = ({x_{j_i}^{pub}}^\top \theta_v u_i - y_i)u_i \in \mathbb{R}^d$). Thus, the gradients lie in the subspace generated by $\{x_j e_l^\top, j \in [m], l \in [d]\}$, where e_l are the canonical basis vectors. This is a subspace of dimension md. Theorem 3 shows that the excess risk of SSP scales with the subspace dimension $\sqrt{md}$, instead of the ambient dimension $\sqrt{pd}$.*

It's worth observing that DP-SGD is known to achieve a similar improvement for unconstrained *convex problems (Song et al., 2021; Li et al., 2022b). We show that SSP achieves this improvement for* constrained *convex problems. Such an improvement is not possible without public features: (Bassily et al., 2014a, Theorem 5.4) gives a $\mathcal{O}(\sqrt{p/\epsilon})$ lower bound for constrained Lipschitz convex problems, where p is the ambient dimension. Note that this lower bound applies even when the feature matrix is low rank (as in our setting): the*

rank of the (public) feature matrix $X^{pub} \in \mathbb{R}^{p \times D}$ is no larger than the size of the data set D, thus for problems with few data points the low rank condition is satisfied.

3.5 Other Practical Considerations

Mini-batching Our algorithms were described in the full-batch setting for simplicity. In this section, we discuss their mini-batched variants.

In SSP-i, this can be achieved by replacing Lines 7-9 with the following: uniformly sample a batch of examples $B \subset [D]$, then compute the sufficient statistics over the batch, see Algorithm 4 in the appendix.

In SSP-c, mini-batching can be achieved by replacing Line 7 with the following: uniformly sample a batch $B \subset [m]$ of *items*, then let $\hat{G}^t = \sum_{j \in B} \frac{\partial v_{\theta_v^t}(x_j^{\text{pub}})}{\partial \theta_v} (\hat{A}_j v_{\theta_v^t}(x_j^{\text{pub}}) - \hat{b}_j)$, see Algorithm 5 in the appendix. Note that in this case, we sample items instead of training examples, which in many cases has a computational advantage that we discuss below.

This per-item sampling is not recommended, however, in DP-SGD or SSP-i, because sampling per-item precludes amplification by sampling (Bassily et al., 2014b; Abadi et al., 2016) which would lead to worse privacy guarantees. Per-item sampling is possible in SSP-c because we only add noise once and reuse the statistics across iterations (so there is no need for amplification).

Computational Complexity Per-item sampling in minibatch SSP-c has important implications on computational cost, which we now discuss.

First, notice that the gradient of an example i is $\mathcal{J}_{j_i}[u_i u_i^\top v_{j_i} - y_i u_i]$ (where we write $v_j = v_{\theta_v}(x_j^{\text{pub}})$ for the encoder's output, and $\mathcal{J}_j := \frac{\partial v_{\theta_v}(x_j^{\text{pub}})}{\partial \theta_v}$ for the Jacobian of item j). This involves computing a forward pass v_j and a partial back-propagation $\mathcal{J}_j$. Under per-record sampling (mini-batch DP-SGD or mini-batch SSP-i), an item j is revisited many times during one epoch (since it appears in all records in Ω_j), so the forward/backward passes for this item are recomputed at each visit. Whereas under per-item sampling (minibatch SSP-c), the forward/backward passes are computed exactly once per item per epoch, reducing the cost of gradient computation. At the same time, SSP-c has the additional overhead of computing the sufficient statistics, but since this is done once, one can hope to amortize it across iterations.

The following proposition compares the computational cost of minibatch SSP-c and DP-SGD–more precisely, we will compare to an optimized version of DP-SGD (that avoids recomputing the forward/backward passes if the same item appears more than once in the batch, see Algorithm 6 in the appendix).

Proposition 2. *Let c be the average cost of computing one forward/backward pass of the public encoder. Let β_j be the expected number of batches in which item j appears per epoch. Then the total expected cost over e epochs is bounded as follows: The cost of minibatch DP-SGD (Algorithm 6) is $\Omega(ce\sum_{j=1}^m \beta_j)$, and the cost of mini-batch SSP-c (Algorithm 5) is $\mathcal{O}(Dd^2 + (c + d^2)em)$.*

We give a detailed comparison in Appendix C under different regimes. Here, we discuss a simplified case. Suppose that $e \geq D/m$, then the cost of SSP-c becomes $\mathcal{O}((c + d^2)em)$, and we can more easily compare it to DP-SGD. The ratio (cost of SSP-c by cost of DP-SGD) is $\mathcal{O}\left(\frac{m(c+d^2)}{c\sum_{j=1}^m \beta_j}\right) = \mathcal{O}(\frac{m}{\beta}(1 + \frac{d^2}{c}))$, where $\beta = \sum_{j=1}^m \beta_j$. The first term $\frac{m}{\beta}$ is smaller than 1 (typically orders of magnitude smaller): indeed, the quantity β is at least m (the full batch case) and at most D (when the batch size is 1). It can be estimated precisely if one knows the item counts (we show examples in Appendix C).

Finally, to bound the term $\frac{d^2}{c}$, notice that if the encoder has at least one hidden layer of width $\geq d$ (a reasonable assumption, since d is the encoder's output dimension), then $c \geq d^2$, and the ratio simplifies to $\mathcal{O}(\frac{m}{\beta})$ which, as argued above, can be very small. Our experiments fall under this setting, and the improvement in our case is approximately two orders of magnitude (see next section).

4 EXPERIMENTAL RESULTS

We evaluate our methods on two standard private recommendation benchmarks (Jain et al., 2018; Chien et al., 2021; Krichene et al., 2023; Curmei et al., 2023), based on the MovieLens data sets (Harper and Konstan, 2016). The first is a regression task on MovieLens 10M (abbreviated as ML10M in the results), and the second is a classification task on MovieLens 20M (abbreviated as ML20M).

4.1 Experimental Setup

Data Each training example is a tuple $(x_i^{\text{priv}}, x_{j_i}^{\text{pub}}, y_i)$ where x_i^{priv} is a user's id, $x_{j_i}^{\text{pub}}$ is the feature vector of a movie, and y_i is a rating that the user assigned to the movie. The public features X^{pub} consist of metadata contained in the original MovieLens data (movie genre and release year) along with a richer set of metadata extracted from Wikidata by Curmei et al. (2023), containing information such as movie cast and fine-grained genres. Statistics about the data and features are detailed in Appendix E.1.

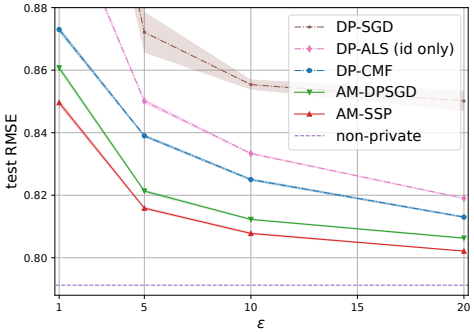


Figure 1: Privacy-RMSE trade-off in ML10M (lower is better).

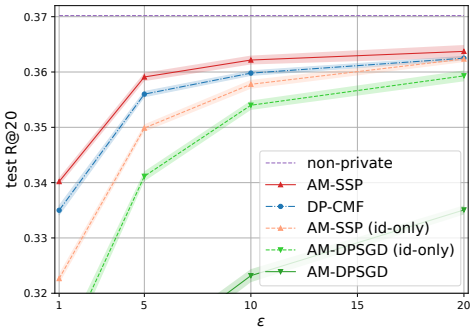


Figure 2: Privacy-Recall trade-off in ML20M (higher is better).

Models and Algorithms The current SOTA on these DP benchmarks consist of matrix completion methods: the DP-ALS algorithm of Chien et al. (2021) (which does not use features, only movie ids), and the DP-CMF algorithm of Curmei et al. (2023) (which factorizes a concatenation of the rating matrix and the public feature matrix). Matrix completion is a special case of the multi-encoder setting (1), where both encoders are linear and the feature vectors are one-hot.

In our experiments, we keep the same id-based user encoder, but train a more general item encoder, consisting of a wide embedding layer $\in \mathbb{R}^{p\times d}$ (which embeds the categorical features) followed by a dense layer $\in \mathbb{R}^{d\times d}$. The model is trained using alternating minimization (AM, see Algorithm 1) where `UserUpdate` is the same as in DP-ALS/DP-CMF, and we evaluate different methods for the `ItemUpdate`[2]. We consider two variants: AM-DPSGD uses DP-SGD for the `ItemUpdate`, and AM-SSP uses SSP-c for the `ItemUpdate` (we also compare SSP-i and SSP-c).

Evaluation Protocol We follow the exact protocol used in prior work on these benchmarks (Jain et al., 2018; Chien et al., 2021; Krichene et al., 2023; Curmei et al., 2023) and use the same code for data processing and privacy accounting. Each data set is split into training/validation/test, we tune hyper-parameters on the validation data and report metrics on test data (the shaded region on each plot shows the standard deviation across ten runs). Utility is measured using RMSE on ML10M and Recall@20 on ML20M. We report *user-level* (ϵ, δ) privacy guarantees, where $\delta = 10^{-5}$ for ML10M and 8.10^{-6} for ML20M. A fully detailed account of the setup (and additional experiments) can be found in Appendix E.

[2]Since we design algorithms for training the public encoder, our comparison uses the same user encoder and the same `UserUpdate`, so quality differences can be attributed to the public encoder and the `ItemUpdate`.

