Differential Privacy and its applications in Machine Learning

Student Name: Vraj Thakkar
Enrollment ID: **202103052**
B. Tech. Project (BTP) Report
BTP Mode: ***On Campus***
Dhirubhai Ambani University(DAU)
Gandhinagar, India
202103052 [at] daiict.ac.in

Mentor's Name: Prof. Rachit Chhaya
Dhirubhai Ambani University (DAU)
Near Indroda Circle
Gandhinagar 382007, India
rachit_chhaya [at] daiict.ac.in

***Abstract*—Modern applications often require the collection and use of an individual's private data. This data is used to train machine learning models or get some statistical inference, often requiring high-performance accuracy. Using regular techniques for data analysis can result in privacy leakage, where an adversary can devise methods to infer private data of individuals. Stronger privacy notions allow more information gain to be used for various machine learning tasks and also allow larger participation of individuals in providing their private data. Differential Privacy is a state-of-the-art notion of privacy that has gained traction in recent years. Here we study differential privacy in detail and focus on its application in a machine learning setup.**

***Index Terms*—Differential Privacy, Privacy Mechanisms, Machine Learning Algorithms.**

I. Introduction

Privacy has been increasingly discussed in the age where data has become the most important thing. Privacy is also one of the most misunderstood concepts amongst the masses. When we talk about privacy violations, we should ask the question that what constitutes privacy? Privacy is a guarantee offered by a stakeholder to an individual while using his/her data. The most common example is recommendation systems; companies like Netflix, Meta, and Google use private surfing data of individuals to suggest better recommendations or ads, products that not only result in increasing profits but also create a better experience for the user. Privacy violation is when an unauthorized person can access private data, or any data that is guaranteed to be used privately. If someone can break into Netflix's recommendation system and see the preferences of a particular individual, then that would be accurately called a privacy violation. However, if a person is publicly voting and commenting on movies on IMDb and someone traces that information to gain some information about the individual, like ethnicity or political stance, then that is not a privacy violation because the user agreed to be a part of public data. Here is when we would like to talk about one of the most significant examples of privacy violations of the Netflix recommendations systems. In 2006-2009, they hosted a challenge with a prize of 1,000,000 USD to make the best recommendation system. The dataset they released consisted of [(anonymized) user ID, movie ID, rating, date]. However, Narayan and Shmatikov [19] were able to do a linkage attack between this data and the public IMDb data and reveal the true identity of the people in the dataset. This led to a lawsuit against Netflix for not preserving the user data as they were required to do so by law.

Several similar attacks happened, and another most discussed example in the community is of US census data reconstruction. Abowd [20] performed an attack internally where, given the aggregated statistics of the microdata, they were able to reconstruct the microdata exactly for 46 percent of the population, and allowing errors in age of ±1 year, for 71 percent of the population. Linking this with commercial databases allowed them to correctly re-identify over 50 million individuals by name. This led to the use of differential privacy in the construction of the 2020 US census.

The increasing versatility of LLMs has resulted in questions of what data was used to train them, It has often been claimed that proprietary data has been used to train these models without content. As the data becomes intractable, there is an increasing possibility of an adversary constructing attacks to gain training data, which might also reveal private information of individuals that it might have captured from some corner of the internet. Here's another example from another celebrated paper, The Secret Sharer [14], where they experimented with these concepts. They experimented in a language model setting by adding a canary like the sentence, Rohin's credit card number is 123-344-009 different number of times and they tried to see if the model memorized it and it memorized the text increasingly with the number of times it was present in the data, to an extent such that if it was given initial phrase then it could spit out the exact credit card number. This is a serious privacy breach and the paper talks about how the notion of differential privacy helps prevent these attacks.

II. Differential Privacy

A. Pure Differential Privacy

Most data science settings have a trusted curator who collects data from individuals and a data analyst who uses the curated data to gain information. In this setting, Differential Privacy has become an obvious standard for reasoning about information leakage. It is a guarantee/promise made by the trusted curator to an individual that allows no information gain to any adversary, whether the individual chooses to participate in giving the data or not. And, as the adversary cannot find out whether the individual's data has been used or not, the user's privacy is protected by plausible deniability. More formally, Differential Privacy(DP) over the probability of the outputs of algorithms as

Definition 2.1: A mechanism $M : \mathcal{X}^n \to \mathcal{V}$ is ϵ-indistinguishable if for all pairs of neighbouring datasets X and X' and all $W \subseteq \mathcal{V}$ we have

$$Pr(M(X) \in W) \leq e^{\epsilon} Pr(M(X') \in W)$$

The above definition holds for specific privacy guarantees known as pure differential privacy. The interesting thing to note here is how the difference in the distribution is bounded over the outputs. This formulation of the distance between the probability distributions is more stringent as it is a worst-case analysis of the difference in the distributions. We know that $e^{\epsilon} \approx 1 + \epsilon$; hence, the difference between the two probability distributions is bounded in a multiplicative sense. [2] discusses what are the implications of using an additive term(hence total variation distance) as the difference in the distribution, showing that for small differences, multiple compositions of the mechanisms might lead to unintuitive results, and for larger values there is an increase in the possibility of information leakage. This is the worst-case guarantee on the output change by changing one example of data, and it guarantees that the output of the algorithm will not change drastically. It should come as no surprise that most algorithms for adding the guarantees of differential privacy will require randomization. With DP, we are not trying to make any difference in the hypothesis of the data analyst; rather, we are just guaranteeing to the participating user that their participation has been abstracted away from the analyst, and his guess of you being in the dataset is as good as random. Now let's see a simple example on how to use this definition with an example, but first we need to define the sensitivity of the function.

