Revisiting Hyperparameter Tuning
with Differential Privacy

Youlong Ding[1]*   **Xueyang Wu**[2]
[1] Department of Computer Science and Software Engineering,
Shenzhen University, Shenzhen, China
[2] Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology, Hong Kong SAR, China
dingyoulon@gmail.com   xwuba@connect.ust.hk

Abstract

Hyperparameter tuning is a common practice in the application of machine learning but is a typically ignored aspect in the literature on privacy-preserving machine learning due to its negative effect on the overall privacy parameter. In this paper, we aim to tackle this fundamental yet challenging problem by providing an effective hyperparameter tuning framework with differential privacy. The proposed method allows us to adopt a broader hyperparameter search space and even to perform a grid search over the whole space, since its privacy loss parameter is independent of the number of hyperparameter candidates. Interestingly, it instead correlates with the utility gained from hyperparameter searching, revealing an explicit and mandatory trade-off between privacy and utility. Theoretically, we show that its additional privacy loss bound incurred by hyperparameter tuning is upper-bounded by the squared root of the gained utility. However, we note that the additional privacy loss bound would empirically scale like a squared root of the logarithm of the utility term, benefiting from the design of doubling step.

1   Introduction

Differential privacy [1, 2] has been the gold standard for quantitative and rigorous reasoning about privacy leakage from the processing of private data. Applying differential privacy to machine learning [3, 4, 5] is a long-lasting challenge due to the dramatic reduction in the model utility compared with the non-private version [6]. This motivates a bunch of work dedicated to designing private learning algorithms without sacrificing utility. However, researchers typically try different hyperparameters for best possible performance but only report the privacy parameter of a single run, which corresponds to the best accuracy achieved. As shown in [7], the choice of hyperparameter would cause leakage of private information such as membership inference [8]. This finding is aligned with the theory of differential privacy if applied strictly. Supposed one run of private learning to be ε-DP, if we repeat the learning process K times using different hyperparameters and select the model with the highest accuracy, the privacy loss parameter would be increased to $O(k\varepsilon)$ with basic composition or $\tilde{O}(\sqrt{k}\varepsilon)$ with advanced composition [9, 10], essentially a multiple of the original privacy loss ε. Several existing works attempt to handle such embarrassment. The stability-based approach [11] leverages the stability assumption of the learning algorithm for improving the privacy loss bounds. RandTune [12, 7] proposes to introduce another level of uncertainty for sharpening the privacy bounds. Concretely, it first draws K from a geometric distribution, then randomly and independently picks K hyperparameters, and runs a private training algorithm for each selected hyperparameter. Then the total privacy parameter is shown to be bounded by 2 or 3 times of the

---

*This work is done during his internship in WeBank Ltd. Co., Shenzhen, China

privacy parameter for a single run. However, the largest challenge in RandTune is to guarantee the success probability of picking the best hyperparameter within only K independent trials, which gets more difficult when the search space of hyperparameters is large. Therefore, a recent work [13] leverages adaptive optimizers to reduce the potential hyperparameter space. Nevertheless, the number of trials remains to be unpredictable, and therefore it is still hard to guarantee the model quality.

In this paper, we propose a framework for hyperparameter tuning with differential privacy, which 1) satisfies rigorous guarantees with differential privacy and 2) allows us to perform a grid search over all candidate hyperparameters. To achieve that, we enable the overall privacy loss to be independent of both hyperparameter search space and the original privacy parameter of DP-SGD. It instead depends on the final utility of the model (i.e., accuracy on the validation set), which is an interesting property because it reveals an explicit and mandatory trade-off between privacy and utility. Additionally, our approach does not require each training run for hyperparameter selection to be differentially private. This property can be utilized for efficient tuning of hyperparameters whose behavior is consistent between private and non-private versions. Comparisons between different methods are in Table 1.

Table 1: Comprehensive comparison between different methods for hyperparameter tuning with DP.

