UNIVERSITY OF TWENTE.

Faculty of EEMCS: Electrical Engineering, Mathematics and Computer Science

Course program in Applied Mathematics

Specialization in Mathematics of Data Science (MDS)

Master's thesis

Membership Inference Attacks and Synthetic Data Generation with Differential Privacy

Advisors:
Dr. **Jins De Jong (TNO)**
Dr. **Cristóbal Guzmán (UT)**

Canditate:
Giorgio Micali

Enschede, August 2022

Acknowledgements

There are way too many people who would like to thank for all they have done for me during these years.

I would like to start by thanking my advisors, Cristóbal and Jins, whose support and intellectual insides were enormously precious to me and for this work. A special acknowledgment goes to Cristóbal in particular, for his availability at the most absurd times and days, for his patience and human touch. Thank you for all your support, I will not forget it.

I would like to thank all my friends, starting with the ones in The Netherlands, Thomas, Bohzo, Mihir and Riccardo. Thank you all for the moments together, the long discussions and the exchange of ideas and fun we had together. Our friendship made me feel welcomed and more bounded to The Netherlands.

Among my Italian friends: I own a lot of what I have achieved so far to Vincenzo Mariani, who always encouraged me to push myself beyond my limits. Most of what happened in the last years was only possible because of him. Likewise, Giuseppe Recupero and Filippo Testa, my ex roommates and dear friends. I had countless hours of valuable discussions and ideas with both of them. They will forever remain a primary source of wisdom and intelligence I can always rely on. Then, my dear friends Rosalba, Alessandra, Daniele and Massimiliano, with whom I spent most of my days back in Pisa. They significantly contributed to my mathematical grow.

I also want to thank my friends around the world, Gabriele e Pierfrancesco, always a great source of knowledge and inspiration. Lastly, I would like to thank my childhood friends, who have accompanied me from the begging of this journey: Emanuele, Simona, Alessandra, Gabriele, Gianpaolo, Salvatore M.D.G and Francesco.

My family has played a crucial role for which I will forever be in debt. I am grateful for all the resources, time and patience you dedicated to me, especially my sister Cristina, the person that has always given me everything I needed. Also, an acknowledgement goes to my cousin Massimilano, who also encouraged me.

Lastly, I want to dedicate this thesis to Kaja, who believed in me the most, starting from the very begging of this work until the end. Thank you for your patience, wisdom, support and intellectual stimulation.

Without you all, this would have not been possible. I hope you enjoy reading this thesis,

Giorgio Micali.

Abstract

Machine learning (ML) is one of the most popular methods for data analysis. A carefully chosen design supplied with a set of training data yields a predictive model for the problem. However, it has been documented that parts of the training data or even complete records can be extracted from the model, which exposes the model to privacy attacks. One such example is a membership inference attack (MIA), where, based on the trained model, the attacker tries to determine whether a single record was in the training data or not. In order to satisfy strong and quantifiable privacy guarantees, differential privacy (DP) is the preferred tool because it introduces randomization in the algorithm, which obfuscates the private data in the model. However, noise and randomness reduce the utility of the data analysis. This means that a balance must be found between privacy and utility. How to resolve this trade-off is a highly non-trivial question, which is the main objective of this project.

To show that such combination is possible, we worked on two aspects simultaneously: *attacking* and *defending* the model. An **attack** takes advantage of the fact that ML models are trained multiple times over the same train data set. As a result, the outcome of a point in the data set can be predicted more easily. A **defence** is therefore needed as a response for the increased concerns about the privacy of individuals whose data is used during the training.

In more details, we show that the effectiveness of a MIA is reduced when the attacked ML model is being trained using a DP-learning algorithm. Particularly, using stochastic gradient descent (DP-SGD) and the DP-Frank Wolfe (DP-FW) which are suitable for privatization. Concretely, we will run two metric based attacks on three simple ML models that were trained using both DP-FW and DP-SGD, and later compare the results of **Linear Regression, Logistic Regression** and **Multiclass Logistic Regression.**

The results of implementation of DP show a decrease of the effectiveness of these attacks, without compromising too much the target accuracy of the ML models. In particular, both DP-FW and DP-SGD show satisfactory privacy protection, but DP-SGD has better computational performances, and it also converges faster. Furthermore, the effectiveness of our MIAs is model dependent. The most meaningful results are obtained on Multiclass Logistic Regression, where the defence through DP is sharper. For other models, the effects of DP are only clearly visible when we impose additional assumptions on the data sets used for both training and testing. Without these assumptions, MIA is barely distinguishable from a *random guess attack*.

The experiments for Logistic and Multiclass Logistic regression also show that the attacks are highly sensitive to changes of a threshold, which measures how much the dataset over-fitted the model. Therefore, such threshold depends on the data and it is needed for building an adequate metric based attack. In principle, this information must be kept private as well. Therefore, we will also explain how to set such threshold adequately. In the second part of the thesis, we switch points of view. Instead of creating a

mechanism that guarantees privacy computation over a data set, artificial data is generated with the purpose of preserving privacy. Synthetic data is created by using different types of algorithms, such as Multiplicative Weights. The output is a dataset whose statistical properties are similar to the original data, but does not reveal any information regarding real data. More specifically, we create synthetic data by minimizing the error we commit when querying the dataset over a fixed set of statistical queries. Such formulations yields a saddle point optimization problem, for which different types of regularization are used. Since real datasets usually do not contain many repetitions of the same individual's data, it was given more focus to regularizations that promote high entropy.

Contents

Introduction

As the collection and analyses of sensitive data become more prevalent, there is an increasing need to protect individuals private information. Differential privacy (DR13) is the rigorous tool that defines the mathematical notion of privacy.

This work primarily focuses on two privacy related problems: analysis of membership inferences attacks and synthetic data generation.

In the next section we present an overview of what these two problems are, and how to design privacy preserving strategies. We can consider this introduction as an overview of what comes in the next chapters, giving a first glimpse and guiding the reader through the main results of this work.

Firstly, we present the mathematical background necessary to understand what are the goals of this work. This includes differentially private stochastic optimization, membership inferences attacks and synthetic data generation. Lastly, we will clearly state what we achieved, at the end of every section.

Differential privacy

As stated in the abstract, *differential privacy* is a new tool introduced to protect leakage of information from machine learning models. We will informally introduce the idea which leads to the mathematical definition of privacy.

Let us consider the situation described in Figure 1. Suppose there is an analyst who wants to perform some statistics on a dataset. Moreover, suppose the data analyst has full access to these data. What normally happens is that the data analyst queries the dataset and, in return, he extracts information that are later processed. A privacy issue arises in the moment that this dataset contains sensitive information, as the analyst would get full and direct access to the data, Figure 1 (UP).

A possible solution could be the following: when the analyst queries the dataset, some sort of *mechanism* adds noise to the data. To be precise, it is enough noise to protect individual's privacy and not too much to compromise the quality of the statistics either. In other words, the mechanism takes as input the query of the analyst and releases back noisy information from the dataset.

In this way, both goals are achieved: the privacy of the data is protected and the data analyst can perform his statistics, even if the data are sensitive. This is the basic intuition behind differential privacy. Now, we shall formalize this idea.

Idea behind the definition

Consider two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ in **one** individual. We write this condition as $\mathcal{D}_1 \sim \mathcal{D}_2$. Let $\mathcal{A}$ be a function of the dataset whose output releases noisy data. We call such function a *mechanism*.

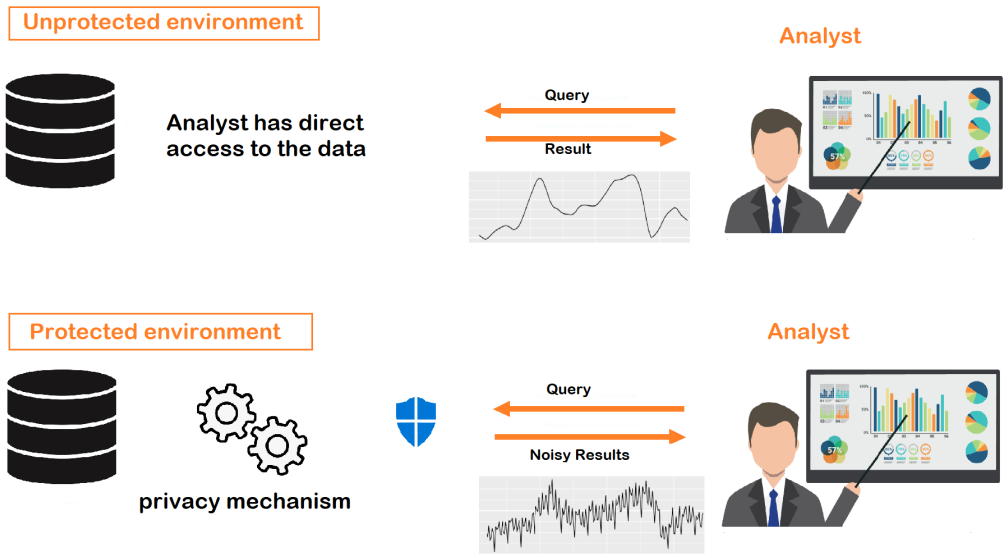


Figure 1: Two scenarios: (UP) the data analyst has full access to the dataset, violating privacy. (DOWN) the information are protected using a mechanism that slightly alter the data.

If the outcomes of the mechanism on both datasets are similar — we write $\mathcal{A}(\mathcal{D}_1) \cong \mathcal{A}(\mathcal{D}_2)$ — then *data analyst* cannot determine whether $\mathcal{D}_1$ or $\mathcal{D}_2$ was used.
This means that he did not learn much about the individual in which they differ. This motivates the definition of differential privacy

Definition 1. (DR13) A mechanism $\mathcal{A}$ is (ε, δ) differentially private if for every $\mathcal{D}_1 \sim \mathcal{D}_2$ and for every set of responses T

$$\mathbb{P}(\mathcal{A}(\mathcal{D}_1) \in T) \leq e^{\varepsilon}\mathbb{P}(\mathcal{A}(\mathcal{D}_2) \in T) + \delta \, .$$

This definition captures the idea that the probability of obtaining two different outcomes is *roughly* the same, where *roughly* refers exactly to that multiplicative factor $\exp(\varepsilon)$. There is also a small correcting term δ - whose meaning will be explained later in chapter 1. For the moment, the take-home message is that the smaller ε and δ are, the higher is the privacy, since the two probabilities on left-hand-side (LHS) and right-hand-side (RHS) would be the same. Furthermore, one might ask why something as odd as $\exp(\varepsilon)$ would be chosen as a regulating factor, while another term, mathematically easier, could have been picked, like $1 + \varepsilon$, without changing the meaning of the definition. It turns out that the formulation with $\exp(\varepsilon)$ is rather convenient when *composing* more private mechanisms. The total level of privacy results to be the algebraic sum of all the single privacy budgets.

The whole chapter 1 is dedicated to formally introduce the theory of differential privacy. At high level, privacy means adding noise to protect data.

Where does differential privacy come into the picture of membership inference attacks (MIA) and synthetic data generation (SD)? It turns out that both MIA and SD boil down to solving an optimization problem, which is formulated in terms of the sensitive information contained in the dataset. That is why we will make use of what is known as *differentially private convex optimization*, object of the next section.

Differentially private stochastic convex optimazation

The goal of machine learning is to output a set of parameters w that minimizes a *loss* function. For example, this happens in linear regression or neural networks. To be more specific, we usually minimize a function of of the form $\frac{1}{|\mathcal{D}|}\sum_i l(w, d_i)$, where w is the parameter we are taking the minimization over, $\mathcal{D} = \{d_1, \cdots, d_N\}$ is the dataset of points, and $l(\cdot, d_i)$ is the *loss* on each datapoint.

However, this minimization may reveal sensitive information. For instance, this phenomena is evident Support Vector Machine, where it is well known that the optimal parameter vector is entirely determined by the data points which are closest to it — the support vectors. Removing or adding one of these points shifts the parameter vector, again violating privacy.

Differentially private stochastic convex optimization (DP-SCO) is one way of performing the training (learning phase) in machine learning models, while protecting privacy. More specifically, DP-SCO privatizes the set of parameters defining the model, with respect to the training dataset $\mathcal{D}$. Mathematically:

Definition 2. Let $\mathcal{C}$ be a convex and compact set. $\mathcal{D}$ dataset of points containing sensitive information and $Z \sim \mathcal{P}$ r.v whose support is in $\mathcal{D}$.

Population Risk: Let $\mathcal{P}$ a distribution over $\mathcal{D}$ and $l : \mathcal{X} \times \mathcal{D} \longrightarrow \mathbb{R}$. We define the population risk as

$$\mathcal{L}(w; D) = \mathbb{E}_{Z\sim\mathcal{P}}[l(w, Z)] \ .$$

When $\mathcal{P} = U(\mathcal{D})$, $\mathcal{L}$ is called Empirical Risk.

SCO: Let $\mathcal{C} \subset \mathcal{X}$ be a convex and closed set (note that we do not assume compactness in the definition). We define of minimizing the population risk

$$\min_{w\in\mathcal{C}} \mathcal{L}(w; D)$$

as **S**tochastic **C**onvex **O**ptimization.

DP-SCO: The problem of determining a DP private mechanism $\mathcal{A}(\mathcal{D}) \in \mathcal{X}$ such that

$$\mathbb{E}_{\mathcal{A}}[\mathcal{L}(\mathcal{A}(\mathcal{D}); \mathcal{D}) - \mathcal{L}^\star] \xrightarrow{N\longrightarrow\infty} 0$$

is called Differentially Private Stochastic Convex Optimization.

DP-SCO is at the core of the two problems we want to address in this thesis (not yet presented). Nevertheless, how do we design a differentially private learning algorithm? How do we concretely privatize learning? In the context of DP-SCO, the noise to ensure privacy can either be injected once the learning is completed (output perturbation) or during the learning (gradient perturbation). Gradient perturbation is the approach we will be following, (BST14).

Idea behind gradient perturbation

Let us consider for a moment the gradient descent algorithm, whose steps are listed below

Choose w_0

For $t = 0 : T - 1$ **do**.

1. Compute $\nabla\mathcal{L}(w_t)$
2. $w_{t+1} = w_t - \gamma\nabla\mathcal{L}(w_t)$

End

Since computing the gradient $\nabla\mathcal{L}(w_t) = \frac{1}{N}\sum_i \nabla l(w_t, d_i)$ directly involves the individuals in the dataset d_i, which are supposed to be private, an idea could be to add noise to the gradient. Concretely, the step 2. would be replaced by $w_{t+1} = w_t - \gamma(\nabla\mathcal{L}(w_t) + noise)$ where, intuitively, noise should be a smallish term, so that the new direction does not deviate too much from the steepest descent one. We add white noise with appropriately tuned variance, obtaining:

Algorithm 1 DP-SGD

1: **Input**: $(\mathcal{D}, L_0, \mathcal{C}, \varepsilon, \delta)$
2: $\sigma = \frac{32N^2\log(N/\delta)\log(1/\delta)}{\varepsilon^2}$
3: select $w_0 \in \mathcal{C}$
4: **for** $t = 1 : N^2$ **do**
5: pick $d \sim U(\mathcal{D})$
6: $w_{t+1} = \pi_{\mathcal{C}}(w_t - \gamma(\nabla l(w_t, d) + \xi)) \quad \xi \sim \mathcal{N}(0, \sigma^2)$
7: **Output**: $w_{DP} = w_{N^2}$

The projection is needed since the updated point w_{t+1} might end up outside the set $\mathcal{C}$. In that case we have to project it back and restart the iterations.

It can be shown that introducing this Gaussian noise makes the final output (ε, δ) private. The proof makes use of advanced tools that are presented in chapter 1.

Moreover, using the following theorem, we can also prove that Algorithm 1 converges:

Theorem 3. *(SZ13) Let $\mathcal{L}$ be a convex function and $w^\star = \arg\min_{w\in\mathcal{C}} \mathcal{L}(w)$. Consider the iterations $w_{t+1} = \pi_{\mathcal{C}}(w_t - \gamma G_t(w_t))$ where $\mathbb{E}[G_t(w_t)] = \nabla\mathcal{L}(w_t)$ and $\mathbb{E}[||G_t(w_t)||_2^2] \leq G^2$, with learning rate $\gamma_t = \mathcal{C}_2/(G\sqrt{t})$. Then, for $T > 0$ iterations, we have*

$$\mathbb{E}[\mathcal{L}(w_t) - \mathcal{L}^\star] = \mathcal{O}\left(\frac{\mathcal{C}_2 G\log T}{\sqrt{T}}\right).$$

In fact, in step 6 : of Algorithm 1 we are using an unbiased estimator of the gradient.

Despite the good performances of SGD, in this work, we will focus on a specialized algorithm, DP-Frank Wolfe algorithm. Why do we need to consider a second algorithm? The reason is that, SGD suffers from the high dimensionality of the data, by a factor proportional to $\mathcal{O}(\sqrt{d})$, whereas Frank Wolfe reduces it to a factor proportional to $\mathcal{O}(\log(d))$. Moreover, Frank Wolfe is projection-free, which makes the implementation easier. In fact, unlike SGD, Frank-Wolfe does not create iterates that move along the steepest directions. Given a convex loss function $\mathcal{L}$ over convex and compact set $\mathcal{C}$, we consider the first order Taylor expansion at a given point w_t. By convexity, the graph of its linearization lies all below the graph of the loss function $\mathcal{L}(w_t)$. Then, we find the minimizer $s_t = \arg\min_{s\in\mathcal{C}}\langle\nabla\mathcal{L}(w_t), s - w_t\rangle$ and move towards the direction determined by s_t, i.e $w_{t+1} = w_t + \gamma(s_t - w_t)$. In this way, we are performing convex updates. Moreover, the iterations are kept within $\mathcal{C}$, thus the projection is not needed anymore. Lastly, from this moment on, we always assume that $\mathcal{C}$ is polyhedron for reasons that will be explained later. In practice, we run the following steps

For $t = 1 : T$ **do**.

$s_t = \underset{s \in \text{vertex}(Q)}{\arg\min} \ (\langle \nabla \mathcal{L}(q_t), s \rangle)$

$\gamma = \frac{2}{2+t}$

$q_{t+1} = q_t + \gamma_t(s_t - q_t)$

End

Exactly as in SGD, we are using the information contained in the dataset while computing the gradient of the loss $\mathcal{L}$. Again, the privatization is done by gradient perturbation, obtaining

Algorithm 2 DP-Frank Wolfe

1: **Input**: $(\mathcal{D}, L_0, \mathcal{C}_1, T, \varepsilon, \delta)$
2: select $w_0 \in \mathcal{C}$
3: **for** $t = 1 : T$ **do**
4: $\quad s_t = \underset{s \in \text{Vertex}(\mathcal{C})}{\arg\min} \ (\langle \nabla \mathcal{L}(w_t, \mathcal{D}), s \rangle + u_s)$ where $u_s \sim \text{Lap}(\lambda)$
5: $\quad w_{t+1} = w_t + (s_t - w_t)\gamma_t$ where $\gamma_t = \frac{2}{2+t}$
6: **Output**: $w_{DP} = w_T$

This time the white noise was introduced by Laplacian random variable with zero mean. Similarly to the SGD, we can prove that the noise introduced in one iteration privatizes the whole chain of iterations, making the output w_{DP} completely (ε, δ) differentially private.

Throughout the thesis, we will always use the Frank Wolfe framework, both for membership inference attacks and synthetic data generation. It is worth to double remark the answer to the question *why would we use something as odd as this Frank Wolfe when we have already DP-SGD available?* As we said, Frank Wolfe does not need a projection to be implemented. This constitutes an advantage since implementing the projection is rather complicated. Furthermore, since our set of constraints $\mathcal{C}$ is assumed to be polyhedron, Frank Wolfe presents an advantage: if we compare the utility guarantees of the two algorithms for the same number of iterations we get

Gradient Descent	Frank Wolfe
$\mathbb{E}[R(w_T)] \leq \mathcal{O}\left(\frac{\sqrt{d\log(1/\delta)}}{N\varepsilon}\right)$	$\mathbb{E}[R(w_T)] \leq \mathcal{O}\left(\frac{\log(\lvert vertex(\mathcal{C})\rvert)\sqrt{T\log(1/\delta)}}{N\varepsilon}\right)$

Table 1: Excess risk comparison between Stochastic gradient descent and Frank Wolfe using the same number of iteration. If the dataset is made of points that belog to $\mathbb{R}^d$, a polyedron in $\mathbb{R}^d$ needs to have $\mathcal{O}(d)$ number of vertices

The number of vertices must be of the same order as the dimension on the space $\mathbb{R}^d$, otherwise the relative interior of $\mathcal{C}$ would be empty. This means, that for highly dimensional data, the factor $\log(d)$ in the Frank Wolfe upper bound increases more slowly than $\sqrt{d}$, hence Frank Wolfe has stronger guarantees on the excess risk. Moreover, the d factor appearing in the upper bound of the gradient descent is solely determined by the Gaussian mechanism we used to perturb the gradient, whereas Frank Wolfe controlled the convergence by adding noise for every vertex.

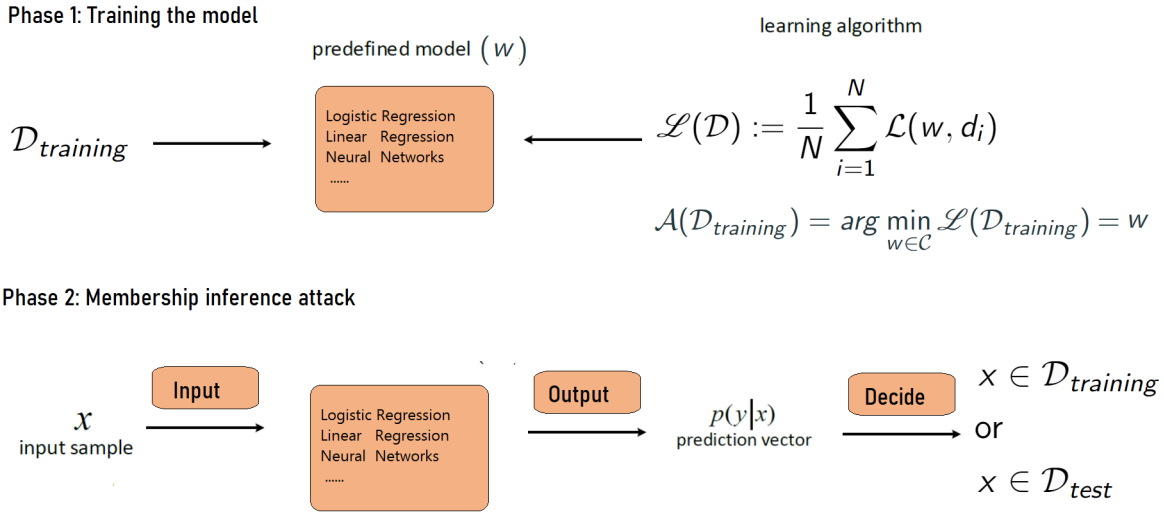


Figure 2: Scheme of a membership inference attack. UP: standard scheme of a statistical model. DOWN: once the model has been trained, an input sample x is fed as input; the attack tries to recover the membership of x from its output $p(y|x)$.

In the next two sections we will see how the theory of DP-SCO naturally appears and applies for membership inference attacks and synthetic data generation.

Membership inference attacks

Let us consider the situation in Figure 2 (up). Suppose we are given a classic statistical model, for instance logistic regression. Our goal is to find the set of parameters defining the model; in our case this means that we want to find $w \in \mathbb{R}^d$ for $p(y|x) = \frac{1}{1+\exp -w\cdot x}$. To this purpose, we will be given a data set $\mathcal{D}_{training}$ that we feed to the model, collecting losses $\mathscr{L}(w, d_i)$ for every data point. Summing over all the losses and averaging by the number of points in the datset, we get the *loss* function we want to minimize. The argmin of $\mathscr{L}$ is the vector of parameters we are looking for. This phase is called *Training the model*.

Now, suppose we have completed the training phase and we want to test the model. Moreover, suppose that $\mathcal{D}_{training}$ is drawn from a distribution we know, so that we can create new dataset $\mathcal{D}_{test}$ sampled from the same distribution. In principle, these two datasets have the same properties and they are indistinguishable.

If the the training was done appropriately, we possess an algorithm which classifies points into classes. Therefore, if we draw $x \in \mathcal{D} = \mathcal{D}_{training} \cup \mathcal{D}_{test}$ and feed it to the model, the model returns an output — in this example a probability vector — telling us what is the class where x should belong to. Driven by curiosity, we ask ourselves a question: given the output, is it possible to determine whether $x \in \mathcal{D}_{training}$ or $\mathcal{D}_{test}$? In other words, can we recover the *membership* of a sample point x? Examples of succeeding in recovering such information happen in the case of overfitting of the training dataset. It is clear that if $\mathcal{D}_{training}$ contains sensitive information then we cannot let anybody determine its true membership, since it would constitute an individual's violation of privacy. As an example, imagine if we could determine whether a person participated in a medical study regarding the HIV. We would learn that a specific individual is affected by the virus.

Results and contributions MIA

In the thesis we will be playing the role of the attacker whose goal is to recover the membership of the individual x, and the defender, who tries to prevent the leakage of information from the dataset. These two roles can and will be performed separately. In chapter 2, we present an overview of possible attacks, depending on what the attacker knows. We will work in *black-box* attack, i.e we assume that an attacker only has access to the output and type of architecture of the model.

The goal of this first half of this thesis is to

1. design a metric based attack: We define a function $\mathcal{M}$ that classifies the input data as a member of $\mathcal{D}_{training}$ or a member of $\mathcal{D}_{test}$ by looking at the output of the model;

2. test the defence while training the model with a Differentially Private learning algorithm, such as Algorithm 1 and Algorithm 2.

Since the goal is to test the effectiveness of differential privacy, we keep the choice of the model simple: Linear Regression, Logistic Regression and Multiclass Logistic Regression. Implementing DP-SCO for neural network would require a complete re-design of the back propagation algorithm, but the whole theory clearly generalizes to it as well.

In order to achieve the two goals 1. and 2. stated above, we implemented the following strategies for both roles of attacker and defender.

Attack: For the sake of the experiment, suppose both $\mathcal{D}_{training}$ and $\mathcal{D}_{test}$ are known.

1. Metric loss attack: we measure the loss of $\mathcal{D} = \mathcal{D}_{training} \cup \mathcal{D}_{test}$ after training without DP and with DP for many values of $\varepsilon \in (0, 1]$. We order the losses from the lowest to the biggest. The first 50% of the losses should correspond to points in the dataset, because during the training they might have overfitted the model. Then, we count how many actually were correctly classified. We should expect to see something slightly above 50 % with a not private attack and around 50 % with DP. Indeed, we cannot be more less accurate of a random guess.
2. Entropy loss attack: Only for logistic regression types. Since the outputs are going to be probability vectors, the outputs corresponding to points in the training dataset should present a higher entropy. The rest is like the point 1.

Defence: The privatization is guaranteed by injecting noise during the training phase of the model, i.e in the training algorithm, that makes use of the of $\mathcal{D}_{training}$, which is supposed to be private. We will compare the performances of DP-SGD and DP-FW. The latter can only be applied if the minimization is constrained to some set. Instead of adding a regularized (unlike what we will happen in the synthetic dataset chapter), what we would normally do is to localize the minimum or design models for which we already know where the minimum should lie, and design ad-hoc polyhedron around it.

The results we would intuitively expect match the outcomes of the numerical simulations: adding more privacy — lowering the value of ε — prevents the attack from being successful. However, it destroys the utility, so a good trade-off needs to be found. The details of numerical results can be found in chapter 4.

Synthetic data generation

In the second part of this thesis, we switch point of view on how to protect data. We assume to be in the same scenario displayed in Figure 1. However, instead of releasing a noisy version of the data, we create a *fake* (synthetic) dataset to operate with. Such dataset is created preserving the same statistical properties as the original one, but guaranteeing individual's privacy. In other words, we create a new dataset that returns similar responses to the all queries asked by the data analyst. We refer to the construction of such dataset as *differentially private synthetic data generation.* The advantage of this method is that we do not need to inject noise after every query. The mechanism $\mathcal{A}$ is no longer required, thus we can work with the synthetic dataset as we would normally do. Creating such dataset boils down to solving an optimization problem, the reason why we will make use of DP-SCO again.

Formally, we consider the space of data $\mathcal{X}$, of dimension d, i.e $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$. We will be working in the general form of $\mathcal{X}$, with $\tilde{d} := |\mathcal{X}| \leq (\max_i |X_i|)^d$. A dataset is a collection of N individuals, hence $\mathcal{D} \in \mathcal{X}^N$. Moreover, we will think of a dataset as its empirical distribution over $\mathcal{X}$: $p_{\mathcal{D}}(x)$ is the frequency of x in $\mathcal{X}$, i.e number of times a point $x \in \mathcal{X}$ appears in $\mathcal{D}$, divided by N. Identifying a dataset as a probability distribution over $\mathcal{X}$, allows us to formally define a *statistical query*

Definition 4. Let $\mathcal{D}$ be a datset and $\mathcal{J} : \mathcal{X} \longrightarrow [0, 1]$ a predicate over $\mathcal{X}$. We define the statistical query q as a function *of* (not on) the dataset $\mathcal{D}$

$$q(\mathcal{D}) = \sum_{x \in \mathcal{X}} \mathcal{J}(x) p_{\mathcal{D}}(x) = \langle q, p_D \rangle \ ,$$

and $p_{\mathcal{D}}(x)$ is the normalized frequency of x in $\mathcal{D}$.

Generating a synthetic dataset translates into finding the probability distribution p for which the largest error on all queries is as small as possible:

$$\text{Find } p \text{ s.t } \ \max |\langle q, p_D \rangle - \langle q, p \rangle| \quad \text{is small} \ ,$$

i.e, p answers as accurately as possible to all queries (answering a query means providing a numerical value for $\langle q, p \rangle$).

Definition 5. The distribution representing the synthetic data is given implicitly by the solution of

$$\min_{p \text{ distib.}} \max_{q \text{ queries}} |\langle q, p_D - p \rangle| \quad (P) \ .$$

This is call min max formulation.

In the previous chapter we showed how to find a solution of this optimization problem, using the DP-SCO algorithms presented earlier. The goal of this second part is to obtain the synthetic data as the argmin of this formulation, by using DP-SCO. In the known literature, as far as we are aware, there are no attempts to generate the synthetic data using this approach. We will also add an entropy regularization term, which serves to keep the distribution p closer to the uniform U. Moreover, we will think of our $d_i \in \mathcal{D}$ as sampled from a distribution $\mathcal{P}$ in $\mathcal{X}$, thus we can generalize the definition of statistical query

Definition 6 (Population)**.** Let us denote $\Delta_{\tilde{d}} = \{p \in \mathbb{R}^{\tilde{d}} \mid \quad \sum_i^{\tilde{d}} p_i = 1 \quad p_i \geq 0\}$. A statistical query $\mathbf{q}$ is a function

$$\begin{aligned} \mathbf{q} : &\Delta_{\tilde{d}} \longrightarrow [0,1] \\ &\mathcal{P} \longrightarrow \langle \mathbf{q}, \mathcal{P} \rangle = \mathbb{E}_{Z \sim \mathcal{P}}[\mathcal{J}(Z)] \end{aligned}$$

Our method to obtain the synthetic dataset, using DP-SCO, goes as follow (for both p_D and $\mathcal{P}$):

1. **Regularization**. Firstly, since we look for distributions $p^\star$ that are *close* to the uniform distribution, we add the regularization term

$$\min_{p \in \Delta_{\tilde{d}}} \max_{q \in Q} |\langle q, p_D - p \rangle| + \alpha H(p) \quad (P_\alpha) \ .$$

2. **Solve the dual**. It turns out that the form of the dual is more convenient to work with. Frank Wolfe is perfectly suitable for this case.

3. **Go back to primal**. We take the value computed by Frank Wolfe $q_T \cong q_\alpha^\star$ approximating the exact solution of (D_α), and we plug in (P_α). The solution $p_\alpha^\star$ of (P_α), is taken as approximation of (P)

4. **Quantify the error**. We will give an exact upper bound on how much $p_\alpha^\star$ differs from $p^\star$, exact solution of the original problem (P).

We can visualize the previous steps in the scheme below

$$\underbrace{(D_\alpha)}_{\text{Frank Wolfe output: } q_T \cong q_\alpha^\star} \rightsquigarrow \underbrace{(P_\alpha)}_{\text{input: } q_T \text{ output: } q_\alpha^\star} \rightsquigarrow \underbrace{(P)}_{\text{input: } q_\alpha^\star \text{ output: } \mathbf{Gap}_{(P)}(p_\alpha^\star)} \ .$$

A section will be entirely dedicated on how to opportunely tune the regularization parameter α. The choice of α minimizes the primal gap, i.e we require α to be such that $\mathbf{Gap}_{(P)}(p_\alpha^\star)$ is as small as possible. Such chosen α will depend on the dataset, the privacy parameters and the number of points.

Results and contributions SD

Not only we designed a method that produces a $p^\star$ whose error $\max_q \langle q, p_{\mathcal{D}} - p^\star \rangle$ matches the best known upper bound, but we also applied the same method on the population mean, the general distribution of the data. As far as we are aware, there is not literature on this last case.

Empirical	Population
$\mathcal{O}\left(\frac{d^{1/5}\log^{2/5}\lvert Q\rvert \log^{1/5}(1/\delta)}{(\varepsilon N)^{2/5}}\right)$	$\mathcal{O}\left(\frac{d^{1/5}\log^{2/5}\lvert Q\rvert \log^{1/5}(1/\delta)}{(\varepsilon N)^{2/5}} + \frac{d^{1/2}}{\sqrt{N}}\right)$

These upper bounds can be reasonably improved. In fact, in both cases we had to upper bound the Frank Wolfe gap, instead of the empirical risk. In doing so, we did not use the convexity of the loss function, which normally leads to stricter upper bounds, but we only used the smoothness of the gradient. The reason why we decided not to upper bound the risk is due to the regularization term. Indeed, the upper bound of $\mathbf{Gap}_{(P)}(p_\alpha^\star)$ is expressed in terms of the Frank Wolfe gap. Trying to express it in term of the empirical risk — that we know how to estimate better — yields an upper bound too large to control. This will be explained in chapter 5.

Outline of the thesis

The thesis begins with chapter 1, where we laid out a general introduction about differential privacy, presenting the most known results and general tools. chapter 2 introduces the theory of the membership inference attacks, giving exact details of what kind of attack and setting we are going to implement. chapter 3 is the hearth of the thesis. We present the general setting of Stochastic Convex Optimization with the Frank Wolfe algorithm, and how to make it private, as well as the proof of the convergence — fundamental for the modification of the algorithm in chapter 5. In chapter 4, the DP-FW and DP-SGD are implemented and compared on various statistical models. We analyzed the privacy guarantees and observed how DP does protect privacy, using both algorithms. Finally, in chapter 5, the general setting of synthetic data generation is explained, then we dive into the adaptation of Frank Wolfe for the corresponding optimization problem, re-adapting the results of chapter 3. The second half on the chapter shows how to tune the regularization parameter and how to obtain an upper bound on the total error we commit in answering all the queries. Lastly, we present the generalization of the method for the case of the excess population risk.

