# MPRI: Foundations of Privacy
# Final Exam

Catuscia Palamidessi, Pablo Piantanida, Ganesh Del Grosso

November 26th, 2019

*Documents authorized. Please use separate sheets between Part I and Part II.*
*Do not copy the questions, turn in the answers only.*
*Specify precisely the question you are answering.*
*Write clearly, and show your reasoning with mathematical rigor.*
*All these statements are important for your score.*

## Part I

### 1) Percentages query (3 points)

In a medical database, we want to allow a query $f$ that reports the percentage of each medical condition, assuming that there are 10 possible such conditions $\{\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_9\}$. For instance, if there are 100 people, of whom 50 are affected by $\mathsf{c}_1$, 15 by $\mathsf{c}_2$, and 35 by $\mathsf{c}_3$, then the answer will be

$$\{(\mathsf{c}_0, 0), (\mathsf{c}_1, 0.5), (\mathsf{c}_2, 0.15), (\mathsf{c}_3, 0.35), (\mathsf{c}_4, 0), \ldots, (\mathsf{c}_9, 0)\}.$$

Assuming that the database contains at least $n$ elements, define a $\varepsilon$-differentially private mechanism for this query, and show that it satisfies indeed the required property ($\varepsilon$-DP). Try to make its utility as high as possible (i.e., don't introduce more noise than necessary).

**Solution** A $\varepsilon$-differentially private mechanism can be achieved by simply adding noise independently drawn from a Laplace to each true percentage. To compute the parameters of the Laplace we need to know the sensitivity of the query $f$. Let $x$ and $x'$ be two adjacent databases ($x \sim x'$) that differ only for one record. Without loss of generality, let us assume that $x'$ contains one more record than $x$, and that the medical condition of the additional record is $c_0$. Let $k_0, k_1, \ldots, k_9$ be the true counts of the medical conditions $\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_9$ in $x$. Then these counts will be $k_0 + 1, k_1, \ldots, k_9$ in $x'$. Observe that the maximum distance between the percentages of $x$ and $x'$ is when the number of records of $x$ is $n$. Let $d$ be the distance between answers of $f$ obtained by summing up the differences in the percentages (corresponding to Manhattan distance). We have:

$$\begin{aligned}
\Delta_f &\overset{\text{def}}{=} \max_{x \sim x'} d(f(x), f(x')) \\
&= \max_{k_0, k_1, \ldots, k_9} \left( \left| \frac{k_0}{n} - \frac{k_0+1}{n+1} \right| + \left| \frac{k_1}{n} - \frac{k_1}{n+1} \right| + \ldots + \left| \frac{k_9}{n} - \frac{k_9}{n+1} \right| \right) \\
&= \max_{k_0} \left( \frac{n-k_0}{n(n+1)} + \frac{1}{n(n+1)} + \ldots + \frac{1}{n(n+1)} \right) \\
&= \frac{n}{n(n+1)} + \frac{1}{n(n+1)} + \ldots + \frac{1}{n(n+1)} \\
&= \frac{n+9}{n(n+1)}
\end{aligned}$$

We can now define a $\varepsilon$-differentially private mechanism $\mathcal{K}_f$ for $f$. Let the true answer of $f$ on $x$ be Let $f(x) = \{(\mathsf{c}_0, a_0), (\mathsf{c}_1, a_1), \ldots, (\mathsf{c}_9, a_9)\}$, and let $z = \{(\mathsf{c}_0, b_0), (\mathsf{c}_1, b_1), \ldots, (\mathsf{c}_9, b_9)\}$ be another generic answer. Let $r_i \overset{\text{def}}{=} |a_i - b_i|$ for $i = 0, 1, \ldots, 9$ and let $r \overset{\text{def}}{=} \sum_i = 0^9 r_i$. The mechanism $\mathcal{K}_f$ is characterized by its probability density function $D$, defined as follows:

$$D(\mathcal{K}_f(x) = z) \overset{\text{def}}{=} c\, e^{\frac{r}{\Delta_f}\varepsilon}$$

where $c$ is a normalization constant defined so that the integration of $D$ on all domain is 1. Since the normalization constant for each individual Laplace is $\frac{\varepsilon}{2\Delta}$, we have $c = 10\,\frac{\varepsilon}{2\Delta_f} = 5\,\frac{\varepsilon}{\Delta_f}$.

