PAC Identifiability in Federated Personalization

Ben London
blondon@amazon.com
Amazon

Abstract

We study privacy in federated learning systems where the model is partitioned into global and local components, the latter of which are personalized for each participating client and never shared. This setting suggests a new type of privacy breach: that the server might learn a client's local model from updates to the global model. Using on-device recommendation as a motivating example, we show that this can in fact happen in various communication protocols, even when the client obscures its update messages with noise. These findings raise new questions and open problems about privacy in an emerging application of federated learning.

1 Introduction

Until recently, the de facto paradigm for industrial machine learning has been *centralization*. Voice queries, browsing behavior and other interactions are uploaded to "the Cloud," where they are mined in the service of user experiences, such as search and recommendation. By aggregating data across vast user bases, companies can train powerful intelligent systems. However, recent discoveries about how companies use and give access to our data, and the repercussions thereof for society, have forced a public conversation about data centralization and user privacy.

In response to increased privacy awareness, *federated learning* [22] has emerged as a promising solution for machine learning with data security. Distributed in nature, federated learning reduces security risks by keeping user data on-device, instead of uploading it to the Cloud. The only data that passes between devices and the Cloud is the model, which is periodically pushed (or pulled) to devices, where it is updated on local data, then returned to the Cloud. This not only protects users' privacy, it also enables enhanced personalization, such as health monitors for high-throughput sensor data, or recommender systems that adapt in real-time to user feedback—things that would be difficult or impossible to accomplish with centralized learning, due to data transfer costs and inconsistent connectivity. While most of the literature on federated learning has focused on data security, personalization is an equally compelling motivation.

In this work, we study a federated learning system designed for personalization—which we refer to as *federated personalization*. In these systems, the *server* (i.e., the Cloud) is associated with a *global model*, which is shared with all *clients* (e.g., mobile phone or smart speaker) and periodically pushed to them. Further, each client is associated with its own *local model*, which is not shared with the server or any other client. All clients are pre-programmed with a known predictor, which, given the global and local models, maps a context (e.g., client state or user behavior) to a label (e.g., rating) or action (e.g., recommendation). For instance, the predictor could be a function composition or some arithmetic operation (e.g., a dot product or affine combination). Like traditional federated learning, model parameters are updated on-device, using data collected from user interactions. However, only updates to the global model—typically, gradient information, which can be used for first-order optimization—are returned to the server, where they are aggregated.

One example application of federated personalization (which motivates our main results in Section 3) is on-device recommendation. Recommendation models are typically partitioned into per-

NeurIPS 2020 Workshop on Scalability, Privacy and Security in Federated Learning (SpicyFL).

sonalized user factors and shared item factors, which interact in a learned latent space. This partitioning naturally lends itself to federation; the global model could be the set of item factors, and the local model for a particular device could be the user's factors. User feedback (e.g., like/dislike), which is used for training, would never have to leave the device.[1]

We scrutinize this setting for potential privacy breaches, which we formalize using a new notion of privacy motivated by federated personalization: namely, we ask whether the server can learn a client's local model using only updates to the global model. As this idea is similar to *probably approximately correct* (PAC) learning [28], we call this property *PAC identifiability*. A dangerous consequence of PAC identifiability is that, if the local model is sufficiently accurate, an adversary can accurately reconstruct a user's preferences. By construction, we show that PAC identifiability is indeed possible in several communication protocols, using modest assumptions about the client and learning setup. In particular, we show that adding noise to update messages does not defend against our attack. This gives rise to new questions and open problems about privacy in federated learning.

Note. *In this abbreviated version, we defer all proofs to Appendix A.*

1.1 Related Work

The foundations of federated learning were laid out in McMahan et al.'s [22] landmark paper, in which they introduced the *federated averaging* (FedAvg) algorithm, which forms the basis of the protocols we consider. Shortly after its introduction, follow-up work extended the FedAvg algorithm to guarantee privacy [14, 23, 2, 15, 26] using the formalism of *differential privacy* [10, 8, 18]. Though differential privacy may be the prevailing notion of privacy in machine learning, we argue that the federated personalization setting motivates studying another form of privacy that accounts for the local model. The learning protocol we consider in Section 4.2 resembles differentially private federated learning, and our analysis reveals that even this setting is vulnerable to attacks.

The topic of personalization in federated learning has surfaced in multiple forms. Some work extended traditional recommender systems, such as matrix factorization [3, 7, 12]. The system we analyze in Section 3 most closely resembles these approaches. Other works have approached personalization by partitioning the model into global and local (personalized) representations [4, 6, 9, 20], which fits into our federated personalization framework. Personalization can also be cast as multitask learning [25] or meta-learning [19, 17, 11], with clients viewed as related learning tasks. From a theoretical perspective, some have analyzed when a hybrid of collaborative and independent training can achieve more accurate personalization than either of the extremes [9, 21].

Prior work has also studied data leakage from gradients [7, 24, 13]. What distinguishes our work is that we are specifically interested in whether one can infer a local model, not the data it was trained on. That said, Chai et al. [7] did describe a method to reconstruct user data that involves estimating the local model—though they do not give formal guarantees as we do. Finally, we note some tangential work that attempts to reverse engineer blackbox models using their predictions [27].

2 Privacy and the Local Model

Keeping user data on-device is a natural way to restrict access to it. However, this alone does not guarantee *privacy*, since clients communicate with the server in other ways. Specifically, when a client uploads an update to the global model, this message may leak critical information.

The standard formalism for privacy in the machine learning literature is *differential privacy* [10, 8, 18]. Informally, differential privacy guarantees that no single client's data is likely to have an outsized impact on the model, which effectively gives users plausible deniability that their data was used in training. To guarantee differential privacy, federated learning algorithms typically apply some form of corruption to the model updates, either at the server or the clients [14, 23, 2, 15, 26].

