Privacy III: Inferential Privacy and Pufferfish

Lecturer: Kamalika Chaudhuri

May 8, 2020

So far we have talked about differential privacy, which offers an excellent privacy solution and applies to a multitude of privacy problems. But not all privacy problems can be tackled with differential privacy. This lecture we will present a few examples of such problems, and talk about how an inferential privacy framework can be used to address them. We will then go ahead and introduce Pufferfish privacy – a formal framework for inferential privacy – and look at a couple of example privacy mechanisms.

1 Motivation

Let us begin with two example privacy problems that are not quite satisfactorily addressed by the standard differential privacy framework.

Example 1: Fine-Grained Location Privacy. Suppose a user has a phone app which provides location-based restaurant recommendations. While the users may be okay if the app knows or records their location to a coarse level – say, within a few blocks – many may be uncomfortable if their location is known or recorded with too much precision – eg, at the building level or even room level.

How can we report location at a coarse level of granularity while still preserving fine-grained privacy? A direct application of differential privacy provides privacy at an individual level, and does not cover this situation. What we need here is a solution that can offer privacy up to a certain distance scale – the length of a block for example.

Example 2: Privacy of Time-Series Data. Suppose we have time-series data on physical activity from a single user; our goal is to publish aggregates over a certain amount of time while hiding what the person is doing precisely at certain moments. For example, we would like to publish that the user walks 5 miles a week, while hiding what they were doing at 10:30am on Jan 4.

How can we do this? Since this is a single person's data, differential privacy will be ineffective – it will add enough noise to hide the effect of participation of a single person, and the result will be noisy enough to be useless. An alternative solution is what has been called *entry* or *element-wise* differential privacy – add enough noise to hide activity at a single time point. Unfortunately this does not offer sufficient privacy – as activities at close-by time points are highly correlated. A third plausible solution is group differential privacy on the entries – where the idea is to ensure that the participation of a correlated group of entries does not make a differenceto the outcome. While this solution is good in situations where there are small highly correlated groups that are uncorrelated

with the rest of the data, it is not as good in this situation. If we consider the worst case, then activity at all time points are correlated!

What kind of solution can we expect here? Intuitively, while activities at all time points are correlated in the worst case, what can help us provide a solution is that the correlation decays with time. For example, there is a high correlation between my activity now and 5 mins later – while 5 hours later is a completely different story. This is the kind of structure that we hope to exploit to get both privacy and utility.

2 Pufferfish: A Formal Framework for Inferential Privacy

We next look at Pufferfish – an elegant formal framework for inferential privacy that can address some of these problems. Unlike differential privacy, Pufferfish is not a single privacy definition – but a very general framework that can be instantiated to produce privacy definitions for specific problems.

Pufferfish privacy has three components, described below.

- **Secret Set S.** The first is a set S of secrets, which is a set of facts which may need to be hidden. Examples of secrets are *Alice's age is 40* or *Bob participated in the data set* or *Carol went to a coffeeshop at 10am.* Observe that the secrets can be considerably more fine-grained than individual level facts that are hidden by differential privacy.

- **Secret Pairs Set Q.** The set of secret pairs Q is a subset of $S \times S$ and represents theset of pairs of secrets that the adversary should not be able to distinguish between. Examples of such secret pairs are (*Alice's age is 25,* Alice's age is 30), (*Bob is in the dataset,* Bob is not in the dataset) and so on. Again, these can be much more fine-grained than individual level.

- **Distribution Class Θ.** Finally, Θ is a class of distributions that could have plausibly generated the data, and represents prior belief of the adversary. For example, for the activity problem, we can imagine that the data is generated by a Markov Model, and $\Theta = \{\text{All Markov Chains with transition matrix in set } P\}$.

With these three components in place, we can now formally define Pufferfish privacy.

Definition 1 (Pufferfish Privacy)**.** *A mechanism A is said to satisfy ϵ-Pufferfish privacy with respect to a framework (S, Q, Θ) if for all secret pairs $(s_i, s_j) \in Q$, for all t in the range of A, and for all $\theta \in \Theta$, we have:*

$$p_{A,\theta}(A(X) = t | s_i, X \sim \theta) \leq e^{\epsilon} p_{A,\theta}(A(X) = t | s_j, X \sim \theta)$$

whenever $P(s_i|\theta) > 0$ and $P(s_j|\theta) > 0$.

