PATE writeup

July 23, 2019

1 Outline

This writeup aims to explain the PATE algorithm from the 2018 Papernot et al. paper "Scalable Private Learning with PATE" [1]. All mentions of sections, theorems etc. in the following refer to this paper unless otherwise noted.

The privacy analysis of the PATE algorithm consists of 2 parts, which will be discussed separately. Part 1 provides a *data-dependent* privacy analysis of the GNMax algorithm and its variants in appendix A. Because this privacy analysis is data-dependent, the resulting epsilon cannot be released without compromising privacy. Thus the second part, in appendix B, uses a smooth sensitivity analysis of this epsilon value in order to provide an epsilon which can be released.

2 Data-dependent privacy analysis

One iteration of the GNMax algorithm as defined in section 4.1 perturbs each bin of the histogram containing predictions from the teacher ensemble. Then the label with the most noisy votes is released. This simple histogram release is Renyi Differetially Private (RDP) and the common privacy analysis is stated in proposition 8 and used as a baseline. Since each bin is now a Gaussian random variable, it is straightforward to compute the probability that a specific bin in a given histogram is not chosen as the max value. For the likely outcome, i.e. the largest unperturbed bin, this data-dependent quantity is called q in proposition 7.

The major insight lies in theorem 6, which states that a data-dependent RDP guarantee can be derived from a mechanism satisfying data-independent RDP for the values (μ_1, ε_1) and (μ_1, ε_1) and information about the quantity q. An analysis of this proof will be added later. Given the tradeoff between μ and ε, most RDP mechanisms fulfil RDP for many different value pairs. Proposition 10 derives, which orders μ_1 and μ_2 must be chosen to minimize the derived data-dependent RDP via theorem 6. At each histogram release, both the data-independend and the data-dependent RDP bound are computed and the smaller value is chosen, leading to the definition of $\boldsymbol{\beta}_\sigma(q)$ on page 21 before corollary 11. (This is done for a number of candidate orders λ. The optimal order is chosen

later) Over multiple histogram releases, these values are added up using RDP composition from theorem 4.

Finally, the RDP guarantees for different orders can be converted into (ϵ, δ) guarantees for a selected δ using theorem 5 and the minimum can be chosen. However, this privacy guarantee is itself data-dependent if the data-dependent bound was used at some point and thus cannot be released without compromising privacy. Deriving an epsilon value which can be released is the topic of the next section.

3 Releasing epsilon via smooth sensitivity analysis

The results of the data-dependent sensitivity analysis in theorem 6 are released by bounding their smooth sensitivity (SS), given in definition 12. In order to compute $SS(\bar{n})$, one must be able to compute the worst case local sensitivity $LS(\bar{n}')$ for a histogram $\bar{n}'$ at distance d from $\bar{n}$. Most of this section is dedicated to deriving, which histograms $\bar{n}'$ maximize these local sensitivities, as, after finding them, one can simply iterate all distances d to compute $SS(\bar{n})$.

The proofs in appendices B.1 to B.3 rely on new notation and a number of conditions stated on page 24 which are (mostly) proven in B.4. Most important for understanding the following section are conditions C5 and C6. C5 introduces the point q_0 and requires that $\boldsymbol{\beta}(q)$ (previously $\boldsymbol{\beta}_\sigma(q)$) be non-decreasing in $[0, q_0]$ and constant in $[q_0, 1]$. The intuition here is that if $q_0 \leq q$, the data-dependent privacy cost exceeds the data-independent one and thus $\boldsymbol{\beta}(q) = \lambda/\sigma^2$ is constant in this interval. Further, C6 requires that $\Delta\boldsymbol{\beta}(q) = \boldsymbol{\beta}(\boldsymbol{B}_U(q)) - \boldsymbol{\beta}(q)$ is non-decreasing in $[0, q_1]$ where $q_1 = \boldsymbol{B}_L(q_0)$. $\boldsymbol{B}_U(q)$ and $\boldsymbol{B}_U(q)$ are defined to be the upper and lower bound on $q(\bar{n}')$ where $\bar{n}'$ is a neighbouring histogram of $\bar{n}$ and $q = q(\bar{n})$.