While differential privacy is considered by many to be the "right" definition of privacy for most data analysis, federated personalization motivates an alternative privacy property. The existence of a local model changes the dynamics of federated learning. Whereas participants in a federation typically collaborate to learn a single model for all clients, the local model learned by each client in a federated personalization system is tailored for an individual user. As we will show, this personalization means

[1] This does not, however, prevent the server from inferring *implicit* preferences, e.g., using playback events.

that the local model can reveal sensitive information about the user. To protect the user's privacy, a client should conceal both the data and the local model.

We therefore ask whether the local model can be *identified* (to be defined in the sequel) based on the messages returned to the server. We are primarily interested in what the server can infer, since the server has direct communication with clients.[2] In the following section, we formalize this notion of privacy using the PAC learning framework, asking, *can the server PAC learn the local model using its update messages?*

3 PAC Identifiability in Recommendation

Our analysis focuses on a simple federated personalization system for recommendation, which we feel is a good starting point. There is precedent for federated recommender systems in the literature [3, 7], which indicates that this setting is not that far-fetched.

Let $\mathcal{I} \triangleq \{1, \dots, I\}$ denote a large (yet finite), fixed catalog of I items (e.g., songs or videos). Let $\mathcal{C}$ denote the set of clients participating in the federation. The system is designed to infer a preference for each user (client), $c \in \mathcal{C}$, with respect to each item, $i \in \mathcal{I}$. For simplicity, we assume that preferences are binary; you either like something or you do not. Accordingly, let $y_{c,i} \in \{\pm 1\}$ denote a binary preference label for client c with respect to item i. Further simplifying the problem, we ignore contextual information and assume that preferences can be modeled by a linear classifier, $\operatorname{sgn}(\mathbf{u}_c \cdot \mathbf{v}_i)$, where $\{\mathbf{v}_i \in \mathbb{R}^k\}_{i \in \mathcal{I}}$ is a set of latent factors representing the items (i.e., the global model) and $\mathbf{u}_c \in \mathbb{R}^k$ is a vector representing the user (i.e., the client's local model).

We assume that each client has collected a dataset of labels, $\{y_{c,i}\}_{i \in \mathcal{S}_c}$, for a subset of N items, $\mathcal{S}_c \subseteq \mathcal{I}$, based on user interactions. They could, for instance, come from like/dislike feedback. Ideally, the labels will be collected in an unbiased manner, though we do not assume as much unless explicitly stated. The dataset may change over time, but we can safely assume that it is static during periods when communication occurs, which should be while the client is inactive.

We also make the following key assumptions about the data and loss function used for training.

Assumption 1. For any client, $c \in \mathcal{C}$, let $p_c \triangleq \frac{1}{N} \sum_{i \in \mathcal{S}_c} \mathbb{1}\{y_{c,i} > 0\}$ denote their *preference rate* (i.e., the fraction of items that they like), and assume that $p_c \in (0, 1/2)$.

Assumption 1 essentially states that users are interested in at most half of the items in their dataset. We believe this assumption is natural, given that most content or product catalogs have millions of items, many of which are irrelevant to any particular user; if $\mathcal{S}_c$ is a representative sample of the user's overall preferences, it will be heavily skewed toward negative labels. There is actually not much generality lost by this assumption, because our analysis would also work with $p_c \in (1/2, 1)$. The key is that $p_c \neq 1/2$, and that we know which direction the user leans.

Assumption 2. Assume that the loss function, $L : \mathbb{R}^k \times \mathbb{R}^k \times \{\pm 1\} \to \mathbb{R}_+$, takes the form $L(\mathbf{u}_c, \mathbf{v}_i, y_{c,i}) = \ell(y_{c,i}\, \mathbf{u}_c \cdot \mathbf{v}_i)$, where $\ell : \mathbb{R} \to \mathbb{R}_+$ is an auxiliary function of the *margin*, $y_{c,i}\, \mathbf{u}_c \cdot \mathbf{v}_i$. Further, assume that the derivative of ℓ evaluated at 0 is nonzero; i.e., $\ell'(0) \neq 0$.

One example of a loss function that satisfies Assumption 2 is the *negative margin loss*, $\ell(z) = -z$, which penalizes incorrect (or rewards correct) predictions proportional to how much they disagree (or agree) with the label. Some other examples are the *hinge loss*, $\ell(z) = \max\{0, 1 - z\}$, and the *log loss*, $\ell(z) = \ln(1 + e^{-z})$, which is equivalent to the *binary cross-entropy*.

We now formally define PAC identifiability in this context.

Definition 1. A client, $c \in \mathcal{C}$, using a given protocol (which may be stochastic), is *PAC identifiable* if, for any $\epsilon \in (0, 1)$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$ over $T = \operatorname{poly}(\epsilon^{-1}, \delta^{-1})$ interactions with the server, the server can output an estimate, $\hat{\mathbf{u}}_c$, of the local model (after interaction), $\mathbf{u}_c$, such that $\frac{1}{I} \sum_{i \in \mathcal{I}} \mathbb{1}\{\operatorname{sgn}(\hat{\mathbf{u}}_c \cdot \mathbf{v}_i) \neq \operatorname{sgn}(\mathbf{u}_c \cdot \mathbf{v}_i)\} \leq \epsilon$.

To see the threat posed by PAC identifiability, consider the following repercussion.

Proposition 1. *If a client, $c \in \mathcal{C}$, is PAC identifiable, then for any $\epsilon \in (0, 1)$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$ over $T = \operatorname{poly}(\epsilon^{-1}, \delta^{-1})$ interactions with c, the server can output $\hat{\mathbf{u}}_c$*

[2] One could also consider a third-party adversary (e.g., another client), but this is out of scope for our work.

such that, if $\mathbf{u}_c$ *(after interaction) has error rate* $\frac{1}{I}\sum_{i\in\mathcal{I}} \mathbb{1}\{\operatorname{sgn}(\mathbf{u}_c\cdot\mathbf{v}_i)\neq y_{c,i}\}=\eta$*, then* $\hat{\mathbf{u}}_c$ *has error rate* $\frac{1}{I}\sum_{i\in\mathcal{I}} \mathbb{1}\{\operatorname{sgn}(\hat{\mathbf{u}}_c\cdot\mathbf{v}_i)\neq y_{c,i}\}\leq\epsilon+\eta$*.*

Thus, if a client is PAC identifiable, and the local model is reasonably accurate, then the server could infer all of the user's preferences with reasonable accuracy. We could alternately redefine the error quantities with respect to the client's labeled items, $\mathcal{S}_c$. Then, if the local model minimizes empirical risk, PAC identifiability implies recovery of the client's data. That said, in certain circumstances, gradients can leak client data regardless of the quality of the local model [7, 24, 13], so PAC identifiability should be viewed as a sufficient, but not necessary, condition for data leakage.