4.2 Privacy Utility Trade-off

We compare the privacy/utility trade-offs of the different algorithms. The results for ML10M are reported in Figure 1. We start by observing that feature-based methods (DP-CMF, AM-DPSGD and AM-SSP) all improve upon the DP-ALS baseline (which uses only item ids). Our method (AM-SSP) significantly improves upon the prior SOTA (DP-CMF), bringing utility much closer to the non-private baseline. The absolute improvement ranges from 0.025 (at $\epsilon = 1$) to 0.012[3] (at $\epsilon = 20$).

Turning to the ML20M benchmark (Figure 2), we see a more modest (though non-trivial) improvement compared to DP-CMF. One interesting observation, however, is that AM-DPSGD performs quite poorly on this benchmark: observe the large gap between AM-SSP and AM-DPSGD. This gap is much larger than the same gap on ML10M – this is consistent with our analysis in Section 3.4, which predicts a larger gap when the number of items is small (in ML20M the number of items is five times smaller, see Table 2).

To further investigate this performance gap, we apply the same methods to an id-only model that doesn't use any of the item features (the results are in dashed lines on the same figure). Remarkably, adding features improves quality under AM-SSP, but it *degrades it under AM-DPSGD*. Recall that SSP takes explicit advantage of the fact that movie features are public, while DPSGD doesn't. This is an example where adapting to the public-private feature separation leads to substantial gains, and without it, the benefits of side features are lost under DPSGD.

[3]To put this into perspective, an absolute improvement of 0.004 on the ML10M benchmark is considered significant, and some works have reported even smaller improvements, see (Rendle et al., 2019) for a survey.

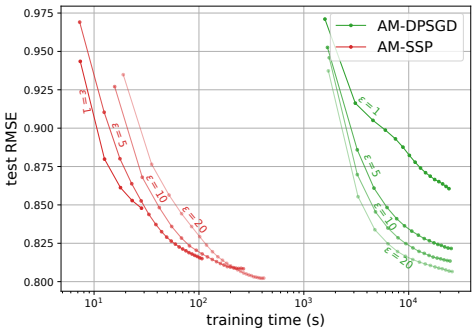


Figure 3: Time-evolution of test RMSE on ML10M.

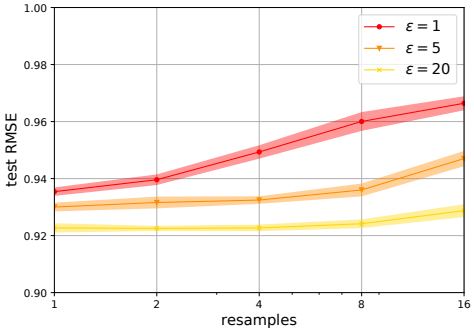


Figure 4: Comparison of SSP-i and SSP-c on ML10M.

4.3 Computational Cost

Besides quality improvements, we also highlight the computational advantage of AM-SSP. We plot, in Figure 3, the time-evolution of the test RMSE for AM-SSP and AM-DPSGD (with the optimal hyper-parameters tuned separately for each method). The results show approximately two-orders of magnitude difference in training time between the two methods. This illustrates the complexity advantage of SSP discussed in Section 3.5. The extent of the difference will generally depend on the problem parameters (such as number of items, sparsity, and batch size). This example shows that the improvement can be quite significant in practice (similar results are shown in the appendix for ML20M).

4.4 Comparing SSP-i and SSP-c

Finally, we assess the empirical difference between the SSP-i and SSP-c variants, see Figure 4. In this experiment, we only optimize the public encoder (i.e., without alternating minimization). Recall that SSP-i resamples noise at every step, while SSP-c adds noise once and reuses the statistics across iterations. Here we experiment with varying the number of times noise is resampled, so resamples=1 corresponds to SSP-c, while resamples=16 (16 is the total number of steps) corresponds to SSP-i. We find that as we increase the resampling frequency, quality degrades, and this appears to be more significant for lower ϵ (similar results hold on ML20M).

5 DISCUSSION

The AM-SSP method has several advantages in practice. It is specifically designed to take advantage of public features, it is computationally cheaper in many cases, it preserves gradient sparsity, and the fact that we can reuse noised statistics (see Remark 6) seems to further improve quality.

Beyond multi-encoder models This paper was focused on multi-encoder models due to their popularity, and the fact that their structure is naturally suited to the private-public feature separation. However, we expect that similar improvements can be achieved for more general architectures. Consider for example a general model with a large input embedding layer for the public features. The loss of such a model can be written as $f(W^{\text{pub}}x^{\text{pub}}, \dots)$, where W^{pub} are the weights of the public embedding layer. The gradient w.r.t. W^{pub} is of the form $\sum_i x^{\text{pub}}_{j_i} u_i^\top$ and one could define sufficient statistics in terms of the u_i. We leave such extensions for future work.

Limitations One limitation of the SSP approach is that the number of statistics to protect increases with the number of items. The method will generally work better when there are fewer items (and more examples per item). When too many fine-grained public features are used, this may lead to an increase in the number of unique items, and a degradation of quality. To give a hypothetical example, suppose font size is used as an ad feature. Then ads that are otherwise identical except for the font size will be treated as different items. We recommend excluding such fine-grained features from the public encoder to control the number of unique items (these excluded features can still be used in other parts of the model).

Open questions An interesting open question is to develop a more systematic approach to learning items that are *similar but not identical* (in the sense that their public feature vectors are close in some metric). One promising observation is that since the feature matrix X^{pub} is public, one can compute such feature-based similarity without paying a privacy cost.

Another direction is motivated by our empirical observation that SSP-c (the correlated noise version) performs better in practice than SSP-i (the independent noise version). Developing an analytical understanding of this variant is an interesting open question.

Acknowledgements

We are grateful to Prateek Jain, Fernando Pereira, Mukund Sundararajan, and the anonymous reviewers for their insightful comments and discussions.

References

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

D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 19–28, New York, NY, USA, 2009. Association for Computing Machinery.

E. Amid, A. Ganesh, R. Mathews, S. Ramaswamy, S. Song, T. Steinke, V. M. Suriyakumar, O. Thakkar, and A. Thakurta. Public data-assisted mirror descent for private model training. In *International Conference on Machine Learning*, pages 517–535. PMLR, 2022.

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

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

R. Bassily, S. Moran, and A. Nandi. Learning from mixtures of private and public populations. *Advances in Neural Information Processing Systems*, 33:2947–2957, 2020.

R. Bassily, M. Mohri, and A. T. Suresh. Principled approaches for private adaptation from a public source. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 8405–8432. PMLR, 2023.

K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

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

K. Chaudhuri and D. Hsu. Sample complexity bounds for differentially private learning. In *Proceedings of the 24th Annual Conference on Learning Theory*, volume 19 of *Proceedings of Machine Learning Research*, pages 155–186. PMLR, 2011.

H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 7–10, New York, NY, USA, 2016.

S. Chien, P. Jain, W. Krichene, S. Rendle, S. Song, A. Thakurta, and L. Zhang. Private alternating least squares: Practical private matrix completion with tighter rates. In *International Conference on Machine Learning*, pages 1877–1887. PMLR, 2021.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005.

V. Cohen-Addad, P. Kacham, V. Mirrokni, and P. Zhong. Differentially private vertical federated learning primitives. In *Theory and Practice of Differential Privacy*, 2023.

L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai. Exploiting shared representations for personalized federated learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2089–2099. PMLR, 2021.

P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 191–198, New York, NY, USA, 2016. Association for Computing Machinery.

R. Cummings, D. Desfontaines, D. Evans, R. Geambasu, M. Jagielski, Y. Huang, P. Kairouz, G. Kamath, S. Oh, O. Ohrimenko, et al. Challenges towards the next frontier in privacy. *arXiv preprint arXiv:2304.06929*, 2023.

M. Curmei, W. Krichene, L. Zhang, and M. Sundararajan. Private matrix factorization with public item features. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 805–812, New York, NY, USA, 2023. Association for Computing Machinery.

S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle. Unlocking high-accuracy differentially private image classification through scale. *CoRR*, abs/2204.13650, 2022.

X. Dong and J. Shen. Triplet loss in siamese network for object tracking. In *Computer Vision – ECCV 2018*, pages 472–488, Cham, 2018. Springer International Publishing.

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

A. Ganesh, M. Haghifam, M. Nasr, S. Oh, T. Steinke, O. Thakkar, A. Guha Thakurta, and L. Wang. Why is public pretraining necessary for private model training? In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10611–10627. PMLR, 2023.

B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, and C. Zhang. Deep learning with label differential privacy. In *Advances in Neural Information Processing Systems*, 2021.

B. Ghazi, P. Kamath, R. Kumar, E. Leeman, P. Manurangsi, A. Varadarajan, and C. Zhang. Regression with label differential privacy. In *The Eleventh International Conference on Learning Representations*, 2023a.

B. Ghazi, R. Kumar, P. Manurangsi, and T. Steinke. Algorithms with more granular differential privacy guarantees. In Y. T. Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 54:1–54:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023b.

A. Golatkar, A. Achille, Y.-X. Wang, A. Roth, M. Kearns, and S. Soatto. Mixed differential privacy in computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8376–8386, 2022.

I. Goodfellow. Efficient per-example gradient computations. *CoRR*, abs/1510.01799, 2015.

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

Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, 2008.

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

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

P. Kairouz, M. R. Diaz, K. Rush, and A. Thakurta. (nearly) dimension independent private erm with adagrad rates
via publicly estimated subspaces. In *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 2717–2746. PMLR, 2021.

