Privacy for attribution algorithms

Sushant Agarwal*† Yukti Makhija*◦ Rishi Saket*◦
◦ Google DeepMind † Northeaster University

[1]

1 The Exponential Mechanism

The exponential mechanism is a natural way for designing differentially private algorithms which have a "selection" flavor, when when adding noise to the output doesn't make much sense. Consider problems with the following structure

- A dataset $X \in \mathcal{X}^n$
- A set of possible outputs $\mathcal{Y}$ (assume finite for now, but not necessary)
- A score function $s : \mathcal{X}^n \to \mathcal{Y}$, where $s(y, X)$ measures how 'good' the output y is for dataset X.

Given the above, what we want from a selection algorithm is to return an output $y \in \mathcal{Y}$ that maximizes (or approximately maximizes) the score function s on dataset X, that is, it returns

$$\operatorname*{argmax}_{y \in \mathcal{Y}} s(y, X).$$

We define the sensitivity of the score function as

$$\Delta = \max_{y \in \mathcal{Y}} \max_{X, X'} |s(y, X) - s(y, X')|,$$

where $X, X' \in \mathcal{X}^n$ are neighbouring (i.e., 1 row in the dataset is allowed to modify its value). Specifying the above elements is enough to describe the exponential mechanism.

Algorithm 1 Exponential Mechanism

Input: $X, \mathcal{Y}, s, \Delta$ **Output:** $y \in \mathcal{Y}$

1: Select Y from the distribution where $\mathbb{P}[Y = y] \propto \exp(\frac{\epsilon}{2\Delta} s(y, X))$
2: **return** Y

In particular, when $\mathcal{Y}$ is finite,

$$\mathbb{P}[Y = y] = \frac{\exp(\frac{\epsilon}{2\Delta} s(y, X))}{\sum_{y \in \mathcal{Y}} \exp(\frac{\epsilon}{2\Delta} s(y, X))}$$

We state the following privacy and utility guarantees for the exponential mechanism.

Theorem 1. *The exponential mechanism satisfies ϵ-differential privacy*

[1] *Alphabetical Order

Proof. Assume $\mathcal{Y}$ is finite. Then,

$$\mathbb{P}[y|X] = \frac{\exp(\frac{\epsilon}{2\Delta}q(y,X))}{\sum_y \exp(\frac{\epsilon}{2\Delta}q(y,X))}$$

Consider $X \sim X'$. Then

$$\frac{\exp(\frac{\epsilon}{2\Delta}q(y,X))}{\exp(\frac{\epsilon}{2\Delta}q(y,X'))} = \exp\left(\frac{\epsilon}{2\Delta}(q(y,X)) - q(y,X'))\right) \leqslant \exp\left(\frac{\epsilon}{2\Delta}\cdot\Delta\right) = e^{\frac{\epsilon}{2}}$$

Similarly, for the normalising constants

$$\frac{\sum\exp(\frac{\epsilon}{2\Delta}q(y,X))}{\sum\exp(\frac{\epsilon}{2\Delta}q(y,X'))} \leqslant \sup_y \exp\left(\frac{\epsilon}{2\Delta}(q(y,X)) - q(y,X'))\right) \leqslant e^{\frac{\epsilon}{2}}$$

Hence,

$$\frac{\mathbb{P}[y|X]}{\mathbb{P}[y|X']} \leqslant e^{\frac{\epsilon}{2}}\cdot e^{\frac{\epsilon}{2}} = e^{\epsilon}$$

□

The best we can hope for from a selection algorithm is that, on input a data set X, it outputs an element $y \in \mathcal{Y}$ with the maximum possible score, denoted by

$$s_{\max}(X) :== \max_{y\in\mathcal{Y}} s(y,X)$$

It can be shown that the exponential mechanism returns an element with near-maximum score, with high probability, as described in the statement below.