4 Protocols and Attacks

We now discuss some specific communication protocols and corresponding attacks.

4.1 A Simple Protocol and Attack

To warm up, we consider the protocol outlined in Protocol 1. The server initiates updates by selecting a (random) subset of the available clients to execute the update (in parallel) on their respective local data. Each selected client receives the current global model and updates its internal state; then, performs $R\geq 0$ rounds of SGD,[3] with random batches of size $B\geq 1$, sampled without replacement from its labeled items, $\mathcal{S}_c$.[4] When finished, the client returns only the change in the global model to the server. The server then aggregates the updates from all selected clients. Normally, the server would iteratively execute this protocol with a random subset of available clients. However, a malicious server may repeatedly target a single client or modify the server-side protocol.

Protocol 1 Federated Recommendation Communication Protocol

```
procedure SERVER
    Select (random) subset of available clients, $\mathcal{C}'\subseteq\mathcal{C}$
    for $c\in\mathcal{C}'$ in parallel do
        $\{\Delta\mathbf{v}_i^{(c)}\}_{i\in\mathcal{I}}\leftarrow$ CLIENTUPDATE($c,\{\mathbf{v}_i\}_{i\in\mathcal{I}},R$)
    $\forall i\in\mathcal{I},\ \mathbf{v}_i\leftarrow\mathbf{v}_i+\sum_{c\in\mathcal{C}'}\Delta\mathbf{v}_i^{(c)}$
procedure CLIENTUPDATE($c,\{\mathbf{v}_i^{(0)}\}_{i\in\mathcal{I}},R$)
    $\forall i\in\mathcal{I},\ \mathbf{v}_i\leftarrow\mathbf{v}_i^{(0)}$
    for $r=1,\ldots,R$ do
        Sample $\mathcal{B}\subseteq\mathcal{S}_c:|\mathcal{B}|=B$, uniformly at random, without replacement
        $\mathbf{g}_c\leftarrow\frac{1}{B}\sum_{i\in\mathcal{B}}\nabla_{\mathbf{u}_c}L(\mathbf{u}_c,\mathbf{v}_i,y_{c,i})$
        $\forall i\in\mathcal{B},\ \mathbf{v}_i\leftarrow\mathbf{v}_i-\alpha\nabla_{\mathbf{v}_i}L(\mathbf{u}_c,\mathbf{v}_i,y_{c,i})$
        $\mathbf{u}_c\leftarrow\mathbf{u}_c-\alpha\mathbf{g}_c$
    return $\{\mathbf{v}_i-\mathbf{v}_i^{(0)}\}_{i\in\mathcal{I}}$
```

We will show, by construction, that clients that execute the above client-side protocol are PAC identifiable. To give some intuition of how the construction works, we use the chain rule to derive the parameter gradients:

$$\nabla_{\mathbf{u}_c}L(\mathbf{u}_c,\mathbf{v}_i,y_{c,i})=\ell'(y_{c,i}\,\mathbf{u}_c\cdot\mathbf{v}_i)\,y_{c,i}\,\mathbf{v}_i \tag{1}$$

$$\text{and}\quad \nabla_{\mathbf{v}_i}L(\mathbf{u}_c,\mathbf{v}_i,y_{c,i})=\ell'(y_{c,i}\,\mathbf{u}_c\cdot\mathbf{v}_i)\,y_{c,i}\,\mathbf{u}_c, \tag{2}$$

where $\ell'(z)$ denotes the derivative of the auxiliary function, ℓ. From this, we make two observations. First, if all item vectors are zero, $\mathbf{v}_i=\mathbf{0}$, then the gradient with respect to $\mathbf{u}_c$ is also zero; as such, after only one update, the local model will not change. Second, we have that the gradient with respect to $\mathbf{v}_i$ is just $\mathbf{u}_c$ scaled by some unknown value, $s_i\triangleq\ell'(y_{c,i}\,\mathbf{u}_c\cdot\mathbf{v}_i)\,y_{c,i}$. Given only the sign of s_i, if $s_i\neq 0$, then

$$\operatorname{sgn}(\operatorname{sgn}(s_i)\,\nabla_{\mathbf{v}_i}L(\mathbf{u}_c,\mathbf{v}_i,y_{c,i})\cdot\mathbf{v}_i)=\operatorname{sgn}(\,|s_i|\,\mathbf{u}_c\cdot\mathbf{v}_i)=\operatorname{sgn}(\mathbf{u}_c\cdot\mathbf{v}_i).$$

[3] If $R=0$, then the SGD loop does not execute; this is equivalent to pushing the global model to the client without forcing an update.

[4] We assume that available clients have collected at least as much data as the batch size.

This identity enables a much stronger violation than just PAC identifiability; it enables perfect reconstruction ($\epsilon = 0$). Of course, the sign of s_i is *not* known, because the sign of $y_{c,i}$ is unknown, but we can leverage Assumptions 1 and 2 to infer the average value of s_i over all of the examples used to compute the updates.

