École Normale Supérieure Paris-Saclay

Master MVA

Research Internship Report

Differential Privacy for Machine Learning

École Polytechnique Fédérale de Lausanne

Machine Learning & Optimization Laboratory

Author:
Eloïse Berthier

Supervisors:
Prof. Martin Jaggi
Sai Praneeth Karimireddy

September 11, 2019

Abstract

Machine learning algorithms can leak private information contained in particular training data. Differential privacy ensures that an algorithm does not rely too strongly on any individual data point. Differentially private machine learning can be achieved by injecting noise in the training process. In particular, the privacy of DP-SGD has already been well-studied, yet only in the case where each training example is sampled with replacement. We focus on the more practical case of sampling without replacement, or shuffling, and try to provide privacy guarantees for this algorithm. We also explore possible relaxations of differential privacy.

Keywords: differential privacy, machine learning, stochastic gradient descent, sampling schemes.

Contents

Introduction

Over the past few years, data collection has increased dramatically, allowing large scale data analyses, in fields as diverse as advertising, online recommendation, computer vision, natural language processing or medicine. Some of these data are related to individuals, such as demographic data, video surveillance, web traffic logs, geolocation, emails, health records or even genomic data. This of course raises privacy issues, as pointed out by the recent establishment of General Data Protection Regulation (GDPR) in the European Union. Now states and private companies are also willing to share data, with the restriction that they do not release those that could harm their competitive edge or the privacy of their users.

In addition to whole datasets, it is likely that trained machine learning models will become a new trading value to be exchanged or released, for commercial or scientific purposes. This hypothesis is plausible, especially because of the large computational and human resources required during training, which will not be accessible to all organizations. However, it is unclear how such a model exposes its training data, especially in the case of complex models, such as neural networks which are still poorly understood. Recent striking experimental studies suggest that some models indeed memorize and leak private information from some of their training data points.

Meanwhile, individual are becoming more suspicious about the way their personal data are collected and used. That could lead them, often rightly, to withdraw from data collection processes when they are not informed of the intentions of data collectors. It would be sensible to provide them with some guarantee, asserting how much their privacy will be harmed by some process using their personal data. The notion of differential privacy is one possible approach to cope with this issue. In this report, we are particularly interested in how a machine learning model exposes its training data, that is in the use of differential privacy for machine learning.

This has sparked some interest in the last five years, both from compute security and machine learning communities. It has led to a few privacy preserving learning algorithms, some of them with practical implementations, others hardly applicable. However, there is a trade-off between preserving privacy and yet providing a useful algorithm. It seems that for now, the loss in utility is still crippling as compared to the gain in privacy, which is in addition hardly interpretable. It might be that the current definition of differential privacy is not well suited for machine learning. That is why a significant part of the effort in this field has been focused on finding the right relaxations and definitions of privacy. A clear consensus does not seem to have emerged yet, keeping the room open for new propositions.

In this report, we first quickly present the main privacy challenges in machine learning, before defining the notion of differential privacy and its main properties. Then we move on to its applications to learning and optimization processes, with a particular focus on the stochastic gradient descent algorithm. This part gathers most of the work conducted during this internship. Finally, we explore other possible notions of privacy, focusing on distributed settings.
Some parts of this report are based on existing literature, the others present explorations carried out during the internship and can be considered as independent. These are marked with a * in their title.

Acknowledgments

I would like to express my thanks to Martin for having welcomed me in the MLO lab and for his kind encouragements. A warm thank you to Praneeth who accompanied me in this quick trip into privacy and who very often guided my steps with great patience. Thank you to the whole MLO team: you have made my internship optimal!
I acknowledge support from the *Direction Générale de l'Armement*.

1 Motivation: Attacks Against Privacy

Machine learning models extract knowledge from their training data. When the data are personal or sensitive, the model could potentially reveal pieces of information that are specific to one data point. A striking example of data leakage has been recently provided in the field of natural language processing [6]. A neural network could memorize a random secret word inserted in its training dataset and this word could be reconstructed from the trained model only. Similarly, an auto completion model could possibly learn to predict a user's password each time its username is entered. This issue has significant implications in other fields such as medicine, census records or finance, as it spoils opportunities to share trained models across organizations.

There are two main categories of attacks than can be performed to recover information from the training data:

- *membership* attack: an adversary wants to know if a particular data point was used to train the model;
- *reconstruction* attack: an adversary wants to reconstruct some training data points or some of their features.

A membership attack is successively performed if the adversary manages to build a classifier that predicts if a data point (rather its discretization) was used with a significant discrimination, for instance measured by the area under the ROC curve. The reconstruction attack is harder to carry out as it implies ability to perform membership attacks. Suppose the training process is invertible, than it would theoretically be possible to reconstruct the dataset from the model by brute force. Yet in general it would require an exponential number of operations. Robustness towards reconstruction attacks should instead be quantified against an adversary with limited polynomial time computing power.

These kinds of attacks come with two modalities:

- *white-box* attack: the adversary is granted access to the weights of the model;
- *black-box* attack: the adversary is only granted access to a possibly limited number of inference queries to the model.

Obviously it is easier to perform the attack in the white-box setting. The black-box setting is closer to the whole area of adversarial examples.

An intuitive way to carry a membership attack is to look at the confidence of the model. Typically, the model will be overconfident when it runs inference on training examples. An adversary would try to maximize the confidence of the model with respect to the input data. In the white-box setting, one can simply do gradient ascent on the input, whereas in the black-box setting, one could use for

Figure 1: This situation is not too far from the results of [6] and the issue should be taken seriously for natural language processing (xkcd comic of June 28, 2019).

instance zero-order optimization, yet requiring a higher number of queries to converge.

The authors of [6] base their attack on a measure of surprise - the log-perplexity - of the model with respect to a sequence of inputs. They can efficiently retrieve a secret word inserted in the training set. Yet models trained with differential privacy do not memorize the secret word, and are immune to this attack for reasonable privacy parameters.

This is our main motivation to study differential privacy applied to the learning process. In the following sections, we will implicitly focus on the white-box setting. Indeed, we will study the privacy of an algorithm that takes training data as input and outputs the weights of a model.

2 Differential Privacy

The main definition of differential privacy (DP) has been stated in 2006 by Dwork et al [10]. It is an information theoretic guarantee that applies to any algorithm using data. Consider two datasets D and D' that differ only by one data point. An algorithm $\mathcal{A}$ is differentially private if its output is not significantly different when it is run on D or D'. The difference is measured by some statistical distance between the two outputs. Achieving this guarantee requires to add some randomness to the result of data queries, hence hiding potential specificities of each data point.

2.1 Definition

The following three sections are based on the extensive review of differential privacy [11]. We consider a dataset as a collection of records from a space $\mathcal{X}$. Each dataset x can be represented by an histogram over $\mathcal{X}$, that is, $x \in \mathbb{N}^{|\mathcal{X}|}$. We can measure the distance between two datasets as

$$\|x - y\|_1 := \sum_{i=1}^{|\mathcal{X}|} |x_i - y_i|$$

It quantifies how many records differ between x and y. An algorithm $\mathcal{A}$ acts on a dataset by reading its records and optionally auxiliary data, and returning some result. When the algorithm is randomized, it maps the training data to a probability distribution $\mathcal{A}(x) = \mu$. Differential privacy quantifies how much one particular record from the dataset can influence the output of the algorithm.

Definition 2.1 ((ε, δ) differential privacy)**.** *A randomized algorithm $\mathcal{A}$ is said to be (ε, δ)-differentially private if $\forall S \subset Im(\mathcal{A})$ and $\forall x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:*

$$\mathbb{P}[\mathcal{A}(x) \in S] \leq e^{\varepsilon}\mathbb{P}[\mathcal{A}(y) \in S] + \delta$$

or equivalently, with $\mathcal{A}(x) = \mu$ and $\mathcal{A}(y) = \nu$:

$$\sup_{S \subset Im(\mathcal{A})} \mu(S) - e^{\varepsilon}\nu(S) \leq \delta$$

This definition is already a relaxation of the original *pure* ε DP, corresponding to $\delta = 0$ and stronger than (ε, δ) privacy with $\delta > 0$. One of the reasons for this relaxation is that ε DP focuses on all events with non-zero probability, even the very unlikely ones. (ε, δ) DP ignores events with probability less than δ, and roughly means that ε DP holds with high probability $1 - \delta$.

Small values of ε correspond to higher privacy, $\varepsilon = 0$ meaning that the algorithm does not use the data at all. Of course, to achieve some utility, the algorithm will reach some $\varepsilon > 0$. δ should ideally

be set as *cryptographically* small. In practice, one should keep it negligible with respect to the inverse of the number of data points n. Indeed, completely releasing one data point at random satisfies $(0, \delta)$ DP for $\delta = 1/n$. This value of δ is hence considered dangerous.

Note that the definition requires the inequality to hold simultaneously for any two neighboring datasets, and the roles of x and y are symmetric. Then if $\mathcal{A}$ returns a deterministic response that depends on the data, μ and ν are just diracs in different locations, and (ε, δ) DP is impossible to reach for any $\varepsilon < +\infty$.

2.2 Privacy Techniques

A simple method to build a DP algorithm is to add some noise to the result of a query on the dataset. More precisely, adding properly scaled Laplace noise guarantees ε DP, while adding Gaussian noise only guarantees (ε, δ) DP. This is probably the main reason of the adoption of (ε, δ) DP, together with better composition rules (see next section).

The art of DP lies in scaling the noise with respect to the privacy. Intuitively, adding more noise masks more effectively the data and should increase the privacy and decrease ε. Privacy is related to a third factor, the sensitivity of the query. It measures the maximum deviation (with respect to some norm) of a query f when one data point is changed, or:

$$S(f) := \sup_{x,y,\|x-y\|\leq 1} \|f(x) - f(y)\|$$

Sometimes, $S(f)$ is simple to estimate, but for complex queries like training a neural network, this is much harder.

In dimension $d \geq 1$, the noise is added independently to each component of the query. Here is a simple relation between noise, privacy and sensitivity:

Proposition 2.1 (Laplace mechanism)**.** *Let $S_1(f)$ be the sensitivity of the query f with respect to the ℓ_1 norm. Adding Laplace noise to the query, sampled according to Lap$(S_1(f)/\varepsilon)$, guarantees ε DP.*

There is an equivalent result for Gaussian noise:

Proposition 2.2 (Gaussian mechanism)**.** *Let $S_2(f)$ be the sensitivity of the query f with respect to the ℓ_2 norm. Let $c^2 > 2\log(1.25/\delta)$ and $\sigma \geq cS_2(f)/\varepsilon$. Adding Gaussian noise to the query, sampled according to $\mathcal{N}(0, \sigma^2)$, guarantees (ε, δ) DP.*

A simple example The simplest privacy technique, which is adapted to discrete output spaces, is randomized response. Say you want to do a survey in a city to get an estimation of the number of

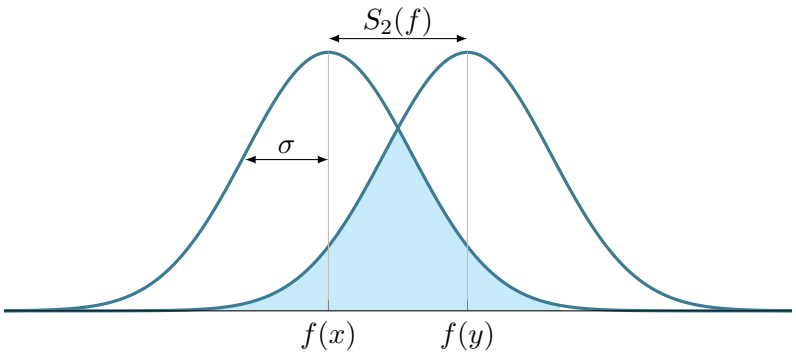

Figure 2: Masking the output of a query by the Gaussian mechanism: as more noise is added, the outputs become indistinguishable.

drug consumers. You can ask the question directly to individuals – *Are you a drug consumer?* – but the answer will obviously harm their privacy. Instead, you can propose them the following protocol. Flip a coin, if it is heads, answer truthfully. Else flip another coin and answer yes if heads and no if tails. By doing so, the answer has some random noise that allows the respondent to negate his answer and pretend it was only accidental. If the number of respondents is large enough, the surveyor can still compute a good estimate of drug consumers without compromising the privacy of each individual. This algorithm satisfies $\varepsilon = \log(3)$ DP with balanced coins, which can be modulated by changing the odds of the first coin.

2.3 Properties

DP is a constructive guarantee that is usually computed by using simple properties as building blocks. The example above highlights some of its fundamental properties. The most basic one is robustness to post-treatment. It is not possible to further compromise the privacy of an individual just by *thinking* about his answer. Another important property is composition. If two requests with DP ε_1 and ε_2 are performed on the same dataset, then the composition of both has privacy $\varepsilon_1 + \varepsilon_2$. In other words, performing several sensitive queries significantly degrades privacy. This property is intuitive: if one repeats 100 times the survey described above on the same person, one will obtain a very reliable estimate of his true answer. A last useful property is that of subsampling: if an individual has a probability less than one of being included in the study, then his privacy is preserved more.

We do not state here these properties more formally, as they will be further developed in other parts of this report. Robustness to post-treatment and basic composition are proved for an alternative definition of DP in the next subsection, with very similar proofs as for DP. Subsampling properties will be developed more in the next section for Rényi differential privacy.

Here we only give a glimpse of the composition property. ε DP satisfies basic composition, which

means that if $\mathcal{A}_1$ has DP ε_1 and $\mathcal{A}_2$ has DP ε_2, then $(\mathcal{A}_1, \mathcal{A}_2)$ has DP $\varepsilon_1 + \varepsilon_2$, in the worst case. This linear scaling is quite annoying because it limits drastically the number of queries that can be done on a dataset. However, a better sublinear composition property can be obtained for (ε, δ) DP, up to a slight loss in δ:

Theorem 2.1 (Advanced Composition Theorem)**.** *For all $\varepsilon, \delta, \delta' \geq 0$, the class of (ε, δ) DP algorithms satisfies $(\varepsilon', k\delta + \delta')$ DP under k-fold adaptive composition for:*

$$\varepsilon' = \sqrt{2k \log(1/\delta')}\varepsilon + k\varepsilon(e^{\varepsilon} - 1)$$

k-fold adaptive composition means that k algorithms can be run sequentially on the same dataset, and we allow each of them to take as auxiliary input the outputs of the previous ones. This means that a curious adversary could adapt his queries to the outputs he gets. We obtain a scaling in $\sqrt{k}$ instead of k for the basic composition, but this only applies to small values of ε. Another weakness of this theorem is that it only allows each mechanism to compose to have the same privacy (ε, δ). It has been later shown that finding the optimal composition theorem for general $(\varepsilon_i, \delta_i)_{1 \leq i \leq k}$ algorithms is NP-hard [22].