Method	Full search	# runs	Priv. run per search	Privacy budget
Naive	Yes	$|\mathcal{S}|$	Required	$\tilde{O}(\sqrt{|\mathcal{S}|}\varepsilon)$
RandTune[7, 12]	No	$K \sim \mathcal{D}^\star$	Required	2ε or 3ε
Ours	Yes	$|\mathcal{S}|$	Not required	$\varepsilon + \tilde{O}(\sqrt{\frac{u^* - u_0}{g}})\varepsilon_0$

$\star$: K is drawn from a Truncated Negative Binomial Distribution or a Poisson distribution.

2 Propose-test Hyperparameter Tuning with Doubling Step

Our method described in Algorithm 1 is inspired by a bunch of classical algorithms. Specifically, we inherit the AboveThreshold component (line 18) of Sparse Vector Techinique [14] to check whether the current candidate is eligible for utility accumulation. We also apply Subsample and Aggregate [15] (lines 3-11) to help us obtain a proxy utility function with relatively low sensitivity (also used by PATE [16, 17]). To potentially decrease the number of iterations (and thus the privacy loss parameter), we use the doubling algorithms (lines 22 and 27) originally designed to solve Lowest-Common-Ancestor (LCA) in the tree. Note that the naive combination of doubling and randomized algorithm may cause the termination problem. To accommodate this, we add an additional threshold checking (line 16) to ensure termination. The overall privacy guarantee is as follows.

Theorem. *The algorithm 1 guarantees* $(\varepsilon + \tilde{O}(\sqrt{\frac{u^* - u_0}{g}})\varepsilon_0, \delta)$-*differential privacy for all* ε_0, $\varepsilon_1 > 0$, $\delta \in (0, 1)$, $\mu_0 \in [0, 1)$, *and* $g \in (0, 1)$.

Proof. The execution of the main loop (line 12) can be treated as running a sequence of procedures $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_t, ..., \mathcal{M}_T$, where $\mathcal{M}_t : \mathcal{D}_{train}, \mathcal{S}, \{u_s^{(t)}\}, u^{(t)} \to s^{(t)}, u^{(t+1)}$. Fix any two neighbouring training set $\mathcal{D}$ and $\mathcal{D}'$, and let the outputs on them (with the same set of hyperparameters $\mathcal{S}$, $\{u_s^{(t)}\}$, and $u^{(t)}$) be $\mathcal{A}$ and $\mathcal{A}'$, respectively. We first prove that every single mechanism $\mathcal{M}_t$ is differential private, and the result follows from the advanced composition.

Firstly, we bound the ℓ_1 sensitivity of $u_s^{(t)}(\mathcal{D})$, which is the maximum change in ℓ_1 norm caused by adding or removing one sample from $\mathcal{D}$. Observe that the output accuracy score for each split is bounded by 1, and therefore, the maximum change caused by one split is at most $1/k$. Secondly, suppose the output $\mathcal{A}$ is $s^{(t)} = k$, we define $u_{\mathcal{S}}(\mathcal{D}) = \max_{s < k}(u_s^{(t)}(\mathcal{D}) + \gamma_s^{(t)})$, representing the maximum noisy utility of all hyperparameters tried on $\mathcal{D}$. We then fix the values of $\{\gamma_s^{(t)}|s < k\}$. That is, we assume the two runs on $\mathcal{D}$ and $\mathcal{D}'$ share the same value of noise assigned for the corresponding hyperparameter candidate's utility. Note that although this will weaken the privacy protection effect (but easy for analysis) since we somewhat reduce the amount of the uncertainty underlying the algorithm, as we will show later, it is still sufficient to obtain the desired privacy loss bound. After fixing, the randomness on the output is over $\tilde{u}^{(t)}$ and $\gamma_k^{(t)}$. The probability that $\mathcal{M}^{(t)}$ on $\mathcal{D}$ outputs

Algorithm 1: Propose-test Hyperparameter Tuning with Doubling Step

---

Input: Hyperparameter candidates set $\mathcal{S}$; Privacy parameters $\varepsilon, \varepsilon_0 > 0, \delta \in (0, 1)$; Training set
$\mathcal{D}_{train}$; Validation set $\mathcal{D}_{valid}$; Number of partitions k; Utility granularity $g \in (0, 1)$;
Utility lower bound $u_0 \in [0, 1)$.