We now prove that $\mathcal{K}_f$ is $\varepsilon$-differentially private. Let $f(x)$ and $z$ be defined as above, and let $x'$ be a database adjacent to $x$. Let $f(x') = \{(\mathsf{c}_0, e_0), (\mathsf{c}_1, e_1), \ldots, (\mathsf{c}_9, e_9)\}$, and let $s_i \overset{\text{def}}{=} |e_i - b_i|$ for $i = 0, 1, \ldots, 9$ and $s \overset{\text{def}}{=} \sum_i = 0^9 s_i$.

$$\begin{array}{rcl} D(\mathcal{K}_f(x) = z) & = & c\, e^{\frac{r}{\Delta_f}\varepsilon} \\ & \leq & c\, e^{\frac{s}{\Delta_f}} e^{\frac{|r-s|}{\Delta_f}\varepsilon} \\ & \leq & c\, e^{\frac{s}{\Delta_f}} e^{\varepsilon} \\ & = & e^{\varepsilon}\, D(\mathcal{K}_f(x') = z). \end{array}$$

## 2) Non-numerical queries (4 points)

Consider a non-numerical domain $\mathcal{Y}$ (for instance, a set of fruit types) and a population where each individual has a certain secret (for instance a favorite fruit) which he wants to protect, but also gets some utility from. Suppose that we have a "score" function $u : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ (where $\mathbb{R}$ is the domain of the reals) which represents the usefulness of a reporting a certain fruit type with respect to the true preferred one. For instance, if the true favorite fruit is "*apple*" then $u(\mathit{apple}, \mathit{apple}) = 1$, $u(\mathit{apple}, \mathit{pear}) = 0.9$ and $u(\mathit{apple}, \mathit{strawberry}) = 0.5$ represent the fact that reporting "*apple*" is better (from the utility point of view) than reporting "*pear*", and the latter is much better than reporting "*strawberry*". A typical mechanism used in this case, to balance privacy and utility, is the one that, when the true favorite fruit is $y$, reports $z$ with probability

$$p(z|y) = \alpha_y \cdot e^{\frac{u(y,z)}{\beta_y}}$$

where $\alpha_y$ and $\beta_y$ are suitable values that may (or may not) depend on $y$.

Determine the values of $\alpha_y$ and $\beta_y$ so that the above is a $\varepsilon$-differential-private mechanism in the local model, and find (the smallest) such $\varepsilon$.

**Solution** Let

$$\Delta_u \overset{\text{def}}{=} \max_{y,y',z} \left(u(y,z) - u(y',z)\right)$$

and

$$\beta_y \overset{\text{def}}{=} \frac{2\Delta_u}{\varepsilon}$$

Note that $\beta_y$ does not depend on $y$. As for $\alpha_y$, it is simply a normalization factor, namely it is defined so that the summation of $p(z|y)$ on all $z$ is 1. Hence:

$$\alpha_y \overset{\text{def}}{=} \frac{1}{\sum_z e^{\frac{u(y,z)}{2\Delta_u}\varepsilon}}$$

We show now that the mechanism is differentially private. We start by observing that, for all $y, y, z$:

$$e^{\frac{u(y',z)}{2\Delta_u}\varepsilon} \leq e^{\frac{u(y,z)}{2\Delta_u}\varepsilon} \cdot e^{\frac{|u(y',z)-u(y,z)|}{2\Delta_u}\varepsilon} \leq e^{\frac{u(y,z)}{2\Delta_u}\varepsilon} \cdot e^{\frac{\varepsilon}{2}}$$