2.4 Total Variation Differential Privacy*

2.4.1 Definition

The condition for ε DP is a divergence and cares about the ratios of the probabilities. This means that regions where ν has a very small probability is given unduly large importance, even though they may never happen in practice. Here we consider a relaxation of DP using total variation. We replace the divergence with a distance measure and use total variation:

$$TV(\mu, \nu) := \sup_S |\mu(S) - \nu(S)| \leq \delta$$

This corresponds to (ε, δ) DP for $\varepsilon = 0$. Similarly, we must ensure that δ is very small compared to the inverse of the size of the dataset, we require typically that $\delta = o(1/n)$.
This relaxation has been scarcely studied, the only reference being [2]. It is both a distance and an f-divergence. It is weaker than the KL divergence, according to Pinsker's inequality. For all probability distributions μ, ν:

$$TV(\mu, \nu) \leq \sqrt{\frac{1}{2} KL(\mu || \nu)}$$

It is a particular case of a Wasserstein distance with transport cost $c(x, y) = \mathbf{1}_{x \neq y}$. However, it is not

obvious why other transport distances would be suitable for a definition related to indistinguishabilty, and not to proximity in a ground space. One could have two very close Dirac distributions, which are close with respect to the W_2 distance, but still clearly distinguishable.

2.4.2 Main Properties and Techniques

We check some properties of TV privacy by adapting results from [11]. The proofs are in the appendix. For a set $\mathcal{R}$, we write $\mathcal{M}(\mathcal{R})$ for the set of probability distributions on $\mathcal{R}$.

Proposition 2.3 (Immunity to post-processing)**.** *Let $\mathcal{A} : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{M}(\mathcal{R})$ be a randomized algorithm that is δ TV-private. Let $f : \mathcal{M}(\mathcal{R}) \to \mathcal{M}(\mathcal{R}')$ an arbitrary randomized mapping. Then $f \circ \mathcal{A} : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{M}(\mathcal{R}')$ is δ TV-private.*

Theorem 2.2 (Composition theorem)**.** *Let $\mathcal{A}_1 : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{M}(\mathcal{R}_1)$ be a δ_1 TV-private algorithm, and let $\mathcal{A}_2 : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R}_1 \to \mathcal{M}(\mathcal{R}_2)$ be a δ_2 TV-private algorithm. Then their adaptive composition $\mathcal{A}_{1,2} : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{M}(\mathcal{R}_1 \times \mathcal{R}_2)$ by the mapping $\mathcal{A}_{1,2}(x) = (\mathcal{A}_1(x), \mathcal{A}_2(x, \mathcal{A}_1(x)))$ is $\delta_1 + \delta_2$ TV-private.*

This result holds not only when the two algorithms are run independently, but even when subsequent computations can incorporate the outcomes of the preceding computations. We only require the randomness created by each algorithm to be independent.

Laplace Mechanism. Recall that the ℓ_1 sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^d$ is:

$$\Delta f := S_1(f) = \sup_{x,y \in \mathbb{N}^{|\mathcal{X}|}, \|x-y\|_1 \leq 1} \|f(x) - f(y)\|_1$$

Suppose we want to compute a query f on the dataset x. The Laplace mechanism consists in computing $f(x)$ and then adding some properly scaled Laplace noise to the result. The Laplace distribution is particularly well-behaved to create ε differential privacy. Here we show that it works as well with TV-privacy, with a differently scaled noise (variance $2b^2 = \dfrac{\Delta f}{\delta}$ vs $\dfrac{2\Delta f^2}{\varepsilon^2}$ for DP).

Proposition 2.4. *For any function $\mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^d$, the Laplace mechanism is defined as:*

$$\mathcal{M}_L(x, f, \delta) := f(x) + (Y_1, ..., Y_d)$$

where the Y_i are iid random variables drawn from $Lap(b)$. The Laplace mechanism with $b = \sqrt{\dfrac{\Delta f}{2\delta}}$ is δ TV-private.

Gaussian Mechanism. We use the same principle, except that here we consider the ℓ_2 sensitivity of functions and $\Delta f := S_2(f)$. Since the KL divergence between two Gaussian distributions can be computed exactly, one can directly upper-bound the total variation using Pinsker's inequality:

$$TV(\mathcal{N}(\mu_1, \sigma^2 I_d), \mathcal{N}(\mu_2, \sigma^2 I_d)) \leq \sqrt{\frac{1}{2} KL(\mathcal{N}(\mu_1, \sigma^2 I_d) \| \mathcal{N}(\mu_2, \sigma^2 I_d))} = \frac{\|\mu_2 - \mu_1\|_2}{2\sigma}$$

We could also get more precise upper-bounds on the total variation between two Gaussian distributions using for instance the results of [9]. For the Gaussian mechanism, we bound the total variation between $\mathcal{N}(f(x), \sigma^2 I_d)$ and $\mathcal{N}(f(y), \sigma^2 I_d)$ which is less than $\frac{\|f(x) - f(y)\|_2}{2\sigma} \leq \frac{\Delta_2 f}{2\sigma}$.

Proposition 2.5. *For any function $\mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^d$, the Gaussian mechanism is defined as:*

$$\mathcal{M}_G(x, f, \delta) := f(x) + (Y_1, ..., Y_d)$$

where the Y_i are iid random variables drawn from $\mathcal{N}(0, \sigma^2)$. The Laplace mechanism with $\sigma = \frac{\Delta_2 f}{2\delta}$ is δ TV-private.

Stronger Composition Rule. Our basic composition property is not fully satisfactory as it scales linearly with the number of queries k. For the Gaussian mechanism, one can do better and obtain a sublinear scaling in $\sqrt{k}$, just like the strong composition theorem for (ε, δ) privacy. We directly upper-bound the total variation between multidimensional Gaussian, which covers the adaptive composition of Gaussian mechanisms.

Suppose a first algorithm outputs $\mathcal{M}_1(x) = f(x) + \mathcal{N}(0, \sigma_1^2 I_d)$ and a second one takes as input the dataset and the output of $\mathcal{M}_1$, and outputs $\mathcal{M}_2(x) = g(x, \mathcal{M}_1(x)) + \mathcal{N}(0, \sigma_2^2 I_d)$.

We are interested in $TV((\mathcal{M}_0(x), \mathcal{M}_1(x)), (\mathcal{M}_0(y), \mathcal{M}_1(y)))$. The Kullback-Leibler divergence satisfies the chain rule:

$$KL(p(x, y)||q(x, y)) = KL(p(x)||q(x)) + KL(p(y|x)||q(y|x))$$

With Gaussian noise of size $\sigma_1 = \frac{\Delta f}{2\delta_1}$ and $\sigma_2 = \frac{\Delta g}{2\delta_2}$, we have:

$$\begin{cases} KL(\mathcal{M}_1(x) \| \mathcal{M}_1(y)) & \leq \frac{(\Delta f)^2}{2\sigma_1^2} & = 2\delta_1^2 \\ KL(\mathcal{M}_2(x) \| \mathcal{M}_2(y)) & \leq \frac{(\Delta g)^2}{2\sigma_2^2} & = 2\delta_2^2 \end{cases}$$

and then:

$$TV\left((\mathcal{M}_1(x), \mathcal{M}_2(x)), (\mathcal{M}_1(y), \mathcal{M}_2(y))\right) \leq \sqrt{\delta_1^2 + \delta_2^2}$$

This composition holds only for Gaussian noise, but it is more widely applicable than strong composition with (ε, δ) privacy, as we can compose steps with different privacy parameters.

2.4.3 Interpretation

Contrary to (ε, δ) and Rényi differential privacy (defined in the next section), TV privacy has only one parameter. It has also a more intuitive interpretation in terms of hypothesis testing [2].
We first flip a coin and pick at random one of two neighboring datasets x and y. Then we run algorithm $\mathcal{A}$ on the randomly chosen dataset Z, that we keep secret. Consider an adversary who looks to the output of the algorithm. He wants to test between two hypotheses: either the dataset x or y has been used. He is allowed to know x and y, and even to choose them in an adverse manner. He answers $\psi = 0$ if he thinks x was used and $\psi = 1$ else. Guessing at random would result in an expected error of 1/2. Privacy is preserved if no matter what the adversary does, he can never do much better than random guessing.
Type I and II errors are indeed related to the total variation between the two possible outputs of the algorithm. If $\mathcal{A}$ satisfies δ TV privacy, Le Cam's inequality says that:

$$\inf_{\psi} \mathbb{P}\left[\psi = 1 | Z = x\right] + \mathbb{P}\left[\psi = 0 | Z = y\right] \geq 1 - \delta$$

In other words, his probability of success is less than $1/2 + \delta/2$. In particular, this means that our definition only makes sense for small values of δ, keeping in mind that we need $\delta \leq o\left(\frac{1}{n}\right)$ to avoid full disclosure of one random data point. (ε, δ) DP also has a similar interpretation in terms of hypothesis testing.
While this guarantee is less stringent that the original differential privacy, it has a simpler interpretation and only one parameter that could more easily be communicated to a data provider.
This adversarial setting is also interesting to incorporate relaxations or restrictions to the abilities of the adversary. One could for instance consider an adversary with limited polynomial time computing power, or provide him with side information.

2.5 Rényi Differential Privacy

Rényi Differential Privacy (RDP) is a notion of privacy which is in between ε DP and (ε, δ) DP. It was introduced by [21], inspired by prior work from [1] on the moment accountant, a way to track the privacy loss of Gaussian mechanisms. In fact, Gaussian mechanisms have a better composition rule, that can be easily described by RDP. This notion measures the maximal statistical deviation of the output of the algorithm in terms of Rényi divergence of order α.

Definition 2.2 (Rényi Differential Privacy). *Let $\alpha > 1$, x and y two neighboring datasets, and $\mathcal{A}$ an algorithm outputting probability distributions p and q on a set E when run on x and y. $\mathcal{A}$ is said to have Rényi differential privacy $(\alpha, \varepsilon(\alpha))$ if:*

$$D_\alpha(p\|q) := \frac{1}{\alpha - 1} \log \int_E p(u)^\alpha q(u)^{1-\alpha} \mathrm{d}u \leq \varepsilon(\alpha)$$

For $\alpha \to 1$, it gets back to KL divergence, for $\alpha \to +\infty$, this is ε DP.

Proposition 2.6 (RDP of the Gaussian mechanism). *Let s be the sensitivity of the query f with respect to the ℓ_2 norm. Adding Gaussian noise to the query, sampled according to $\mathcal{N}(0, \sigma^2)$, guarantees RDP $\left(\alpha, \frac{\alpha s^2}{2\sigma^2}\right)$ for all $\alpha > 1$.*

An important property is that RDP can be converted into (ε, δ) privacy:

Proposition 2.7 (RDP to (ε, δ) DP). *If $\mathcal{A}$ satisfies (α, ε) RDP, it also satisfies $\left(\varepsilon + \frac{\log(1/\delta)}{\alpha - 1}, \delta\right)$ DP for any $0 < \delta < 1$.*

The composition property is simply additive, but it is enough to get an efficient tracking of the privacy of Gaussian mechanisms. In practice, we track the full privacy accountant $(\alpha, \varepsilon(\alpha))$ for all $\alpha > 1$, and then convert it to (ε, δ) privacy at some level $\delta > 0$, taking:

$$\varepsilon = \min_{\alpha > 1} \ \varepsilon(\alpha) + \frac{\log(1/\delta)}{\alpha - 1}\,.$$

3 Application to Machine Learning

3.1 Learning Privately

Given a dataset $\mathcal{D} = (x_1, ..., x_n) \in \mathbb{R}^{n \times r}$ of training examples, we want to learn a model $\theta \in \mathbb{R}^d$ that minimizes some training loss measured by a function $\ell : \mathbb{R}^d \times \mathbb{R}^r \to \mathbb{R}$. The problem is to find:

$$\theta^* \in \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i)$$

Now we consider the same problem, but we want in addition the model θ^* to *be* differentially private. This is a slight abuse of notation, as DP usually refers to an algorithm. Here we say that θ^* is DP if it was produced by a DP algorithm. The problem becomes:

$$\theta^* \in \underset{\theta \text{ private}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i)$$

For now, we found no way in the literature to express this constraint easily, so that the problem could be directly solved like a constrained minimization problem. It would at least require to learn probability distributions on θ, but it is unclear how to include the notion of worst case with respect to the datasets. This is still an open problem today that would possibly require further relaxations of the definition of DP.

For learning algorithms, the private data units that will be used and shared are stochastic gradients $\nabla\ell(\theta, x_i)$. Although they are not directly data points x_i, they can leak almost all the information contains by x_i: e.g. for logistic regression, each stochastic gradient is a scalar times the data point.

3.2 Main Approaches

Three main methods have been proposed to solve the empirical risk minimization problem with differential privacy. All of them introduce randomness at different steps of the optimization process:

- Objective perturbation;
- Output perturbation;
- Gradient perturbation[1].

For all of these approaches, the gradients or stochastic gradients are assumed to be bounded by a constant L:

$$\sup_{x,\theta} \|\nabla\ell(\theta, x)\| \leq L$$

[1] We will mainly consider this technique in the rest of this report, and we will sometimes call it DP-SGD.

L can be either computed with the parameters of the problem, or more frequently, it is ensured by gradient clipping. This operation ensures that no gradient gets too large, hence influencing the algorithm too much as an outlyer. Gradient clipping takes parameter L and scales down the norm of a vector to L if it is larger:

$$\mathit{Clip}(u, L) = \begin{cases} u & \text{if } \|u\| \leq L \\ \dfrac{L}{\|u\|} u & \text{if } \|u\| > L\,. \end{cases}$$

Gradient clipping will not be a major concern in the rest of this report, as it is known in practice not too harm convergence, and sometimes helps as a regularizing factor.

3.2.1 Objective Perturbation

The idea is to run an standard optimization algorithm on a modified objective function [7]. Some properly scaled random noise is added to the objective in the form of an affine function of the parameter:

$$\theta^* \in \operatorname*{argmin}_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i) + \eta^\top \theta, \quad \text{for } \eta \sim Lap(b)$$

This method has the advantage of being applicable to any optimization method, potentially first or second order methods. On the other hand, it only works for strongly convex objectives. The intuition is that in this case, the new optimal parameter θ^* should remain close to the non-private one.
If the problem is solved iteratively with gradient descent or SGD, this method can also be interpreted as a gradient perturbation technique, where each gradient is perturbed with a constant noise vector η sampled once and for all:

$$\theta_{t+1} = \theta_t - \gamma(\nabla \ell(\theta_t, x_{i(t)}) + \eta)$$

The major drawback of this method is that privacy guarantees are only provided for θ^* in the argmin of the objective function. If the optimization process is stopped before convergence, of if one of the intermediate iterates is released, then this comes with no privacy guarantee. For machine learning applications, this should be a major concern, as early stopping is often used to prevent overfitting. Overall, this method is well suited for simple strongly convex problems, with a careful tuning of the regularization parameters.

3.2.2 Output Perturbation

The idea here is to consider the whole optimization process as a single algorithm and to add noise directly to its output [8]. This requires to scale the noise with respect to the sensitivity of the whole

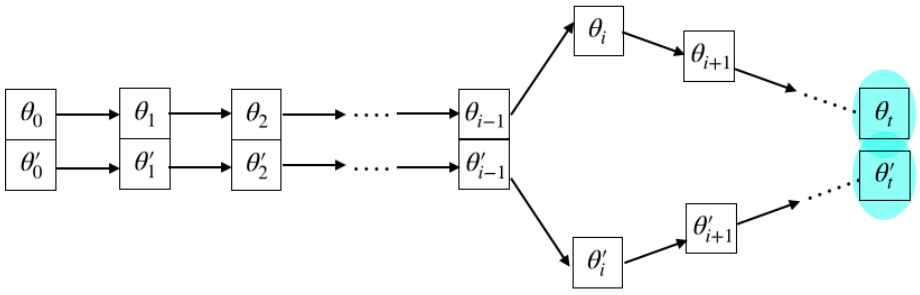

Figure 3: Output perturbation on a smooth, strongly convex objective. The top line represents the algorithm run on $\mathcal{D}$, the bottom one on $\mathcal{D}'$. The data point differing between $\mathcal{D}$ and $\mathcal{D}'$ is x_i encountered at step i. The two processes first diverge from each other, but then the contraction mapping brings them back close together. The noise (in blue) is only added at the last step.

procedure. This can be achieved for SGD on smooth, strongly convex objectives. Once the sensitivity is tightly upper-bounded, it is enough to add Gaussian noise with variance $\sigma^2 = \frac{\alpha s^2}{2\varepsilon^2}$ to the final output to get RDP (α, ε).