Output: Model parameters θ^* along with the selected hyperparameters $s^* \in \mathcal{S}$.

---

1 Initialize $s^* \leftarrow \emptyset$, $u \leftarrow u_0$, $step \leftarrow 1$;
2 Initialize $count \leftarrow 0$;   // Help variable for analysis
3 Partition the training set $\mathcal{D}_{train}$ into disjoint subsets $\mathcal{D}_i$, $k = 1, 2, ..., k$;
4 **foreach** $s = 1, 2, ..., |\mathcal{S}|$ **do**
5   Initialize $u_s \leftarrow 0$;
6   **foreach** $i = 1, 2, ..., k$ **do**
7        $\theta \leftarrow \text{Train}(\mathcal{D}_i, \mathcal{S}_s)$ or $\text{PrivateTrain}(\mathcal{D}_i, \mathcal{S}_s, \varepsilon, \delta)$;
8        $u_s^{(i)} \leftarrow \text{Acc}(\mathcal{D}_{valid}, \theta)$;   // Compute utility for each partition
9   **end**
10    $u_s \leftarrow \frac{1}{k} \sum_{i=1}^{k} u_s^{(i)}$;   // Compute average utility across partitions
11 **end**
12 **while** $step \neq 0$ **do**
13    $count \leftarrow count + 1$
14   Let $\hat{u} \leftarrow u + step \times g + \text{Lap}(2/(k\varepsilon_0))$;   // Propose a utility threshold
15    $isAccumulated \leftarrow$ **False**;
16   **foreach** $s = 1, 2, ..., |\mathcal{S}|$ **do**
17       $\gamma_s \leftarrow \text{Lap}(4/(k\varepsilon_0))$;   // Sample noise
18   **if** $u_s + \gamma_s \geq \hat{u}$ **then**
19          $s^* \leftarrow s$;   // Record the current hyperparameter
20          $u \leftarrow u + step \times g$;   // Accumulate utility
21          $isAccumulated \leftarrow$ **True**;
22          $step \leftarrow step \times 2$;   // Double-expand current step
23   **break;**
24   **end**
25   **end**
26   **if** *not isAccumulated* **then**
27       $step \leftarrow \lfloor step/2 \rfloor$;   // Double-compress current step
28   **end**
29   **if** $u \geq 1$ **then**
30   **break** ;   // Required for termination guarantee
31   **end**
32 **end**
33 $\theta^* \leftarrow \text{PrivateTrain}(\mathcal{D}_{train}, S_{s^*}, \varepsilon, \delta)$;   // Using the selected hyperparameter

---

$\mathcal{A}^{(t)}$ can be bounded as follows. Let $\Delta u_\mathcal{S} = u_\mathcal{S}(\mathcal{D}) - u_\mathcal{S}(\mathcal{D}')$, $\Delta u_k^{(t)} = u_k^{(t)}(\mathcal{D}') - u_k^{(t)}(\mathcal{D})$, then