From these definitions, the calculation of the local sensitivity upper bound is straightforward as

$$LS(q) \leq max\Big\{\Delta\boldsymbol{\beta}(q), \Delta\boldsymbol{\beta}(\boldsymbol{B}_L(q))\Big\} \tag{1}$$

with the first term peaking at q_1 and the second term reaching the same maximum at q_0. Algorithm 3 computes $\tilde{LS}(q)$, a looser upper bound on LS, which outputs $\Delta\boldsymbol{\beta}(q_1)$ when $q_1 \leq q \leq q_0$. The validity of this bound (and thus the correctness of algorithm 3) is shown in proposition 13[1]. As a result, $\tilde{LS}(q)$ is non-decreasing in $[1, q_1]$, constant in $[q_1, q_0]$ and non-increasing in $[q_0, 1]$ (proposition 14). This becomes relevant in algorithm 4.

The next important result comes in proposition 19, which, given a histogram $\bar{n}$ with certain properties, provides maximally different histograms $\bar{n}^*$ and $\bar{n}^{**}$

[1] I think it might be possible to complete the analysis without relaxing the bound in this way, but things do become more complicated and more computation-intensive. Maybe there's room for a small improvement here

at distance d such that $q(\bar{n}^*) \geq q(\bar{n}')$ and $q(\bar{n}^{**}) \leq q(\bar{n}')$ for all $\bar{n}'$ at distance d from $\bar{n}$. With this result, it becomes possible to construct bounds on $\tilde{LS}(q(\bar{n}'))$ for histograms at a given distance from $\bar{n}$. Now algorithm 4 can be explained:

Algorithm 4 upper bounds the local sensitivity of histograms at distance d from $\bar{n}$. It distinguishes between 3 cases based on the shape of $\tilde{LS}$, which has been discussed above:

Case 1: $q_1 \leq q(\bar{n}) \leq q_0$ (lines 3-5)

- In this case $\tilde{LS}(q(\bar{n}))$ is already maximal and can't be increased.

Case 2: $q(\bar{n}) \leq q_1$ (lines 6-17) $\tilde{LS}(q(\bar{n}))$ is increased by increasing q.

- Use proposition 19.1 to find the maximal $q(\bar{n}^*)$ for $\bar{n}^*$ at distance d.
- If prop 19.1 is not applicable return the data-independent bound $\tilde{LS}(q_1)$ and stop. (line 8)
- If $q(\bar{n}^*) > q_1$, larger d will not increase this bound, so return $\tilde{LS}(q_1)$ and stop (line 13).
- Otherwise, return $\tilde{LS}(q(\bar{n}^*))$ and continue searching for larger d (line 15).

Case 3: $q_0 \leq q(\bar{n})$ (lines 18-35) $\tilde{LS}(q(\bar{n}))$ is increased by decreasing q.

- Use proposition 19.2 to find the minimal $q(\bar{n}^{**})$ for $\bar{n}^{**}$ at distance d.
- If prop 19.2 is not applicable, $\bar{n}^{**}$ is the histogram with all votes in one bin and $q(\bar{n}^{**})$ can't be decreased further. Return $\tilde{LS}(q(\bar{n}^{**}))$ and stop. (lines 19-22)
- If $q(\bar{n}^{**}) < q_0$, smaller d will not increase this bound, so return $\tilde{LS}(q_0)$ and stop (line 30).
- Otherwise, return $\tilde{LS}(q(\bar{n}^{**}))$ and continue searching for larger d (line 32).

This goes together with algorithm 5, which calls algorithm four for increasing d until the return argument tells it to stop and then returns the maximum over discounted local sensitivities as the smooth sensitivity bound. This is made explicit in theorem 20.

Now that smooth sensitivities for individual iterations can be computed, Theorem 24 gives us a straightforward way of aggregating them by tracking local sensitivity bounds for all values of $d \in [0, \#teachers]$ and choosing the discounted maximum over d at the end. (Note that algorithm 5 is therefore not actually used in the exact way in which it is described.)

3.1 the GNSS mechanism

Now that the smooth sensitivity of epsilon has been derived, it can be used to release a noisy version of it by adding Gaussian noise scaled by $SS_\beta(D)$, which is introduced as the GNSS mechanism in definition 22. This process has an additional privacy cost stated in theorem 23 and so the final release privacy guarantee is the sum of the noisy GNSS release and the cost of GNSS. Note that this value has a non-zero chance of being lower than the actual data-dependent RPD bound and may thus be too optimistic. The variance estimates provided in table 2 on page 34 do however suggest that this event is fairly unlikely.