Theorem 2. *Suppose $|\mathcal{Y}| = d$. Given $X, \mathcal{Y}, s, \Delta$, if the output of the exponential mechanism is y, then for any $k > 0$, we have*

$$\mathbb{P}\left(s(y,X) < s_{\max}(X) - \frac{2\Delta(\ln d + k)}{\epsilon}\right) \leqslant e^{-k}$$

In addition, we have that

$$\mathbb{E}(s(y,X) \geqslant s_{\max}(X) - \frac{2\Delta(\ln d + 1)}{\epsilon}$$

Proof. Let us define

$$B_t = \{y \in \mathcal{Y} : q(y) \leqslant q_{\max} - \frac{2\Delta}{\epsilon}(\ln d + t)\}$$

Let $y_{\max}$ be an element corresponding to $q_{\max}$

$$\mathbb{P}(B_t) < \frac{\mathbb{P}(B_t)}{\mathbb{P}(y_{\max})} = \frac{\sum_{y\in B_t}\exp(\frac{\epsilon}{2\Delta}q(y))}{\exp(\frac{\epsilon}{2\Delta}q_{\max})}$$

$$y \in B_t \implies q(y) \leqslant q_{\max} - \frac{2\Delta}{\epsilon}(\ln d + t)$$

$$\implies \mathbb{P}(B_t) \leqslant \frac{|B_t|\exp(\frac{\epsilon}{2\Delta}q_{\max} - (\ln d + t))}{\exp(\frac{\epsilon}{2\Delta}q_{\max})}$$

$$= |B_t|e^{-\ln d - t} \leqslant |B_t|\frac{1}{d}e^{-t} < e^{-t}$$

□

2 Attribution algorithms

We are given a sequence of click times $t_1 < t_2 < \ldots < t_n$, and a conversion time $t_C > t_n$. An attribution algorithm aims to output element $i \in [n]$, such that it is responsible for the conversion at t_C. Due to Apple's new Link Decoration Deprecation policy, we do not know which $i \in [n]$ is responsible for the conversion. Regardless, we want some way to predict which click was responsible for the conversion. The Supernova attribution algorithm is a heuristic that outputs element n (closest to the conversion time) as the element responsible for the conversion, with the intuition that the closest click to the conversion time should most likely be responsible for the conversion.

2.1 Privacy

To maintain privacy, each user $i \in [n]$ has the capability to change their reported click time t_i by an amount τ. To maintain good utility, we would like to discourage them from doing so. In return, we promise to modify the attribution algorithm such that it satisfies differential privacy. In particular, we guarantee that if a single user modifies their click time from t_i to $t'_i \in [t_i - \tau, t_i + \tau])$, the output of the privatised attribution algorithm will only change slightly.

We now formalise this guarantee, for which we first define the notion of neighbouring datasets. For our setting, a dataset T denotes the set of click times, i.e., $T = \{t_1, t_2, \ldots, t_i, \ldots, t_n\}$. We denote the output of the attribution algorithm A on T by $A(T)$.

Definition 1 (Neighbouring datasets)**.** *Two datasets T, T' are neighbouring if they differ in the click value of only a single element, say element i, and the difference in click value is bounded by τ. More formally, $T = \{t_1, t_2, \ldots, t_i, \ldots, t_n\}$, $T' = \{t_1, t_2, \ldots, t'_i, \ldots, t_n\}$, and $|t_i - t'_i| \leqslant \tau$.*

Definition 2 (ϵ-Differential Privacy for an attribution algorithm)**.** *An attribution algorithm A satisfies ϵ-Differential Privacy if for every pair of neighbouring datasets T, T' and every $B \subseteq \mathcal{Y}$,*

$$\frac{\mathbb{P}[A(T) \in B]}{\mathbb{P}[A(T') \in B]} \leqslant e^{\epsilon}.$$

2.2 Exponential mechanism

Recall that to define the exponential mechanism, we need to specify the dataset X, the set of outputs $\mathcal{Y}$, and a score function s. The dataset here is given by T, the set of click times, i.e., $T = \{t_1, t_2, \ldots, t_i, \ldots, t_n\}$. An attribution algorithm outputs an element $i \in [n]$, hence $\mathcal{Y} = [n]$. One can choose a score function that best suits the problem.

