Machine Learning **Spring 2021**

Lecture 13: Differential Privacy, Boosting

Lecturer: Liwei Wang *Scribe: Yutong Yin, Haowei Lin, Yuhan Guo, Jingwu Tang, Siyuan Chen*

13.1 Local Differential Privacy

In the past decades, Differential Privacy, which gives strong privacy guarantees for users has risen to popularity. The key idea is that a user is given plausible deniability by adding random noise to their input. We've talked about mechanisms like Laplace mechanism and BLR mechanism to preserve both privacy and accuracy well. However, the setting we have discussed in is *centralized Differential Privacy* setting, where noise is added to the database. This approach to Differential Privacy requires users to have trust the database maintainer to keep their privacy because all the raw personal data is collected by the dataholder.

A stronger privacy guarantee for each individual users can be given in the local setting as there is no need to trust a centralized authority. In Local Differential Privacy setting, each user encodes and perturbs their own inputs before transmitting them to the untrusted server. This server can then compute statistical queries on the input data. We can give the formal definition of LDP proposed by [1] as follows.

Definition 13.1 *We say that an algoirhm A satisfies ϵ-Local Differential Privacy where $\epsilon > 0$ if and only if for any input v and v'*

$$\forall y \in Output(A) : \frac{Pr[A(v) = y]}{Pr[A(v') = y]} \leq e^{\epsilon}$$

where $Output(A)$ denotes every possible output of the algorithm A.

In order to show how to deal with LDP problem, an example was given.

Example 13.2 *Assume there is a survey released for an university to know whether the students are satisfied with their president or not. Respondents should answer "Yes" or "No" with their real names known by the surveyor. The problem is, how to design a mechanism to get meaningful statistics while preserving the privacy of every respondent ?*

The problem could be solved by the following mechanism.

Randomized Response Although Local Differential Privacy has only recently been gaining traction and popularity, the ideas behind it are much older. The key idea of Local Differential Privacy is *randomized response*, a surveying technique first introduced by Warner in [2]. To answer the question the respondent gives a randomized output, which is truthful for probability p and fake for probability $1 - p$. If p is close to $1 - p$, the output distribution of algorithm A for different inputs will be similar. This preserves respondents' privacy by giving them strong plausible deniability while allowing for computation of accurate population statistics.

Given that respondents comply with the protocol, respondents will answer the question truthfully 55% of the time. An unbiased estimate of the true number of "Yes" answer can therefore be computed by $10(X - 0.45)$ where X refers to the fraction of respondents who answered in the positive. The survey participants are given a privacy guarantee of $\epsilon = \ln(\frac{0.55}{0.45}) = \ln\frac{11}{9}$.

LDP has been seen in practical deployments by major technology organizations such as Apple [3], who use it to collect usage statistics and find commonly used emojis, new words that are not part of the dictionary yet and to improve user behaviour, Google [1], who use it in Chrome to collect commonly chosen homepages, settings, and other web browsing behaviour, and Microsoft [4], who use it for their collection of telemetry data.

13.2 Composition of Privacy Leakage

Previously, we have defined two kinds of differential privacy: ϵ-differential privacy (pure-dp) and (ϵ, δ)-differential privacy (approximate-dp). And we also discussed the composition of privacy leakage in $\epsilon - dp$ scenario.

Theorem 13.3 *For Laplace Mechanism preserving ϵ_0-differential privacy at each round, the total privacy leakage of k rounds is $k\epsilon_0$ (i.e. k rounds in total preserves $k\epsilon_0$-dp).*

Lemma 13.4 *(McDiarmid inequality)Let $X_1, ..., X_m \in \chi^m$ be a set of $m \geq 1$ independent random variables and assume that there exist c1,......,cm > 0 such that f: $\chi^m \rightarrow \mathbb{R}$ satisfies the following conditions:*

$$|f(x_1, ..., x_i, ..., x_m) - f(x_1, ..., x_i^{'}, ..., x_m)| \leq c_i$$

for all i ∈[m] and any points $x_1, ..., x_m, x_i^{'} \in \chi$. Let f(S) denote f($X_1, ..., X_m$), then, for all ϵ >0, the following inequalities hold:

$$Pr[f(S) - E[f(S)] \geq \epsilon] \leq exp(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2})$$

$$Pr[f(S) - E[f(S)] \leq -\epsilon] \leq exp(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2})$$

Theorem 13.5 *For all $\epsilon_0, \delta \geq 0$, k rounds of independent $Lap(\sigma)$, ϵ_0-dp mechanisms satisfies (ϵ, δ)-dp where:*

$$\epsilon = \sqrt{2kln(1/\delta)}\epsilon_0 + 2k\epsilon_0^2$$

Proof: From the Proposition 12.3 in last lecture, we need to proof the bound of f($a_1, ..., a_k$) $= \sum_{i=1}^k ln\frac{p_i(a_i)}{q_i(a_i)}$, where $\frac{p_i(a_i)}{q_i(a_i)}$ denotes $\frac{Pr(A_i(D)=a_i)}{Pr(A_i(D^{'})=a_i)}$. Notice that it is a problem about the sum of k i.i.d. variables, we can estimate its expectation at first and then use concentration inequality to proof its bound.

