Learning rate adaptation for federated and differentially private learning

Antti Koskela\textsuperscript{1} and Antti Honkela\textsuperscript{1,2,3}

\textsuperscript{1} Helsinki Institute for Information Technology HIIT, Department of Mathematics and Statistics, University of Helsinki, Finland
\textsuperscript{2} Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland
\textsuperscript{3} Department of Public Health, University of Helsinki, Finland

Abstract

We propose an algorithm for the adaptation of the learning rate for stochastic gradient descent (SGD) that avoids the need for validation set use. The idea for the adaptiveness comes from the technique of extrapolation: to get an estimate for the error against the gradient flow which underlies SGD, we compare the result obtained by one full step and two half-steps. The algorithm is applied in two separate frameworks: federated and differentially private learning. Using examples of deep neural networks we empirically show that the adaptive algorithm is competitive with manually tuned commonly used optimisation methods for differentially privately training. We also show that it works robustly in the case of federated learning unlike commonly used optimisation methods.

Introduction

Most modern machine learning and especially deep learning methods rely heavily on optimising model and algorithm hyperparameters using a validation data set. Sometimes this may be infeasible if there are limitations to the use of the data, for example because of privacy or distributed nature of the data set in federated learning. In these settings, effective adaptive algorithms can be extremely important for efficient learning.

Differential privacy (DP) (Dwork et al., 2006; Dwork & Roth, 2014) has recently risen as the dominant paradigm for privacy-preserving machine learning. A number of differentially private algorithms have been proposed addressing both important specific models (e.g. Chaudhuri & Monteleoni, 2008; Dwork et al., 2014; Abadi et al., 2016) as well as more general approaches to learning (e.g. Chaudhuri et al., 2011; Dimitrakakis et al., 2014; Zhang et al., 2016; Park et al., 2016; Jätkö et al., 2017).
Like in machine learning more generally, differentially private stochastic gradient descent (DP-SGD) (Rajkumar & Agarwal, 2012; Song et al., 2013; Abadi et al., 2016) has emerged as an important tool for implementing differential privacy for a number of applications. The introduction of very tight bounds on the privacy loss occurring during the iterative algorithm computed via the moments accountant (Abadi et al., 2016) has made these algorithms particularly attractive. Furthermore, DP’s invariance to post-processing means that the same privacy guarantees apply to any algorithm that uses the same kind of gradient information, including many adaptive and accelerated methods such as AdaGrad (Duchi et al., 2011), RMSProp (Tieleman & Hinton, 2012) and Adam (Kingma & Ba, 2015). In addition to deep learning, SGD and more recently the moments accountant have been used in algorithms for other paradigms, such as Bayesian inference (Wang et al., 2015; Jalko et al., 2017; Li et al., 2017).

Beyond the use of specific DP algorithms, DP introduces additional procedural complications to learning. As any computation or decision made based on access to the data can leak private information, they all carry a privacy cost that must be taken into account. This challenges the common paradigm of SGD-based learning especially with deep neural networks where the model structure and the algorithm contain a number of tunable parameters that are optimised by running the algorithm repeatedly and evaluating the results on a validation set. This tuning is very important not only for finding a good model structure but also for tuning the SGD learning rate, because all widely used SGD algorithms are highly sensitive to such tuning.

Existing adaptive SGD alternatives such as AdaGrad, RMSProp and Adam are not as sensitive to tuning as plain SGD, but nevertheless require tuning for good performance. Furthermore, Wilson et al. (2017) argue that commonly used adaptive methods such as AdaGrad, RMSProp and Adam can lead to very poor generalisation performance in deep learning and that properly tuned basic SGD is a very competitive approach.

Federated learning (McMahan et al., 2017) has become popular as a means for communication-efficient learning with distributed data, and a useful tool for further improving privacy as well. The federated setup can severely restrict the possibility of using validation, because individual clients may differ from each other significantly which limits the value of generalising hyperparameters across clients, while at the same time limited availability of data and compute at an individual client may limit the use of local validation. In an extreme case the distribution of samples may be extremely biased between different clients, requiring the use of very different local learning rates that are impossible to tune with classical methods. Adaptive learning rate tuning can greatly increase both learning efficiency and stability in such cases.

In this paper, we propose a rigorous adaptive method for finding a good learning rate for SGD, and apply in DP and federated learning settings. The adaptation is performed during learning, which implies that the learning process only has to be executed once, leading to savings in compute time and efficient use of the pri-
privacy budget. We prove the privacy of our method based on the moments accountant mechanism. Our method is the first one to rigorously address this question under DP.

The challenges in DP hyperparameter tuning have been discussed previously by [Abadi et al. (2016)], but they only provide a coarse outline of possible solutions, not practical efficient algorithms. DP hyperparameter adaptation has also been discussed by [Kusner et al. (2015)] through DP Bayesian optimisation, but their method only protects the privacy of the validation data, not the original training data. In general, the trivial approach to DP hyperparameter adaptation would involve running the learning algorithm several times with different hyperparameter values, but the privacy cost of this approach is very high, leading to significant degradation of accuracy under a fixed privacy budget.