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

W. Krichene, P. Jain, S. Song, M. Sundararajan, A. Guha Thakurta, and L. Zhang. Multi-task differential privacy under distribution skew. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17784–17807. PMLR, 2023.

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

T. Li, M. Zaheer, S. Reddi, and V. Smith. Private adaptive optimization with side information. In *Proceedings of the 39th International Conference*

on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pages 13086–13105. PMLR, 2022a.

X. Li, D. Liu, T. B. Hashimoto, H. A. Inan, J. Kulkarni, Y.-T. Lee, and A. Guha Thakurta. When does differentially private learning not suffer in high dimensions? In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 28616–28630. Curran Associates, Inc., 2022b.

X. Li, F. Tramer, P. Liang, and T. Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2022c.

Z. Li, T. Wang, and N. Li. Differentially private vertical federated clustering. *Proc. VLDB Endow.*, 16 (6):1277–1290, feb 2023.

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

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

H. Mehta, A. G. Thakurta, A. Kurakin, and A. Cutkosky. Towards large scale transfer learning for differentially private image classification. *Transactions on Machine Learning Research*, 2023.

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

S. Rendle, L. Zhang, and Y. Koren. On the difficulty of evaluating baselines: A study on recommender systems. *CoRR*, abs/1905.01395, 2019.

F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823. IEEE Computer Society, 2015.

Z. Shen, J. Ye, A. Kang, H. Hassani, and R. Shokri. Share your representation only: Guaranteed improvement of the privacy-utility tradeoff in federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, J. K. Rush, and S. Prakash. Federated reconstruction: Partially local federated learning. In *Advances in Neural Information Processing Systems*, 2021.

S. Song, T. Steinke, O. Thakkar, and A. Thakurta. Evading the curse of dimensionality in unconstrained private glms. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2638–2646. PMLR, 13–15 Apr 2021.

L. Tran, T. Castiglia, S. Patterson, and A. Milanova. Privacy tradeoffs in vertical federated learning. In *Federated Learning Systems (FLSys) Workshop @ MLSys 2023*, 2023.

M. Volkovs, G. Yu, and T. Poutanen. Dropoutnet: Addressing cold start in recommender systems. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Y. Wang. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018*, pages 93–103. AUAI Press, 2018.

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

D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz, et al. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*, 2021a.

D. Yu, H. Zhang, W. Chen, J. Yin, and T.-Y. Liu. Large scale private learning via low-rank reparametrization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12208–12218. PMLR, 2021b.

H. Zhang, I. Mironov, and M. Hejazinia. Wide network learning with differential privacy. *CoRR*, abs/2103.01294, 2021.

Y. Zhou, S. Wu, and A. Banerjee. Bypassing the ambient dimension: Private SGD with gradient subspace identification. In *International Conference on Learning Representations*, 2021.

L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] The setting is described in Section 2, the algorithms are described in Section 3.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] See discussion in Section 3.5, and further discussion in Appendix C.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] Code is included in the supplementary material, together with the tuned hyper-parameters for each model.
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] See Assumption 1.
   (b) Complete proofs of all theoretical results. [Yes] All proofs are provided in Appendix B.
   (c) Clear explanations of any assumptions. [Yes] We include a discussion following Assumption 1.
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] Code is available at the following repository: `https://github.com/google-research/google-research/tree/master/dp_alternating_minimization`.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] This is discussed in Section 4 (briefly, due to space constraints). Full details are given in Appendix E.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] The standard deviation of the metrics is measured and reported as a shaded region around the mean. Note that the standard deviation on ML10M is very small for most methods except the DP-SGD baseline (this is consistent with prior results reported on this benchmark) and this makes it hard to clearly see the shaded regions in Figure 1.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] See Appendix E.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] We reference the MovieLens data sets.
   (b) The license information of the assets, if applicable. [Yes] See Appendix E.
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes] New code is provided in the supplementary material.
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes] See Appendix E.
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A NOTATION

We summarize notation used throughout the paper in this section for easy reference.

For a positive integer n, we denote by $[n]$ the set $\{1, \dots, n\}$. Given a convex set $\mathcal{C}$, we denote by $\Pi_{\mathcal{C}}$ the Euclidean projection on $\mathcal{C}$ and by $\|\mathcal{C}\|$ the diameter of $\mathcal{C}$. For a vector x, we denote by $\mathrm{Clip}(x, \Gamma)$ the Euclidean projection of x on the L2 ball of radius Γ.

The data is given by

$$\mathcal{D} = \{(x_i^{\mathrm{priv}}, y_i, j_i, k_i) \in \mathbb{R}^q \times \mathbb{R} \times [m] \times [n]\}_{i \in \{1,\dots,D\}},$$

and we will always use i to index training examples, j to index items (i.e. rows in the public matrix X^{pub}), and k to index users (for user-level DP). We also use the following partitions of the data set: $\mathcal{D} = \sqcup_{j=1}^{m} \Omega_j = \sqcup_{k=1}^{n} \Omega^k$, where $\Omega_j = \{i : j_i = j\}$ is the set of examples that involve public item j, and $\Omega^k = \{i : k_i = k\}$ is the set of examples that belong to user k.

The model is given by

$$f_{\theta_u, \theta_v}(x_i^{\mathrm{priv}}, x_j^{\mathrm{pub}}) = v_{\theta_v}(x_j^{\mathrm{pub}}) \cdot u_{\theta_u}(x_i^{\mathrm{priv}}),$$

and we use the shorthand $u_i(\theta_u) = u_{\theta_u}(x_i^{\mathrm{priv}})$, and $v_j(\theta_v) = v_{\theta_v}(x_{j_i}^{\mathrm{pub}})$. This allows to emphasize the dependence on the encoder's parameters.

Finally, the dimensions of the problem are summarized in the table below. In the simple bilinear case (each encoder is linear), we have $d_u = qd$ and $d_v = pd$.

m	number of items
n	number of users
D	number of examples
p	dimension of public features ($x^{\mathrm{pub}} \in \mathbb{R}^p$)
q	dimension of private features ($x^{\mathrm{priv}} \in \mathbb{R}^q$)
d	encoders' output dimension ($v_j(\theta_v) \in \mathbb{R}^d$)
d_v, d_u	encoders' parameters ($\theta_u \in \mathbb{R}^{d_u}$, $\theta_v \in \mathbb{R}^{d_v}$)

Table 1: Dimension Parameters

B PROOFS

B.1 Proof of Theorem 1

Proof. The result is an application of the Gaussian mechanism. Let $\mathcal{D}, \mathcal{D}'$ be two neighboring data sets that differ by a single example $(x_i^{\mathrm{priv}}, x_{j_i}^{\mathrm{pub}}, y_i, j_i)$. This example only affects the statistics of item j_i (all other statistics remain unchanged). Denote by A_{j_i}, b_{j_i} (resp. A'_{j_i}, b'_{j_i} the statistics computed on data sets $\mathcal{D}$ (resp. $\mathcal{D}'$). Then

$$\|A_{j_i} - A'_{j_i}\|_F^2 = \|\bar{u}_i \bar{u}_i^\top\|_F^2 \le \Gamma_u^4,$$

thus releasing $A_{j_i} + \sigma \Gamma_u^2 \mathcal{N}^{d \times d}$ is $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP for all $\alpha > 1$. Similarly, we have

$$\|b_{j_i} - b'_{j_i}\|_2^2 = \|\bar{y}_i \bar{u}_i\|_2^2 \le \Gamma_u^2 \Gamma_y^2,$$

so releasing $b_{j_i} + \sigma \Gamma_u \Gamma_y \mathcal{N}^{d \times d}$ is $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP.

By RDP composition, we have that Algorithm 2 is $(\alpha, \alpha \frac{T \bar{w}^2}{\sigma^2})$-RDP (we release $2T$ noisy statistics), and Algorithm 3 is $(\alpha, \alpha \frac{\bar{w}^2}{\sigma^2})$-RDP (we release only 2 such statistics).

Translation from RDP to DP is standard, see for example (Mironov, 2017, Proposition 3): if a process is $(\alpha, \alpha\beta)$-RDP for all α, and $\beta \le \frac{\epsilon^2}{8 \log 1/\delta}$, then the process is (ϵ, δ)-DP. The result follows from setting $\beta = \frac{T}{\sigma^2}$ for Algorithm 2, and $\beta = \frac{1}{\sigma^2}$ for Algorithm 3. □

B.2 Proof of Theorem 2

Proof. Let $\mathcal{D}, \mathcal{D}'$ be two neighboring data sets that differ by a single user k. The user contributes to the sufficient statistics of all items $\{j_i, i \in \Omega^k\}$ (by definition of Ω^k). Define A to be the matrix $[A_1|\dots|A_m] \in \mathbb{R}^{d\times md}$, and let A' be the same matrix computed on $\mathcal{D}'$ instead of $\mathcal{D}$. Then

$$\|A - A'\|_F^2 = \sum_{i\in\Omega^k} \|w_i \bar{u}_i \bar{u}_i^\top\|_F^2 \leq \sum_{i\in\Omega^k} w_i^2 \Gamma_u^4 \leq \bar{w}^2 \Gamma_u^4$$

where we used the definition of $\bar{w}^2 = \max_k \sum_{i\in\Omega^k} w_i^2$. Similarly, defining $b = [b_1, \dots, b_m] \in \mathbb{R}^{d\times m}$ we have that

$$\|b - b'\|_F^2 = \sum_{i\in\Omega^k} \|w_i \bar{u}_i y_i\|_2^2 \leq \sum_{i\in\Omega^k} w_i^2 \Gamma_u^2 \Gamma_y^2 \leq \bar{w}^2 \Gamma_u^2 \Gamma_y^2.$$

Therefore, releasing $A + \Gamma_u^2 \sigma^2 \mathcal{N}^{d\times md}$ is $(\alpha, \frac{\alpha \bar{w}^2}{2\sigma^2})$-RDP, and so is releasing $b + \sigma \Gamma_u \Gamma_y \mathcal{N}^{d\times m}$.