Because of Lap(σ), we have $-\epsilon_0 \leq ln\frac{p_i(a_i)}{q_i(a_i)} \leq \epsilon_0$. Let g(a) denote $ln\frac{p(a)}{q(a)}$, observe $1 = \int p(a)da = \int q(a)da = \int p(a)e^{-g(a)}da$. So we have:

$$E_{a_i \ p_i}[ln\frac{p_i(a_i)}{q_i(a_i)}] = \int p(a)g(a)da \tag{13.1}$$

$$= 1 - 1 + \int p(a)g(a)da \tag{13.2}$$

$$= \int p(a)[e^{-g(a)} - 1 + g(a)]da \tag{13.3}$$

Let h(x) denote $e^{-x}-1+x$ where x $\in [-\epsilon_0, \epsilon_0]$. Using derivation we have $h(x) \leq max\{h(-\epsilon_0), h(\epsilon_0)\}$. With the same way, we can get $max\{h(-\epsilon_0), h(\epsilon_0)\} \leq 2\epsilon_0^2$. So, $E_{a_i \ p_i}[ln\frac{p_i(a_i)}{q_i(a_i)}] \leq 2\epsilon_0^2$.

For $a_1, ...a_k$ are i.i.d. and $|f(a_1, ...a_i, ..., a_k) - f(a_1, ...a_i', ..., a_k)| = |ln\frac{p_i(a_i)}{q_i(a_i)} - ln\frac{p_i(a_i')}{q_i(a_i')}| \leq 2\epsilon_0$, with Lemma 4.4, we have:

$$Pr[f(a_1, ...a_i, ..., a_k) \geq \epsilon' + 2k\epsilon_0^2] \leq Pr[f(a_1, ...a_i, ..., a_k) - E[f(a_1, ...a_i, ..., a_k)] \geq \epsilon'] \leq exp(\frac{-\epsilon'^2}{2m\epsilon_0^2})$$

Let $\delta = exp(\frac{-\epsilon'^2}{2k\epsilon_0^2})$, we have:

$$\epsilon' = \sqrt{2kln(1/\delta)}\epsilon_0$$

Therefore:

$$Pr[f(a_1, ...a_i, ..., a_k) \geq \sqrt{2kln(1/\delta)}\epsilon_0 + 2k\epsilon_0^2] \leq \delta$$

■

13.3 Multiplicative Weight Updating Mechanism

Accuracy and privacy in the interactive setting. Formally, an *interactive mechanism* $M(x)$ is a stateful randomized algorithm which holds a histogram $x \in R^N$. It receives successive counting queries $f_1, f_2, \cdots \in \mathcal{F}$ one by one, and in each round t, on query f_t , it outputs a (randomized) answer a_t (a function of the input histogram, the internal state, and the mechanism's coins). For privacy guarantees, we always assume that the queries are given to the mechanism in an adversarial and adaptive fashion by a randomized algorithm A called the *adversary*. For accuracy guarantees, while we usually consider adaptive adversarial, we will also consider non-adaptive adversarial queries chosen in advance–we still consider such a mechanism to be interactive, because it does not know in advance what these queries will be. The main query class we consider throughout this work is the class $\mathcal{F}$ of all counting queries, as well as sub-classes of it.