**Main contributions**

We propose the first learning rate adaptive DP SGD method. We give rigorous moment bounds for the method, and using these bounds, we can compute tight \((\varepsilon, \delta)\)-bounds using the so called moments accountant technique. By simple derivations, we show how to determine the additional tolerance hyperparameter in the algorithm. In computational experiments we show that it compares well with commonly used optimisation methods. We further demonstrate that the method can help stabilise federated learning especially when the data are non-uniformly distributed to different clients.

**Motivation for the learning rate adaptation: extrapolation of differential equations**

The main ingredient of the learning rate adaptation comes from numerical extrapolation of ordinary differential equations (ODEs), see e.g. [Stoer (1974)]. We next describe this idea.

Let \( g \) be a differentiable function \( g : \mathbb{R}^d \to \mathbb{R} \). The gradient descent method

\[
\theta_{\ell+1} = \theta_\ell - \eta_\ell \nabla g(\theta_\ell)
\]

is a first-order method for finding a (local) minimum of the function \( g \). It can be seen as an explicit Euler method with a step of size \( \eta_\ell \) applied to the system of ODEs

\[
\frac{d}{dt} \theta(t) = -\nabla g(\theta), \quad \theta(0) = \theta_0 \in \mathbb{R}^d,
\]

which is also called the gradient flow corresponding to \( g \). For basics of numerical methods for ODEs, we refer to [Hairer et al. (1987)].

To get an estimate of the error made in the numerical approximation (1), we extrapolate it as follows. Consider one Euler step of size \( \eta \) applied to the gradient flow (1).

\[
\theta_1 = \theta_0 - \eta \nabla g(\theta_0),
\]

(2)
and $\hat{\theta}_1$ which is a result of two steps of size $\eta^2/2$:

$$
\theta_{1/2} = \theta_0 - \frac{\eta}{2} \nabla g(\theta_0)
$$

$$
\hat{\theta}_1 = \theta_{1/2} - \frac{\eta}{2} \nabla g(\theta_{1/2}).
$$

Then, assuming $g$ is twice differentiable, we get by the Taylor expansion

$$
\hat{\theta}_1 = \theta_0 - \eta \nabla g(\theta_0) + \eta^2 \frac{J_g(\theta_0)}{4} \nabla g(\theta_0) + O(\eta^3).
$$

From (2) and (3) we see that

$$
\hat{\theta}_1 - \theta_1 = \eta^2 \frac{J_g(\theta_0)}{4} \nabla g(\theta_0) + O(\eta^3).
$$

On the other hand, from the Taylor expansion of the true solution $\theta(\eta)$, we see that

$$
\theta(\eta) - \theta_1 = \frac{\eta^2}{2} J_g(\theta_0) \nabla g(\theta_0) + O(\eta^3).
$$

Thus, we see from (4) and (5) that $2(\hat{\theta}_1 - \theta_1)$ gives an $O(\eta^3)$-estimate of the local error generated by the GD step (1).

If at step $\ell$ of the iteration we have the error estimate

$$
err_\ell = \|\hat{\theta}_\ell - \theta_\ell\|,
$$

and if a local error of size $tol$ is desired, a simple mechanism for updating the step size is given by

$$
\eta_{\ell+1} = \min \left( \max \left( \frac{\text{tol}}{\text{err}_\ell}, \alpha_{\text{min}} \right), \alpha_{\text{max}} \right) \cdot \eta_\ell,
$$

where $\alpha_{\text{min}} < 1$ and $\alpha_{\text{max}} > 1$. In our experiments we have used $\alpha_{\text{min}} = 0.9$, $\alpha_{\text{max}} = 1.1$. In case $\nabla g$ has a large Lipschitz constant, a condition $\text{err}_\ell < \text{tol}$ for the update $\theta_{\ell+1} \leftarrow \theta_\ell$ gives a more stable algorithm. This whole procedure is depicted in Algorithm 1.

We apply Algorithm 1 to the differentially private SGD method and to the federated learning algorithm. The challenge in the DP setting is that for privacy reasons the gradients are blurred by the additive DP-noise.

**Differential Privacy**

**Definition of Differential Privacy**

We first recall some basic definitions of differential privacy (Dwork & Roth 2014).

