A General Approach to Adding Differential Privacy to Iterative Training Procedures

H. Brendan McMahan
mcmahan@google.com

Galen Andrew
galenandrew@google.com

Abstract

In this work we address the practical challenges of training machine learning models on privacy-sensitive datasets by introducing a modular approach that minimizes changes to training algorithms, provides a variety of configuration strategies for the privacy mechanism, and then isolates and simplifies the critical logic that computes the final privacy guarantees. A key challenge is that training algorithms often require estimating many different quantities (vectors) from the same set of examples — for example, gradients of different layers in a deep learning architecture, as well as metrics and batch normalization parameters. Each of these may have different properties like dimensionality, magnitude, and tolerance to noise. By extending previous work on the Moments Accountant for the subsampled Gaussian mechanism, we can provide privacy for such heterogeneous sets of vectors, while also structuring the approach to minimize software engineering challenges.

1 Introduction

There has been much work recently on integrating differential privacy (DP) techniques into iterative training procedures like stochastic gradient descent [Chaudhuri et al., 2011, Bassily et al., 2014, Abadi et al., 2016, Wu et al., 2017, Papernot et al., 2017]; for completeness we provide a formal definition of DP in Appendix A. Although these works differ in the granularity of privacy guarantees offered and the method of privacy accounting, most proposed approaches share the general idea of iteratively computing a model update from training data and then applying the Gaussian mechanism for differential privacy to the update before incorporating it into the model. Our goal in this work is to decouple, to the extent possible, three aspects of integrating a privacy mechanism with the training procedure:

a) the specification of the training procedure itself (e.g., stochastic gradient descent with batch normalization and simultaneous collection of accuracy metrics and training data statistics),
b) the selection and configuration of the privacy mechanisms to apply to each of the aggregates collected (model gradients, batch normalization weight updates, and metrics), and
c) the accounting procedure used to compute a final (ϵ, δ)-DP guarantee.

This separation is critical: the person implementing a) is likely not a DP expert, and this code typically already exists; there are many configuration options for b), which will likely require experimentation, and this configuration logic may become complex; thus isolating the key privacy calculations in c) and keeping them as simple (and well tested) as possible prevents bugs in a) or b) from introducing errors in the calculation of the actual privacy achieved.

While model training is our primary motivation, the approach is applicable to any iterative procedure that fits the following template. We have a database with n records. A *record* might correspond to a single training example, a "microbatch" of examples, or all of the data from a particular user or entity (e.g., to achieve user-level DP as in McMahan et al. [2018]). On each round, a random subset of records (a *sample*) is selected and the training procedure consumes the results of a number of vector

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

	Per-example SGD	Microbatch SGD	Federated learning (user-level DP)
record	gradient on one example	average gradient on one *microbatch* (~10 examples)	model update from one user
sample	minibatch (~100 examples)	minibatch (~10 microbatches containing 100 examples)	set of participating user devices for the round

Table 1: Defining *record* and *sample* in different training contexts.

queries over that sample; see Table 1. Such vector queries may include the average gradient for each layer, updates to batch-normalization parameters, or the average value for different training accuracy metrics. We describe a general approach to allocating a privacy budget across each of these queries and analyzing the privacy cost of the complete mechanism, all respecting the decoupling of concerns described earlier. Our analysis builds on the Moments Accountant approach of Abadi et al. [2016], which applies to a single vector query per round, and generalizes the extension of McMahan et al. [2018] to multi-vector queries.

We focus on the following basic building block for a single vector. Suppose we have a database X with n records consisting of vectors $x^i \in \mathbb{R}^D$ and we are interested in estimating the average[1] $\frac{1}{n}\sum_i x^i$. Given a selection probability q, clipping threshold S, and noise scale z, the procedure is:

1. Select a subset of the records $R \subseteq [1, \dots, n]$ by choosing each record with probability q.
2. Clip each x^i for $i \in R$ to have maximum L_2 norm S using $\pi_S(x) = x \cdot \min(1, S/\|x\|)$.
3. Output $\hat{x} = \frac{1}{qn}\left(\sum_{i \in R} \pi_S\left(x^i\right) + \mathcal{N}(0; \sigma^2 I)\right)$ where $\sigma = zS$.

The quantity $\sum_{i \in R} \pi_S\left(x^i\right) + \mathcal{N}(0; \sigma^2 I)$ is the output of the Gaussian mechanism for sums. As $\mathbb{E}[|R|] = qn$, scaling it by $1/qn$ produces an unbiased estimate of the average. The *noise scale* $z \equiv \sigma/S$ (the ratio of the noise to the L_2-sensitivity of the query) acts as a knob to trade off privacy vs. utility. If we choose $z = \frac{1}{\epsilon}\sqrt{2 \ln 1.25/\delta}$, the mechanism is $(q\epsilon, q\delta)$-differentially private with respect to the full database [Beimel et al., 2014, Dwork and Roth, 2014]. Importantly, the privacy cost of this mechanism is fully specified by q together with the *privacy tuple* (S, σ), where S is an upper bound on the L_2 norm of the vectors being summed, and σ is the standard deviation of the noise added to the sum.