Theorem 1. *Under Assumptions 1 and 2, we have that any client, $c \in \mathcal{C}$, that faithfully executes Protocol 1 is PAC identifiable, for $\epsilon = 0$ and any $\delta \in (0, 1)$, using*

$$T \geq \frac{2 \ln(1/\delta)}{B(1 - 2p_c)^2} \quad \textit{interactions.}$$

The proof uses the modified server-side protocol (i.e., attack) given in Protocol 2. For T rounds, the server executes a single ($R = 1$) update on the client using a fake global model in which all item vectors are set to zero. When finished, the server aggregates the returned updates and divides by $(\alpha \ell'(0) T B)$ to obtain $\hat{\mathbf{u}}_c$, an estimate of the client's local model. Finally, the server restores the actual global model on the client by executing the update command with $R = 0$. This step is only needed if the attacker wishes the client to function as before the attack.

Protocol 2 Server-side Attack

```
procedure SERVER
    for $t = 1, \ldots, T$ do
        $\{\Delta \mathbf{v}_i^{(t)}\}_{i \in \mathcal{I}} \leftarrow$ CLIENTUPDATE$(c, \{\mathbf{0}\}_{i \in \mathcal{I}}, 1)$
    $\hat{\mathbf{u}}_c \leftarrow (\alpha\, \ell'(0)\, T B)^{-1} \sum_{t=1}^{T} \sum_{i \in \mathcal{I}} \Delta \mathbf{v}_i^{(t)}$
    CLIENTUPDATE$(c, \{\mathbf{v}_i\}_{i \in \mathcal{I}}, 0)$        ▷ (Optional) Restore global model on client
```

The attack works because $\hat{\mathbf{u}}_c$ turns out to be the local model, scaled by the average of all the labels used in the updates. By our assumption that the label distribution skews negative, the average of labels should, with high probability, be negative. Thus, having determined the sign of the multiplier, we can correct for it and obtain a correctly signed copy of the local model.

One interesting takeaway is that the number of required interactions *decreases* as the positive preference rate, p_c, approaches zero. If p_c is representative of the user's overall preference rate (e.g., if $\mathcal{S}_c$ is an unbiased sample), then users with discriminating tastes are easier to identify. This is dangerous because knowing their distinctive preferences may reveal very specific characteristics about them.

It is important to emphasize that Protocol 2 is only a theoretical construction designed to show the existence of an attack. It is not intended to "fly under the radar." Indeed, it is straightforward for a client to detect such an attack. Regardless, if the client faithfully performs the communication protocol, then the server can identify the local model.

4.2 Noisy Messages

Federated learning is typically paired with a differential privacy mechanism. The distributed nature of federated learning is particularly well suited for *local* differential privacy, a form of differential privacy that does not require a trusted third-party aggregator [18]. In local differential privacy, a privacy mechanism is applied at each client. Typically, this amounts to adding noise to whatever data is returned to the aggregator (e.g., statistics of the client's local data). In the context of federated learning, each client adds noise to the model updates it returns to the server [2].[5]

We therefore investigate whether adding noise to the update messages affects PAC identifiability. We consider a modified client-side protocol in which the client returns a corrupted copy of the global model update:

$$\tilde{\Delta} \mathbf{v}_i \triangleq \begin{cases} \mathbf{v}_i - \mathbf{v}_i^{(0)} + \boldsymbol{\varepsilon}_i, \text{ with } \boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}) & \text{if } \mathbf{v}_i \neq \mathbf{v}_i^{(0)}, \\ \mathbf{0} & \text{if } \mathbf{v}_i = \mathbf{v}_i^{(0)}, \end{cases} \tag{3}$$

where $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ is a k-dimensional, isotropic Gaussian, with variance σ^2. Noise is only added for items whose parameters have changed; this is designed to reduce unnecessary data transfer. With

[5] A client should also use *gradient clipping* to reduce the *sensitivity* of the model updates [1], but we will ignore this detail for simplicity of exposition. One could always assume that gradients are naturally bounded.

proper scaling of σ (and gradient clipping), this mechanism is differentially private. However, does it prevent the local model from being PAC identified?

In the following, we show that the system is still PAC identifiable using the same attack as before. The key to the attack's success is that, averaged over multiple updates, the noise concentrates around its mean, which is zero. Thus, since the noise is additive, the original update is eventually revealed.

Theorem 2. *For $c \in \mathcal{C}$ and $\epsilon \in [0,1]$, let $\tau_c(\epsilon)$ denote the largest $\tau \geq 0$ such that $\frac{1}{I}\sum_{i\in\mathcal{I}} \mathbb{1}\{|\mathbf{u}_c \cdot \mathbf{v}_i| \leq \tau\} \leq \epsilon$. Let $M \triangleq \max_{i\in\mathcal{I}} \|\mathbf{v}_i\|$ denote the maximum magnitude of any item vector. Under Assumptions 1 and 2, we have that any client, $c \in \mathcal{C}$, that faithfully executes Protocol 1 using the modified global model update message given in Equation 3 is PAC identifiable, for any $\epsilon \in [0,1]$, $\delta \in (0,1)$ and $\beta \in (0, 1-2p_c)$, using*

$$T \geq \max\left\{\frac{2\ln(2/\delta)}{B(1-2p_c-\beta)^2}, \frac{2\sigma^2 M^2 \ln(4I/\delta)}{B\left(\alpha\,\beta\,\ell'(0)\,\tau_c(\epsilon)\right)^2}\right\} \quad \textit{interactions.}$$

We note that, though we assumed Gaussian noise in Equation 3, Theorem 2 holds for any sub-Gaussian noise distribution.

4.3 SGD Without Replacement

The client-side protocol in Protocol 1 uses stochastic gradients, sampled with replacement between batches, and runs for a given number of updates. We can also consider a variant that instead iterates over the entire dataset for a given number of epochs (with shuffling before each epoch, if desired). In this scenario, the attack in Protocol 2 is still effective—even more so, because the proportion of positive labels used in an update is always the same. Provided we know which way the label imbalance leans, we can correct the sign of the estimated user vector with probability one.

The modified protocol is given in Protocol 3 (see Appendix A.4). On the client side, the number of updates, R, is determined by the number of epochs, E, the size of the data, N, and the batch size, B (which we assume evenly divides N). Since identifiability would be trivial without adding noise, we use the noisy update messages from Equation 3. The only change on the server side is that it executes CLIENTUPDATE using E instead of R.