We use the following notation. An input set containing $N$ data points is denoted as $X = (x_1, \ldots, x_N) \in \mathcal{X}^N$, where $x_i \in \mathcal{X}$, $1 \leq i \leq N$. For giving the definition of the actual differential privacy we need the following definition.
Algorithm 1 The update mechanism defined by the parameters $\alpha_{\min}, \alpha_{\max}, \text{tol}$

Take one Euler step

$$\theta_{\ell+1} \leftarrow \theta_{\ell} - \eta \nabla g(\theta_{\ell}),$$

Take two steps of size $\frac{\eta}{2}$:

$$\theta_{\ell+1/2} \leftarrow \theta_{\ell} - \frac{\eta}{2} \nabla g(\theta_{\ell})$$

$$\tilde{\theta}_{\ell+1} \leftarrow \theta_{\ell+1/2} - \frac{\eta}{2} \nabla g(\theta_{\ell+1/2})$$

Evaluate: $\text{err}_\ell \leftarrow \|\tilde{\theta}_{\ell+1} - \theta_{\ell+1}\|$  

if $\text{err}_\ell < \text{tol}$: then  

$$\theta_{\ell+1} \leftarrow \theta_{\ell}$$

end if

update: $\eta_{\ell+1} = \min(\max\left(\frac{\text{tol}}{\text{err}_\ell}, \alpha_{\min}\right), \alpha_{\max}) \cdot \eta_{\ell}.$

**Definition 1.** We say that two data sets $X$ and $X'$ are adjacent if they only differ in one record, i.e., if $x_i \neq x'_i$ for some $i$, where $x_i \in X$ and $x'_i \in X'$.

The following definition formalises the $(\varepsilon, \delta)$-differential privacy of a randomised mechanism $M$.

**Definition 2.** Let $\varepsilon > 0$ and $\delta \in [0, 1]$. Mechanism $M : X^N \rightarrow Z$ is $(\varepsilon, \delta)$-DP if for every pair of neighbouring data sets $X, X'$ and every measurable set $E \subset Z$ we have

$$\Pr(M(X) \in E) \leq e^\varepsilon \Pr(M(X') \in E) + \delta.$$  

This definition is closed under post-processing which means that if a mechanism $A$ is $(\varepsilon, \delta)$-differential private, then so is the mechanism $B \circ A$ for all functions $B$ that do not depend on the data.

Assuming $X$ and $X'$ differ only by one record $x_i$, then by observing the outputs, the ability of an attacker to tell whether the output has resulted from $X$ or $X'$ remains bounded. Thus, the record $x_i$ is protected. As the record in which the two data sets differ is arbitrary, by definition, the protection applies for the whole data set.

**Moments accountant**

We next recall some basic definitions and results concerning the moments accountant technique which is an important ingredient for our proposed method and crucial for obtaining tight $(\varepsilon, \delta)$-privacy bounds for the differentially private stochastic gradient descent. We refer to [Abadi et al. (2016)] for more details.
Definition 3. Let $M : X^N \to Y$ be a randomised mechanism, and let $X$ and $X'$ be a pair of adjacent data sets. Let aux denote any auxiliary input that does not depend on $X$ or $X'$. For an outcome $o \in Y$, the privacy loss at $o$ is defined as

$$c(o; M, aux, X, X') = \log \frac{\Pr(M(aux, X) = o)}{\Pr(M(aux, X') = o)}.$$

Definition 4. $\lambda$th moment generating function $\alpha_M(\lambda; aux, X, X')$ is defined as

$$\alpha_M(\lambda; aux, X, X') = \log \mathbb{E}_{o \sim M(aux, X)}(\exp(\lambda c(o; M, aux, X, X'))).$$

Definition 5. Let $M : X^N \to Y$ be a randomised mechanism, and let $X$ and $X'$ be a pair of adjacent data sets. Let aux denote any auxiliary input that does not depend on $X$ or $X'$. The moments accountant with an integer parameter $\lambda$ is defined as

$$\alpha_M(\lambda) = \max_{aux, X, X'} \alpha_M(\lambda; aux, X, X').$$

The privacy of our proposed method is based on the composability theorem given by Abadi et al. (2016):

Theorem 1. Suppose that $M$ consists of a sequence of adaptive mechanisms $M_1, \ldots, M_k$, where $M_i : \prod_{j=1}^{i-1} Y_j \times X \to Y_i$, and $Y_i$ is in the range of the $i$th mechanism, i.e., $M = M_k \circ \ldots \circ M_1$. Then, for any $\lambda$

$$\alpha_M(\lambda) \leq \sum_{i=1}^{k} \alpha_{M_i}(\lambda),$$

where the auxiliary input for $\alpha_{M_i}(\lambda)$ is defined as all $\alpha_{M_j}(\lambda)$’s outputs for $j < i$, and $\alpha_{M_i}(\lambda)$ takes $M_i$’s output, for $i < k$, as the auxiliary input.

Moreover, for any $\varepsilon > 0$, the mechanism $M$ is $(\varepsilon, \delta)$-differentially private for

$$\delta = \min_{\lambda} \exp(\alpha_M(\lambda) - \lambda \varepsilon).$$

The inequality (7) gives an upper bound for the total moment $\alpha_M(\lambda)$ of an iterative algorithm $M$ if the moments $\alpha_{M_i}(\lambda)$ of each iteration $i$ are known. Important for the additivity of the moments is that the noise of the mechanisms $M_i$ is pairwise independent. Using (8), the privacy parameters $\varepsilon$ and $\delta$ can be numerically computed from $\alpha_M(\lambda)$-values.