Observe that while the definition appears mathematically similar to differential privacy, there are a few differences. The first and foremost is that the definition follows quite a different philosophy – instead of reasoning about participation, it reasons about what can be learnt or inferred by an adversary with prior knowledge θ. A second difference is that the randomness in the likelihood $p_{A,\theta}(\cdot)$ depends on both A and θ; thus, unlike differential privacy, a privacy mechanism does not *need to* inject external randomness to be Pufferfish private. In fact, there are some Pufferfish mechanisms which do not use external noise.

2.1 Interpretation

What does this definition mean concretely? If an algorithm A is ϵ-Pufferfish private, then we can show that for all t, all $\theta \in \Theta$ and $(s_i, s_j) \in Q$ such that $\Pr(s_i|\theta) > 0$ and $\Pr(s_j|\theta) > 0$, we have:

$$e^{-\epsilon} \cdot \frac{\Pr(s_i|X \sim \theta)}{\Pr(s_j|X \sim \theta)} \leq \frac{\Pr(s_i|X \sim \theta, A(X) = t)}{\Pr(s_j|X \sim \theta, A(X) = t)} \leq e^{\epsilon} \cdot \frac{\Pr(s_i|X \sim \theta)}{\Pr(s_j|X \sim \theta)}$$

In other words, for any pair $(s_i, s_j) \in Q$, the posterior odds ratio $\Pr(s_i|X \sim \theta, A(X) = t)/\Pr(s_j|X \sim \theta, A(X) = t)$ of the adversary after he sees the output of the algorithm A is only at most a factor of e^{ϵ} away from his prior odds ratio $\Pr(s_i|X \sim \theta)/\Pr(s_j|X \sim \theta)$. Thus the output of the algorithm A does not help him distinguish with any higher confidence whether secret s_i or s_j is true – than he had before observing the output of A.

Even though the guarantees of differential privacy involve a different philosophy, one can interpret it in an inferential light. We give a somewhat loose description here, and a more formal statement is provided by [KM14].

Suppose $s_{i,a}$ means Person i's record has value a, and $s_{i,+}$ (respectively $s_{i,-}$) means person i participated (respectively did not participate) in the dataset. Consider a Pufferfish framework:

$$\begin{aligned} S &= \{s_{i,a}, s_{i,+}, s_{i,-}, \forall i, \forall a\} \\ Q &= \{(s_{i,a}, s_{i,b}), \forall i, \forall a \neq b\} \cup \{(s_{i,+}, s_{i,-}), \forall i\} \\ \Theta &= \{\theta | \text{Presence/absence/value of Person i independent of} \\ &\qquad \text{presence/absence/value of Person j, } \forall i \neq j\} \end{aligned}$$

Then, any mechanism that is ϵ-differentially private is also ϵ-Pufferfish private in the framework (S, Q, Θ) and vice versa.

Finally, just as there is a No Free Lunch Theorem for machine learning, there is also one for privacy. It (loosely) states that if Θ is the set of all distributions, then no utility is possible. This means that choosing Θ carefully while selecting a Pufferfish framework is very important; if we make Θ too narrow, there may not be enough privacy. If we make it too broad, then we will not be able to get enough utility.

2.2 Examples

Let us now go back to the examples that we looked at in Section 1 and instantiate them in Pufferfish.

Example 1: Fine-grained Location Privacy. Let $\mathcal{X}$ be a space of locations, and for $x \in \mathcal{X}$, $s_{x,t}$ denote the secret that the user is in location x at time t. Then, the fine grained location privacy problem we discussed earlier can be represented by the following Pufferfish framework:

$$\begin{aligned} S &= \{s_{x,t} : x \in \mathcal{X}, t = 1, \ldots, T\} \\ Q &= \{(s_{x,t}, s_{x',t}) : x, x' \in \mathcal{X}, \|x - x'\| \leq r, t = 1, \ldots, T\} \\ \Theta &= \{\text{Location at time } t \text{independent of location at time } t', t \neq t'\} \end{aligned}$$

This privacy framework is called geoindistinguishability, and is a special case of Blowfish privacy – which in turn is a special case of Pufferfish. Observe that the secret pairs set Q is much smaller than what differential privacy would dictate.

Example 2: Privacy of Time-Series Data. Let A be a set of activities, and let P be a set of $|A| \times |A|$ stochastic matrices. Let $s_{a,t}$ denote the secret that the user is engaging in activity a at time t. Then the activity recognition problem that we discussed earlier can be represented in the following Pufferfish framework:

$$\begin{aligned} S &= \{s_{a,t} : a \in A, t = 1, \ldots, T\} \\ Q &= \{(s_{a,t}, s_{b,t}) : a, b \in A, a \neq b, t = 1, \ldots, T\} \\ \Theta &= \{\text{All Markov Chains with state space } A\text{and transition matrix in } P\} \end{aligned}$$

3 Mechanisms

Having looked at Pufferfish frameworks, let us now take a look at some Pufferfish mechanisms.