The rest of the proof proceeds as in the proof of Theorem 1. □

B.3 Proof of Theorem 3

Proof. We will use the following standard lemma, which can be found for example in (Bassily et al., 2014b, Lemma 2.5).

Lemma 1. *Let $\mathcal{L}$ be a convex function defined on a bounded domain Θ, and let $\theta_v^* = \operatorname{argmin}_{\theta_v\in\Theta} \mathcal{L}(\theta_v)$. Consider the SGD algorithm*

$$\theta_v^{t+1} = \Pi_\Theta \left[\theta_v^t - \eta_t \hat{g}^t\right],$$

where $\hat{g}^t$ satisfy the following: $\mathbb{E}[\hat{g}^t] = \nabla\mathcal{L}(\theta_v^t)$, and $\mathbb{E}\|\hat{g}^t\|_2^2 \leq \mathcal{G}^2$. Let $\eta_t = \frac{|\Theta|}{\mathcal{G}\sqrt{t}}$. Then for all t,

$$\mathbb{E}[\mathcal{L}(\theta_v^t)] - \mathcal{L}(\theta_v^*) = \mathcal{O}\left(\frac{|\Theta|\mathcal{G}\log t}{\sqrt{t}}\right).$$

To apply the lemma, we will show that $\hat{G}^t$ (Line 10 in Algorithm 2) is an unbiased estimate of the gradient and bound its second moment. To make the notation concise, we will write $v_j^t = v_{\theta_v^t}(x_j^{\text{pub}})$. Recall that the true gradient is

$$\nabla\mathcal{L}(\theta_v) = \sum_{j=1}^m x_j^{\text{pub}} (A_j v_j - b_j)^\top = \sum_{j=1}^m x_j^{\text{pub}} \rho_j^\top$$

where we defined $\rho_j = A_j v_j - b_j$. And by definition of the SSP-i algorithm, the gradient estimate is

$$\begin{aligned}
\hat{G}^t &= \sum_{j=1}^m x_j^{\text{pub}} [\hat{A}_j^t v_j^t - \hat{b}_j^t]^\top \\
&= \sum_{j=1}^m x_j^{\text{pub}} [(A_j + \sigma\Gamma_u^2 \mathcal{N}^{d\times d}) v_j^t - (b_j + \sigma\Gamma_y\Gamma_u \mathcal{N}^d]^\top \\
&= \nabla\mathcal{L}(\theta_v^t) + \sigma \sum_{j=1}^m x_j^{\text{pub}} [\Gamma_u^2 \mathcal{N}^{d\times d} v_j^t - \Gamma_y\Gamma_u \mathcal{N}^d]^\top,
\end{aligned}$$

where, importantly, the noise samples $\mathcal{N}^{d\times d}, \mathcal{N}^d$ are conditionally independent of the trajectory up to step t. Thus, it follows by independence that $\mathbb{E}[\hat{G}^t] = \nabla\mathcal{L}(\theta_v^t)$, and we can bound the second moment

$$\mathbb{E}[\|\hat{G}^t\|_2^2] = \mathbb{E}[\|\nabla\mathcal{L}(\theta_v^t)\|_2^2] + \sigma^2 \sum_{j=1}^m \mathbb{E}[\|x_j^{\text{pub}} [\Gamma_u^2 \mathcal{N}^{d\times d} v_j^t - \Gamma_y\Gamma_u \mathcal{N}^d]^\top\|_2^2] \tag{6}$$

(the cross-terms vanish by independence). It remains to bound individual terms. First, observe that

$$\|\rho_j\|_2 = \|A_j v_j - b_j\|_2 \le \|A_j\|_2 \|v_j\|_2 + \|b_j\|_2,$$

where $\|A_j\|_2 = \|\sum_{i\in\Omega_j} u_i u_i^\top\|_2 \le |\Omega_j|\Gamma_u^2$, $\|v_j\|_2 \le \Gamma_y/\Gamma_u$ (by Assumption 1), and $\|b_j\|_2 = \|\sum_{i\in\Omega_j} y_i u_i\|_2 \le |\Omega_j|\Gamma_y\Gamma_u$. Thus $\|\rho_j\|_2 \le 2|\Omega_j|\Gamma_y\Gamma_u$, and

$$\begin{aligned}\|\nabla\mathcal{L}(\theta_v^t)\|_2 &\le \sum_{j=1}^m \|x_j^{\text{pub}}\rho_j^\top\|_2 \le \sum_{j=1}^m \|x_j^{\text{pub}}\|_2\|\rho_j\|_2 \\ &\le \sum_{j=1}^m 2\Gamma_x|\Omega_j|\Gamma_y\Gamma_u \\ &= 2D\Gamma_x\Gamma_y\Gamma_u \end{aligned} \tag{7}$$

where we used that $\sum_j |\Omega_j| = D$. This gives a bound on the first term in (6).

Turning to the second term in (6), we have that $\mathbb{E}[\|\mathcal{N}^{d\times d}\|_2] = \sqrt{d}$ (induced norm) and $\mathbb{E}[\|\mathcal{N}^d\|_2] = \sqrt{d}$ (Euclidean norm), thus

$$\mathbb{E}[\|x_j^{\text{pub}}[\Gamma_u^2\mathcal{N}^{d\times d}v_j^t - \Gamma_y\Gamma_u\mathcal{N}^d]^\top\|_F] \le 2\Gamma_x\Gamma_y\Gamma_u\sqrt{d}. \tag{8}$$

Combining bounds (7)-(8) into (6), we get

$$\mathbb{E}[\|\hat{G}^t\|_2^2] \le [2\Gamma_x\Gamma_y\Gamma_u]^2(D^2 + \sigma^2 md)$$

Next, taking $\sigma^2 = \rho^2 T$ and applying Lemma 1 with $\mathcal{G} = 2\Gamma_x\Gamma_y\Gamma_u\sqrt{D^2 + mdT\rho^2}$, we get

$$\mathbb{E}[\mathcal{L}(\theta_v^T)] - \mathcal{L}(\theta_v^*) = \tilde{\mathcal{O}}\left(|\Theta|2\Gamma_x\Gamma_y\Gamma_u\sqrt{\frac{D^2}{T} + md\rho^2}\right).$$

Finally we choose T to equalize the two terms, $T = \frac{D^2}{md\rho^2}$ (under this choice of T, $\mathcal{G} = 2\sqrt{2}\Gamma_x\Gamma_y\Gamma_u D$), to obtain the final desired bound $\mathcal{O}\left(2|\Theta|\Gamma_x\Gamma_y\Gamma_u\rho\sqrt{2md}\right)$. □

C BATCHED ALGORITHMS AND COMPLEXITY ANALYSIS

In this section, we give additional details related to the batched variants of our algorithms, and discuss computational complexity.

The mini-batched versions of SSP-i and SSP-c are given respectively in Algorithms 4 and 5, where we highlight differences compared to the full-batch case in blue.

For mini-batch DP-SGD, notice that if several examples in a batch have the same feature vector x_j^{pub}, then the forward/backward passes $v_j = v_{\theta_v}(x_j^{\text{pub}})$ and $\mathcal{J}_j = \frac{\partial v_{\theta_v}(x_j^{\text{pub}})}{\partial\theta_v}$ only have to be computed once. This is summarized in Algorithm 6.

Proof of Proposition 2 We now turn to the complexity analysis of mini-batch SSP-c and mini-batch DP-SGD.

Proof. Let c be the cost of computing one forward and one backward pass, i.e. the cost of computing v_j and $\mathcal{J}_j$ for one item j. We also recall the following quantities: β_j is the expected number of batches that contain item j in one epoch, and $\beta = \sum_{j=1}^m \beta_j$.

First, consider DP-SGD. For some item j, every time the item is visited, we compute one forward/backward pass for that item (Lines 6-7), for a cost of $c\sum_{j=1}^m \beta_j = c\beta$, hence the cost of DP-SGD over e epochs is at least $\Omega(ce\beta)$. Note that there is the additional cost of computing and clipping the gradients (Line 8), but for the purposes of our analysis, we only need a lower bound on the cost of DP-SGD (we will compare an upper bound on the cost of SSP-c to this lower bound on the cost of DP-SGD).