Chapter 1

Differential Privacy

The purpose of this first chapter is to recall briefly the notion differential privacy and some basic properties. What does it mean to rigorously ensure privacy? In particular, more emphasis is put on the Laplacian Mechanism and its properties, since it is going to be the main differentially private tool used throughout 3. The chapter will begin with an example meant to laid out the intuition behind the notion of privacy in a mathematical sense. Most of the theory presented in this chapter is covered in (DR13).

1.1 Randomized Response

Suppose that we are given n samples $X_j \in \{0, 1\}$ from a survey where X_j takes the value 1 if the $i-th$ subject satisfies a certain property P and 0 otherwise. In this scenario, the collection of these values form a data set $\{X_1, \cdots, X_n\}$ that could be used by an analyst to do some statistics. What if $X_j = 1$ contains information that we would not like to share, i.e if it is private, but we are forced to send to the analyst? Maybe we could send to the analyst information that is not exactly true, but not so compromised to make the statistics meaningless either. For instance, the $i-th$ subject could send the variable Y_i defined as

$$Y_i = \begin{cases} X_i & \text{w.p } \frac{1}{2} + \gamma \\ 1 - X_i & \text{w.p } \frac{1}{2} - \gamma \end{cases}$$

so that $Y_i \sim B\left(0, \frac{1}{2} + \gamma\right)$ and for $\gamma \in [0, 1/2]$. The information encoded in the variable Y_i has the following meaning: we can either send the correct information X_i with probability controlled by γ, or the false one with the complementary probability. To get a deeper insight, if $\gamma = 1/2$ the people would be transmitting the the true answers with probability 1, i.e they would be outputting the whole data set. This a totally *non private* scenario. On the other hand $\gamma = 0$ means that with equal probability the people are likely to send the correct or false answer to the analyst, i.e maximum privacy. The inner values of γ guarantee intermediate levels of privacy.

Through this mathematical trick, somehow we compromised the answers that have to be sent to the analyst, who will not be able to say anything about a particular person i, since the answer maybe be not the real one. In a sense, we *protected* the individual's privacy. What about the statistics that still needs to be done? Are the data too compromised now?

The short answer to this question is that it depends on the statistics we are interested in. Let us assume that the original goal was to estimate the mean of the data set, so that

we wanted to compute

$$p = \frac{1}{n}\sum_{k=1}^{n} X_k \ .$$

Now, the analyst does not see the X_i, but he sees the Y_i. He observes that

$$\mathbb{E}[Y_i] = X_i\left(\frac{1}{2}+\gamma\right) + (1-X_i)\left(\frac{1}{2}-\gamma\right) = 2\gamma X_i - \gamma + \frac{1}{2}$$

hence,

$$\mathbb{E}\left[\frac{1}{2\gamma}\left(Y_i+\gamma-\frac{1}{2}\right)\right] = X_i.$$

Using this observation, we found out that the quantity

$$\hat{p} = \frac{1}{2\gamma n}\sum_{i=1}^{n}\left(Y_i+\gamma-\frac{1}{2}\right)$$

is an unbiased estimator of p. In a way, we can still estimate the mean without using the real information about the data set.

This is the spirit of the differential privacy: *we learned something about the population, without learning anything specific about a single individual.*

This method just exposed is called *Randomized Response*, and it is out first example of differential privacy.

How do we formalize this idea? In the next section we laid out the formal definition of privacy. Before going into the details of the definition of privacy, recall that we think of a *randomized algorithm*, map or *mechanism* , as a machine M that computes $M = M(x, l)$, where x is the problem input and l is a random variable that might or might not depend on the input x. Also, we have the following definition,

Definition 7. We say that two data sets $\mathcal{D}_1, \mathcal{D}_2$ are neighboring if they only differ in one element. In such case with write $\mathcal{D}_1 \approx \mathcal{D}_2$.

Definition 8. Let $\mathcal{X}^n$ be the set containing data sets of n and $\mathcal{M} : \mathcal{X}^n \longrightarrow \mathcal{Y}$ be a randomized map defined on this set. We say that M is ε differentially private if for every couple of neighbouring data sets $\mathcal{D}_1 \sim D_2$ and for every event $T \subset \mathcal{Y}$,

$$\mathbb{P}(M(\mathcal{D}_1) \in T) \leq e^{\varepsilon}\mathbb{P}(M(\mathcal{D}_2) \in T)$$

What is the meaning of this definition?

- $\varepsilon = 0$ is equivalent to absolute privacy. It can be derived directly from the definition of Differential Privacy. Briefly, $\varepsilon = 0$ is equivalent to $\mathbb{P}(M(\mathcal{D}_1) \in T) = \mathbb{P}(M(\mathcal{D}_2) \in T)$, this leads to the algorithm M being independent of the data and thus protects privacy perfectly, at the expense of complete loss in accuracy.

- Suppose M represents the learning algorithm for a statistical model, like a classifier. Decreasing ε leads to a decreasing in the accuracy of the target model. If an algorithm M is 0-DP, namely it protects privacy well, then it has very performances in terms of target accuracy, thus it would be useless.

Observation. From the definition, instead of imposing bounded ratio between the two probabilities $\mathbb{P}(M(\mathcal{D}_1) = T)$ and $\mathbb{P}(M(\mathcal{D}_2) = T)$, we can also bound their respectibe PDFs. In fact observe that if

$$\mathbb{P}(M(\mathcal{D}) \in T) = \int_T p_{\mathcal{D}}(z)dz$$

it follows that

$$p_{\mathcal{D}_1}(z) \leq \exp(\varepsilon)p_{\mathcal{D}_2}(z) \Longrightarrow \int_T p_{\mathcal{D}_1}(z)dz \leq \exp(\varepsilon)\int_T p_{\mathcal{D}_2}(z)dz.$$

Properties of Differential Privacy:

Theorem 9 (post-processing)**.** *Let $\mathcal{M}, \mathcal{F}$ be randomized algorithms such that $\mathcal{M} : \mathcal{X}^n \longrightarrow \mathcal{Y}$ is ε-DP and $\mathcal{F} : \mathcal{Y} \longrightarrow \mathcal{Z}$. Then, $\mathcal{F} \circ \mathcal{M}$ is $\varepsilon-DP$.*

Proof. Let $Z \in \mathcal{Z}$ and $X \in \mathcal{X}^n$.

$$\mathbb{P}(\mathcal{F}\circ M(X) \in Z) = \mathbb{P}(M(X) \in \mathcal{F}^{-1}(Z)) \leq \exp(\varepsilon)\mathbb{P}(M(Y) \in F^{-1}(Z))) = \mathbb{P}(\mathcal{F}\circ M(Y) \in Z).$$

□

Observe that $\mathcal{F}$ does not have to be $\varepsilon - DP$. As a results, once a mechanism $\mathcal{M}$ has been made ε-DP, no matter how many more operations we perform on $\mathcal{M}(\mathcal{D})$, the final product $\mathcal{F} \circ \mathcal{M}$ keeps protecting privacy, being ε-DP.

1.2 Laplace Mechanism

Suppose we are given a data set X we need for doing some statistics. The collection, or we would rather say the composition, of all the operations performed on X will be results in a deterministic function $f : \mathcal{X}^n \longrightarrow \mathcal{Y}$. Such functions are known as *statistics* of the data set $\mathcal{D}$. Since f is deterministic, it is interesting to think about way to turn $f(\mathcal{D})$ into a DP mechanism. Also, what is the good amount *privacy* needed not to compromise significantly the effectiveness of outcome of $f(\mathcal{D})$? In this paragraph we will laid out a methodology which privatizes deterministic outcomes $f(\mathcal{D})$, and it is called Laplacian Mechanism. We shall use it throughout the whole chapter 3. We need some definitions first.

Definition 10 (Laplace distribution)**.** The Laplace Probability Distribution Function (p.d.f) $Lap(\mu, b)$ is defined as

$$d(x|\mu, b) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right).$$

Definition 11 (Sensitivity)**.** Let $f : \mathcal{X}^n \longrightarrow \mathbb{R}^k$. The sensitivity of f is defined as

$$\Delta = \sup_{X\sim Y} ||f(\mathcal{D}_1) - f(\mathcal{D}_2)||_{\ell^1}$$

where $\mathcal{D}_1 \approx \mathcal{D}_2$.

The sensitivity of f expresses how much f mutates when the data set changes in just one individual.

Definition 12 (Laplace Mechanism)**.** Let $\mathcal{D}$ be a data set of points and let $f : \mathcal{X}^n \longrightarrow \mathbb{R}^k$ be a statistics of $\mathcal{D}$. We call *Laplace Mechanism* the random map

$$M(\mathcal{D}) = f(\mathcal{D}) + [Y_1, \cdots, Y_k]$$

where Y_j are i.i.d Laplacian distributed as $Y_i \sim Lap\left(\frac{\Delta}{\varepsilon}\right)$.

Theorem 13. *The Laplace Mechanism is $\varepsilon - DP$.*

Proof. Let $\mathcal{D}_1 \sim \mathcal{D}_2$ be neighbouring data sets and $p_{\mathcal{D}_1}(z)$, $p_{\mathcal{D}_2}(z)$ the pdf of $M(\mathcal{D}_1)$ and $M(\mathcal{D}_2)$ respectively. Further, since all the elements of $\mathcal{D} - 1, \mathcal{D}_2$ are the same but one, wlog we indicate by j the index for which they differ.

$$\begin{aligned}\frac{p_{\mathcal{D}_1}(z)}{p_{\mathcal{D}_2}(z)} =& \frac{\Pi_{i=1}^n \exp\left(-\frac{\varepsilon|z_i-(f(\mathcal{D}_1))_i|}{\Delta}\right)}{\Pi_{i=1}^n \exp\left(-\frac{\varepsilon|z_i-(f(\mathcal{D}_2))_i|}{\Delta}\right)} \underbrace{=}_{\mathcal{D}_1 \sim \mathcal{D}_2} \exp\left(\frac{\varepsilon}{\Delta}(|z_j - (f(\mathcal{D}_2))_j| - |z_j - (f(\mathcal{D}_1))_j|)\right) \\ &\leq \exp\left(\frac{\varepsilon}{\Delta}|(f(\mathcal{D}_1))_j - (f(\mathcal{D}_2))_j|\right) \leq \exp\left(\frac{\varepsilon}{\Delta}\Delta\right) = \exp(\varepsilon).\end{aligned}$$

□

1.3 (ε, δ) Differential Privacy

The notion of $\varepsilon-$DP can be rephrase using a more convenient tool, called *Privacy Loss Random Variable*.

Definition 14 (Privacy Loss Random variable)**.** Let Y, Z be two random variable defined on the same support. We define the *privacy loss random variable* as

$$\mathcal{L}_{Y||Z}(t) = \log\left(\frac{d\mathbb{P}(Y=t)}{d\mathbb{P}(Z=t)}\right) \quad t \sim Y,$$

where $d\mathbb{P}(\cdot)$ is used to the denote the p.d.f of Y and Z.

Remark: It is a random variable whose distribution is obtained drawing and outputting a value from the distribution of Y. In simpler words, when Y takes a value $t \in \mathbb{R}$ with some probability $\Longrightarrow \mathcal{L}_{Y||Z}$ takes value $\log\left(\frac{d\mathbb{P}(Y=t)}{d\mathbb{P}(Z=t)}\right)$ with the same probability.

A simple algebraic manipulation shows that the following equivalence holds

$$\mathcal{M} \text{ is } \varepsilon - \text{DP} \iff \mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)} \leq \varepsilon.$$

There is a straight geometric representation of $\mathcal{L}$ that helps to understand the need for this variable. Suppose we are about to plot the graph of $\mathcal{L}$ in a plan. In the horizontal axis we put the outcomes of t of Y and on the vertical axis the value $\mathcal{L}_{Y||Z}(t)$. Essentially, $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}$ is being plotted in the plot $\mathcal{L}\hat{0}Y$ plan - meaning that on the vertical axis we plot $\mathcal{L}$ on the horizontal axis there is the support of the real valued random variable Y.

Remark: If still unclear: Given $t \in \text{Supp}(Y)$ then $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t)$ is a function. There is not randomness anymore, hence we plot it in a graph $(t, \mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t))$. To make things even clearer, if we had any real valued random variable X, representing (X, X^2) would still look like a parabola, of course not entirely defined on $\mathbb{R}$ but only on the support of X, but still a parabola, independently from the distribution of X.

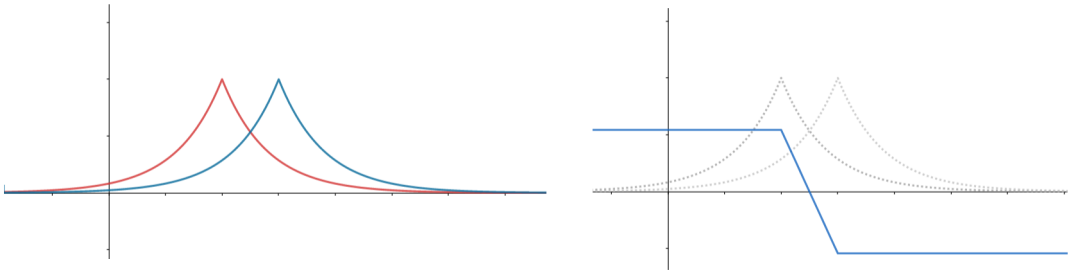


Figure 1.1: On the left two laplacian distribution with the same variance but different mean. On the right, in blue it is represented the graph of the Privacy Loss random variable for the Laplacian mechanism.

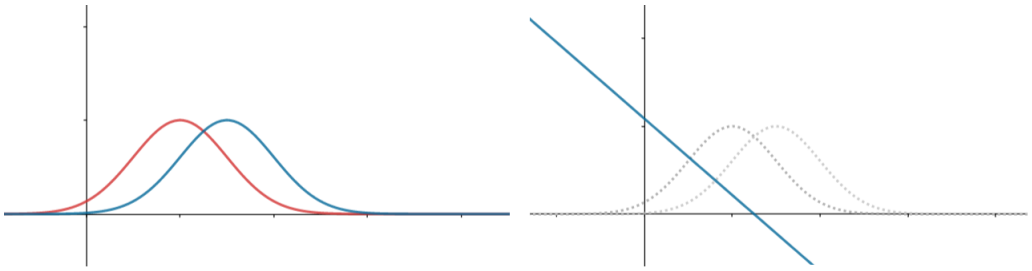


Figure 1.2: Same plot like in figure 1.1, but with Gaussian distribution. The Privacy Loss random variable will be line in the plan, hence it will blow up to infinity when $t \to \infty$.

In this plan, $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}$ is completely bounded from above by this threshold ε. For instance, this is true for the Laplacian mechanism, whose graph is represented in 1.1

and this correspond to geometrical meaning of ε-DP privacy.

Suppose that we want to preserve the methodology of the Laplace Mechanism, i.e adding some noise to a deterministic outcome of the data set, but using another distribution. Unfortunately, it might happen that the graph of $\mathcal{L}$ is no longer bounded with the new distribution. In fact, this is the case of the Gaussian Mechanism (that has not been covered yet) for which the resulting $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}$ will be unbounded. Then, what can we say in terms of privacy since the mechanism will not be ε-DP? For this purpose, it is needed a relaxation of the definition of ε DP.

Definition 15. A stochastic algorithm $\mathcal{M} : \mathcal{X}^n \longrightarrow \mathcal{Y}$ is (ε, δ) differentially private if for every couple of neighbouring data sets $\mathcal{D}_1, \mathcal{D}_2$ and for every event $T \subset \mathcal{Y}$,

$$\mathbb{P}(M(\mathcal{D}_1) \in T) \leq e^{\varepsilon}\mathbb{P}(M(\mathcal{D}_2) \in T) + \delta$$

What is the role of δ? As it was said, for the Gaussian mechanism the graph of $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t)$ is unbounded on $t \in \mathbb{R}$ (figure 1.2), thus it is impossible to conclude that the mechanism is $\varepsilon-$DP. The same behaviour for $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t)$ is preserved if we re-scale the horizontal axis into $F_Y(t)$ with $Y := \mathcal{M}(\mathcal{D}_1)$. This means that we are plotting into the $(F_Y(t), \mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t))$ plane, figure 1.3 . Then, fixing the level $y = e^{\varepsilon}$, the set of all points $x = F_Y(t)$ in the horizontal axis such that $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t) \leq e^{\varepsilon}$ will be finite, due to the fact that $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t)$ is monotone. Since the the horizontal axis is scaled, the measure of all the points for which $\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t)$ is above e^{ϵ} has some measure that we call $\delta = F_Y(\hat{t})$, for a certain $\hat{t}$. This shows that

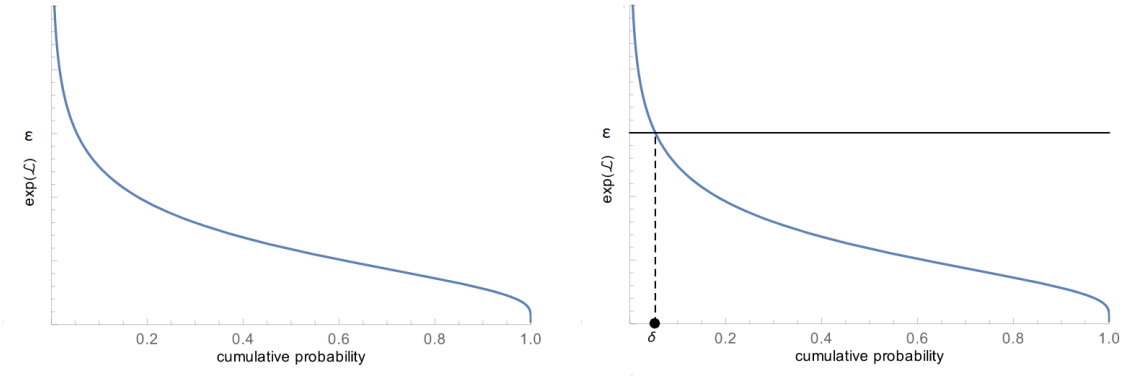


Figure 1.3: graph of figure 1.2 with the scaled horizontal axis. On the right the geometric meaning of δ and ε - in the graph it should be e^{ε} instead of ε.

$$\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t) \geq \varepsilon \iff Y \leq \hat{t}$$

and, taking the probabilities on both sides

$$\mathbb{P}(\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t) \geq \varepsilon) = \mathbb{P}(Y \leq \hat{t}) = \delta.$$

Translated into words: (ε, δ)-DP is equivalent to saying that the absolute value of the privacy loss random variable is bounded by ε with probability $1 - \delta$:

$$\mathbb{P}(\mathcal{L}_{\mathcal{M}(\mathcal{D}_1)||\mathcal{M}(\mathcal{D}_2)}(t) \leq \varepsilon) = 1 - \delta.$$

Report Noisy Max Mechanism

Suppose we have $a_j : \mathcal{X}^n \longrightarrow \mathbb{R}$ functions of a data set $\mathcal{D}$, all with sensitivity $\Delta(a_j) \leq L$ for some constant L. The mechanism RNM defined as

$$RNM(\mathcal{D}) = \underset{i=1,\cdots,n}{\arg\max} \left[a_i(\mathcal{D}) + Lap\left(\frac{nL}{\varepsilon}\right) \right] \quad \text{Lap i.i.d}$$

is called Report Noisy Max Mechanism. Essentially, taking n statistics of the data set $\mathcal{D}$ the mechanism outputs the max and the index for which it reaches the maximum.

Proposition 16. ***RNM*** *is* $\varepsilon - DP$

Proof. The result follows from the composition theorem and the Laplace Mechanism. Formally,

$$RNM(\mathcal{D}) = \underset{i=1,\cdots,n}{\arg\max} \underbrace{\left(\begin{bmatrix} a_1(\mathcal{D}) \\ \vdots \\ a_n(\mathcal{D}) \end{bmatrix} + \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} \right)}_{\text{Lap Mechanism}} \quad Y_i \sim \text{Lap}(nL/\varepsilon).$$

RNM($\mathcal{D}$) is the composition of the deterministic map $\pi :=\text{argmax} : \mathbb{R}^n \to [n]$ and the Laplace mechanism $\mathcal{M}_L(\mathcal{D})$. The composition theorem makes us conclude that $\pi \circ \mathcal{M}_L(\mathcal{D})$ is ε private when the Laplacian random variables are distributed as $\text{Lap}\left(\frac{\Delta_a}{\varepsilon}\right)$. To conclude:

$$\Delta = \sup_{\mathcal{D}\sim\mathcal{D}'} ||a(\mathcal{D}) - a(\mathcal{D}')||_1 = \sup_{\mathcal{D}\sim\mathcal{D}'} \sum_{i=1}^{n} |a_i(\mathcal{D}) - a_i(\mathcal{D}')|$$
$$\leq \sum_{i=1}^{n} \sup_{\mathcal{D}\sim\mathcal{D}'} |a_i(\mathcal{D}) - a_i(\mathcal{D}')| \leq \sum_{i=1}^{n} \Delta(a_i) \leq nL.$$

□

Observation. This proof is rather straightforward. However, it misses one important point. We are introducing a noise that is n times greater that the upper bound on each $a_i's$ sensitivity.

Thinking of RNM as a composition means that we first output the whole vector a_i and then we select the maximum. Instead, thinking of it as an whole, i.e only revealing the argmax, leads to a better result in term of noise introduced. This is the content of the following:

Theorem 17. *The RNM with Lap$\left(\frac{L}{N}\right)$ is $\varepsilon - DP$.*

For the proof see (DR13). Note that they only prove it for counting queries, i.e each a_j is a function that counts how many times a certain property is satisfied for the dataset $\mathcal{D}$.

some properties of pure differential privacy are preserved over to the relaxed DP.

Lemma 18. *Let $M : \mathcal{X}^n \longrightarrow \mathcal{Y}$ - (ε, δ)-DP and $F : \mathcal{Y} \longrightarrow \mathcal{Z}$ any random map. Then the composition $F \circ M$ is (ε, δ) -DP*

The following is an important result the we will need later in Chapter 3.

Theorem 19 (Advanced Composition Theorem(DRV10))**.** *For all $\varepsilon, \delta, \delta' > 0$, let $M = [M_1, \cdots, M_T]$ be a sequence of $(\varepsilon, \delta) - DP$ algorithms where M_i are sequentially and adaptively chosen. Then, the whole chain M is $(\hat{\varepsilon}, \hat{\delta})$ where*

$$\begin{cases} \hat{\varepsilon} = \varepsilon\sqrt{2T\log(1/\delta')} + T\varepsilon\frac{e^\varepsilon - 1}{e^\varepsilon + 1} \\ \hat{\delta} = T\delta + \delta' \end{cases}.$$

A specialized version of this theorem holds

Theorem 20 (Advanced Composition Theorem)**.** *For all $\varepsilon > 0$ and $1 > \delta > 0$ with $\log(1/\delta) \geq \varepsilon^2 T$, let $M = [M_1, \cdots, M_T]$ be a sequence of $(\varepsilon, 0) - DP$ algorithms where M_i are sequentially and adaptively chosen. Then, the whole chain M is $(\hat{\varepsilon}, \delta)$ where*

$$\hat{\varepsilon} = 4\varepsilon\sqrt{2T\log(1/\delta)}.$$

Another important results is the following:

Lemma 21 ((BKN10))**.** *Suppose an algorithm is $\varepsilon < 1$ differentially private. If it is executed on a uniformly random subset of the data set of size γn then it will be $2\gamma\varepsilon$-DP.*

Basically, this means that subsampling amplifies privacy.

Chapter 2

Membership Inference Attacks

In the previous chapter the definition of DP has been introduced in its pure and approximate forms. Also, we claimed that the purpose of DP is to avoid leakage of information from a data set $\mathcal{D}$, but a rigouros explanation of what this precisely means is still missing. We shall begin this chapter proving an example of how an *attacker* does not learn much about a data set when the information he receives is the outcome of a DP algorithm, proving that DP protects the data set $\mathcal{D}$. The most known attack in literature is the Membership Inference Attack: the attacker tries to guess whether a $x \in \mathcal{D}$ was in the data set or not. Afterwards, the most known methodologies for constructing such attacks against ML models shall be explained.

So far, the idea behind DP has been the following: if a mechanism behaves similarly for $\mathcal{D}_1 \sim \mathcal{D}_2$ then an *attacker* cannot tell what whether $\mathcal{D}_1$ or $\mathcal{D}_2$ was used. More specifically, we assume to have the following situation

- There is an **attacker or adversary** who wants to retrieve information about the data set $\mathcal{D}$. In order to achieve his goal, the attacker asks queries to the curator.

- There is a **data set curator** that is the only one who has access to the data set $\mathcal{D}$. The curator releases a model $M(\mathcal{D})$ as response to the queries, which is thought as a mechanism of the data set. Moreover, M is DP.

In particular, the attacker knows $\mathcal{D} - \{x\}$ and wants to understand if x is the missing element in $\mathcal{D}$ or not. Somehow he concludes that either $\mathcal{D} = \mathcal{D}_1$ or $\mathcal{D} = D_2$, with $\mathcal{D}_1$ and $\mathcal{D}_2$ that only differ in one element, exactly x. The prior probability of x is known and it is $\mathbb{P}(x \in \mathcal{D})$. He cannot distinguish between $\mathcal{D}_1$ and $\mathcal{D}_2$, therefore he starts querying the curator, who replays outputting the $M(\mathcal{D}) - \varepsilon$DP. How much does the attacker learn from this information? In other words, how does the prior probability change now that he can estimate $\mathbb{P}(x \in \mathcal{D}|M(\mathcal{D}))$?

Using Bayes's

$$\mathbb{P}(x \in \mathcal{D}|M(\mathcal{D})) = \frac{\mathbb{P}(M(\mathcal{D})|x \in \mathcal{D})\mathbb{P}(x \in \mathcal{D})}{\mathbb{P}(M(\mathcal{D})|x \in \mathcal{D})\mathbb{P}(x \in \mathcal{D}) + \mathbb{P}(M(\mathcal{D})|x \notin \mathcal{D})\mathbb{P}(x \notin \mathcal{D})}$$

Since $x \in \mathcal{D}_1$ and $x \notin \mathcal{D}_2$ then $\mathbb{P}(M(\mathcal{D})|x \notin \mathcal{D}) = \mathbb{P}(M(D_2))$, Further, M is DP private, hence $\mathbb{P}(M(D_2)) \leq e^{\varepsilon}\mathbb{P}(M(D_1)) = e^{\varepsilon}\mathbb{P}(M(\mathcal{D})|x \in \mathcal{D})$.

Using this inequality in the expression above we get

$$\begin{aligned}\mathbb{P}(x \in \mathcal{D}|M(\mathcal{D})) =& \frac{\mathbb{P}(M(\mathcal{D})|x \in \mathcal{D})\mathbb{P}(x \in \mathcal{D})}{\mathbb{P}(M(\mathcal{D})|x \in \mathcal{D})\mathbb{P}(x \in \mathcal{D}) + \mathbb{P}(M(\mathcal{D})|x \notin \mathcal{D})\mathbb{P}(x \notin \mathcal{D})} \\ \geq& \frac{\mathbb{P}(M(\mathcal{D})|x \in \mathcal{D})\mathbb{P}(x \in \mathcal{D})}{\mathbb{P}(M(\mathcal{D})|x \in \mathcal{D})\mathbb{P}(x \in \mathcal{D}) + e^{\varepsilon}\mathbb{P}(M(\mathcal{D})|x \in \mathcal{D})\mathbb{P}(x \notin \mathcal{D})} \\ =& \frac{\mathbb{P}(x \in \mathcal{D})}{\mathbb{P}(x \in \mathcal{D}) + e^{\varepsilon}\mathbb{P}(x \notin \mathcal{D})} = \frac{\mathbb{P}(x \in \mathcal{D})}{\mathbb{P}(x \in \mathcal{D})(1 - e^{\varepsilon}) + e^{\varepsilon}}\end{aligned}$$

and, since $1 - e^{\varepsilon} < 0$ for all $\varepsilon > 0$, we have

$$\mathbb{P}(x \in \mathcal{D}|M(\mathcal{D})) \geq \mathbb{P}(x \in \mathcal{D})e^{-\varepsilon}.$$

Also, using the symmetry of the DP definition,

$$\mathbb{P}(x \in \mathcal{D}|M(\mathcal{D})) \leq \mathbb{P}(x \in \mathcal{D})e^{\varepsilon}.$$

In conclusion, the total information earn boils down to an estimation on the posterior probability bounded as

$$\mathbb{P}(x \in \mathcal{D})e^{-\varepsilon} \leq \mathbb{P}(x \in \mathcal{D}|M(\mathcal{D})) \leq \mathbb{P}(x \in \mathcal{D})e^{\varepsilon}.$$

This inequality gets stricter as ε gets closer to zero. In other words, the lower ε is the higher is the level of privacy insured.

The example just described shows the full spirit of DP. An attacker cannot really learn much more from a query $M(\mathcal{D})$ when DP is implemented. However, this does not mean that he will simply give up. On the contrary, the attacks will be designed to gain as much as possible, even though theoretical bounds restrict the learning limits, like the inequality above.

2.1 Attacks in Machine Learning models

As we know, the goal of defining ML models or algorithms is to define function that cannot be written analytically. Typically, we want to learn a set of parameters * which defines a function or *model* $f(x;\theta)$. Obtaining $\theta^\star$ is a process that involves the training of the model f over a given data set $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ with a learning algorithm $\mathcal{A}$. The training is concluded when $\theta^\star$ which minimizes the empirical risk on the training set is obtained through the algorithm $\mathcal{A}$. The model $f(x;\theta^\star)$ is then taken as a new prediction function on the unseen data.

A Membership Inference Attack (MIA) aims to determine whether a given data instance is part of the target model's training data set or not. The strategy is based on the attacker's knowledge, for which there are many scenarios.

- **Data knowledge:** the adversary knows the distribution of $\mathcal{D}_{\text{train}}$. He can get a *copy* $\mathcal{D}'$ distributed as $\mathcal{D}$. Furthermore, the new set of data can be disjoint (here the distributions are meant to be continuous)

- **Training knowledge:** The adversary knows the learning algorithm $\mathcal{A}$, including the procedure to solve the optimization problem, number of training steps and etc.

- **Model knowledge:** The adversary knows the architecture or "model" of the NN, $f(x;\theta)$ - topological architecture and model parameters.

- **Output knowledge:** The adversary knows the final output of the model, for instance the prediction vector of a forward neural network, $p(y|x)$.

It is always assumed that the adversary knows at least the output vector. Based on what he/she knows, the attack is further divided in *black-box* (only output knowledge) and *white-box* (full access to the items above) attacks.

The upcoming sections are based from (HSS+21).

2.2 MIA on Neural Networks

Remember that the goal of the membership attacks is to determine whether a data point is a member of a training data set or not. How do we recognize if a point belongs to the data set? For the moment, let us assume our model to be a simple forward neural network that does classification. How do we attack the model? The idea behind the attack is that, more than often the NNs are trained multiple times over the same $\mathcal{D}_{\text{train}}$. As a result, a point in $\mathcal{D}_{\text{train}}$ is more likely to have an higher confidence score. To make it more concrete, let us suppose that we have a NN that classifies the data input into 5 classes which outputs a vector $p(y|\theta^\star)$. If we are given an $x \in \mathcal{D}_{\text{train}}$ as input, the confidence score might look like $[0.9, 0.05, 0.04, 0.01, 0]$ i.e assigning 90% of confidence to the first class. If we give as input another $x' \notin \mathcal{D}_{\text{train}}$ whose label is *class 1*, the confidence score might look like $[0.6, 0.3, 0.1, 0, 0]$. The point will be assigned the correct class (and this is the goal of the classifier after all) but the confidence score is significantly lower than before.

The attacks are built upon the above example. We shall try to capture this behaviour. There are two types of attacks methods. One is **NN based attacks** and the other one is **NN metric based attacks**.

Even if these attacks are defined for NNs, we can partially recycle the same approaches for other ML models.

2.2.1 Neural Network based attacks

From now on, the Neural Network is referred to *target model* . What comes before the attack is phase in which the adversary prepares the data set. It is assumed that the he only has access to the distribution on the data set and the output of the target model. Also, he can use the model, i.e given a data set $\mathcal{D}$, the adversary can observe the output $f(\mathcal{D}, \theta^\star) = P$.

The first step is to create a data set of points that will be used to train a binary classifier (the adversarial NN) - which recognizes whether a point is a member or not. To get this data sets, the procedure is the following

1) Let us drawn $\mathcal{D}_1, \cdots, \mathcal{D}_n$ from the distribution of $\mathcal{D}_{\text{train}}$ such that they are disjoint from $\mathcal{D}_{\text{train}}$. These are called shadow training sets.

2) Train the target model on each data set $\mathcal{D}_i$ and get the ready functions $f(\mathcal{D}_i, \theta^\star) = P_i$.

3) Furthermore, consider other *testing sets* $T_1, \cdots, T_k$ disjoint from $\mathcal{D}_1, \cdots, \mathcal{D}_n$

4) Feed the target model with the previous data sets, getting :

$$P_j^m = f(\mathcal{D}_j, \theta^\star) \quad \text{and} \quad P_j^{nm} = f(T_j, \theta^\star) \quad \text{for } j = 1, \cdots, n$$

where nm and m stand for *non member* and *member* respectively.

5) $P_j^m \times \{1\} \bigcup P_j^{nm} \times \{0\}$ form the new data set of points that we are going to use to train the adversarial Neural Network. The labels 1 and 0 represent being a member or not.

The adversarial NN, which has the architecture of a binary classifies (it could be a simple forward model) is trained on $\{P_j^m \times \{1\} \bigcup P_j^{nm} \times \{0\}\}$.

This works in both white and black box setting. Of course there are small differences. The adversarial NN shall be denoted as $g(\cdot, \theta)$ and takes P_j as input and it outputs the confidence score for 0 and 1.

2.2.2 NN based attacks in black-box setting

As a reminder, in this setting the adversary has only access to the output of the target network and he can only create the data set to train his own NN following the procedure above.

Let $\mathcal{L}$ be the loss function of the binary classifier. and I the indicator function (it is a vector whose entries are all zero except for one).

The *learning phase* boils down to solving the optimization problem

$$\arg\min_\theta \frac{1}{2n} \sum_{j=1}^{2n} \mathcal{L}(I(x_j \in P^m), g(p(y|x_j), \theta))$$

which returns the optimal parameter $\theta = \theta^\star$ and therefore the adversarial classifier $g(\cdot, \theta^\star)$.