Differentially private stochastic gradient descent

Suppose we want to find a minimum (w.r.t. $\theta$) of a loss function of the form

$$L(\theta, X) = \frac{1}{N} \sum_{i=1}^{N} f(\theta, x_i).$$
At each step of the differentially private SGD (see Algorithm 2), we compute the gradient $\nabla_\theta f(\theta, x_i)$ for a random minibatch $B$, clip the 2-norm of each gradient belonging to the minibatch, compute the average, add noise in order to protect privacy, and take a GD step using this noisy gradient. For a data set $X$, the basic mechanism is then given by
\[
\mathcal{M}(X) = \sum_{i \in B} \tilde{\nabla} f(\theta, x_i) + \mathcal{N}(0, C^2 \sigma^2 I),
\]
where $\tilde{\nabla} f(\theta, x_i)$'s denote the gradients clipped with a constant $C > 0$. We recall a result by Abadi et al. (2016), which gives a privacy bound for the mechanism (9).

**Lemma 1.** Suppose that $f : X \rightarrow \mathbb{R}^d$ with $\|f(\cdot)\| \leq 1$. Let $\sigma \geq 1$ and $B$ a minibatch with sampling probability $q$, i.e., $q = \frac{|B|}{N}$, where $N$ is the number of records in the data. If $q < \frac{1}{16\sigma}$, then for any positive integer $\lambda \leq \sigma^2 \ln \frac{1}{q\sigma}$, the mechanism $\mathcal{M}(X) = \sum_{i \in B} f(\theta, x_i) + \mathcal{N}(0, \sigma^2 I)$ satisfies
\[
\alpha_{\mathcal{M}}(\lambda) \leq \frac{q^2 \lambda (\lambda + 1)}{(1-q)\sigma^2} + O(q^3 \lambda^3 / \sigma^3).
\]

In numerical experiments we compute the moments using the numerical methods of Abadi et al. (2016).

**Algorithm 2** Differentially private SGD

Input: Data records $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate $\eta$, noise level $\sigma$, group size $L$, gradient clipping constant $C$.

Initialise $\theta_0$ randomly.

for $\ell = 0, 1, \ldots$ do
  Draw a batch $B$, with $q = |B| / N$.
  Compute the gradients:
  For every $i \in B$, compute: $f_\ell(x_i) = \nabla_\theta \mathcal{L}(\theta_\ell, x_i)$
  Clip the gradients:
  $\hat{f}_\ell(x_i) \leftarrow f_\ell(x_i) / \max\{1, \frac{\|f_\ell(x_i)\|_2}{C}\}$
  Add noise:
  $G_\ell \leftarrow \frac{1}{|B|} \left( \sum_{i \in B} \hat{f}_\ell(x_i) + \mathcal{N}(0, \sigma^2 I) \right)$
  Descent:
  $\theta_{\ell+1} \leftarrow \theta_\ell - \eta_\ell G_\ell$
end for

**Adaptive DP algorithm**

The result of applying the learning rate adaptation to DP-SGD is depicted in Algorithm 3. We abbreviate this method as ADADP. Instead of (6), we use for the error
estimate the 2-norm of the function \( \text{err}(\theta, \hat{\theta}) \), where

\[
\text{err}(\theta, \hat{\theta}) = \frac{|\theta_i - \hat{\theta}_i|}{\max(1, |\theta_i|)}
\]  

(10)
as this was found to perform better numerically. In the case of DP learning the algorithm was found to be stable without the condition \( \text{err}_i < \text{tol} \) so we omit it. Here the factor \(|B|\) is dropped, as it only scales the learning rate \( \eta \).

**Algorithm 3** ADADP update mechanism defined by the parameters \( \alpha_{\text{min}}, \alpha_{\text{max}}, \text{tol} \)

1. Draw a batch \( B_1 \), with probability \( q = |B| / N \).
   - Clip the gradients and evaluate at \( \theta_\ell \):
     \[
     G_1 \leftarrow \sum_{i \in B_1} \tilde{\nabla} f_{\theta_\ell}(x_i) + \mathcal{N}(0, C^2 \sigma^2 I).
     \]
   - Take one Euler step of size \( \eta_\ell \): \( \hat{\theta}_{\ell+1} \leftarrow \theta_\ell - \eta_\ell G_1 \),

2. Draw a batch \( B_2 \), with probability \( q = |B| / N \).
   - Clip the gradients and evaluate at \( \theta_\ell + 1/2 \):
     \[
     G_2 \leftarrow \sum_{i \in B_2} \tilde{\nabla} f_{\theta_{\ell+1/2}}(x_i) + \mathcal{N}(0, C^2 \sigma^2 I).
     \]
   - Take two steps of size \( \frac{\eta_\ell}{2} \):
     \[
     \theta_{\ell+1/2} \leftarrow \theta_\ell - \frac{\eta_\ell}{2} G_1, \quad \tilde{\theta}_{\ell+1} \leftarrow \theta_{\ell+1/2} - \frac{\eta_\ell}{2} G_2
     \]

Evaluate: \( \text{err}_\ell \leftarrow \|err(\theta_{\ell+1}, \hat{\theta}_{\ell+1})\|_2 \)

Update: \( \eta_{\ell+1} \leftarrow \min\left( \max\left( \frac{\text{tol}}{\text{err}_\ell}, \alpha_{\text{min}} \right), \alpha_{\text{max}} \right) \cdot \eta_\ell \).