The main idea is the following (see figure 3). Let μ be the strong convexity parameter, β the smoothness parameter of $\ell(., x)$ for any x. Let the gradient step operator be:

$$G_x(u) := u - \gamma \nabla \ell(u, x)$$

For any x, G_x is a contraction mapping with parameter $\rho := \max\{|1 - \gamma\mu|, |1 - \gamma\beta|\}$:

$$\forall u, v \quad \|G_x(u) - G_x(v)\| \leq \rho \|u - v\|$$

If the learning rate is taken not too large, for instance $\gamma = 2/(\mu + \beta)$, then $\rho < 1$ and this is a strict contraction. As illustrated by figure 3, two runs of the same SGD algorithm on datasets D and D' will be first identical, until they reach the different data point at step i. Then they will move away from each other, but by no more than $2\gamma L$:

$$\sup_{x,y} \|G_x(u) - G_y(u)\| \leq 2\gamma \sup_x \|\nabla \ell(u, x)\| \leq 2\gamma L$$

After this step, we can use the property of contraction mapping to show that eventually the two iterates θ_t and θ'_t will move closer to each other. Hence the overall sensitivity of one epoch of SGD is way less than the sensitivity of one gradient step $2\gamma L$. This is true except for the last data point processed during the epoch. The sensitivity with respect to a change of this data point is precisely $2\gamma L$. The trick to get rid of this dependency on the order of processing is to shuffle the dataset at

the beginning. Then the overall privacy can be computed as a *softmax* average of the privacies with respect to each of the $n!$ possible orderings, thanks to the following lemma [8]:

Lemma 3.1 (Softmax averaging of Rényi divergences)**.** *Let $H_\alpha(p,q) := \exp((\alpha-1)D_\alpha(p,q))$. Let $\mathcal{M}_1, ..., \mathcal{M}_m$ be mechanisms and $q = [q_1, ..., q_m]$ be a probability vector over $\{1, ..., m\}$. Let $\mathcal{M}$ be an algorithm that on input D, samples $i \sim q$ and returns $\mathcal{M}_i(D)$. Then:*

$$H_\alpha(\mathcal{M}(D), \mathcal{M}(D')) \leq \sum_{j=1}^{m} q_j H_\alpha(\mathcal{M}_j(D), \mathcal{M}_j(D'))$$

3.2.3 Gradient Perturbation

The third method applies to SGD. The idea is to add Gaussian noise to each stochastic gradient used in the algorithm. Each step is now:

$$\theta_{t+1} = \theta_t - \gamma \left(\nabla \ell(\theta_t, x_{i(t)}) + \eta_t \right)$$

with $\eta_t \sim \mathcal{N}(0, \sigma^2 I_d)$. This is different from objective perturbation since all the η_t are sampled independently instead of being identical.

This process of adding noise to the gradients is sometimes signaled as related to the stochastic gradient Langevin dynamics [31], however for reasonable privacy parameters, the noise to add for SGD with DP has a much larger scaling, typically of the order of magnitude of L, the maximal norm of the gradients.

There are different ways to compute the privacy of SGD with gradient perturbation, depending on the assumptions on the objective function. The first one applies only to smooth, convex objective functions and relies on similar contraction arguments [13]. When the objective function is β-smooth and convex but not strongly convex, one can only show that the gradient step operator G_x is non expansive for $\gamma \leq 2/\beta$:

$$\forall u, v \quad \|G_x(u) - G_x(v)\| \leq \|u - v\|$$

Non expansiveness is not enough to get the contraction phenomenon as in figure 3. The phenomenon is described in figure 4. The analysis of [13] relies on finer arguments that study how the added noise is carried through the iterations, a phenomenon they call *amplification by iteration*. Instead of looking directly at Rényi divergences, they track the behavior of a noise *potential* quantified by the shifted Rényi divergence:

$$D_\alpha^{(z)}(\mu \| \nu) := \inf_{\mu' \ s.t. \ W_\infty(\mu, \mu') \leq z} D_\alpha(\mu' \| \nu)$$

where W_∞ is the ∞-Wasserstein distance, where $W_\infty(\mu, \mu') \leq z$ if and only if there exists random

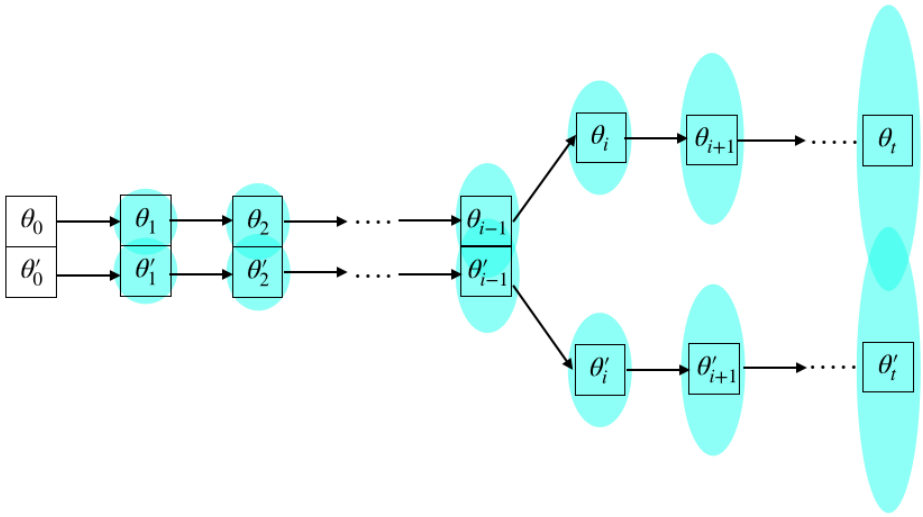

Figure 4: Gradient perturbation on a smooth, strongly convex objective. The two processes first diverge from each other, then the non expansive mapping ensures that they don't move further away from each other. The noise (in blue) is added at each step. It is *transported* through the iterations in the form of a *potential* that grows with the iterations, and is finally large enough to make the last iterates indistinguishable.

variables (U, V) such that $U \sim \mu$, $V \sim \mu'$ and $\mathbb{P}\left[\|U - V\| \leq z\right] = 1$.

The shifted Rényi divergence between the successive iterates *stores* the noise added at each steps and takes into account the effect of the diverging i-th step. It can be converted back to a proper Rényi divergence at the last step by picking $z = 0$. Once again, the obtained privacy guarantee depends on the ordering of the data points, and the last to be processed has weaker privacy. Lemma 3.1 is then not enough to find meaningful privacy guarantees when the dataset is shuffled. Assuming that $\sigma \geq L\sqrt{2(\alpha - 1)\alpha}$ (high noise regime), an asymptotically stronger version of the lemma can be used an gives privacy guarantees which are similar to the strongly convex case, up to constants and logarithmic factors.

When the training examples at each gradient step are sampled independently at random, similar privacy guarantees can be obtained without any smoothness or convexity assumption on the objective function. The proof uses only simple arguments and relies on composition and subsampling properties. It was originally derived by [1], who developed a set of techniques called *moment accountant*, which later motivated the introduction of Rényi differential privacy by [21].

The sketch of the privacy analysis is the following (see figure 5). Each gradient step is considered independently and the RDP composition theorem says that the RDP on one epoch of SGD (n gradient steps) is n times the RDP of one step. The sensitivity of each step is upper-bounded by $2\gamma L$. The obtained privacy can in fact be amplified by a factor $1/n^2$ using the fact that each SGD step only

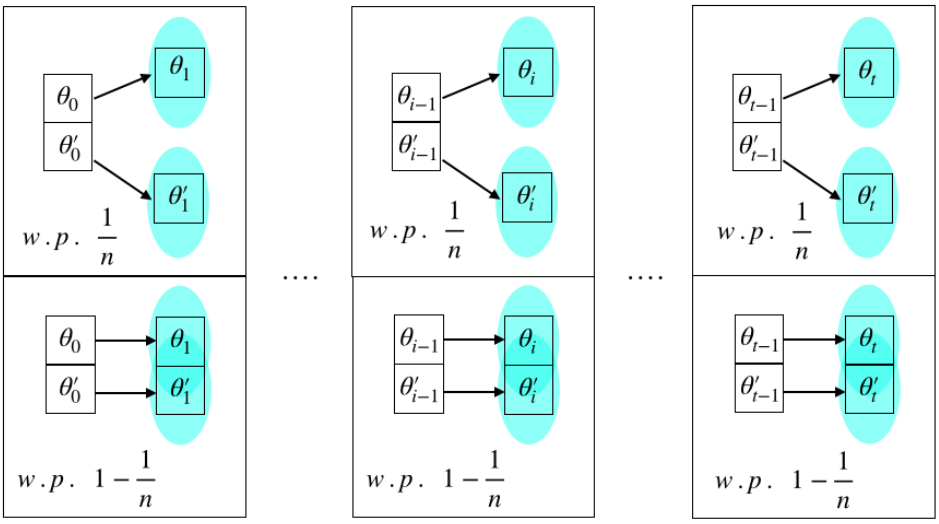

Figure 5: Gradient perturbation on any objective, with uniform sampling of stochastic gradients. The privacy of each gradient step is analyzed independently. Each step is a subsampled mechanism with subsampling parameter $\rho = 1/n$, yielding an amplification of RDP with factor $\rho^2 = 1/n^2$. The overall RDP after one epoch is the sum of the RDP of each of the n steps.

uses one stochastic gradients among the n possible ones. This phenomenon is called *amplification by subsampling* and has been precisely (and non-asymptotically) described by [30], with the following result:

Theorem 3.2 (RDP amplification by subsampling)**.** *Suppose that $\mathcal{A}$ is $(\alpha, \varepsilon(\alpha))$-RDP for any integer α when run on data $\mathcal{D}$. Let Sub be the algorithm that takes $\mathcal{D}$ as input and returns one data point of $\mathcal{D}$ sampled uniformly at random. Let $0 < \rho < 1$ be the subsampling parameter. Then the procedure $\mathcal{A} \circ$ Sub is $(\alpha, \varepsilon^\sharp(\alpha))$-RDP with:*

$$\varepsilon^\sharp(\alpha) \leq \frac{1}{\alpha - 1} \log \left(1 + 4\rho^2 \binom{\alpha}{2} (e^{\varepsilon(2)} - 1) + 2 \sum_{j=3}^{\alpha} \rho^j \binom{\alpha}{j} e^{(j-1)\varepsilon(j)} \right) .$$

In our case $\rho = 1/n$. For σ^2 large enough (high noise regime), $4(e^{\varepsilon(2)} - 1) \leq 8/\sigma^2$ and we get an asymptotic scaling (when $n \to \infty$) in $O\left(n \times \frac{1}{n^2\sigma^2}\right) = O\left(\frac{1}{n\sigma^2}\right)$.
Since it does not assume any properties on the objective function, this is probably the most widely used privacy analysis and it is the one currently implemented in TensorFlow [24].

3.2.4 Comparison of Privacy Guarantees

Most of the effort in the field of differentially private machine learning is focused on how to scale the injected noise to achieve some level of privacy. For a fair comparison, we consider a standard setting of one epoch of SGD with batch size one, that is n stochastic gradient steps. Following [29],

method	assumptions	Excess loss for (ε, δ) DP
objective perturbation	strong convexity	$O\left(\frac{d}{n^2\varepsilon^2}\right)$
output perturbation	shuffling, strong convexity	$O\left(\frac{d}{n^2\varepsilon^2}\right)$
gradient perturbation	shuffling, convexity	$O\left(\frac{\sqrt{d}}{n\varepsilon^2}\right)$
gradient perturbation	subsampling	$O\left(\frac{\sqrt{d}}{n\varepsilon}\right)$

Table 1: Asymptotic excess loss for (ε, δ) private algorithms, ignoring logarithmic dependencies in δ.

it is possible to convert the added noise into an excess population loss:

$$\mathbb{E}[L(\theta^{private}, D)] - \min_{\theta} L(\theta, D).$$

We give a glimpse of the relation between privacy ε and utility (excess loss) in table 1. A more systematic comparison of the different available methods can be found in [29].

3.3 Smooth Objective with Shuffling*

Here we are interested in extending the results of [13] to the smooth, non-convex setting. Only the last iterate is released (when using composition theorems, all of them are released). We sample the stochastic gradients in some permuted ordering. This does not allow a direct use of classical results on privacy amplification by subsampling. The idea is to study the evolution of the shifted version of Rényi divergence during SGD, to get at the end an upper-bound on the RDP. We aim at results matching those from [1].

The main idea is based on the fact that for smooth, non-convex objectives, the gradient step operator is C-lipschitz for $C = 1 + \gamma\beta$. If the (constant) learning rate is small enough, then we can still upper bound the deviation between the iterates on $\mathcal{D}$ and $\mathcal{D}'$ (see figure 6). The amplification of Gaussian noise on the iterates is not modified in the non-convex case.

3.3.1 Adaptation of Known Results

We have to adapt some of the results from [13]. This first lemma that quantifies the effect of adding noise (with a convolution) is not modified:

Lemma 3.3 (shift reduction lemma 20 [13])**.** *Let μ, ν and ζ be distributions over a Banach space. Then for any $a \geq 0$:*

$$D_\alpha^{(z)}(\mu * \zeta \| \nu * \zeta) \leq D_\alpha^{(z+a)}(\mu \| \nu) + R_\alpha(\zeta, a)$$

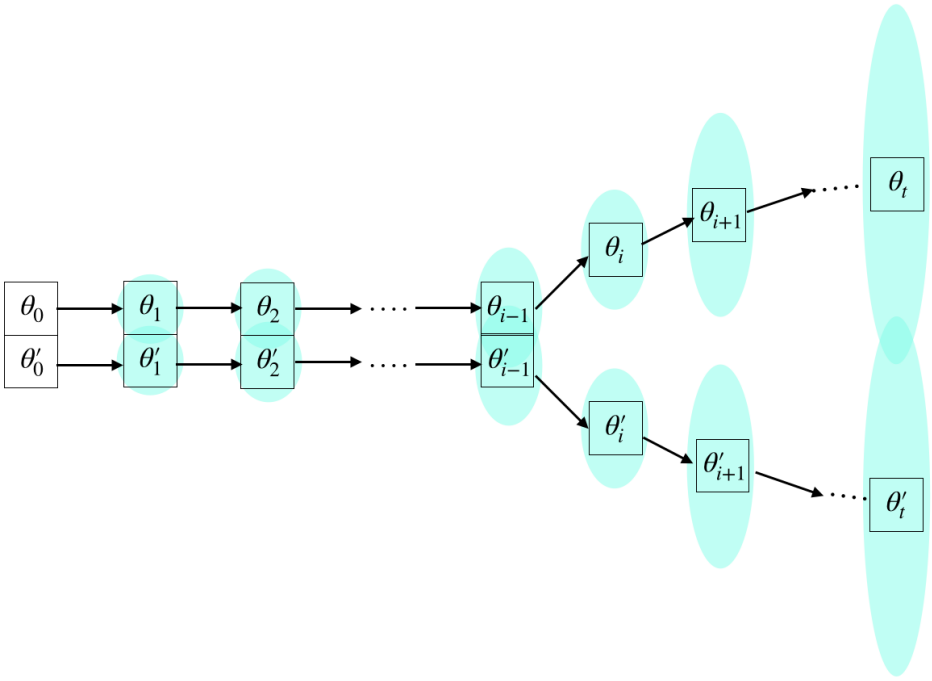

Figure 6: Gradient perturbation on a smooth, non-convex objective. The gradient step operator is still Lipschitz, and if the learning rate is small enough, the Lipschitz constant stays close to one.