2.2.3 NN based attacks in white-box setting

There is only one significant different respect to the black-box scenario. The adversary knows more information since he has full access to the intermediate computations of the target model. We are going to use this further information. Let $h(x; \theta_s)$ the intermediate computations at the layer s (it is assumed that the NN has the structure of a forward NN), $\frac{\partial \mathcal{L}}{\partial \theta_i}$ the partial derivative in the backward step at layer i and $p(y|x_j)$ the output prediction vector.

Let v be the row vector containing

$$v = \left[\frac{\partial \mathcal{L}}{\partial \theta_1}, h(x, \theta_1), \cdots, \frac{\partial \mathcal{L}}{\partial \theta_n}, h(x, \theta_n), p(y|x) \right],$$

then, the data set is amplified as $(v, 0)$ for non members and with $(v, 1)$ for members. The optimization problem that we need to solve is therefore

$$\arg\min_\theta \frac{1}{2n} \sum_{j=1}^{2n} \mathcal{L}(I(x_j \in P^m), g(v_j, \theta)).$$

We can solve both optimization using SGD. However, we will see later on that there are finer techniques to these purposes.

2.3 Metric Based Attacks

Unlike NN based attacks, the Metric based attacks directly define a function $\mathcal{M}$ which classifies the input data as member (1) or not member (0) by measuring some "metrics". Based on what we measure, there are four types of attacks, all based in black-box mode.

- *Prediction correctness based attack*: an input data instance x is a member if it is correctly predicted by the target model, otherwise it is not a member. The intuition is that the target model should make correct predictions on its own training data set. Therefore, the decision metric $\mathcal{M}$ is

$$\mathcal{M}(p(y|x); y) = I(\underbrace{\arg\max p(y|x)}_{\text{class}} = \underbrace{y}_{\text{label}}).$$

- *Prediction loss based attack*: Once the target network is trained, the loss of computed on one of its training data should be quite smaller compared to the loss on any other input. Using this intuition, we define a metric $\mathcal{M}$ which assigns a input data as a member if the input has "low loss". This "low loss " is controlled through a threshold $\tau \geq 0$. Therefore,

$$\mathcal{M}(p(y|x), y) = I(\mathcal{L}(p(y|x), y)) \leq \tau).$$

- *Prediction confidence based attack*: Using the same reasoning as just done above, it is intuitively clear the if the input is a data used in the training face, we expect an higher confidence score (like shown in the example in the introduction). Hence, we can classify a point correctly if the highest probability of the class - the confidence score - is bigger of a set threshold:

$$\mathcal{M}(p(y|x)) = I(\max p(y|x) \geq \tau).$$

- *Prediction entropy based attack*: Recall that for a discrete distribution $[p_1, \cdots, p_n]$ s.t $\sum_j p_j = 1$, the entropy is defined as

$$H(p_1, \cdots, p_n) = -\sum_{j=1}^{n} p_j \log_2 p_j$$

  and it achieves its maximum value when all $p_j = \frac{1}{n}$ for all j, and its minimum when all $p_j = 0$ except for one $p_i = 1$.

  Intuitively, the higher the *disorder* of the distribution, the higher is the entropy. When a point belongs to the data set, exactly as we said earlier, we expect its output vector $p(y|x)$ to be close to the distribution with all zeros and only one entry that accumulates the mass distribution, i.e it should have a low entropy. The classification is therefore defined as

$$\mathcal{M}(p(y|x), y) = I(H(p(y|x), y)) \leq \tau),$$

  for a set threshold τ.

In the chapter 3 will be using only two attacks, that seem to be more effective; *loss based attack* and *entropy based attack*, or their variations. The reason is that they better capture the overfitting of data. The drawback comes to the choice of the threshold τ, not easy to set. Whereas the less effective is the *prediction correctness based attack*: even if the data overfitted the model, a well trained classifier will likely classify correctly an unseen data. After all, this is what it is requited from a classifier.

Chapter 3

Differentially Private Optimization

3.1 Stochastic Convex Optimization

Definition 22. Let $\mathcal{D}$ be a bounded set of N data points in $\mathbb{R}^d$, $w \in \mathbb{R}^p$ the parameter vector.

$$\text{minimize} \quad \mathcal{L}(w;\mathcal{D}) := \frac{1}{N}\sum_{j=1}^{N} l(w, x_i, y_i) + \mu R(w)$$

is called Empirical Risk Minimization (ERM) with regularization $R(w)$. The minimization is taken over the set of parameter w.

The parameter μ controls how much we want to constrain our minimization.
$\mathcal{L}(w;\mathcal{D})$ is the loss function that we are going to use here.
The minimization of

$$\text{minimize} \quad \mathcal{L}(w;\mathcal{D}) := \frac{1}{N}\sum_{j=1}^{N} l(w, x_i, y_i)$$

$$\text{subject to } ||w||_1 \leq h$$

can be shown to be equivalent to ERM for a certain h, via dual Lagrange formulation.

Notation: For convenience, I denoted with $\mathcal{L}(w, \mathcal{D})$ the sum of the losses $\mathcal{L}$, i.e using the same $\mathcal{L}$.

Definition 23. Let $\mathcal{D} = \{d_1, \cdots, d_N\}$ be N samples drawn from a distribution and $\mathcal{C}$ a polyhedron. We define the risk

$$\hat{R}(w, \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} l(w, d_i) - \min_{w\in\mathcal{C}} \frac{1}{N}\sum_{i=1}^{N} l(w, d_i).$$

We are looking for an a random mechanism $\mathcal{A}$ of $\mathcal{D}$ to the space of parameters w, i.e $\mathcal{A} : \mathcal{D}^n \longrightarrow \mathbb{R}^p$ such that

$$\lim_{N\to\infty} \mathcal{R}(A) := \mathbb{E}_{\mathcal{A}}\left[\hat{R}(\mathcal{A}(\mathcal{D}), \mathcal{D})\right] = 0.$$

To minimize the empirical risk we need to know the data set $\mathcal{D}$. However, the access to the data set $\mathcal{D}$ might bring to a privacy violation that should be avoided. We will try to minimize $R(\mathcal{A})$) in a private way.

The following observation is useful to generalize the concept of ERM

Observation. Suppose $Z \sim U(\mathcal{D})$. Then,

$$\mathcal{L}(w, \mathcal{D}) = \frac{1}{N}\sum_{j=1}^{N} l(w, d_i)\frac{1}{N} = \sum_{j=1}^{N} l(w, Z = d_i)\mathbb{P}(Z = d_i) = \mathbb{E}_{Z\sim U(\mathcal{D})}[l(w, Z)] \ .$$

This means that the empirical risk corresponds to the expectation of $l(w, Z)$ when Z is a uniform distributed random variable.

Definition 24. We define

Population Risk: Let $\mathcal{P}$ a distribution over $\mathcal{D}$ and $l : \mathcal{X} \times \mathcal{D} \longrightarrow \mathbb{R}$. We define the population risk as

$$\mathcal{L}(w; D) = \mathbb{E}_{Z\sim\mathcal{P}}[l(w, Z)] \ .$$

When $\mathcal{P} = U(\mathcal{D})$, $\mathcal{L}$ is called Empirical Risk.

SCO: Let $\mathcal{C} \subset \mathcal{X}$ be a convex and closed set (we assumptions on the compactness in the definition). We define of minimizing the population risk

$$\min_{w\in\mathcal{C}} \mathcal{L}(w; D)$$

as **S**tochastic **C**onvex **O**ptimization.

DP-SCO: The problem of determining a DP private mechanism $\mathcal{A}(\mathcal{D}) \in \mathcal{X}$ such that

$$\mathbb{E}_{\mathcal{A}}[\mathcal{L}(\mathcal{A}(\mathcal{D}); \mathcal{D}) - \mathcal{L}^{\star}] \xrightarrow{N\longrightarrow\infty} 0$$

is called Differentially Private Stochastic Convex Optimization.

Throughout the chapters we will always assume that $l(\cdot, d)$ is $L-$ Lipschitz and $\beta-$smooth in the first entry.

Related work

A long line of works on differentially private private stochastic convex optimization focus on empirical risk minimization (ERM). Special emphasis is put on SGD algorithm for both SCO and ERM. In Table 3.1, the most known upper bounds of Excess Risk for both cases, are provided. In (BST14) it is provided a version of DP-SGD algorithm which achieves $\mathcal{O}\left(1/\sqrt{n} + \frac{\sqrt{d\log(1/\delta)}}{\epsilon N}\right)$, known to be tight. Moreover, if $d/\epsilon^2 = \mathcal{O}(n)$ the bound is comparable to the non-private SCO bound of $\mathcal{O}(1/\sqrt{N})$. However, the price for privatizing is reflected in both efficiency and utility, for which this algorithm requires $\mathcal{O}(\min\{N^{3/2}, N^{5/2}/d\})$ computations, i.e is less effective than its non-private version, that requires N computations.

Algorithm	No. of Iters.	Accuracy	Tot. comp. cost
DP-SGD (ERM) (BST14)	$T = N$	$\mathcal{O}\left(\frac{\sqrt{d\log(1/\delta)}}{\epsilon N}\right)$	
DP-SGD (SCO) (BFTT19)	$T = N^2$	$\mathcal{O}\left(1/\sqrt{N} + \frac{\sqrt{d\log(1/\delta)}}{\epsilon N}\right)$	$\mathcal{O}\left(\min\{N^{3/2}, \frac{N^{5/2}}{d}\}\right)$
DP-SGD (SCO) (HSS$^+$21)	$T = N$	$\mathcal{O}\left(1/\sqrt{N} + \frac{\sqrt{d\log(1/\delta)}}{\epsilon N}\right)$	$\mathcal{O}\left(\min\{N, \frac{N^{5/2}}{d}\}\right)$

Table 3.1: The accuracy of DP-SGD algorithm on dimension d solely depends on the noise introduced with the Gaussian mechanism.

3.2 Frank Wolfe Algorithm

Let $\mathcal{C}$ be the polyhedron defined through a convex hull of vertices, i.e $\mathcal{C} = \text{conv}\{s_1, \cdots, s_k\} \subset \mathbb{R}^p$ and let L be the Lipschitz constant of the loss ($||\nabla l(x) - \nabla l(y)|| \leq L||x - y||$). Let $\mathcal{D} = \{(x_1, y_1), \cdots (x_N, y_N)\}$ be the data set of points $(x_j, y_j) \in \mathbb{R}^d \times \mathbb{R}$. Then, we define the following

Algorithm 3 Differentially Private Frank-Wolfe, version 1

1: **Input**: ($\mathcal{D} = \{(x_1, y_1), \cdots (x_N, y_N)\}$, L, $\mathcal{C}_1$, T)
2: select $w_0 \in \mathcal{C}$
3: **for** $t = 1 : T$ **do**
4: $\quad \forall s \in \text{Vertex(C)}$, $\mathcal{M}_t(\mathcal{D}) := (\langle \nabla \mathcal{L}(w_t, \mathcal{D}), s\rangle + u_s)$ where $u_s \sim \text{Lap}(\lambda)$
5: $\quad \hat{s_t} = \arg\min_s \mathcal{M}_t(\mathcal{D})$
6: $\quad w_{t+1} = w_t + (\hat{s_t} - w_t)\gamma_t$ where $\gamma_t = \frac{2}{2+t}$
7: **Output**: $w_{DP} = w_T$

We will show that this algorithm is DP using 19. In 3, We are injecting noisy by a RMN mechanism. This makes the line 4. of 3 $\varepsilon-$DP. Furthermore, w_t depends by the previous w_{t-1} for all t, i.e all the conditions of the adaptive composition theorem are satisfied. To be more precised, with a correct choice of λ this algorithm is $(\varepsilon, \delta) - DP$.

Theorem 25 (Privacy Guarantee)**.** *The DP-FW algorithm 3 is (ε, δ) differentially private if λ is set as*

$$\lambda = \frac{8L||\mathcal{C}||_1 \cdot ||\mathcal{D}||_\infty \sqrt{2T\log(1/\delta)}}{N\varepsilon},$$

with $T\varepsilon^2 \leq \log\left(\frac{1}{\delta}\right)$.

Proof. Every iterate of [3] is $\hat{\varepsilon}$- DP private because $u_j \sim \text{Lap}\left(\frac{\gamma}{\hat{\varepsilon}}\right)$ with $\lambda = \gamma\ \hat{\varepsilon}$. By the advanced composition theorem with $\delta = 0$, the whole algorithm is then

$$\begin{cases} 4(2\hat{\varepsilon})\sqrt{2T\log(1/\delta)} \\ \delta \end{cases} \qquad - DP.$$

We want our final outcome to be $(\varepsilon, \delta) = 8\hat{\varepsilon}\sqrt{2T\log(1/\delta)}, \delta)$ DP, for any choice of $\varepsilon > 0$. Hence, $\hat{\varepsilon}$ needs to be chosen as

$$\hat{\varepsilon} = \frac{\varepsilon}{8\hat{\varepsilon}\sqrt{2T\log(1/\delta)}}$$

that, substituted in λ,

$$\lambda = \frac{\gamma 8\sqrt{2T\log(1/\delta)}}{\varepsilon}.$$

The only thing left to do is to estimate the sensitivity of

$$\Delta_j := \sup_{\mathcal{D}\sim\mathcal{D}'} |\langle \mathcal{L}(w_t, \mathcal{D}), s_j\rangle - \langle \mathcal{L}(w_t, \mathcal{D}'), s_j\rangle|,$$

when $\mathcal{D}$ and $\mathcal{D}'$ differ in one element.

Remark: it is enough to estimate the sensitivity of each coordinate and not as a vector of all of them because it is being used a Report noisy Max mechanism.

Before we move on, we need a technical lemma first.

Lemma 26. *Let $l(\cdot, z)$ be both L_0 Lipschitz and convex w.r.t $||\cdot||$ in the first argument. Then*

$$||\nabla l(w, z_1) - \nabla l(w, z_2)||_\star \leq 2L_0 \quad \forall w \in \mathcal{C}, \quad \forall z_1, z_2 \in \mathcal{D}\ .$$

Proof. By convexity of $l(\cdot, z)$

$$\begin{aligned} l(w', z_1) \geq& l(w, z_1) + \langle \nabla l(w, z_1), w' - w\rangle \\ l(w', z_2) \geq& l(w, z_2) + \langle \nabla l(w, z_2), w' - w\rangle \end{aligned}$$

and subtracting these two inequalities

$$\begin{aligned} \langle \nabla l(w, z_1) - \nabla l(w, z_2), w' - w\rangle \leq& l(w, z_2) - l(w', z_2) + l(w', z_1) - l(w, z_1) \\ \leq& |l(w, z_2) - l(w', z_2)| + |l(w', z_1) - l(w, z_1)| \\ \leq& L_0||w - w'|| + L_0||w - w'|| = 2L_0||w - w'|| \quad \forall w \neq w'\ . \end{aligned}$$

Dividing both sides by $||w-w'||$, and considering that the inequality holds **for any** $w' \in \mathbb{R}^N$

$$\langle \nabla l(w, z_1) - \nabla l(w, z_2), \frac{w' - w}{||w - w'||}\rangle \underbrace{=}_{\forall w'} \underbrace{\max_{||z||=1} \langle \nabla l(w, z_1) - \nabla l(w, z_2), z\rangle}_{=:||\nabla l(w,z_1)-\nabla l(w,z_2)||_\star} \leq 2L_0\ .$$

□

Using the Holder inequality, each $|\langle \nabla\mathcal{L}(w_t, \mathcal{D}) - \nabla\mathcal{L}(w_t, \mathcal{D}'), s_j\rangle|$ is bounded by

$$|\langle \nabla\mathcal{L}(w_t, \mathcal{D}) - \nabla\mathcal{L}(w_t, \mathcal{D}'), s_j\rangle| \leq ||\nabla\mathcal{L}(w_t, \mathcal{D}) - \nabla\mathcal{L}(w_t, \mathcal{D}')||_\infty \cdot ||\mathcal{C}||_1$$

$$\begin{aligned} ||\nabla\mathcal{L}(w_t, \mathcal{D}) - \nabla\mathcal{L}(w_t, \mathcal{D}')||_\infty &= ||\frac{1}{N}\sum_{j=1}^{N} \nabla l(w_t, d_j) - \nabla l(w_t, d'_j)||_\infty \\ &\underbrace{=}_{\mathcal{D}\sim\mathcal{D}'} \frac{1}{N}||\nabla l(w_t, d_k) - \nabla l(w_t, d'_k)||_\infty \leq \frac{2L_0}{N} =: \gamma. \end{aligned}$$

where $||\mathcal{D}||_\infty$ denotes the maximum in infinity norm of the points in the datasets, that are bounded by hypothesis. To conclude,

$$\lambda = \frac{16L_0 \cdot \mathcal{C}_1\sqrt{2T\log(1/\delta)}}{N\varepsilon}.$$

□

Observation. The noise does not depend on the number of vertices, but it only depends on the sensitivity of each $\langle \mathcal{L}(w_t, \mathcal{D}), s_j\rangle$. This is a property that follows immediately from the Report Noisy Mechanism. Also, observe that if we had used the property of the Laplacian or Exponential mechanism, we would had gotten the same λ multiplied by the the number of vertices, i.e a noise p times greater.

About the convergence: We have to show that the empirical risk is bounded by a function with nice asymptotic properties:

Theorem 27 (Utility Guarantee, similar to (TGTZ15))**.** *Suppose $l(\cdot, d)$ is L_0 Lipschitz and L_1 smooth w.r.t $||\cdot||_1$. The algorithm 3 above, with λ determined by the previous theorem and performing T iterations, has a utility guarantee of*

$$\mathbb{E}[\mathcal{L}(w_T, \mathcal{D}) - \mathcal{L}(w^\star, \mathcal{D})] \leq \mathcal{O}\left(\frac{L||\mathcal{D}||_\infty \cdot \mathcal{C}_1 \log(k)\sqrt{T\log(1/\delta)}}{N\varepsilon}\right)$$

Proof. Using the convexity of $\mathcal{L}$ and the L_1-smoothness, by the descent lemma we get the inequality

$$\begin{aligned}\mathcal{L}(w_{t+1}) \leq& \mathcal{L}(w_t) + \langle \nabla\mathcal{L}(w_t), w_{t+1} - w_t\rangle + \frac{L_1}{2}||w_{t+1} - w_t||^2 &(3.1)\\ \leq& \mathcal{L}(w_t) + \gamma_t\langle \nabla\mathcal{L}(w_t), s_t - w_t\rangle + \frac{L_1}{2}\mathcal{C}_1\gamma_t^2 &(3.2)\end{aligned}$$

where we used the step 6. of algorithm (3) for w_{t+1}.

Furthermore, we observe that by definition of $\mathbf{s}_t$

$$\langle \nabla\mathcal{L}(w_t), s_t\rangle + \min_{s\in\mathcal{C}} u_s \leq \langle \nabla\mathcal{L}(w_t), s_t\rangle + u_{s_t} \leq \langle \nabla\mathcal{L}(w_t), s\rangle + \max_{s\in\mathcal{C}} u_s \quad \forall s \in \mathcal{C},$$

which implies

$$\langle \nabla\mathcal{L}(w_t), s_t\rangle \leq \langle \nabla\mathcal{L}(w_t), s\rangle + (\max_{s\in\mathcal{C}} u_s - \min_{s\in\mathcal{C}} u_s) \quad \forall s \in \mathcal{C}.$$

Notice that $\max_{s\in\mathcal{C}} u_s$ and $\min_{s\in\mathcal{C}} u_s$ are pointwise max and min of random variables. Therefore, this is also a random upper bound. There was no way to avoid this drawback since s_t is also a random variable. This upper bound holds for any $s \in \mathcal{C}$, thus for $s = w^\star$.

$$\begin{aligned}\mathcal{L}(w_{t+1}) \leq& \mathcal{L}(w_t) + \gamma_t\langle \nabla\mathcal{L}(w_t), s_t - w_t\rangle + \frac{L_1}{2}\mathcal{C}_1\gamma_t^2 &(3.3)\\ \leq& \mathcal{L}(w_t) + \gamma_t\langle \nabla\mathcal{L}(w_t), w^\star - w_t\rangle + \gamma_t(\max_{s\in\mathcal{C}} u_s - \min_{s\in\mathcal{C}} u_s) + \frac{L_1}{2}\mathcal{C}_1\gamma_t^2. &(3.4)\end{aligned}$$

Now, by convexity of $\mathcal{L}$

$$\langle \nabla\mathcal{L}(w_t), w^\star - w_t\rangle \leq \mathcal{L}(w^\star) - \mathcal{L}(w_t).$$

Using this upper bound in the inequality above and subtracting $\mathcal{L}(w^\star)$ on both sides

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w^\star) \leq \mathcal{L}(w_t) - \mathcal{L}(w^\star) - \gamma_t(L(w_t) - \mathcal{L}(w^\star)) + \gamma_t(\max_{s\in\mathcal{C}} u_s - \min_{s\in\mathcal{C}} u_s) + \frac{L_1}{2}\mathcal{C}_1\gamma_t^2.$$

and with $\hat{R}(w_t) = \mathcal{L}(w_t) - \mathcal{L}(w^\star)$,

$$\hat{R}(w_{t+1}) \leq (1-\gamma_t)\hat{R}(w_t) + \gamma_t(\max_{s\in\mathcal{C}} u_s - \min_{s\in\mathcal{C}} u_s)u_j + \frac{L_1}{2}\mathcal{C}_1\gamma_t^2.$$

We can now take the expectation on both sides over the randomness of the laplacian Mechanism (recalling that $R = \mathbb{E}[\hat{R}]$)

$$R(w_{t+1}) \leq (1-\gamma_t)R(w_t) + \gamma_t\mathbb{E}\left[\max_{s\in\mathcal{C}} u_s - \min_{s\in\mathcal{C}} u_s\right] + \frac{L_1}{2}\mathcal{C}_1\gamma_t^2.$$

It can be shown (see Appendix) that

$$\mathbb{E}\left[\max_{s\in\mathcal{C}} u_s - \min_{s\in\mathcal{C}} u_s\right] = \mathcal{O}(\lambda\log(k)) \cong \lambda\log(k) + small(k,\lambda).$$

Let us go back to the inequality. To simplify the notation, let $M_1 = \lambda\log(k)$ and $M_2 = L_1/2\mathcal{C}_1$. Then, we find a recursive relation

$$R(w_{t+1}) \leq (1-\gamma_t)R(w_t) + \gamma_t M_1 + \gamma_r^2 M_2.$$

It can be proved, (see [49]) that

$$R(w_t) \leq M_1 + M_2 a_t,$$

where

$$\begin{cases} a_0 = 1 \\ a_{t+1} = a_t(1-\gamma_t) + \gamma_t^2 \quad \gamma_t = \frac{2}{2+t}. \end{cases}$$

This sequence $(a_t)_{t\geq 0}$ is monotone (decreasing) and positive for all t and converges to 0 as $\frac{1}{t}$. In conclusion, with $T = t$ iterations,

$$\mathbb{E}[\mathcal{L}(w_T,\mathcal{D}) - \mathcal{L}(w^\star,\mathcal{D})] := R(w_T) \leq M_1 + M_2 a_t \sim M_1 + \frac{1}{T}M_2 \quad \forall T >> 1.$$

There exists a sufficiently large T, such that $\frac{1}{T}M_2 \leq M_1(T)$, hence $\mathbb{E}[R(w_T)] \leq 2M_1(T)$. Substituting the value of $M_1 = M_1(\lambda(T))$ and M_2, taking into account that λ is set equal to $\lambda = \frac{16L_0\mathcal{C}_1\sqrt{2T\log(1/\delta)}}{N\varepsilon}$ to ensure $(\epsilon,\delta) - DP$:

$$\mathbb{E}[\mathcal{L}(w_T,\mathcal{D}) - \mathcal{L}(w^\star,\mathcal{D})] \leq \mathcal{O}\left(\frac{L_0\mathcal{C}_1\ \log(k)\sqrt{2T\log(1/\delta)}}{N\varepsilon}\right).$$

□

Observation. To be more specific, we can do better. We know that $\mathbb{E}[\mathcal{L}(w_T,\mathcal{D}) - \mathcal{L}(w^\star,\mathcal{D})] := R(w_T) \sim M_1 + \frac{1}{T}M_2 \quad \forall T >> 1$. Therefore,

$$\mathbb{E}[\mathcal{L}(w_T,\mathcal{D}) - \mathcal{L}(w^\star,\mathcal{D})] \leq \mathcal{O}\left(\frac{L_0\mathcal{C}_1\ \log(k)\sqrt{2T\log(1/\delta)}}{N\varepsilon} + \frac{L_1\mathcal{C}_1}{2T}\right).$$

Optimizing over T,

$$T = \mathcal{O}\left(\frac{L_1^{2/3}(N\varepsilon)^{2/3}}{L_0^{2/3}[\log(1/\delta)]^{1/3}[\log(k)]^{2/3}}\right) .$$

Plugging this number of iterations into the upper bound, we do find — analogous of (TGTZ15)

$$\mathbb{E}[\mathcal{L}(w_T, \mathcal{D}) - \mathcal{L}(w^\star, \mathcal{D})] \leq \mathcal{O}\left(\frac{\mathcal{C}_1 L_0^{2/3} L_1^{1/3}[\log(1/\delta)]^{1/3}[\log(k)]^{2/3}}{(N\varepsilon)^{2/3}}\right) .$$

Lemma 28. *The expectation* $\mathbb{E}[\max_{s\in\mathcal{C}} u_s]$ *satisfies*

$$\mathbb{E}[\max_{s\in\mathcal{C}} u_s] = \mathcal{O}(p\log(k))$$

where k *is the number of vertices of the polyhedron* $\mathcal{C}$ *and* p *is the dimension of the space.*

Proof. Since $\min u_s = -\max -u_s$ and $-u_s$ is still a Laplace with same parameter λ - because the mean is 0 and Laplacian has a symmetric density, without loss of generality we assume to bound $\max_s u_s$.

For every $r > 0$, using the fact that u_j are i.i.d,

$$\exp\left(t\mathbb{E}[\max_{j\in\mathcal{C}} u_j]\right) \leq \mathbb{E}[\exp\left(r\max_{j\in\mathcal{C}} u_j\right)] \leq \mathbb{E}\left[\sum_{i=1}^{k}\exp\left(ru_i\right)\right] = k\mathbb{E}[\exp\left(tu\right)]$$

where $u \sim Lap(\lambda)$. Taking log on both sides (log is monotone, so it preserves the inequality)

$$\mathbb{E}[\max_{j\in\mathcal{C}} u_j] \leq \frac{\log\left(k\mathbb{E}[\exp\left(tu\right)]\right)}{t} \quad \text{for all } t > 0.$$

Since the density function of the Laplacian distribution is known, $\mathbb{E}[\exp(tu)]$ can easily be computed as an integral in the usual way. It can also be shown that this value is finite if and only if $0 < t < \frac{1}{\lambda}$ and that

$$\mathbb{E}[\exp\left(tu\right)] = \frac{2}{(1+t\lambda)(1-\lambda t)} \leq \frac{2}{1-\lambda t} \quad \text{for all } 0 < t < \frac{1}{\lambda}.$$

Choosing $t = \frac{1}{2\lambda}$, we finally get the upper bound

$$\mathbb{E}[\max_{j\in\mathcal{C}} u_j] \leq 2\lambda\log\left(4k\right).$$

□

This lemma overkills, and indeed we can do better computing more precisely the expectation and obtaining $\lambda\log(k)$

Remarks:

1) The higher is the value of ε, the lower is the expected risk. However, λ of the form $\lambda = \frac{\cdot}{\varepsilon}$ implies an amplification of the noise introduced in the learning phase, hence a very poor privacy guarantee.

3) It is natural that T appears at the numerator, though it might be counter intuitive. We want that the empirical risk goes to zero when the number of points in the dataset goes to infinity. We then adjust the T accordingly, i.e as a function of N.

4) A similar bound is found in (TGTZ15). The difference is that the curvature function has not been used, but we assumed smoothness of the loss function $\mathcal{L}$ - continuity of the gradient. Therefore, asymptotically is the same result, we are only changing constants.

	Noise: δ, λ	Excess Risk
SDG	$8L^2 \log(1/\delta)/\varepsilon^2$	$\mathcal{O}\left(L\|\|\mathcal{D}\|\|\frac{1+\sqrt{8p\log(1/\delta)}}{\sqrt{N}\varepsilon}\right)$
DP-FW	$8L\|\|D\|\| \cdot \|\|\mathcal{C}\|\|\sqrt{2\log(1/\delta)}/\sqrt{N\varepsilon}$	$\mathcal{O}\left(\frac{L\|\|\mathcal{D}\|\|\cdot\|\|\mathcal{C}\|\|\log(2p)\sqrt{\log(1/\delta)}}{\sqrt{N}\varepsilon}\right)$

Table 3.2: Noise and Excess Risk with $T = N$, The crucial difference is in the factor $\log(2p)$ and p in the Excess Risk. FW is expected to perform better on highly dimensional data respect to SDG. Here FW is being performed over the ℓ^1 ball, thus there are $k = 2p$ vertices.

3.3 Stochastic Frank Wolfe

As previously discussed in the utility guarantee of Frank Wolfe, the gradient $\nabla\mathcal{L}(w) = \sum_i \nabla l(w, d_i)$, takes $\mathcal{O}(pN)$ computations, which makes it quite slow. We shall brie As in SGD, we need to construct an unbiased estimator of the gradient that is cheap in terms of computations, and replace it with $\nabla\mathcal{L}$. We will denote such estimator with ∇_t.

Suppose we want to consider the population minimization problem $\min_w \mathbb{E}_{z\sim\mathcal{P}}[l(w, z)]$. Further, given the distribution $\mathcal{P}$, we can sample b $(z_i)_{i=1}^b$ i.i.d from $\mathcal{P}$. Then, an unbiased estimator of the gradient $\mathcal{L}$ can be constructed by

$$\frac{1}{b}\sum_{i=1}^{b} \nabla l(w, z_i) \quad z_i \sim \mathcal{P} \quad i.i.d\ .$$

In fact,

$$\begin{aligned}\mathbb{E}_{z\sim\mathcal{P}}\left[\frac{1}{b}\sum_{i=1}^{b}\nabla l(w, z_i)\right] =& \frac{1}{b}\sum_{i=1}^{b}\mathbb{E}_{z\sim\mathcal{P}}[\nabla l(w, z_i)] \\ =& \frac{1}{b}\sum_{i=1}^{b}\nabla \underbrace{\mathbb{E}_{z_i\sim\mathcal{P}}[l(w, z_i)]}_{\mathcal{L}(w,z)} = \frac{1}{b}\sum_{i=1}^{b}\nabla\mathcal{L}(w, z) \\ =& \nabla\mathcal{L}(w, z)\ .\end{aligned}$$

We used the fact that z_i are i.i.d and that here we can interchange derivatives and integrals. This estimator posses a fundamental property

Lemma 29. *Suppose $\mathcal{L}(w) = \mathbb{E}_{z\sim\mathcal{P}}[l(w, z)]$ with $\max_{supp(z)} ||\nabla l(w, z)||_\star \leq G$ for some constant G, then the estimator ∇ satisfies*

$$\mathbb{E}[||\nabla - \nabla\mathcal{L}(w, z)||] \leq \frac{G}{\sqrt{b}}\ .$$

In other words, the variance of the estimator is bounded. This yields the following stochastic version of Frank Wolfe:

Algorithm 4 Stochastic Frank-Wolfe, population (RSPS16)

1: **Input:** T number of iterations, batch size b, step size γ
2: select $w_0 \in \mathcal{C}$
3: **for** $t = 0 : T - 1$ **do**
4: Sample $z_1, \cdots, z_b$ i.i.d from $\mathcal{P}$.
5: Compute $\nabla_t = \frac{1}{b} \sum_i^b \nabla l(w_t, z_i)$
6: $\hat{s}_t = \underset{s \in \text{vertex}(\mathcal{C})}{\arg\min} \langle \nabla_t, s \rangle$
7: Update: $w_{t+1} = w_t + (\hat{s}_t - w_t)\gamma$
8: **Output**: w_U where $U \sim$Uni$(\{0, \cdots, T-1\})$.

Remark 1. Note that nothing is yet private.

It can be shown that this algorithm converges ref and that it can be made private in the usual sense we imagine, by adding Laplacian noise and using a report max mechanism in line 6:. We will be using this algorithm as framework in chapter 5 for algorithm 6.

Chapter 4

Numerical Experiments

In this Chapter we will implement the attacks and defence presented in the previous chapters. We will consider three models: Linear Regression, Logistic Regression and Multinomial Logistic Regression

4.1 ℓ^1 Constrained Regression

We shall laid down the details regarding the implementation of Algorithm 3 on LASSO (equivalent to ℓ^1 Constrained Regression) and perform a simple MIA attack, explained below. This attack is based on the framework of the Prediction Based attack presented in the second chapter.

Recall the setting: Let $\{x_1, \cdots, x_N\} = \mathcal{D}_{training}$ be a data set with labels $y_i = w^\star + noise$. For every point we have the loss l_i defined as

$$l_i : \mathbb{R} \longrightarrow \mathbb{R} \quad t \to \frac{1}{2}(t - y_i)^2.$$

Then, the LASSO is defined as the minimization of the following

$$\begin{aligned} \min_{w} \quad & \frac{1}{N}\sum_{i=1}^{N} l_i(\langle w, [1, x_i]\rangle) \\ \text{s.t.} \quad & ||w||_1 \leq C \end{aligned} \tag{4.1}$$

where $w \in \mathbb{R}^{p+1}$.

Observations and assumptions:

- It is known that the ℓ^1 ball in $\mathbb{R}^{p+1}$ is the convex hull of the canonical basis of $\mathbb{R}^{p+1}$: $||w||_1 \leq C = C \cdot \text{conv}\{\pm e_1, \cdots, \pm e_{p+1}\}$. Thus, the number of vertices is $2(p+1) = \mathcal{O}(p)$.

- Without loss of generality, we assume that $||\mathcal{D}|| = 1$.

- each $l_i(\langle w, x\rangle)$ is smooth with constant $L = \mathcal{O}(1)$

- The gradient of the *Empirical Loss*

$$\nabla_w \left(\sum_{i=1}^{N} l_i(\langle w, [1, x_i] \rangle) \right) = \sum_{i=1}^{N} \nabla_w l_i(\langle w, [1, x_i] \rangle) = \sum_{i=1}^{N} l_i'(\langle w, [1, x_i] \rangle)[1, x_i] = \\ = \sum_{i=1}^{N} (\langle w, [1, x_i] \rangle - y_i)[1, x_i].$$

- The actual parameters of the model, $w^\star$, might not be in $C\ell^1$. In order to keep things simple, $C := ||w^\star||$. In particular, this choice implies that the exact minimizer could be computed in practice, even though the minimization includes constrains.

