1 Comparison to the Private Selection algorithm by Liu and Talwar

<table>
<thead>
<tr>
<th></th>
<th>Mean acc.</th>
<th>Std acc.</th>
<th>Mean evals</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADADP</td>
<td>0.6349</td>
<td>0.0033</td>
<td>1</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-9}$</td>
<td>0.6386</td>
<td>0.0011</td>
<td>601.04</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-8}$</td>
<td>0.6368</td>
<td>0.0113</td>
<td>224.30</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-7}$</td>
<td>0.6317</td>
<td>0.0399</td>
<td>131.76</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-6}$</td>
<td>0.6266</td>
<td>0.0516</td>
<td>49.77</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-5}$</td>
<td>0.6215</td>
<td>0.0552</td>
<td>32.25</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-4}$</td>
<td>0.6053</td>
<td>0.0662</td>
<td>14.79</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-3}$</td>
<td>0.5576</td>
<td>0.1280</td>
<td>6.72</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-2}$</td>
<td>0.5467</td>
<td>0.1278</td>
<td>3.42</td>
</tr>
<tr>
<td>Priv. $\gamma = 2^{-1}$</td>
<td>0.4728</td>
<td>0.1688</td>
<td>2.04</td>
</tr>
</tbody>
</table>

Table 1: Comparison of ADADP and the private selection algorithm [1, Alg. 2] for different values of the parameter $\gamma$. 'Mean evals' denotes the mean of the number of training runs (100 epochs each) needed for one evaluation of each algorithm. 'Mean acc.' and 'Std acc.' denote the mean and standard deviation of the test accuracy of the resulting model, respectively. Both methods have the same $(\varepsilon, \delta)$-privacy for the training dataset. The private selection algorithm needs also a validation set which it also exposes with $(\varepsilon, \delta)$-DP.

2 Additional Figure to Section 5.5

Figure 1 shows the likelihoods of the test data as the learning progresses for the Gaussian mixture model of Section 5.5. The values of $\varepsilon$ are for $\delta = 10^{-6}$.

3 Application to Federated Learning

Federated learning [2] presents another setting where classical hyperparameter adaptation with a validation set may be impractical. 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.

3.1 Adaptive federated averaging algorithm

We consider the federated averaging algorithm described in [2]. 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 [2, Algorithm 1], a random subset of clients is considered at each aggregation. We consider the case $C = 1$ where each client participates in every aggregation, and replace the gradient step in client update with a non-private variant of Algorithm 2 given in the main text.

In [2], 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 pro-
Algorithm 1 Learning rate adaptive client update

\textbf{ClientUpdate}(k, \theta)

Split \mathcal{P}_k into batches of size \(|B|\).

\begin{algorithmic}
\For{each local step \(\ell = 1, \ldots, E\)}
\State Draw a batch \(B_1\) and evaluate at \(\theta_\ell\): \(G_1 \leftarrow \frac{1}{|B|} \sum_{i \in B_1} \nabla f_{\theta_\ell}(x_i)\).
\State Take a step size \(\eta_\ell\):
\State \(\theta_{\ell+1/2} \leftarrow \theta_\ell - \frac{\eta_\ell}{2} G_1\).
\State Take a step of size \(\eta_\ell\):
\State \(\hat{\theta}_{\ell+1} \leftarrow \theta_{\ell+1/2} - \frac{\eta_\ell}{2} G_1\).
\State Draw a batch \(B_2\), and evaluate at \(\theta_{\ell+1/2}\):
\State \(G_2 \leftarrow \frac{1}{|B|} \sum_{i \in B_2} \nabla f_{\theta_{\ell+1/2}}(x_i)\).
\State Take a step of size \(\eta_\ell\):
\State \(\tilde{\theta}_{\ell+1} \leftarrow \theta_{\ell+1/2} - \frac{\eta_\ell}{2} G_2\).
\State Evaluate: \(\text{err}_\ell \leftarrow \|\text{err}(\theta_{\ell+1}, \hat{\theta}_{\ell+1})\|_2\)
\If{\(\text{err}_\ell > \tau\)}
\State \(\theta_{\ell+1} \leftarrow \theta_\ell\) \hspace{1em} \text{(Discard step)}
\EndIf
\State update: \(\eta_{\ell+1} = \min(\max(\frac{\tau}{\text{err}_\ell}, \alpha_{\text{min}}), \alpha_{\text{max}}) \cdot \eta_\ell\).
\EndFor
\end{algorithmic}
Figure 2: CIFAR-10 test accuracies for the federated averaging algorithm after 200 communication rounds using ADADP for different initial learning rates $\eta_0$ and constant learning rate SGD for different $\eta$. The training data is interpolated between the pathological case and the uniformly random distribution of data. All points are averages of three runs.

Adam gave poor results in this example. Figure 4 shows the test accuracies in the interpolated case, 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 3a.

Figure 3: CIFAR-10 test accuracies for the federated averaging algorithm using ADADP and a learning rate tuned SGD, when the training data is interpolated between the pathological case and the uniformly random distribution of data.
References