In the case of Gaussian noise $\zeta = \mathcal{N}(0, \sigma^2 I_d)$, we have:

$$R_\alpha(\zeta, a) = \frac{\alpha a^2}{2\sigma^2}$$

The following lemma must be modified, since the gradient step is no longer a contraction (the proof is straightforward):

Lemma 3.4 (lemma 21 [13] smooth + non convex)**.** *Suppose ψ and ψ' are such that $\sup_x \|\psi(x) - \psi'(x)\| \leq s$, and that ψ (or ψ') is C-lipschitz, then for any random variables X and X':*

$$D_\alpha^{(Cz+s)}(\psi(X)||\psi'(X')) \leq D_\alpha^{(z)}(X||X')$$

In our case, a stochastic gradient step is $\psi(u) := u - \gamma \nabla \ell(u, x_i)$, it is $C = 1 + \gamma\beta$ Lipschitz as soon as the loss is β-smooth. Then we must change the main theorem (theorem 22) to fit with our setting.

Theorem 3.5 (theorem 22 [13] smooth + non convex)**.** *Let X_n and X'_n be the output of one epoch of noisy SGD with datasets $\mathcal{D}$ and $\mathcal{D}'$, and noise distributions $\zeta_1, ..., \zeta_n$. Let $s_1, ..., s_n$ be the successive sensitivities and $a_1, ..., a_n$ a sequence of free parameters. Let $z_0 = 0$ and $\forall t$, $z_{t+1} = Cz_t + s_{t+1} - a_{t+1}$ such that $\forall t$, $z_t \geq 0$. Then*

$$D_\alpha^{(z_n)}(X_n||X'_n) \leq \sum_{t=1}^{n} R_\alpha(\zeta_t, a_t)$$

In particular, if $z_n = 0$

$$D_\alpha(X_n||X'_n) \leq \sum_{t=1}^{n} R_\alpha(\zeta_t, a_t)$$

Proof. We show by recursion that

$$D_\alpha^{(z_t)}(X_t||X'_t) \leq \sum_{k=1}^{t} R_\alpha(\zeta_k, a_k)$$

This is true for $t = 0$ ($z_0 = 0$ and same initialization). Then with $Z_{t+1} \sim \zeta_{t+1}$:

$$\begin{aligned}
D_\alpha^{(z_{t+1})}(X_{t+1}||X'_{t+1}) &= D_\alpha^{(z_{t+1})}(\psi_{t+1}(X_t) + Z_{t+1}||\psi'_{t+1}(X'_t) + Z_{t+1}) \\
&\leq D_\alpha^{(z_{t+1}+a_{t+1})}(\psi_{t+1}(X_t)||\psi'_{t+1}(X'_t)) + R_\alpha(\zeta_{t+1}, a_{t+1}) \\
&= D_\alpha^{(Cz_t+s_{t+1})}(\psi_{t+1}(X_t)||\psi'_{t+1}(X'_t)) + R_\alpha(\zeta_{t+1}, a_{t+1}) \\
&\leq D_\alpha^{(z_t)}(X_t||X'_t) + R_\alpha(\zeta_{t+1}, a_{t+1}) \\
&\leq \sum_{k=1}^{t} R_\alpha(\zeta_k, a_k) + R_\alpha(\zeta_{t+1}, a_{t+1})
\end{aligned}$$

□

3.3.2 Privacy of One Epoch after Shuffling

Here we state a result similar to theorem 23 that applies in our case. This is a consequence of the previous theorem. For one particular ordering of the samples x_i, we have privacy:

Theorem 3.6 (theorem 23 [13] smooth + non convex)**.** *Consider one epoch of DP SGD where each step is:*

$$\theta_{t+1} = \theta_t - \gamma(\nabla \ell(\theta_t, x_t) + Z)$$

where $Z \sim \mathcal{N}(0, \sigma^2 I_d)$*. Then this algorithm is* $\left(\alpha, \frac{2\alpha L^2}{\sigma^2} \frac{1 - C^{-2}}{1 - (C^{-2})^{n-t+1}}\right)$*-RDP for its* t*-th data point. In particular, the points used earlier have stronger privacy guarantees.*

To combine the previous situations for $t = 1, ..., n$, we can use the result by [8], that we have already reproduced as lemma 3.1. Suppose $\mathcal{D}$ and $\mathcal{D}'$ differ only in their t-th element. We first make a permutation. There are $n!$ possible permutations of $\{1, ..., n\}$. Among these, the different entry is at any index i with equal probability $1/n$. Grouping them in the sum we get:

$$H_\alpha(M(D), M(D')) \leq \frac{1}{n} \sum_{t=1}^{n} H_\alpha(M_t(D), M_t(D'))$$

The Rényi divergence is not convex, but we get this relation:

$$D_\alpha(M(D)||M(D')) \leq \frac{1}{\alpha-1}\log\left(\frac{1}{n}\sum_{t=1}^{n}\exp\left((\alpha-1)D_\alpha(M_t(D)||M_t(D'))\right)\right)$$

This exponentials have a *softmax* effect that will make the weakest privacy guarantee (when point t is seen at the last step) dominate all the rest. Authors of [13] use instead a different version that directly applies to D_α if $\sigma \geq L\sqrt{2(\alpha-1)\alpha}$ (high noise regime) and almost gives convexity:

$$D_\alpha(M(D)||M(D')) \leq \frac{2}{n}\sum_{t=1}^{n} D_\alpha(M_t(D)||M_t(D'))$$

We still have to find a sharp upper bound on $D_\alpha(M_t(D)||M_t(D'))$ hence on:

$$\frac{2}{n}\sum_{t=1}^{n}\frac{2\alpha L^2}{\sigma^2}\frac{1-C^{-2}}{1-(C^{-2})^{n-t+1}} = \frac{4\alpha L^2}{\sigma^2 n}\sum_{t=1}^{n}\frac{1-C^{-2}}{1-(C^{-2})^{n-t+1}} = \frac{4\alpha L^2}{\sigma^2 n}\sum_{i=1}^{n}\frac{1-C^{-2}}{1-(C^{-2})^{i}}\,.$$

3.3.3 Small Learning Rate Regime

Our privacy guarantee gets better as C gets closer to one, as tipped in figure 6. To recover asymptotic rates which are similar to known results, C should go to one with n. C can only be related to n through the constant learning rate: $C = 1 + \beta\gamma$ and we can choose for instance $\gamma \sim \frac{1}{\beta\sqrt{n}}$ or $\gamma \sim \frac{1}{\beta n}$.

Proposition 3.1 (Privacy of one epoch for small learning rate)**.**

• *If the learning rate is set constant to $\gamma \sim \frac{1}{\beta\sqrt{n}}$, one epoch of SGD with shuffling has an asymptotic (when $n \to \infty$) RDP*

$$\left(\alpha, \frac{4\alpha L^2}{\sigma^2}\frac{\log(n)}{\sqrt{n}}\right)\,.$$

• *If the learning rate is set constant to $\gamma \sim \frac{1}{\beta n}$, one epoch of SGD with shuffling has an asymptotic RDP*

$$\left(\alpha, \frac{4\alpha L^2}{\sigma^2}\frac{\log(n)}{n}\right)\,.$$

The result for $\gamma \sim 1/n$ is asymptotically similar to [13], but we are making a very strong assumption on the learning rate. How unrealistic is it? This is quite far from Robbins and Monro's canonical choice of learning rate for SGD [25], but here we are looking at a very noisy version of SGD. In our experiments (see section 4), we had to choose very small learning rates for DP-SGD, otherwise the iterates just diverged. In a number of experiments, the learning rate we used was not too far from $1/\sqrt{n}$ or even $1/n$. It would be also interesting to look at the case of a decreasing learning rate

$\gamma_t \sim 1/t$ or $\gamma_t \sim 1/\sqrt{t}$.

Still these results are not very satisfactory because they depend too strongly on the choice of the learning rate, which is not the case for other classical analyses.

3.4 Noise Feedback with Affine Gradients*

3.4.1 Noise Feedback

One of the questions that was explored during this internship is: is there a way to (partially) cancel the noise that is added in the gradient perturbation scheme, in such a way that it maintains both convergence and privacy? This is inspired by recent work on gradient compression [18], that shows that it is possible to use *error feedback* to fix the effect of compressing the gradients used in SGD, with no price to pay in the convergence rates. We may see gradient perturbation as a random compression scheme, so that we would be able to use the same results. Obviously, since we recover convergence, we should not expect the final model to be private. We would still like to see if a reasonable trade-off between privacy and utility can be obtained using error feedback techniques.

Instead of simple gradient perturbation, we consider the following updates:

$$\begin{cases} \eta_0 = 0 \\ \forall t \geq 0,\ \eta_t \sim \mathcal{N}(0, \sigma^2 I_d) \\ \forall t \geq 0,\ \theta_{t+1} = \theta_t - \gamma \nabla \ell(\theta_t, x_t) - \eta_{t+1} + \eta_t \end{cases}$$

The idea is simply to remove at step t the noise that was added to the previous iterate at step $t-1$. The noise added at each step is now $\eta_{t+1} - \eta_t \sim \mathcal{N}(0, 2\sigma^2)$. However, conditioned on θ_t, this noise is not Gaussian anymore! It is not straightforward to describe its probability distribution in general as it depends on ℓ and the x_i.

That is why we will focus on a simple setting where it is possible to compute exactly the probability distribution of all the iterates θ_t.

3.4.2 Affine Gradients

We look at the particular case where the gradient of the individual loss function with respect to the parameters is affine. In particular this is the case of linear regression. This setting is interesting, because when we add Gaussian noise to the gradients at each step, then the iterates have a Gaussian distribution, hence we can explicitly compute the final DP.

We are in the setting of linear Gaussian graphical models for which inference can be performed

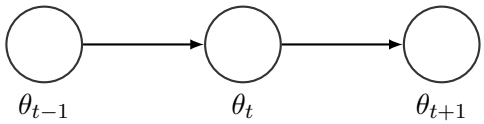

Figure 7: Graphical model of the iterates of DP SGD without noise feedback.

exactly. We consider the sampling scheme of stochastic gradients to be fixed. We also consider that we do only one pass on each data point (infinite amount of data, or one epoch). Unlike the composition theorem that gives privacy for the whole sequence of iterates, we only look at the distribution of the last iterate. The stochastic gradients are $g_t(\theta_t) := \alpha_t\theta_t + \beta_t$, with $\alpha_t \in \mathbb{R}^{d\times d}$, $\beta_t \in \mathbb{R}^d$ and $\alpha_t \succeq 0$ (SDP matrix). In the particular case of linear regression with feature vector x_t and target value y_t, we would have:

$$g_t(\theta_t) = x_t x_t^\top \theta_t - y_t x_t \,.$$

SGD with DP and no error feedback The first iterate $\theta_0 \in \mathbb{R}^d$ is fixed: $\theta_0 \sim \mathcal{N}(\theta_0, 0)$. At each step we add noise $\mathcal{N}(0, \Sigma)$ with $\Sigma = \sigma^2 I_d$ to the gradient. For any t, we have:

$$\theta_{t+1}|\theta_t \sim \mathcal{N}(\theta_t - \gamma g_t(\theta_t), \sigma^2 I_d)$$

We can show recursively that if $\theta_t \sim \mathcal{N}(m_t, \Sigma_t)$, then $\theta_{t+1} \sim \mathcal{N}(m_{t+1}, \Sigma_{t+1})$ and find the relation between successive parameters[2].

Proposition 3.2 (DP-SGD on Affine Gradients)**.** *For all $t \geq 0$, $\theta_t \sim \mathcal{N}(m_t, \Sigma_t)$ with the following recursion:*

$$\begin{cases} m_0 = \theta_0; \quad \Sigma_0 = 0 \\ m_{t+1} = (I_d - \gamma\alpha_t)m_t - \gamma\beta_t \\ \Sigma_{t+1} = \sigma^2 I_d + (I_d - \gamma\alpha_t)\Sigma_t(I_d - \gamma\alpha_t)^\top \end{cases}$$

For γ small enough (and $\|\alpha_t\|$ bounded), all the matrices are SDP, $(I_d - \gamma\alpha_t)^\top = (I_d - \gamma\alpha_t)$ so that:

$$\Sigma_{t+1} = \sigma^2 I_d + (I_d - \gamma\alpha_t)\Sigma_t(I_d - \gamma\alpha_t)$$

SGD with DP and error feedback Each iteration has the form:

$$\theta_{t+1} = \theta_t - \gamma(\alpha_t\theta_t + \beta_t) - \eta_{t+1} + \eta_t$$

[2]This is a direct application of formulas from http://user.it.uu.se/~thosc112/pubpdf/schonl2011.pdf

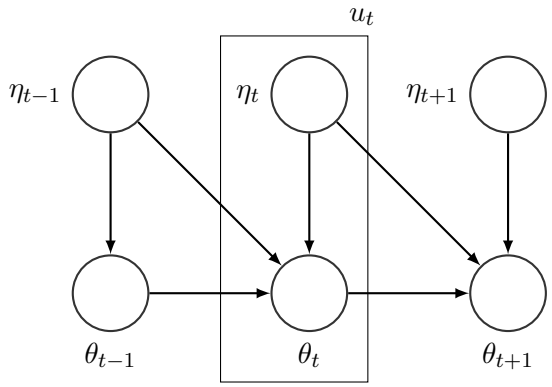

Figure 8: Graphical model of the iterates of DP SGD with noise feedback.

The model is now slightly more complicated. To make the recursion work, we first consider the random variable $u_t = (\eta_t, \theta_t)^\top$ and then take the marginal to keep only θ_t. The complete proof is in the appendix.

Proposition 3.3 (Noise feedback SGD with Affine Gradients)**.** *Under the same assumptions, for all* $t \geq 0$*,* $\theta_t \sim \mathcal{N}(m_t, \Sigma_t)$ *with the following recursion:*

$$\begin{cases} m_0 = \theta_0; \quad \Sigma_0 = 0 \\ m_{t+1} = (I_d - \gamma\alpha_t)m_t - \gamma\beta_t \\ \Sigma_{t+1} = 2\gamma^2\sigma^2\alpha_t^2 + (I_d - \gamma\alpha_t)\Sigma_t(I_d - \gamma\alpha_t) \end{cases}$$

3.4.3 Effect of Noise Feedback

The mean of θ_t with both methods is the same: $m_{t+1} = (I_d - \gamma\alpha_t)m_t - \gamma\beta_t$. However, the variances differ:

$$\Sigma_{t+1}^{noEF} = \sigma^2 I_d + (I_d - \gamma\alpha_t)\Sigma_t^{noEF}(I_d - \gamma\alpha_t)$$

$$\Sigma_{t+1}^{EF} = 2\gamma^2\sigma^2\alpha_t^2 + (I_d - \gamma\alpha_t)\Sigma_t^{EF}(I_d - \gamma\alpha_t)$$

Ideally we would like to have the largest possible variance, meaning that the iterates carry more noise, for a constant variance σ^2 added to the gradients. The first resulting variance is greater than $\sigma^2 I_d$ at all time steps, this is not true for the second one, assuming $\gamma\|\alpha_t\| < 1$. This means that most of the noise added at step t is in fact canceled afterwards. This was to be expected because the error feedback technique is supposed to reduce the noise, but then this also cancels most of the privacy. To keep the same level of privacy, one should add much more noise σ^2 to the gradients with error feedback than without.