3.1 The Wasserstein Mechanism: A Generic Pufferfish Mechanism

A natural question is whether there is a generic Pufferfish mechanism that can be tailored to any Pufferfish framework. Fortunately, there is such a thing – the Wasserstein Mechanism – which is a generalization of the global sensitivity mechanism in differential privacy. The Wasserstein Mechanism considers the following problem. We are given a Pufferfish framework (S, Q, Θ), a (scalar-valued) function f, a dataset D and a privacy parameter ϵ. Our goal is to release an ϵ-Pufferfish private approximation to $f(D)$.

What is the main challenge in designing a generic Pufferfish mechanism? To understand this, let us look back at the global sensitivity method. There, we looked at pairs of datasets D and D' that differ in the private value of a single person, and looked at the maximum achievable distance $|f(D) - f(D')|$; this was our global sensitivity. In Pufferfish, when we consider secrets s_i vs. s_j, instead of a simple distance, we have two distributions $p(f(X)|s_i, \theta)$ and $p(f(X)|s_j, \theta)$; how should we measure the distance between the two?

As it happens, the relevant distance in question is the infinity-Wasserstein distance. For two distributions P and Q, this distance is defined as follows:

$$W_\infty(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \sup_{(x,y) \in supp(\gamma)} |x - y|,$$

where $\Gamma(P, Q)$ is the set of all joint distributions with marginals P and Q.

The Wasserstein Mechanism is provided in Algorithm 1. The main idea is to calculate the maximum, over all secret pairs (s_i, s_j) and all $\theta \in \Theta$, infinity Wasserstein distance between $p(f(X)|s_i, \theta)$ and $p(f(X)|s_j, \Theta)$. If this quantity is W^*, then we add Laplace noise calibrated to W^*/ϵ.

Algorithm 1 The Wasserstein Mechanism

Let $W^* = \max_{i,j} \sup_{\theta \in \Theta} W_\infty(p(f(X)|s_i, \theta), p(f(X)|s_j, \theta)$.
Output $f(D) + \frac{W^*}{\epsilon} Z$, where $Z \sim Lap(0, 1)$.

The main advantage of the Wasserstein Mechanism is of course its generality, but its disadvantage is that because of this extreme generality, it is also quite computationally inefficient. Thus while it might be a useful start for a new Pufferfish privacy framework, in many cases, one may have to design framework-specific mechanisms for computational efficiency.

3.2 Geoindistinguishability

The geoindistinguishability mechanism attempts to publish a location x' which is a perturbation of a location x; the perturbation is follows:

$$x' = x + \frac{r}{\epsilon} Z,$$

where Z is drawn from the two dimensional Laplace mechanism. That is, it is distributed as $p(Z = z) \propto e^{-\epsilon \|z\|}$. Observe that this is an improvement over differential privacy, which would have required noise calibrated to $diam(\mathcal{X})/\epsilon$ instead.

3.3 The Markov Quilt Mechanism

The Markov Quilt Mechanism applies to any Pufferfish framework where secrets S and secret pairs Q are of the form in Example 2, and where the distribution class Θ is based on a Bayesian network. For simplicity, we will describe it for a Markov Chain, which is the most common use case. Given a function f, the Markov Quilt Mechanism calculates an ϵ-Pufferfish approximation to $f(X_1, \ldots, X_T)$ where $X_1 \to X_2 \to \ldots \to X_T$ is a Markov chain. For simplicity, we assume that f has global sensitivity 1 – that is, changing one input changes f by at most 1.

The main idea behind the Markov Quilt Mechanism is as follows. Suppose we would like to obscure the state of the node X_t to an adversary who observes the output of the mechanism – in other words, ensure that the mechanism M ensures:

$$p_{M,\theta}(M(X) = t | X_t = a, \theta) \leq e^{\epsilon} p_{M,\theta}(M(X) = t | X_t = b, \theta), \forall a, b \in A \tag{1}$$

How much noise do we need to add? Since changing X_t can only change f by at most one, if X_t were independent of all other nodes, then it would be enough to add $\approx \frac{1}{\epsilon}$ noise to acheive ϵ-Pufferfish privacy. However, since X_t is correlated with other nodes, we also need to take into account the effect of this correlation. What is this effect? Observe that for most chains, X_{t+1} is highly correlated with X_t, but X_{t+i} for large i is almost independent. In short, nodes close to X_t are highly correlated with it, while the farther we go, the weaker the correlations become.