Definition 2.2: Let $f : \mathcal{X}^n \to \mathbb{R}^k$. The ℓ_p-sensitivity of f is

$$\Delta_p^{(f)} = \max_{X,X'} \|f(X) - f(X')\|_p$$

where X and X' are neighbouring databases. Here, p depends on a technicality, mainly on what distributions we choose and how we prove their privacy bounds.

Definition 2.3: Let $f : \mathcal{X}^n \to \mathbb{R}^k$. The Laplace mechanism is defined as

$$M(X) = f(X) + (Y_1, \ldots, Y_k)$$

where the Y_i are independent Laplace (Δ_1/ϵ) random variables.

Let us say we have the task of counting queries, $f = \frac{1}{n}\sum_{i=1}^{n} X_i$ where $X_i = \{0, 1\}$. This is a simple application of the above definition where $k = 1$. It is easy to see that $\Delta_1 = 1/n$. Output of our algorithm would be $\tilde{m} = f(X) + \gamma$, where γ is Laplace $(1/\epsilon n)$. If we define $m = f(X)$, we have that $\mathbf{E}[\tilde{m}] = m$, since $\mathbf{E}[\gamma] = 0$. Computing the variance, we have $\mathrm{Var}[\tilde{m}] = \mathrm{Var}[\gamma] = O\left(1/\epsilon^2 n^2\right)$, and using Chebyshev's inequality gives that $|\tilde{m} - m| \leq O(1/\epsilon n)$ with high probability. Now let's show that this is ϵ-DP in the worst case.

Theorem 1: The Laplace mechanism is ϵ-differentially private.

Proof. Let X and X' be any neighbouring databases, differing in any one entry. We let $p_X(z)$ and $p_Y(z)$ be the probability density functions of $M(X)$ and $M(X')$ evaluated at a point $z \in \mathbb{R}^k$. To prove differential privacy, we will show that their ratio is bounded above by $\exp(\epsilon)$, for an arbitrary choice of z and neighboring X and Y.

$$\begin{aligned}
\frac{p_X(z)}{p_Y(z)} &= \frac{\prod_{i=1}^{k} \exp\left(-\frac{\epsilon|f(X)_i - z_i|}{\Delta}\right)}{\prod_{i=1}^{k} \exp\left(-\frac{\epsilon|f(X')_i - z_i|}{\Delta}\right)} \\
&= \prod_{i=1}^{k} \exp\left(-\frac{\epsilon\left(|f(X)_i - z_i| - |f(X')_i - z_i|\right)}{\Delta}\right) \\
&\leq \prod_{i=1}^{k} \exp\left(-\frac{\epsilon\,|f(X')_i - f(X)_i|}{\Delta}\right) \\
&= \exp\left(\frac{\epsilon \sum_{i=1}^{k} |f(X)_i - f(X')_i|}{\Delta}\right) \\
&= \exp\left(\frac{\epsilon\|f(X) - f(X')\|_1}{\Delta}\right) \\
&\leq \exp(\epsilon)
\end{aligned}$$

In this case, we see that Δ_1 was necessary to prove the privacy guarantee and hence we define sensitivity in the first norm. There are two main properties of pure differential privacy that are very crucial and simple. Let's look at these properties as theorems without proving them.

Theorem 2: Let $M : \mathcal{X}^n \to \mathcal{V}$ be ϵ-differentially private, and let $F : \mathcal{V} \to \mathcal{Q}$ be an arbitrary randomized mapping. Then $F \circ M$ is ϵ-differentially private.

We looked at the example of the attack on Netflix data. This post-processing property guarantees that such attacks are not possible if the data has been protected via the guarantees of differential privacy. Next we look at group privacy where

instead of the neighbouring dataset differing in just one instance, we have neighbours differing in exactly k instances.

Theorem 3: Let $M : \mathcal{X}^n \to \mathcal{V}$ be an ϵ-differentially private algorithm. Suppose X and X' are two datasets which differ in exactly k positions. Then for all $W \subseteq \mathcal{V}$, we have

$$\Pr[M(X) \in W] \leq \exp(k\epsilon) \Pr\left[M\left(X'\right) \in W\right]$$

The proof for this argument is quite trivial, and we can see that in some sense our definition allows graceful degradation of privacy in this setting. These kinds of guarantees are especially important when we have data from different users, and each user is allowed to have multiple instances. In those cases, while providing a user-level DP-guarantee, we would like to exploit this theorem. Next, we see one of the most important implications of pure DP, which allows us to use it in practical settings.