Algorithm 4: Mini-batch SSP-i

Inputs: Public features X^{pub}, training data $\mathcal{D} = \{(x_i^{\text{priv}}, y_i, j_i, k_i)\}_{i\in\{1,\dots,D\}}$, optional weights $\{w_i\}$, number of steps T^D, clipping parameters Γ_y, Γ_u, noise standard deviation σ, learning rate η_t, initial parameters θ_v^0, θ_u^0.
Let $\bar{u}_i = \text{Clip}(u_{\theta_u}(x_i^{\text{priv}}), \Gamma_u)$, $\bar{y}_i = \text{Clip}(y_i, \Gamma_y)$
for $0 \le t \le T^D - 1$ **do**
  Uniformly sample $B \subset [D]$
  for $1 \le j \le m$ **do**
    $\hat{A}_j^t = \sum_{i\in\Omega_j\cap B} w_i \bar{u}_i \bar{u}_i^\top + \sigma\Gamma_u^2 \mathcal{N}^{d\times d}$
    $\hat{b}_j^t = \sum_{i\in\Omega_j\cap B} w_i \bar{y}_i \bar{u}_i + \sigma\Gamma_y\Gamma_u \mathcal{N}^d$
  $\hat{G}^t \leftarrow \sum_{j=1}^m \frac{\partial v_{\theta_v^t}(x_j^{\text{pub}})}{\partial \theta_v} (\hat{A}_j^t v_{\theta_v^t}(x_j^{\text{pub}}) - \hat{b}_j^t)$
  $\theta_v^{t+1} \leftarrow \theta_v^t - \eta_t \hat{G}^t$
return θ_v^T

Algorithm 5: Mini-batch SSP-c

Inputs: Public features X^{pub}, training data $\mathcal{D} = \{(x_i^{\text{priv}}, y_i, j_i, k_i)\}_{i\in\{1,\dots,D\}}$, optional weights $\{w_i\}$, number of steps T^I, clipping parameters Γ_y, Γ_u, noise standard deviation σ, learning rate η_t, initial parameters θ_v^0, θ_u^0.
Let $\bar{u}_i = \text{Clip}(u_{\theta_u}(x_i^{\text{priv}}), \Gamma_u)$, $\bar{y}_i = \text{Clip}(y_i, \Gamma_y)$
for $1 \le j \le m$ **do**
  $\hat{A}_j \leftarrow \sum_{i\in\Omega_j} w_i \bar{u}_i \bar{u}_i^\top + \sigma\Gamma_u^2 \mathcal{N}^{d\times d}$
  $\hat{b}_j \leftarrow \sum_{i\in\Omega_j} w_i \bar{y}_i \bar{u}_i + \sigma\Gamma_y\Gamma_u \mathcal{N}^d$.
for $0 \le t \le T^I - 1$ **do**
  Uniformly sample $B \subset [m]$
  $\hat{G}^t \leftarrow \sum_{j\in B} \frac{\partial v_{\theta_v^t}(x_j^{\text{pub}})}{\partial \theta_v} (\hat{A}_j v_{\theta_v^t}(x_j^{\text{pub}}) - \hat{b}_j)$
  $\theta_v^{t+1} \leftarrow \theta_v^t - \eta_t \hat{G}^t$
return θ_v^T

Algorithm 6: Mini-batch DP-SGD (adapted to multi-encoder models for improved efficiency)

Inputs: Public features X^{pub}, training data $\mathcal{D} = \{(x_i^{\text{priv}}, y_i, j_i, k_i)\}_{i\in\{1,\dots,D\}}$, optional weights $\{w_i\}$, number of steps T^D, clipping parameters Γ_g, noise standard deviation σ, learning rate η_t, initial parameters θ_v^0, θ_u^0.
Let $u_i = u_{\theta_u^0}(x_i^{\text{priv}})$
for $0 \le t \le T^D - 1$ **do**
  Uniformly sample a batch $B \subset [D]$.
  for $j \in \{j_i : i \in B\}$ **do**
    $v_j \leftarrow v_{\theta_v}(x_j^{\text{pub}})$
    $\mathcal{J}_j \leftarrow \frac{\partial v_{\theta_v}(x_j^{\text{pub}})}{\partial \theta_v}$
  $\hat{G}^t \leftarrow \sum_{i\in B} \text{Clip}\left(\mathcal{J}_{j_i}(u_i^T v_{j_i} - y_i) u_i, \Gamma_g\right) + \sigma\Gamma\mathcal{N}^{d_v}$
  $\theta_v^{t+1} \leftarrow \theta_v^t - \eta_t \hat{G}^t$
return θ_v^T

We now consider SSP-c. First, there is the cost of computing the sufficient statistics (Lines 3-5). This can be done by iterating over all examples in $\mathcal{D}$ and accumulating the statistics. This costs $\mathcal{O}(D(d^2+d))$ (d^2 for accumulating the $\hat{A}_j$'s, and d for accumulating the $\hat{b}_j$'s). Second, for each batch, we compute the gradient $\hat{G}^t$ (Line 8). This consists of the following operations for each $j \in B$: computing the forward/backward passes $v_j, \mathcal{J}_j$ (a cost of c), then computing the vector $\hat{A}_j v_j - \hat{b}_j$ (a cost of d^2), and finally multiplying this vector by $\mathcal{J}_j$ (a cost of at most c, because the cost of computing the Jacobian is greater than the cost of multiplying by the Jacobian). The total cost for item j is therefore $\mathcal{O}(c+d^2)$. Finally, the total cost over e epochs $\mathcal{O}(Dd^2 + em(c+d^2))$ (the first term is the cost of computing statistics, and the second is the cost of computing gradients from the statistics). This completes the proof. □

Comparison of Computational Complexity We expand on the complexity discussion of Section 3.5. The ratio between the cost of SSP-c and the cost of DP-SGD (see Proposition 2) is bounded by

$$\mathcal{O}\left(\frac{Dd^2 + em(c+d^2)}{ce\beta}\right) = \mathcal{O}\left(\frac{m}{\beta}\left(1 + \frac{d^2}{c} + \frac{Dd^2}{cem}\right)\right) \tag{9}$$

This depends on several problem parameters:

- β/m: this represents the average number of visits per item: the larger this number is, the bigger the advantage of SSP-c.
- $\frac{d^2}{c}$: this term depends on the encoder's architecture. c is the cost of one forward/backward pass, and d is the output dimension of the encoder.
- the last term $\frac{Dd^2}{cem}$ represents the relative overhead of computing statistics in SSP, and decreases with the number of epochs e (if the number of epochs is large enough, the overhead is amortized).

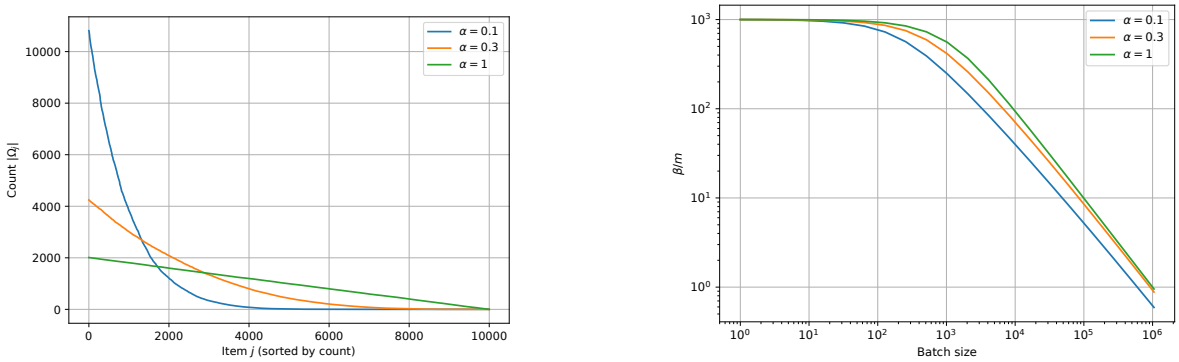
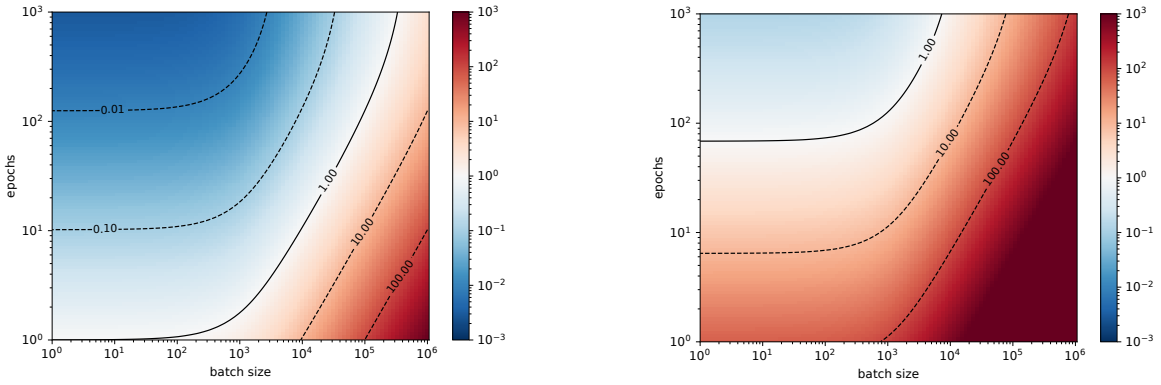


Figure 5: Item visits under a power law distribution. In this example the data set size is $D = 10^6$ and the number of items is $m = 10^3$. The distribution of item counts follows a power law distribution with density $f(x) \propto x^{-\alpha}$ where α is a positive parameter. The left plot shows the count distribution, and the right plot shows the average number of visits β/m as we vary the batch size B.