Hence:

$$\frac{\alpha_y}{\alpha_{y'}} = \frac{\sum_z e^{\frac{u(y',z)}{2\Delta_u}\varepsilon}}{\sum_z e^{\frac{u(y,z)}{2\Delta_u}\varepsilon}} \leq \frac{\sum_z e^{\frac{u(y,z)}{2\Delta_u}\varepsilon}}{\sum_z e^{\frac{u(y,z)}{2\Delta_u}\varepsilon}} e^{\frac{\varepsilon}{2}} = e^{\frac{\varepsilon}{2}}$$

Therefore, we derive:

$$\frac{p(z|y)}{p(z|y')} = \frac{\alpha_y \cdot e^{\frac{u(y,z)}{2\Delta_u}\varepsilon}}{\alpha_{y'} \cdot e^{\frac{u(y',z)}{2\Delta_u}\varepsilon}} \leq e^{\frac{\varepsilon}{2}} \cdot \frac{e^{\frac{u(y,z)}{2\Delta_u}\varepsilon}}{e^{\frac{u(y',z)}{2\Delta_u}\varepsilon}} = e^{\frac{\varepsilon}{2}} \cdot e^{\frac{u(y,z)-u(y',z)}{2\Delta_u}\varepsilon} \leq e^{\frac{\varepsilon}{2}} \cdot e^{\frac{|u(y,z)-u(y',z)|}{2\Delta_u}\varepsilon} \leq e^{\frac{\varepsilon}{2}} \cdot e^{\frac{\varepsilon}{2}} = e^{\varepsilon}$$

## 3) Information flow in dice-tossing (3 points)

Alice and Bob are playing a game: Alice tosses two dices and tells Bob the sum of the two results, without telling him the individual result of each dice. Bob then has to guess the individual result of each dice. If he guesses correctly, then he wins and he gets a sum $\alpha$ from Alice. If he guesses wrongly, then Alice wins and she gets a sum $\beta$ from Bob. We assume that the prior distribution on the dices is uniform (for each dice, any of the results from 1 to 6 is equally probable) and that the dices are tossed independently (the result of the one does not influence the result of the other).

**Solution** All the configurations of values for the two dices are equally probable, hence the probability $p(x)$ of each configuration $x$ is

$$p(x) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$$

The prior vulnerability $V_{\text{prior}}$ therefore is

$$V_{\text{prior}} = \frac{1}{36}$$

.

We now compute the posterior vulnerability $V_{\text{post}}$, i.e. the probability to guess correctly the pair $x$ from their sum $y$. The possible sums are $y = 2, 3, \ldots, 12$. Hence:

$$V_{\text{post}} = \sum_{y=2}^{12} \max\left(p(y|x) \cdot p(x)\right) = \frac{1}{36} \cdot \sum_{y=2}^{12} \max p(y|x) = \frac{1}{36} \cdot \sum_{y=2}^{12} 1 = \frac{11}{36}$$

Therefore, the min-entropy leakage $\mathcal{L}$ is:

$$\mathcal{L} = \log \frac{V_{\text{post}}}{V_{\text{prior}}} = \log \frac{11 \cdot 36}{36} = \log 11.$$

## Premium question: 2 extra points

In Question (3) above, what should be the values of $\alpha$ and $\beta$ so that the game is *fair*? By "*fair*" we mean that, in the long run, the amount of money won by Alice and that won by Bob tend to be the same.

**Solution** From previous solution we know that Bob is expected to win in average 11 times out of 36. This means that, out of 36 times, Bob is expected to win 11 times and Alice 25 times. Hence in order to be a fair game, it should be

$$\alpha = \frac{25}{11} \cdot \beta.$$

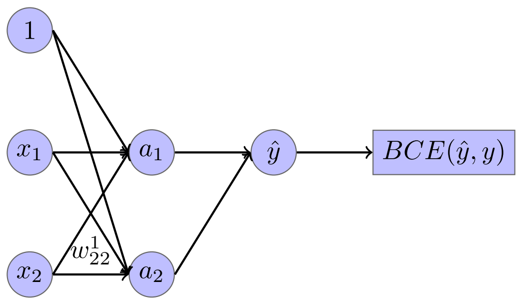

Figure 1: Small Neural Net.

| **Apartment** | $x_1$ | $x_2$ | $\hat{y}$ |
|---|---|---|---|
| 1 | 1.200 | 2.0 | 1 |
| 2 | 1.100 | 1.2 | 1 |
| 3 | 0.500 | 4.0 | 0 |
| 4 | 0.700 | 3.0 | 0 |

Table 1: Training set for the apartment classification algorithm.