Theorem 4: If $M = (M_1, M_2.....M_k)$ is a sequence of any ϵ-DP mechanisms then M is $k\epsilon$ - Differentially Private.

This theorem is very important and describes the privacy guarantees offered by composing multiple private algorithms on a dataset when you are outputting the dataset after each iteration. We can see that if I am using some randomized mechanism to noise a value of a user in a dataset, then by applying k such iterations and averaging the results, I can get a pretty good estimate of the original value, which increases in accuracy with k, and hence the privacy loss increases.

B. Approximate Differential Privacy

There are a few caveats with using the definition of pure differential privacy, mainly that it is a very stringent requirement and might result in very high utility loss due to the significant tails of the Laplacian distribution. We know that the Gaussian distribution is famous for its tighter tails(which will result in less noise addition and hence more utility), but using it will result in a relaxation of the definition of pure differential privacy. Let's first look at what privacy loss distribution is.

Definition 2.4: Let Y and Z be two random variables. The privacy loss random variable $\mathcal{L}_{Y\|Z}$ is distributed by drawing $t \sim Y$, and outputting $\ln\left(\frac{\Pr[Y=t]}{\Pr[Z=t]}\right)$. If the supports of Y and Z are not equal, then the privacy loss random variable is undefined.

We can immediately see that the probability of the value of privacy loss distribution over all outputs $X = \ln\left(\frac{\Pr[Y=t]}{\Pr[Z=t]}\right)$ for pure differential privacy is bounded by ϵ as

$$\Pr(X \geq \epsilon) = 0$$

However, in the case of approximate differential privacy, we have some non-zero δ such that

$$\Pr(X \geq \epsilon) = \delta$$

.

Definition 2.5: An algorithm $M : \mathcal{X}^n \to \mathcal{V}$ is (ϵ, δ)-differentially private (i.e., it satisfies approximate differential privacy) if, for all neighbouring databases $X, X' \in \mathcal{X}^n$, and all $W \subseteq \mathcal{V}$,

$$\Pr[M(X) \in W] \leq e^{\epsilon} \Pr\left[M\left(X'\right) \in W\right] + \delta$$

We should mention that δ has to be very small, a common notion is $\delta \ll 1/n$, where n is the number of instances in the dataset. This suggestion comes from a worst-case analysis where we are outputting a record with probability δ. Even though δ is the failure probability, our mechanisms are usually not designed to violate privacy completely when that event occurs. For example, the Gaussian mechanism has some value of ϵ such that the mechanism is (ϵ, δ) with probability δ, and hence any catastrophic failure is avoided. We would like to suggest this excellent vlog [6] for further reading.

Like the Laplacian mechanism is used for pure differential privacy, we have the Gaussian mechanism for creating algorithms with (ϵ, δ) approximate differential privacy. We use the ℓ_2 sensitivity in this case to facilitate the proof of privacy, which reduces the noise addition by a factor of $\mathcal{O}(\sqrt{d})$ where d is the dimension of the output of the function. The tighter tails of the Gaussian distribution also ensure that too much noise is not added(for the same variance as the Laplace mechanism) and which also adds to the higher utility of the mechanism. Another important thing to consider is the composition, because of all these nice properties of the tails of the Gaussian mechanism, we can get a tighter composition bound than the basic composition for pure differential privacy with an order of $\mathcal{O}(\sqrt{k})$ instead of k. The Gaussian mechanism is defined below.

Definition 2.6: Let $f : \mathcal{X}^n \to \mathbb{R}^k$. The Gaussian mechanism is defined as

$$M(X) = f(X) + (Y_1, \ldots, Y_k)$$

where the Y_i are iids from $\mathcal{N}\left(0, 2\ln(1.25/\delta)\Delta_2^2/\epsilon^2\right)$

We do not provide a proof of privacy for the result, but we refer the reader to [3]. Next up, we would like to talk about the properties of approximate differential privacy. The post-processing property holds as it is in pure differential privacy. The group privacy theorem holds with a different expression when we track the failure probability, but we leave the reader to refer to [3] for a better read. Lastly, we give the most important result from the composition of Gaussian mechanisms.

Theorem 5: For all $\epsilon, \delta, \delta' > 0$, let $M = (M_1, \ldots, M_k)$ be a sequence of (ϵ, δ)-differentially private algorithms, where the M_i 's are potentially chosen sequentially ano adaptively. Then M is $(\tilde{\epsilon}, \tilde{\delta})$-differentially private, where $\tilde{\epsilon} = \epsilon\sqrt{2k\log\left(1/\delta'\right)} + k\epsilon\frac{e^{\epsilon}-1}{e^{\epsilon}+1}$ and $\tilde{\delta} = k\delta + \delta'$.