We will design an attack as follow: [3] is used to learn a set of parameters w_{DP}. Then we can generate another data set $\mathcal{D}_{test}$ with the same distribution of $\mathcal{D}_{training}$. The enriched data set $\mathcal{D}' := \mathcal{D}_{training} \cup \mathcal{D}_{test}$ shall be used for the attack. $\mathcal{D}'$ is being fed to the model, i.e we are computing the labels $y_{predict} = w_{DP} \cdot \mathcal{D}$. Reasonably, the points belonging to the data set $\mathcal{D}_{training}$ which was used for training the model have somehow over-fitted the model during training, thus their loss is expected to be lower.

Algorithm 5 Prediction Based Membership Inference Attack

Input: $\mathcal{D}_{training} = \{(x_1, y_1), \cdots (x_N, y_N)\}$
Draw $\mathcal{D}_{test} \sim \mathcal{D}_{training}$
Set $\mathcal{D} = \mathcal{D}_{test} \cup \mathcal{D}_{training}$
Set δ
for $\varepsilon \in (0, 1]$ **do**
    Compute $w_{DP}^{\varepsilon} = DP.FW(\mathcal{D}_{training}, \varepsilon, \delta)$
    for $x \in \mathcal{D}$ **do**
        Compute $\mathcal{L}(\mathcal{D}).append(\mathcal{L}(w_{DP}^{\varepsilon}, x))$
    $\mathcal{L}(D) \leftarrow sort(\mathcal{L}(\mathcal{D}))$
    $Counter = 0$
    for $k = 1 : N/2$ **do**
        find x such that $\mathcal{L}(w_{DP}^{\varepsilon}, x) \in \mathcal{L}(\mathcal{D})[k]$
        if $x \in \mathcal{D}_{training}$ **then**
            $Counter++$
    for $k = N/2 + 1 : N$ **do**
        find x such that $\mathcal{L}(w_{DP}^{\varepsilon}, x) \in \mathcal{L}(\mathcal{D})[k]$
        if $x \in \mathcal{D}_{test}$ **then**
            $Counter++$
    Store $accuracy.append(Counter/N\%)$ for ε
Output: *accuracy*

Therefore, the most natural attack would be the one in alg. 5. After having fixed a level of privacy ε, we run a $DPFW$ with this ε and compute the global loss $\mathcal{L}(\mathcal{D})$. Sorting this vector from the lowest to the highest value, it reasonable to think that the first N points with the lowest loss are those who were in the training data set $\mathcal{D}_{training}$.

Applying the algorithm 3 in step 7. of algorithm 5 with $T = N\varepsilon$ number of iterations, we obtain the upper bound on the excess risk of $\mathbb{E}[\mathcal{L}(w_{DP}^{\varepsilon}, \mathcal{D}) - \mathcal{L}^\star] = \mathcal{O}\left(\frac{C \log(4p) \sqrt{\log(1/\delta)}}{\sqrt{N\varepsilon}} \right)$.

These are some results of the MIA attack with DP-SGD and DP-FW alg 3.

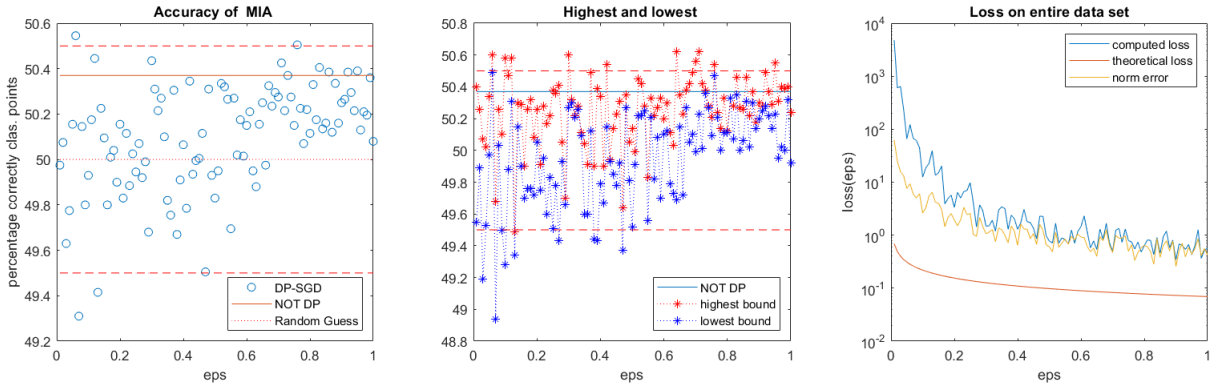


Figure 4.1: Left: Membership inference attack with $N = 1000$ points. On the left: Averages over 10 epochs. In the middle, the highest and lowest values of MIA over the 10 epochs. Right: Theoretical loss bound, computed loss and $||w^{\varepsilon}_{DP_{SGD}} - w^{\star}||$.

This means that the model has been trained on $\mathcal{D}_{training}$ with DP-SGD. For every ε, we ran [5] for $epochs = 10$ times and we display the averages $\frac{1}{epochs}\sum_{i=1}^{epochs} MIA(w^{\varepsilon}_{DP}, \mathcal{D})$ on the left. The two dot lines in red define a neighbor of 50%. We say that the attack is effective if most of the blue points lie within this region and are below the NOT DP MIA attack. In the middle, for every ε it is plotted $\max_{epochs} MIA(w^{\varepsilon}_{DP}, \mathcal{D})$ and $\min_{epochs} MIA(w^{\varepsilon}_{DP}, \mathcal{D})$.

For clearness, combining the first and second plot and fitting the blue dots helps to understand what is the behaviour of the MIA attack in $(0, 1]$, in Fig. 4.2.

Slightly better results are obtained increasing the number of points, as it is shown in the graph below and in Fig 4.4:

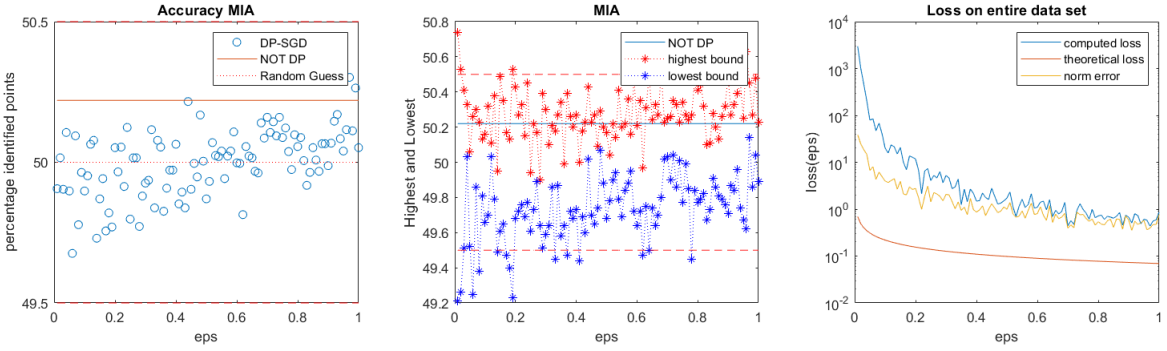


Figure 4.3: Left: Membership inference attack with $N = 10000$ points. The graphs shows the results of the accuracy for 10 epochs. The highest and the lowest recording has been added. Right: Theoretical loss bound vs the computed loss.

Similarly, the results of the attack when the defense uses DP-FW:

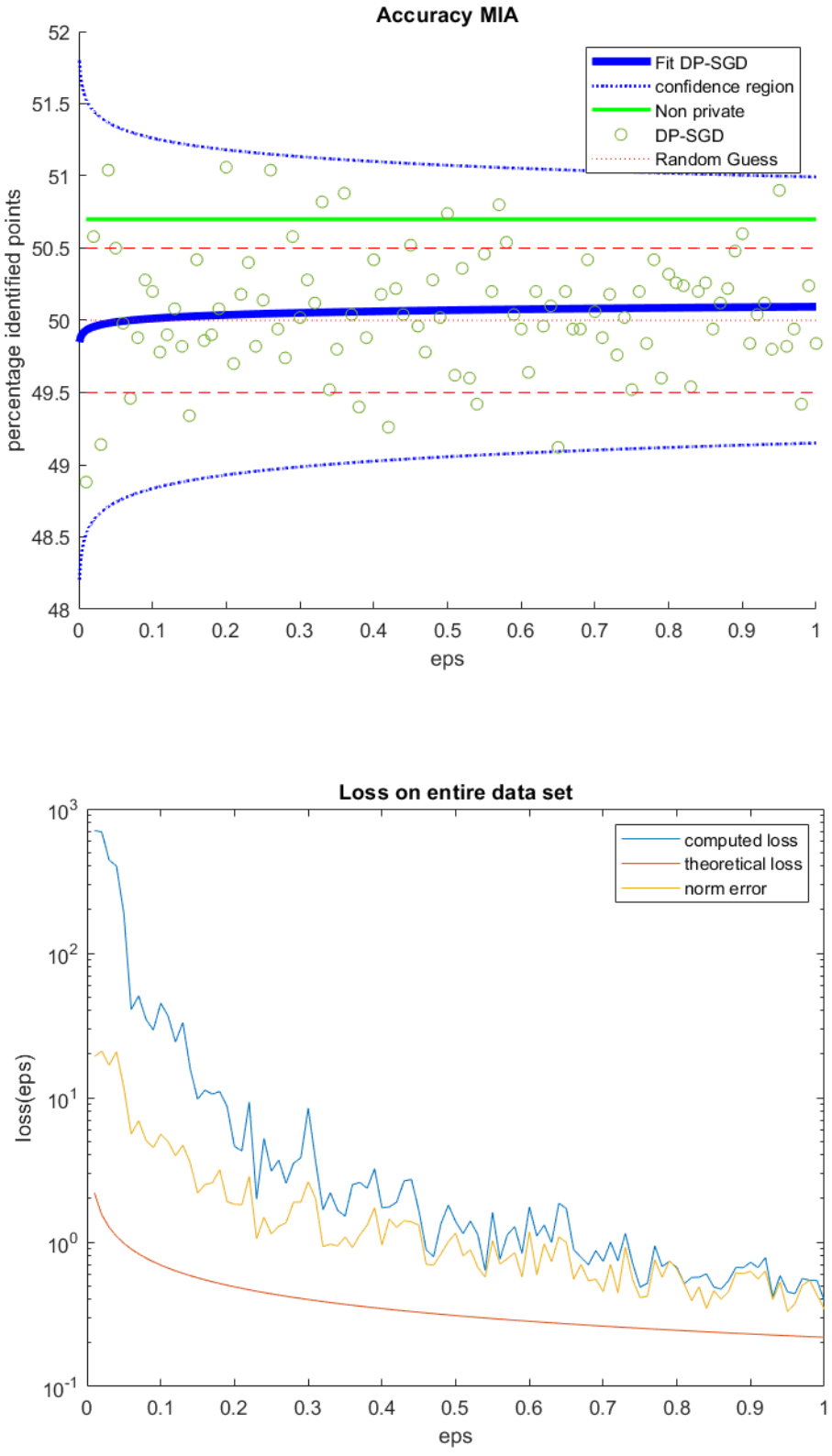
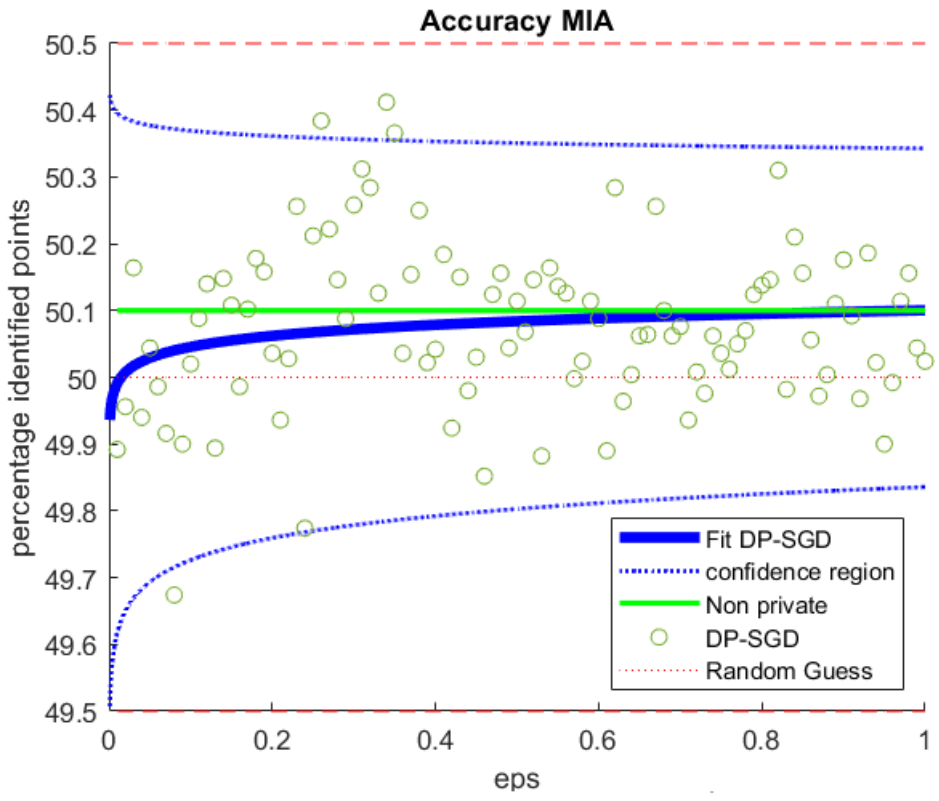


Figure 4.2: N=1000. The algorithm has been rerun and the plot in the middle of fig 4.1 appears as confidence regions; the red and blue dots have been fitted with a log model -shadow blue lines.



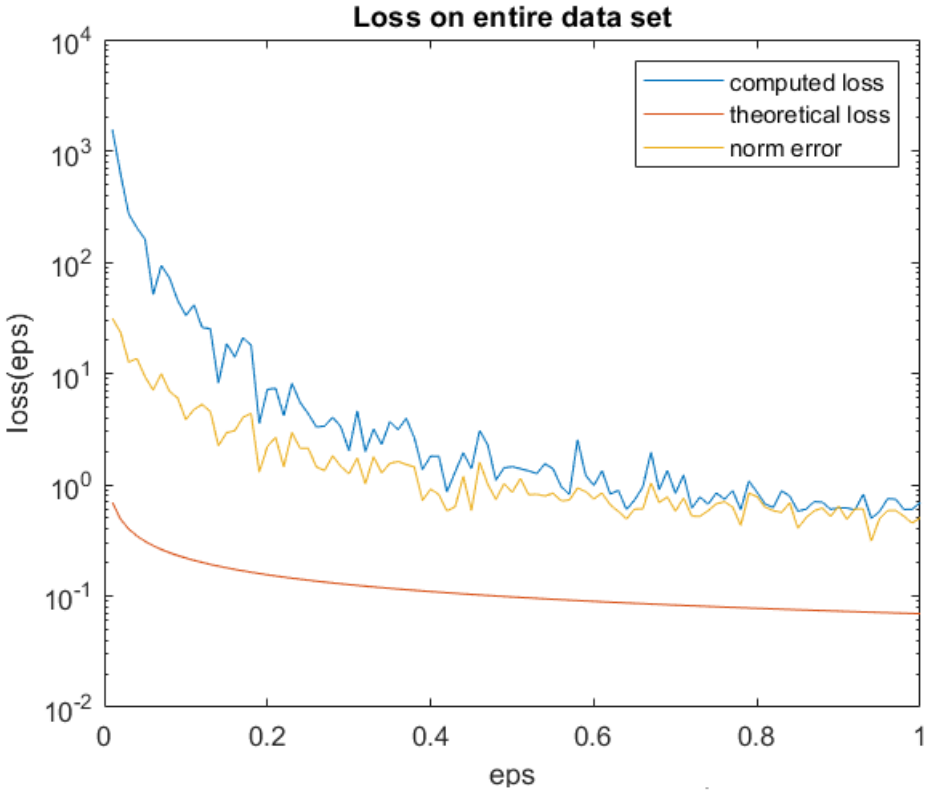


Figure 4.4: N=1000. appears as confidence regions; Not that the log fit converges to the Non private attack as ϵ gets larger.

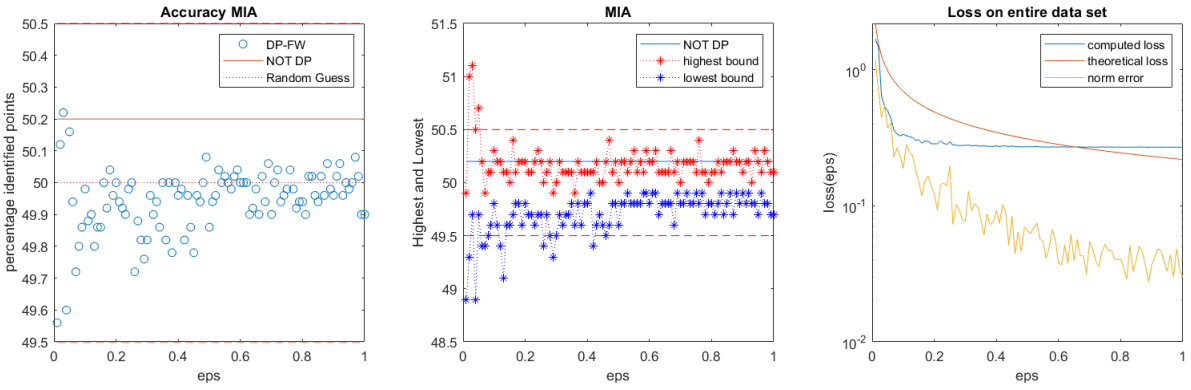


Figure 4.5: N=1000. The attack seems to be stable, even though for small ε the attack is meaningless. The loss does not decrease as fast as it did with DP-SGD.

The range of $\varepsilon \in (0, 1]$ is nice for privacy guarantees. However, one might wonder if it is enough to consider this range and not a large interval for ε. The experiments suggest that the more ε increases, the more the attacks level out.

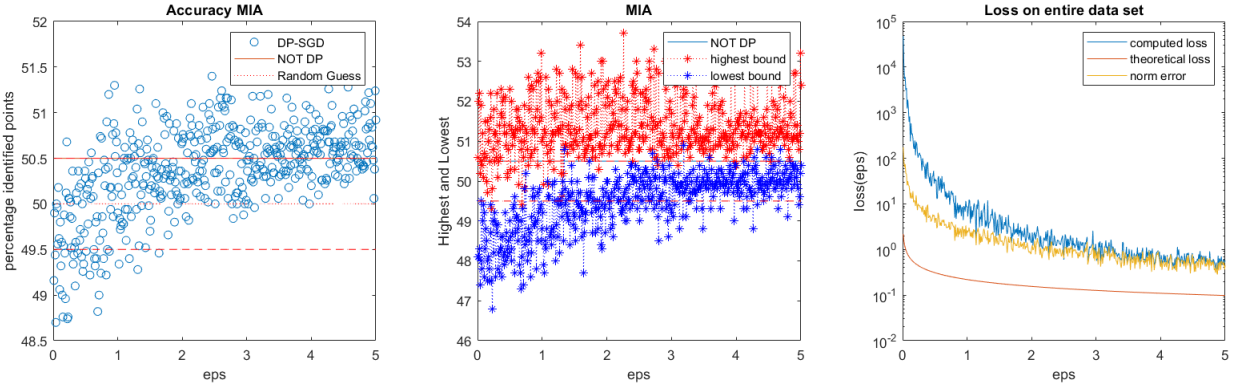


Figure 4.6: N=1000, for $\varepsilon \in (0, 5]$. The attack appears to be stable at $\varepsilon = 1$ already.

and

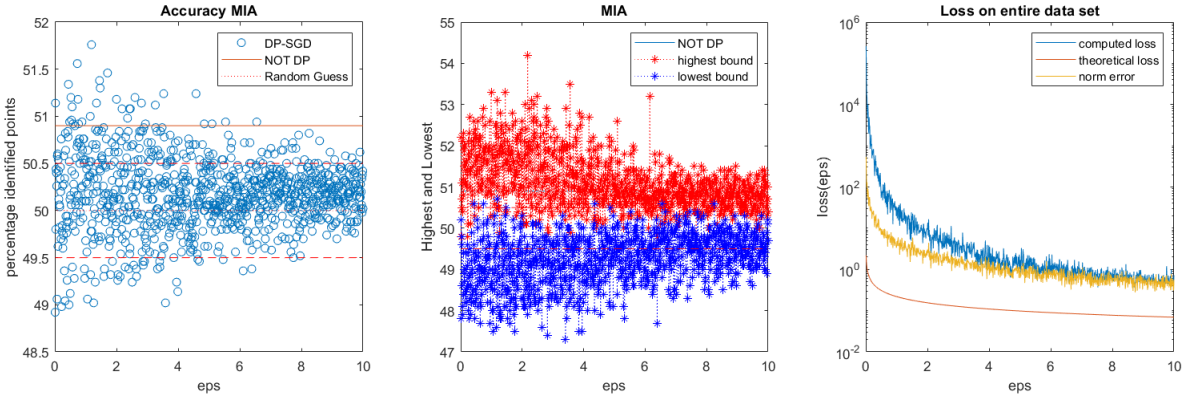


Figure 4.7: N=1000, for $\varepsilon \in (0, 10]$. The stabilization is sharper in this example. Small values of ε yield to larger fluctuations.

Let us comment these results. Empirically, the points level out at a value of ε around 1.

Most of the averages values are below the non private attack and usually slightly less than 50%. This proves and measures the protection of DP. In theory, with a perfectly trained model, a MIA attack has an expect value of 50% of accuracy, with a variance that decreases with the training. In other words this means that we expect MIA attacks without DP to have an accuracy slightly above 50%. Indeed, half of the points are generated from the same distribution of $\mathcal{D}_{training}$ and a perfect attack should have the same expected accuracy of a *random guess*. Goal of MIA attack is to improve this accuracy, pushing it above 50%, while the role of DP is to make it lower. As it can be seen from these simulations, the DP offers a slight protection against these attacks. Moreover, our intuition suggests that the smaller ε gets, the more the accuracy of the attack decreases, in accordance with the results displayed above, for which we observe this *log-trend*. On overall, we are right: most of the time the graphs will look like in 4.6. However, for values of ε in $(0, 0.2]$ the attack seems not to be so relevant for the privacy analysis. In fact, with $T = \varepsilon N$, for $\epsilon = 0.1$ we already need at least $N = 1000$ to perform $T = 100$ iterations, i.e we are not learning at all. Indeed, the graphs in 4.3 and 4.5 emphasize this discrepancy: with $N = 10000$, the blue points are shrunk (look at the band delimited by the two red lines). Also, rerunning [3] with the same setting as 4.6 we get:

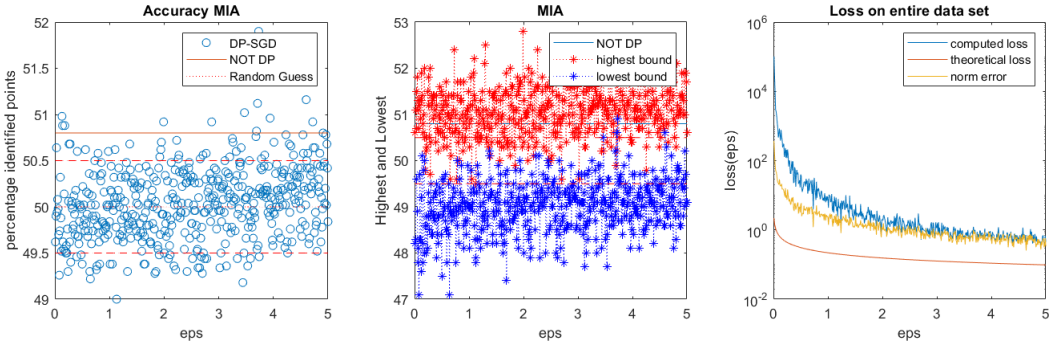


Figure 4.8: N=1000. for $\varepsilon \in (0, 5]$. The *log-trend* is still visible, but less evident. The attack is still effective, the most of the points are below the threshold of the non private attack.

In conclusion, most of the time the defences are effective against the attacks: the percentages do not lie outside the red band too much and most of the points are below the not DP attack, that is what we wanted. Also, we observed that the more privacy we add, the more the results are unreliable. Of course, the drawback is that that we need less privacy budget to make up for the lack of accuracy, clearly bringing to a privacy issue. However, the good new is that the attacks do converge and level out pretty soon, as we expected from the theory. Empirically - by looking at the graphs- this stabilization starts in a neighbor of $\varepsilon = 1$, thus it still makes sense to restrict the analysis to the range $(0.2, 1]$, keeping in mind that the greater ε, the better are the results. This shows that considering ranges of $(0, 100]$, like in (CWS20) is not necessary, unless the dimension $p \gg N$. In this case we need a privacy budget extremely high, in accordance with what we see in table 3.2.

4.2 Logistic Regression

We will repeat the same experiments of the previous section on the Logistic - Regression model. Recall that the goal is to perform classification: We want to classify a point $x \in \mathbb{R}^n$ as belonging to two classes, either 0 or 1. This information is stored in a label $y \in \{0, 1\}$. For the sake of experiments, we assume that the data can be linearly separable. The parameters of this model shall be denoted with $\beta = [\beta_0, \beta_1, \cdots, \beta_p]$. Recall that in a logistic regression we directly estimate the probability of a class, assuming there is relation between the weights and a probability of a class as

$$\log\left(\frac{\mathbb{P}(Y=1|X=x)}{1-\mathbb{P}(Y=1|X=x)}\right) = \langle [1,x], \beta\rangle.$$

From this relation, we obtain the probability of one class expressed as

$$\mathbb{P}(Y=1|X=x) = \frac{1}{1+e^{-\langle [1,x],\beta\rangle}} = \phi_\beta(\langle [1,x],\beta\rangle)$$

where $\phi(x)$ is the sigmoid function.

In practice, we are given the dataset of points $\mathcal{D} = \{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathbb{R}^p \times \{0, 1\}$, and we have to recover the vector of parameters β. We use the *maximum likelihood method*. First, note that the probability of $y_j \in \{0, 1\}$ given $X = x_j$ can compactivately written as

$$\begin{aligned}\mathbb{P}(Y=y_j|X=x_j) =& \mathbb{P}(Y=1|X=x_j)^{y_j}\mathbb{P}(Y=0|X=x_j)^{1-y_j}\\ =& \mathbb{P}(Y=1|X=x_j)^{y_j}(1-\mathbb{P}(Y=1|X=x_j))^{1-y_j}\\ =& p(x_j)^{y_j}(1-p(x_j))^{1-y_j}.\end{aligned}$$

Therefore, the log-likelihood will be

$$\begin{aligned}l(\beta) =& \log\left(\Pi_{j=1}^N p(x_j)^{y_j}(1-p(x_j))^{1-y_j}\right) = \sum_{j=1}^N y_j \log(p(x_j)) + (1-y_j)\log(1-p(x_j))\\ =& \sum_{j=1}^N y_j \log\left(\frac{1}{1+e^{-\langle [1,x_j],\beta\rangle}}\right) + (1-y_j)\log\left(\frac{e^{-\langle [1,x_j],\beta\rangle}}{1+e^{-\langle [1,x_j],\beta\rangle}}\right)\\ =& \sum_{j=1}^N y_j\langle [1,x_j],\beta\rangle - \log(1+e^{-\langle [1,x_j],\beta\rangle}).\end{aligned}$$

Thus

$$\beta^\star = \underset{\beta\in\mathbb{R}^{p+1}}{\arg\min} \sum_{i=1}^N \left[\log(1+\exp(\beta\cdot[1,x_i])) - y_i\beta\cdot[1,x_i]\right].$$

In order to apply the SGD and FW, we need to compute the derivative of $-l(\beta)$, that is

$$\frac{\partial}{\partial\beta}(-l(\beta)) = \sum_{i=1}^N (\phi_\beta([1,x_i]) - y_i)[1,x_i].$$

For the attack, it has been used a *Entropy Based attack*. The classification rule for a point $x \in \mathbb{R}^p$ with logistic regression is the following:

$$\text{If} \quad \phi_\beta(x) \geq 0.5 \quad \Longrightarrow \quad x \text{ is classified as 1.}$$

The intuition behind is clear: from the relation above, $\phi_\beta(x)$ is the probability that x is labeled or classified in class 1. Thus, if this probability is above $1/2$ then it is reasonable to think that x has label 1. More formally,

$$\phi_\beta(x) = \begin{bmatrix} p(x) \\ 1 - p(x) \end{bmatrix}.$$

As usual, an $x \in \mathcal{D}_{training}$ is inclined to have overfitted the model. As a results, it is expected that when such x is fed to the model, its output vector will be similar to

$$\phi_\beta(x) = \begin{bmatrix} \text{close to 1} \\ \text{close to 0} \end{bmatrix} \quad \text{or} \quad \phi_\beta(x) = \begin{bmatrix} \text{close to 0} \\ \text{close to 1} \end{bmatrix}.$$

Either way, the *entropy* of such $\phi_\beta(x)$ is roughly 0. This observation yields to the definition of our MIA:

Algorithm 6 Entropy Based Membership Inference Attack

Input: $\mathcal{D}_{training} = \{(x_1, y_1), \cdots (x_N, y_N)\}$
Draw $\mathcal{D}_{test} \sim \mathcal{D}_{training}$
Set $\mathcal{D} = \mathcal{D}_{test} \cup \mathcal{D}_{training}$
Set δ
for $\varepsilon \in (0, 1]$ **do**
    Compute $\beta^{\varepsilon}_{DP} = DP.FW(\mathcal{D}_{training}, \varepsilon, \delta)$
    for $x \in \mathcal{D}$ **do**
        Compute $\phi_{\beta^{\varepsilon}_{DP}}(x)$
        Compute $H(\phi_{\beta^{\varepsilon}_{DP}}(x))$
    $Counter = 0$
    for $x \in \mathcal{D}$ **do**
        if $H(\phi_{\beta^{\varepsilon}_{DP}}(x)) \leq \tau$ **then**
            $y_{Membership-pred}(x) = 1$
        else
            $y_{Membership-pred}(x) = 0$
        if $y_{Membership-pred}(x) = M(x)$ **then**
            $Counter++$
    Store $accuracy.append(Counter/N\%)$ for ε
Output: *accuracy*

- **Data**: The points $x \in \mathcal{D}$ are randomly generated and drawn uniformly in $[a, b]^p$ (first class) and $[c, d]^p$ (second class), in a way that they are linearly separable. a, b, c, d are chosen by hand, tuning the intersection between the two classes

- **Attack type**: In the experiments, we run alg. 6 with both DP-FW and DP-SGD. Also, the attack in alg 6 identifies the number of points whose *membership* has been guessed correctly. We are not identifying the actual number of members in $\mathcal{D}_{training}$. That would be the true-positive rate.

- **True model**: The actual vector of parameters β has been computing using a FW without privacy with number of iteration fixed to 10^5. This guarantees an accuracy of $\mathcal{O}(10^{-5})$ and empirical trials show that the resulting β achieves the highest accuracy in terms of **classification**. Hence, β will be set and consider as the true underneath model.

- **Delta**: $\delta = \exp(-\varepsilon^3 N)$ fixed in all experiments - this choice satisfies $\log(1/\delta) \geq \varepsilon^2 T$.

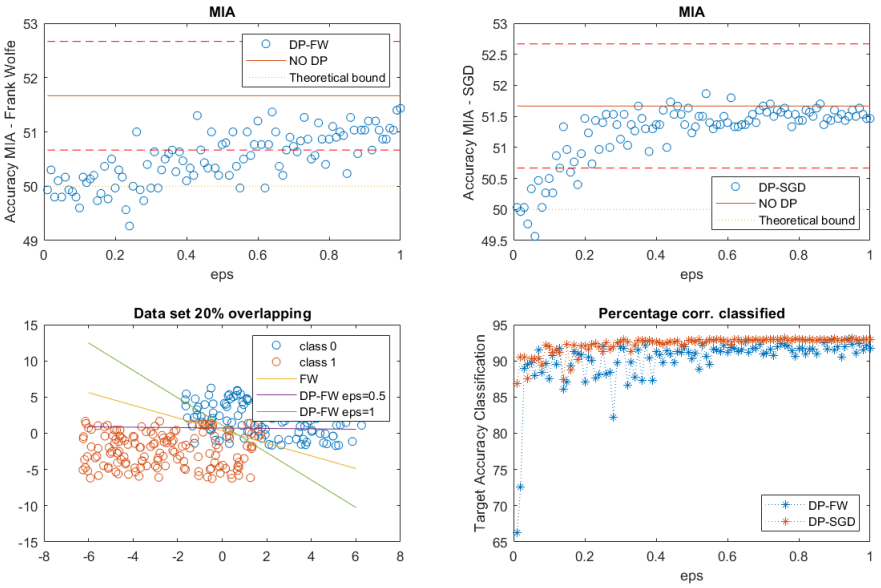


Figure 4.9: Up left: Entropy based attack with $DP - FW$ protection. Up right: Same attack with $DP - SGD$ learning. The two graphs look similar. However, the SGD seems to show stabilize a bit quicker while FW guarantees more privacy. Indeed, observe that most of the points are below the threshold of the true NOT DP attack with DP-FW. Down left: plotting of the data set for $N = 300$, and plotting of the classifier for values of $\varepsilon \in \{0.5, 1\}$ and the optimal one. Down right: Target accuracy for both DP-SGD and DP-FW.

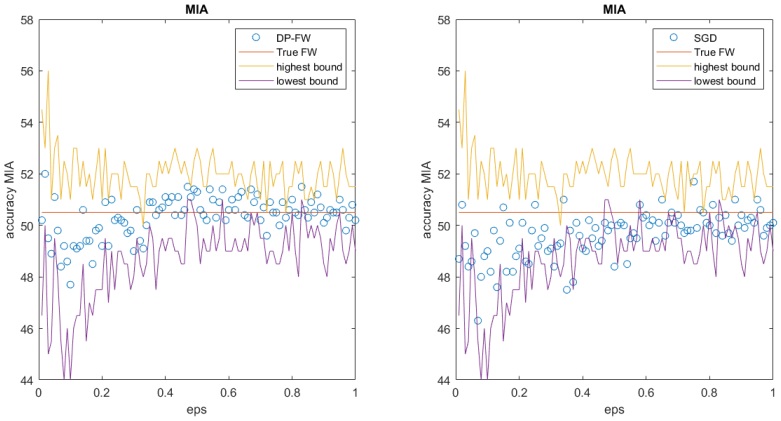


Figure 4.10: Rerunning the same attack and plotting the highest and lowest record, dimension $p = 11$.