Theorem 3. *For $c \in \mathcal{C}$ and $\epsilon \in [0,1]$, let $\tau_c(\epsilon)$ denote the largest $\tau \geq 0$ such that $\frac{1}{I}\sum_{i\in\mathcal{I}} \mathbb{1}\{|\mathbf{u}_c \cdot \mathbf{v}_i| \leq \tau\} \leq \epsilon$. Let $M \triangleq \max_{i\in\mathcal{I}} \|\mathbf{v}_i\|$ denote the maximum magnitude of any item vector. Under Assumptions 1 and 2, we have that any client, $c \in \mathcal{C}$, that faithfully executes Protocol 3 is PAC identifiable, for any $\epsilon \in [0,1]$ and $\delta \in (0,1)$, using*

$$T \geq \frac{2\sigma^2 M^2 \ln(2I/\delta)}{N\left(\alpha\,(1-2p_c)\,\ell'(0)\,\tau_c(\epsilon)\right)^2} \quad \textit{interactions.} \tag{4}$$

5 Discussion and Future Work

Though federated learning is a promising tool for personalization, there are still many privacy issues to address. Differential privacy may be the de facto definition of privacy in machine learning—for good reason—but we argue that PAC identifiability is equally compelling in the setting of federated personalization. Hopefully, this paper will inspire more research in this direction.

Our PAC identifiability results demonstrate that certain protocols are vulnerable to simple attacks. One could argue that our assumptions or protocols are overly simplistic, or that our attacks are easily detectable. We do not dispute that more work is needed to relax our assumptions and accommodate more realistic protocols; and we acknowledge that there may already be techniques (e.g., *secure multiparty computation* [5]) that protect against PAC identifiability—certainly against our current crude attacks. However, we conjecture that even more sophisticated techniques are vulnerable to some attacks, and that it is just a matter of identifying the right attacks.

Finally, we stress that our goal is *not* to provide a blueprint for adversaries to violate users' privacy. To the contrary, our goal is to identify and raise awareness of these issues, so that they can be addressed in future work.

References

[1] M. Abadi, A. Chu, I. Goodfellow, B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[2] N. Agarwal, A. Suresh, F. Yu, S. Kumar, and B. McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Neural Information Processing Systems*, 2018.

[3] M. Ammad-ud-din, E. Ivannikova, S. Khan, W. Oyomno, Q. Fu, K. Tan, and A. Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system. *CoRR*, abs/1901.09888, 2019.

[4] M. Arivazhagan, V. Aggarwal, A. Singh, and S. Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[6] D. Bui, K. Malik, J. Goetz, H. Liu, S. Moon, A. Kumar, and K. G. Shin. Federated user representation learning. *CoRR*, abs/1909.12535, 2019.

[7] D. Chai, L. Wang, K. Chen, and Q. Yang. Secure federated matrix factorization. *CoRR*, abs/1906.05108, 2019.

[8] K. Chaudhuri, C. Monteleoni, and A. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011.

[9] Y. Deng, M. Kamani, and M. Mahdavi. Adaptive personalized federated learning. *CoRR*, abs/2003.13461, 2020.

[10] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9:211–407, 2014.

[11] A. Fallah, A. Mokhtari, and A. Ozdaglar. Personalized federated learning: A meta-learning approach. *CoRR*, abs/2002.07948, 2020.

[12] A. Flanagan, W. Oyomno, A. Grigorievskiy, K. Tan, S. Khan, and M. Ammad-ud-din. Federated multi-view matrix factorization for personalized recommendations. *CoRR*, abs/2004.04256, 2020.

[13] J. Geiping, H. Bauermeister, H. Dröge, and Michael Moeller. Inverting gradients: How easy is it to break privacy in federated learning? *CoRR*, abs/2003.14053, 2020.

[14] R. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *CoRR*, abs/1712.07557, 2017.

[15] B. Ghazi, R. Pagh, and A. Velingker. Scalable and differentially private distributed aggregation in the shuffled model. *CoRR*, abs/1906.08320, 2019.

[16] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[17] Y. Jiang, J. Konecný, K. Rush, and S. Kannan. Improving federated learning personalization via model agnostic meta learning. *CoRR*, abs/1909.12488, 2019.

[18] S. Kasiviswanathan, H. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? *SIAM Journal of Computing*, 40(3):793–826, 2011.

[19] M. Khodak, M.-F. Balcan, and A. Talwalkar. Adaptive gradient-based meta-learning methods. In *Neural Information Processing Systems*, 2019.

[20] P. Liang, T. Liu, Z. Liu, R. Salakhutdinov, and L.-P. Morency. Think locally, act globally: Federated learning with local and global representations. *CoRR*, abs/2001.01523, 2020.

[21] Y. Mansour, M. Mohri, J. Ro, and A. Suresh. Three approaches for personalization with applications to federated learning. *CoRR*, abs/2002.10619, 2020.

[22] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, 2017.

[23] H. McMahan, D. Ramage, Kunal Talwar, and L. Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Represetations*, 2018.

[24] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, 2019.

[25] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar. Federated multi-task learning. In *Neural Information Processing Systems*, 2017.

[26] O. Thakkar, G. Andrew, and B. McMahan. Differentially private learning with adaptive clipping. *CoRR*, abs/1905.03871, 2019.

[27] F. Tramèr, F. Zhang, A. Juels, M. Reiter, and T. Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*, 2016.

[28] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

A Deferred Proofs

This appendix contains proofs deferred from the main manuscript.

A.1 Proof of Proposition 1

By the triangle inequality,

$$\mathbb{1}\{\mathrm{sgn}(\hat{\mathbf{u}}_c \cdot \mathbf{v}_i) \neq y_{c,i}\} \leq \mathbb{1}\{\mathrm{sgn}(\hat{\mathbf{u}}_c \cdot \mathbf{v}_i) \neq \mathrm{sgn}(\mathbf{u}_c \cdot \mathbf{v}_i)\} + \mathbb{1}\{\mathrm{sgn}(\mathbf{u}_c \cdot \mathbf{v}_i) \neq y_{c,i}\}.$$

Averaging over $i \in \mathcal{I}$ completes the proof.