This suggests the following idea. Suppose we could break up the chain into two parts – nodes that are close to X_t and hence highly correlated with it, and nodes that are far off and hence almost independent. We call the first set the *local set* X_N and we add noise proportional to the size of this set to hide high – even worst case – correlations with X_t. We call the remaining set the *remote set* X_R and we add a correction term to account for the effect of these nodes; the saving grace is that this effect will not be too large.

How do we measure correlations? It turns out that for privacy purposes, the relevant quantity is a metric called max-influence. The max-influence $e_\Theta(X_R | X_t)$ of a node X_t on a set of nodes X_R and a distribution class Θ is defined as:

$$e_\Theta(X_R | X_t) = \max_{a,b \in A} \sup_{\theta \in \Theta} \max_{x_R} \log \frac{\Pr(X_R = x_R | X_t = a, \theta)}{\Pr(X_R = x_R | X_t = b, \theta)}$$

If X_t and X_R are independent, then the max-influence is zero, and low values imply almost independence. Observe that if Θ is a finite set, or a set with some structure, then the max-influence can be calculated. The max-influence is useful for privacy because it can be shown that if X_N and

X_R are the local and remote sets respectively, then the amount of noise that we need to add to obscure the state X_t is $\sigma(X_N, X_R, X_t) = \frac{\|X_N\|}{\epsilon - e_\Theta(X_R|X_t)}$.

To complete the algorithm and ensure it has a good privacy-accuracy tradeoff, it thus remains to find, for each t, sets X_N and X_R that have the lowest $\sigma(X_N, X_R, X_t)$ – which in turn, involves finding large sets of the chain which are almost independent of X_t. Normally, this can be quite daunting, but two properties help signficantly simplify the computation for Markov Chains:

- To hide the effect of X_t, then it is sufficient to search over sets X_N of the following form: $X_{t-a:t+b} = \{X_{t-a}, X_{t-a+1}, \ldots, X_{t+b}\}$.
- The max-influence $e_\Theta(X \setminus X_{t-a:t+b}|X_t) = e_\Theta(\{(X_{t-a}, X_{t+b})\}|X_t)$.

Thus, the algorithm reduces to finding, for each t, a set $\{X_{t-a}, X_{t+b}\}$ that minimizes $\frac{a+b+1}{\epsilon - e_\Theta(\{(X_{t-a}, X_{t+b})\}|X_t)}$. A naive search for each t gives a $O(T^3)$ running time, and for symmetric chains further optimizations can be used to speed it up to $O(T^2)$ or even less. The full algorithm is provided in Algorithm 2.

Algorithm 2 The Markov Quilt Mechanism

for $t = 1, \ldots, T$ **do**
  $\sigma_t = \frac{T}{\epsilon}$.
  for $a = 1, \ldots, t-1$ **do**
    for $b = 1, \ldots, T - t + 1$ **do**
      $\sigma_t = \min(\sigma_t, \frac{a+b+1}{\epsilon - e_\Theta(\{(X_{t-a}, X_{t+b})\}|X_t)})$.
    end for
  end for
end for
$\sigma^* = \max_t \sigma_t$.
Output $f(X_1, \ldots, X_T) + \frac{\sigma^*}{\epsilon} Lap(0, 1)$.

Some remarks are in order. First, the algorithm preserves ϵ-Pufferfish privacy – . Second, unlike the Wasserstein Mechanism for this Pufferfish framework, it is computationally efficient – it can be run in time polynomial in the size of the chain. Finally, it also provides good sequential and parallel composition properties – which general Pufferfish frameworks and mechanisms do not. All these properties make it a good mechanism for use in use-cases involving time-series data.

4 Bibliographic Notes

The Pufferfish privacy framework and its properties are due to [KM14], which also contains a number of other interesting Pufferfish frameworks and associated mechanisms; the No Free Lunch theorem and some of its consequences were shown by [KM11]. Geoindistinguishability is due to [ABCP13]. The Wasserstein and the Markov Quilt Mechanism are due to [SWC17].

References

[ABCP13] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems.

In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914, 2013.

[KM11] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204, 2011.

[KM14] Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems (TODS)*, 39(1):1–36, 2014.

[SWC17] Shuang Song, Yizhen Wang, and Kamalika Chaudhuri. Pufferfish privacy mechanisms for correlated data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1291–1306, 2017.