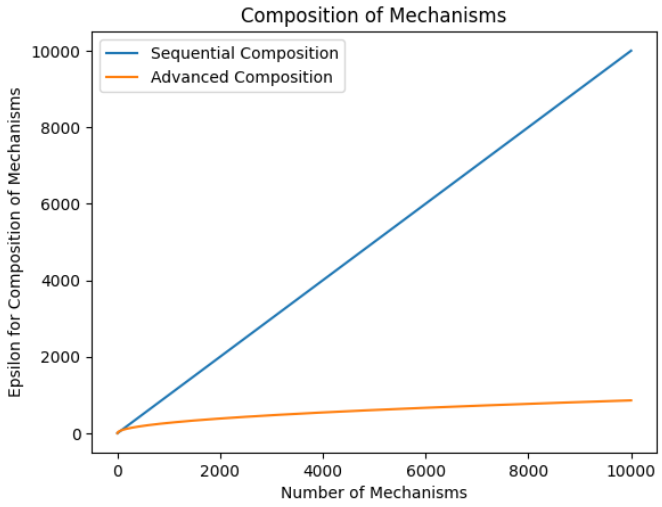

Fig. 1. Advanced Composition vs Basic Composition

There is a lot of literature on composition, which is driving the use of differential privacy. Moreover, there are other notions of differential privacy, like Renyi DP [4] and Concentrated DP [5]. These notions not only give better compositions but also degrade gracefully, avoiding catastrophic privacy failures.

C. Private Selection of Objects

We have studied mechanisms for privatizing numeric queries, but what if we are to select some objects in the privatized setting? A simple example would be median selection; we know that outputting the median exactly might result in outputting the data point, so we want a mechanism to do this with the guarantees of differential privacy. We talk more about this in the next section. To our rescue, we have the exponential mechanism. In this setting, we have the dataset $\mathcal{X}^n$, which is private, the set of objects H(public) where you will select an $h \in H$ such that a score function $s : \mathcal{X}^n \times H \to \mathbb{R}$. We define the sensitivity of the score function as

$$\Delta s = \max_{h \in H} \max_{X, X'} |s(X, h) - s(X', h)|$$

Then, the exponential mechanism is defined as

Definition 2.7: The exponential mechanism M_E on inputs X, H, s, selects and outputs some object $h \in H$, where the probability a particular h is selected is proportional to $\exp\left(\frac{\epsilon s(X,h)}{2\Delta}\right)$.

We do not give the proof of privacy for the result, but it can be found in [3]. There are various applications of this mechanism apart from median selection, but we do not describe those here and suggest the reader to [7]. It is also important to note that this mechanism only works for discrete objects and would require good discretization if used in the continuous domain. In the next section, we will use these mechanisms and guarantees to do statistics estimation privately.

III. Private Statistics Estimation

There has been a lot of work in this area, but in this section, we just explore the most intuitive and basic techniques for private statistics estimation using the definitions from the previous section and do not discuss the current research trends in this paradigm.

A. Median

If we are taking a sorted array $\mathcal{X}$ as an input and $\backslash$ is the size of our dataset $\mathcal{X}$ then the score function for median selection is defined as

$$s(X, h) = |rank_x - n/2|$$

and $\Delta s(X, h) = 1$ because any neighbouring dataset we have

$$|s(X, h) - s(X', h')| = ||rank_x(r) - n/2| - |rank_{x'}(r') - (n \pm 1)/2||$$

Let's take, for example, we have the array $A = [1, 2, 3, 4, 5]$ and let $A' = [2, 3, 4, 5, 6]$, then $s(A, 3) = 0.5$ and $s(A', 3) = -0.5$ and the difference $|s(A, 3) - s(A', 3)| = 1$. So now given an array, all we have to do is find the score of each number and convert them into probability by adding all the scores and using the value as the denominator, and make a random sampling to pick one number. Values closer to the median will have a high probability of getting picked up.

B. Histogram

Let's say we have some data and we want to arrange it in some categories and output a histogram of the data. For simplicity, say we are monitoring the age of people into three categories: 20-30, 30-40, 40-50. With this we know that there are going to be some frequencies associated with each category $(f_{20-30}, f_{30-40}, f_{40-50})$. We also know that changing one datapoint will result in adding one to the frequency of one interval and subtracting one from another. This will effectively result in the sensitivity being 2. In other words the ℓ_1 difference between the vectors is 2, for example $a = (10, 23, 21)$ and $b = (11, 23, 20)$, $||a - b||_1 = ||(1, 0, -1)||_1 = 2$ remember the number of people participating in the survey remain the same just the configuration of one person changes another possible one and we look at all such individual changes to define the highest sensitivity of the function. Now we can add noise proportional to $Laplace(2/\epsilon)$ to each bin and get differentially private results. Of course, the new readings might not be integers, and we should use this histogram as a probability distribution to generate synthetic data and then release the readings if we want to do it right. This also implied that we don't need to add noise proportional to the number of bins, but the error that occurs increases with the number of bins logarithmically. For the utility of the mechanism, refer to lecture 4 [8].