First, consider the term β/m. Assuming that examples are sampled independently and with replacement, we can get a precise estimate of β: let $p_j = \frac{|\Omega_j|}{D}$ be the probability of sampling item j. For a batch size B, the probability that item j appears (at least once) in the batch is $q_j = 1-(1-p_j)^B$. The number of batches in which item j appears is a Binomial distribution with probability q_j, and since there are D/B batches in one epoch, the expected number of batches containing j in one epoch is $\beta_j = D/B(1-(1-p_j)^B)$. This is entirely determined by the item frequencies p_j, the data size D, and the batch size B. We plot a few examples in Figure 5, where we take the item count distribution to follow a power-law distribution, commonly encountered in practice in recommender systems and ads applications (Yin et al., 2012; Liu and Zheng, 2020). The plot shows that as the batch size increases, the total number of visits β decreases. When $B = 1$, $\beta = D$, and when $B = D$ (full batch), $\beta/m \approx 1$ (if the batch is all of $\mathcal{D}$, then $\beta = m$, but since we are sampling with replacement, some items may not be sampled at all so one could have β slightly lower than m as can be seen on the plot). In the example, we take $D = 10^6$ and $m = 10^3$. Notice that β/m remains large (more than a hundred) for batch sizes up to a few thousands.



(a) $c = d^2$ (b) $c = d$

Figure 6: Complexity ratio (cost of SSP divided by cost of DP-SGD) as we vary the batch size and number of epochs. In this example, $D = 10^6, m = 10^3$. Two representative cases are shown: $c = d^2$ corresponding to encoders with at least one hidden layer (left), and $c = d$, corresponding to id-only linear encoders (right). The solid line shows the case where both algorithms have comparable cost.

Once β/m is estimated, we can easily compute the ratio (9). In Figure 6, we plot the ratio (9) as we vary the batch size and the number of epochs. We fix the data size $D = 10^6$ (changing this parameter will change the scale of the graph, but the trends will be similar).

As discussed above, the ratio $\frac{d^2}{c}$ depends on the model architecture, we consider two representative cases: the first is when the encoder contains at least a hidden layer of width larger than d, in which case we have $c \geq d^2$ (whenever the encoder has a hidden layer, its width is usually more than the width of the last layer). The second case is when the encoder is a single linear layer, in which case $c = kd$ where k is the number of non-zero features per item. In the simplest case (the most favorable to DP-SGD), there is a single active feature per item and $c = d$.

In the case $c = d^2$ (Figure 6a), there is a large region in which SSP-c has a computational advantage, and the advantage increases as the number of epochs increases and as the batch size decreases. The advantage can easily be several orders of magnitude in the small batch regime (batch size of 100-1000).

The case $c = d$ (Figure 6b) is intuitively the least favorable to SSP-c, since the cost of computing the statistics (d^2) is much higher than the cost of the encoder ($c = d$), so it takes longer to amortize the cost of computing the statistics. In this case, we expect SSP-c to be more expensive, unless the number or epochs is very large.

D SSP ALGORITHM WITH NON-QUADRATIC LOSSES

In this section, we discuss a generalization of our algorithms from the quadratic to the convex loss case. The main idea is to compute and optimize successive quadratic approximations of the loss, much like in second-order methods. The main difference is that instead of computing the approximation in the θ_v space (which would be intractable in most practical settings), we compute it in the v space, i.e. a quadratic approximation w.r.t. *the encoder's output.*

More precisely, consider the loss (2), reproduced below

$$\mathcal{L}(\theta_v) = \sum_{i=1}^{D} \ell\left(u_i \cdot v_{j_i}(\theta_v), y_i\right),$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is convex (note however that $\mathcal{L}$ is not, since the item encoder v_{θ_v} is typically non-linear). Writing $\ell_i(v) = \ell(u_i \cdot v, y_i)$, the loss $\mathcal{L}$ is the composition $\mathcal{L}(\theta_v) = \sum_{i=1}^{D} \ell_i(v_{j_i}(\theta_v))$.

By the chain rule, the gradient and Hessian of ℓ_i at v^0 are given respectively by $\nabla \ell_i(v^0) = \ell'(u_i \cdot v^0, y_i) u_i$ and $\nabla^2 \ell_i(v^0) = \ell''(u_i \cdot v^0, y_i) u_i u_i^\top$, where ℓ' and ℓ'' denote the first and second derivatives of ℓ w.r.t. its first argument. To simplify notation, let us write $\ell_i(v^0) = \ell(u_i \cdot v^0, y_i)$, $d\ell_i(v^0) = \ell'(u_i \cdot v^0, y_i)$ and $d^2\ell_i(v^0) = \ell''(u_i \cdot v^0, y_i)$.

Now, a simple Taylor expansion of ℓ_i around v^0 yields the quadratic approximation

$$\tilde{\ell}_i(v) = \ell_i(v^0) + d\ell_i(v^0)u_i \cdot (v - v^0) + \frac{1}{2}d^2\ell_i(v^0)(v - v^0)^\top u_i u_i^\top (v - v^0).$$

Denoting $v_j^0 = v_j(\theta_v^0)$, we obtain the following approximation of $\mathcal{L}$ around θ_v^0:

$$\tilde{\mathcal{L}}(\theta_v) = \sum_{j=1}^m c_j + g_j^\top (v_j(\theta_v) - v_j^0) + \frac{1}{2}(v_j(\theta_v) - v_j^0)^\top H_j (v_j(\theta_v) - v_j^0), \tag{10}$$

where

$$c_j = \sum_{i \in \Omega_j} \ell_i(v^0), \qquad g_j = \sum_{i \in \Omega_j} d\ell_i(v_j^0)u_i, \qquad H_j = \sum_{i \in \Omega_j} d^2\ell_i(v_j^0)u_i u_i^\top, \tag{11}$$

Since this is a quadratic function in v, this gives us a similar gradient decomposition, which we state below.

Proposition 3. *The gradient of the quadratic approximation* (10) *is given by*

$$\nabla\tilde{\mathcal{L}}(\theta_v) = \sum_{j=1}^m \mathcal{J}_j(\theta_v)[H_j(v_j(\theta_v) - v_j^0) + g_j], \tag{12}$$

where we use the shorthands $\mathcal{J}_j(\theta_v) = \frac{\partial v_{\theta_v}(x_j^{pub})}{\partial \theta_v}$ *and* $v_j(\theta_v) = v_{\theta_v}(x_j^{pub})$.

As in Proposition 1, the terms $\mathcal{J}_j$ and v_j only depend on public data and don't need protection. The only difference is in the sufficient statistics: Instead of A_j, b_j, the decomposition now involves H_j, g_j. This motivates Algorithm 7. The main difference with Algorithm 2 is that H_j, g_j depend on where we take the approximation (note the dependence on v^0 in (11)), so they need to be periodically recomputed.

Algorithm 7: SSP-convex: Sufficient Statistics Perturbation for Convex Losses (with correlated noise)

Inputs: Public features X^{pub}, training data $\mathcal{D} = \{(x_i^{\text{priv}}, y_i, j_i, k_i)\}_{i \in \{1,\dots,D\}}$, optional weights $\{w_i\}$, number of steps T, number of inner iterations $\tau^{\max}$, clipping parameters Γ_H, Γ_g, noise standard deviation σ, learning rate η, initial parameters θ_v^0, θ_u^0.
Let $\bar{u}_i = \text{Clip}(u_{\theta_u}(x_i^{\text{priv}}), \Gamma_u)$, $\bar{y}_i = \text{Clip}(y_i, \Gamma_y)$
for $0 \le t \le T-1$ **do**
  for $1 \le j \le m$ **do**
    `/* Compute statistics for the quadratic approximation around` v^t `*/`
    $\hat{H}_j^t \leftarrow \sum_{i \in \Omega_j} w_i \text{Clip}(d^2\ell_i(v_j^t)\bar{u}_i\bar{u}_i^\top, \Gamma_H) + \sigma\Gamma_H\mathcal{N}^{d \times d}$
    $\hat{g}_j^t \leftarrow \sum_{i \in \Omega_j} w_i \text{Clip}(d\ell_i(v_j^t)\bar{u}_i, \Gamma_g) + \sigma\Gamma_g\mathcal{N}^d$.
  $\theta_v^{(0)} \leftarrow \theta_v^t$
  for $0 \le \tau \le \tau^{\max} - 1$ **do**
    `/* Optimize the quadratic approximation */`
    $\hat{G}^{(\tau)} \leftarrow \sum_{j=1}^m \mathcal{J}_j(\theta_v^{(\tau)})[\hat{H}_j^t(v_j(\theta_v^{(\tau)}) - v_j(\theta_v^t)) + \hat{g}_j^t]$
    $\theta_v^{(\tau+1)} \leftarrow \theta_v^{(\tau)} - \eta\hat{G}^{(\tau)}$
  $\theta_v^{t+1} \leftarrow \theta_v^{(\tau^{\max})}$
return θ_v^T

Algorithm 7 consists of multiple calls to SSP-c, each applied to the quadratic approximation around the current iterate v^t. Note that the main iterates are denoted by θ_v^t, while the inner loop uses $\theta_v^{(\tau)}$.

The privacy guarantee is an immediate extension of Theorems 1-2: to guarantee (ϵ, δ)-DP, it suffices to take $\sigma = \frac{\sqrt{8T\log 1/\delta}}{\epsilon}$ for example-level DP, and $\sigma = \frac{\bar{w}\sqrt{8T\log 1/\delta}}{\epsilon}$ for user-level DP. Notice that both scale with $\sqrt{T}$ (since sufficient statistics are recomputed at each step).

Empirical evaluation of Algorithm 7 is outside the scope of this paper, and is left as future work.

E ADDITIONAL EXPERIMENTS

In this section, we provide additional details regarding the exact experimental setup, along with additional results.