As we can see from example 4.9, there differences between the two learning algorithms. DP-SGD converges faster and it levels out sooner, while one could conclude that DP-FW guarantees more privacy than DP-SGD. Again, 4.10 shows this way of thinking misleads to wrong conclusion. 4.9 is only one instance of a process, and in4.10 we can observe that DP-SGD has stronger privacy guarantees. What keeps being true is the meaningless of the results for $\varepsilon \in (0, 0.2]$ and the asymptotic convergence for increasing ε. Whereas, it is interesting to observe one thing: The two classes overlap for 20% of their data sets. What happens if this overlapping increases?

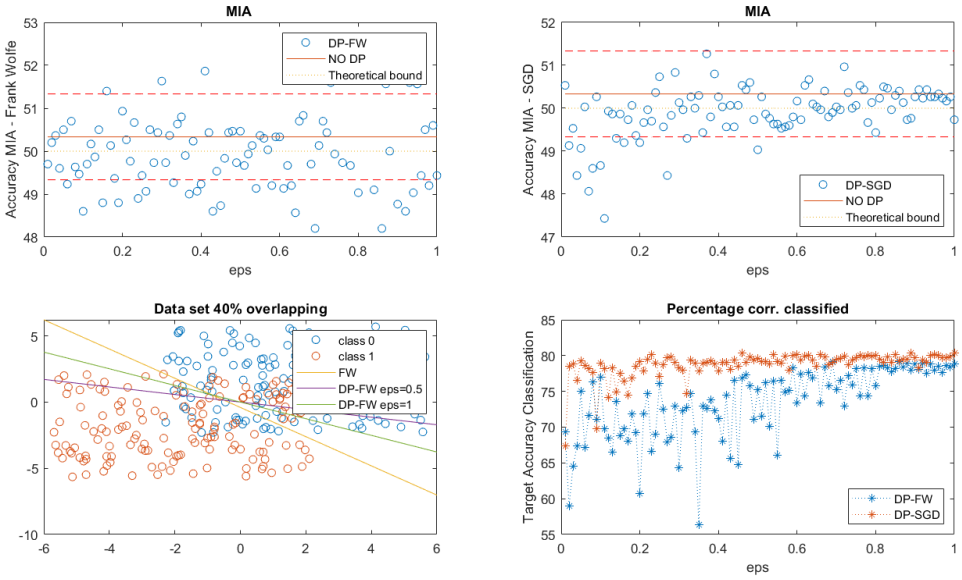


Figure 4.11: Overlap of 40% of the data.

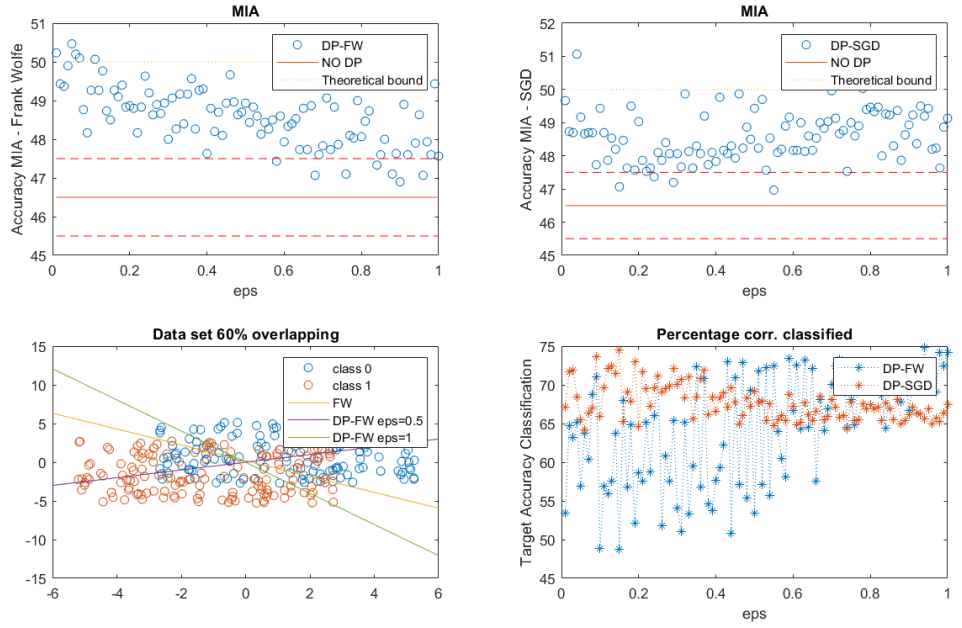


Figure 4.12: Overlap of 60% of the data.

On average, the private learning gets less effective, until it completely fails for 60% overlapping.

Fitting a log-curve for the blue dots helps to understand how the trend would smooth out in each case, obtaining

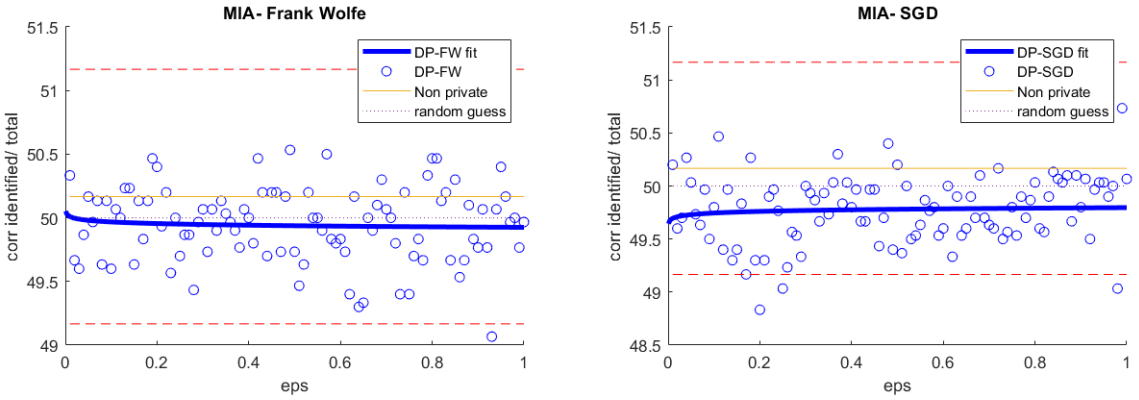


Figure 4.13: Overlap of 20% of the data, with log-fitting of the MIA attacks. The attack has been rerun again.

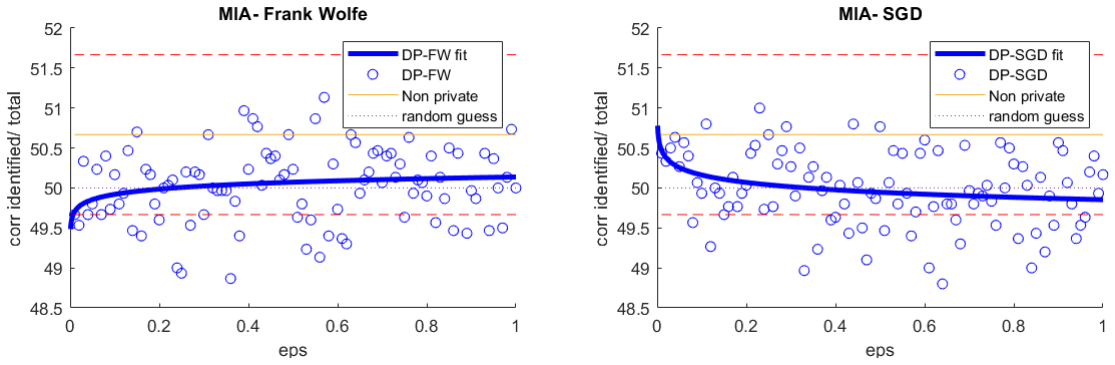


Figure 4.14: Overlap of 40% of the data, with log-fitting of the MIA attacks. The attack has been rerun again.

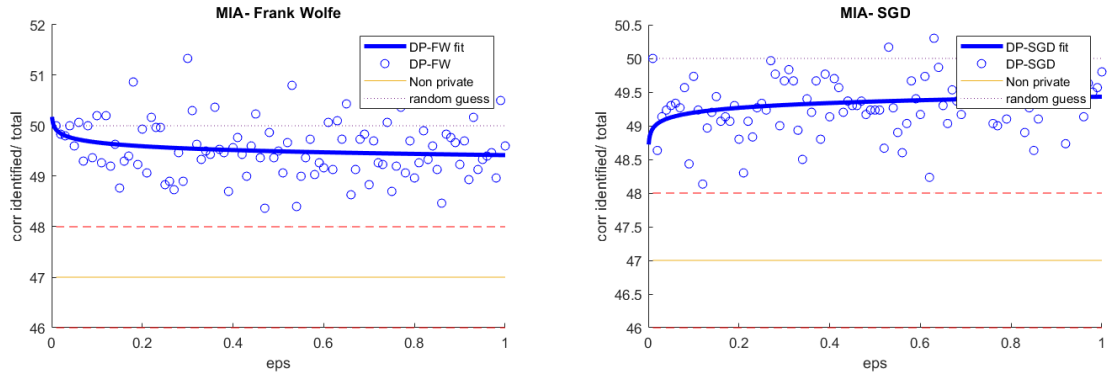


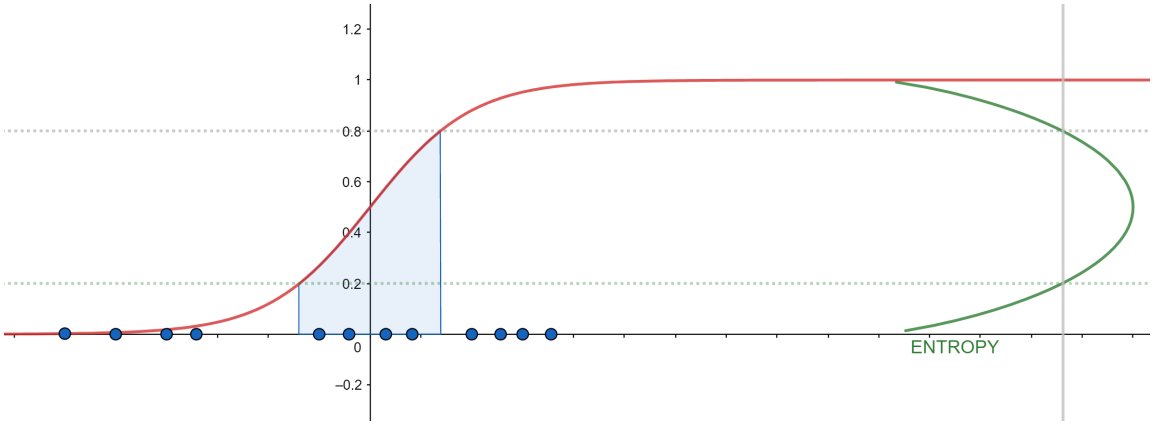
Figure 4.15: Overlap of 60% of the data, with log-fitting of the MIA attacks. The attack has been rerun again.

Note that the fit is straighter 4.13 and it gets more *log-like* when the overlapping

increases. Why do we observe this effect? To answer this question, we need to explain how to choose the τ first.

4.3 Threshold

How is τ chosen? The simulations above are too good because both data and τ have been chosen *ad hoc*. More specifically, an entropy based attack the points with a higher entropy are those close to the margin of the classifier. One can realize it immediately by looking at the 1D case;



On the right it is displayed the rotated graph of the entropy function and the fix threshold $x = \tau$ (vertical line). The margin of this classifier is the *hyperplane* $x = 0$. In other words, whatever is on the right of 0 belongs to the class *one* and whatever on the left is in *zero*. The blue shadow corresponds to the region of points whose entropy is at least τ. It is clear that the smaller τ is, the smaller this region becomes. When we perform an entropy based attack, the strategy consists of declaring members those points whose entropy is small ($H(\phi(x)) \leq \tau$). Those points correspond to the ones close to the margin. Ideally, if the blue show did not contain any points, every single data would be labeled as *Member* during the attack, yielding to a perfect accuracy of 50%. This happens when the two classes are perfectly separable: it is possible to pick a τ sufficiently close to 1 for which there are no points close to the margins. Whereas, the hyperplanes computed through DP-learning can move the classifier a bit, therefore some points falling in the blue region will be classified as *Non Member.*

Keeping all in mind, in the analaysis above, the data set of points was overlapping only for the 20% of the points, i.e only a few of them was in the shadow. While τ was always chosen as

$$\tau = \frac{\tau_{training} + \tau_{test}}{2}$$

with $\tau_{training}$ and τ_{test} the mean estimation $\mathcal{D}_{training}$ and $\mathcal{D}_{test}$ respectively. The motivation behind this choice is that τ_{test} is generally greater than $\tau_{training}$, hence their arithmetic mean separates quite well the two classes. Of course, this choice has the clear drawback that we need to know the data sets in advance, nullifying all efforts done to protect privacy. Even though all these strategies have been taken into account, we barely beat a *random guess* attack. Even with the highest level of privacy - smallest ε used - we did not below 48%.

Setting a slightly higher overlapping of the two classes brings while keeping the same criteria for τ leads to

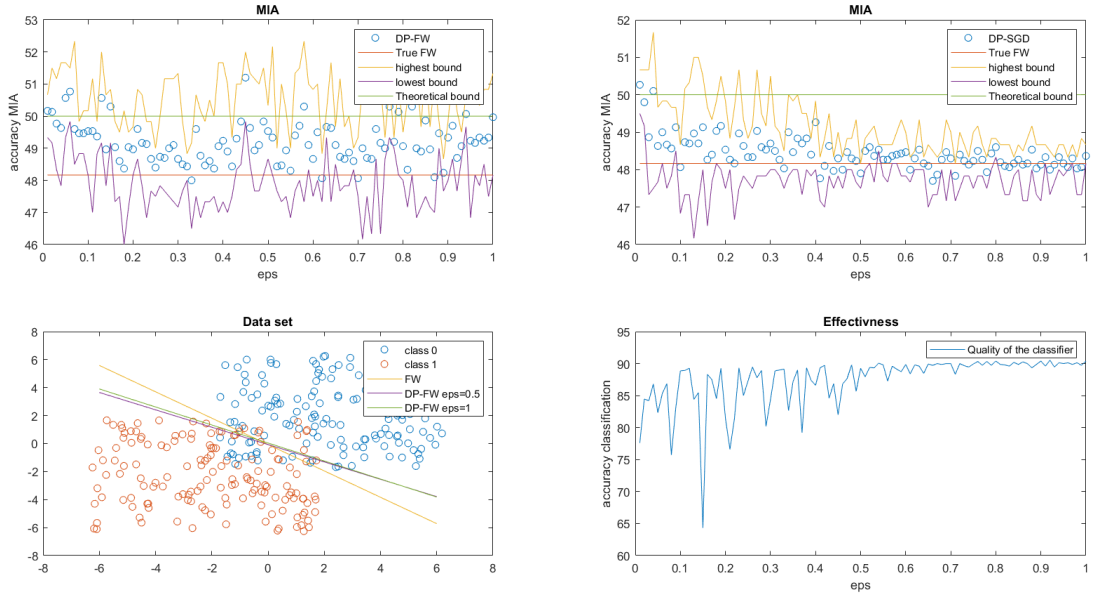


Figure 4.16: Down-Right: Percentage of points correctly classified with DP-FW.

There are no meaningful differences with a random guess. As a result of the overlapping of the two classes (for each data set), the shadow zone around **any** hyperplane will cross the cloud of points and contain approximately the same number of elements from $\mathcal{D}_{training}$ and $\mathcal{D}_{test}$, since they are drawn from the same distribution. In other words, it is an equivalent attack to the random guess.

Quick proof of what we are claiming: setting $\tau = \max_x H(\phi(x))$ leads to

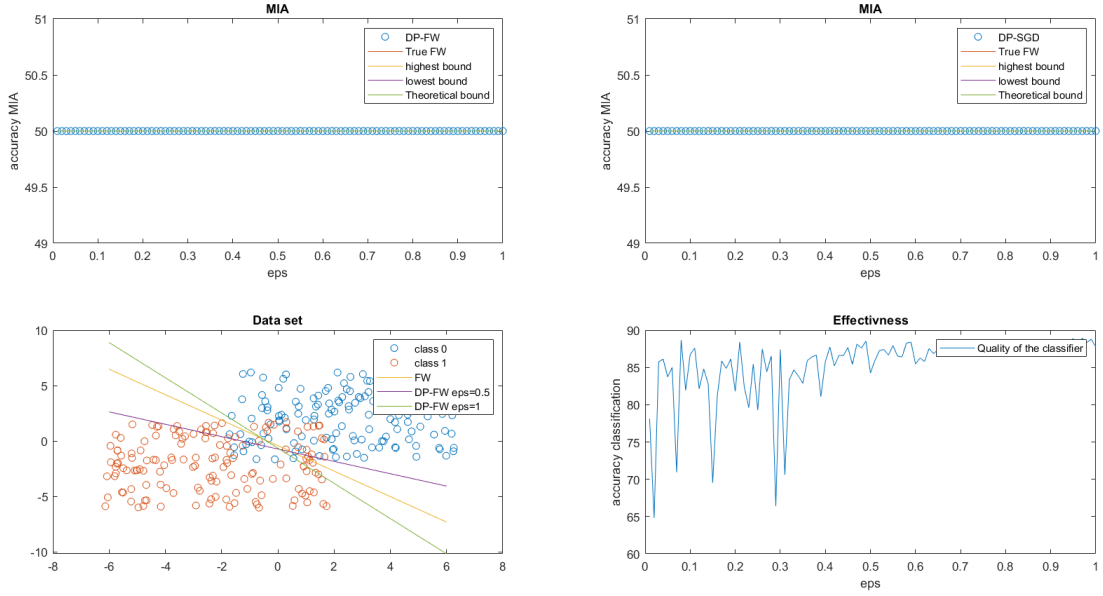


Figure 4.17: Down-Right: Percentage of points correctly classified with DP-FW.

Figure 4.18: $\tau = \max_x H(\phi(x))$

τ makes the shadow as big as the whole data set, thus every point is labeled a member and only for half of them the prediction is correct.

After all that has been discussed, we deduce that the quality of MIA strongly depends on two features: 1) the average **entropy** of $\mathcal{D}$, which might not be available in *black box* attacks, and 2) on the the **separability** of the classes.

Before we conclude this section, it is interesting to investigate the correlation between MIA and the dimension p. In the examples below, we analyse the differences between DP-FW and DP-SGD for increasing values of p.

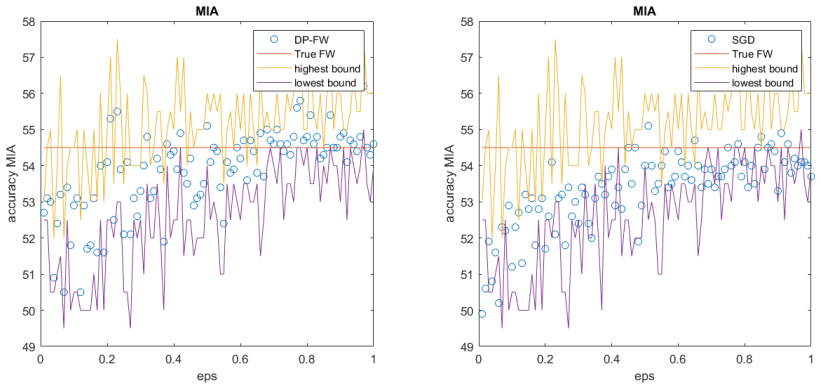


Figure 4.19: $p = 10$, i.e β has 11 entries. We only represented the averages with highest and lowest records, ignoring the analysis of the accuracy.

In general, if p increases so does the excess risk of DP-SGD and DP-FW. Nevertheless, from table 3.2 SGD will be roughly proportional to $\mathcal{O}(\sqrt{p})$. Similarly FW will have to get worse but with a factor of $\mathcal{O}(\log(p))$. In other words, we expect SGD to produce a β_{SGD} whose corresponding hyperplane performs a worse classification than FW. Setting $p = 10000$ we get

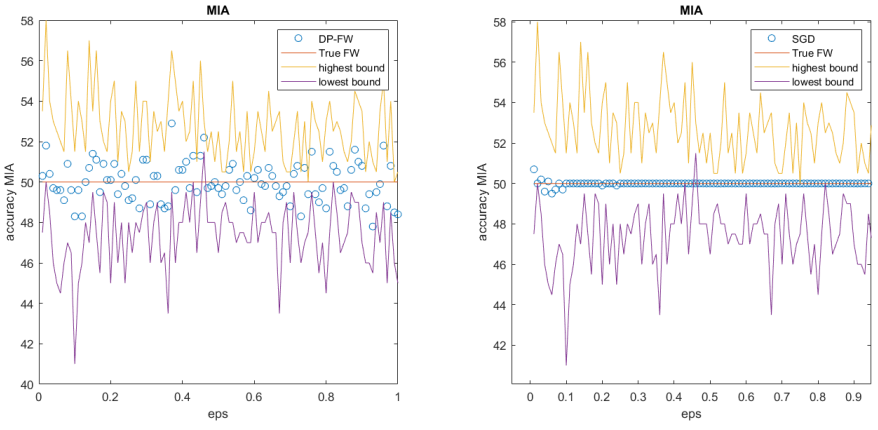


Figure 4.20: Left: Entropy based attack with $DP - FW$ protection. Right: Same attack with $DP - SGD$ learning. The graphs show averages of 10 epochs for every ε. Note that the variance of the difference between the highest and lowest record has significantly increased.

Essentially, the hyperplane produced by the DP-SGD will pass through the points of $\mathcal{D}$ in random directions without a criteria. As a result, this will make both the classification

and the attack almost "nearly" a random guess. This is why on the figure on the right we observe an **average** of 50% for the MIA attacks. Also, the variance of the difference between the highest and lowest record has increased consistently, of almost the same factor: In other words, there exists $0 \leq \gamma \ll 1$ such that Newhigh=50+γ and NewLow=50-γ. The same happens with DP-FW, even though the results are more stable, due to two reasons; FW has better Empirical Risk guarantees for the dependence of the of the dimension; the minimization over the ℓ^1 ball constraints the choices of β. A way to counter balance the worsening of the result is by increasing the same size N. However, the drawback is that the computational time increases too.

Conclusion: under strong assumptions, it is possible to observe the effects of the DP in the context of protection as well as the effectiveness of the MIA attack. However, when these conditions are not met, both attack and defence loose meaning, or it would be better to say that they are not as effective as they should are expected to be. In general, Logistic Regression with two classes is not a good model for applying an entropy attack.

Furthermore, there is space left for improvements. Some experiments were compelled by the computational power available. We had to use a not optimal number of steps, $T = \tilde{\mathcal{O}}(N\varepsilon)$, instead of $T = \tilde{\mathcal{O}}((N\varepsilon)^{2/3})$ (with $N = 1000$ and biggest $\varepsilon = 1$, we barely execute 63 iterations) due to the high computational costs of gradients as function of N. Whenever $N > 10^4$, the time needed per execution gets prohibitive. However, the results would mirror the same conclusions we presented here (they must, as proven in the theory above), although with sharper evidences.

4.4 Multinomial Logistic Regression

In the previous section we analysed how the effect of DP optimization for a logistic regression with two classes. That model is rather convenient to use as toy model, but a model that only deal with two classes rarely appears in real life model. The purpose of this section is to generalize the results of the previous one for an higher number of classes. There is no shortage of algorithms performing multiple classes classification, like KNN or even a simple FNN. However, for the sake of the applicability, the model we are going to present has the advatantage that can be implemented more easily. After all, the goal of the project is to exploits how to produce MIA attacks and defences against the former.

Notation: In this section, the $i-th$column vector is indicated as v_i.

Suppose we are given the following problem: let $\mathcal{D} = \{(x_i, y_i)\} \in \mathbb{R}^p \times \mathbb{R}^k$ a data set of points. We want to classify points $x \in \mathbb{R}^p$ into k different classes. The point x_i belongs only to one of the classes $\{1, \cdots, k\}$. Suppose that x_i belongs to the $j-th$ class. We express this condition with the label y_i as

$$y_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow j-th.$$

Given a new $x \in \mathbb{R}^p$, we deal with the problem of classifying it. The idea is to think of y as a vector of probabilities for each class. The point x is going to assigned to the class whose index maximizes the entries of the label y. In the label above, the $j-th$ entry is 1, i.e there is 100% of probability of being classified as member of the class j.

Let see how to compute these probabilities. Let us define

$$\pi_{ij} = \mathbb{P}(Y = e_j | X = x_i),$$

the probability that x_i belongs to the $j-th$ class. Furthermore, we assume that between two classes there is a relationship [idea explained later] of the form:

$$\log\left(\frac{\pi_{ij}}{\pi_{ik}}\right) = \langle [1, x_i], \beta_j \rangle$$

for a certain $\beta_j \in \mathbb{R}^{p+1}$.

We want to find a close expression for each π_{ij}. We start by exponenting out both sides, obtaining the relation

$$\pi_{ij} = e^{\langle [1,x_i],\beta_j \rangle} \pi_{ik} \quad j = 1, \cdots, k-1.$$

Now we use the fact that for fix i, π_{ij} is a probability distribution

$$1 = \sum_{j=1}^{k} \pi_{ij} = \pi_{ik} + \pi_{ik} \sum_{j=1}^{k-1} e^{\langle [1,x_i],\beta_j \rangle} = \pi_{ik} \left(1 + \sum_{j=1}^{k-1} e^{\langle [1,x_i],\beta_j \rangle}\right).$$

Hence,

$$\pi_{ij} = \begin{cases} \frac{e^{\langle [1,x_i],\beta_j\rangle}}{1+\sum_{j=1}^{k-1} e^{\langle [1,x_i],\beta_j\rangle}}, & j = 1,\cdots,k-1 \\ \\ \frac{1}{1+\sum_{j=1}^{k-1} e^{\langle [1,x_i],\beta_j\rangle}}, & j = k. \end{cases}$$

Finally, it has been obtained the *multinomial distribution*

$$\boxed{\mathbb{P}(Y = y_i | X = x_i) = \pi_{i1}^{y_{1i}} \cdots \pi_{ik}^{y_{ki}} = \Pi_{j=1}^{k} \pi_{ij}^{y_{ji}}}.$$

Observation. The values π_{ij} do depend on $X = x_i$. The expression above is indeed a conditional probability.

Estimation of the parameters $\rightarrow$ *Likelihood method.* The likelihood is

$$\mathcal{L}\left(\begin{array}{c|c|c|c|c} \beta_1^T & \cdots & \beta_k^T & y_1 & \cdots & y_N \end{array} \right) = \Pi_{i=1}^{N} \Pi_{j=1}^{k} \pi_{ij}^{y_{ji}}.$$

As usual, it is better to use the log of the Likelihood

$$l := \log \mathcal{L}(\beta_1^T, \cdots, \beta_k^T) = \sum_{i=1}^{N} \sum_{j=1}^{k} y_{ji} \log(\pi_{ij}).$$

The matrix of parameters $\hat{\beta}^T := [\beta_1^T, \cdots, \beta_k^T]$ of the weights is taken as

$$\hat{\beta} = \underset{[\beta_1,\cdots,\beta_k]\in\mathbb{R}^{p+1\times k}}{\arg\max} \sum_{i=1}^{N} \sum_{j=1}^{k} y_{ji} \log(\pi_{ij})$$

and, since we want it in a minimization form we get

$$\boxed{\hat{\beta} = \underset{[\beta_1,\cdots,\beta_k]\in\mathbb{R}^{p+1\times k}}{\arg\min} \sum_{i=1}^{N} \sum_{j=1}^{k} -y_{ji} \log(\pi_{ij})}.$$

As the minimization in the previous paragraphs, this is a convex minimization problem with a $L-$smooth function.

Notation: Let us group the data in the following structures

$$\hat{\beta} = \underset{k-1\times p+1}{\left[\begin{array}{c} \beta_1 \\ \hline \vdots \\ \hline \beta_{k-1} \end{array}\right]}, \quad \Pi = \underset{N\times k}{\begin{bmatrix} \pi_{11} & \cdots & \pi_{1k} \\ \vdots & & \vdots \\ \pi_{N1} & \cdots & \pi_{Nk} \end{bmatrix}}$$

$$X = \underset{p+1\times N}{\left[\begin{array}{c|c|c} 1 & \cdots & 1 \\ \hline x_{11} & \cdots & x_{1N} \\ \vdots & & \vdots \\ \underbrace{x_{p1}}_{x_1} & \cdots & \underbrace{x_{pN}}_{x_N} \end{array}\right]}, \quad Y = \underset{k\times N}{\left[\begin{array}{c|c|c} y_{11} & \cdots & y_{1N} \\ \vdots & & \vdots \\ \underbrace{y_{k1}}_{y_1} & \cdots & \underbrace{y_{kN}}_{y_N} \end{array}\right]}$$

In order to apply our minimization algorithms we need to compute the gradient of $l(\hat{\beta}, \mathcal{D})$ w.r to $\hat{\beta}$. In order words, we need to compute $\frac{\partial l}{\partial \beta_{st}}$ for every row $s = 1, \cdots, k$ and column $t = 0, \cdots, p$. It can be shown that (see [50])

$$\frac{\partial l}{\partial \beta_{st}} = -X_{\text{row}(t)}(Y^T_{\text{row}(s)} - \Pi_{\text{col}(s)}) \text{ for } s = 1, \cdots, k-1 \quad t = 0, \cdots, p.$$

Now that we have computed the gradient the derivative respect to each component of β, we can apply The Frank Wolfe algorithm. Once we get the outcome $\hat{\beta}_{FW}$ we find the predicted labels as

$$y_{predicted} = \begin{bmatrix} \arg\max_{[k]}[\pi_{11}, \cdots, \pi_{1k}] \\ \vdots \\ \arg\max_{[k]}[\pi_{N1}, \cdots, \pi_{Nk}] \end{bmatrix}.$$

Attack: The implementation of the attack is the one described in [6]. Let us get started by setting the number of classes $k = 3$ in dimension $p = 2$, so that visualizing the data helps us to find out feasible correlation. τ is chosen as the mean entropy of the two data sets. As usual, we will have a type of plot which compares SGD and FW with a band around 50 %, and one plot including the highest and lowest record.

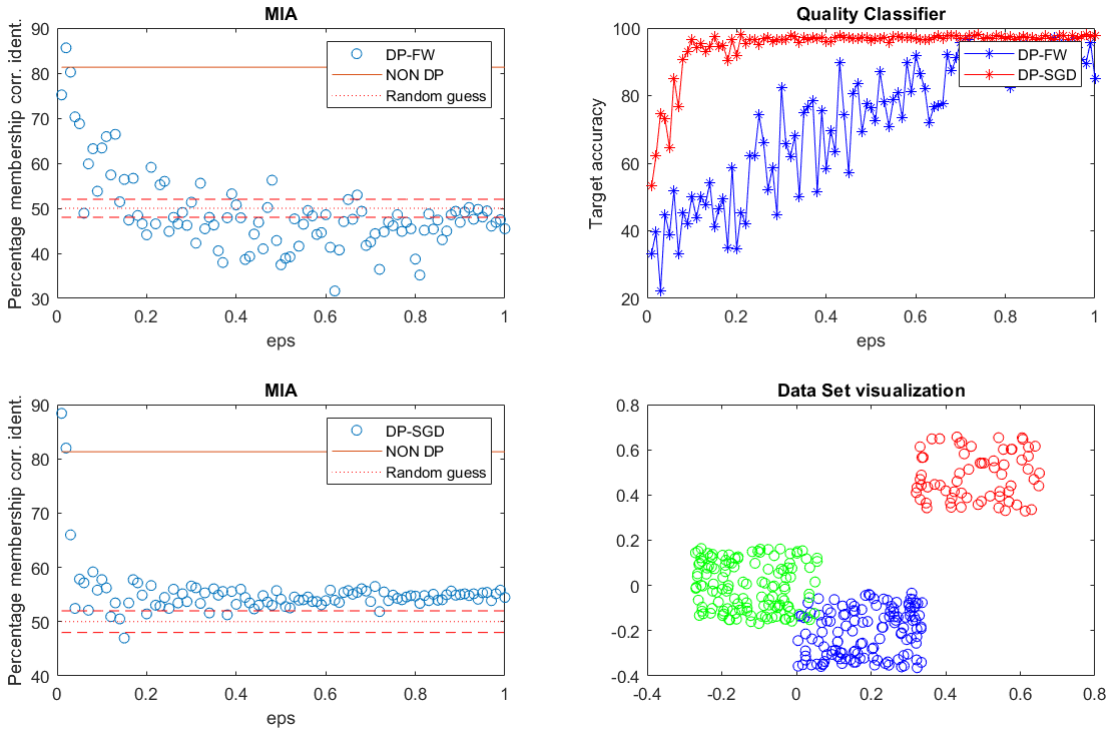


Figure 4.21: $N = 300, \delta = \exp(-\varepsilon^3 N)$ and $\varepsilon \in (0, 1]$.

Unlike the previous models, for multiclass regression the attack seems to make more sense. The *non private* attack a very high accuracy, and at the same attack the differential privacy does lower the accuracy of the attack, dropping out to 55% and 50%. It is not just a lucky outcome. In fact, rerunning the code brings to

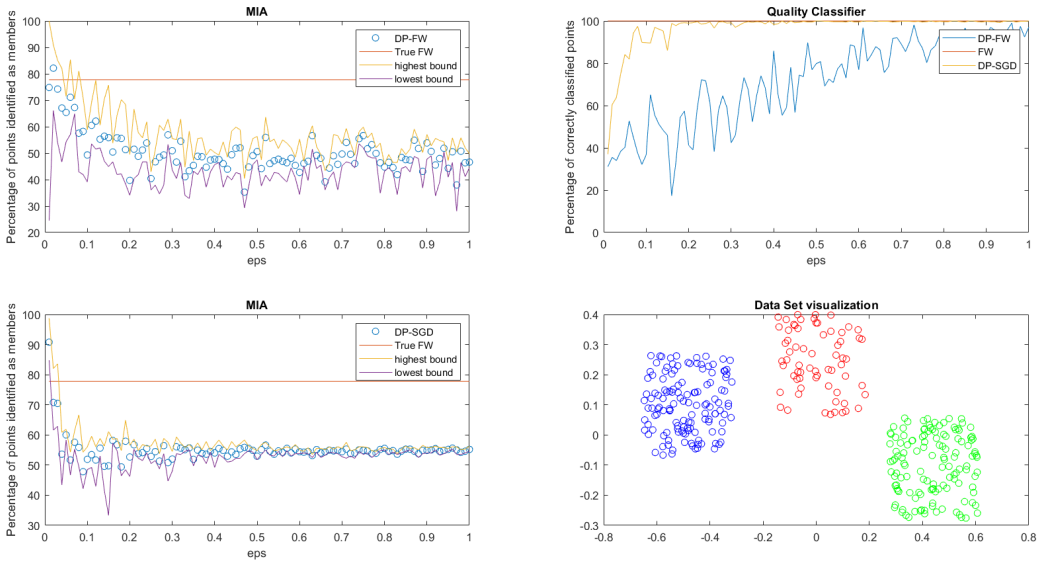


Figure 4.22: Same attack as in 4.21, displaying the highest and lowest record.