4 Notes on the application of PATE

- parameter search is necessary for β, up to three σ (threshold, GNMax, epsilon release)
- accounting for the threshold mechanism in confident and interactive GNMax is not given but functions analogously...
- Since epsilon must be released, targeting a specific privacy budget is tough. One may use a more pessimistic data-independent analysis as a guideline, for instance. Other valid stop conditions are maximal numbers of labeled or seen datapoints.
- One odd feature of the provided implementation is that currently the ThresholdLocalSensitivity algorithm returns 0 for distances d which are greater than $\max(v^{(1)}, \tau - v^{(1)})$, so the local sensitivity is not increasing in d, which it should in my opinion be, as theorem 24 takes a max over distances $\leq d$. However, fixing this does not actually change the empirical results of the analysis. (Maybe there is some reasoning here about the use of data-independent sensitivity, but I'm not sure it makes sense to me.)

4.1 Application Walkthrough

Let's assume we want to use the **confident GNMax** mechanism.

Before applying the PATE privacy analysis for, specify the following quantities:

- τ: the number of teacher models
- $n : \mathcal{X} \to \bar{\mathcal{N}}$: the function mapping data to teacher votes
- $\lambda_{1:k}$: a list of candidate orders, for which RDP is calculated
- $\sigma_{\text{threshold}}, \sigma_{\text{gnmax}}, \sigma_{gnss}$: standard deviation of noise added at thresholding, GNMax and GNSS mechanism.
- T: a threshold on the necessary confidence of the teachers. (I.e., the consensus should have at least T votes)

- δ: the target for converting RDP into (ε, δ)-DP at the end.
- m: the number of labels (not an independently chosen quantity of course)

The Following algorithm, based on the implementation accompanying the paper, releases a set of labeled datapoints and the (approximate) associated privacy cost.

4.2 Pseudocode of Confident GNMax

Algorithm 1 The confident GNMAx algorithm with PATE analysis

procedure ConfidentGNMax(args)
  $e_{1:k} \leftarrow \mathbf{0}$ ▷ accumulates ε RDP costs for orders $\lambda_{1:k}$
  $S \leftarrow \emptyset$ ▷ stores ($\bar{n}$, accept/reject) tuples for GNSS algorithm
  $V \leftarrow \emptyset$ ▷ labeled data tuples to be released at the end
  for all $x_i \in X$ **do**
    $q \leftarrow \text{ThresholdQ}(n(x_i), \sigma_{\text{threshold}}, T)$
    $e_{1:k} \leftarrow e_{1:k} + \text{DataDependentCost}(\lambda_{1:k}, \sqrt{2}\sigma_{\text{threshold}}, q)$
    if $\max_j\{n(x_i)_j\} + \mathcal{N}(0, \sigma_{\text{threshold}}) \geq T$ **then**
      $q \leftarrow \text{GNMaxQ}(n(x_i), \sigma_{\text{gnmax}})$
      $e_{1:k} \leftarrow e_{1:k} + \text{DataDependentCost}(\lambda_{1:k}, \sigma_{\text{gnmax}}, q)$
      $S \leftarrow S + \{(n(x_i)), \text{released}\}$
      $V \leftarrow V + \{(x_i, \arg\max_j\{(n(x_i) + \mathcal{N}(\mathbf{0}, \sigma_{\text{gnmax}}))_j\})\}$
    else
      $S \leftarrow S + \{(n(x_i)), \text{rejected}\}$
    if StopCondition then
      break
  $\omega \leftarrow \text{ChooseOrder}(e_{1:k}, \lambda_{1:k}, \delta)$
  $\varepsilon_{\text{gnss}}, s_\beta \leftarrow \text{GNSS}(S, \beta, T, \sigma_{\text{threshold}}, \sigma_{\text{gnmax}}, \sigma_{\text{gnss}}, \lambda_\omega, \tau, m)$
  $\tilde{e}_\omega \leftarrow e_\omega + \mathcal{N}(0, \sigma_{\text{gnss}} * s_\beta)$
  $\varepsilon_{RDP} \leftarrow \tilde{e}_\omega + \varepsilon_{gnss}$
  $\varepsilon_{\delta DP} \leftarrow \text{RDPtoDDP}(\varepsilon_{RDP}, \lambda_\omega, \delta)$
  return $V, \varepsilon_{\delta DP}$