We generalize the above procedure to the case where each record corresponds to a collection of vectors. We still do the sampling step (1) only once, but we estimate the average of each of the vectors separately, potentially with different clipping thresholds and noise scales. Let $(v_1, \dots, v_m)$ be the total set of vectors for which averages are to be estimated privately. In general, we may partition this set of m vectors into multiple groups, e.g., fully connected layers vs. convolutional layers vs. metrics. We assume the user (that is, the person using the privacy tools defined here) has identified the relevant set of groups whose averages are needed in the training procedure. For each of these, she needs to specify a privacy mechanism together with some hyperparameters. We first describe the privacy mechanisms that can be applied to individual vectors or groups of vectors, then show how the privacy cost of the full collection of mechanisms can be calculated, and finally propose strategies for choosing the parameters to achieve the desired privacy versus utility tradeoff.

2 Privacy mechanisms for a group of vectors

In this section, we describe two strategies that can be applied to a single group of vectors, *WLOG* the first k, $(v_1, \dots, v_k)$, for $k \leq m$; when $k = 1$, the two mechanisms described are identical.

Separate clipping and noise parameters. This strategy essentially treats the whole group as a single concatenated vector $v = (v_1, \dots, v_k)$. The user provides S_g, a clipping parameter, and σ_g, a noise parameter. For now, assume both of these parameters are simply chosen so as to provide

[1] For simplicity, we focus on unweighted average queries, for example to compute average gradient on a batch of examples; the generalization to weighted average and vector sum queries is straightforward. We also restrict attention to the fixed expected denominator f_f of McMahan et al. [2018]; extension to other estimators for averages like their f_c is straightforward.

reasonable utility for the resulting average; we will discuss strategies for choosing these parameters in detail in Section 4. The output of the mechanism is

$$\hat{v}_j = \frac{1}{qn} \sum_{i \in R} \pi_{S_g}\left(v^i\right)_j + \mathcal{N}(0; \sigma_g^2 I) = \frac{1}{qn} \left(\sum_{i \in R} \pi_{S_g}\left(v^i\right) + \mathcal{N}(0; \tilde{\sigma}_g^2 I) \right)_j ,$$

where $\tilde{\sigma}_g = qn\sigma_g$. The final expression shows that the mechanism is equivalent to the Gaussian mechanism for sums with privacy tuple $(S_g, \tilde{\sigma}_g)$. Applying this mechanism with $S_g = S^*$ to all m vectors recovers the "flat clipping" approach of McMahan et al. [2018], and applying this mechanism separately to each of the vectors with $S_g = S^*/\sqrt{m}$ recovers their "per-layer clipping" approach (where S^* is the total L_2 bound). Another reasonable strategy that takes into account dimensionality is to apply the mechanism separately with $S_g = S^*/\sqrt{d_g/D}$ where d_g is the dimensionality of v_g.

Joint clipping. Here we introduce a new mechanism that allows us to clip less aggressively than applying the previous strategy to each vector individually, while still letting different vectors live on different multiplicative scales. The user supplies as input scale parameters $\alpha_1, \ldots \alpha_k$ for $v_1, \ldots, v_k$, an overall σ_g the whole group, and a total clipping parameter $S_g \in \left[1, \sqrt{k}\right]$; noise $\alpha_j \sigma_g$ is added to the estimate for vector v_j. The strategy first does a pre-processing step via the scaling operator $\mathbf{s}(v; \alpha_{1:k}) = (v_1/\alpha_1, \ldots, v_k/\alpha_k)$. Observe that if $\forall j,\ \|v_j\| \leq \alpha_j$, then $\|\mathbf{s}(v; \alpha_{1:k})\| \leq \sqrt{k}$, but it may typically be less; thus $S_g = \sqrt{k}$ is a conservative default choice. The mechanism's output then scales the vectors back by the α_j factor in post-processing:

$$\begin{aligned} \hat{v}_j &= \frac{1}{qn} \sum_{i \in R} \alpha_j \pi_{S_g}\left(\mathbf{s}(v^i; \alpha_{1:k})\right)_j + \mathcal{N}\left(0; (\alpha_j \sigma_g)^2 I\right) \\ &= \frac{\alpha_j}{qn} \left(\sum_{i \in R} \pi_{S_g}\left(\mathbf{s}(v^i; \alpha_{1:k})\right) + \mathcal{N}\left(0; \tilde{\sigma}_g^2 I\right) \right)_j , \end{aligned}$$

where again $\tilde{\sigma}_g = qn\sigma_g$. The final expression shows the output can be written as a post-processing of the subsampled Gaussian mechanism for sums with privacy tuple $(S_g, \tilde{\sigma}_g)$. Note that if no clipping happens then $\alpha_j \pi_{S_g}\left(\mathbf{s}(v^i; \alpha_{1:k})\right)_j = v_j^i$ for all j.