A.2 Proof of Theorem 1

Recall from Equation 1 that, when all $\mathbf{v}_i$ are zero, the gradient with respect to $\mathbf{u}_c$ is zero. Hence, after $R = 1$ updates, $\mathbf{u}_c$ will not have moved. Further, for every $t \in \{1, \ldots, T\}$ and $i \in \mathcal{B}_t$ (where $\mathcal{B}_t$ denotes the t^{th} batch), via Equation 2, we have that

$$\Delta \mathbf{v}_i^{(t)} = -\alpha \nabla_{\mathbf{v}_i} L(\mathbf{u}_c, \mathbf{v}_i, y_{c,i}) = -\alpha\, \ell'(y_{c,i}\, \mathbf{u}_c \cdot \mathbf{v}_i)\, y_{c,i}\, \mathbf{u}_c = -\alpha\, \ell'(0)\, y_{c,i}\, \mathbf{u}_c,$$

where we have used the fact that $\mathbf{u}_c \cdot \mathbf{v}_i = 0$ if $\mathbf{v}_i = \mathbf{0}$. We also have that $\Delta \mathbf{v}_i^{(t)} = \mathbf{0}$ for every $i \notin \mathcal{B}_t$. Therefore,

$$\begin{aligned}
\hat{\mathbf{u}}_c &= \frac{1}{\alpha\, \ell'(0)\, TB} \sum_{t=1}^{T} \sum_{i \in \mathcal{I}} \Delta \mathbf{v}_i^{(t)} \\
&= \frac{1}{\alpha\, \ell'(0)\, TB} \sum_{t=1}^{T} \sum_{i \in \mathcal{B}_t} -\alpha\, \ell'(0)\, y_{c,i}\, \mathbf{u}_c \\
&= \mathbf{u}_c \left(-\frac{1}{TB} \sum_{t=1}^{T} \sum_{i \in \mathcal{B}_t} y_{c,i} \right).
\end{aligned}$$

If a strict majority of preferences in $(\mathcal{B}_1, \ldots, \mathcal{B}_T)$ are negative, then

$$-\frac{1}{TB} \sum_{t=1}^{T} \sum_{i \in \mathcal{B}_t} y_{c,i} > 0,$$

and $\hat{\mathbf{u}}_c$ will be a positively-scaled copy of $\mathbf{u}_c$, thus enabling perfect reconstruction of the local predictions.

Therefore, all that remains now is to show that, with high probability over draws of $(\mathcal{B}_1, \ldots, \mathcal{B}_T)$, a majority of sampled preferences will be negative. For this, we will use Hoeffding's inequality [16]. Recall that each batch, $\mathcal{B}_t$, is generated by sampling uniformly, without replacement, from $\mathcal{S}_c$ (but items are replaced for each new batch). While Hoeffding's inequality is typically stated for independent draws from a distribution, it also holds for sampling with or without replacement from a population (as a consequence of [16, Theorem 4]).

Lemma 1. *Let $\mathcal{X} \triangleq \{x_1, \ldots, x_N\}$ denote a finite population of N values such that $\forall i,\ 0 \leq x_i \leq 1$. Let $(X_1, \ldots, X_n)$ denote a random sample drawn with or without replacement from $\mathcal{X}$. Let $\mu \triangleq \frac{1}{N}\sum_{i=1}^{N} x_i$ denote the population mean, and $\hat{\mu} \triangleq \frac{1}{n}\sum_{i=1}^{n} X_i$ denote the sample mean. Then, for $\tau > 0$,*

$$\Pr\{\hat{\mu} \geq \mu + \tau\} \leq \exp(-2n\tau^2).$$

Thus, for $p_c = \frac{1}{N}\sum_{i \in \mathcal{S}_c} \mathbb{1}\{y_{c,i} > 0\} < 1/2$ (by Assumption 1) and

$$\hat{p}_c \triangleq \frac{1}{TB} \sum_{t=1}^{T} \sum_{i \in \mathcal{B}_t} \mathbb{1}\{y_{c,i} > 0\}, \tag{5}$$

we have that

$$\begin{aligned}
\Pr\left\{-\frac{1}{TB}\sum_{t=1}^{T}\sum_{i\in\mathcal{B}_t} y_{c,i} > 0\right\} &= \Pr\left\{\hat{p}_c < \frac{1}{2}\right\} \\
&= 1 - \Pr\left\{\hat{p}_c \geq \frac{1}{2}\right\} \\
&= 1 - \Pr\left\{\hat{p}_c \geq p_c + \left(\frac{1}{2} - p_c\right)\right\} \\
&\geq 1 - \exp\left(-2TB\left(\frac{1}{2} - p_c\right)^2\right).
\end{aligned}$$

Taking $\delta = \exp(-2TB(1/2 - p_c)^2)$ and solving for T completes the proof.

A.3 Proof of Theorem 2

We can again use the attack in Protocol 2, albeit with a different value of T.