C. Mean

There has been some attention in the area of private mean calculation in academic circles, mainly because it is hard to define a sensitivity bound in mean calculation for most tasks. Most of the literature assumes that the data comes from some Gaussian distribution and that the mean $|\mu| < R$, then guarantees some utility for the algorithms. The setting is trivial; we are getting some samples from a distribution and calculating the mean of those samples, but we want to release this with the guarantees of differential privacy. The obvious initial method would be to bound the data within a reasonable range because it is coming from a Gaussian distribution, which is supposed to have most of the data lying nearer to the mean. Once we have the clipped data, we can define the sensitivity with respect to this interval and just add noise to the output mean. In this case, we would like to bound the number of samples we have to do to provide a utility bound, and they are linear w.r.t the bound over mean, which means if we don't have a good enough estimate of the mean already, we're not going to get a good estimate easily. There is this work [9] which brings the sampling rate in logarithmic terms w.r.t the bound on mean. The idea is to start with a noisy histogram of the dataset with bin size $\mathcal{O}(logn)$, select the bin with the highest count, and output the new interval for the mean containing the two neighboring bins, as the mean is bound to lie in these three bins. Now we can just output the mean with noise proportional to this interval. For a more formal and mathematical understanding, we would like the reader to go to [9]. Finally, we would like to describe recent work *CoinPress*, which iteratively improves the confidence interval and gives good bounds. Unlike the previous methods, it works even in the multivariate case. The idea is initially we clip the data within some interval, now find the mean for this data privately, now we can theoretically estimate the intervals around this mean such that most of the data lies within this interval. Doing this procedure a few times(2-4) results in a good estimation of the mean [10]. We would want the readers to note that there are three types of noise that these methods incur: noise due to the randomness of sampling, noise due to privacy addition, and noise due to clipping. Recent methods propose differentially private sampling algorithms that give much better utility by foregoing the last two noise terms [11]. These methods can have various implications in making differentially private statistics estimations and adding differential privacy to the machine learning pipeline. There are more efficient tools for histogram calculation, in particular, the sparse vector technique in [3] gives more intuitive and tight bounds, it also hides the bins with very less frequencies, below a threshold.

IV. Private Machine Learning algorithms

Now we talk about how differential privacy is used in the machine learning paradigm. We should first mention here that we are not going to talk about how differential privacy protects individual information in different settings, more specifically, what kind of attacks differential privacy protects against. That field of study is known as Differential privacy auditing, and there has been interesting research in that field as well. DP-auditing is all about forming attacks like membership inference attack, attribute inference attack, etc., to check (ϵ, δ) bounds, and usually these bounds are way better than our guaranteed bounds through the composition mechanism. For our discussion, we will provide algorithms with the guarantee of differential privacy, and that guarantee will promise data anonymity. There are different tools for data pre-processing, like DP-based quantile estimation, which can be used for removing outliers, and algorithms for DP-based Principal Component Analysis, which would potentially help in feature selection. Due to the time constraint, we were not able to look at these. Hence, we do not talk about differentially private data pre-processing methods and assume that we have the data ready to be used directly in the machine learning model, so we walk right into how differential privacy is added to machine learning models. Let's first look at what types of differential privacy guarantees we can give.

- Local DP - In this setting, we do not trust any central curator, and the adversary has access to the output of a user, as it is already transformed using DP mechanisms. This guarantee is one of the strongest as it requires that the output of each user be almost indistinguishable, which might result in a serious utility drop.
- Central DP - It is also known as the trusted aggregator DP and is one of the most common settings where DP is used. Here, it is assumed that the adversary only has access to the outputs of the models as given by the trusted aggregator.
- Distributed DP - Distributed DP seeks to cover the utility of central DP without any trusted aggregator by doing the computations in a 'secured box', which is protected enough that not even the companies using its output can breach it to get user data.

Next, we look at places in the machine learning pipeline where we can add differential privacy. Through the post-processing guarantees of differential privacy, we have the following choices.

- Input Data - It is one of the hardest places to add DP because defining a privacy unit is tough in most cases. Due to the post-processing property of DP, any training of the data or statistics estimation will intrinsically be differentially private.
- Training process- This is one of the most common places where DP is added; there are methods like gradient perturbation and objective perturbation that fall into this category. In the cases where the threat model requires us to share the model weights, this method is useful as we don't have to go through the pain of adding DP guarantees to the input data and get good results.
- Prediction - This is only possible when we only the output is supposed to be visible only to the adversary. Defining the sensitivity of the output function can be challenging, and hence, we might have to use some heuristics.

In most machine learning algorithms, it's easier to add mechanisms in the training process. We now look at different algorithms to provide differential privacy guarantees in logistic regression and linear regression.