2.2.1 Score function

The score function $s(y, X)$ for the exponential mechanism intuitively measures how 'good' the output y is for dataset X. In our setting, given dataset T, we want to assign a score to each click value t_i. We follow the intuition of the Supernova attribution algorithm, that the closest click to the conversion time should most likely be responsible for the conversion. Hence, we choose a monotonically increasing score function that assigns a larger value to larger time values. What would be the best score function for our situation? That depends on how we model the situation, whether we assume that the clicks follow a particular distribution (for eg., Poisson), etc. Regardless, below are some choices of score functions:

1. $s(i, T) = t_i$
2. $s(i, T) = e^{t_i}$

3. $s(i,T) = -\frac{1}{t_i}$ (if we assume $t_C = 0$)

We can scale the above functions such that the scores sum up to 1, but it doesn't make a difference for the exponential mechanism. The corresponding sensitivities are

1. $\Delta = \tau$
2. We will need to assume an upper bound on t_i to bound Δ. t_C is one bound we can use, leading to $\Delta \leqslant e^{t_C}(e^{\tau} - 1)$
3. We will need to assume a separation between t_i and t_C, say α. Then $\Delta = \left(\frac{1}{\alpha - \tau} - \frac{1}{\alpha}\right)$

2.2.2 Performance guarantees

Different score functions would give different performance guarantees. To illustrate, let us choose the first score function for now. We get the following utility guarantee on applying the exponential mechanism.

Theorem 3. *Given T, s, if the output of the exponential mechanism is t_i, then for any $k > 0$, we have*

$$\mathbb{P}\left(s(t_i,T) < s_{\max}(T) - \frac{2\tau(\ln n + k)}{\epsilon}\right) \leqslant e^{-k}$$

In addition, we have that

$$\mathbb{E}(s(t_i,T) \geqslant s_{\max}(T) - \frac{2\tau(\ln n + 1)}{\epsilon}$$

Let us plug in some values to see what this guarantee means in practice. Consider $T = \{t_1, t_2, \ldots, t_i, \ldots, t_n\}$, where $n = 32, t_n = 100, \tau = 2.5, \epsilon = 5$. Setting $k = 2$, what this guarantee tells us is that with probability > 0.86, the exponential mechanism will return an element within $7s$ of the optimal. I have to confirm, but there seems to be an new accuracy analysis that improves the one here by a factor of 4, hence saying that we will be within $1.75s$ of the optimal.

3 Privacy in Matchings

We consider a more general attribution problem where there are multiple, possibly overlapping bags. Let us say there are m bags, and n clicks, where $n > m$. There are weighted edges from bags to clicks, and finding the optimal attribution can be thought of as finding the maximum weight matching on a bipartite graph, which in turn is the problem we study here.

There are known algorithms for finding the max-weight matching on a bipartite graph, but we want to do this task privately. A dataset here is a bipartite graph, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We restrict our attention to the case where there are m vertices on the left side denoting bags, and n vertices on the right side denoting clicks. There is a weight function $\mathcal{W} : \mathcal{E} \to \mathbb{R}^+$. We want to design a private matching algorithm, which we denote by A. A takes in a graph and outputs a matching M. We assume there is always a click matched to every bag, and restrict our output set $\mathcal{Y}$ to the set of these matchings.

To define privacy in this setting, we first need to define the notion of neighboring datasets.

Definition 3 (Neighbouring datasets)**.** *Consider 2 graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ such that $\mathcal{V} = \mathcal{V}'$, $\mathcal{E} = \mathcal{E}'$. They are said to be neighbouring if there exists exactly one click v such that edges incident to v can have different weights, and the l_1 distance between the weight functions $\mathcal{W}, \mathcal{W}'$ is bounded by 1.*

Definition 4 (ϵ-Differential Privacy for a matching algorithm)**.** *A matching algorithm A satisfies ϵ-Differential Privacy if for every pair of neighbouring graphs $\mathcal{G}, \mathcal{G}'$ and every $B \subseteq \mathcal{Y}$,*

$$\frac{\mathbb{P}[A(\mathcal{G}) \in B]}{\mathbb{P}[A(\mathcal{G}') \in B]} \leqslant e^{\epsilon}.$$

3.1 Lower Bound

We show the following lower bounds for A.