4 Privacy of SGD with Shuffling*

4.1 Motivation

We are interested in analyzing the effect of the order in which data is processed on privacy. Consider one epoch of SGD with batch size one. At each step, one can independently pick uniformly at random a new training sample. We call this method *sampling with replacement*. Another option is to first make a permutation of the whole dataset, and then process all the samples according to this permuted ordering. This second method is known as *sampling without replacement* or as cyclic SGD or shuffling. It has been known empirically to convergence faster [4] and is hence more widely used in practice. It has been only recently proved [14] that sampling without replacement indeed enjoys faster convergences rates for strongly convex objectives after a reasonable number of epochs: $O(1/T^2)$ *vs* $O(1/T)$ for SGD with replacement.

While the privacy of the "sampling with replacement" method is well-known [1], it is unclear how usual proof techniques extend to the second approach. The only existing results must assume at least smoothness and convexity [13]. Recently, the effect of shuffling on pure ε differential privacy has been studied [12]. However, relaxations such as (ε, δ) differential privacy or Rényi differential privacy are more popular since they are able to obtain smaller privacy budgets.

We look at the RDP of SGD without replacement, with a particular focus on privacy guarantees that still hold in a non-convex setting. We **conjecture** privacy guarantees for sampling without replacement which are asymptotically similar to those for sampling with replacement, with the restriction that each epoch is stopped part-way through the dataset has been processed.

4.2 Algorithm

Given a dataset $\mathcal{D} = (x_1, ..., x_n) \in \mathbb{R}^{r \times n}$, we run e epochs of cyclic SGD with batch size one. Our algorithm takes as inputs the dataset, an initial parameter $\theta_0 \in \mathbb{R}^d$, a learning rate $\gamma > 0$, a loss function $\ell : \mathbb{R}^d \times \mathbb{R}^r \to \mathbb{R}$ and a gradient clipping parameter $L > 0$. Every epoch, we make a permutation S_i of $\mathcal{D}$, then return a sequence of parameters $(\theta_1, ..., \theta_t)$ computed after t clipped gradient steps with loss ℓ, processing the dataset in order. At each gradient step, Gaussian noise is added to the stochastic gradient, so that each gradient step (ignoring gradient clipping for readability) is:

$$\theta_{t+1} = \theta_t - \gamma \left(\nabla_\theta \ell(\theta_t, x_{s(t)}) + \eta_t \right) \quad \text{with } \eta_t \sim \mathcal{N}(0, \sigma^2 I_d) \, .$$

Importantly, algorithm 1 only processes $\tau \in (0, 1)$ fraction of the shuffled data before moving on to the next epoch—it does not perform a full cycle. The algorithm interpolates between sampling with

Algorithm 1 Private SGD with Shuffling

procedure PRIVATE SGD WITH SHUFFLING($\mathcal{D} = (x_i)_{i=1,\dots,n}, \theta_0, \gamma, \ell, L, e, \tau, \sigma^2$)
  $\theta \leftarrow \theta_0$
  for $i = 1$ to e/τ **do** ▷ e is the expected number of passes over each data point
    $(s_1, \dots, s_n) \leftarrow$ RandomPermutation($\{1, \dots, n\}$) ▷ Shuffle the data
    for $j = 1$ to $\lfloor \tau n \rfloor$ **do** ▷ Each pass over the dataset is stopped part-way
      $x \leftarrow x_{s_j}$
      $g \leftarrow$ *Clip*$(\nabla_\theta \ell(\theta, x), L)$
      Sample $\eta \sim \mathcal{N}(0, \sigma^2 I_d)$
      $\theta \leftarrow \theta - \gamma(g + \eta)$
  return θ ▷ Return the final model or alternatively the whole sequence of $\{\theta_t\}$

replacement ($\tau = 1/n$) and sampling with replacement ($\tau = 1$). We will not find any straightforward privacy guarantee for $\tau = 1$.

4.3 Conjecture on the Privacy

Conjecture 1 (Privacy of one step)**.** *At any epoch e and step t of Algorithm 1, let $\theta_{[1:t]} = \{\theta_1, \dots, \theta_t\}$ be the outputs. There exists $C \geq 16$ such that conditioned on $\theta_{[1:t]}$ and assuming $\sigma^2 \geq 4L^2$, the $t+1$-th step satisfies $(\alpha, \varepsilon_t(\alpha))$-RDP for*

$$\varepsilon_t(\alpha) \leq \frac{C\alpha L^2}{(n-t+1)^2\sigma^2} + \mathcal{O}\left(\frac{1}{(n-t+1)^3}\right).$$

The privacy bound degrades with t, and ultimately at $t = n$ no privacy is obtained. This is the main reason why we only process τ fraction of each epoch. The condition on σ^2 corresponds to the high noise/high privacy regime considered in [1]. **If conjecture 1 holds**, conjecture 2 follows and shows how to obtain the following global privacy guarantee on our procedure.

Conjecture 2 (Global privacy)**.** *There exists $C \geq 16$ such that the output sequence of parameters $\{\theta_t\}$ given by Algorithm 1 satisfies $(\alpha, \varepsilon(\alpha))$-Rényi differential privacy for*

$$\sigma^2 = \max\left(\frac{Ce\alpha L^2}{n(1-\tau)\varepsilon(\alpha)}, 4L^2\right) + \mathcal{O}\left(\frac{1}{n^2}\right).$$

Proof. We assume here that conjecture 1 holds. We mainly rely on (Proposition 1) from [21] which states that a composition of a (α, ε_1) and (α, ε_2) RDP procedures is $(\alpha, \varepsilon_1 + \varepsilon_2)$-RDP. We thus have

that assuming $\sigma^2 \geq 4L^2$, the privacy of Algorithm 1 is

$$\varepsilon(\alpha) \leq \sum_{i=1}^{\lfloor \frac{e}{\tau} \rfloor} \sum_{j=1}^{\lfloor n\tau \rfloor} \left\{ \frac{C\alpha L^2}{(n-j+1)^2\sigma^2} + \mathcal{O}\left(\frac{1}{(n-t+1)^3}\right) \right\}$$
$$\leq \frac{Ce\alpha L^2}{\tau\sigma^2} \left\{ \int_{(1-\tau)n}^{n} \frac{\mathrm{d}x}{x^2} + \mathcal{O}\left(\frac{1}{n^2}\right) \right\} \leq \frac{Ce\alpha L^2}{n(1-\tau)\sigma^2} + \mathcal{O}\left(\frac{1}{n^2}\right). \qquad \square$$

Our conjecture recovers the same asymptotic rate as for sampling with replacement [1]. We can easily translate the bounds into (ε, δ) differential privacy: Algorithm 1 has privacy $(\varepsilon(\alpha) + \log(1/\delta)/(\alpha - 1), \delta)$ for any $0 < \delta < 1$ [21]. Tighter (ε, δ) bounds can be obtained using a privacy accountant as in [1] that keeps track of the privacy loss at each step and uses the optimal α.

4.4 Partial Proof Overview

4.4.1 Privacy Amplification via Non-Uniform Subsampling

Here we prove a useful tool for later. Subsampling the dataset with ratio $\rho = 1/n$ before using an (α, ε)-RDP algorithm roughly increases its privacy to $(\alpha, \rho^2\varepsilon)$ [30] and this is used to compute the RDP of SGD with replacement. Suppose instead that we are subsampling with non uniform weights $(w_1, ..., w_n)$ [16]. These weights are supposed to be constant and not to depend on the processed data. In particular, they must be the same on datasets $\mathcal{D}$ and $\mathcal{D}'$ differing by one data point. We assume that they can be upper bounded $w_i \leq c$. Then we have the following generalization of a result from [30]:

Lemma 4.1 (Non uniform subsampling on ternary RDP)**.** *Let Sub be the algorithm that selects one index at random in $\{1, ..., n\}$ with probabilities $(w_1, ..., w_n)$. We assume that $\forall i, w_i \leq c$. The w_i cannot depend on the dataset processed. Let a mechanism M obey ζ-ternary-$|\chi|^\alpha$-DP, then the algorithm $M \circ Sub$ obeys $c\zeta$-ternary-$|\chi|^\alpha$-DP.*

Theorem 4.2 (Non uniform subsampling on RDP)**.** *Suppose that $\mathcal{A}$ is $(\alpha, \varepsilon(\alpha))$-RDP for any integer α. Let $0 < c < 1$ and $(w_1, ..., w_n)$ be such that $\forall i, 0 \leq w_i \leq c$, and $\sum_{i=1}^{n} w_i = 1$. Let $Sub(w_1, ..., w_n)$ be the algorithm that takes as input a dataset with n points and returns one of the points with non uniform probabilities $(w_1, ..., w_n)$. We assume that the w_i cannot depend on the dataset processed. Then $\mathcal{A} \circ Sub(w_1, ..., w_n)$ is $(\alpha, \varepsilon^{\#}(\alpha))$-RDP:*

$$\varepsilon^{\#}(\alpha) \leq \frac{1}{\alpha - 1} \log \left(1 + 4c^2 \binom{\alpha}{2} (e^{\varepsilon(2)} - 1) + 2 \sum_{j=1}^{\alpha} c^j \binom{\alpha}{j} e^{(j-1)\varepsilon(j)} \right)$$

Proof. This is a generalization of Theorem 9 from [30] and closely follows the same proof. The only difference being in Proposition 21 [30] (lemma 4.1 here) which states the effect of subsampling on the ternary DP, where we need to replace $1/n$ by c with an inequality. The full proof of lemma 4.1 is detailed in the appendix. □

4.4.2 Separating Two Sources of Leakage

We will assume for simplicity that Algorithm 1 runs only for one epoch with output $(\theta_1, ..., \theta_k) \in \mathbb{R}^{k\times d}$ for $k = \lfloor \tau n \rfloor$. Let $\theta_{[1:t]} := (\theta_1, ..., \theta_t)$ denote all the past iterates (in this epoch) until t. Conditioned on the previous iterates $\theta_{[1:t-1]}$, θ_t is a mixture of n Gaussians where each distribution corresponds to using a particular datapoint x. In the case of uniform sampling with replacement, this mixture has uniform weights $1/n$, which allows the direct use of subsampling theorems. However in the case of sampling without replacement, the sampling weights could possibly depend on the previous iterates $\theta_{[1:t-1]}$ since within an epoch each datapoint is sampled at most once.

There will be two different sources of privacy leakages:

- each iterate θ_t leaks information about the data point that is processed at step t;
- each iterate θ_t indirectly leaks information about the indices of the data points that will be processed at the next steps.

Recall that $(s_1, \dots, s_n)$ are the random variables indicating the shuffled indices. The following holds, for any $t \in \{1, ..., n\}$ and $\theta_{[1:t-1]}$:

$$p(\theta_t|\theta_{[1:t-1]}) = \sum_{k=1}^{n} p(\theta_t|\theta_{t-1}, s_t = k)p(s_t = k|\theta_{[1:t-1]})$$

where we used the fact that $\theta_t \perp \theta_{[1:t-2]}|\theta_{t-1}, s_t$. Here $p(\theta_t|\theta_{t-1}, s_t = k)$ is a Gaussian distribution with mean $\theta_{t-1} - \gamma\nabla_\theta \ell(\theta_{t-1}, x_k)$ and variance $\gamma^2\sigma^2 I_d$. Further, note that almost surely $p(s_t = k|\theta_{[1:t-1]}) \leq 1/(n-t+1)$. We **cannot directly use directly theorem 4.2** to bound the privacy of θ_t because the subsampling weights $p(s_t = k|\theta_{[1:t-1]})$ **are different between two datasets** $\mathcal{D}$ and $\mathcal{D}'$ that differ only by one data point. If we could apply theorem 4.2, then conjecture 1 would easily follow.

One possible idea to separate the two sources of privacy leakage is the following. We track the evolution of a *belief* about the sampling weights with this random variable:

$$\forall t \geq 1, k \in \{1, ..., n\}, \quad B_t^k := \Pr[s_t = k|\theta_{[1:t-1]}, B_{[1:t-1]}]$$

We have:

$$\begin{aligned} p(\theta_1, ..., \theta_t, B_1, ..., B_t) &= p(\theta_t, B_t|\theta_1, B_1, ..., \theta_{t-1}, B_{t-1}) \times ... \times p(\theta_1, B_1) \\ &= p(\theta_t|B_t, \theta_1, B_1, ..., \theta_{t-1}, B_{t-1}) \times p(B_t|\theta_1, B_1, ..., \theta_{t-1}, B_{t-1}) \\ &\quad \times ... \times p(\theta_1|B_1) \times p(B_1) \end{aligned}$$

The composition theorem for RDP says that the overall RDP of an algorithm releasing $\theta_1, ..., \theta_t, B_1, ..., B_t$ is less than the sum of the RDPs of all the conditional algorithms. In other words, the RDP of the terms in blue and in red can be studied separately.

We know that almost surely, $B_t^k \leq 1/(n-t+1)$. Conditioned on $\theta_{[1:t-1]}$ and $B_{[1:t]}$, then θ_t becomes indeed a mixture of n Gaussians, with **fixed sampling weights** $(B_t^k)_{k\in\{1,...,n\}}$ that do not depend on the dataset:

$$p(\theta_t|\theta_{[1:t-1]}, B_{[1:t]}) = \sum_{k=1}^{n} p(\theta_t|\theta_{t-1}, s_t = k) B_t^k \,.$$

Theorem 4.2 can then be applied directly to $\theta_t|\theta_{[1:t-1]}, B_{[1:t]}$ and gives the right amplification by subsampling for the blue terms. Then we get for each blue term an RDP of:

$$\varepsilon_t(\alpha) \leq \frac{16\alpha L^2}{(n-t+1)^2\sigma^2} + \mathcal{O}\left(\frac{1}{(n-t+1)^3}\right) .$$

The remaining terms are the red ones $p(B_t|\theta_1, B_1, ..., \theta_{t-1}, B_{t-1})$. It is not straightforward to see how to upper-bound their RDP. It somehow quantifies how much the belief about the sampling weights can be affected when seeing a new iterate of the algorithm. We would hope to find an upper-bound on their RDP which is asymptotically the same as the one of the blue terms: $\mathcal{O}\left(\frac{1}{(n-t+1)^2\sigma^2}\right)$, so that the overall privacy is not affected, up to constants. This, we have **not achieved** yet. It may also require additional assumptions.