A. Logistic Regression

We assume that the reader is familiar with the details of logistic regression and start with the simple algorithm, which provides differentially private logistic regression by adding noise to the obtained weight vector after normal logistic regression. Let's see a simple algorithm given in [12]

Algorithm 1: A simple algorithm for DP-based logistic regression

1) Compute W^*, the classifier obtained by regularized logistic regression on the labelled examples $(x_1, y_1), \ldots, (x_n, y_n)$.

2) Pick a noise vector γ according to the following density function:
$$h(\gamma) \propto e^{-\frac{n\epsilon\lambda}{2}\|\gamma\|}.$$
To pick such a vector, choose the norm of γ from the $\Gamma\left(d, \frac{2}{n\epsilon\lambda}\right)$ distribution, and the direction of γ uniformly at random.

3) Output $W^* + \gamma$.

We know the sensitivity [12] of the logistic regression function, and we can just use that to add noise to the obtained weight vector and guarantee differential privacy. This is a very simple algorithm, but adding noise to the weight vector directly after training might result in greater utility loss. Hence, the authors of [12] propose another algorithm where we do objective function perturbation and then use the noisy objective function to train on the data and get results.

Algorithm 2: Objective perturbation

1) We pick a random vector b from the density function $h(b) \propto e^{-\frac{\epsilon}{2}\|b\|}$. To implement this, we pick the norm of b from the $\Gamma\left(d, \frac{2}{\epsilon}\right)$ distribution, and the direction of b uniformly at random.

2) Given examples $x_1, \ldots, x_n$, with labels $y_1, \ldots, y_n$ and a regularization constant λ, we compute
$$W^* = \text{argmin}_w \frac{1}{2}\lambda W^T W + \frac{b^T W}{n} + \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-y_i W^T x_i}\right)$$
.

The loss function of this method is convex and very similar to the initial logistic regression problem, so the running time of the algorithm does not change very much. It can be shown that the privacy guarantee of this method doesn't depend on λ like the previous method, which improves the utility of this algorithm. It is also shown in [12] that the second

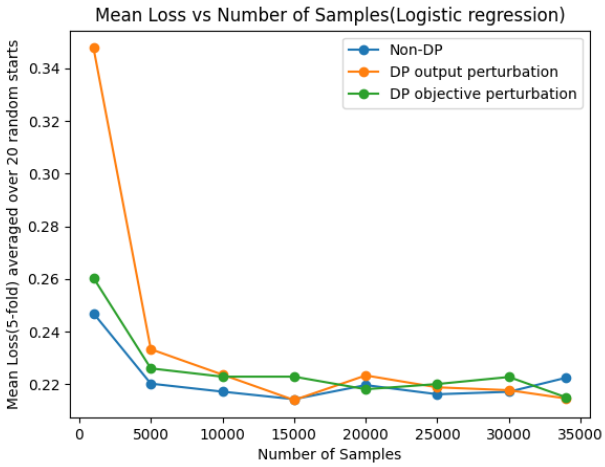

Fig. 2. Comparison of loss of different methods for logistic regression with the number of samples

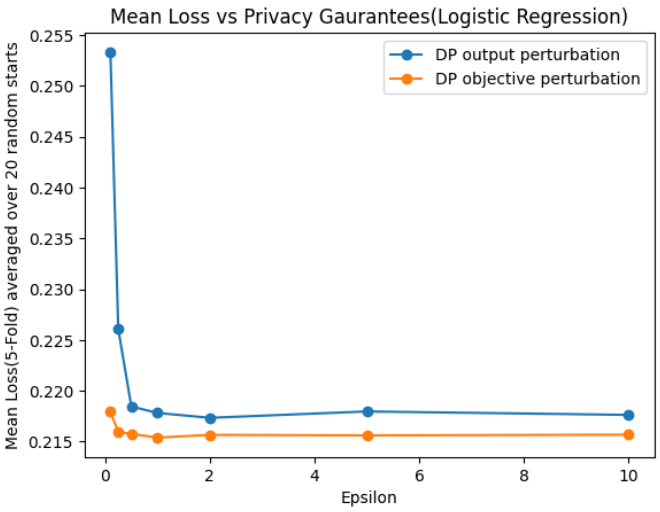

Fig. 3. Comparison of different methods for privacy parameter and loss for different methods in logistic regression case

algorithm has much greater performance guarantees than the first algorithm. The method of objective perturbation can be used for other convex optimization problems as well, which satisfy certain properties as given in [12]. As the paper [12] used synthetic datasets, we used real-life datasets to see the results of the above-described methods. We are using the standard UCI credit card fraud dataset for our implementation. [18]

We can see that the objective perturbation method performs almost as good as the non-private case. It would also be great to look at how the algorithms perform by increasing the value of the privacy parameter. Here, also with our experiments, we agree that objective perturbation is better than output

perturbation as it consistently gives less loss with an increase in noise parameter.

B. Simple Linear Regression

There is a lot of literature in the paradigm of differentially private linear regression, but there are certain caveats regarding multivariate linear regression. We know that given multiple features, one important thing is to select the features for applying linear regression. Feature selection would mean we calculate statistics such as correlation, which itself would lead to some privacy leakage. There are methods for DP-based principal component analysis, but we do not explore those here and just give algorithms for the case of simple linear regression case, where we have just one feature and one corresponding numeric label. Here, we have the following setting for simple linear regression.