In the beginning of the algorithm, the step size is likely to either monotonously increase or decrease until appropriate level is achieved. To shorten this initial phase, \( \alpha_{\text{min}} \) and \( \alpha_{\text{max}} \) further away from 1 could be used in the beginning, e.g. \( \alpha_{\text{min}} = 0.5 \) and \( \alpha_{\text{max}} = 2 \).

**Privacy preserving properties of the method**

By the very construction of Algorithm 3 and due to the post-processing property of differential privacy, we have the following result.
Theorem 2. Let \( q = |B|/N \), \( \sigma \geq 1 \) and \( C > 0 \). Let \( \alpha_{\mathcal{M}}(\lambda) \) be the moments accountant of a mechanism of the form \( \mathcal{M} \) for these parameter values. Let \( \tilde{\mathcal{M}} \) denote the mechanism of Algorithm 3 using these parameter values. Then,

\[
\alpha_{\tilde{\mathcal{M}}}(\lambda) \leq 2\alpha_{\mathcal{M}}(\lambda).
\]

By Theorem 2, using the same parameter values, we are allowed to run Algorithm 3 half as many times as differentially private SGD in order to have the same privacy.

Accumulation of noise and choice of parameter \( tol \)

For simplicity, consider the situation where we apply DP-SGD with step sizes \( \{\eta_\ell\} \).

After \( T \) steps

\[
\theta_T = \theta_0 - \sum_{\ell=0}^{T-1} \eta_\ell g(\theta_\ell) + \mathcal{N}
\left(
0, \sum_{\ell=0}^{T-1} \eta_\ell^2 C^2 \sigma^2 I
\right),
\]

where \( g(\theta_\ell) = \sum_{i \in B_\ell} \nabla f(x_i) \). Clearly, \( \|g(\theta_\ell)\|_2 \leq C |B| \). It holds

\[
\|\theta_{\ell+1} - \hat{\theta}_{\ell+1}\| = \frac{\eta_\ell}{2} \|g(\theta_\ell) - g(\hat{\theta}_\ell) + \mathcal{N}(0, 2C^2 \sigma^2 I)\|.
\]

Assuming \( \sqrt{d} \gg |B| \) (Recall: \( \theta \in \mathbb{R}^d \)) and taking the expectation value, we may approximate \( \|\theta_{\ell+1} - \hat{\theta}_{\ell+1}\| \approx \eta_\ell \sigma C \sqrt{\frac{d}{2}} \). If we set this estimate to \( tol \), we have approximately \( \eta_\ell^2 = \frac{2tol^2}{\sigma^2 C^2 d} \). Substituting this into the third term on the right hand side of (11), we see that after \( T \) steps each element of that term is approximately \( \sqrt{2tol} \). By requiring that this noise is, for example, \( O(1) \), we find a suitable value for the parameter \( tol \). In our experiments with neural networks \( \frac{2T}{\sigma} = O(1) \) and we use \( tol = 1.0 \). In the other extreme, i.e., \( \sqrt{d} \ll |B| \), the term \( g(\theta_\ell) - g(\hat{\theta}_\ell) \) is likely to dominate the estimate \( \|\theta_{\ell+1} - \hat{\theta}_{\ell+1}\| \). Then it is the Lipschitz constant of \( g \) that dictates the suitable step size \( \eta_\ell \) and potentially a smaller value of \( tol \) is needed.

Adaptive federated learning

We consider next the federated averaging algorithm given by McMahan et al. (2017). The idea is such that the same model is first distributed to several clients. The clients update their models based on their local data, and these models are then aggregated after a given interval by a server which then averages the models to obtain a global model. This global model is then again distributed to the clients.

In the algorithm described in McMahan et al. (2017) a random subset of clients is considered at each aggregation. We consider a fixed set of clients. Suppose there are \( K \) clients and denote the set of clients by \( S \). Suppose each client \( k \in S \) has
Algorithm 4 The federated averaging method

Server executes:
for \( \ell = 0, 1, \ldots \) do
  for each client \( k \in S \) in parallel: do
    \( \theta_{\ell+1}^k \leftarrow \text{ClientUpdate}(k, \theta_{\ell}) \)
  end for
  \( \theta_{\ell+1} \leftarrow \sum_{k \in S} \frac{n_k}{n} \theta_{\ell+1}^k \)
end for