4.5 Shuffling can be Less Private

One might wonder why all privacy guarantees seem to be weaker for sampling without replacement than for sampling with replacement, while both would be expected to have similar behaviors. Here we present a toy example where DP depends on the type of sampling. θ is initialized at 0 and each training example corresponds either to the *non-convex* loss function ℓ_1 or to ℓ_2 (see figure 9). The algorithm does two SGD steps with step-size 1, then with probability p returns the final θ, otherwise chooses a response uniformly among $\{-2, 0, 2\}$ (*output perturbation*). With $p = 1/2$, sampling with replacement has $\log(1.75)$-DP, while sampling without replacement only has $\log(4)$-DP.

Intuitively, the example is built such that choosing twice the same data point preserves more privacy

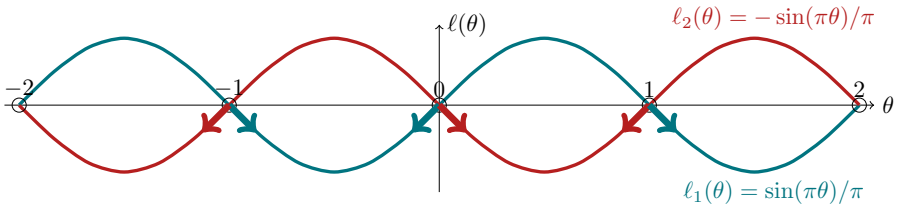

Figure 9: The two possible loss functions, depending on the training example.

(one cannot tell between (ℓ_1, ℓ_1) and (ℓ_2, ℓ_2)) than choosing different ones ((ℓ_1, ℓ_2) and (ℓ_2, ℓ_1) end up in different states). This is a very simple example of randomized response where pure DP can be computed exactly, but the same principle can be applied to SGD with gradient perturbation, albeit with approximate computations. Besides, it does not strongly rely on non convexity and similar examples can be built with quadratic losses. Finally, the privacy shift between the two situations is not related to the disclosure of the full sequence of iterates.

4.6 Experiments

SGD without replacement is known to converge faster than SGD with replacement, on strongly convex objectives. This is well established, both empirically and now theoretically. However, this is true only in a non-private setting. We would like to check empirically if this still holds with DP-SGD, where the added noise has the same magnitude as the stochastic gradients: $\sigma \geq 2L$, where L is the gradient clipping parameter.

4.6.1 Setting

We consider a simple two-class classification task on the dataset `epsilon` from `libsvm`[3]. For practical reasons, we only use the first $n = 1,000$ data points (x_i, y_i) of the dataset (out of 400k). The dimension of the features x_i is $d = 2,000$.

We use a logistic regression, with an ℓ^2 regularization to ensure strong convexity. The minimization problem is:

$$\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\Big\{y_i \log(\sigma(\theta^\top x_i)) + (1-y_i)\log(\sigma(-\theta^\top x_i))\Big\} + \frac{1}{n}\|\theta\|^2$$

We solve this problem with SGD, successively without and with gradient perturbation. Using the parameters of algorithm 1, we consider different values of $\tau \in (0, 1]$. We track the *training loss* depending on the number of training iterations, and more precisely the suboptimality compared to the minimal loss obtained by SGD run on 10 times more epochs. Each experiment is repeated 10 times and the plots show the mean $\pm$ one standard deviation.

[3] available here: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

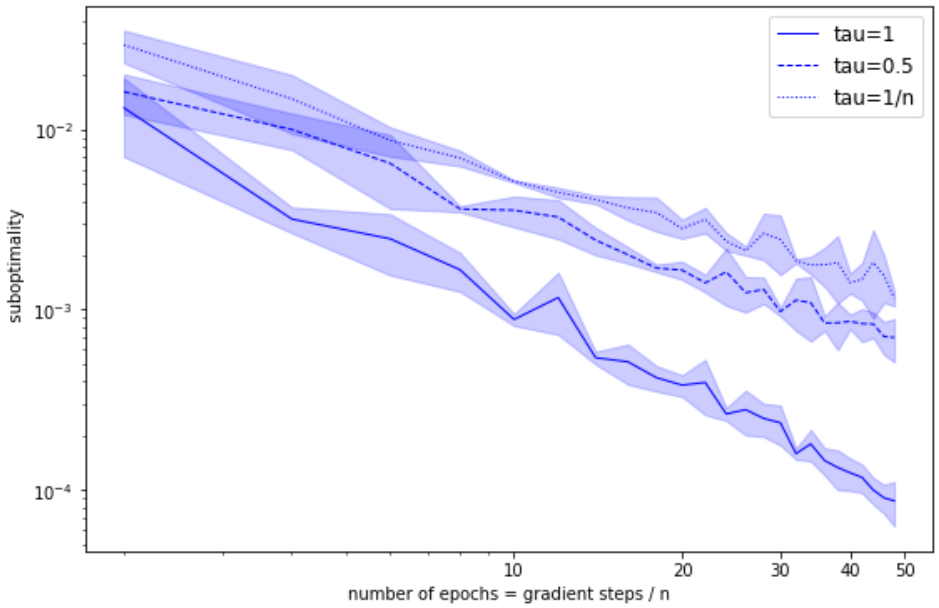

Figure 10: Training loss of non-private SGD for different τ (log-log scale).

For DP-SGD, the privacy is tracked with the privacy accountant defined in conjecture 1, using the most optimistic version ($C = 16$). For sampling with replacement, corresponding to $\tau = 1/n$, this is exactly the same accountant as the one proved by [1]. We don't use $\tau = 1$ for DP-SGD as we have no efficient method to account for the privacy. The final privacy result is given in term of (ε, δ) differential privacy, for $\delta = 10^{-4} \ll 1/n = 10^{-3}$.

4.6.2 Results

For non-private SGD (see figure 10), there is indeed a different between values of τ. After a reasonable number of epochs, sampling without replacement ($\tau = 1$) is one order of magnitude better than sampling with replacement ($\tau = 1/n$). This is similar to the experiment of [4] on the larger dataset `rcv1`. The training losses for intermediary values of τ interpolate between the two.

For DP-SGD, we see no difference of loss between values of τ (see figure 11). Note that the scale of the loss is very different in this experiment, and the suboptimality is not very different from at initialization (taken constant to $\theta_0 = 0$ across all experiments). This is because the learning rate must be chosen very small, otherwise the loss quickly diverges. In a limited number of epochs, very little progress can be achieved in terms of loss. Importantly, the privacy guarantees are independent from the learning rate, hence we suggest to pick the largest possible learning rate avoiding divergence. On this classification task, after 50 epochs the non private models achieved more than 95% training accuracy, *vs* around 60% for the private ones.

The privacy loss is accounted on figure 12. The dotted line corresponds to an already known accoun-

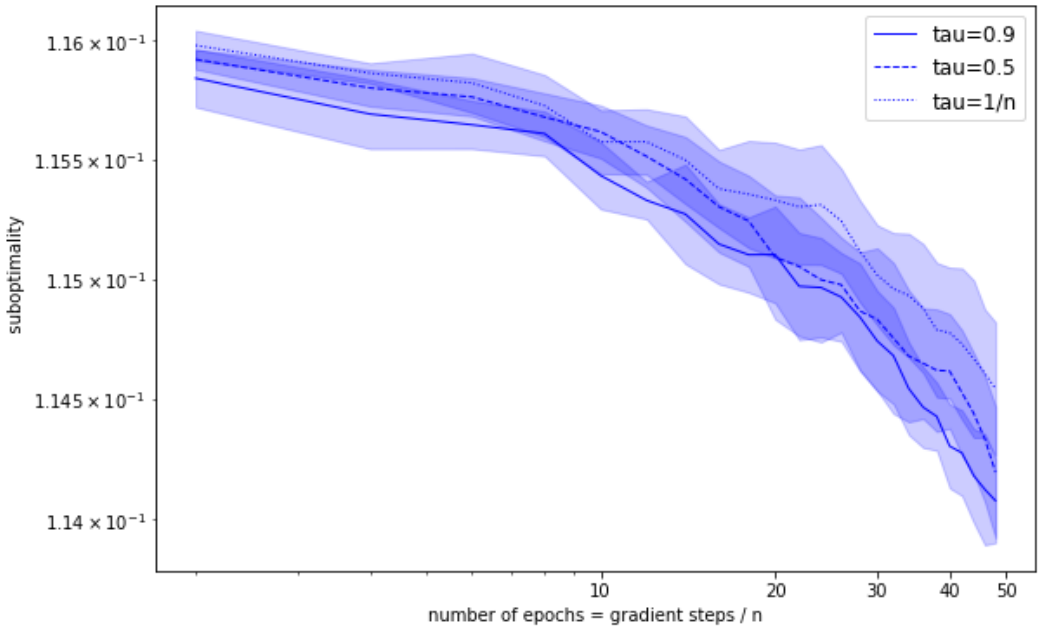

Figure 11: Training loss of DP-SGD for different τ.

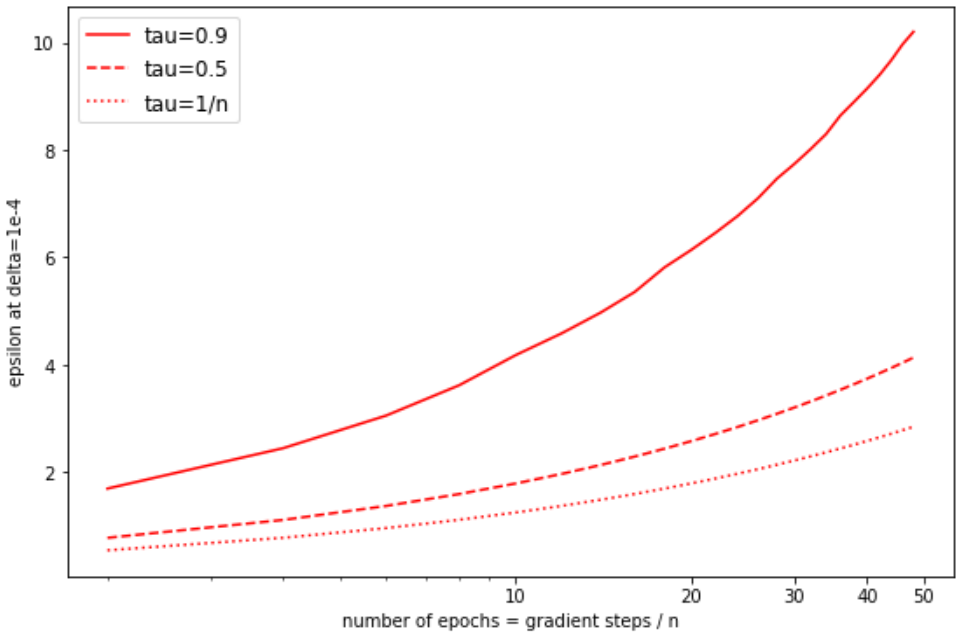

Figure 12: Privacy ε at $\delta = 10^{-4}$ of DP-SGD for different τ, using conjecture 1.

tant, the other two should be considered only conjectures. If conjecture 1 holds, then they are only a constant factor away ($1/(1-\tau)$). It should be noticed that ε roughly grows linearly with the number of epochs, forbidding the use of too many epochs for DP-SGD. This is a very intrinsic limitation that prevents from going *to optimality* while preserving *privacy*.

4.6.3 Discussion

How come we do not see any difference between sampling with and without replacement for DP-SGD?

A first possible explanation would be that the choice of the learning rate imposes a learning regime which is too far from convergence. If we repeat the same experiment, for non-private SGD with learning rates similar to DP-SGD, then we do not see either any difference between the two sampling methods. It may be possible to detect a difference after a much larger number of epochs e, but unfortunately this is not possible if we want to keep meaningful privacy guarantees since ε grows linearly with e.

Another explanation could be the following. Intuitively, SGD without replacement works better on strongly convex objectives because one epoch approximates a full gradient step. For DP-SGD with replacement, at each epoch some new noise is added to the stochastic gradient of the same data point. It suggests that several consecutive epochs cannot be considered as full gradient steps on the same dataset. Imagine the step size is 0. Then without noise, the sum of stochastic gradients after one epoch correspond exactly to the full gradient, and this will be the same after several epochs. For DP-SGD with step size 0, after one epoch the sum of the gradients is a noisy version of the full gradient, and at the next epoch it will be different noisy version, as if it was computed on a different dataset.

With DP-SGD we are in fact solving the following problem with stochastic optimization:

$$\min_{\theta} \mathbb{E}_{\substack{\eta\sim\mathcal{N}(0,\sigma^2)\\ i\sim U(1,\dots,n)}} \left[\ell(\theta, x_i) + \theta^\top \eta\right]$$

At each step, η and i are sampled according to their probability distributions and one step is performed with stochastic gradient $g_t := \nabla\ell(\theta, x_i) + \eta$. In this case, the objective function is no longer a finite sum, and distinguishing sampling with or without replacement does not make sense.

5 Beyond Differential Privacy

5.1 Is it ok to Relax?

Several DP techniques now match the lower bounds on machine learning problems. Yet practical results are still very unsatisfactory:

- either DP causes a big drop in accuracy: from 95% to 65% on CIFAR-10 after careful hyperparameter tuning [1];
- either the privacy guarantees become meaningless: $\varepsilon = 10$ means that there is a ratio of e^{10} between the probability densities of the output under $\mathcal{D}$ and $\mathcal{D}'$.

In fact, even the initial relaxation from ε to (ε, δ) DP has been criticized for providing weaker privacy guarantees, and certainly less interpretable ones. The exact effect of ε and δ is not simple to explain, since it essentially focuses in worst-case analyses. Most of the provided upper-bounds are probably far from what happens in real-life applications.

Recently, Apple has been accused of pretending to use DP while the (non-disclosed) privacy parameters they were using were in fact huge ($\varepsilon \gg 10$) [28]. From the point of view of the user, being ensured $\varepsilon = 10$ DP or no DP at all is not very different. This is supported by practical experiments showing that membership attacks can still be performed successfully on DP model, if the privacy parameters are too large [17]. Obviously the original notion of DP was meant to be used with a small privacy parameter ε.

5.2 Relation with Stability

In machine learning, the definition of uniform stability is quite close to the one of differential privacy:

Definition 5.1 (Uniform stability)**.** *A randomized algorithm $\mathcal{A}$ is ε-uniformly stable if for all datasets $\mathcal{D}$, $\mathcal{D}'$ that differ in at most one example, we have:*

$$\sup_z \mathbb{E}_{\mathcal{A}} \left[\ell(\mathcal{A}(\mathcal{D}), z) - \ell(\mathcal{A}(\mathcal{D}'), z) \right] \leq \varepsilon \,.$$

This closely corresponds to DP applied to the loss function $z \mapsto \ell(\theta^*, z)$ (black-box model). The only difference is that the statistical deviation is measured by an expectation over the randomness of $\mathcal{A}$. This is not a *worst-case* analysis. The definition was introduced by Bousquet & Elisseeff [5] and is used to provide *in expectation* generalization guarantees (see theorem 2.2 in [15]). If an algorithm is ε-uniformly stable, then its generalization error is smaller than ε.

ε-differential privacy implies ε-uniform stability. The converse is not true. It could be interesting to consider ε-uniform stability as a *weak privacy* guarantee, since it matches the intuition that a private

model should not overfit and generalize well. This weak guarantee has a better behavior than DP in some cases and can be ensured for instance by limiting the number of passes over the data (*train faster, generalize better* [15]). Such analyses are very similar to the ones for DP, e.g. [8]. Importantly, stability guarantees can be achieved *without noise addition*. However it is unclear if uniform stability is a useful protection against membership or reconstruction attacks. Yet ε-uniform stability with a small ε could still be better than ε-DP with a large ε.