# Part II

## 1) Generative Adversarial Networks (GANs) (2 points)

Let $D(\cdot)$ denotes the discriminator network and $G(Z)$ the generator from random noise $Z$. Given a random value $z$, $D(G(z))$ is a value between 0 and 1 representing the confidence of $D(\cdot)$ that the input is generated by $G(Z)$ (0) or a sample from the real distribution (1).

(i) Early in the training, is the value of $D(G(z))$ closer to 0 or closer to 1? Explain.

(ii) The GAN is trained when $D(G(z))$ is close to 1. Is this assertion *True* or *False*? Explain.

(iii) A common method to accelerate the training of GANs is to update the Generator $k\,(> 1)$ times for each time you update the Discriminator. Is this assertion: *True*, *False* or *It depends on the architecture of the GAN*. Explain.

**Solution**

(i)

(ii) False. At the end of training the discriminator should be unable to tell between natural and artificial samples; therefore, the output of the discriminator should be close to 0.5 for both natural and artificial inputs.

(iii) False. This is not a common method. In fact the opposite is often seen in practice, that is, the discriminator is updated $k$ times more than the generator. Nevertheless, the number of updates per training iteration of the discriminator in relation to the number of updates per training iteration of the generator is a parameter to tune in order to stabilize training and any choice of this relation can be valid as long as it helps stabilize training.

## 2) Back-propagation (4 points)

We are trying to train an algorithm that classifies apartments according to their location. Two classes are considered: inside of Paris (labeled 1), and outside of Paris (labeled 0). The training set is described in table 1, where $x_1$ corresponds to the rent per month in euros, and $x_2$ corresponds to the surface of the house in square meters.

The values $x_1$ and $x_2$ are divided by $1,000$ and $10$ respectively to have comparable values for both features. To solve this classification problem, the neural net depicted in figure 1 is proposed. At each node (circles in the figure), the inputs to the node are multiplied by a weight summed and passed through the sigmoid activation function ($\sigma(x) = \frac{1}{1+e^{-x}}$). The Binary Cross-Entropy $BCE$ is chosen as the loss function to train this algorithm, where $BCE(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$.

(i) Run a training epoch to learn the value of $w^1_{22}$, by:

- Naming and initializing the weights. Initialize the biases as 0, the weights that multiply $x_1$ as 1 and the weights that multiply $x_2$ as $-1$. Initialize the weights of the output layer as 0.5.
- For each example:
  - Compute the forward pass. For simplicity, consider:

$$\sigma(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0.5 \text{ if } x = 0 \\ -1 \text{ if } x < 0 \end{cases}$$

  - Compute the loss, $BCE(\hat{y}, y)$.
  - Compute the backward pass for $w^1_{22}$ (derivative of the loss with respect to $w^1_{22}$).
- Average the derivatives of the loss with respect to $w^1_{22}$.
- Update the weight using the Gradient Descent algorithm with learning rate $\lambda = 1$.

(ii) Does this correspond to Gradient Descent, Stochastic Gradient Descent or Stochastic Gradient Descent with mini-batches? Explain.

(iii) Justify the choice of the Binary Cross-Entropy as the loss function for this problem.