$$
\begin{aligned}
p(\mathcal{A}^{(t)} = \{s^{(t)}, u^{(t)}\}) &= p(u_\mathcal{S}(\mathcal{D}) < \tilde{u}^{(t)} \leq u_k^{(t)}(\mathcal{D}) + \gamma_k^{(t)}) \\
&= \int_{\tilde{u}} \int_{\gamma_k} p(\tilde{u}^{(t)} = \tilde{u}) \cdot p(\gamma_k^{(t)} = \gamma_k) \cdot \mathbf{1}[u_\mathcal{S}(\mathcal{D}) < \tilde{u} \leq u_k^{(t)}(\mathcal{D}) + \gamma_k] d\gamma_k d\tilde{u} \\
&= \int_{\tilde{u}} \int_{\gamma_k} p(\tilde{u}^{(t)} = \tilde{u} + \Delta u_\mathcal{S}) \cdot p(\gamma_k^{(t)} = \gamma_k + \Delta u_\mathcal{S} + \Delta u_k^{(t)}) \\
&\quad \cdot \mathbf{1}[u_\mathcal{S}(\mathcal{D}) < \tilde{u} + \Delta u_\mathcal{S} \leq u_k^{(t)}(\mathcal{D}) + (\gamma_k + \Delta u_\mathcal{S} + \Delta u_k^{(t)})] d\gamma_k d\tilde{u} \\
&\leq \int_{\tilde{u}} \int_{\gamma_k} e^{\varepsilon_0/2} p(\tilde{u}^{(t)} = \tilde{u}) \cdot e^{\varepsilon_0/2} p(\gamma_k^{(t)} = \gamma_k) \cdot \mathbf{1}[u_\mathcal{S}(\mathcal{D}') < \tilde{u} \leq u_k^{(t)}(\mathcal{D}') + \gamma_k] d\gamma_k d\tilde{u} \\
&\leq e^{\varepsilon_0} p(\mathcal{A}^{'(t)} = \{s^{(t)}, u^{(t)}\})
\end{aligned}
\tag{1}
$$

Therefore, each mechanism $\mathcal{M}_t$ is ε_0-differential private. In the following, we bound T, which is the
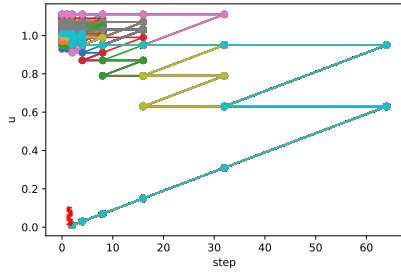total number of mechanisms in the sequence, i.e., the number of iterations. Although it is difficult

Figure 1: Visulaization of $step$-u line over each iteration (starting from left bottom, ending at left top) for 1000 different random seeds (shown in color, except for red).

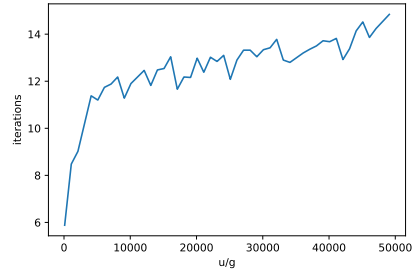
Figure 2: Plot of #iterations versus $\frac{u^*-u_0}{g}$. As can be seen, in practice, it is nearly a logarithmic dependency.

to analyze the behavior of the doubling step in the randomized setting, we fortunately find that the worst-case scenario is easy to attain. If we record the value of the variable $isAccumulated$ over each iteration, the outcome sequence in the worst case will be

$$\{\text{True}, \text{False}, \text{True}, \text{False}, \text{True}, \text{False}, ...\} \tag{2}$$

This behavior corresponds to the zig-zag phenomenon in the $step$-u plot (See the top part in Figure 1). In the worst case, the utility only gains g every two iterations from the beginning, and we know that the utility is upper-bounded by 1. Thus, the total number of iterations can be bounded by $O(\frac{u^*-u_0}{g})$. Using advanced composition, the additional privacy loss is bounded by $\tilde{O}(\sqrt{\frac{u^*-u_0}{g}})\varepsilon_0$. Finally, we run the DP-SGD using the best parameter with the privacy parameter of (ε, δ). From basic composition, the total privacy parameter of the algorithm is thus $(\varepsilon + \tilde{O}(\sqrt{\frac{u^*-u_0}{g}})\varepsilon_0, \delta)$. □