To see where this mechanism might be superior to the first, suppose v_1 and v_2 have $\|v_1\| \leq 1$ and $\|v_2\| \leq 100$, and suppose they can tolerate noise standard deviations of 0.01 and 1 respectively. Additionally, assume it is known that either v_1^i or v_2^i will be zero for any record i. We could clip these separately, but this ignores the (useful) side information that one is always zero. On the other hand, if we treat them as a single group, we cannot take into the account the fact they are on very different scales; in particular, we must pick a single noise value which will either be insufficient to add privacy for v_2, or will completely obscure the signal in v_1. The joint mechanism proposed here lets us directly handle this situation using $\alpha_1 = 1$, $\alpha_2 = 100$, $\sigma_g = 0.01$, and $S = 1$.

3 Composing privacy guarantees for multiple vector groups

Now, suppose we have partitioned the m vectors into G groups, and selected a privacy mechanism for each one, producing privacy tuples $(S_g, \tilde{\sigma}_g)$ for $g \in \{1, \ldots, G\}$. From a privacy accounting point of view, each of these mechanisms is equivalent to running a Gaussian sum query on vectors w_g with $\|w_g\| \leq S_g$ and then adding noise $\tilde{\sigma}_g$ to the final sum. We now demonstrate a transformation that lets us analyze this composite mechanism as a single Gaussian sum query on the sample.

First, we scale each vector $\mathbf{s}(w; \tilde{\sigma}_{1:G}) = (\frac{w_1}{\tilde{\sigma}_1}, \ldots, \frac{w_G}{\tilde{\sigma}_G})$, so $\|\mathbf{s}(w; \tilde{\sigma}_{1:G})\| \leq S^* \equiv \sqrt{\sum_g \left(S_g/\tilde{\sigma}_g\right)^2}$. Now, we imagine a single Gaussian sum query with noise standard deviation $\sigma = 1$, and output the estimate after rescaling by the $\tilde{\sigma}_g$ factors. This is equivalent since

$$\hat{w}_g = \frac{1}{qn} \left(\sum_{i \in R} w_g^i + \mathcal{N}(0; \tilde{\sigma}_g^2 I) \right) = \frac{\tilde{\sigma}_g}{qn} \left(\sum_{i \in R} \mathbf{s}(w^i; \tilde{\sigma}_{1:G}) + \mathcal{N}(0; I) \right)_g . \tag{1}$$

The final expression is a simple post-processing on the output of a single Gaussian sum query with parameters $(S^*, \sigma = 1)$. Thus, we can apply the moments accountant to bound the privacy loss of iterative applications of this mechanism.

4 Hyperparameter selection strategies

Here we consider selecting hyperparameters q, S_g, and σ_g to achieve a particular privacy vs. utility tradeoff. Recall for both mechanisms, $\tilde{\sigma}_g = qn\sigma_g$, so the key quantity is

$$z = \frac{1}{S^*} = \left(\sum_g (S_g/\tilde{\sigma}_g)^2 \right)^{-1/2} = qn \left(\sum_g (S_g/\sigma_g)^2 \right)^{-1/2} .$$

Typically, a value of $z \approx 1$ will provide a reasonable privacy guarantee. If z is too small for the desired level of privacy, the user has several knobs available: clip more aggressively by decreasing the S_g's; noise more aggressively by scaling up the σ_g's; or increasing q. When datasets are large and the additional computational cost of processing larger samples R is affordable, this last approach is generally preferable, as observed by McMahan et al. [2018]. If additionally the total number of iterations T is known, then since the privacy cost scales monotonically with any of these adjustments to z, a binary search can be performed using the moments accountant repeatedly with different parameters to find e.g. the precise value of q needed to achieve a particular (ϵ, δ)-DP guarantee.

Choosing σ_g and S_g. Typical approaches to setting S_g include: 1) using an *a priori* upper bound on the L_2 norm; 2) choosing S_g so that "few" vectors are clipped; or 3) running parameter tuning grids to find a value of S_g that does not reduce utility (e.g., the accuracy of the model) by too much. If private data is used in 2) or 3), the privacy cost of this should be accounted for. Similar strategies can be used to choose σ_g, e.g., selecting a value that will introduce an *a priori* acceptable amount of error, or more likely for model training, running experiments to find the largest amount of noise that does not slow the training procedure.