Consider an arbitrary vector, $\mathbf{v} \in \mathbb{R}^k$. Because the noise model is additive, via linearity, we have that

$$\begin{aligned}
\hat{\mathbf{u}}_c \cdot \mathbf{v} &= \frac{1}{\alpha\,\ell'(0)\,TB}\sum_{t=1}^{T}\sum_{i\in\mathcal{I}} \tilde{\Delta}\mathbf{v}_i^{(t)} \cdot \mathbf{v} \\
&= \frac{1}{\alpha\,\ell'(0)\,TB}\sum_{t=1}^{T}\sum_{i\in\mathcal{B}_t} \left(\Delta\mathbf{v}_i^{(t)} + \boldsymbol{\varepsilon}_i^{(t)}\right) \cdot \mathbf{v} \\
&= \underbrace{\left(-\frac{1}{TB}\sum_{t=1}^{T}\sum_{i\in\mathcal{B}_t} y_{c,i}\right)\mathbf{u}_c \cdot \mathbf{v}}_{\text{(a)}} \;+\; \underbrace{\frac{1}{\alpha\,\ell'(0)}\left(\frac{1}{TB}\sum_{t=1}^{T}\sum_{i\in\mathcal{B}_t} \boldsymbol{\varepsilon}_i^{(t)} \cdot \mathbf{v}\right)}_{\text{(b)}}.
\end{aligned}$$

If $|\mathbf{u}_c \cdot \mathbf{v}| > \tau_c(\epsilon)$, then conditions

$$-\frac{1}{TB}\sum_{t=1}^{T}\sum_{i\in\mathcal{B}_t} y_{c,i} > \beta \tag{6}$$

$$\text{and} \quad \left|\frac{1}{TB}\sum_{t=1}^{T}\sum_{i\in\mathcal{B}_t} \boldsymbol{\varepsilon}_i^{(t)} \cdot \mathbf{v}\right| < |\alpha\,\beta\,\ell'(0)\,\tau_c(\epsilon)| \tag{7}$$

ensure that (a) has greater magnitude than (b), implying $\operatorname{sgn}(\hat{\mathbf{u}}_c \cdot \mathbf{v}) = \operatorname{sgn}(\mathbf{u}_c \cdot \mathbf{v})$. Let E_1 denote the event that Equation 6 holds. Letting $E_2^{(i)}$ denote the event that Equation 7 holds for a specific $\mathbf{v}_i$, we define $E_2 \triangleq \bigwedge_{i\in\mathcal{I}} E_2^{(i)}$ as the event that it holds for all $i \in \mathcal{I}$. Since at most an ϵ fraction of items have $|\mathbf{u}_c \cdot \mathbf{v}_i| \leq \tau_c(\epsilon)$, the error rate will be at most ϵ with probability $\Pr\{E_1 \wedge E_2\}$.

We therefore want to lower-bound $\Pr\{E_1 \wedge E_2\}$. Using the fact that

$$\Pr\{E_1 \wedge E_2\} = 1 - \Pr\{\neg E_1 \vee \neg E_2\} \geq 1 - \Pr\{\neg E_1\} - \Pr\{\neg E_2\},$$

we will upper bound $\Pr\{\neg E_1\}$ and $\Pr\{\neg E_2\}$ separately. The first probability can be bounded using Hoeffding's inequality (Lemma 1). Using our previous definition of $\hat{p}_c$ (Equation 5), we have

$$\begin{aligned}
\Pr\{\neg E_1\} &= \Pr\left\{ -\frac{1}{TB} \sum_{t=1}^{T} \sum_{i \in \mathcal{B}_t} y_{c,i} \leq \beta \right\} \\
&= \Pr\left\{ \hat{p}_c \geq \frac{1-\beta}{2} \right\} \\
&= \Pr\left\{ \hat{p}_c \geq p_c + \left(\frac{1-\beta}{2} - p_c \right) \right\} \\
&\leq \exp\left(-2TB \left(\frac{1-\beta}{2} - p_c \right)^2 \right). \qquad (8)
\end{aligned}$$

To bound $\Pr\{\neg E_2\}$, first observe that each $\boldsymbol{\varepsilon}_i^{(t)} \cdot \mathbf{v}$ is an independent, normally distributed random variable with mean 0 and variance $\sigma^2 \|\mathbf{v}\|^2 \leq \sigma^2 M^2$. Using elementary tail bounds for averages of normally distributed random variables,

$$\Pr\left\{ \left| \frac{1}{TB} \sum_{t=1}^{T} \sum_{i \in \mathcal{B}_t} \boldsymbol{\varepsilon}_i^{(t)} \cdot \mathbf{v} \right| \geq |\alpha\, \beta\, \ell'(0)\, \tau_c(\epsilon)| \right\} \leq 2 \exp\left(-\frac{TB\, (\alpha\, \beta\, \ell'(0)\, \tau_c(\epsilon))^2}{2\sigma^2 M^2} \right),$$

for any $\mathbf{v} \in \mathbb{R}^k : \|\mathbf{v}\| \leq M$. Thus, taking a union bound over all $i \in \mathcal{I}$,

$$\Pr\{\neg E_2\} \leq \sum_{i \in \mathcal{I}} \Pr\left\{ \neg E_2^{(i)} \right\} \leq 2I \exp\left(-\frac{TB\, (\alpha\, \beta\, \ell'(0)\, \tau_c(\epsilon))^2}{2\sigma^2 M^2} \right). \qquad (9)$$

Having upper-bounded both probabilities, we set the righthand sides of Equations 8 and 9 to $\delta/2$, then solve for the lower bound of T that satisfies each. We then have that $\Pr\{\neg E_1\} \leq \delta/2$ if $T \geq \frac{2 \ln(2/\delta)}{B(1-2p_c-\beta)^2}$, and $\Pr\{\neg E_2\} \leq \delta/2$ if $T \geq \frac{2\sigma^2 M^2 \ln(4I/\delta)}{B(\alpha\, \beta\, \ell'(0)\, \tau_c(\epsilon))^2}$. The maximum of the lower bounds for T ensures that both probabilities are upper-bounded, which means that $\Pr\{E_1 \wedge E_2\} \geq 1 - \delta$.