Algorithm 2 auxiliary functions

procedure GNMaxQ($\bar{n}, \sigma$)
  $i^* = \arg\max(\bar{n})$
  return $\frac{1}{2}\sum_{i \neq i^*} \text{erfc}\left(\frac{\bar{n}_{i^*} - \bar{n}_i}{2\sigma}\right)$ ▷ Prop. 7
procedure ThresholdQ($\bar{n}, \sigma, T$)
  $p_{accept} \leftarrow \frac{1}{2}\text{erfc}\left(\frac{T - \max(\bar{n})}{\sqrt{2}\sigma}\right)$
  return $\min(p_{accept}, 1 - p_{accept})$
procedure DataDependentCost($\lambda_{1:k}, \sigma, q$)
  $\mu_2 \leftarrow \sigma\sqrt{-\log(q)}$ ▷ Prop. 10
  $\mu_1 \leftarrow \mu_2 + 1$
  $\varepsilon_1, \varepsilon_2 \leftarrow \frac{\mu_1}{\sigma^2}, \frac{\mu_2}{\sigma^2}$ ▷ Prop. 8
  for $i \leftarrow 1, k$ **do** ▷ Thm. 6
    if $\lambda_i > \mu_1$ or $q > \exp\{(\mu_2 - 1)\varepsilon_2\} / \left(\frac{\mu_1}{\mu_1 - 1} \cdot \frac{\mu_2}{\mu_2 - 1}\right)^{\mu_2}$ **then**
      $e_i \leftarrow \frac{\lambda_i}{\sigma^2}$
    else
      $A \leftarrow (1 - q) / \left(1 - (q \cdot e^{\varepsilon_2})^{\frac{\mu_2 - 1}{\mu_2}}\right)$
      $B \leftarrow e^{\varepsilon_1} / q^{\frac{1}{\mu_1 - 1}}$
      $e_i \leftarrow \min\left\{\frac{1}{\lambda_i - 1}\log\left((1 - q) \cdot A^{\lambda_i - 1} + q \cdot B^{\lambda_i - 1}\right), \frac{\lambda_i}{\sigma^2}\right\}$
  return $e_{1:k}$
procedure ChooseOrder($e_{1:k}, \lambda_{1:k}, \delta$)
  return $\arg\min_{1 \leq i \leq k}\{\text{RDPtoDDP}(e_i, \lambda_i, \delta)\}$
procedure RDPtoDDP($\varepsilon_{RDP}, \lambda, \delta$)
  return $\varepsilon_{RDP} + \frac{\log(1/\delta)}{\lambda - 1}$

Algorithm 3 Data-dependent epsilon release

procedure GNSS($S, \beta, T, \sigma_{\text{threshold}}\sigma_{\text{gnmax}}, \sigma_{gnss}, \lambda, \tau, m$)
  $ls'_{1:\tau} \leftarrow \mathbf{0}$
  for all $(v, \text{decision}) \in S$ **do**
    $ls_{1:\tau} \leftarrow ls_{1:\tau} + \text{ThresholdLocalSensitivity}(v, \tau, \sigma_{\text{threshold}}, \lambda, m, T)$
    if decision = release **then**
      $ls_{1:\tau} \leftarrow ls_{1:\tau} + \text{GNMaxLocalSensitivity}(v, \tau, \sigma_{\text{gnmax}}, \lambda, m)$
  $s_\beta \leftarrow \max_{1 \leq d \leq \tau}\{\exp(-\beta d) \cdot ls_d\}$ ▷ Thm. 24
  assert $1 < \lambda < 1/(2\beta)$ ▷ Thm. 23
  $\varepsilon_{gnss} \leftarrow \frac{\lambda \cdot \exp(2\beta)}{\sigma^2} + \frac{\beta\lambda - 0.5\log(1 - 2\lambda\beta)}{\lambda - 1}$
  return $\varepsilon_{gnss}, s_\beta$