Theorem 4. *Let there be n clicks, and m bags. Suppose that mechanism A is (ϵ, δ)-differentially private, and computes a matching with weight at least $OPT - \alpha m$ with probability $1 - \gamma$. Then, $\alpha \geqslant \theta(\epsilon, \delta, \gamma)$.*

Proof. The proof relies on the following lemma

Lemma 5. *Let mechanism $A : \{0,1\}^n \to \{0,1\}^n$ be (ϵ, δ)-differentially private, and suppose that for all input databases D, with probability at least $1 - \gamma$, $||A(D) - D||_1 \leqslant \alpha n$. Then,*

$$\alpha \geqslant 1 - \frac{e^\epsilon + \delta}{(1 + e^\epsilon)(1 - \gamma)} := \theta(\epsilon, \delta, \gamma).$$

This lemma basically states that no (ϵ, δ)-differentially private mechanism can reconstruct more than a fixed constant fraction of its input database. The lower bound proof uses the above lemma as follows

1. First, we describe how to convert a database $D \in \{0,1\}^m$ to a matching problem in our setting, by specifying the bags, the clicks, and edge weights.
2. Next, we analyze how the graph changes when a single bit in D is changed. This will control how private the matching algorithm is with respect to the original database.
3. Finally, we show how to output a database guess $\hat{D}$ from the matching produced by the private matching algorithm.

This composition of three steps will be a private function from $\{0,1\}^m \to \{0,1\}^m$, so we can apply the lemma to lower bound the error, implying a lower bound on the error of the matching algorithm.

Step 1: Let $D \in \{0,1\}^m$. Consider a graph with m non-overlapping bags, and each bag is connected to $k \geqslant 2$ clicks, so $n = mk$. Denote the bags by $u_1, \ldots, u_m$, and the corresponding clicks to bag u_i by $v_1^i, \ldots, v_k^i$. The edge weights between u_i and v_1^i is given by $1 + d_i$, between u_i and v_2^i is given by $2 - d_i$, and between u_i and all other clicks is given by 1.

Step 2: Changing a bit d_i will change the valuation of the 2 clicks v_1^i and v_2^i. Hence if $D \sim D'$ are 1 perturbation apart (they differ in 1 bit, hence neighbouring), then G and G' are 2 perturbations apart. Hence, by group privacy, applying an (ϵ, δ)-differentially private matching algorithm on G will be $(2\epsilon, 2\delta)$-differentially private on D.

Step 3: To guess $\hat{D}$ from the output of the matching algorithm, we let $\hat{d}_i = 1$ if u_i is matched to v_1^i, $\hat{d}_i = 0$ if u_i is matched to v_2^i, and arbitrary otherwise

Note that the max weight matching assigns u_i its preferred good determined by d_i, and OPT$= 2m$. If A computes an matching with welfare OPT $-\alpha m$, it must give all but an α fraction of bags the click corresponding to d_i. So, the reconstructed database will miss at most αm bits with probability $1 - \gamma$, and by the previous lemma,

$$\alpha \geqslant \theta(2\epsilon, 2\delta, \gamma).$$

Note that the lower bound construction here assumes non-overlapping bags, but that is not essential. In the above construction, we can add additional edges of weight 1 between any bag and click, and the proof would go through the same way. □

3.2 Upper Bound

We show the following upper bounds for A.

3.2.1 Laplace Mechanism

Theorem 6. *For any $\epsilon, \gamma > 0$ and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with weight function $\mathcal{W}$, there is an algorithm A that is ϵ-differentially private and releases with probability $1-\gamma$ a perfect matching of weight at most $O\left(\frac{m}{\epsilon}\log(n/\gamma)\right)$ smaller than optimal.*

Proof. We just apply the Laplace mechanism here to release a noisy version of the edge weights incident to each click, then run a deterministic max weight matching algorithm on the resultant graph. To describe it, we first recall the notion of neighbouring graphs. 2 graphs are said to be neighbouring if there exists exactly one click v such that edges incident to v can have different weights, and the l_1 distance between the weight functions $\mathcal{W}, \mathcal{W}'$ is bounded by 1. Hence, the l_1 sensitivity Δ is 1.