Algorithm 5 Learning rate adaptive client update

ClientUpdate\((k, \theta)\)
Split \( P_k \) into batches of size \(|B|\).
for each local step \( \ell = 1, \ldots, E \) do
  Draw a batch \( B_1 \) and evaluate at \( \theta_{\ell} \):
  \[ G_1 \leftarrow \sum_{i \in B_1} \nabla f_{\theta_{\ell}}(x_i). \]
  Take one Euler step of size \( \eta_{\ell} \):
  \[ \hat{\theta}_{\ell+1} \leftarrow \theta_{\ell} - \eta_{\ell} G_1, \]
  Draw a batch \( B_2 \), and evaluate at \( \theta_{\ell+1}/2 \):
  \[ G_2 \leftarrow \sum_{i \in B_2} \nabla f_{\theta_{\ell+1}/2}(x_i). \]
  Take two steps of size \( \frac{\eta_{\ell}}{2} \):
  \[ \theta_{\ell+1/2} \leftarrow \theta_{\ell} - \frac{\eta_{\ell}}{2} G_1, \quad \tilde{\theta}_{\ell+1} \leftarrow \theta_{\ell+1/2} - \frac{\eta_{\ell}}{2} G_2 \]
Evaluate: \( \text{err}_{\ell} \leftarrow \| \text{err}(\theta_{\ell+1}, \hat{\theta}_{\ell+1}) \|_2 \)
if \( \text{err}_{\ell} < \text{tol} \) then
  \( \theta_{\ell+1} \leftarrow \theta_{\ell} \)
end if
update: \( \eta_{\ell+1} = \min(\max(\frac{\text{tol}}{\text{err}_{\ell} \cdot \alpha_{\text{min}}}, \alpha_{\text{max}}) \cdot \eta_{\ell}). \)
end for
a dataset $\mathcal{P}_k$ containing $n_k$ elements. Denote $\sum_{k \in S} n_k = n$. Then, the federated averaging algorithm can be described by Algorithm 4.

In McMahan et al. (2017) SGD with a constant learning rate is used for the updates of the clients. The motivation for using the learning rate adaptation comes from the fact that after averaging and distributing, the model at each client may be very far from the optimum for the local data and thus small steps are needed in the beginning of each sub training. Moreover, the data may vary considerably between the clients, leading to varying optimal learning rates.

For the learning rate adaptation, we use the same procedure as in ADADP, but without the additive noise and clipping of the gradients. We also add the condition $\text{err}_i < \text{tol}$ for the model update as it makes the algorithm considerably more stable. This adaptive client update is described in Algorithm 5. We use here also the error function (10). In all experiments using Algorithm 5 we used $\text{tol} = 0.1$ (see the discussion of Subsection).

**Experiments**

We compare ADADP with two commonly used optimisation methods: DP-SGD and Adam combined with DP gradients. The federated averaging algorithm with adaptive learning rates is also compared with Adam and SGD.

We compare the methods on two standard datasets: MNIST (LeCun et al., 1998) and CIFAR-10.

The random sampling of minibatches is approximated as in Abadi et al. (2016), i.e., by randomly permuting the data elements and then partitioning them into minibatches of a fixed size. As we see, both Algorithm 3 and Algorithm 4 need two minibatches per iteration: one to compute the vector $G_1$ and then the next one to compute $G_2$. Therefore, in one epoch we run $\frac{N}{2|B|}$ iterations. Then the number of gradient evaluations per epoch is the same as for SGD and Adam and thus the computation times are essentially equivalent. When using ADADP, also the per epoch privacy cost is then the same for all the methods considered, for a fixed value of the noise parameter $\sigma$.

In the DP setting the methods are compared by measuring the test accuracy for a given $\varepsilon$-value, when $\delta = 10^{-5}$. The $\varepsilon$-values are computed using the moments accountant method described in Abadi et al. (2016).

The values $\alpha_{\text{min}} = 0.9$ and $\alpha_{\text{max}} = 1.1$ were used in all experiments when using Algorithms 3 and 4. In all experiments with ADADP we used the value $\text{tol} = 1.0$ and in all non-DP experiments the value $\text{tol} = 0.1$.

All experiments are implemented using PyTorch.

**Datasets and test architectures**

In MNIST each example is a $28 \times 28$ size gray-level image. The training set contains 60000 and the test set 10000 examples. For MNIST we use a feedforward
neural network with 2 hidden layers with 256 hidden units. As a result, the total number of parameters for this network is 334336. We use ReLU units and the last layer is passed to softmax of 10 classes with cross-entropy loss. Without additional noise (\( \sigma = 0 \)) we reach an accuracy of around 96%.

CIFAR-10 consists of colour images classified into 10 classes. The training set contains 50000 and the test set 10000 examples. Each example is a 32 \( \times \) 32 image with three RGB channels. For CIFAR-10 we use a simple neural network, which consists of two convolutional layers followed by three fully connected layers. The convolutional layers use 3 \( \times \) 3 convolutions with stride 1, followed by ReLU and max pools, with 64 channels each. The output of the second convolutional layer is flattened into a vector of dimension 1600. The fully connected layers have 500 hidden units. Last layer is passed to softmax of 10 classes with cross-entropy loss. The total number of parameters for this network is about \( 10^6 \). Similarly to the experiments of Abadi et al. (2016), in the DP setting we pre-train the convolutional layers using the CIFAR-100 data set and the differentially private optimisation is carried out only for the fully connected layers.