Also, observe that SGD stabilizes quicker than FW, and both do not provide consistent results for $\varepsilon \in (0, 0.2]$. We prove that this is the asymptotic behaviour by letting ε vary in a bigger interval, obtaining

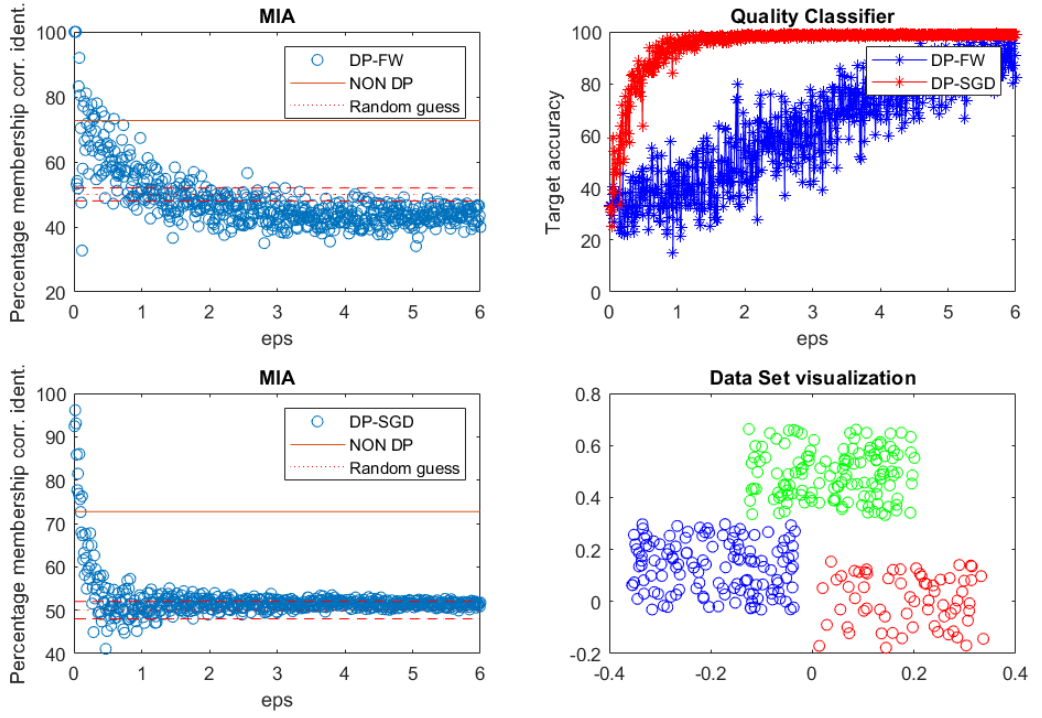


Figure 4.23: $\varepsilon \in (0, 6]$ with same number of points $N = 300$.

Of course, as previously explained for Logistic regression, the results are amazingly nice because we are dealing with an ideal case where the classes do not overlap at all. Both SGD and FW reach a target accuracy on $\mathcal{D}_{test}$ of nearly 100%, which does not happen in

real examples.

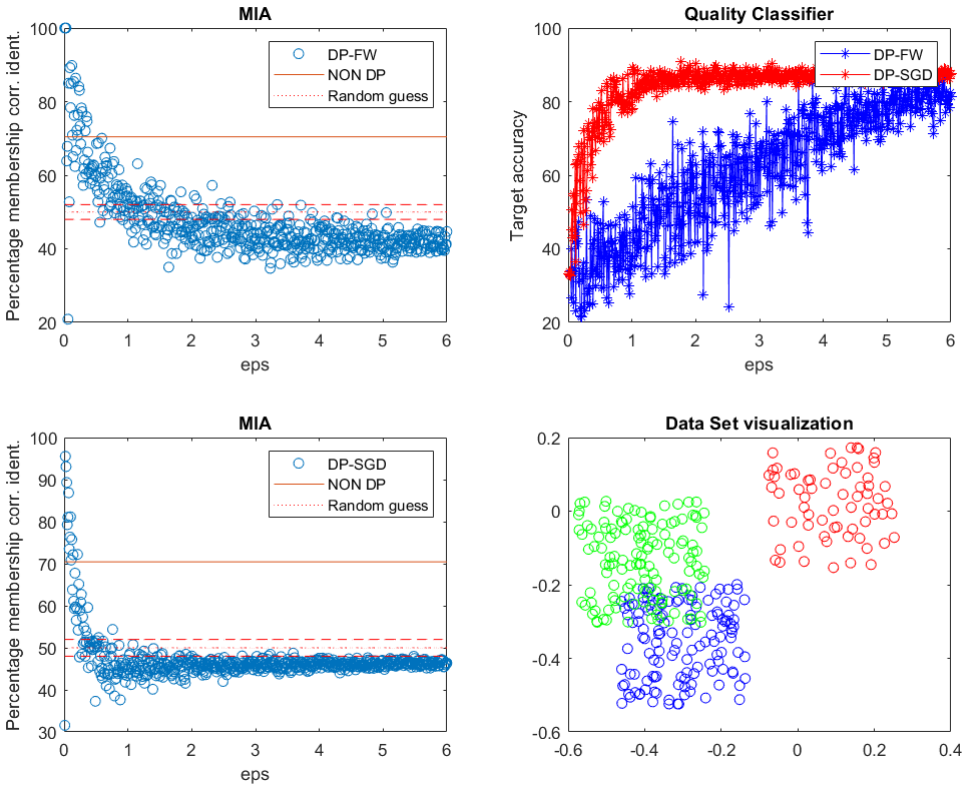


Figure 4.24: $\varepsilon \in (0, 6]$, $N = 300$ and slightly overlapping classes.

A slight overlap of two classes results in a drop of 10% on both DP and NOT DP attack, as well as the target accuracy.

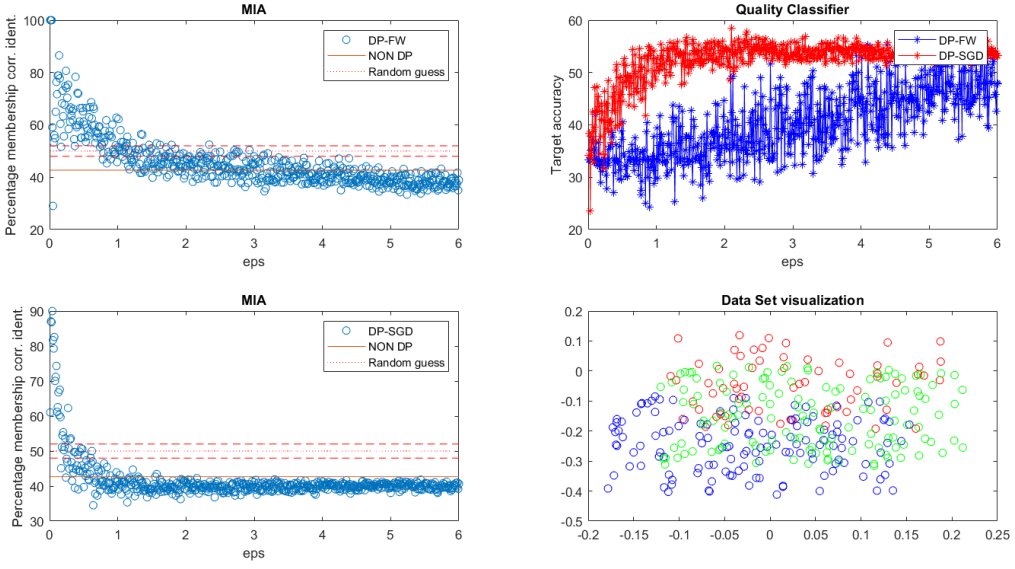


Figure 4.25: $\varepsilon \in (0, 6]$, $N = 300$ and completely overlapping classes.

Conclusions and future work

Among the three models analyzed, the multiclass logistic regression seem to show more more meaningful results in terms of privacy protection. This makes us believe that using this model for real data sets will lead to similar results. For instance, for the MNIST data set we can imagine that every digit is a point in $\mathbb{R}^{m \times n}$ with a sparse structure due to the color map in black and white of the image, like in the figure:

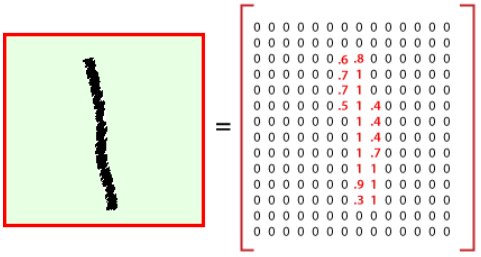


Figure 4.26: A sample from MNST data set. Every image can be thought a vector with a sparse structure.

Then, the whole data sets can be thought as a collection of 10 classes spread around the vertices of the ℓ^1 ball in $\mathbb{R}^{m \times n}$. Also, the sparseness should improve the performances of our FW implementation, regardless from the high dimension of these data. As we saw, a drawback of FW is that uses the whole batch of points. 4 is designed to overcome this issue, and by the privacy amplification lemma, we should also expect the same *speed* in leveraging out as we observed in the SGD. In SGD, for every iteration, one point is sampled from a data set that needs to be protected, and then keep running the DP algorithm. This amplifies the privacy and it is the reason why in the simulations SGD has almost always beated FW. It is also quicker in terms of computation costs because FW needs to compute $\mathcal{O}(N)$ gradients per iteration, while for SGD it is done in constant time $\mathcal{O}(1)$. 4 is expected to perform better than both SGD and FW: we should get the computational speed of SGD and higher privacy protection by lemma 21.

From what discussed in this Chapter, **DP does protect** against MIA attacks even though DP has been tested on simple model and ideal situations. Especially, for logistic regression when the data sets are truly linearly separable, and the constant τ is chosen properly, we can clearly observe a stable and coherent behaviour of the DP defence in accordance with the theoretical results.

Chapter 5

Synthetic Data Generation

In the previous chapters we focused on how to privately train a statistical model and measuring the effectiveness of privacy when a Membership inference attack is performed on it.

In this chapter we switch point of view. Suppose we have a data set containing sensitive information that we want to protect. Instead of coming up with a differentially private algorithm which realises noisy data, we shall try to construct a new dataset that preserves the same statistical properties. This construction will be differentially private. The advantage consists in the fact that the new data set can be used *naked*, it does not need a mechanism that injects noise.

5.1 Min Max formulation

Let $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ be the space of the data or records. We call *data* i the $i-th$ element of this space, and we will indicate it either as z_i or d_i. $\mathcal{X}_i$ can be any kind of set. As an example, d_i = [name, surname, high, weight]. The first two entries are strings, whereas the last two are values in $\mathbb{Q}$. Further, let $\tilde{d} = |\mathcal{X}|$.

A dataset $\mathcal{D}$ is a set containing data d_i of N different people, $\mathcal{D} = [d_1, \cdots, d_N]$ with $d_i \in \mathcal{X}$. In this context, a data set is not thought as a mathematical set but rather as a multiset, i.e we allow $\mathcal{D}$ to have repetitions of the same record. Continuing on the line of our example above, $\mathcal{D}$ might be containing the data of two identical twins with the same name. We now pose the problem of answering statistical queries over such data set. We introduce the notion of statistical query over a dataset first.

Definition 30 (Empirical)**.** Let $\mathcal{D}$ be a datset and $\mathcal{J} : \mathcal{X} \longrightarrow [0, 1]$ a predicate over $\mathcal{X}$. We define the statistical query q as a function *of* (not on) the dataset $\mathcal{D}$

$$q(\mathcal{D}) = \sum_{x \in \mathcal{X}} \mathcal{J}(x) p_{\mathcal{D}}(x)$$

and $p_{\mathcal{D}}(x)$ is the normalized frequency of x in $\mathcal{D}$.

Note that the sum is taken over $x \in \mathcal{X}$. In other words, we think of a dataset as its empirical distribution: for every $x \in \mathcal{X}$ we count how many times it appears in $\mathcal{D}$ and then we divide by the number of total people N. In other words, a dataset can be thought as probability distribution $p_{\mathcal{D}}$ over $\mathcal{X}$.

This definition can be extended, thinking of a dataset as realization of a sample from a distribution. Such probability distribution representing $\mathcal{D}$ would be a vector in the unitary simplex of $\mathbb{R}^N$. This observation motivates the following extension

Definition 31 (Population)**.** Let us denote $\Delta_{\tilde{d}} = \{p \in \mathbb{R}^{\tilde{d}} | \quad \sum_i^{\tilde{d}} p_i = 1 \quad p_i \geq 0\}$. A statistical query $\mathbf{q}$ is a function

$$\begin{aligned} \mathbf{q} :& \Delta_{\tilde{d}} \longrightarrow [0,1] \\ & \mathcal{P} \longrightarrow \langle \mathbf{q}, \mathcal{P} \rangle = \mathbb{E}_{\mathbf{z} \sim \mathcal{P}}[\mathcal{J}(\mathbf{z})] \end{aligned}$$

where $\mathbf{q} = [\mathcal{J}(z_1), \cdots, \mathcal{J}(z_{\tilde{d}})]$ and $\mathbf{z}$ a random variable taking values in $\mathcal{X}$ and distributed as $\mathcal{P}$. We can think of $\mathbf{q}$ as a linear function over the unitary symplex.

Let Q be a set of queries $\mathbf{q}$ that we would like to answer. Answering a query means that we have to provide a value for $\langle \mathbf{q}, \mathcal{P} \rangle$. Of course, in a not private setting, it would be enough to compute this *dot* product algebraically, since the data set $\mathcal{P} \in \Delta_{\tilde{d}}$ would be fully available. Of course, in a private setting, such information is not public, thus the need to compute $\langle \mathbf{q}, \mathcal{P} \rangle$ arises.

To this purpose, the idea would be to privately *create* a new data set which preserves the same statistical properties of the data set we want to keep private, i.e a new probability distribution p over $\mathcal{D}$. We will refer to such created data set as **synthetic data set**. Next, we fully release the synthetic data and we work with it as it was not private.

Mathematically, we have the following definition

Definition 32 (synthetic data)**.** Given a private probability distribution $p_{\mathcal{D}}$ representing a private data set $\mathcal{D}$, and a set of statistical queries Q, the problem of finding a synthetic probability distribution p, such that the maximum error over all the queries is minimized, namely $\max_{q \in Q} \langle q, p_{\mathcal{D}} - p \rangle$, is called *private synthetic data release. p is called synthetic data set or distribution.*

In practice, we want to find the $p^\star$ that minimizes

$$\min_{p \in \Delta_{\tilde{d}}} \max_{q \in Q} |\langle q, p_{\mathcal{D}} - p \rangle| \qquad (P) \ .$$

Observation. Note that in a not private setting, (P) admits solution $p = p_{\mathcal{D}}$. The privacy constraint ensures that $p \neq p_{\mathcal{D}}$ - otherwise the private data set would be realised.

Remark 2. since the problem is linear, $Q = conv\{q_1, \cdots, q_k\}$, we take the polyhedron as the convex hull of our finite set of queries. The max will be achieved at one of the vertices.

Remark 3. We do not have to make confusion between empirical distribution of the data set and population distribution.

As we said, $d_i \in \mathcal{D}$ is an element of $\mathcal{X}$ too. Suppose our dataset $\mathcal{D}$ is made of elements $(d_j)_{j=1}^N$ sampled *i.i.d* from a distribution $\mathcal{P}$ over $\mathcal{X}$, which is **unknown** to us. According to our definition of query q, we want to compute

$$q(\mathcal{D}) = \mathbb{E}_{Z \sim \mathcal{P}}[\mathcal{J}(Z)] \ .$$

To do so, we would need the distribution $\mathcal{P}$ and integrate over it, which is not available. However, we do have access to a sample $\mathcal{D} = \{d_1, \cdots, d_N\}$ drawn from $\mathcal{P}$. When this is the case, using the law of large numbers, the sample mean converges in probability to the expectation of its distribution, namely

$$\frac{\mathcal{J}(d_1) + \cdots + \mathcal{J}(d_N)}{N} \longrightarrow \mathbb{E}_{Z \sim \mathcal{P}}[\mathcal{J}(Z)] \ .$$

Further, note that

$$\begin{aligned}
\frac{\mathcal{J}(d_1)+\cdots+\mathcal{J}(d_N)}{N} &= \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}}\mathcal{J}(x) \\
&= \frac{1}{N}\sum_{x\in\mathcal{D}}\mathcal{J}(x) + 0 \\
&= \sum_{x\in\mathcal{D}\cap\mathcal{X}}\mathcal{J}(x)p_{\mathcal{D}}(x) + \sum_{x\notin\mathcal{D}\cap\mathcal{X}}\mathcal{J}(x)\underbrace{p_{\mathcal{D}}(x)}_{=0} \quad p_{\mathcal{D}}(x) = \frac{|\text{frequency of } x \text{ in } \mathcal{D}|}{N} \\
&= \sum_{x\in\mathcal{X}}\mathcal{J}(x)p_{\mathcal{D}}(x) \quad \text{(def of empirical query)} \\
&= \sum_{x\in\mathcal{X}}\mathcal{J}(x)p_{\mathcal{D}}(x) \xrightarrow{\mathbb{P}} \mathbb{E}_{Z\sim\mathcal{P}}[\mathcal{J}(Z)] \ .
\end{aligned}$$

In conclusion, we work with empirical distribution $p_{\mathcal{D}}$ and we try to find a synthetic data w.r.t this $p_{\mathcal{D}}$. This also shows that

$$q(\mathcal{D}) = \langle q, p_{\mathcal{D}}\rangle = \sum_{x\in\mathcal{X}}\mathcal{J}(x)p_{\mathcal{D}}(x) = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}}\mathcal{J}(x)$$

and,

$$\underbrace{\langle q, p_{\mathcal{D}}\rangle}_{\text{depends on } N} = \mathbb{E}_{Z\sim p_{\mathcal{D}}}[\mathcal{J}(Z)] \xrightarrow[N\to\infty]{\mathbb{P}} \mathbb{E}_{Z\sim\mathcal{P}}[\mathcal{J}(Z)] = \langle q, \mathcal{P}\rangle \ .$$

Related work

Private synthetic data for query release has been studied in a long series of works (BLR11),(RR09). The largest class of available algorithms for private synthetic data for query release are based under the framework of the private multiplicative weights algorithm (VTB$^+$20). They called this method Multiplicative Weights Exponential Mechanism (MWE). MWEM iterative uses a differentially private selection mechanism, like the exponential mechanism, to pick a query with high error q_t, and then it uses this query to compute an approximate distribution p_t (solution of (P)) by solving an optimization problem. Many variants are known, depending on how this optimization problem is solved and regularized (VTB$^+$20), (ABK$^+$21). These methods compute an l-approximate solution $p^\star$ such that $\langle q, p_{\mathcal{D}} - p^\star\rangle \leq l$ for all queries $q \in Q$. The most known algorithms and their corresponding quality of approximation are listed in Tab. 5.1.

Our approach to design an algorithm for answering linear queries is however different. Firstly, the Multiplicative Weights framework requires a finite number of statistical queries in Q, whereas our method will run over a Q that is a polyhedron given by the convex hull of those finite queries. Secondly, their running time suffers the highly dimentionality of the data, while the approach we present inherits the convergence rates from the corresponding method to optimize the problem. Also, under the MW frame work, (GAH$^+$14) uses an idea similar to ours. By looking at the problem as a zero sum game, they find an $l-$approximate solution to the Nash Game; this leads to [theorem 4.6] an algorithm they call DualQuery, which finds a synthetic database that answers all queries in Q within additive error α with probability $1-\beta$.

Algorithm	l
MWEM	$\mathcal{O}\left(\frac{d^{1/4}\log^{1/2}\lvert Q\rvert \log^{1/2}(1/\delta)}{(\varepsilon N)^{1/2}}\right)$
DualQuery	$\mathcal{O}\left(\frac{d^{1/6}\log^{1/2}\lvert Q\rvert \log^{1/6}(1/\delta)}{(\varepsilon N)^{1/3}}\right)$
sepFEM	$\mathcal{O}\left(\frac{d^{3/4}\log^{1/2}\lvert Q\rvert \log^{1/2}(1/\delta)}{(\varepsilon N)^{1/2}}\right)$
DQRS	$\mathcal{O}\left(\frac{d^{1/5}\log^{3/5}\lvert Q\rvert \log^{1/5}(1/\delta)}{(\varepsilon N)^{2/5}}\right)$
This work	$\mathcal{O}\left(\frac{d^{1/5}\log^{2/5}\lvert Q\rvert \log^{1/5}(1/\delta)}{(\varepsilon N)^{2/5}}\right)$

Table 5.1: Q set of queries, Q_1 diameter in norm 1 if Q is a polyhedron, $|Q|$ number of queries otherwise. d dimension of the data, (ε, δ) are privacy parameters and N is the number of points.

5.2 Stochastic Optimization for Synthetic Data release

As explained in remark 3, we are always given a dataset $\mathcal{D}$ whose data are drawn from some unknown distribution $\mathcal{P}$. As of now, we shall focus on the empirical version of the problem, finding the private p that is close to $p_{\mathcal{D}}$. This latter represents our sample dataset. Let us get started with some assumptions.

Since a data set usually does not contain many repetitions of a same record, it is reasonable to speculate that the entropy of $p_{\mathcal{D}}$ is quite large (that the entropy of a distribution is maximum when the distribution is uniform). Therefore, one possibility is forcing the entropy of p to be as high as possible. In other words, we could add a entropy based regularizing term to our primal (P) and find p such that

$$\max_{q\in Q}\left(\langle q, p_{\mathcal{D}} - p\rangle + \alpha H(p)\right)$$

is as small as possible, where $p_{\mathcal{D}}$ is the initial empirical distribution of the dataset. Of course, $\alpha > 0$ should be small for promoting higher entropy. In conclusion, we have to solve the following

$$\boxed{\min_{p\in\Delta_{\tilde{d}}}\max_{q\in Q}\left(\langle q, p_{\mathcal{D}} - p\rangle + \alpha H(p)\right)\ .} \tag{5.1}$$

Remark 4. The absolute values have been removed because we are assuming that Q is symmetric, i.e if $q \in Q$ then $-q \in Q$.

Observation. 5.1 can be written differently, since strong duality holds

$$\begin{aligned}
\min_{p\in\Delta_{\tilde{d}}}\max_{q\in Q}\left(\langle q, p_{\mathcal{D}} - p\rangle + \alpha H(p)\right) &\iff \max_{q\in Q}\min_{p\in\Delta_{\tilde{d}}}\left(\langle q, p_{\mathcal{D}} - p\rangle + \alpha H(p)\right)\\
&\iff \max_{q\in Q}\left[\langle q, p_{\mathcal{D}}\rangle + \min_{p\in\Delta_{\tilde{d}}} -\alpha\left(\langle\frac{q}{\alpha}, p\rangle - H(p)\right)\right]\\
&\iff \max_{q\in Q}\left[\langle q, p_{\mathcal{D}}\rangle - \alpha\max_{p\in\Delta_{\tilde{d}}}\left(\langle\frac{q}{\alpha}, p\rangle - H(p)\right)\right]\\
&\iff \max_{q\in Q}\left[\langle q, p_{\mathcal{D}}\rangle - \alpha H^{\star}\left(\frac{q}{\alpha}\right)\right].
\end{aligned}$$

$H^\star$ denotes the Fenchel's conjugate of H. In conclusion, solving 5.1 boils down to maximizing first

$$q_\alpha^\star := \arg\max_{q \in Q} \left[\langle q, p_{\mathcal{D}} \rangle - \alpha H^\star \left(\frac{q}{\alpha} \right) \right] . \tag{5.2}$$

Proposition 33. *Let* $\phi_\alpha(p) := \phi(p) + \alpha H(p)$ *where* $\phi(p) = \max_{q \in Q} \langle q, p_D - p \rangle$ *and* $\psi_\alpha(q) := \langle q, p_{\mathcal{D}} \rangle - \alpha H^\star \left(\frac{q}{\alpha} \right)$. *Then,*

$$\boxed{\min_{p \in \Delta_{\tilde{d}}} \phi_\alpha(p) = \phi(p) + \alpha H(p) \qquad (P_\alpha)} \tag{5.3}$$

and

$$\boxed{\max_{q \in Q} \psi_\alpha(q) := \langle q, p_{\mathcal{D}} \rangle - \alpha H^\star \left(\frac{q}{\alpha} \right) \qquad (D_\alpha)} \tag{5.4}$$

are primal and dual with respect to each other and strong duality holds.

Since Q is a polyhedron, a convenient approach is to privately solve 5.2 by using Frank Wolfe. Adapting Frank Wolfe to 5.4 will be the goal of the next section.

Remark 5. Solving the regularized problem will result in an $l-$approximate solution, whose upper bound will be analyzed in the upcoming section.

5.3 Frank Wolfe for dual formulation of synthetic data

Before we start analyzing the algorithm 3 on 5.4, we need to recall some definitions and facts that will be fundamental for the rest of the analysis.

Recall that H is the *negative* entropy $H(x_1, \cdots, x_N) = \sum_i x_i \log(x_i)$.

Proposition 34. *Let* $||\cdot||$ *be* ***any*** *norm.* $f : \underbrace{E}_{cvx} \longrightarrow \mathbb{R} \in \mathcal{C}^2(E)$.

$$f \text{ is } k - \text{strongly convex w.r.t } ||\cdot|| \iff \langle \nabla^2 f(x)h, h \rangle \geq k||h||^2 \quad \forall h, x \in E.$$

Proposition 35. *Let* $||\cdot||$ *be* ***any*** *norm.* $f : \underbrace{E}_{cvx} \longrightarrow \mathbb{R} \in \mathcal{C}^2(E)$.

$$f \text{ is } k - \text{smooth w.r.t } ||\cdot|| \iff \langle \nabla^2 f(x)h, h \rangle \leq k||h||^2 \quad \forall h, x \in E.$$

For f and its Fenchel's conjugate $f^\star$, strongly convex and smoothness of the gradient are dual properties:

Theorem 36 (Zalinescu: 1983)**.** *Let* $||\cdot||$ *be a norm and* $k > 0$. *Then*

1) *If* $f : E \to \mathbb{R}$ *is convex and* $1/k-$ *smooth w.r.t* $||\cdot||$, *then* $f^\star$ *is* $k-$*strongly convex w.r.t* $||\cdot||_\star$.

2) *If* $f : E \to \mathbb{R}$ *is* $k-$*strongly convex w.r.t* $||\cdot||$, *then* $f^\star$ *is* $1/k-$*smooth w.r.t* $||\cdot||_\star$.

Lemma 37. *The Negative Entropy* $H(x_1, \cdots, x_d) = \sum_i x_i \log(x_i)$ *is* $1-$*strongly convex w.r.t the* $||\cdot||_\infty$.

Proof. The Hessian of H is

$$\nabla^2 H(x) = \begin{bmatrix} \frac{1}{x_1} & & \\ & \ddots & \\ & & \frac{1}{x_d} \end{bmatrix} \quad \forall x \in \Delta_{\tilde{d}}$$

therefore,

$$\langle \nabla^2 H(x), h \rangle = \sum_{i=1}^{d} \frac{1}{x_i} h_i^2 \geq \frac{1}{x_s} ||h||_\infty^2 \geq \underbrace{1}_{k} ||h||_\infty^2$$

where $s = \arg\max_i |h_i|$ and $x_i \in (0, 1)$.

□

By Relinescu's Theorem 54, it follows that $H^\star$ is $1 = \frac{1}{k}-$ smooth w.r.t $(||\cdot||_\infty)_\star = ||\cdot||_1$. Thus, $||\nabla H^\star(q_1) - \nabla H^\star(q_2)||_\infty \leq ||q_1 - q_2||_1$.

Frank Wolfe steps

When applying Frank Wolfe 3 on 5.2, with objective $\langle q, p_{\mathcal{D}} \rangle - \alpha H^\star \left(\frac{q}{\alpha}\right)$ we perform three steps

1. Compute the gradient $\nabla_q \psi_\alpha(q_t)$
2. $s_t = \arg\max_{s \in \text{Vertex}(Q)} \left(\langle \nabla_q \psi_\alpha(q_t), s \rangle + Lap(\lambda)\right)$
3. $q_{t+1} = q_t + \gamma_t(s_t - q_t)$

Privacy Analysis

To study the privacy of the algorithm, we need to know what is the sensitivity of the the function $\psi_\alpha(q)$ (which implicitly depends on the dataset). We are applying a Report-Noisy-Max Mechanism in step 2. above.

As we know, we only have to bound in norm $||\cdot||_1$ the sensitivity of $\mathcal{A}_s(p_{\mathcal{D}}) := \langle \nabla_q \psi_\alpha(q_t, p_D), s \rangle$

$$\Delta(\mathcal{A}_s) = \sup_{p_{\mathcal{D}} \sim p_{\mathcal{D}'}} ||p_{\mathcal{D}} - \nabla H^\star \left(\frac{q}{\alpha}\right) - p_{\mathcal{D}'} + \nabla H^\star \left(\frac{q}{\alpha}\right)||_1 = \sup_{p_{\mathcal{D}} \sim p_{\mathcal{D}'}} ||p_D - p_{\mathcal{D}'}||_1 = \frac{1}{N} .$$

The sensitivity of $\mathcal{A}_s$ is bounded in statistical sense by $\frac{1}{N}$ for any vertex s. As a results, if we want step 2. to be -DP, we need to use Laplacian noise tuned by $\lambda = \frac{\Delta_s}{\tilde{\varepsilon}} = \frac{1}{N\tilde{\varepsilon}}$.

Since we repeat DP-FW T times, by using 20 the whole algorithm will be

$$\varepsilon = 4\tilde{\varepsilon}\sqrt{2T \log\left(\frac{1}{\delta}\right)}, \delta - DP .$$

Thus λ will be set as

$$\lambda = \frac{4\sqrt{2T \log\left(\frac{1}{\delta}\right)}}{N\varepsilon} .$$

Convergence Analysis

In order to analyze the convergence of FW applied on 5.2, we have to be sure that $\psi_\alpha(q)$ is $L-$smooth w.r.t some norm. It has been shown above that $H^\star$ is $1-$smooth w.r.t the $||\cdot||_1$ norm.

Hence,

$$||\nabla\psi_\alpha(q_1)-\nabla\psi_\alpha(q_2)||_\infty = ||\nabla H^\star\left(\frac{q_1}{\alpha}\right)-\nabla H^\star\left(\frac{q_2}{\alpha}\right)||_\infty \underbrace{\leq}_{1-\text{smooth}} ||\frac{q_1}{\alpha}-\frac{q_2}{\alpha}||_1 = \frac{1}{\alpha}||q_1-q_2||_1 \ .$$

As a result, we can make use of the descent lemma with $||\cdot||_1$ and repeat the same proof of the previous chapter.

Recall that by setting $M_1 = \lambda\log(k)$ and $M_2 = L/2 \cdot Q_1$ [proof of 3], we found the recursive relation

$$R(q_{t+1}) \leq (1-\gamma_t)R(q_t) + \gamma_t M_1 + \gamma_r^2 M_2 \ .$$

It can be shown 49 that

$$R(q_t) \leq M_1 + a_t M_2 \ ,$$

where $a_t \sim \frac{1}{t}$.

Substituting M_1 and M_2 for T iterations,

$$M_1 + \frac{1}{T}M_2 = \frac{4\log(k)\sqrt{2T\log\left(\frac{1}{\delta}\right)}}{N\varepsilon} + \frac{Q_1}{\alpha T} \ .$$

Here, we used the fact that the $L-$smoothness constant is $\frac{1}{\alpha}$. Moreover, the diameter $Q_1 = 2$.

The only thing left to do is to optimize over T to reach the lowest value for $M_1 + \frac{1}{T}M_2$.

The function $g(T) = \frac{4\log(k)\sqrt{2T\log\left(\frac{1}{\delta}\right)}}{N\varepsilon} + \frac{Q_1}{\alpha T}$ admits one global minimum that we can find by setting the first derivative equal zero. In fact, for small and large values of T, g goes to $+\infty$ and the derivative has only one stationary point. This is enough for concluding that g has a unique minimizer. Formally, by setting $g'(T) = 0$, the optimal number of iterations is

$$T = \frac{4(N\varepsilon)^{2/3}}{\alpha^{\frac{2}{3}}(\log(k))^{2/3}\left(\log\frac{1}{\delta}\right)^{2/3}} \ .$$

Substituting the value of T in the upper bound and λ found before, gives the following

Proposition 38. *Let* $\max_{q\in Q}\left[\langle q, p_{\mathcal{D}}\rangle - \alpha H^\star\left(\frac{q}{\alpha}\right)\right] = \psi_\alpha(q)$*, where* Q *is a polyhedron with* k *vertices in* $\mathbb{R}^{\tilde{d}}$*. Let* ε, δ *be privacy parameters. Running DP-Frank Wolfe 3 on this problem with* $T = \frac{4(N\varepsilon)^{2/3}}{\alpha^{\frac{2}{3}}(\log(k))^{2/3}\left(\log\frac{1}{\delta}\right)^{2/3}}$ *iterations and Laplacian noise* $\lambda = \frac{\sqrt{128}\left[\log\left(\frac{1}{\delta}\right)\right]^{1/6}}{(N\varepsilon)^{2/3}\alpha^{1/3}[\log(k)]^{1/3}}$ *yields*

$$\mathbb{E}[\psi_\alpha(q_T) - \psi_\alpha(q^\star)] \leq \mathcal{O}\left(\frac{[\log(k)]^{2/3}\left[\log\left(\frac{1}{\delta}\right)\right]^{1/3}}{\alpha^{2/3}(N\varepsilon)^{2/3}}\right) \ .$$

The algorithm produces an output q_T *that is* ε, δ*-DP.*

Note that the number of iterations explodes when we take a low α. α small is desirable for our regularization: this promotes an higher entropy in our dataset. Thus, α must also be tuned.

In other words, introducing a regularization parameter amplifies our upper bound of a factor $1/\alpha^{2/3}$. In the upcoming section we shall present a discussion on how to tune α to get a small primal gap.