We have the normalized data $\{(x_i, y_i)\}_i^n \in ([0,1] \times [0,1])^n$ and the model as $y = \alpha x + \beta$, also this means that the solution for the problem is $\hat{\alpha} = \frac{\text{ncov}(\mathbf{x},\mathbf{y})}{\text{nvar}(\mathbf{x})}$ and $\hat{\beta} = \bar{y} - \hat{\alpha}\bar{x}$.

Now, let's look at a simple algorithm for providing differential privacy guarantees in this model.

Algorithm 3: Noisy-Stats

1) Sample $\gamma_1 \sim \text{Lap}\,(0, 3\Delta_1/\epsilon)$, $\gamma_2 \sim \text{Lap}\,(0, 3\Delta_2/\epsilon)$

2) $\tilde{\alpha} = \frac{\text{ncov}(\mathbf{x},\mathbf{y})+\gamma_1}{\text{nvar}(\mathbf{x})+\gamma_2}$

3) $\Delta_3 = 1/n \cdot (1 + |\tilde{\alpha}|)$

4) Sample $\gamma_3 \sim \text{Lap}\,(0, 3\Delta_3/\epsilon)$, $\tilde{\beta} = (\bar{y} - \tilde{\alpha}\bar{x}) + \gamma_3$

Of course, we make sure initially that $\text{nvar}(\mathbf{x}) > 0$, this algorithm is on similar lines to the first algorithm of logistic regression, hence with a similar argument that because it adds noise directly to the weight parameters, it losses its utility significantly. Next, we look at a robust method for finding the parameters privately.

Algorithm 4: DP-TheilsenMatch

1) Calculate the slope of all pairs of points and choose the median of the slopes privately using the exponential mechanism with parameter ϵ/k where $k = n - 1$.

2) Pick one point (x_k, y_k) privately using the exponential mechanism with score $s(X, m) = -|y_k - mx_k|$ and write b as $b = y_k - mx_k$.

We see that DP-Theilsenmatch consistently performs better than Nosisy-Stats and is a more robust method. All the above-described algorithms give $(\epsilon, 0)$ differential privacy. There exists a DP-TheilsenKmatch algorithm where instead of choosing just one point, we pick up k points and then choose b which minimizes $(y_k - mx_k)^2$; however, here we have to pick the median with $(\epsilon/k, 0)$ privacy. We should mention that this method is more robust, i.e. works well in a maximum number of settings with high utility, which was also seen in our experiments.

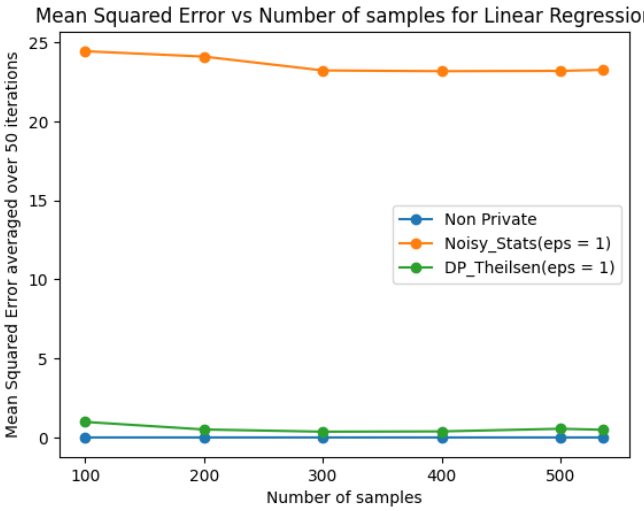

Fig. 4. Loss vs Number of Samples for Linear regression task with different methods

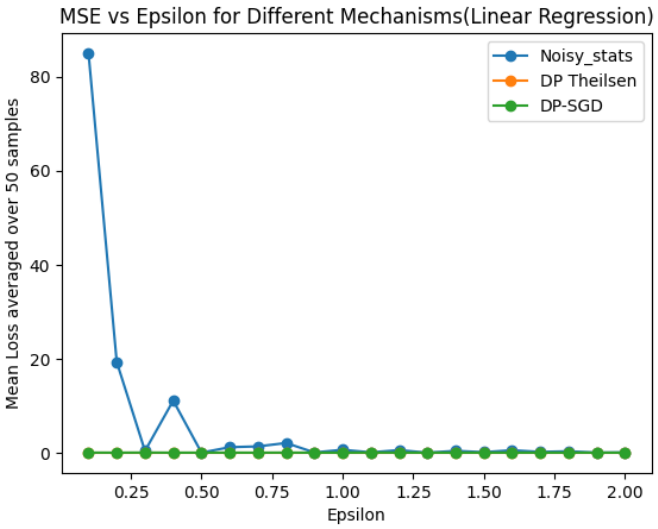

Fig. 5. Loss vs ϵ for Linear regression task with different methods

V. Differentially Private-Stochastic Gradient Descent(DP-SGD)

This is one of the most celebrated algorithms in the differential privacy literature and has seen increasing use cases with neural networks being used everywhere. [14] paper also mentions that differential privacy is the only guarantee that has performed well for preserving privacy in language models. There are private algorithms and utility bounds in the case of convex machine learning [16], but in the case of neural networks, such bounds are not possible due to the

non-convex setting. Let's discuss the algorithm.