Discussion. We have adopted a worst-case analysis to bound T, i.e., the number of iterations. Therefore, the privacy loss bound is upper-bounded by the final result. However, benefiting from the incorporation of the doubling step, the empirical privacy loss bound is even better. The intuition is that the doubling step would bound T by $O(\log_2 \frac{u^*-u_0}{g})$, if the algorithm was deterministic. We then run a simulation to show the actual privacy loss bound. The number of partitions k is set to 10. The additional privacy parameter ε_0 is set to 0.1. We set the number of hyperparameters to 100 and randomly draw 100 samples from the uniform distribution over $(0, 1)$, corresponding to each u_s. We set g to 0.01. Figure 1 visualizes the doubling step process. Each point $(u, step)$ represents the state after an iteration. $(u, step)$ will start at $(0, 1)$ at the beginning, and after each iteration, it will either move left by a half or move right-top by $(step, u + step))$. The algorithm terminates once it reaches the line of $step = 0$ or $u = 1$. The red short line at the left bottom shows the behavior of the upper bound within the same iterations. We can see that the utility accumulates exponentially at the beginning and converges fast w.h.p (over 99.9% here). We then vary the value of $\frac{u^*-u_0}{g}$ to plot the corresponding empirical bound for T in Figure 2. Its dependency is almost logarithmic with $\frac{u^*-u_0}{g}$.

3   Conclusion

In this paper, we propose an ML algorithm-agnostic framework for hyperparameter tuning with differential privacy under rigorous privacy guarantees. Compared to existing differentially private hyperparameter tuning methods that suffer from large hyperparameter search space, our additional privacy loss parameter is free from the size of the hyperparameter candidates set and the original privacy parameter of DP-SGD. Instead, it correlates with the final utility of the tuned model and is upper-bounded by the squared root of the utility term. Therefore, it allows us to perform hyperparameter tuning on a larger range, even with a grid search, leading to potentially higher utility. We believe that our work would be meaningful in the field of privacy-preserving machine learning, and would be valuable for future research in this area.

References

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

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

[3] Song, S., K. Chaudhuri, A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pages 245–248. IEEE, 2013.

[4] Bassily, R., A. Smith, A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. 2014.

[5] Abadi, M., A. Chu, I. J. Goodfellow, et al. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. 2016.

[6] Tramèr, F., D. Boneh. Differentially private learning needs better features (or much more data). In *9th International Conference on Learning Representations*. 2021.

[7] Papernot, N., T. Steinke. Hyperparameter tuning with renyi differential privacy. In *International Conference on Learning Representations*. 2022.

[8] Shokri, R., M. Stronati, C. Song, et al. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy*, pages 3–18. 2017.

[9] Dwork, C., G. N. Rothblum, S. Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[10] Kairouz, P., S. Oh, P. Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

[11] Chaudhuri, K., S. A. Vinterbo. A stability-based validation procedure for differentially private machine learning. *Advances in Neural Information Processing Systems*, 26, 2013.

[12] Liu, J., K. Talwar. Private selection from private candidates. STOC 2019, page 298–309. New York, NY, USA, 2019.

[13] Mohapatra, S., S. Sasy, X. He, et al. The role of adaptive optimizers for honest private hyperparameter selection. pages 7806–7813. AAAI Press, 2022.

[14] Dwork, C., M. Naor, O. Reingold, et al. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390. 2009.

[15] Nissim, K., S. Raskhodnikova, A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. 2007.

[16] Papernot, N., M. Abadi, Ú. Erlingsson, et al. Semi-supervised knowledge transfer for deep learning from private training data. In *Proceedings of 5th International Conference on Learning Representations*. 2017.

[17] Papernot, N., S. Song, I. Mironov, et al. Scalable private learning with PATE. In *Proceedings of 6th International Conference on Learning Representations*. 2018.