**Comparison of ADADP against Adam**

We use all the methods with minibatch size \( |B| = 200 \) and run each method for 100 epochs. The initial learning rate for ADADP is set to \( 10^{-1} \), but the results are quite insensitive to this value as the algorithm will converge to the desired learning rate already during the first epoch.

We first compare ADADP with optimally and almost optimally tuned Adam. This means that in each case (\( \sigma = 2.0, 4.0 \) and 6.0) we search the best and the second best initial learning rate \( \eta_0 \) for Adam on a grid \{... , \( 10^{-2.5} \), \( 10^{-2.0} \), \( 10^{-1.5} \), ...\}. We apply ADADP for 50 steps, then fix the learning rate (denoted \( \eta_{50} \)) and apply SGD with the decaying learning rate

\[
\eta_k = \frac{\eta_{50}}{1 + 0.1 \cdot (k - 50)},
\]

where \( k \) denotes the number of epoch (\( k > 50 \)).

As Figure 1 illustrates, in case of MNIST and the feedforward network, ADADP is competitive with the learning rate optimised Adam and gives better results than Adam with the second best learning rate found from the grid. We see from Figure 2 that in the case of CIFAR-10 and convolutional network, ADADP is again competitive with the learning rate optimised Adam and gives clearly better results than Adam with the second best learning rate.

**Experiments for ADADP and SGD**

Next, we search an optimal learning rate for SGD on a grid \{... , \( 10^{-2.5} \), \( 10^{-2.0} \), \( 10^{-1.5} \), ...\} in the case \( \sigma = 2.0 \). Using this learning rate for SGD, we compare the performance of SGD and ADADP when \( \sigma = 4.0, 6.0 \) and 8.0. As we see from Figures 3 and 4...
ADADP finds an appropriate learning rate and gives better results than SGD for these values of \( \sigma \). This example is motivated by the fact that the learning rate found by ADADP is nearly constant after finding a suitable level. Thus an optimally tuned SGD would necessarily be very competitive against ADADP. One could expect to find a a suitable learning rate using the case \( \sigma = 2.0 \).

**Federated learning experiments**

To illustrate the benefits of the learning rate adaptation in the case of the federated averaging algorithm (Algorithm 3), we consider the CIFAR-10 dataset and the convolutional network described above. We first consider a pathological case, where we divide the data to five clients as follows: Client 1: cars and trucks, Client 2: planes and ships, Client 3: cats and dogs, Client 4: birds and frogs, Client 5: deers and horses.

Then, each client has a training data of 10000 images. For a given minibatch size \(|B|\), each client carries out \( E \) number of sub steps between each aggregation such that \(|B| \cdot E = 10000\), i.e., the clients always run one epoch between each aggregation.

We compare the learning rate adaptive client update algorithm (Algorithm 5) with constant learning rate SGD. We use for the SGD client update the minibatch
size $|B| = 25$. As Figure 5 shows, the adaptiveness is needed to overcome the instabilities of SGD caused by vastly different datasets between clients. Moreover, we notice that a larger number of sub steps $E$ (smaller $|B|$) gives better results for the learning rate adaptive federated averaging method.

Next, we interpolate between this pathological case and the uniformly random distribution of data between five clients. Figure 6 depicts the test accuracies for the learning rate adaptive algorithm and SGD. The learning rate of SGD is tuned in the grid $\{\ldots, 10^{-2.5}, 10^{-2.0}, 10^{-1.5}, \ldots\}$. We use in all alternatives $|B| = 10$. We see that as the distribution of data becomes more pathological (33% of the data chosen randomly), the learning rate adaptive method is able to maintain the overall performance much better than SGD. Notice that Figure 5 corresponds here to 0% random data.

Adam gave poor results in this example. Figure 7 shows the test accuracies in the interpolated case, where 33% of the data is chosen randomly for each client, for the best initial learning rates found from the grid $\{\ldots, 10^{-5.5}, 10^{-5.0}, 10^{-4.5}, \ldots\}$. We use here $|B| = 10$. Notice here the different scale of $y$-axis as in Figure 6.
Figure 3: ADADP and SGD for MNIST. The fixed learning rate $\eta$ of SGD is tuned in the $\sigma = 2.0$-case using the grid \{..., $10^{-2.5}$, $10^{-2.0}$, $10^{-1.5}$, ...\}.

Conclusions

We have proposed the first learning rate adaptive DP-SGD method. We believe this is the first rigorous DP-SGD approach, because all previous works have glossed over the need to tune the SGD learning rate. By simple derivations, we have shown how to determine the additional tolerance hyperparameter in the algorithm. Based on this heuristic analysis, we developed a rule for selecting the parameter and verified the efficiency of the resulting algorithm in a number of diverse learning problems. The results show that our approach is competitive in performance with commonly used optimisation methods even without any tuning, which is infeasible in the DP setting. Overall, our work takes an important step toward truly DP and automated learning for SGD-based learning algorithms.