For the interactive setting, there is a privacy-preserving interactive mechanism which is called private multiplicative weight updating (PMW) mechanism.

```
Parameters: Data universe $\mathcal{X}$ with $|\mathcal{X}| = N$. $D$ is a subset of $\mathcal{X}$ with $|D| = n$. The number of queries $k$, $(\epsilon, \delta) - dp, (\alpha, \beta) - acc$. Put $\sigma = \frac{10 \cdot \log(1/\delta) \log^{1/4} N}{\sqrt{n}\epsilon}, \eta = \frac{\log^{1/4} N}{\sqrt{n}}, T = 4\sigma \cdot (\log k + \log 1/\beta)$.
Input: Database $D \in \mathcal{X}^n$ corresponding to a histogram $x \in \mathbb{R}^N$
Algorithm:
Initialize: Set $x_0 = (1/N, \cdots, 1/N), m = 0$.
For $t = 1, 2, \cdots, k$:
    1. Receive $f_t$
    2. $d_t \leftarrow < f_t, x > - < f_t, x_{t-1} >$
    3. $\hat{d}_t \leftarrow d_t + A_t, A_t \sim Lap(\sigma)$
    4. If $\hat{d}_t > T$ : For all $i$ $s.t.$ $f_t[i] = 1, x_t[i] \leftarrow x_{t-1}[i]e^{\eta}$; otherwise, $x_t[i] \leftarrow x_{t-1}[i]$.
       Else if $\hat{d}_t < -T$: For all $i$: $f_t[i] = 0$,$x_t[i] \leftarrow x_{t-1}[i]e^{\eta}$; otherwise, $x_t[i] \leftarrow x_{t-1}[i]$.
       Else $x_t \leftarrow x_{t-1}$.
       Normalize, $x_t[i] \leftarrow \frac{x_t[i]}{\sum_{i \in \mathcal{X}} x_t[i]}$.Update $m \leftarrow m + 1$.
    5. If $m > n\sqrt{\log N}$: abort and output 'failure'.
       Else if $|\hat{d}_t| > T$, output the noisy answer $< f_t, x > +A_t$. Else, output $< f_t, x_{t-1} >$.
```

In the t-th round, after the t-th query f_t has been specified, we compute a noisy answer bat by adding (properly scaled) Laplace noise to $f_t(x)$—the "true" answer on the real database. We then compare this noisy answer with the answer given by the previous round's database $f_t(x_{t-1})$. If the answers are "close", then this is a "lazy" round, and we simply output $f_t(x_{t-1})$ and set $x_t \leftarrow x_{t-1}$. If the answers are "far", then this is an "update" round and we need to update or "improve" x_t using a multiplicative weights re-weighting. The intuition is that the re-weighting brings x_t "closer" to an accurate answer on f_t. In a nutshell, this is all the algorithm does. The only additional step required is bounding the number of "update" rounds: if the total number of update rounds grows to be larger than (roughly) n, then the mechanism fails and terminates. This will be a low probability event.

We use a_t to denote the true answer on the database on query t, and $\hat{a}_t$ denotes this same answer with noise added to it. We use d_t to denote the difference between the true answer a_t and the answer given by x_{t-1}, and $\hat{d}_t$ to denote the difference between the *noisy* answer and the answer given by x_{t-1}. The boolean variable w_t denotes whether the noisy difference was large or small. If $\hat{d}_t$ is smaller (in absolute value) than $\approx 1/\sqrt{n}$, then this round is lazy. If $\hat{d}_t$ is larger than threshold then this is an *update* round and we set $m \leftarrow m + 1$.

Definition 13.6 *We say that a mechanism M is $(\alpha, \beta, k)-$(adaptively) accurate for a database x, if when it is run for k rounds, for any (adaptively chosen) counting queries, with all but β probability over the mechanism's coins $\forall t \in [k], |a_t - f_t(x)| \leq \alpha$*

Theorem 13.7 *Let X be a data universe of size N. For any $k, \varepsilon, \delta, \beta > 0$, the Private Multiplicative Weights Mechanism above is an $(\epsilon, \delta)-$differentially private interactive mechanism. For any database of size n, the mechanism is $(\alpha, \beta, k)-$accurate for (adaptive) counting queries over X, where*

$$\alpha = O(\varepsilon^{-1} n^{-1/2} \cdot \log(1/\delta) \log^{1/4}(N) \cdot (\log k + \log(1/\beta)))$$

The running time in answering each query is $N \cdot poly(n) \cdot polylog(1/\beta, 1/\varepsilon, 1/\delta)$.

The proof of the theorem can be found in [6].

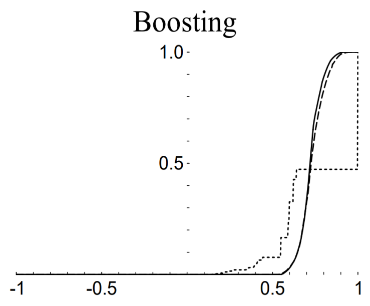
13.4 Generalization of Boosting

In this section, we will discuss the generalization of boosting algorithm based on the margin theory.

We have known that boosting works by voting over a set of base hypotheses, where the result hypothesis as a weighted combination of base hypotheses. As long as the base hypotheses are not so bad, the training error will exponentially decay to zero with respect to the boosting iterations. Considering the test/generalization error of boosting, it is surprising to find out that after the training error is reduced to zero, the test error continues to go down with additional training. This fact is counter-intuitive because most of the time, we assume that the 'over-fitting' on training data will result in bad performance of generalization.