**Solution**

(i)
- On the forward pass: $a_1(x_1, x_2) = a_2(x_1, x_2) = \sigma(x_1 - x_2)$. As $x_2 > x_1$ for all data points, $a_1 = a_2 = 0$ for all data points.
- Then, $\hat{y}(a_1, a_2) = \sigma(\frac{a_1+a_2}{2})$. As $a_1 = a_2 = 0$ for all data points, $\hat{y} = 0.5$ for all data points.
- As $\hat{y} = 0.5$ for all data points, $BCE(y, \hat{y}) = -\log(1/2) = 1$ for $y = 1$ and $y = 0$.
- On the backward pass: By the chain-rule,

$$\frac{\partial BCE}{\partial w^1_{22}} = \frac{\partial BCE}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_2} \frac{\partial a_2}{\partial w^1_{22}} . \tag{1}$$

- The first factor on the right side of equation (1) is,

$$\frac{\partial BCE}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{(1-y)}{1-\hat{y}} ,$$

  for $y = 1$ this is $-2$, for $y = 0$ this is 2.
- The second factor on the right hand side of equation (1) is,

$$\frac{\partial \hat{y}}{\partial a_2} = -\frac{1}{2}\sigma\left(\frac{a_1+a_2}{2}\right)\sigma\left(1 - \frac{a_1+a_2}{2}\right) ,$$

  for all data points $a_1 = a_2 = 0$, then $\frac{\partial \hat{y}}{\partial a_2} = \frac{1}{4}$
- The last factor on the right hand side of equation (1) is,

$$\frac{\partial a_2}{\partial w^1_{22}} = x_2\, \sigma(x_1 w^1_{12} + x_2 w^1_{22} + b_2)\sigma(1 - x_1 w^1_{12} - x_2 w^1_{22} - b_2) ,$$

  since $x_1 w^1_{12} + x_2 w^1_{22} + b_2 < 0$, $\frac{\partial a_2}{\partial w^1_{22}} = 0$ for all data points.
- Finally $\frac{\partial BCE}{\partial \hat{y}} = 0$ for all data points. Note this is due to the approximation we use. Using this approximation, our network would not be able to learn given this initialization.

(ii) Gradient Descent, as we perform an update after evaluating over the whole training set.

(iii) Binary Cross-entropy is typically chosen for binary classification problems. Derived from the KL-divergence, it measures the similarity between the target distribution and the empirical distribution.

## 3) Privacy and Neural Networks (4 points)

(i) Membership inference attacks measure the privacy of deep learning algorithms. *True or False*? Explain.

(ii) What are the privacy risks for members of the training set of a deep learning algorithm, when an attacker has access to use the algorithm?

(iii) Provide an example where there is a compromise and an example where both performance and privacy can be optimized at the same time.

(iv) An attacker has white-box access to a deep learning classification algorithm, including its parameters and architecture. List and explain (at least two ways) in which the attacker could take advantage of these resources.

(v) An attacker has black-box access to a generative model, i.e. the attacker can fetch a finite number of artificial samples. Provide an example of how the attacker could use artificial samples to launch a membership inference attack.

(vi) A neural network has been encrypted on a device, you can access neither its architecture, nor the values of its parameters. Is it possible to create an adversarial example to attack this network? Explain why.

**Solution**

(i) True. Membership privacy can be considered as a minimum privacy requirement. Furthermore, membership privacy is closely related to and can imply differential privacy.

(ii) Membership inference, which can itself represent a privacy breach or lead to further attacks (e.g. feature inference).

(iii) In the case of generative models, both privacy and performance can be optimized simultaneously, because the goal is for the generator to generalize well, and this protects the privacy of members of the training set. On the other hand, classifiers want to provide high confidence on members of the training set, and this usually hurts generalization, and in turn privacy.

(iv) Output of the classifier: The classifier is expected to provide a higher confidence on members of the training set. The attacker could use this property to sort candidates into members and non-members. Gradient of the loss function of the classifier with respect to its model parameters: The classifier is optimized on members of the training set; therefore the gradient of the loss function should be lower on members of the training set. The attacker could use this property to sort candidates into members and non-members.

(v) The attacker could search for similarity between properties of generated samples and members of the training set of the corresponding generator. Alternatively, the attacker could use the generated samples to train a GAN and perform a white-box attack on the discriminator of such GAN.

(vi) Yes. Most methods for crafting adversarial examples rely on computing the gradient of the loss function of the neural network with respect to its input. Using the finite difference approximation, an adversary could compute gradients without need for the architecture or parameters of the network.