In some cases one may have bounds S_g on the norms of G groups plus an overall target value of z, which needs to be distributed across multiple groups. To achieve *proportional* noise, where $\tilde{\sigma}_g \propto S_g$ for all g, we can use $\tilde{\sigma}_g = z\sqrt{G}S_g$. Another reasonable alternative, *dimensionality adjusted* noise assigns noise proportional to the maximum root mean squared value of the components of w_g given its bound and its dimensionality: $\tilde{\sigma}_g = z\sqrt{D/d_g}S_g$, where d_g is the dimensionality of group g.

5 Privacy ledger

In principle, privacy accounting (via, e.g. the moments accountant) could be done in tandem with calls to the mechanism to keep an online estimate of the (ϵ, δ) privacy guarantee. However we advocate a different approach which cleanly separates concerns b) and c) from the introduction. We maintain a *privacy ledger* and record two types of events: *sampling events*, which record that a set R of records has been drawn using parameters q and n, and *sum query* events, which record that a Gaussian sum query has been performed over some group of vectors with privacy tuple $(S_g, \tilde{\sigma}_g)$. Then the privacy accountant can process the ledger *post hoc* to produce a privacy guarantee, first converting each group of one sampling event plus some sum query events to an equivalent single sum query event with parameters $(S^*, \sigma = 1)$ using Equation (1).

There are two main advantages of this approach. First, bugs in the hyperparameter selection strategy code cannot affect the privacy estimate. Second, it allows the privacy accounting mechanism to be changed and the ledger reprocessed if, for example, a tighter bound on the privacy loss is discovered after the data has been processed.

6 Conclusion

We have shown how the Gaussian mechanism can be applied to vectors of different types with different norm bounds and noise standard deviations, enabling training over heterogeneous parameter vectors, as well as simultaneous privacy-preserving estimation of other statistics such as classifier accuracy, or the number of instances in each class. By implementing iterative training algorithms in terms of a series of Gaussian sum queries and then recording for each query privacy events to a ledger to be processed by a privacy accountant, we separate the three major concerns of implementing privacy-preserving iterative training procedures while allowing flexibility in the specification of clipping strategy and noise allocation.

References

Martin Abadi, Andy Chu, Ian Goodfellow, Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *23rd ACM Conference on Computer and Communications Security (ACM CCS)*, 2016.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 464–473, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-6517-5. doi: 10.1109/FOCS.2014.56. URL `http://dx.doi.org/10.1109/FOCS.2014.56`.

Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. *Machine Learning*, 94(3):401–437, 2014. doi: 10.1007/s10994-013-5404-1. URL `http://dx.doi.org/10.1007/s10994-013-5404-1`.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12, July 2011.

Cynthia Dwork and Aaron Roth. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2014.

Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://openreview.net/pdf?id=BJ0hF1Z0b`.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *Proceedings of the International Conference on Learning Representations*, 2017. URL `https://arxiv.org/abs/1610.05755`.

Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey F. Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of SIGMOD*, 2017.

A Differential Privacy

The formal definition of (ϵ, δ)-differential privacy is provided here for reference:

Definition 1. *A randomized mechanism* $\mathcal{M} \colon \mathcal{D} \mapsto \mathcal{R}$ *satisfies* (ϵ, δ)**-differential privacy** *if for any two adjacent datasets* $X, X' \in \mathcal{D}$ *and for any measurable subset of outputs* $\mathcal{Y} \subseteq \mathcal{R}$ *it holds that* $\Pr[\mathcal{M}(X) \in \mathcal{Y}] \leq e^{\epsilon} \Pr[\mathcal{M}(X') \in \mathcal{Y}] + \delta$.

The interpretation of *adjacent datasets* above determines the unit of information that is protected by the algorithm: a differentially private mechanism guarantees that two datasets differing only by addition or removal of a single unit produce outputs that are nearly indistinguishable. For machine learning applications the two most common cases are *example-level* privacy (e.g., Chaudhuri et al. [2011], Bassily et al. [2014], Abadi et al. [2016], Wu et al. [2017], Papernot et al. [2017]), in which an adversary cannot tell with high confidence from the learned model parameters whether a given example was present in the training set, or *user-level* privacy (e.g., McMahan et al. [2018]) in which adding or removing an entire user's data from the training set should not substantially impact the learned model.[2]

[2] It is also possible to consider X and X' to be adjacent if they differ by *replacing* a training example (or an entire user's data) with another, which would increase the ϵ by a factor of two.