A.4 Proof of Theorem 3

Protocol 3 Communication Protocol Using SGD Without Replacement

```
procedure SERVER
    Select (random) subset of available clients, $\mathcal{C}' \subseteq \mathcal{C}$
    for $c \in \mathcal{C}'$ in parallel do
        $\{\Delta\mathbf{v}_i^{(c)}\}_{i \in \mathcal{I}} \leftarrow$ CLIENTUPDATE($c, \{\mathbf{v}_i\}_{i \in \mathcal{I}}, E$)
    $\forall i \in \mathcal{I}$, $\mathbf{v}_i \leftarrow \mathbf{v}_i + \sum_{c \in \mathcal{C}'} \Delta\mathbf{v}_i^{(c)}$
procedure CLIENTUPDATE($c, \{\mathbf{v}_i^{(0)}\}_{i \in \mathcal{I}}, E$)
    $\forall i \in \mathcal{I}$, $\mathbf{v}_i \leftarrow \mathbf{v}_i^{(0)}$
    for $e = 1, \ldots, E$ do
        Randomly split $\mathcal{S}_c$ into $R \triangleq N/B$ disjoint batches, $\{\mathcal{B}_r\}_{r=1}^{R}$, of size $B$
        for $r \in 1, \ldots, R$ do
            $\mathbf{g}_c \leftarrow \frac{1}{B} \sum_{i \in \mathcal{B}_r} \nabla_{\mathbf{u}_c} L(\mathbf{u}_c, \mathbf{v}_i, y_{c,i})$
            $\forall i \in \mathcal{B}_r$, $\mathbf{v}_i \leftarrow \mathbf{v}_i - \alpha \nabla_{\mathbf{v}_i} L(\mathbf{u}_c, \mathbf{v}_i, y_{c,i})$
            $\mathbf{u}_c \leftarrow \mathbf{u}_c - \alpha \mathbf{g}_c$
    return $\{\tilde{\Delta}\mathbf{v}_i\}_{i \in \mathcal{I}}$ per Equation 3
```

To account for changes in the client-side protocol, we use a slight modification of the attack in Protocol 2. Like before, we call CLIENTUPDATE with the item vectors all set to zero, $\{\mathbf{0}\}_{i \in \mathcal{I}}$; however, instead of using $R = 1$, we use $E = 1$. Further, we estimate the user vector as

$$\hat{\mathbf{u}}_c \leftarrow \frac{1}{\alpha\, \ell'(0)\, T\, N} \sum_{t=1}^{T} \sum_{i \in \mathcal{I}} \tilde{\Delta}\mathbf{v}_i^{(t)}$$

Note that it is easy to determine N based on which items are updated. Even if this number were slightly perturbed, it would have little impact on identifiability. Also note that $\mathbf{u}_c$ will not change after one epoch, since each item vector is used in exactly one update and (via Equation 1) will contribute zero to the gradient with respect to $\mathbf{u}_c$.

Similar to the proof of Theorem 2, we will show that, with enough interactions, the noise in $\hat{\mathbf{u}}_c$ averages out to zero (its mean). However, unlike Theorem 2, we do not have to reason about the distribution of labels used in $\hat{\mathbf{u}}_c$, because it is always determined by p_c.

For an arbitrary vector, $\mathbf{v} \in \mathbb{R}^k$, we have that

$$\begin{aligned}
\hat{\mathbf{u}}_c \cdot \mathbf{v} &= \frac{1}{\alpha\,\ell'(0)\,T\,N} \sum_{t=1}^{T} \sum_{i \in \mathcal{S}_c} \left(\Delta \mathbf{v}_i^{(t)} + \boldsymbol{\varepsilon}_i^{(t)} \right) \cdot \mathbf{v} \\
&= \frac{1}{\alpha\,\ell'(0)\,T\,N} \sum_{t=1}^{T} \sum_{i \in \mathcal{S}_c} \left(-\alpha\,\ell'(0)\,y_{c,i}\,\mathbf{u}_c + \boldsymbol{\varepsilon}_i^{(t)} \right) \cdot \mathbf{v} \\
&= \underbrace{(1 - 2p_c)\,\mathbf{u}_c \cdot \mathbf{v}}_{\text{(a)}} \; + \; \underbrace{\frac{1}{\alpha\,\ell'(0)} \left(\frac{1}{TN} \sum_{t=1}^{T} \sum_{i \in \mathcal{S}_c} \boldsymbol{\varepsilon}_i^{(t)} \cdot \mathbf{v} \right)}_{\text{(b)}} .
\end{aligned}$$

If $|\mathbf{u}_c \cdot \mathbf{v}| > \tau_c(\epsilon)$, then

$$\left| \frac{1}{TN} \sum_{t=1}^{T} \sum_{i \in \mathcal{S}_c} \boldsymbol{\varepsilon}_i^{(t)} \cdot \mathbf{v} \right| < |\alpha\,(1 - 2p_c)\,\ell'(0)\,\tau_c(\epsilon)| \tag{10}$$

ensures that (a) has greater magnitude than (b), implying $\operatorname{sgn}(\hat{\mathbf{u}}_c \cdot \mathbf{v}) = \operatorname{sgn}(\mathbf{u}_c \cdot \mathbf{v})$. Let E_i denote the event that Equation 10 holds for $\mathbf{v}_i$. Then, since at most an ϵ fraction of items have $|\mathbf{u}_c \cdot \mathbf{v}_i| \leq \tau_c(\epsilon)$, the error rate will be at most ϵ with probability $\Pr\{\bigwedge_{i \in \mathcal{I}} E_i\}$.

We lower-bound $\Pr\{\bigwedge_{i \in \mathcal{I}} E_i\}$ using the same proof technique we used for Theorem 2. Since $\Pr\left\{\bigwedge_{i \in \mathcal{I}} E_i\right\} \geq 1 - \sum_{i \in \mathcal{I}} \Pr\{\neg E_i\}$, it suffices to upper-bound $\Pr\{\neg E_i\}$. The lefthand side of Equation 10 is an absolute average of TN independent, zero-mean Gaussians, each with variance upper-bounded by $\sigma^2 M^2$. Thus, using standard Gaussian tail bounds,

$$\Pr\{\neg E_i\} \leq 2 \exp\left(-\frac{TN\,(\alpha\,(1 - 2p_c)\,\ell'(0)\,\tau_c(\epsilon))^2}{2\sigma^2 M^2} \right).$$

Summing over $i \in I$, then solving for T, we obtain Equation 4.