5.3 Robustness to Attacks

Perhaps the most important weakness of DP is that it is impossible to measure in practice. Given a model, and without access to how it was built from a dataset, it is not possible to estimate its DP. It is only a constructive guarantee that is built from the training algorithm $\mathcal{A} : \mathcal{D} \mapsto \theta^*$. Depending on the threat model, it might be difficult to assert that a malicious training algorithm indeed follows a DP protocol.

A more practical notion of privacy could be robustness to some membership or reconstruction attacks. There is a line of research looking at how to measure robustness towards specific attacks and how to develop counter measures [23]. This enables to consider more realistic adversarial setting, e.g. when the adversary has a limited computing power, or a limited access to inference queries from the model. Of course, these are much weaker privacy guarantees as they only depend on specific attack schemes, as compared to DP which is an information theoretic guarantee.

5.4 Poor Man's Privacy in Distributed Settings*

In some cases, we do not care so much about the privacy leakage caused by the final trained model because it will not be publicly release, or because this is too constraining. We could nonetheless require that the *training procedure* should not reveal more than what the final model reveals. We call this notion the poor man's privacy.

This could prove useful in distributed settings, where the training data are spread over different nodes. The nodes agree to jointly train a model, but they do not want their data to be revealed to the other nodes or to a central authority. This corresponds to federated learning [20, 27] or decentralized optimization [19], where the data stay *on device*.

Ensuring privacy in such contexts is part of the broader framework of secure multi-party computation, which can be achieved with cryptographic methods. One of them is fully homomorphic encryption [3] that enables simple computations on encrypted data, but it is excessively slow on large-scale problems.

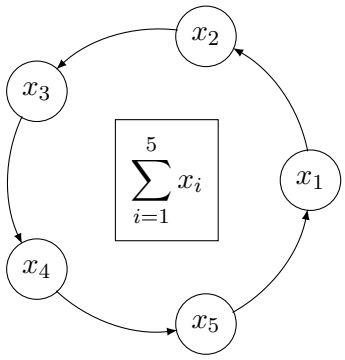

Figure 13: Each node holds a data point x_i. They collaborate to compute the sum without revealing their individual data.

It could be interesting to describe lighter methods inspired by DP and based on the addition of noise. Consider the following problem. $n \geq 3$ nodes, each one holding a data point $x_i \in \mathbb{R}^d$, want to collaborate to compute the sum S. All of them have access to the final result S, but each node should learn no more than this about the data of the other nodes (see figure 13). We assume that no collusion is possible between the nodes. They must share their value and compute the result in a private way, which can be achieved by addition and subtraction of arbitrarily large noise. If the network's topology is a ring, techniques like the *ring all-reduce* algorithm [26] can be extended to this private setting. Such an algorithm ensures both communication efficiency and privacy. Future research directions include the extension of such algorithms to arbitrary topologies and to iterative algorithms, the computation of the sum being applied to recover a full gradient from stochastic gradients.

Conclusion

We have explored the use of differential privacy for privacy preserving machine learning. It is a relatively recent field which has made rapid progress over the last five years. In particular, the analysis of differentially private SGD has focused a lot of attention. Yet the analysis of DP-SGD with sampling without replacement is still an open problem. We only provide a conjecture about the privacy guarantees that could be foreseen. Our experiments show that one should not expect huge improvements in terms of convergence speed compared to sampling with replacement, in private settings.

Although the mechanisms to create differential privacy are now well-understood, it unclear if it is the right notion for machine learning. In particular, the intuitive relation between privacy and overfitting is still not obvious, and privacy has not yet been fully included in optimization processes, e.g. in the form of a constraint. This could possibly be achieved with weaker privacy guarantees, yet more adapted to learning procedures. Important applications of privacy include distributed machine learning. This is not fully covered by differential privacy and paves the way to new privacy definitions and new private protocols. One should not fear to use relaxed privacy guarantees, as soon as they are well-understood.

References

[1] 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*, pages 308–318. ACM, 2016.

[2] R. F. Barber and J. C. Duchi. Privacy and statistical risk: Formalisms and minimax bounds. *arXiv preprint arXiv:1412.4451*, 2014.

[3] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[4] L. Bottou. Curiously fast convergence of some stochastic gradient descent algorithms. In *Proceedings of the symposium on learning and data science, Paris*, 2009.

[5] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[6] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv preprint arXiv:1802.08232*, 2018.

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

[8] C. Chen, J. Lee, and D. Kifer. Renyi differentially private erm for smooth objectives. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2037–2046, 2019.

[9] L. Devroye, A. Mehrabian, and T. Reddad. The total variation distance between high-dimensional gaussians. *arXiv preprint arXiv:1810.08693*, 2018.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

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

[12] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[13] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 521–532. IEEE, 2018.

[14] J. Z. HaoChen and S. Sra. Random shuffling beats sgd after finite epochs. In *International Conference of Machine Learning*, 2019.

[15] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

[16] H. Hartley, J. Rao, et al. Sampling with unequal probabilities and without replacement. *The Annals of Mathematical Statistics*, 33(2):350–374, 1962.

[17] B. Jayaraman and D. Evans. When relaxations go bad:" differentially-private" machine learning. *arXiv preprint arXiv:1902.08874*, 2019.

[18] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261, 2019.

[19] A. Koloskova, S. U. Stich, and M. Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

[20] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

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

[22] J. Murtagh and S. Vadhan. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*, pages 157–175. Springer, 2016.

[23] M. Nasr, R. Shokri, and A. Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 634–646. ACM, 2018.

[24] N. Papernot. Machine learning with differential privacy in tensorflow - cleverhans-blog. http://www.cleverhans.io/privacy/2019/03/26/machine-learning-with-differential-privacy-in-tensorflow.html, Mar 26, 2019. Accessed: 2019-07-23.

[25] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[26] A. Sergeev and M. Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

[27] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18:230, 2018.

[28] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. *arXiv preprint arXiv:1709.02753*, 2017.

[29] D. Wang, M. Ye, and J. Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, pages 2722–2731, 2017.

[30] Y.-X. Wang, B. Balle, and S. Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. *arXiv preprint arXiv:1808.00087*, 2018.

[31] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

A Missing Proofs

A.1 Proof of Proposition 2.3

Proof. We first consider a deterministic function f. Let $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x-y\|_1 \leq 1$ and $S \subset R'$:

$$\big|\mathbb{P}[f(\mathcal{A}(x)) \in S] - \mathbb{P}[f(\mathcal{A}(y)) \in S]\big| = \big|\mathbb{P}[\mathcal{A}(x) \in f^{-1}(S)] - \mathbb{P}[\mathcal{A}(y) \in f^{-1}(S)]\big| \leq \delta$$

because $f^{-1}(S) \subset \mathcal{R}$ and $\mathcal{A}$ is δ TV-private.
When f is a random variable over some space of functions, then we have:

$$\mathbb{P}[f(\mathcal{A}(x) \in S] = \mathbb{E}\big[\mathbb{P}[f(\mathcal{A}(x) \in S \mid f]\big]$$

Hence:

$$\begin{aligned}
\big|\mathbb{P}[f(\mathcal{A}(x)) \in S] - \mathbb{P}[f(\mathcal{A}(y)) \in S]\big| &= \big|\mathbb{E}\big[\mathbb{P}[f(\mathcal{A}(x)) \in S \mid f]\big] - \mathbb{E}\big[\mathbb{P}[f(\mathcal{A}(y)) \in S \mid f]\big]\big| \\
&= \big|\mathbb{E}\big[\mathbb{P}[f(\mathcal{A}(x)) \in S \mid f] - \mathbb{P}[f(\mathcal{A}(y)) \in S \mid f]\big]\big| \\
&\leq \mathbb{E}\big[\big|\mathbb{P}[f(\mathcal{A}(x)) \in S \mid f] - \mathbb{P}[f(\mathcal{A}(y)) \in S \mid f]\big|\big] \\
&\leq \delta
\end{aligned}$$

where we have used Jensen's inequality and the result for a deterministic function. □

A.2 Proof of Theorem 2.2

Proof. We first make the proof when the algorithms are run independently. Let $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\| \leq 1$. Let $(S_1, S_2) \subset \mathcal{R}_1 \times \mathcal{R}_2$. Then:

$$\begin{aligned}
&\big|\mathbb{P}[\mathcal{A}_{1,2}(x) \in (S_1, S_2)] - \mathbb{P}[\mathcal{A}_{1,2}(y) \in (S_1, S_2)]\big| \\
&\qquad = \big|\mathbb{P}[\mathcal{A}_1(x) \in S_1]\mathbb{P}[\mathcal{A}_2(x) \in S_2] - \mathbb{P}[\mathcal{A}_1(y) \in S_1]\mathbb{P}[\mathcal{A}_2(y) \in S_2]\big| \\
&\qquad = \big|\mathbb{P}[\mathcal{A}_1(x) \in S_1]\big(\mathbb{P}[\mathcal{A}_2(x) \in S_2] - \mathbb{P}[\mathcal{A}_2(y) \in S_2]\big) \\
&\qquad\qquad + \big(\mathbb{P}[\mathcal{A}_1(x) \in S_1] - \mathbb{P}[\mathcal{A}_1(y) \in S_1]\big)\mathbb{P}[\mathcal{A}_2(y) \in S_2]\big| \\
&\qquad \leq \big|\mathbb{P}[\mathcal{A}_1(x) \in S_1]\big(\mathbb{P}[\mathcal{A}_2(x) \in S_2] - \mathbb{P}[\mathcal{A}_2(y) \in S_2]\big)\big| \\
&\qquad\qquad + \big|\big(\mathbb{P}[\mathcal{A}_1(x) \in S_1] - \mathbb{P}[\mathcal{A}_1(y) \in S_1]\big)\mathbb{P}[\mathcal{A}_2(y) \in S_2]\big| \\
&\qquad \leq 1 \times \delta_2 + \delta_1 \times 1 \\
&\qquad = \delta_1 + \delta_2
\end{aligned}$$

When the algorithms are run sequentially and each algorithm takes as input the outputs of previous algorithms, the proof remains valid. We just have to use:

$$\mathbb{P}[\mathcal{A}_{1,2}(x) \in (ds_1, S_2)] = \mathbb{P}[\mathcal{A}_2(x, \mathcal{A}_1(x)) \in S_2 \mid \mathcal{A}_1(x) \in ds_1] \, \mathbb{P}[\mathcal{A}_1(x) \in ds_1]$$

□

A.3 Proof of Proposition 2.4

Proof. We first consider the case $d = 1$. Let $x \in \mathbb{N}^{|\mathcal{X}|}$ and $y \in \mathbb{N}^{|\mathcal{X}|}$ with $\|x - y\|_1 \leq 1$. Let p_x and p_y be the probability densities of respectively $\mathcal{M}_L(x, f, \delta)$ and $\mathcal{M}_L(y, f, \delta)$. For any $z \in \mathbb{R}$:

$$|p_x(z) - p_y(z)| = \frac{1}{2b}\Big| \exp\big(-\frac{|f(x) - z|}{b}\big) - \exp\big(-\frac{|f(y) - z|}{b}\big)\Big|$$

Since $u \mapsto e^{-u}$ is 1-Lipschitz on $\mathbb{R}_+$, we have:

$$\begin{aligned}
|p_x(z) - p_y(z)| &\leq \frac{1}{2b}\Big|\frac{|f(x) - z|}{b} - \frac{|f(y) - z|}{b}\Big| \\
&\leq \frac{1}{2b^2}\Big||f(x) - z| - |f(y) - z|\Big| \\
&\leq \frac{1}{2b^2}|f(x) - f(y)| \leq \delta
\end{aligned}$$

Now for dimension $d \geq 2$. We use a recursion. Suppose for all $d' < d$ and $z \in \mathbb{R}^{d'}$, we have $|p_x(z) - p_y(z)| \leq \frac{1}{2b^2}\|f(x) - f(y)\|_{\mathbb{R}^{d'}}$. Let $z \in \mathbb{R}^d$. Since the Y_i are independent,

$$p_x(z) = \prod_{i=1}^{d} \exp\big(-\frac{|f(x)_i - z_i|}{b}\big)$$

so we can move $\exp\big(-\frac{|f(x)_d - z_d|}{b}\big)$ out and we get:

$$\begin{aligned}
|p_x(z) - p_y(z)| &= \frac{1}{2b}\Big|\prod_{i=1}^{d} \exp\big(-\frac{|f(x)_i - z_i|}{b}\big) - \prod_{i=1}^{d} \exp\big(-\frac{|f(y)_i - z_i|}{b}\big)\Big| \\
&= \frac{1}{2b}\Big| \exp\big(-\frac{|f(x)_d - z_d|}{b}\big) \prod_{i=1}^{d-1} \exp\big(-\frac{|f(x)_i - z_i|}{b}\big) \\
&\qquad - \exp\big(-\frac{|f(y)_d - z_d|}{b}\big) \prod_{i=1}^{d-1} \exp\big(-\frac{|f(y)_i - z_i|}{b}\big)\Big|
\end{aligned}$$

Using the same technique as in the proof of theorem 2.2, we get:

$$|p_x(z) - p_y(z)| \leq \frac{1}{2b}\Big|\exp\big(\frac{-|f(x)_d - z_d|}{b}\big)\Big(\prod_{i=1}^{d-1}\exp\big(\frac{-|f(x)_i - z_i|}{b}\big) - \prod_{i=1}^{d-1}\exp\big(\frac{-|f(y)_i - z_i|}{b}\big)\Big)\Big| + \frac{1}{2b}\Big|\Big(\exp\big(\frac{-|f(x)_d - z_d|}{b}\big) - \exp\big(\frac{-|f(y)_d - z_d|}{b}\big)\Big)\prod_{i=1}^{d-1}\exp\big(\frac{-|f(y)_i - z_i|}{b}\big)\Big|$$

Using the hypothesis in dimension 1 and $d-1$, we get:

$$\begin{aligned}|p_x(z) - p_y(z)| &\leq 1 \times \frac{1}{2b^2}\|f(x) - f(y)\|_{\mathbb{R}^{d-1}} + \frac{1}{2b^2}|f(x)_d - f(y)_d| \times 1 \\ &= \frac{1}{2b^2}\|f(x) - f(y)\|_{\mathbb{R}^d} \leq \frac{\Delta f}{2b^2} \leq \delta\end{aligned}$$

where $\|f(x) - f(y)\|_{\mathbb{R}^{d-1}}$ means that we only consider the first $d-1$ components of $f(x) - f(y)$. We have proved that the Laplace mechanism in any dimension d, with parameter $b = \sqrt{\frac{\Delta f}{2\delta}}$ is δ TV-private. □

A.4 Proof of Theorem 3.6

Proof. The proof outline is the same as for the original theorem. Let $S = (x_1, ..., x_n)$ and $S' = (x_1, ..., x_{t-1}, x'_t, x_{t+1}, ..., x_n)$ two datasets that differ only at entry t. The ζ_k are all noise distributions $\mathcal{N}(0, (\gamma\sigma)^2 I_d)$. Hence $R_\alpha(\zeta_k, a) = \frac{\alpha a^2}{2\sigma^2\gamma^2}$. We have (bounded gradients):

$$\sup_\theta \|g_t(\theta) - g'_t(\theta)\| = \gamma \sup_\theta \|\nabla\ell(\theta, x_t) - \nabla\ell(\theta, x'_t)\| \leq 2\gamma L$$

Now $\forall i \neq t$, $s_i = 0$ and $s_t = 2\gamma L = s$.