E.1 Experimental Setup

E.1.1 Data Sets and Public Features

We use two benchmarks based on MovieLens data[4]. The first is a regression task proposed by Lee et al. (2013), and the second is a classification task proposed by Liang et al. (2018).

The MovieLens data include movie features consisting of the release year and 19 movie genres. Curmei et al. (2023) expanded the movie features by extracting additional metadata from the Wikidata website; the additional features consist of a finer set of 310 genres, as well as 58,139 persons covering 55 roles. While the release year is a univalent feature, all other features (genres, persons, and roles) are multivalent features. Furthermore, roles and persons are paired, and one person may have multiple roles in a movie, and a role, such as "actor", may be associated to multiple persons. In our experiments, we simply treat each feature category as a bag of words (where we count repetitions, so if a feature such as "actor" is repeated, this will be treated as a weight of that feature value). One alternative is to use a feature cross between person and role, so each unique pair defines a feature value.

Statistics about the two data sets are summarized in Table 2.

data set	n (users)	m (items)	p (features)
ML10M	69,878	10,677	58,619
ML20M	136,677	2010	25,805

Table 2: Statistics of the MovieLens data sets

Following previous work (Jain et al., 2018; Chien et al., 2021; Krichene et al., 2023), ML10M uses the entire data, while in ML20M, training is restricted to the top 10% most frequent movies (but evaluation is always done on the entire set of movies). The histograms of the number of features active for each item are reported in Figure 7.

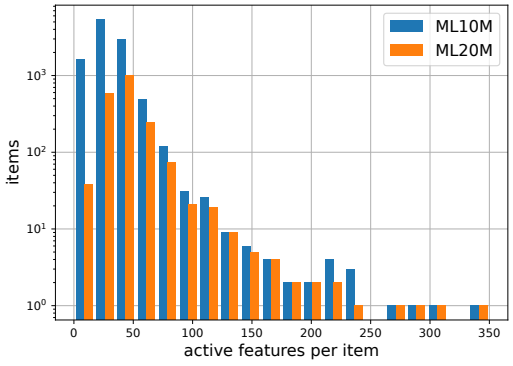


Figure 7: Density of features across movies

[4]The license information can be found on the GroupLens website at the following URLs: https://files.grouplens.org/datasets/movielens/ml-10m-README.html and https://files.grouplens.org/datasets/movielens/ml-20m-README.html

E.1.2 Model Architecture and Training

The public encoder consists of a two-layer model. The first layer is an embedding layer where each one of the 5 features (year, original genre, extended genre, person, role) is mapped to an embedding vector[5] in $\mathbb{R}^d$ (multivalent features are treated as a bag of words and their embeddings are averaged). These 5 embeddings are concatenated to form the hidden layer in $\mathbb{R}^{5d}$, and followed by a dense layer[6] with output dimension $\mathbb{R}^d$. In our experiments, the dimension d is treated as a hyper-parameter, and we consider dimensions up to $d = 64$, following the DP-ALS and DP-CMF baselines we compare to. We observe that higher quality can be achieved on ML20M with higher dimension (but to make a fair comparison to the baselines, we restrict to $d \leq 64$).

As for the user encoder, we follow the same setup as (Jain et al., 2018; Chien et al., 2021; Krichene et al., 2023; Curmei et al., 2023). The user encoder is an embedding lookup, i.e. each user k is mapped to a unique vector u_k (there are no other user features except the id). The model's output is therefore $u_{k_i} \cdot v_{\theta_v}(x^{\text{pub}}_{j_i})$ (k_i is the user id, j_i is the movie id).

Each model is trained on an NVIDIA Tesla P100 GPU. The model is trained by minimizing the regularized quadratic loss

$$\mathcal{L}(\theta_v) = \frac{1}{2}\sum_{i=1}^{D}(u_i \cdot v_{j_i}(\theta_v) - y_i)^2 + \lambda_u\|u_i\|_2^2 + \lambda_v\|v_{j_i}(\theta_v)\|_2^2.$$

We use the same `UserUpdate` for all methods (see footnote 2). During the `UserUpdate`, since we learn an embedding per user (with no shared parameters across users), the problem reduces to n decoupled least squares problems (one per user):

$$u_k^* = \text{argmin}_u \frac{1}{2}\sum_{i\in\Omega^k}(u \cdot v_{j_i}(\theta_v) - y_i)^2 + \lambda_u\|u_k\|_2^2,$$

for which we use the closed form solution $u_k^* = (\sum_{i\in\Omega^k} v_{j_i}v_{j_i}^\top + \lambda_u I)^{-1}(\sum_{i\in\Omega^k} y_i v_{j_i})$.

E.1.3 Evaluation

We follow the protocol of the original benchmarks Lee et al. (2013); Liang et al. (2018), which we summarize below. Each data set is split into training/validation/test.

In ML10M, the set of ratings is split at random, and utility is measured using RMSE, defined as $\text{RMSE} = \sqrt{\frac{\sum_{i\in\mathcal{D}_{\text{test}}}(u_i\cdot v_{j_i} - y_i)^2}{|\mathcal{D}_{\text{test}}|}}$, where $\mathcal{D}_{\text{test}}$ is the set of test ratings.

In ML20M, the validation and test splits consist of held-out users. Since validation/test users are not present in the training data, at evaluation time, one first needs to compute an embedding u_k for those users to be able to generate predictions. For this purpose the benchmark also splits the examples of each validation/test user k into an 80-20 split $\Omega^k_{\text{history}} \sqcup \Omega^k_{\text{target}}$, where the first part is used to compute the user embedding (by running `UserUpdate` on $\Omega^k_{\text{history}}$), and the second is used as the target labels. Utility is measured using Recall@20, defined as follows. A prediction score is computed for all movies except in $\Omega^k_{\text{history}}$ (to avoid penalizing a model that recommends items from the user's history). Let $\hat{\Omega}^k$ be the set of top-20 predictions, then the recall is defined as $\text{Recall@20} = \frac{1}{n}\sum_{k=1}^n \frac{|\Omega^k_{\text{target}}\cap\hat{\Omega}^k|}{\min(20, |\Omega^k_{\text{target}}|)}$.

Hyper-parameters are tuned on the validation data and metrics are reported on test data. Plots in Figures 1,2,4 report the mean metric value across ten runs and the standard deviation of the metric value is depicted as a shaded region (small standard deviations are barely visible in some instances).

E.1.4 Privacy Budget Allocation

All algorithms (including the DP-ALS and DP-CMF baselines) use the budget allocation method of Krichene et al. (2023). The method consists of computing example weights w_i (these are the inputs weights w_i in Algorithms 2-

[5]For simplicity, we use the same embedding dimension d for all features, and we also use the same dimension d as the output dimension of the second layer.

[6]Note that for this architecture, the cost c in our complexity analysis would be approximately $5d^2$, since the dense layer is a matrix in $\mathbb{R}^{5d\times d}$.

3) that are designed to control the sensitivity of each example. By allocating a higher weight (hence sensitivity) to tail items, it was found that overall utility can improve. The weights are computed following the method of (Krichene et al., 2023, equations (11)-(12)), which we summarize here. First, compute a DP estimate of the item counts, $\hat{n}_j$, then define the weights $w_i = \bar{w}\frac{\hat{n}_{j_i}^{-1/4}}{\sqrt{\sum_{i\in\Omega^k} n_{j_i}^{-1/4}}}$, where $\bar{w}$ is a parameter that control the privacy guarantee. By definition, we have that $\sum_{i\in\Omega^k} w_i^2 = \bar{w}^2$, so this parameter corresponds to the $\bar{w}^2$ in Theorems 2. Finally, privacy accounting is done via RDP composition (Mironov, 2017), where we also account for the estimation of item counts described above. The accounting is identical across all algorithms we evaluated.

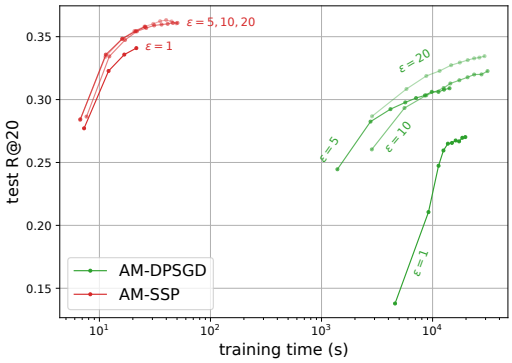


Figure 8: Comparison of SSP-i and SSP-c on ML20M

E.1.5 The DP-SGD baseline

In addition to SOTA baselines (DP-ALS and DP-CMF), we also compare to plain DP-SGD, where all model parameters are optimized jointly (without alternating minimization). This can be considered a weaker baseline, since it is not adapted to the problem's structure. The DP-SGD result is reported in Figure 1. It achieves a lower utility than all other methods, for example the RMSE at $\epsilon = 20$ is worse than the RMSE of competing methods at $\epsilon = 5$. Similarly for ML20M, the utility is much worse than other methods: we measured a Recall@20 of 0.206 at $\epsilon = 20$, while all other methods have better Recall@20 even at $\epsilon = 1$.

E.2 Computational Cost on ML20M

Figure 8 compares training a recall model with AM-DPSGD and AM-SSP on the ML20M dataset. The overall observation is similar to the one drawn from Figure 4, and in this case there is approximately three-orders of magnitude difference in training time between the two methods.