To explain this fact, the main observation is that boosting continues to enlarge the *margin* even after the training error is reduced to zero. Consider a binary classification task with sample space $\mathcal{X} \times \{1, -1\}$, distribution $\mathcal{D}$ on $\mathcal{X} \times \{1, -1\}$, hypothesis space $\mathcal{H}$, the voted classifier given by boosting $f(x) = \Sigma_{i=1}^{N} \alpha_i h_i(x)$, where the weights αs are normalized to have sum 1. We define the *margin* of a sample $(x, y) \in \mathcal{X} \times \{1, -1\}$ as $yf(x)$. It is easy to see that *margin* is a number in $[-1, 1]$, and that a sample is classified correctly if and only if it has positive *margin*. Moreover, a large *margin* can be interpreted a "confident" classification.

To visualize the observation, we plot the fraction of examples whose margin is at most θ as a function of $\theta \in [-1, 1]$, we refer to these graph as *margin distribution graph.* In experiments, we observe that as the training progresses, the boosting algorithm tends to increase the margins associated with examples and converge to a margin distribution in which most examples has large margins. The graph below is a demo of *margin distribution graph* of a binary classification, where the short-dashed, long-dashed and solid curves correspond to the margin distribution after 5, 100, 1000 iterations.



Now, we formalize the bound of generalization error of boosting by giving the following theorem given by Robert E. Schapire and Yoav Freund in [7].

Theorem 13.8 *Let S be a sample of n examples chosen independently at random according to $\mathcal{D}$. Assume that the base hypothesis space $\mathcal{H}$ is finite, and let $\delta > 0$. Then with probability at least $1 - \delta$ over random choice of the training set S, every voted classifier f satisfies the following bound for all $\theta > 0$:*

$$\boldsymbol{P}_{\mathcal{D}}[yf(x) \leq 0] \leq \boldsymbol{P}_S[yf(x) \leq \theta] + O(\frac{1}{\sqrt{n}}(\frac{log\ n\ log\ |\mathcal{H}|}{\theta^2} + log(1/\delta))^{1/2})$$

This theorem states that if the voting classifier generates a good margin distribution, that is, most training examples have large marins so that $P_S(yf(x) \leq \theta)$ is small for not too small θ, then the generalization error is also small. In [7], it has also been shown that for the AdaBoost algorithm $P_S(yf(x) \leq \theta)$ decreases to zero exponentially fast with respect to the number of boosting iterations if θ is not too large. These results imply that the excellent performance of AdaBoost is due to its good margin distribution.

Breiman[8] also proved an upper bound for the generalization error of voting classifiers. This bound depends only on the minimum margin, not on the entire margin distribution.

Theorem 13.9 *Let θ_0 be the minimum margin defined as*

$$\theta_0 = min\{yf(x) : (x, y) \in S\}$$

, where S is the training set. If

$$\mathcal{H} < \infty,\ \theta_0 > 4\sqrt{\frac{2}{|\mathcal{H}|}},\ R = \frac{32log(2|\mathcal{H}|)}{n\theta_0^2} \leq 2n$$

then for any $\delta > 0$, with probability at least $1 - \delta$ over the random choice of the training set S of n examples, every voting classifier f satisfies the following bounds:

$$\boldsymbol{P}_D(yf(x) \leq 0) \leq R(log(2n) + log\frac{1}{R} + 1) + \frac{1}{n}log(\frac{|\mathcal{H}|}{\delta})$$

Breiman pointed out that this bound is sharper that that of Schapire's, as the second term of this bound is $O(\frac{log\ n}{n})$ while that in Schapire's is $O(\sqrt{\frac{log\ n}{n}})$.

References

[1] Erlingsson U, Pihur V, Korolova A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. pp. 1054–1067. ACM (2014).

[2] Warner S.L. Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association 60(309), 63–69 (1965).

[3] Apple Inc. Differential Privacy Team: Learning with privacy at scale (2017).

[4] Ding B, Kulkarni J, Yekhanin S. Collecting telemetry data privately. In: Advances in Neural Information Processing Systems. pp. 3571–3580 (2017).

[5] Bebensee B. Local differential privacy: a tutorial[J]. arXiv preprint arXiv:1907.11908, 2019.

[6] Hardt M, Rothblum G.N. A multiplicative weights mechanism for privacy-preserving data analysis. in Proc. Ann. IEEE Symp. Found. Comput. Sci. (FOCS), Las Vegas, NV, 2010, pp. 61-70.

[7] Bartlett P, Freund Y, Lee W S, et al. Boosting the margin: A new explanation for the effectiveness of voting methods[J]. The annals of statistics, 1998, 26(5): 1651-1686.

[8] Breiman L. Prediction games and arcing algorithms[J]. Neural computation, 1999, 11(7): 1493-1517.