Consequently, the Laplace mechanism here just adds noise X_e distributed according to $\text{Lap}\left(\frac{\Delta}{\epsilon}\right)$ for each edge $e \in \mathcal{E}$. After this, we release the maximum-weight matching on the resulting graph $(\mathcal{G}, \mathcal{W}')$. This is ϵ-differentially private, since it is post-processing of the Laplace mechanism. We now show that the resulting error is small. With probability $1-\gamma$, we have that

$$|X_e| \leqslant \frac{1}{\epsilon}\log(|\mathcal{E}|/\gamma)$$

for every $e \in \mathcal{E}$. Consequently, conditioning on this event, if M is the matching released by the algorithm and M^* is the maximum-weight matching, then we have that

$$\begin{aligned}
\mathcal{W}(M) &\geqslant \mathcal{W}'(M) - \frac{m}{\epsilon}\log(|\mathcal{E}|/\gamma) && (|M| \leqslant m)\\
&\geqslant \mathcal{W}'(M^*) - \frac{m}{\epsilon}\log(|\mathcal{E}|/\gamma) && (M \text{ is the max-weight matching for } (\mathcal{G}, \mathcal{W}'))\\
&\geqslant \mathcal{W}(M^*) - \frac{2m}{\epsilon}\log(|\mathcal{E}|/\gamma)\\
&\geqslant \mathcal{W}(M^*) - \frac{2m}{\epsilon}\log(mn/\gamma) && (|\mathcal{E}| \leqslant mn)\\
&= \mathcal{W}(M^*) - O\left(\frac{m}{\epsilon}\log(n/\gamma)\right) && (n > m)
\end{aligned}$$

□

3.2.2 Exponential mechanism

Recall that to define the exponential mechanism, we need to specify the dataset X, the set of outputs $\mathcal{Y}$, and a score function s. The dataset X here is the graph $\mathcal{G}$. Our output set $\mathcal{Y}$ is the set of matchings that always has a click matched to every bag. For now, choose our score function of a matching to be the sum of weights in the matching, i.e., $s(M) = \mathcal{W}(M)$. Recall that 2 graphs are said to be neighbouring if there exists exactly one click v such that edges incident to v can have different weights, and the l_1 distance between the weight functions $\mathcal{W}, \mathcal{W}'$ is bounded by 1. Hence, the sensitivity Δ of the score function is 1. Note that $|\mathcal{Y}| \leqslant n^m$. We get the following guarantee for the matching if we apply the exponential mechanism

Theorem 7. *Given $G, \mathcal{Y}, s$, if the output of the exponential mechanism is M, then for any $k > 0$, we have*

$$\mathbb{P}\left(s(M, G) < OPT - \frac{2(m\ln n + k)}{\epsilon}\right) \leqslant e^{-k}$$

In addition, we have that

$$\mathbb{E}(s(M, X) \geqslant OPT - \frac{2(m \ln n + 1)}{\epsilon}$$

Greedy Exponential The above algorithm is not efficient, and runs in exponential time. A faster algorithm would be to run the exponential algorithm for each bag to choose the click it is matched to. More precisely, we iterate over the bags $u_1, \ldots, u_m$, and assign a click to each bag. Then we remove that click and move on to the next bag until we terminate.

Recall that if we apply the exponential mechanism to 1 bag and n clicks, we get an additive error of $O(\log n)$ (and this is the best known analysis of the exponential mechanism). Assuming m non-overlapping bags and n clicks, we get an additive error of $O(m \log n)$, so in general we do not get better error with this algorithm, but it is more efficient.