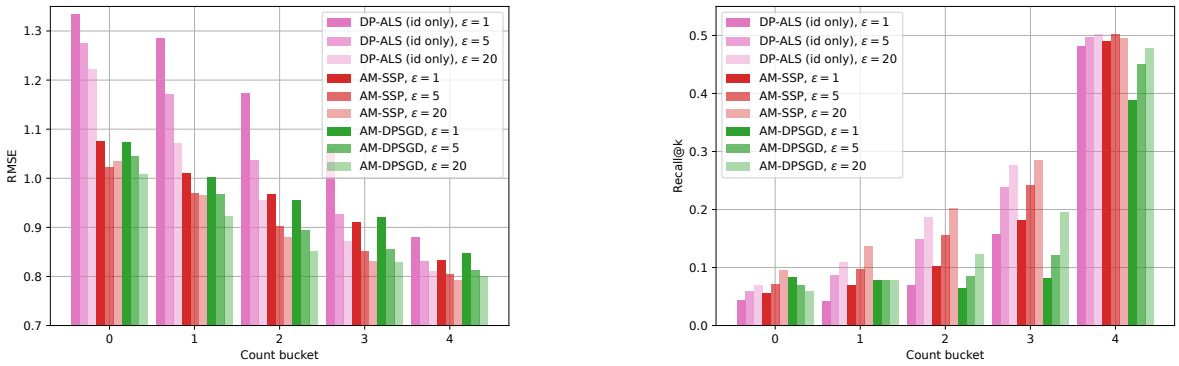
It's worth mentioning that we use an efficient implementation for DP-SGD (using the TensorFlow Privacy library[7] that implements fast per-example gradient clipping (Goodfellow, 2015)). We acknowledge that run times depend on implementation, though we believe a 2-3 orders of magnitude improvement to be significant, and the improvement should persist despite variations due to implementation.

E.3 Quality on the Long Tail

Prior work (Chien et al., 2021; Krichene et al., 2023; Curmei et al., 2023) reported that tail movies, i.e. those with fewer training examples, are more susceptible to the noise added to guarantee DP, and this manifests in larger quality losses on the long tail. To assess the impact of our methods on tail quality, we report in Figure 9 utility metrics sliced by movie count, for different values of ϵ.

On ML10M, we observe that the AM methods perform much better on the tail than the DP-ALS method (they also improve on the top slice, but the improvement is more significant on the tail). Recall that both AM methods train a feature-based encoder, while DP-ALS trains an id based encoder. This improvement on the tail perhaps indicates that using (public) features helps learn better representations of the long tail. Now comparing AM-

[7] www.tensorflow.org/responsible_ai/privacy



(a) ML10M (b) ML20M

Figure 9: Sliced quality metrics. The movies are sorted by increasing count, and partitioned in five buckets of equal size. So bucket 0 corresponds to the rarest 20% of the movies, while bucket 4 corresponds to the most frequent 20%.

DPSGD and AM-SSP, is appears that AM-SSP generally performs better on the head slices, and worse on the tail slices.

On ML20M, AM-DPSGD performs worse on all slices (consistent with Figure 2). Comparing AM-SSP to DP-ALS, we do observe more improvements on tail slices.

F AM-SSP IN FEDERATED LEARNING

Alternating minimization has also been studied in the Federated Learning (FL) literature (Singhal et al., 2021; Collins et al., 2021), where the goal is to perform distributed training while keeping each user's data on the user's device. The class of multi-encoder models is particularly well-suited to the federated setting, as the model reflects the natural separation between the user's data (to be protected) and the public features. In this section, we discuss implications of our algorithms on the FL setting. Figure 10 is an illustration of a federated implementation of the SSP-c algorithm.

The system consists of the following components:

- A server that holds the public feature matrix X^{pub}. The server is assumed untrusted (so any data or model parameters sent to the server need to be DP protected).
- n client devices (one per user). Client k holds the training data of user k, consisting of the private feature vector x_k^{priv}, together with the items and labels of the user, $\{(j_i, y_i)\}_{i \in \Omega^k}$.
- A secure distributor, which simply broadcasts model parameters to the n clients.
- A secure aggregator, responsible for computing and protecting the sufficient statistics. Techniques for secure aggregation have been studied for example by Bonawitz et al. (2017).

The training of the public encoder is described in Algorithm 8. Note that the steps are identical to Algorithm 5, except that we additionally specify on which component each operation is done.

Advantages of SSP in the Federated Setting Besides the advantages that we discussed in the general case (preserving gradient sparsity, noise that does not scale with the number of steps, etc.), SSP-c has additional advantages in the federated setting, compared to federated DP-SGD.

First, since we only need to add noise to the sufficient statistics once (see Remark 6), the clients only need to be involved in *one communication round*: they send data to the aggregator once, then the server takes several gradient steps. This is in contrast to federated DP-SGD, where a new communication round is initiated after each update to the encoder's parameters (the new parameters need to be broadcast to clients so gradients can be computed on clients and aggregated). Reducing the number of synchronization barriers (waiting for clients to become available) may have a significant impact on training time in practice.

Algorithm 8: Federated SSP-c

Distributor

Send: θ_u to clients.

Client k

Input: x_k^{priv}, $\mathcal{D}^k = \{(y_i, j_i)\}_{i \in \Omega^k}$

$u_k \leftarrow u_{\theta_u}(x_k^{\text{priv}})$.

Send: $u_k, \mathcal{D}^k$ to aggregator.

Aggregator

Input: Clipping parameters Γ_y, Γ_u, noise standard deviation σ, optional weights $\{w_i\}$

Let $\bar{u}_i = \text{Clip}(u_{\theta_u}(x_i^{\text{priv}}), \Gamma_u)$, $\bar{y}_i = \text{Clip}(y_i, \Gamma_y)$

for $1 \le j \le m$ **do**

$\hat{A}_j \leftarrow \sum_{i \in \Omega_j} w_i \bar{u}_i \bar{u}_i^\top + \sigma \Gamma_u^2 \mathcal{N}^{d \times d}$

$\hat{b}_j \leftarrow \sum_{i \in \Omega_j} w_i \bar{y}_i \bar{u}_i + \sigma \Gamma_y \Gamma_u \mathcal{N}^d$.

Send: $\{\hat{A}_j, \hat{b}_j\}_{1 \le j \le m}$ to server.

Server

Input: X^{pub}, number of steps T^I, learning rate η_t, statistics $\{\hat{A}_j, \hat{b}_j\}_{1 \le j \le m}$

for $0 \le t \le T^I - 1$ **do**

Uniformly sample $B \subset [m]$

$\hat{G}^t \leftarrow \sum_{j \in B} \frac{\partial v_{\theta_v^t}(x_j^{\text{pub}})}{\partial \theta_v} (\hat{A}_j v_{\theta_v^t}(x_j^{\text{pub}}) - \hat{b}_j)$

$\theta_v^{t+1} \leftarrow \theta_v^t - \eta_t \hat{G}^t$

return θ_v^T

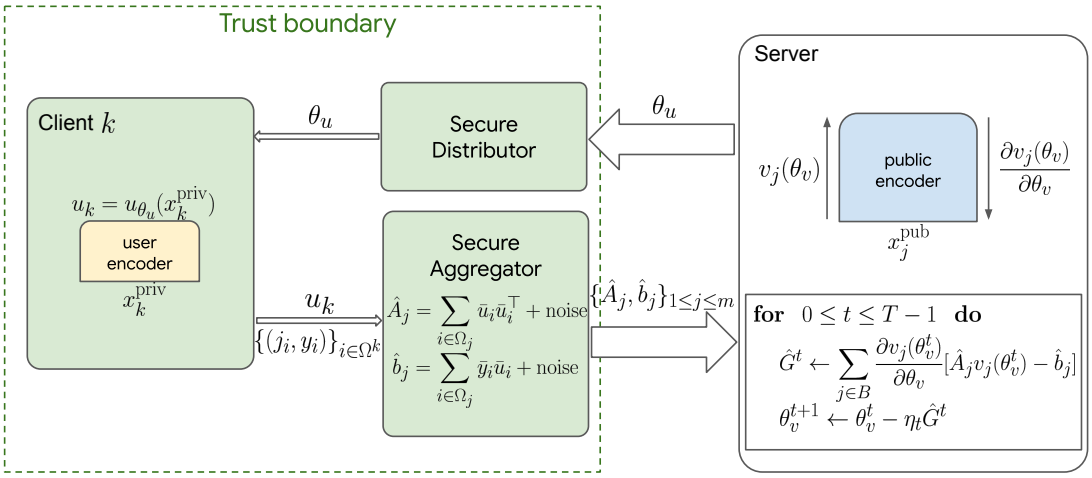


Figure 10: Illustration of Federated SSP-c

Second, the communication cost is drastically reduced. Each client only needs to send to the aggregator the output of the private encoder (u_k), and the labels $\{j_i, y_i\}_{i \in \Omega^k}$. Contrast this with federated DP-SGD, where client need to send model *gradients* (which can be significantly larger), and they do so at *each round*. In SSP-c, communication cost does not scale with the number of rounds nor with the public encoder's intermediate size such as feature vocab or hidden layers (as long as its output dimension is fixed).

Finally, gradient computation can happen on an *untrusted* server, since dependence on sensitive data is isolated into the protected sufficient statistics. This makes it possible to (i) reduce the computational burden on client devices (each client only needs to compute a single forward pass), and (ii) avoid having to send the public item features X^{pub} to the client devices. This further reduces the communication cost.