Algorithm 4 local sensitivity bounds

procedure GNMaxLocalSensitivity(v, τ, σ, λ, m) ▷ $\approx$ Alg. 4+5
  $q \leftarrow \text{GNMaxQ}(v, \sigma)$
  $q_0 \leftarrow \text{FindQ}_0(\sigma, \lambda)$ ▷ found numerically, not covered here
  $q_1 \leftarrow \text{B}_\text{L}(q_0, \sigma, m)$
  $ls_{1:\tau} \leftarrow \tilde{\text{LS}}(q_1, q_0, q_1, \lambda, \sigma, m)$
  if $q_1 \leq q \leq q_0$ **then**
    return $ls_{1:\tau}$
  $v^{(1)} \leftarrow \tilde{\text{LS}}(q, q_0, q_1, \lambda, \sigma, m))$ ▷ $v^{(n)}$: n-th largest bin
  $d \leftarrow 0$
  go-left $\leftarrow q > q_0$
  while (go-left $\wedge\, q > q_0 \wedge v^{(2)} > 0) \vee (\neg$go-left $\wedge\, q < q_1)$ **do**
    $d \leftarrow d + 1$
    if go-left **then**
      $v^{(1)} \leftarrow v^{(1)} + 1$
      $v^{(2)} \leftarrow v^{(2)} - 1$
    else
      $v^{(1)} \leftarrow v^{(1)} - 1$
      $v^{(2)} \leftarrow v^{(2)} + 1$
    $q \leftarrow \text{GNMaxQ}(v, \sigma)$
    $ls_d \leftarrow \tilde{\text{LS}}(q, q_0, q_1, \lambda, \sigma, m)$
  return $ls_{1:\tau}$
procedure ThresholdLocalSensitivity($v, \tau, \sigma, \lambda, m, T$)
  $q \leftarrow \text{ThresholdQ}(v, \sigma)$
  $ls_{1:\tau} \leftarrow \mathbf{0}$
  for $d \leftarrow 1, \max(v^{(1)}, \tau - v^{(1)})$ **do**
    $ls_{up}, ls_{down} \leftarrow -\infty$
    if $\max(v^{(1)} + d \leq \tau$ **then**
      $ls_{up} \leftarrow \text{ThreshLS}(v^{(1)} + d, \tau, \lambda, \sigma, m, T)$
    if $\max(v^{(1)} - d \geq 0$ **then**
      $ls_{down} \leftarrow \text{ThreshLS}(v^{(1)} - d, \tau, \lambda, \sigma, m, T)$
    $ls_d \leftarrow \max(ls_{up}, ls_{down})$
  return $ls_{1:\tau}$

Algorithm 5 local sensitivity helpers

procedure $\tilde{\mathrm{LS}}(q, q_0, q_1, \lambda, \sigma, m)$ ▷ Alg. 3
  if $q_1 \le q \le q_1$ **then**
    $q \leftarrow q_1$
  $c \leftarrow \mathrm{DataDependentCost}(\lambda_{1:1}, \sigma, q)$
  $c_u \leftarrow \mathrm{DataDependentCost}(\lambda_{1:1}, \sigma, \mathrm{B_U}(q, \sigma, m))$
  $c_l \leftarrow \mathrm{DataDependentCost}(\lambda_{1:1}, \sigma, \mathrm{B_L}(q, \sigma, m))$
  return $\max\{c_u - c, c - c_l\}$

procedure $\mathrm{B_U}(q, \sigma, m)$
  return $\min\left\{\frac{m-1}{2}\mathrm{erfc}\left(\mathrm{erfc}^{-1}\left(\frac{2q}{m-1}\right) - \frac{1}{\sigma}\right), 1\right\}$

procedure $\mathrm{B_L}(q, \sigma, m)$
  return $\frac{m-1}{2}\mathrm{erfc}\left(\mathrm{erfc}^{-1}\left(\frac{2q}{m-1}\right) + \frac{1}{\sigma}\right)$

procedure ThreshLS$(h, \tau, \lambda, \sigma, m, T)$
  for $i \leftarrow 0, \tau$ **do** ▷ treat i as a 1-bin histogram
    $c_i \leftarrow \mathrm{DataDependentCost}(\lambda_{1:1}, \sigma, \mathrm{ThresholdQ}(i, \sigma, T))$
  if $h = 0$ **then**
    return $|c_{h+1} - c_h|$
  if $h = \tau$ **then**
    return $|c_{h-1} - c_h|$
  return $\max(|c_{h+1} - c_h|, |c_{h-1} - c_h|)$