Federated learning presents another setting where classical hyperparameter adaptation with a validation set may be impractical and also leads to suboptimal results. One obvious pain point is skewed distribution of data on different clients, which may lead to different clients requiring very different learning rates that would be very difficult to tune without an adaptive algorithm. Our algorithm can handle even highly pathological cases here with ease.

As a future work, it would be useful to develop a better understanding of the tolerance hyperparameter. Furthermore, it would be important to study the adaptation of other key algorithmic parameters of DP-SGD, such as the gradient clipping...
Figure 4: ADADP and SGD for CIFAR-10. The fixed learning rate $\eta$ of SGD is tuned in the $\sigma = 2.0$-case using the grid $\{\ldots, 10^{-2.5}, 10^{-2.0}, 10^{-1.5}, \ldots\}$.

threshold and the minibatch size. Balles et al. (2017) provide an interesting non-private implementation of minibatch adaptation, but unfortunately their approach cannot easily be applied in the DP case.

Acknowledgements
This work has been supported by the Academy of Finland [Finnish Center for Artificial Intelligence FCAI and grant 313124].

References
Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In Proc. CCS 2016, 2016. doi: 10.1145/2976749.2978318.

Balles, L., Romero, J., and Hennig, P. Coupling adaptive batch sizes with learning rates. In Proc. UAI 2017, 2017.

Chaudhuri, K. and Monteleoni, C. Privacy-preserving logistic regression. In Adv. Neural Inf. Process. Syst. 21, pp. 289–296, 2008.
Figure 5: Test accuracies for the federated averaging algorithm in the pathological case for the learning rate adaptive method for different batch sizes $|B|$ and SGD for different learning rates $\eta$.

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12:1069–1109, July 2011. ISSN 1532-4435.

Dimitrakakis, C., Nelson, B., Mitrokotsa, A., and Rubinstein, B. I. P. Robust and private Bayesian inference. In *Proc. ALT 2014*, pp. 291–305. 2014.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. ISSN 1532-4435.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014. ISSN 1551-305X. doi: 10.1561/0400000042.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proc. TCC 2006*, pp. 265–284. 2006. ISBN 978-3-540-32732-5. doi: 10.1007/11681878_14.

Dwork, C., Talwar, K., Thakurta, A., and Zhang, L. Analyze gauss: Optimal bounds for privacy-preserving principal component analysis. In *Proc. STOC 2014*, pp. 11–20, 2014. ISBN 978-1-4503-2710-7. doi: 10.1145/2591796.2591883.
Figure 6: Test accuracies for the federated averaging algorithm using the adaptive method and a learning rate tuned SGD, when the training data is interpolated between the pathological case and the uniformly random distribution of data.

Hairer, E., Norsett, S., and Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Computational Mathematics*. Springer, Berlin, 1987.

Jäلكö, J., Honkela, A., and Dikmen, O. Differentially private variational inference for non-conjugate models. In *Proc. UAI 2017*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proc. ICLR 2015*, 2015.

Kusner, M. J., Gardner, J. R., Garnett, R., and Weinberger, K. Q. Differentially private Bayesian optimization. In *Proc. ICML 2015*, pp. 918–927, 2015.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Li, B., Chen, C., Liu, H., and Carin, L. On connecting stochastic gradient MCMC and differential privacy. 2017. arXiv:1712.09097 [stat.ML].

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A.
Figure 7: Test accuracies for the federated averaging algorithm, when the client updates are done using Adam using different initial learning rates $\eta_0$.

Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, 2017.

Park, M., Foulds, J., Chaudhuri, K., and Welling, M. Variational Bayes in private settings (VIPS). 2016. arXiv:1611.00340 [stat.ML].

Rajkumar, A. and Agarwal, S. A differentially private stochastic gradient descent algorithm for multiparty classification. In *Proc. AISTATS 2012*, pp. 933–941, 21–23 Apr 2012.

Song, S., Chaudhuri, K., and Sarwate, A. D. Stochastic gradient descent with differentially private updates. In *Proc. GlobalSIP 2013*, pp. 245–248, 2013. doi: 10.1109/GlobalSIP.2013.6736861.

Stoer, J. Extrapolation methods for the solution of initial value problems and their practical realization. In *Proc. Conf. on the Numerical Solution of Ordinary Differential Equations*, pp. 1–21, 1974.

Tieleman, T. and Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.
Wang, Y., Fienberg, S. E., and Smola, A. J. Privacy for free: Posterior sampling and stochastic gradient Monte Carlo. In Proc. ICML 2015, pp. 2493–2502, 2015.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. In Adv. Neural Inf. Process. Syst. 30, pp. 4148–4158. 2017.

Zhang, Z., Rubinstein, B., and Dimitrakakis, C. On the differential privacy of Bayesian inference. In Proc. Conf. AAAI Artif. Intell. 2016, 2016.