Parameters a_i and z_i can be chosen freely, provided $z_0 = z_n = 0$ and $\forall k, z_k \geq 0$. We aim at solutions with minimal Rényi divergence, hence with minimal $\sum_i a_i^2$.

To ensure the constraints, we must have $a_1 = ... = a_{t-1} = 0$ and $z_0 = z_1 = ... = z_{t-1} = 0$. After that, $z_t = s - a_t$, $z_{t+1} = C(s - a_t) - a_{t+1}$, $z_{t+2} = C^2 s - C^2 a_t - C a_{t+1} - a_{t+2}$...

At final time step n:

$$z_n = C^{n-t}s - C^{n-t}a_t - C^{n-t-1}a_{t+1} - ... - Ca_{n-1} - a_n$$

Since we want $z_n = 0$, then

$$C^{n-t}s = C^{n-t}a_t + C^{n-t-1}a_{t+1} + ... + Ca_{n-1} + a_n$$

or equivalently:

$$s = a_t + C^{-1}a_{t+1} + ... + C^{t-n+1}a_{n-1} + C^{t-n}a_n$$

Minimizing $\sum_i a_i^2$ becomes an easy QP problem in dimension $n - t + 1$.

Writing $b_i := a_{t-1+i}$ for $i = 1, ..., n - t + 1$, then $b \in \mathbb{R}^{n-t+1}$. $\sum_{i=1}^{n} a_i^2 = \sum j = 1^{n-t+1} b_j^2$. The constraint is $\mu^\top b = s$ with $\mu = (1, C^{-1}, ..., C^{t-n+1}, C^{t-n})^\top$.

We want to solve:

$$\min_{b:\ \mu^\top b = s} \|b\|^2$$

This is equivalent to

$$\max_{\lambda} \min_{b} \|b\|^2 - \lambda \mu^\top b + \lambda s$$

Then $b = \frac{1}{2}\lambda\mu$ and $\lambda = \frac{2s}{\|\mu\|^2}$, hence $b = \frac{s}{\|\mu\|^2}\mu$ and $\|a\|^2 = \|b\|^2 = \frac{s^2}{\|\mu\|^2} = \frac{4\gamma^2 L^2}{\|\mu\|^2}$.

Then

$$\begin{aligned} D_\alpha(X_n || X'_n) &\leq \sum_{k=1}^{n} R_\alpha(\zeta_k, a_k) \\ &= \sum_{k=1}^{n} \frac{\alpha a_k^2}{2\sigma^2\gamma^2} \\ &= \frac{\alpha}{2\sigma^2\gamma^2} \sum_{k=1}^{n} a_k^2 \\ &= \frac{2\alpha L^2}{\sigma^2 \|\mu\|^2}. \end{aligned}$$

We still have to compute $\|\mu\|^2 = \sum_{i=0}^{n-t} (C^{-2})^i = \frac{1 - (C^{-2})^{n-t+1}}{1 - C^{-2}}$.

We finally get:

$$D_\alpha(X_n || X'_n) \leq \frac{2\alpha L^2}{\sigma^2} \frac{1 - C^{-2}}{1 - (C^{-2})^{n-t+1}}$$

□

A.5 Proof of Proposition 3.1

Proof. $\gamma \sim \frac{1}{\sqrt{n}}$ is such that $C^{-2} = 1 - \frac{1}{\sqrt{n}}$. In this case we seek an upper bound on:

$$\frac{1}{n\sqrt{n}} \sum_{i=1}^{n} \frac{1}{1 - (1 - \frac{1}{\sqrt{n}})^i}$$

Using Taylor with Cauchy remainder on $u \mapsto (1+u)^\alpha$ for $\alpha \geq 1$, we get:

$$\forall -1 < u \leq 0, \quad (1+u)^\alpha \leq 1 + \alpha u + \frac{\alpha(\alpha-1)}{2} u^2$$

In our case:

$$\left(1 - \frac{1}{\sqrt{n}}\right)^k \leq 1 - \frac{k}{\sqrt{n}} + \frac{k(k+1)}{2n}$$

hence

$$1 - \left(1 - \frac{1}{\sqrt{n}}\right)^k \geq \frac{k}{\sqrt{n}} - \frac{k(k+1)}{2n}$$

Now we look at two different cases:

- when $k \leq 2\sqrt{n} - 2$ we take the invert of the previous lower bound
- when $k \geq 2\sqrt{n} - 1$ we can do a direct upper bound

In the first case, $1 - \left(1 - \frac{1}{\sqrt{n}}\right)^k \geq \frac{k}{\sqrt{n}} - \frac{k(k+1)}{2n} \geq 0$ and so

$$\frac{1}{1 - (1 - \frac{1}{\sqrt{n}})^k} \leq \left(\frac{k}{\sqrt{n}} - \frac{k(k+1)}{2n}\right)^{-1} = \frac{\sqrt{n}}{k} \left(1 - \frac{k+1}{2\sqrt{n}}\right)^{-1}$$

Since $k \leq 2\sqrt{n} - 2$, $\frac{k+1}{2\sqrt{n}} \leq 1 - \frac{1}{2\sqrt{n}}$, hence $1 - \frac{k+1}{2\sqrt{n}} \geq \frac{1}{2\sqrt{n}} > 0$ and

$$\left(1 - \frac{k+1}{2\sqrt{n}}\right)^{-1} \leq 2\sqrt{n}$$

and finally

$$\frac{1}{1 - (1 - \frac{1}{\sqrt{n}})^k} \leq \frac{2n}{k}$$

In the second case, $k \geq 2\sqrt{n} - 1$, we can directly upper bound:

$$\frac{1}{1 - (1 - \frac{1}{\sqrt{n}})^k} \leq \frac{1}{1 - (1 - \frac{1}{\sqrt{n}})^{2\sqrt{n}-1}}$$

Putting all together we get:

$$\begin{aligned}
\frac{1}{n\sqrt{n}}\sum_{i=1}^{n}\frac{1}{1-(1-\frac{1}{\sqrt{n}})^i} &\leq \frac{1}{n\sqrt{n}}\left(\sum_{k=1}^{2\sqrt{n}-2}\frac{2n}{k}+\frac{n-2\sqrt{n}+2}{1-(1-\frac{1}{\sqrt{n}})^{2\sqrt{n}-1}}\right)\\
&\leq \frac{1}{n\sqrt{n}}\left(2n\sum_{k=1}^{2\sqrt{n}-2}\frac{1}{k}+\frac{n}{1-(1-\frac{1}{\sqrt{n}})^{2\sqrt{n}-1}}\right)\\
&\leq \frac{1}{n\sqrt{n}}\left(2n(1+\log(2\sqrt{n}))+\frac{n}{1-(1-\frac{1}{\sqrt{n}})^{2\sqrt{n}-1}}\right)\\
&\leq \frac{1}{\sqrt{n}}\left(2+2\log(2\sqrt{n})+\frac{1}{1-(1-\frac{1}{\sqrt{n}})^{2\sqrt{n}-1}}\right)
\end{aligned}$$

Asymptotically, the rate is $\frac{\log(n)}{\sqrt{n}}$ (with a learning rate $\gamma \sim \frac{1}{\sqrt{n}}$). This must be compared with the convex case in [13], where they get $\frac{\log(n)}{n}$. Overall the mechanism is RDP with parameters

$$\left(\alpha, \frac{4\alpha L^2}{\sigma^2}\frac{\log(n)}{\sqrt{n}}\right)$$

However we can show similarly that if $\gamma \sim \frac{1}{n}$, then we recover $\left(\alpha, \frac{4\alpha L^2}{\sigma^2}\frac{\log(n)}{n}\right)$ RDP. Better upper bounds can be obtained with Riemann sums. □

A.6 Proof of Proposition 3.3

Proof.

Step 1: Build the distribution of $u_{t+1}|u_t$.

$\eta_{t+1}|u_t \sim \mathcal{N}(0, \sigma^2 Id)$ and $\theta_{t+1}|\eta_{t+1}, u_t \sim \mathcal{N}((I_d - \gamma\alpha_t)\theta_t - \gamma\beta_t - \eta_{t+1} + \eta_t, 0)$ (degenerate gaussian, amounts to a dirac).

Now we use the formula (under the conditional probability p_{u_t}) that gives the joint probability from conditionals:

$$u_{t+1}|u_t \sim \mathcal{N}(\begin{pmatrix} 0 \\ (I_d - \gamma\alpha_t)\theta_t - \gamma\beta_t + \eta_t \end{pmatrix}, \begin{pmatrix} \sigma^2 I_d & -\sigma^2 I_d \\ -\sigma^2 I_d & \sigma^2 I_d \end{pmatrix})$$

Step 2: Compute the distribution of u_{t+1}.

Suppose $u_t \sim \mathcal{N}((n_t, m_t)^T, V_t)$ (true for $u_0 = (0, \theta_0)$), then $u_{t+1} \sim \mathcal{N}((n_{t+1}, m_t)^\top, V_{t+1})$. We know that:

$$u_{t+1}|u_t \sim \mathcal{N}(Au_t + B, \begin{pmatrix} \sigma^2 I_d & -\sigma^2 I_d \\ -\sigma^2 I_d & \sigma^2 I_d \end{pmatrix})$$

with $A = \begin{pmatrix} 0 & 0 \\ I_d & I_d - \gamma\alpha_t \end{pmatrix}$ and $B = \begin{pmatrix} 0 \\ -\gamma\beta_t \end{pmatrix}$.
Formulas give:

$$\begin{pmatrix} n_{t+1} \\ m_{t+1} \end{pmatrix} = A \begin{pmatrix} n_t \\ m_t \end{pmatrix} + B = \begin{pmatrix} 0 \\ n_t + (I_d - \gamma\alpha_t)m_t - \gamma\beta_t \end{pmatrix}$$

and we easily see that $\forall t, n_t = 0$ so that:

$$m_{t+1} = (I_d - \gamma\alpha_t)m_t - \gamma\beta_t$$

For the variances:

$$V_{t+1} = \begin{pmatrix} \sigma^2 I_d & -\sigma^2 I_d \\ -\sigma^2 I_d & \sigma^2 I_d \end{pmatrix} + AV_tA^\top$$

Step 3: Keep only the marginal distribution of θ_{t+1}

The mean is already computed:

$$m_{t+1} = (I_d - \gamma\alpha_t)m_t - \gamma\beta_t \quad ; \quad m_0 = \theta_0$$

For the variance, we must work with block matrices. For all t, we write

$$V_t = \begin{pmatrix} a_t & b_t \\ c_t & d_t \end{pmatrix}$$

where we are only interested in d_t. The recursion between variances gives:

$$\begin{pmatrix} a_{t+1} & b_{t+1} \\ c_{t+1} & d_{t+1} \end{pmatrix} = \begin{pmatrix} \sigma^2 I_d & -\sigma^2 I_d \\ -\sigma^2 I_d & \sigma^2 I_d \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ I_d & I_d - \gamma\alpha_t \end{pmatrix} \begin{pmatrix} a_t & b_t \\ c_t & d_t \end{pmatrix} \begin{pmatrix} 0 & I_d \\ 0 & (I_d - \gamma\alpha_t)^\top \end{pmatrix}$$

$$\begin{pmatrix} a_{t+1} & b_{t+1} \\ c_{t+1} & d_{t+1} \end{pmatrix} = \begin{pmatrix} \sigma^2 I_d & -\sigma^2 I_d \\ -\sigma^2 I_d & \sigma^2 I_d \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & a_t + (I_d - \gamma\alpha_t)c_t + (b_t + (I_d - \gamma\alpha_t)d_t)(I_d - \gamma\alpha_t) \end{pmatrix}$$

For all $t \geq 1$, $a_t = \sigma^2 I_d$ and $b_t = c_t = -\sigma^2 I_d$. Hence we have a recursive formula for d_t:

$$d_{t+1} = \sigma^2 I_d + \sigma^2 I_d - \sigma^2(I_d - \gamma\alpha_t) - \sigma^2(I_d - \gamma\alpha_t) + (I_d - \gamma\alpha_t)d_t(I_d - \gamma\alpha_t)$$

$$d_{t+1} = 2\gamma^2\alpha_t^2\sigma^2 + (I_d - \gamma\alpha_t)d_t(I_d - \gamma\alpha_t)$$

Taken the same notations as before, this amounts to:

$$\Sigma_{t+1} = 2\gamma^2\sigma^2\alpha_t^2 + (I_d - \gamma\alpha_t)\Sigma_t(I_d - \gamma\alpha_t)\,.$$

□

A.7 Proof of Lemma 4.1

Proof. If three datasets X, X', X'' of size n are mutually adjacent, they must differ on the same data point. Let it be the i-th, and the remaining $n-1$ data points are the same. Let p, q, r denote the distributions $M \circ subsample(X)$, $M \circ subsample(X')$, $M \circ subsample(X'')$, respectively.
Let E be the event such that the subsample includes the ith item (and E^c be complement event). We have $p(E) = q(E) = w_i$ (by assumption), we have:

$$p = w_i p(.|E) + (1 - w_i)p(.|E^c)$$

$$q = w_i q(.|E) + (1 - w_i)q(.|E^c)$$

and by construction, $p(.|E^c) = q(.|E^c)$.
Then $p - q = w_i(p(.|E) - q(.|E))$. Hence:

$$\begin{aligned}
D_{|\chi|^j}(p, q||r) &= \mathbb{E}_r\left[\left(\frac{|p-q|}{r}\right)^j\right] \\
&= \mathbb{E}_r\left[\left(\frac{|w_i(p(.|E) - q(.|E))|}{r}\right)^j\right] \\
&= w_i^j\mathbb{E}_r\left[\left(\frac{|p(.|E) - q(.|E)|}{r}\right)^j\right] \\
&\leq c^j\mathbb{E}_r\left[\left(\frac{|p(.|E) - q(.|E)|}{r}\right)^j\right] \\
&= c^j D_{|\chi|^j}(p(.|E), q(.|E)||r)
\end{aligned}$$

Let k be the random index chosen by the subsample operator. Its distribution is non uniform but we still get:

$$r(\theta) = \mathbb{E}_k[r(\theta|k)]$$

Now, define functions g and g' on index k such that:

$$g(k) := p(\theta|i)$$

and

$$g'(k) := q(\theta|i)$$

Check that $p(\theta|E) = \mathbb{E}_k[g(k)]$ and $q(\theta|E) = \mathbb{E}_k[g'(k)]$.

Then we can write:

$$\begin{aligned}\mathbb{E}_r\left(\frac{|p(\theta|E) - q(\theta|E)|}{r(\theta)}\right)^j &= \int \frac{|p(\theta|E) - q(\theta|E)|^j}{r(\theta)^{j-1}} d\theta \\ &\leq \int \mathbb{E}_k\left[\frac{|g(k) - g'(k)|^j}{r(\theta|k)^{j-1}}\right] d\theta \\ &= \mathbb{E}_k\mathbb{E}_r\left[\left(\frac{|g(k) - g'(k)|}{r(\theta|k)}\right)^j |k\right] \leq \zeta(j)^j \,.\end{aligned}$$

Finally we get the result:

$$D_{|\chi|^j}(p, q||r) = w_i^j D_{|\chi|^j}(p(.|E), q(.|E)||r) \leq (c\zeta(j))^j \,.$$

□