3.2.3 Gaussian mechanism

If we relax our privacy guarantee to (ϵ, δ)-differential privacy, I think we get a better upper bound of $O(m\sqrt{\log n})$, but I need to double check

3.3 Further directions

3.3.1 Releasing partial information

Instead of outputting the whole matching along with the edges, can we output lesser information and see if we get better error? We look at 2 cases below.

Value: What happens when we just reveal the weight of the output matching, instead of all the edges? Do we get better error? For our notion of neighbours, it is easy to show that neighbouring graphs will have value of the max-weight matching differing by at most 1. Hence, the sensitivity Δ of the output weight is also bounded by 1. Consequently, the Laplace mechanism here just adds noise distributed according to $\text{Lap}\left(\frac{\Delta}{\epsilon}\right) = \text{Lap}\left(\frac{1}{\epsilon}\right)$ to the value of the max-weight matching of the input to ensure ϵ-DP. This mechanism gives $O(1)$ error.

Can we show the above error is optimum? Apart from the output matching value, is there any other summary we could consider?

Joint-DP: Similar to Hsu et al. [1], we can look at settings where each agent (click) only finds out which item (bag) it is matched with (if at all), and none of the other edges of the matching. Each agent then wants to protect privacy against each other agent (and other agents are allowed to collude). This is basically the same as Hsu et al. [1]. To differentiate from them, we can consider some variations along the following dimensions

1. Edges vs. agents? Do we want to protect the privacy at the level of each agent as above? Or at the level of each edge?

2. Collude vs. no collude? Do we allow for colluding between agents, or no?

3.3.2 Stochastic setting

What happens when edge weights come from a distribution (eg., Gaussian) instead of being completely controlled by the user (here a user is either an edge or a node (click))? Let's say each user can only change the mean of their Gaussian distribution by some amount, and then the actual weights are drawn from this distribution.

Let us consider the scenario where we want to protect the privacy of the parameters of each users distribution (eg., mean of the Gaussian), and not the actual values. Then running a normal max-weight matching algorithm on the values is DP by itself, because we are only drawing values from the Gaussian distribution around the mean, providing a local DP guarantee. We can add some additional noise on top to reach the desired ϵ, δ. Assuming a bound on how much a user can change their mean, we get similar lower and upper bound values as in the non-stochastic setting, using similar analysis.

One can also consider the case where we want to protect the privacy of the actual values. Let us first consider the case where a user is an edge, and each edge weight is drawn from the same Gaussian distribution $\mathcal{N}(\mu, 1)$ where μ could be unknown. Here the edge weights will be unbounded, but we can use prior knowledge about the distribution to clip the weights and bound the sensitivity. For example, if we know that that $|\mu| \leqslant B$, and if there are k edges, we can clip the edge values to be between $(-B - O\sqrt{\log k}, B + O\sqrt{\log k})$, and with high probability all the weights will lie in between this interval. Now, $\Delta = B + O\sqrt{\log k}$, and we can then add noise distributed according to Lap $\left(\frac{\Delta}{\epsilon}\right)$ to the value of each edge to ensure ϵ-DP. This would give us an extra $O(B + O\sqrt{\log k}) \leqslant O(B + O\sqrt{\log mn}) = O(B + O\sqrt{\log k})$ factor in the error as compared to the non-stochastic setting. Can we get a better upper bound? What lower bound can we show?

We can also consider variants of this setup along the following dimensions.

1. Is a user an edge or a click? I.e., do we want to preserve the privacy of each edge or each click?
2. Same vs. different distributions for each user?

Consider point 1. If we want to preserve the privacy of each click instead, the l_1 sensitivity across neighbouring datasets is now $k(B + O\sqrt{\log n})$, where k is the number of bags a click can be assigned to. We know that $k \leqslant m$, hence $\Delta = m(B + O\sqrt{\log n})$, and we an extra m factor in the upper bound as compared to the case where a user is an edge.

Point 2 should not matter as long as $\mu \leqslant B$.

References

[1] Justin Hsu, Zhiyi Huang, Aaron Roth, Tim Roughgarden, and Zhiwei Steven Wu. Private matchings and allocations. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 21–30, New York, NY, USA, 2014. Association for Computing Machinery.