5.4 Effect of regularization

Definition 39. Let $\min_p \phi(p)$ and $\max_q \psi(q)$. We define the

$$\mathbf{Gap}_{(P)}(p) = \phi(p) - \phi^\star$$

$$\mathbf{Gap}_{(D)}(q) = \psi^\star - \psi(q).$$

The **Gap** measures how much a feasible point differs from its optimal. Of course, **Gap**(q)=0 if and only if $p = p^\star$. Our goal is to analyze how much the solution of of (P_α) differs from the solution of (P). In other words, we want to compute an upper bound for the $l-$approximation $\max\langle q, p_{\mathcal{D}} - p_\alpha\rangle \leq l$. A list of such upper bounds can be found in 5.1. In practice, instead of computing the exact solution of the primal (P), the problem have been moved and transformed a few times

$$(P) \rightsquigarrow (P_\alpha) \rightsquigarrow (D_\alpha)\ .$$

Thus, the procedure goes backwards: we solve the (D_α) (5.4) obtaining the approximate $q_T \cong q_\alpha^\star$. Then, substituting q_T into (P_α) (5.3), we obtain **exactly** $p_\alpha^\star$ (of course with not exact input $q_\alpha^\star$) as approximation of $p^\star$, exact solution of (P). The computation chain:

$$\underbrace{(D_\alpha)}_{\text{Frank Wolfe output: } q_T \cong q_\alpha^\star} \rightsquigarrow \underbrace{(P_\alpha)}_{\text{input: } q_T \text{ output: } p_\alpha^\star} \rightsquigarrow \underbrace{(P)}_{\text{input: } p_\alpha^\star \text{ output: } \mathbf{Gap}_{(P)}(p_\alpha^\star)}\ .$$

We would like to measure what is the $\mathbf{Gap}_{(P)}(p_\alpha^\star)$ in terms of α. Intuitively, we are trying to understand what is the price we pay in terms of utility when we introduce a regulation parameter.

By definition of ϕ

$$\begin{aligned}
\mathbf{Gap}_{(P)}(p_\alpha^\star) =& \phi(p_\alpha^\star) - \phi^\star \\
=& \phi_\alpha(p_\alpha^\star) - \alpha H(p_\alpha^\star) - \min_{r\in\Delta_{\tilde{d}}} \phi(r) \\
\leq& \phi_\alpha(p_\alpha^\star) - \alpha H(p_\alpha^\star) - \min_{r\in\Delta_{\tilde{d}}} (\phi(r) + \alpha H(r)) \\
\leq& \phi_\alpha(p_\alpha^\star) + \alpha \log(\tilde{d}) - \phi_\alpha^\star \quad \text{property of } H \\
=& \phi_\alpha(p_\alpha^\star) - \psi_\alpha^\star + \alpha \log(\tilde{d}) \quad \text{strong duality} \\
=& \phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_T) + \psi_\alpha(q_T) - \psi_\alpha^\star + \alpha \log(\tilde{d}) \\
=& \phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_T) - \mathbf{Gap}_{(D_\alpha)}(q_T) + \alpha \log(\tilde{d}) \\
=& \phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_T) + \mathbf{Gap}_{(D_\alpha)}(q_T) + \alpha \log(\tilde{d})\ .
\end{aligned}$$

Let us comment what we have found so far. $\mathbf{Gap}_{(D_\alpha)}(q_T)$ is bounded in expectation according to the utility analysis of FW 38, and it does depend on α; $\alpha \log(d)$ remains as it is; we only have to deal with $\phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_T)$.

Lemma 40. *Let $\overline{q} \in \mathbb{R}^d$ any vector and $p_\alpha^\star$*

$$p_\alpha^\star = \arg\min_{p\in\Delta_{\tilde{d}}} \left[\langle \overline{q}, -p\rangle + \alpha H(p)\right]\ .$$

Then,

$$\langle \overline{q}, p_\alpha^\star \rangle = \alpha H(p_\alpha^\star) + \alpha H^\star \left(\frac{\overline{q}}{\alpha} \right) .$$

Now, we can go back to estimating $\phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_T)$.

$$\begin{aligned} \phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_T) &= \langle q_\alpha^\star, p_\mathcal{D} - p_\alpha^\star \rangle + \alpha H(p_\alpha^\star) - \langle q_T, p_\mathcal{D} \rangle + \alpha H^\star \left(\frac{q_T}{\alpha} \right) \pm \langle q_T, \overline{q} \rangle \\ &= \langle q_\alpha^\star - q_T, p_\mathcal{D} - p_\alpha^\star \rangle \underbrace{- \langle q_T, p_\alpha^\star \rangle + \alpha H(p_\alpha^\star) + \alpha H^\star \left(\frac{q_T}{\alpha} \right)}_{=0, \text{by lemma}} \\ &= \langle q_\alpha^\star - q_T, p_\mathcal{D} - p_\alpha^\star \rangle . \end{aligned}$$

Therefore we have,

$$\mathbf{Gap}_{(P)}(p_\alpha^\star) \leq \mathbf{Gap}_{(D_\alpha)}(q_T) + \langle q_\alpha^\star - q_T, p_\mathcal{D} - p_\alpha^\star \rangle + \alpha \log(\tilde{d}) . \tag{5.5}$$

The term $\langle q_\alpha^\star - q_T, p_\mathcal{D} - p_\alpha^\star \rangle$ can be further estimated.

Relationship among $q_\alpha^\star, q_T, p_\mathcal{D}, p_\alpha^\star$

We know that by definition

$$q_\alpha^\star := \arg\max_{q \in Q} \left[\langle q, p_\mathcal{D} \rangle - \alpha H^\star \left(\frac{q}{\alpha} \right) \right] .$$

Note that Q is a compact and symmetric polyhedron with k vertices. Symmetric means that if $q \in Q \implies -q \in Q$. Using the Fourier–Motzkin algorithm, we know that there exist $S \in \mathbb{R}^{m \times N}$ matrix and $b \in \mathbb{R}^m$ such that $Q = \{q | \quad Sq \leq b \quad \text{point wise and for some } m\}$. Expressing Q as constrains, the maximization problem is now rewritten as

$$q_\alpha^\star := \arg\max_{Sq - b \leq 0} \left[\langle q, p_\mathcal{D} \rangle - \alpha H^\star \left(\frac{q}{\alpha} \right) \right] .$$

Lagrangian

$$L(q, \lambda) = \langle q, p_\mathcal{D} \rangle - \alpha H^\star \left(\frac{q}{\alpha} \right) + \lambda^T (Sq - b) .$$

Then, $q_\alpha^\star$ satisfies the KKT 53 conditions

KKT

$$\begin{cases} p_D - \nabla H^\star \left(\frac{q_\alpha^\star}{\alpha} \right) + S^T \lambda = 0 \\ \lambda_i \geq 0 \quad \text{for} i = 1, \cdots, m \\ \lambda^T (Sq - b) = 0 . \end{cases}$$

and from the first of the KKT's

$$\nabla H^\star \left(\frac{q_\alpha^\star}{\alpha} \right) - S^T \lambda = p_D .$$

Observation. $\lambda_i = 0 \iff$ the constraint i is not active.

For $q_\alpha^\star \in Q^\circ \implies \lambda_i = 0 \quad \forall i \in [m]$.

Now, consider the definition of $p^\star_\alpha$

$$p^\star_\alpha = \arg\min_{p\in\Delta_N} \left[\langle q_T, p_\mathcal{D} - p\rangle + \alpha H(p)\right] .$$

Lagrangian

$$L(p,\mu) = \langle q_T, p_\mathcal{D} - p\rangle + \alpha H(p) + \mu\left(1 - \sum_i^d p_i\right) .$$

Observation. The Lagrange multipliers w.r.t the constraints $p_i \geq 0$ all vanish.

Similarly, from the first of the KKT's

$$\nabla H(p^\star_\alpha) = \frac{q_T}{\alpha} + \frac{\mu}{\alpha}\vec{\mathbf{1}} ,$$

where $\vec{\mathbf{1}}$ is the vector whose entries are all 1. Applying $\nabla H^\star$ on both sides on this expression,

$$p^\star_\alpha = \nabla H^\star(\nabla H(p^\star_\alpha)) = \nabla H^\star\left(\frac{q_T}{\alpha} + \frac{\mu}{\alpha}\vec{\mathbf{1}}\right) = \nabla H^\star\left(\frac{q_T}{\alpha}\right) .$$

The first equality follows from properties of Fenchel's conjugate. The last equality follows doing the mathematics with the expression $\nabla H^\star(y_1, \cdots, y_N)$. [add computations in the appendix].

In conclusion

1) $p_D = \nabla H^\star\left(\frac{q^\star_\alpha}{\alpha}\right) - S^T\lambda$

2) $p^\star_\alpha = \nabla H^\star\left(\frac{q_T}{\alpha}\right)$.

Inequality

Now, let us go back to the inequality 5.5,

$$\mathbf{Gap}_{(P)}(p^\star_\alpha) \leq \mathbf{Gap}_{(D_\alpha)}(q_T) + \langle q^\star_\alpha - q_T, p_\mathcal{D} - p^\star_\alpha\rangle + \alpha\log(N) .$$

Everything is now ready to estimate the term in the middle $\langle q^\star_\alpha - q_T, p_\mathcal{D} - p^\star_\alpha\rangle$.

$$\begin{aligned}
\langle q^\star_\alpha - q_T, p_\mathcal{D} - p^\star_\alpha\rangle &= \langle q^\star_\alpha - q_T, \nabla H^\star\left(\frac{q^\star_\alpha}{\alpha}\right) - \nabla H^\star\left(\frac{q_T}{\alpha}\right) - S^T\lambda\rangle \quad \text{(using relation above)} \\
&= \langle q^\star_\alpha - q_T, \nabla H^\star\left(\frac{q^\star_\alpha}{\alpha}\right) - \nabla H^\star\left(\frac{q_T}{\alpha}\right)\rangle + \langle q^\star_\alpha - q_T, -S^T\lambda\rangle \\
&\leq \langle q^\star_\alpha - q_T, \nabla H^\star\left(\frac{q^\star_\alpha}{\alpha}\right) - \nabla H^\star\left(\frac{q_T}{\alpha}\right)\rangle + 2b^T\lambda \quad (q^\star_\alpha, -q_T \in Q) \\
&\leq ||q^\star_\alpha - q_T||_1 ||\nabla H^\star\left(\frac{q^\star_\alpha}{\alpha}\right) - \nabla H^\star\left(\frac{q_T}{\alpha}\right)||_\infty + 2b^T\lambda \quad \text{(Holder)} \\
&\leq \frac{1}{\alpha}||q^\star_\alpha - q_T||_1^2 + 2b^T\lambda \quad (H^\star \text{ is } 1-\text{smooth}).
\end{aligned}$$

Note that $q^\star_\alpha \in Q$ by definition and $q_T \in Q \implies -q_T \in Q$ because q_T is computed by Frank Wolfe updates, which are convex and within Q.

We can now glue all the pieces together. Moreover, taking the expectation (over the randomness of Frank Wolfe) on both sides of 5.5,

$$\mathbf{Gap}_{(P)}(p_\alpha^\star) \leq \mathbb{E}[\mathbf{Gap}_{(D_\alpha)}(q_T)] + \frac{1}{\alpha}||q_\alpha^\star - q_T||_1^2 + 2b^T\lambda + \alpha\log(\tilde{d}) \ . \tag{5.6}$$

Comments:

1) The $\mathbb{E}[\mathbf{Gap}_{(D_\alpha)}(q_T)]$ depends on the Frank Wolfe version we are using. For instance in 38, $\mathbb{E}[\mathbf{Gap}_{(D_\alpha)}(q_T)] \leq \mathcal{O}\left(\frac{[\log(k)]^{2/3}[\log(\frac{1}{\delta})]^{1/3}}{\alpha^{2/3}(N\varepsilon)^{2/3}}\right)$. For tuning α, we will use the $\mathbb{E}[\mathbf{Gap}_{(D_\alpha)}(q_T)]$ provided in (BGN21), which turns out to be

$$\mathbb{E}[\mathbf{Gap}_{(D_\alpha)}(q_T)] = \mathcal{O}\left(\frac{Q_1^2 + Q_1}{\alpha\varepsilon\sqrt{N}}\log\left(\frac{N}{\log(k)}\right)\log\left(\frac{KN}{\beta}\right)\sqrt{log(1/\delta)}\right) \quad \text{w.p } 1-\beta \ .$$

Q_1 denotes the diameter of Qin $||\cdot||_1$.

2) $||q_\alpha^\star - q_T||_1^2 \leq 4Q_1^2$.

3) $\lambda = \underline{0}$ if and only $q_\alpha^\star \in Q^\circ$. Unfortunately, at least one λ_i must be positive. The reason is that $H^\star$ is neither strongly nor strictly convex. For instance, if take the direction $t\vec{\mathbf{1}}$ then $t \to H(t\vec{\mathbf{1}}) = \log(N) + t$, i.e $t \to H(t\vec{\mathbf{1}})$ goes linearly. In other words, there exists a direction in which the graph of $H^\star$ is flat. All we can do is to leave $2\beta^T\lambda$ on the RHS of 5.6 and minimize for α. The minimizer $\alpha^\star$ will not depend on $2\beta^T\lambda$.

Tuning α

We minimize the right hand side of 5.6 respect to α. The optimal $\alpha^\star$ will then be

$$\alpha^\star = \sqrt{\frac{Q_1^2 + Q_1}{\log(N)\varepsilon\sqrt{N}}\log\left(\frac{N}{\log(k)}\right)\log\left(\frac{KN}{\beta}\right)\sqrt{log(1/\delta)} + \frac{Q_1^2}{\log(N)}} \ .$$

5.5 Frank Wolfe on regularized dual problem of synthetic data for empirical risk

The resulting upper bound on $\mathbf{Gap}_{(P)}(p_\alpha^\star)$ is $\mathcal{O}(1)$, due to the term $S^T\lambda$. The problem is that the Holder inequality leads to large upper bounds. We shall deal with that chain of inequality using something less strong. Let us begin by recalling a few notions from the theory of convex functions:

$$\text{If } \mathcal{L} \text{ is convex } \implies \mathcal{L}(q^\star) \geq \mathcal{L}(q_t) + \langle\nabla(q_t), q^\star - q_t\rangle \ .$$

We can then write

$$0 \leq \mathcal{L}(q_t) - \mathcal{L}(q^\star) \leq \langle\nabla\mathcal{L}(q_t), q_t - q^\star\rangle \leq \max_{s\in Q}\langle\nabla\mathcal{L}(q_t), q_t - s\rangle =: g_{\mathcal{L}}(q_t).$$

Hence, we define

Definition 41. Let $\mathcal{L}$ be a convex function. We call Frank Wolfe gap on x the quantity

$$g(x) = \max_{s\in Q}\langle\nabla\mathcal{L}(x), s - x\rangle \ .$$

Remark: This is a measure which bounds the excess risk, $g_{\mathcal{L}}(x) \geq \mathcal{L}(x) - \mathcal{L}^\star = \mathbf{Gap}_{(P)}(p_\alpha^\star)$.

We will use this notion to form an upper bound for $\mathbf{Gap}_{(P)}(p_\alpha^\star)$. For the moment, suppose we have the outcome of Frank Wolfe type algorithm, which we call q_U - and we will later see how to create such algorithm. We then denote with $p_\alpha^\star$ the solution of the 5.3 with q_U.

Let us repeat the same steps we had previously

$$
\begin{aligned}
\mathbf{Gap}_{(P)}(p_\alpha^\star) \leq & \mathbf{Gap}_{(P_\alpha)}(p_\alpha^\star) + \alpha \log(\tilde{d}) \\
= & \phi_\alpha(p_\alpha^\star) - \phi_\alpha^\star + \alpha \log(\tilde{d}) \\
= & \phi_\alpha(p_\alpha^\star) - \psi_\alpha^\star + \alpha \log(\tilde{d}) \quad \text{(strong duality)} \\
= & \phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_U) + \psi_\alpha(q_U) - \psi_\alpha^\star + \alpha \log(\tilde{d}) \quad \text{(for any iteration } q_U\text{)} \\
= & \phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_U) - \mathbf{Gap}_{(D_\alpha)}(q_U) + \alpha \log(\tilde{d}) \\
\leq & \phi_\alpha(p_\alpha^\star) - \psi_\alpha(q_U) + 0 + \alpha \log(\tilde{d}) \\
= & \langle q_\alpha^\star - q_U, p_{\mathcal{D}} - p_\alpha^\star \rangle + \alpha \log(\tilde{d}) \quad \text{(lemma)} \\
= & \langle q_\alpha^\star - q_U, p_{\mathcal{D}} - H^\star\left(\frac{q_U}{\alpha}\right) \rangle + \alpha \log(\tilde{d}) \quad \text{(previous section)} \\
= & \langle q_\alpha^\star - q_U, \nabla \psi_\alpha(q_U) \rangle + \alpha \log(\tilde{d}) \\
= & \langle q_U - q_\alpha^\star, \nabla(-\psi_\alpha)(q_U) \rangle + \alpha \log(\tilde{d}) \quad (-\psi_\alpha \text{ is convex}) \\
\leq & g_{-\psi_\alpha}(q_U) + \alpha \log(\tilde{d}) \ .
\end{aligned}
$$

We used that

$$
g_{-\psi_\alpha}(q_U) \geq -\psi_\alpha(q_U) - (-\psi_\alpha^\star) = \psi_\alpha^\star - \psi_\alpha(q_U) = \mathbf{Gap}_{(D_\alpha)}(q_U) \geq 0 \ .
$$

Therefore,

$$
\mathbf{Gap}_{(P)}(p_\alpha^\star) \leq g_{-\psi_\alpha}(q_U) + \alpha \log(\tilde{d}) \tag{5.7}
$$

$$
\implies \mathbb{E}[\mathbf{Gap}_{(P)}(p_\alpha^\star)] \leq \mathbb{E}[g_{-\psi_\alpha}(q_U)] + \alpha \log(\tilde{d}) \ . \tag{5.8}
$$

Note that q_U can be any vector in Q. We did not use any property of any algorithm. If we can produce a variant of Frank Wolfe which provides an estimate for $\mathbb{E}[g_{-\psi_\alpha}(q_U)]$, then 5.8 will be upper bounded.

Let $\mathcal{L}$ be $\mathcal{L} = -\psi_\alpha(q)$, that is convex. All we know from the previous section is that

- $\mathcal{L}$ is convex and $\frac{1}{\alpha}-$smooth w.r.t $||\cdot||_1$.
- The sensitivity of $\mathcal{A}_p(s) = \langle \nabla \mathcal{L}(q_t), s \rangle$ is at most $\frac{1}{N}$.

Therefore, for the moment we use the usual Frank Wolfe framework, following the steps

For $t = 0 : T - 1$ **do**.

$$
s_t = \underset{s \in \text{vertex}(Q)}{\arg\min} \ (\langle \nabla \mathcal{L}(q_t), s \rangle + u_s) \quad u_s \sim Lap(\lambda)
$$

$$
q_{t+1} = q_t + \gamma(s_t - q_t)
$$

End

with the difference that now γ is a fixed step size. Right now, we do not worry too much about having an output corresponding to these lines of pseudo-code. Our usual analysis of the advanced composition theorem tunes λ as

$$\lambda = \frac{4\sqrt{2}\sqrt{T\log(1/\delta)}}{\varepsilon N}$$

so that each iteration within the For cycle is (ε, δ) - DP.

Mirroring the analysis of 3, for any q_t, q_{t+1}, the following chain of inequalities holds:

$$\begin{aligned}
\mathcal{L}(q_{t+1}) \leq & \mathcal{L}(q_t) + \langle \nabla \mathcal{L}(q_t), q_{t+1} - q_t \rangle + \frac{1}{2\alpha} ||q_{t+1} - q_t||_1^2 \\
\leq & \mathcal{L}(q_t) + \gamma \langle \nabla \mathcal{L}(q_t), s_t - q_t \rangle + \frac{1}{2\alpha} Q_1^2 \gamma^2 \quad \text{(definition of } q_{t+1}) \\
\leq & \mathcal{L}(q_t) + \gamma \langle \nabla \mathcal{L}(q_t), s - q_t \rangle + \gamma V + \frac{1}{2\alpha} Q_1^2 \gamma^2 \quad \forall s \in Q, \text{ and } V = \max u_s - \min u_s \\
= & \mathcal{L}(q_t) - \gamma g_{\mathcal{L}}(q_t) + \gamma V + \frac{1}{2\alpha} Q_1^2 \gamma^2 \quad (\text{choose } s = \arg\max_{w \in Q} \langle \nabla \mathcal{L}(q_t), q_t - w \rangle) \ .
\end{aligned}$$

Now, we rearrange the inequality

$$\begin{aligned}
& \mathcal{L}(q_{t+1}) \leq \mathcal{L}(q_t) - \gamma g_{\mathcal{L}}(q_t) + \gamma V + \frac{1}{2\alpha} Q_1^2 \gamma^2 \\
\iff & \gamma g_{\mathcal{L}}(q_t) \leq \mathcal{L}(q_t) - \mathcal{L}(q_{t+1}) \ + \gamma V + \frac{1}{2\alpha} Q_1^2 \gamma^2 \\
\iff & g_{\mathcal{L}}(q_t) \leq \frac{1}{\gamma}(\mathcal{L}(q_t) - \mathcal{L}(q_{t+1})) + V + \frac{1}{2\alpha} Q_1^2 \gamma \quad \text{(dividing by } \gamma) \\
\iff & \sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t) \leq \frac{1}{\gamma} \underbrace{\sum_{t=0}^{T-1} (\mathcal{L}(q_t) - \mathcal{L}(q_{t+1}))}_{\mathcal{L}(q_0) - \mathcal{L}(q_T)} + VT + \frac{1}{2\alpha} Q_1^2 \gamma T \quad \text{(summing over } t) \\
\iff & \frac{1}{T} \sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t) \leq \frac{1}{\gamma T}(\mathcal{L}(q_0) - \mathcal{L}(q^\star)) + V + \frac{1}{2\alpha} Q_1^2 \gamma \quad \text{(dividing by } T) \\
\iff & \mathbb{E}\left[\frac{1}{T} \sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t)\right] \leq \frac{1}{\gamma T}(\mathcal{L}(q_0) - \mathcal{L}(q^\star)) + \lambda \log k + \frac{1}{2\alpha} Q_1^2 \gamma \quad \text{(taking expectation)} \ .
\end{aligned}$$

Optimizing over γ, we find that the optimal step size is

$$\gamma = \sqrt{\frac{\mathcal{L}(q_0) - \mathcal{L}(q_\alpha^\star) 2\alpha}{T Q_1^2}}$$

which, plugged in the right hand side of the inequality above (and using the given expression for λ)

$$\mathbb{E}\left[\frac{1}{T} \sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t)\right] \leq \frac{\sqrt{2} Q_1 \sqrt{\mathcal{L}(q_0) - \mathcal{L}(q_\alpha^\star)}}{\sqrt{\alpha T}} + \frac{8\sqrt{2}\sqrt{T\log(1/\delta)}\log(k)}{\varepsilon N} \ .$$

Again, optimizing over T the right hand side, this latter transforms into

$$T = \frac{\sqrt{\mathcal{L}(q_0) - \mathcal{L}(q_\alpha^\star)}Q_1}{\sqrt{\alpha}\sqrt{\log(1/\delta)\log(k)}}\varepsilon N \ .$$

Observation. By definition of expectation we have

$$\frac{1}{T}\sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t) = \mathbb{E}[g_{\mathcal{L}}(q_U)] \ ,$$

where $U \sim \text{Uni}(T)$. In other words, on the left hand side of the previous inequality we have an expectation of a variable, which is again an expectation. q_U is the given by sampling uniformly an iteration in $\{0, \cdots, T-1\}$.

In conclusion, we obtain

$$\mathbb{E}[g_{\mathcal{L}}(q_U)] \leq \mathcal{O}\left(\frac{Q_1^{1/2}\log^{1/4}(1/\delta)\log^{1/2}(k)}{\alpha^{1/4}(\varepsilon N)^{1/2}}\right) \ .$$

What done so far corresponds to running the following algorithm

Algorithm 7 Frank Wolfe for Synthetic Data

1: **Input**: (ε, δ) privacy parameters, k number of vertices, N Number of points, $p_{\mathcal{D}}$ true distribution, Q set of queries.
2: Select $q_0 \in Q$
3: Set $T = \frac{\sqrt{\psi_\alpha(q_\alpha^\star) - \psi_\alpha(q_0)}Q_1}{\sqrt{\alpha}\sqrt{\log(1/\delta)\log(k)}}\varepsilon N$, $\gamma = \sqrt{\frac{\psi_\alpha(q_\alpha^\star) - \psi_\alpha(q_0)2\alpha}{TQ_1^2}}$, $\lambda = \frac{4\sqrt{2}\sqrt{T\log(1/\delta)}}{\varepsilon N}$
4: **for** $t = 0 : T-1$ **do**
5: Compute the gradient $\nabla\psi_\alpha(q_t)$
6: $s_t = \underset{s \in \text{vertex}(Q)}{\arg\min}\ (\langle -\nabla\psi_\alpha(q_t), w\rangle + u_s)$ for $u_s \sim$Lap (λ).
7: $q_{t+1} = q_t + (s_t - q_t)\gamma$
8: Draw $U \sim$Uni$(\{0, \cdots, T-1\})$
9: **Output**: q_U

Observation. The algorithm involves $\mathcal{L}(q_0) - \mathcal{L}(q_\alpha^\star)$, which is impossible to compute. In our analysis, we could have continued the series of upper bounds with $\mathcal{L}(q_0) - \mathcal{L}(q_\alpha^\star) \leq L||q_0 - q_\alpha^\star|| \leq LQ_{||\cdot||}$, and everything we would have gotten the same expression where $LQ_{||\cdot||}$ replaces $\mathcal{L}(q_0) - \mathcal{L}(q_\alpha^\star)$.

Theorem 42. *The algorithm 7 that runs on* $\max_{q \in Q}\psi_\alpha(q)$ *5.4, is* ε, δ *DP and produces the following upper bound on the Frank Wolfe dual gap*

$$\mathbb{E}[g_{-\psi_\alpha}(q_U)] \leq \mathcal{O}\left(\frac{Q_1^{1/2}\log^{1/4}(1/\delta)\log^{1/2}(k)}{\alpha^{1/4}(\varepsilon N)^{1/2}}\right) \ .$$

Optimal α

Lemma 43. *The optimal value of* $\alpha = \alpha^\star$ *which minimizes the primal gap* $\boldsymbol{Gap}_{(P)}(p_\alpha^\star)$ *is given by*

$$\alpha^\star = \mathcal{O}\left(\frac{Q_1^{\frac{2}{5}}\log^{1/5}(1/\delta)\log^{2/5}(k)}{\log^{4/5}(\tilde{d})(\varepsilon N)^{2/5}}\right) \ .$$

Proof. Finally, we are ready to go back to our inequality 5.8.

$$\begin{aligned}\mathbb{E}[\mathbf{Gap}_{(P)}(p_\alpha^\star)] &\leq \mathbb{E}[g_{-\psi_\alpha}(q_U)] + \alpha \log(\tilde{d}) \\ &\leq \frac{Q_1^{1/2} \log^{1/4}(1/\delta) \log^{1/2}(k)}{\alpha^{1/4}(\varepsilon N)^{1/2}} + \alpha \log(\tilde{d})\end{aligned}$$

and, optimizing over α

$$\alpha^\star = \mathcal{O}\left(\frac{Q_1^{\frac{2}{5}} \log^{1/5}(1/\delta) \log^{2/5}(k)}{\log^{4/5}(\tilde{d})(\varepsilon N)^{2/5}}\right) .$$

□

Remark 6. For $k = \mathcal{O}(\tilde{d})$, remembering that $\tilde{d} = cost^d$

$$\alpha^\star = \hat{\mathcal{O}}\left(\frac{[\log(1/\delta)]^{1/5}}{(\varepsilon N)^{2/5} d^{2/5}}\right) \xrightarrow{N\to\infty} 0 .$$

Lastly, plugging $\alpha^\star$ in our gap:

$$\mathbb{E}[\mathbf{Gap}_{(P)}(p_\alpha^\star)] \leq \mathcal{O}\left(\frac{Q_1^{\frac{2}{5}} \log^{1/5}(1/\delta) \log^{2/5}(k) \log^{1/5}(\tilde{d})}{(\varepsilon N)^{2/5}}\right) .$$

Thus, the following result holds

Corollary 44. *For any dataset $\mathcal{D} \in \mathcal{X}^N$, a symmetric polyhedron of statistical linear query Q with $k = |Q|$ vertices (i.e $|Q|$ queries), and privacy parameters ε, δ, then there exists an algorithm that outputs a synthetic dataset $D^\star \rightsquigarrow p^\star$ which answers all the query with an error*

$$\mathbb{E}[\max_{q\in Q} \underbrace{|q(\mathcal{D}) - q(\mathcal{D}^\star)|}_{\langle q, p_D - p^\star\rangle}] \leq \mathcal{O}\left(\frac{d^{1/5} \log^{1/5}(1/\delta) \log^{2/5}|Q|}{(\varepsilon N)^{2/5}}\right) .$$

Remark 7. The number of queries $|Q|$ is the number of vertices k, hence, substituting, we nearly match the best known rate, 5.1

$$\mathbb{E}[\mathbf{Gap}_{(P)}(p_\alpha^\star)] \leq \mathcal{O}\left(\frac{d^{1/5} \log^{2/5}|Q| \log^{1/5}(1/\delta)}{(\varepsilon N)^{2/5}}\right) .$$

Observation. In Corollary 44, we should subtract $\phi^\star$. We can avoid it by considering the primal (P) without DP constraint, which returns $\phi^\star = 0$, and continuing to solve the (P_α) with DP. All the steps in the chain of inequalities are formally identical.

Conclusion Empirical Risk

So far, we have obtained two different algorithm for the empirical risk, 3 and 6 applied on ψ_α. These produce an utility guarantee, for any choice of α, given by

Excess Risk $\mathbb{E}[\mathcal{R}(q_T)]$	Frank Wolfe gap $\mathbb{E}[g(q_U)]$
$\mathcal{O}\left(\frac{\log^{2/3}\lvert Q\rvert \log^{1/3}(1/\delta)}{\alpha^{2/3}(N\varepsilon)^{2/3}}\right)$	$\mathcal{O}\left(\frac{Q_1^{1/2}\log^{1/4}(1/\delta)\log^{1/2}\lvert Q\rvert}{\alpha^{1/4}(\varepsilon N)^{1/2}}\right)$

The upper bound on the left clearly decays faster due as function of N, and it should: not only we did use the differentiability of ψ_α, but we also used its convexity. The upper bound derived on the right column for the Frank Wolfe gap never directly involved the convexity. We only used the $\frac{1}{\alpha}-$smoothness of $\mathcal{L} = -\psi_\alpha$ in the first step. The reason why we were interested in the upper bound on $\mathbb{E}[g(q_U)]$ is due to the inequality 5.8. If we could find a clever bound in the chain of inequality that led to 5.8, i.e the bound on $\mathbf{Gap}_{(P)}(q_\alpha^\star)$, then we could make use of it on the excess risk and produce an expected gap $\mathbb{E}[\mathbf{Gap}_{(P)}(q_\alpha^\star)] \leq \hat{\mathcal{O}}((\varepsilon N)^{-2/3})$, that would be optimal. More work needs be done on 5.6.

5.6 Frank Wolfe on regularized dual problem of synthetic data for Population Risk

As said at the begin of the Chapter, so far only the empirical distribution has been used, and never the real distribution of the population. We will now try to repeat the same analysis for the population risk, using the algorithm 4 in (BGM21). Clearly, the min max formulation presented in the first section holds replacing $\mathcal{P}$ with $p_{\mathcal{D}}$.

Observe that 5.4 can be written as a SCO optimization:

$$\begin{aligned}\psi_\alpha(q) =& \langle q, \mathcal{P}\rangle - \alpha H^\star\left(\frac{q}{\alpha}\right) = \sum_{i\in\mathcal{D}} q_i\mathcal{P}_i - \alpha H^\star\left(\frac{q}{\alpha}\right)\\ =& \mathbb{E}_{z\sim\mathcal{P}}[q_z] - \alpha H^\star\left(\frac{q}{\alpha}\right) = \mathbb{E}_{z\sim\mathcal{P}}\left[q_z - \alpha H^\star\left(\frac{q}{\alpha}\right)\right]\\ =:& \mathbb{E}_{z\sim\mathcal{P}}[l(q,z)]\end{aligned}$$

where q_z is the $z-$th entry of the vector q, our query.

Lemma 45. *$l(q,z) := q_z - \alpha H^\star\left(\frac{q}{\alpha}\right)$ is both $L_0 = 2$ Lipschitz and $L_1 = \frac{1}{\alpha}$ smooth w.r.t $||\cdot||_1$ in the first argument.*

Proof. By definition,

Lipschitz:

$$\begin{aligned} l(q_1,z)-l(q_2,z)| =&|q_z^1-\alpha H^\star\left(\frac{q_1}{\alpha}\right)-q_z^2+\alpha H^\star\left(\frac{q_2}{\alpha}\right)| \\ \leq& \underbrace{|q_z^1-q_z^2|}_{\sum_z |q_z^1-q_z^2|}+\alpha|H^\star\left(\frac{q_1}{\alpha}\right)-H^\star\left(\frac{q_2}{\alpha}\right)| \\ \leq& ||q_1-q_2||_1+\frac{\alpha}{\alpha}||q_1-q_2||_1 \ . \end{aligned}$$

Smoothness:

$$||\nabla l(q_1,z)-\nabla l(q_2,z)||_\infty =||\mathbf{e}_z-\nabla H^\star(q_1/\alpha)-\mathbf{e}_z+\nabla H^\star(q_2/\alpha)||_\infty \leq \frac{1}{\alpha}||q_1-q_2||_1 \ .$$

□

Corollary 46 (population)**.** *For any dataset $\mathcal{D}\in\mathcal{X}^N$, a symmetric polyhedron of statistical linear queries Q with k vertices, and privacy parameters ε,δ, then there exists an algorithm that outputs a synthetic dataset $D^\star \rightsquigarrow p^\star$ which answers all queries with an error*

$$\mathbb{E}[\max_{q\in Q}\underbrace{|q(\mathcal{D})-q(\mathcal{D}^\star)|}_{\langle q,\mathcal{P}-p^\star\rangle}] \leq \frac{Q_1[\log(k)]^{1/3}[\log(N)]^{1/3}[\log(1/\delta)]^{1/12}[\log(\tilde{d})]^{1/2}}{(N\varepsilon)^{1/6}} \ .$$

Furthermore, the optimal $\alpha^\star$ is

$$\alpha^\star=\frac{Q_1[\log(k)]^{1/3}[\log(N)]^{1/3}[\log(1/\delta)]^{1/12}}{[\log(\tilde{d})]^{1/2}(N\varepsilon)^{1/6}} \ .$$

Proof. From (BGM21),

$$\mathbb{E}[g_{-\psi_\alpha}(q_U)] \leq \mathcal{O}\left(Q_1\left(2+\frac{Q_1}{\alpha}\right)\frac{[\log(k)]^{2/3}[\log(N)]^{2/3}[\log(1/\delta)]^{1/6}}{N^{1/3}\varepsilon^{1/3}}\right) \ .$$

Plugging this in 5.8,

$$\mathbf{Gap}_{(P)}(p_\alpha^\star) \leq Q_1\left(2+\frac{Q_1}{\alpha}\right)\frac{[\log(k)]^{2/3}[\log(N)]^{2/3}[\log(1/\delta)]^{1/6}}{N^{1/3}\varepsilon^{1/3}}+\alpha\log(\tilde{d}) \ .$$

Optimizing over α, we get

$$\alpha^\star=\frac{Q_1[\log(k)]^{1/3}[\log(N)]^{1/3}[\log(1/\delta)]^{1/12}}{[\log(\tilde{d})]^{1/2}(N\varepsilon)^{1/6}} \ .$$

For $k=\mathcal{O}(d)$

$$\alpha^\star=\mathcal{O}\left(\frac{Q_1[\log(N)]^{1/3}[\log(1/\delta)]^{1/12}}{[\log(\tilde{d})]^{1/6}(N\varepsilon)^{1/6}}\right)\xrightarrow[N\to\infty]{} 0 \ .$$

This $\alpha^\star$ returns

$$\mathbf{Gap}_{(P)}(p_\alpha^\star) \leq \frac{Q_1[\log(k)]^{1/3}[\log(N)]^{1/3}[\log(1/\delta)]^{1/12}[\log(\tilde{d})]^{1/2}}{(N\varepsilon)^{1/6}} \,.$$

□

Remark 8. For $k = \mathcal{O}(\tilde{d})$

$$\mathbf{Gap}_{(P)}(p_\alpha^\star) \leq \mathcal{O}\left(\frac{Q_1[\log(\tilde{d})]^{5/6}[\log(N)]^{1/3}[\log(1/\delta)]^{1/12}}{(N\varepsilon)^{1/6}}\right) \xrightarrow[N\to\infty]{} 0 \,.$$

5.7 Full Batch Frank Wolfe on dual synthetic data of population risk

We can try to apply 7 on $\psi_\alpha(q) = \mathbb{E}_{z\sim\mathcal{P}}[l(q,z)]$ using an estimator of the gradient for $\nabla\psi_\alpha(q)$.