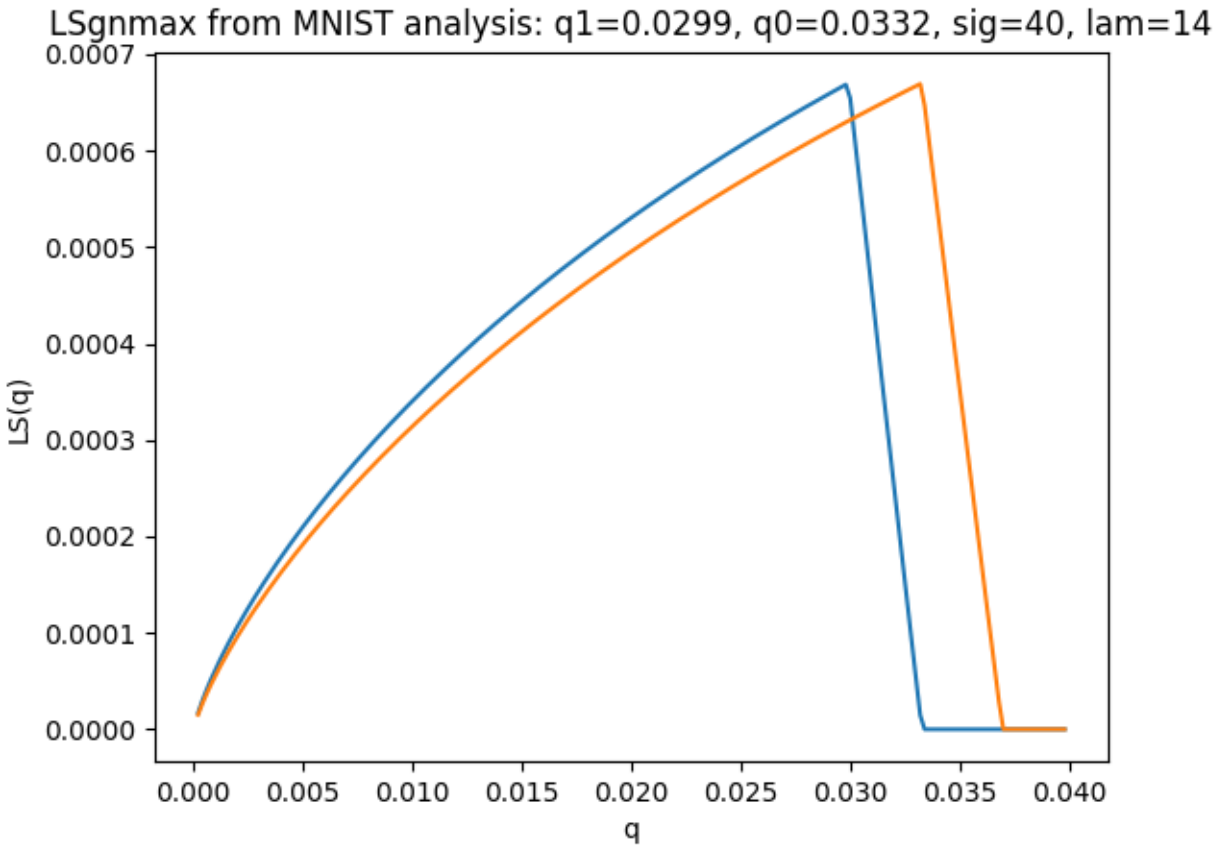

Figure 1: Illustration of the $\tilde{\text{LS}}$ function for parameters from the mnist validation data, plotting $c_u - c$ in blue and $c - c_l$ in orange. This shows that the choosing to linearize in the interval between q_1 and q_0 doesn't overestimate the actual values by much.

References

[1] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson. "Scalable Private Learning with PATE". In: *International Conference on Learning Representations*. 2018.