Theorem 47. *Suppose we apply 7 on $\psi_\alpha(q) = \mathbb{E}_{z\sim\mathcal{P}}[l(q,z)]$ with the unbiased gradient estimator at every iteration t given by $\nabla_t = \frac{1}{N}\sum_{i=1}^N e_{z_i} - \nabla H^\star\left(\frac{q}{\alpha}\right)$, where $z_i \sim \mathcal{P}$ i.i.d (sample dataset). Then,*

$$\mathbb{E}[g_{-\psi_\alpha}(q_U)] \leq \mathcal{O}\left(\frac{Q_1^{1/2}\log^{1/4}(1/\delta)\log^{1/2}(k)}{\alpha^{1/4}(\varepsilon N)^{1/2}} + \frac{Q_1\log^{1/2}(\tilde{d})}{\sqrt{N}}\right) \,.$$

Proof.

$$\begin{aligned}
\mathcal{L}(q_{t+1}) \leq& \mathcal{L}(q_t) + \langle\nabla\mathcal{L}(q_t), q_{t+1} - q_t\rangle + \frac{1}{2\alpha}||q_{t+1} - q_t||_1^2 \\
\leq& \mathcal{L}(q_t) + \gamma\langle\nabla\mathcal{L}(q_t) \pm \nabla_t, s_t - q_t\rangle + \frac{1}{2\alpha}Q_1^2\gamma^2 \quad \text{(definition of } q_{t+1}\text{)} \\
=& \mathcal{L}(q_t) + \gamma\langle\nabla\mathcal{L}(q_t) - \nabla_t, s_t - q_t\rangle + \gamma\langle\nabla_t, s_t - q_t\rangle\frac{1}{2\alpha}Q_1^2\gamma^2 \quad \text{(definition of } q_{t+1}\text{)} \\
\leq& \mathcal{L}(q_t) + \gamma\langle\nabla\mathcal{L}(q_t) - \nabla_t, s_t - q_t\rangle + \gamma\langle\nabla_t, s - q_t\rangle\frac{1}{2\alpha}Q_1^2\gamma^2 + \gamma V \\
=& \mathcal{L}(q_t) + \gamma\langle\nabla\mathcal{L}(q_t) - \nabla_t, s_t - q_t\rangle \pm \gamma\langle\nabla\mathcal{L}(q_t), s - q_t\rangle + \gamma\langle\nabla_t, s - q_t\rangle\frac{1}{2\alpha}Q_1^2\gamma^2 + \gamma V \\
=& \mathcal{L}(q_t) - \gamma g_{\mathcal{L}}(q_t) + \gamma\langle\nabla\mathcal{L}(q_t) - \nabla_t, s_t - s\rangle + \gamma V + \frac{1}{2\alpha}Q_1^2\gamma^2 \quad ,
\end{aligned}$$

choosing $s = \arg\max_{w\in Q}\langle\nabla\mathcal{L}(q_t), q_t - w\rangle)$. Now, we rearrange the inequality

$$\begin{aligned}
\mathcal{L}(q_{t+1}) \leq& \mathcal{L}(q_t) - \gamma g_{\mathcal{L}}(q_t) + \gamma\langle\nabla\mathcal{L}(q_t) - \nabla_t, s_t - s\rangle + \gamma V + \frac{1}{2\alpha}Q_1^2\gamma^2 \\
\iff \gamma g_{\mathcal{L}}(q_t) \leq& \mathcal{L}(q_t) - \mathcal{L}(q_{t+1}) + \gamma\langle\nabla\mathcal{L}(q_t) - \nabla_t, s_t - s\rangle + \gamma V + \frac{1}{2\alpha}Q_1^2\gamma^2 \\
\iff g_{\mathcal{L}}(q_t) \leq& \frac{1}{\gamma}(\mathcal{L}(q_t) - \mathcal{L}(q_{t+1})) + ||\nabla\mathcal{L}(q_t) - \nabla_t||_\infty Q_1 + V + \frac{1}{2\alpha}Q_1^2\gamma \quad \text{(dividing by } \gamma\text{)} \\
\iff \sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t) \leq& \frac{1}{\gamma}\underbrace{\sum_{t=0}^{T-1}(\mathcal{L}(q_t) - \mathcal{L}(q_{t+1}))}_{\mathcal{L}(q_0)-\mathcal{L}(q_T)} + Q_1\sum_{t=0}^{T-1}||\nabla\mathcal{L}(q_t) - \nabla_t||_\infty + VT + \frac{1}{2\alpha}Q_1^2\gamma T
\end{aligned}$$

Note however, that

$$\begin{aligned}||\nabla_t - \nabla\mathcal{L}(q_t)||_\infty =& ||\frac{1}{N}\sum_{i=1}^{N} e_{z_i} - \nabla H^\star\left(\frac{q_t}{\alpha}\right) - \mathbb{E}_{z\sim\mathcal{P}}[e_z] + \nabla H^\star\left(\frac{q_t}{\alpha}\right)||_\infty \\ =& ||\frac{1}{N}\sum_{i=1}^{N} e_{z_i} - \mathbb{E}_{z\sim\mathcal{P}}[e_z]||_\infty \ ,\end{aligned}$$

i.e, it does not depend on t. Therefore

$$\begin{aligned}\iff \sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t) \leq & \frac{1}{\gamma}\underbrace{\sum_{t=0}^{T-1}(\mathcal{L}(q_t) - \mathcal{L}(q_{t+1}))}_{\mathcal{L}(q_0)-\mathcal{L}(q_T)} + TQ_1||\frac{1}{N}\sum_{i=1}^{N} e_{z_i} - \mathbb{E}_{z\sim\mathcal{P}}[e_z]||_\infty + VT + \frac{1}{2\alpha}Q_1^2\gamma T \\ \iff \frac{1}{T}\sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t) \leq & \frac{1}{\gamma T}(\mathcal{L}(q_0) - \mathcal{L}(q^\star)) + Q_1||\frac{1}{N}\sum_{i=1}^{N} e_{z_i} - \mathbb{E}_{z\sim\mathcal{P}}[e_z]||_\infty + V + \frac{1}{2\alpha}Q_1^2\gamma \\ \iff \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t)\right] \leq & \frac{1}{\gamma T}(\mathcal{L}(q_0) - \mathcal{L}(q^\star)) + Q_1\mathbb{E}\left[||\frac{1}{N}\sum_{i=1}^{N} e_{z_i} - \mathbb{E}_{z\sim\mathcal{P}}[e_z]||_\infty\right] + \lambda\log k + \frac{1}{2\alpha}Q_1^2\gamma \ .\end{aligned}$$

From (JN08) or (DGVW10), it holds $\mathbb{E}\left[||\frac{1}{N}\sum_{i=1}^{N} e_{z_i} - \mathbb{E}_{z\sim\mathcal{P}}[e_z]||_\infty\right] \leq \mathcal{O}\left(\sqrt{\frac{\log\tilde{d}}{N}}\right)$. Therefore,

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} g_{\mathcal{L}}(q_t)\right] \leq \frac{1}{\gamma T}(\mathcal{L}(q_0) - \mathcal{L}(q^\star)) + Q_1\sqrt{\frac{\log\tilde{d}}{N}} + \lambda\log k + \frac{1}{2\alpha}Q_1^2\gamma \ .$$

Also, $\frac{Q_1\log(\tilde{d})}{\sqrt{N}}$ neither involves γ nor T, hence the analysis continues exactly like in the utility proof of 7, obtaining

$$\mathbb{E}[g_{-\psi_\alpha}(q_U)] \leq \mathcal{O}\left(\frac{Q_1^{1/2}\log^{1/4}(1/\delta)\log^{1/2}(k)}{\alpha^{1/4}(\varepsilon N)^{1/2}} + \frac{Q_1 d^{1/2}}{\sqrt{N}}\right) \ .$$

□

Moreover, the same observation that $\sqrt{\frac{\log\tilde{d}}{N}}$ does not depend on α yields

Corollary 48 (population)**.** *For any dataset $\mathcal{D} \in \mathcal{X}^N$, a symmetric polyhedron of statistical linear query Q with k vertices, and privacy parameters ε, δ, then there exists an algorithm that outputs a synthetic dataset $D^\star \rightsquigarrow p^\star$ which answers all queries with an error*

$$\mathbb{E}[\max_{q\in Q}\underbrace{|q(\mathcal{D}) - q(\mathcal{D}^\star)|}_{\langle q, \mathcal{P}-p^\star\rangle}] \leq \mathcal{O}\left(\frac{d^{1/5}\log^{1/5}(1/\delta)\log^{2/5}|Q|}{(\varepsilon N)^{2/5}} + \frac{d^{1/2}}{\sqrt{N}}\right) \ .$$

The optimal $\alpha^\star$ is the same as in 43.

Conclusion population risk

As far as we are aware, there are no results for bounding the $\mathbf{Gap}_{(P)}$ using the population risk. What we found applying the algorithms of the previous section is summarized in the table:

Frank Wolfe gap: Population (BGM21)	Frank wolfe gap: Population alg. 7
$\mathcal{O}\left(Q_1\left(2+\frac{Q_1}{\alpha}\right)\frac{[\log(k)]^{2/3}[\log(N)]^{2/3}[\log(1/\delta)]^{1/6}}{N^{1/3}\varepsilon^{1/3}}\right)$	$\mathcal{O}\left(\frac{Q_1^{1/2}\log^{1/4}(1/\delta)\log^{1/2}(k)}{\alpha^{1/4}(\varepsilon N)^{1/2}}+\frac{Q_1\log^{1/2}(\tilde{d})}{\sqrt{N}}\right)$

Using alg 7 we pay an additional term $\mathcal{O}(\sqrt{d}/\sqrt{N})$. Again, during the analysis of the utility, we did not use the convexity of $-\psi_\alpha$, expect for the condition of $\frac{1}{\alpha}$ smoothness. This leave space for possible future improvements.

Chapter 6

Appendix

6.1 Appendix chapter 3

Lemma 49. *Let* $R(w_{t+1}) \leq (1-\gamma_t)R(w_t) + \gamma_t M_1 + \gamma_r^2 M_2$. *Then,*

$$R(w_t) \leq M_1 + M_2 a_t,$$

where

$$\begin{cases} a_0 = 1 \\ a_{t+1} = a_t(1-\gamma_t) + \gamma_t^2 \quad \gamma_t = \frac{2}{2+t}. \end{cases}$$

Proof. By induction

- $t = 0$:

$$R(w_1) \leq \left(1 - \frac{2}{2+0}\right) R(w_0) + \frac{2}{2+0} M_1 + \left(\frac{2}{2+0}\right)^2 M_2 = M_1 + M_2 \cdot 1.$$

- $t \Longrightarrow t+1$

$$\begin{aligned} R(w_{t+1}) \leq & (1-\gamma_t)R(w_t) + \gamma_t M_1 + \gamma_r^2 M_2 \\ \leq & (1-\gamma_t)M_1 + (1-\gamma_t)a_t M_2 + \gamma_t M_1 + \gamma_r^2 M_2 \\ = & M_1 + M_2(a_t(1-\gamma_t) + \gamma_t^2) \\ = & M_1 + M_2 a_{t+1}. \end{aligned}$$

□

Lemma 50. *The partial derivatives of the function*

$$l(\beta) = \sum_{i=1}^{N} \sum_{j=1}^{k} -y_{ji} \log(\pi_{ij})$$

are

$$\frac{\partial l}{\partial \beta_{st}} = -X_{row(t)}(Y_{row(s)}^T - \Pi_{col(s)}).$$

Proof.

$$\frac{\partial l}{\partial \beta_{st}} = -\sum_{i=1}^{N}\sum_{j=1}^{k}\frac{y_{ji}}{\pi_{ij}}\frac{\partial \pi_{ij}}{\partial \beta_{st}} = -\sum_{i=1}^{N}\left(\sum_{j=1}^{k-1}\frac{y_{ji}}{\pi_{ij}}\frac{\partial \pi_{ij}}{\partial \beta_{st}} + \frac{y_{ki}}{\pi_{ik}}\frac{\partial \pi_{ik}}{\partial \beta_{st}}\right).$$

Everything boils down to computing $\frac{\partial \pi_{ij}}{\partial \beta_{st}}$

- CASE $j < k$

$$\frac{\partial \pi_{ij}}{\partial \beta_{st}} = \frac{\partial}{\partial \beta_{st}}\left(\frac{e^{\langle x_i,\beta_j\rangle}}{\underbrace{1+\sum_{r=1}^{k-1}e^{\langle x_i,\beta_r\rangle}}_{=:D}}\right) = \frac{x_{ti}e^{\langle x_i,\beta_j\rangle}\left(\delta_{js}D - e^{\langle x_i,\beta_s\rangle}\right)}{D^2}.$$

- CASE $j = k$

$$\frac{\partial \pi_{ik}}{\partial \beta_{st}} = \frac{\partial}{\partial \beta_{st}}\left(\frac{1}{\underbrace{1+\sum_{r=1}^{k-1}e^{\langle x_i,\beta_r\rangle}}_{=:D}}\right) = -\frac{e^{\langle x_i,\beta_s\rangle}x_{ti}}{D^2}.$$

Substituting these two expressions into the derivatives above:

$$\begin{aligned}
\frac{\partial l}{\partial \beta_{st}} &= -\sum_{i=1}^{N}\left(\sum_{j=1}^{k-1}\frac{y_{ji}}{\pi_{ij}}\frac{\partial \pi_{ij}}{\partial \beta_{st}} + \frac{y_{ki}}{\pi_{ik}}\frac{\partial \pi_{ik}}{\partial \beta_{st}}\right) = \\
&= -\sum_{i=1}^{N}\left(\sum_{j=1}^{k-1}\frac{y_{ji}}{\pi_{ij}}\frac{x_{ti}e^{\langle x_i,\beta_j\rangle}\left(\delta_{js}D - e^{\langle x_i,\beta_s\rangle}\right)}{D^2} - \frac{y_{ki}}{\pi_{ik}}\frac{e^{\langle x_i,\beta_s\rangle}x_{ti}}{D^2}\right) = \\
&= -\sum_{i=1}^{N}\left(\sum_{j=1}^{k-1}\frac{y_{ji}}{\pi_{ij}}\frac{x_{ti}e^{\langle x_i,\beta_j\rangle}\delta_{js}}{D} - \frac{x_{ti}}{D^2}e^{\langle x_i,\beta_s\rangle}\left(\sum_{j=1}^{k-1}\frac{y_{ji}}{\pi_{ij}}e^{\langle x_i,\beta_j\rangle} + \frac{y_{ki}}{\pi_{ik}}\right)\right) \quad , \left[\pi_{ij} = \frac{e^{\langle x_i,\beta_j\rangle}}{D}\right] \\
&= -\sum_{i=1}^{N}\left(\sum_{j=1}^{k-1}\frac{y_{ji}}{\pi_{ij}}\frac{x_{ti}e^{\langle x_i,\beta_j\rangle}\delta_{js}}{D} - \frac{x_{ti}}{D}e^{\langle x_i,\beta_s\rangle}\underbrace{\sum_{j=1}^{k}y_{ji}}_{1}\right), \quad \delta_{js} = 1 \Longleftrightarrow j = s \\
&= -\sum_{i=1}^{N}\left(\frac{y_{si}}{\pi_{is}}\frac{x_{ti}e^{\langle x_i,\beta_s\rangle}}{D} - \frac{x_{ti}}{D}e^{\langle x_i,\beta_s\rangle}\right) = \\
&= -\sum_{i=1}^{N}\left(y_{si} - \pi_{is}\right)x_{ti} = \\
&= -\langle y_s^T - \Pi_s, x_t^T\rangle \quad \text{for } s = 1,\cdots,k-1 \quad t = 0,\cdots,p.
\end{aligned}$$

We can rewrite this last expression using the matrix notation:

$$\frac{\partial l}{\partial \beta_{st}} = -X_{\text{row}(t)}(Y^T_{\text{row}(s)} - \Pi_{\text{col}(s)}).$$

□

6.2 Appendix chapter 5

Definition 51. Let $f : E \longrightarrow \mathbb{R}$ be a convex function. We define the Fenchel conjugate of f as the function $f^\star : E \longrightarrow \mathbb{R}$

$$f^\star(y) = \sup_{x \in E}[\langle y, x\rangle - f(x)] \, .$$

Proposition 52 (properties)**.** *Let $f : E \longrightarrow \mathbb{R}$ be convex. Then*

1. $f(x) + f^\star(y) \geq \langle y, x\rangle \quad \forall x, y \in E$
2. *The equality holds if and only if $y \in \partial f(x)$*
3. $f^{\star\star} = f$

Theorem 53 (KKT conditions.)**.** *Let f, g_i, h_i be convex and differentiable. Consider*

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s.t. & g_i(x) = 0 \quad i = 1, \cdots, n \\ & h_j(x) \leq 0 \quad j = 1, \cdots, m \, . \end{array} \tag{6.1}$$

Then, necessary condition for x to be a minimizer is that

$$\begin{cases} \nabla f(x) + \lambda^T \nabla g(x) + \mu^T \nabla h(x) = 0 \\ g(x) = 0 \quad h(x) \leq 0 \\ \mu^T h(x) = 0 \\ \mu_i \geq 0 \quad \forall i = 1, \cdots, m \, . \end{cases}$$

Theorem 54 (Zalinescu: 1983)**.** *Let $||\cdot||$ be a norm and $k > 0$. Then*

1) If $f : E \to \mathbb{R}$ is convex and $1/k-$ smooth w.r.t $||\cdot||$, then $f^\star$ is $k-$strongly convex w.r.t $||\cdot||_\star$.

2) If $f : E \to \mathbb{R}$ is $k-$strongly convex w.r.t $||\cdot||$, then $f^\star$ is $1/k-$smooth w.r.t $||\cdot||_\star$.

Lemma 55. *The negative entropy*

$$\phi(x) = \begin{cases} \sum_{j=1}^d x_j \ln(x_j) & x_j \geq 0, \quad \sum_{j=1}^d x_j = 1 \\ +\infty & otherwise \end{cases}$$

is strongly convex and its Fenchel conjugate is

$$\phi^\star(y) = \log\left(\sum_{j=1}^d e^{y_j}\right) \, .$$

Proof. We will proof that, since ϕ is convex and continuous, $\phi^{\star\star} = \phi$. Furthermore, for a convex function f that is also $\frac{1}{\kappa}$- smooth respect to a norm $||\cdot||$, then $f^\star$ is strongly convex with respect to the dual $||\cdot||_\star$.

Therefore, it is enough show that $\phi^\star$ is $\frac{1}{\kappa}$- smooth for some norm, and therefore obtaining $\phi^{\star\star} = \phi$ strongly convex.

STEP 1: So, $\phi^\star$ needs to be computed first. By definition $\phi^\star(y)$ is the sup of

$$\begin{aligned} \sup_x \quad & \langle y, x\rangle - \sum_{j=1}^d x_j \ln(x_j) \\ \text{s.t.} \quad & \sum_{j=1}^d x_j - 1 = 0 \\ & -x_j \leq 0 \quad i = 1, \cdots, d \end{aligned} \tag{6.2}$$

Changing the sup with $-\inf -$function, the problem (1) is in standard form and it is possible to apply the KKT conditions.

The Lagrangian is

$$L(x, \lambda, \mu) = \phi(x) - \langle y, x\rangle - \mu^T x + \lambda\left(\sum_{j=1}^d x_j - 1\right).$$

Therefore, the KKT:

$$\begin{aligned} \nabla(x) - y - \mu + \lambda\underline{1} =& 0 \\ \mu_i \geq& 0 \quad i = 1, \cdots, d \\ \mu_i x_i =& 0 \quad i = 1, \cdots, d \\ \sum_{j=1}^d x_j - 1 =& 0 \\ -x_j \leq& 0 \quad i = 1, \cdots, d \end{aligned}$$

- From the first equation:

$$\nabla\phi(x) - y - \mu + \lambda\underline{1} = [\ln(x_1), \cdots, \ln(x_d)] + (\lambda + 1)\underline{1} - y - \mu = 0$$

  so, for each j index

$$x_j = \exp(y_j + \mu_j - 1 - \lambda).$$

- From the condition $\mu_j x_j = 0$, it follows

$$\mu_j = 0$$

  because $x_j > 0$ since it is an exponential, thus $x_j = \exp(y_j + -1 - \lambda)$.

- From $\sum_j x_j = 1$,

$$\sum_{j=1}^d x_j = e^{1-\lambda}\sum_{j=1}^d e^{y_j} = 1$$

and taking the ln on both sides,

$$1-\lambda+\ln\left(\sum_{j=1}^{d}e^{y_j}\right)=0\Longleftrightarrow\lambda=\ln\left(\sum_{j=1}^{d}e^{y_j}\right)+1.$$

Note that each x_j is expressed in term of the value λ just computed and it is the only unknown. In other words, we know $x=x(\lambda)$.

Everything is ready to compute $\phi^\star$. Substituting x in (1) and, taking into account that we changed the sign of the optimization problem:

$$\begin{aligned}\phi^\star(y)=&\langle y,x(\lambda)\rangle-\sum_{j=1}^{d}x_j\ln(x_j)=\sum_{j=1}^{d}y_jx_j-\sum_{j=1}^{d}x_j(1+y_j-\lambda)\\&=\sum_{j=1}^{d}y_jx_j-\sum_{j=1}^{d}y_jx_j-(1-\lambda)\underbrace{\sum_{j=1}^{d}x_j}_{=1}=\lambda-1\\&=\ln\left(\sum_{j=1}^{d}e^{y_j}\right)+1-1=\ln\left(\sum_{j=1}^{d}e^{y_j}\right).\end{aligned}$$

In conclusion,

$$\phi^\star(y)=\ln\left(\sum_{j=1}^{d}e^{y_j}\right).$$

STEP 2: From the theory, we know that, given $f:E\longrightarrow\mathbb{R}$ convex and $\mathcal{C}^2$,

$$L_1\text{ smooth }\Longleftrightarrow 0\leq\langle\nabla^2fh,h\rangle\leq L_1||h||^2.$$

This is the proposition that I am going to use to show the f is L_1 smooth.
We need the derivatives of $\phi^\star$ first. Calling $\sum_j e^{y_j}=S(y)$,

$$\frac{\partial\phi^\star}{\partial y_i}=\frac{e^{e_i}}{S(y)}$$

and

$$\frac{\partial\phi^\star}{\partial y_i\partial y_j}=\frac{\partial\phi^\star}{\partial y_i}\left[\frac{e_j^y}{S(y)}\right]=-\frac{e^{y_i}e^{y_j}}{(S(y))^2}.$$

Now, we should compute $\langle(\nabla^2f)h,h\rangle$.

Observation. For a generic matrix A,

$$\langle x,Ax\rangle=\sum_{i=1}^{d}x_i\sum_{j=1}^{d}a_{ij}x_j\leq\sum_{i=1}^{d}x_i||A||_\infty||x||_\infty\leq||A||_\infty||x||_\infty||x||_1\leq||A||_\infty||x||_1^2.$$

Therefore,

$$||\nabla^2\phi^\star(y)||_\infty=\max_{1\leq i\leq n}\sum_{j=1}^{d}\frac{e^{y_i}e^{y_j}}{(S(y))^2}=\max_{1\leq i\leq n}\frac{e^{y_i}}{(S(y))^2}\sum_{j=1}^{d}e^{y_j}=\frac{e^{y_i}}{(S(y))^2}S(y)=\max_{1\leq i\leq n}\frac{e^{y_i}}{S(y)}\leq 1.$$

To conclude, $\phi^\star$ is L_1 is 1-smooth w.r.t $||\cdot||_1$.

Here, the norms are flipped respect to what it was suggested in the slides. Nevertheless, as consequence of the equivalence of norms in finite spaces, all the norms are equivalent. The request of the exercise was to show the strong convexity. I thought that this should be fine.

STEP 3:

$\phi^\star$ is convex and 1- smooth respect to $||\cdot||_1 \implies \phi^{\star\star} = \phi$ is 1- strongly convex respect to $(||\cdot||_1)_\star$.

□

Lemma 56. *Let $\overline{q} \in \mathbb{R}^N$ and $\overline{p}$*

$$\overline{p} = \underset{p\in\Delta_N}{\arg\min} \left[\langle \overline{q}, -p\rangle + \alpha H(p)\right] .$$

Then,

$$\langle \overline{q}, \overline{p}\rangle = \alpha H(\overline{p}) + \alpha H^\star\left(\frac{\overline{q}}{\alpha}\right).$$

Proof. Consider the definition of $\overline{p}$:

$$\overline{p} = \underset{p\in\Delta_N}{\arg\min} \left[\langle \overline{q}, p_{\mathcal{D}} - p\rangle + \alpha H(p)\right] .$$

Lagrangian

$$L(p,\mu) = \langle \overline{q}, p_{\mathcal{D}} - p\rangle + \alpha H(p) + \mu\left(1 - \sum_i^N p_i\right)$$

Observation. The Lagrange multipliers w.r.t the constraints $p_i \geq 0$ all vanish.

Similarly, from the first of the KKT's

$$\nabla H(\overline{p}) = \frac{\overline{q}}{\alpha} + \frac{\mu}{\alpha}\vec{\mathbf{1}} ,$$

where $\vec{\mathbf{1}}$ is the vector whose entries are all 1. Applying $\nabla H^\star$ on both sides on this expression,

$$\overline{p} = \nabla H^\star(\nabla H(\overline{p})) = \nabla H^\star\left(\frac{\overline{q}}{\alpha} + \frac{\mu}{\alpha}\vec{\mathbf{1}}\right) = \nabla H^\star\left(\frac{\overline{q}}{\alpha}\right) .$$

The first equality follows from properties of Fenchel's conjugate. The last equality follows doing the mathematics with the expression $\nabla H^\star(y_1, \cdots, y_N)$. We can substitute the value of $\overline{q}$ in the objective

$$\begin{aligned} -\langle \overline{q}, \overline{p}\rangle + \alpha H(\overline{p}) &= -\langle \overline{q}, \overline{p}\rangle + \sum_{i=1}^N p_i \log(p_i) \\ &= -\langle \overline{q}, \overline{p}\rangle + \sum_{i=1}^N p_i \left[\frac{q_i}{\alpha} - \log\left(\sum_{j=1}^N e^{\frac{q_j}{\alpha}}\right)\right] = -\alpha H^\star(\overline{q}) . \end{aligned}$$

□

Bibliography

[ABK+21] Sergul Aydore, William Brown, Michael Kearns, Krishnaram Kenthapadi, Luca Melis, Aaron Roth, and Ankit Siva, *Differentially private query release through adaptive projection*, 2021.

[AFKT21] Hilal Asi, Vitaly Feldman, Tomer Koren, and Kunal Talwar, *Private stochastic convex optimization: Optimal rates in ℓ_1 geometry*, 2021.

[BFTT19] Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Thakurta, *Private stochastic convex optimization with optimal rates*, 2019.

[BGM21] Raef Bassily, Cristóbal Guzmán, and Michael Menart, *Differentially private stochastic optimization: New results in convex and non-convex settings*, 2021.

[BGN21] Raef Bassily, Cristóbal Guzmán, and Anupama Nandi, *Non-euclidean differentially private stochastic convex optimization*, CoRR **abs/2103.01278** (2021).

[BKN10] Amos Beimel, Shiva Kasiviswanathan, and Kobbi Nissim, *Bounds on the sample complexity for private learning and private data release*, vol. 94, 02 2010, pp. 437–454.

[BLR11] Avrim Blum, Katrina Ligett, and Aaron Roth, *A learning theory approach to non-interactive database privacy*, 2011.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta, *Differentially private empirical risk minimization: Efficient algorithms and tight error bounds*, 2014.

[CLC+22] M. A. P. Chamikara, Dongxi Liu, Seyit Camtepe, Surya Nepal, Marthie Grobler, Peter Bertok, and Ibrahim Khalil, *Local differential privacy for federated learning in industrial settings*, 2022.

[CMS09] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate, *Differentially private empirical risk minimization*, 2009.

[CWS20] Junjie Chen, Wendy Wang, and Xinghua Shi, *Differential privacy protection against membership inference attack on machine learning for genomic data.*

[DGVW10] Lutz Dümbgen, Sara Geer, Mark Veraar, and Jon Wellner, *Nemirovski's inequalities revisited*, The American mathematical monthly : the official journal of the Mathematical Association of America **117** (2010), 138–160.

[DNT13] Cynthia Dwork, Aleksandar Nikolov, and Kunal Talwar, *Efficient algorithms for privately releasing marginals via convex relaxations*, 2013.

[DR13] Cynthia Dwork and Aaron Roth, *The algorithmic foundations of differential privacy*, Foundations and Trends in Theoretical Computer Science **9** (2013).

[DRV10] Cynthia Dwork, Guy Rothblum, and Salil Vadhan, *Boosting and differential privacy*, Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '10) (Las Vegas, NV), IEEE, IEEE, 23–26 October 2010, p. 51–60.

[GAH+14] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu, *Dual query: Practical private query release for high dimensional data*.

[HLM10] Moritz Hardt, Katrina Ligett, and Frank McSherry, *A simple and practical algorithm for differentially private data release*, 2010.

[HSDZ21] Hongsheng Hu, Zoran Salcic, Gillian Dobbie, and Xuyun Zhang, *Membership inference attacks on machine learning: A survey*, CoRR **abs/2103.07853** (2021).

[HSS+21] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang, *Membership inference attacks on machine learning: A survey*, 2021.

[Jag13] Martin Jaggi, *Revisiting frank-wolfe: Projection-free sparse convex optimization*, vol. 28, 01 2013.

[JN08] Anatoli B. Juditsky and Arkadii S. Nemirovski, *Large Deviations of Vector-valued Martingales in 2-Smooth Normed Spaces*, working paper or preprint, May 2008.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An introduction to statistical learning: with applications in r*, Springer, 2013.

[KD19] K. S. Sesh Kumar and Marc Peter Deisenroth, *Differentially private empirical risk minimization with sparsity-inducing norms*, ArXiv **abs/1905.04873** (2019).

[KHD20] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras, *On the effectiveness of regularization against membership inference attacks*, 2020.

[LM11] Chao Li and Gerome Miklau, *Efficient batch query answering under differential privacy*, 2011.

[LVW21] Terrance Liu, Giuseppe Vietri, and Zhiwei Steven Wu, *Iterative methods for private synthetic data: Unifying framework and new methods*, 2021.

[NDT+20] Geoffrey Négiar, Gideon Dresdner, Alicia Tsai, Laurent El Ghaoui, Francesco Locatello, Robert M. Freund, and Fabian Pedregosa, *Stochastic frank-wolfe for constrained finite-sum minimization*, 2020.

[RR09] Aaron Roth and Tim Roughgarden, *Interactive privacy via the median mechanism*, 2009.

[RSPS16] Sashank J. Reddi, Suvrit Sra, Barnabas Poczos, and Alex Smola, *Stochastic frank-wolfe methods for nonconvex optimization*, 2016.

[SSMS22] Pierre Stock, Igor Shilov, Ilya Mironov, and Alexandre Sablayrolles, *Defending against reconstruction attacks with rényi differential privacy*, 2022.

[SZ12] Ohad Shamir and Tong Zhang, *Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes*, 2012.

[SZ13] ———, *Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes*, Proceedings of the 30th International Conference on Machine Learning (Atlanta, Georgia, USA) (Sanjoy Dasgupta and David McAllester, eds.), vol. 28, Proceedings of Machine Learning Research, no. 1, PMLR, 17–19 Jun 2013, pp. 71–79.

[TGTZ15] Kunal Talwar, Abhradeep Guha Thakurta, and Li Zhang, *Nearly optimal private lasso*, Advances in Neural Information Processing Systems (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[VTB+20] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Zhiwei Steven Wu, *New oracle-efficient algorithms for private synthetic data release*, 2020.

[WCX19] D. Wang, C. Chen, and J. Xu, *Differentially private empirical risk minimization with non-convex loss functions*, 36th International Conference on Machine Learning, ICML 2019, vol. 2019-June, 2019, Cited By :3, pp. 11334–11343 (English).

[YGFJ17] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha, *Privacy risk in machine learning: Analyzing the connection to overfitting*, 2017.

[YKK+22] Da Yu, Gautam Kamath, Janardhan Kulkarni, Tie-Yan Liu, Jian Yin, and Huishuai Zhang, *Per-instance privacy accounting for differentially private stochastic gradient descent*, 2022.

[YSZ+22] Dayong Ye, Sheng Shen, Tianqing Zhu, Bo Liu, and Wanlei Zhou, *One parameter defense – defending against data inference attacks via differential privacy*, 2022.

[ZMX+20] Qiuchen Zhang, Jing Ma, Yonghui Xiao, Jian Lou, and Li Xiong, *Broadening differential privacy for deep learning against model inversion attacks*, 12 2020, pp. 1061–1070.