Algorithm 5: DP-SGD

1) Parameters: Noise rate(σ), Clipping norm(C), Batch size(L), Learning rate(l).
2) Take each point with probability L/N where N is the total number of samples.
3) Compute the gradient for each point and clip it.
4) Add each gradient term and add Gaussian noise according to σ.
5) Perform gradient step according to the learning rate.

In the case of a convex optimization problem we do not have to clip the gradients as we have that the function is L-Lipschitz, which directly bounds the gradient, and we can use that as a sensitivity parameter to noise the gradients. Also, we choose a batch to perform gradient descent rather than just using a single point, as in the convex optimization case, the reason for this is that non-convex training is already quite unstable, and using just one point might make it more unstable. Now, let's discuss the privacy and utility of this algorithm and how to use it.

The idea is that clipping reduces or normalizes the contribution of each example in the gradient descent and hence does not let any example contribute immensely in the training procedure, which will in turn make it hard to trace any single example. If more than one similar sample is present in the data, then it can reduce the privacy guarantee of the algorithm for that sample hence, it is suggested that the samples in the dataset be unique. But clipping the noise reduces the utility of the algorithm significantly, and on top of that, we are also adding Gaussian noise. Using Gaussian noise, we can use the advanced composition method to cumulate the privacy loss in each iteration and output the final privacy guarantee. But for a larger number of iterations, the reported privacy guarantees will be useless. For example if each iteration is noised with (ϵ, δ) guarantees then with advanced composition for 1000 iterations of DP-SGD we will have approximately $(31\epsilon, 1000\delta)$ which are not good privacy guarantees, so if we give better accounting methods, we might be able to get good privacy guarantees with less noise and hence increased utility. Hence, the paper [15] gives the method of moments accountant. We are not going to discuss the details of the algorithm here and refer the reader to [15]. Moments accountant is a data-independent method, and a convenient feature is that by simply tracking the number of steps taken, the sampling probability, the noise level, and the clipping norm, we can compute the overall privacy loss. Lastly, we would like to mention a special point about privacy amplification by sampling, the idea is that the sampling procedure itself introduces this uncertainty of whether a point was used in the iteration for calculating the gradient or not. Hence, there is some privacy gains due to sampling as well. We must mention that using DP-SGD

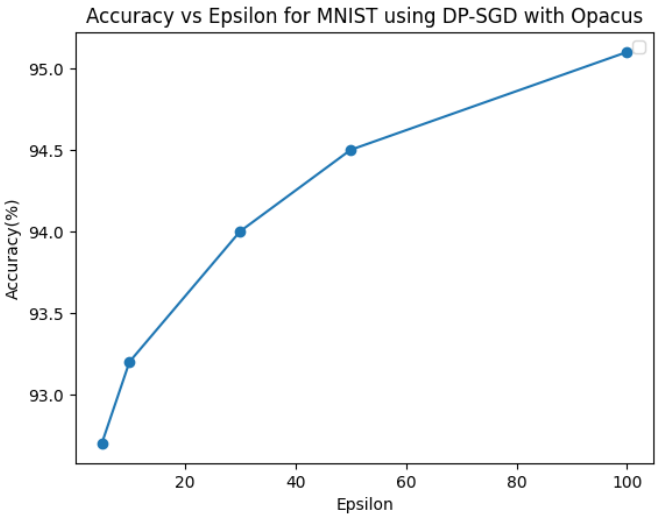

Fig. 6. Simple Neural Network with 3 fully connected linear layers(about one Lakh parameters)

results in a serious utility drop and hence its use remains limited and we therefore we explore more robust methods for each machine learning model.
During training, it is required to calculate per-example gradients, which can be very slow as they cannot leverage the use of GPUs for fast computation. Also, increasing the batch size requires an increase in the running memory required for gradient storage and computation. There are a lot of details around the implementations using DP-SGD, and we couldn't fit all those in the document, so we refer the reader to sections 4 and 5 of [17]. We ran our experiments using the PyTorch Opacus implementation of DP-SGD, and we implemented the algorithm for previous tasks of logistic regression and linear regression as well, and obtained the following results. However, with the same apparatus, we can get an accuracy of about 97.65 % without privacy. We also experimented with the previous datasets and methods. Here are the comparisons

DP-SGD performs well in the linear regression case; however, in the logistic regression case, it was hard to bring it to give good results, and it performed as good as random guessing. With this, we want to highlight that there is a huge amount of computational power and time required for DP-SGD, and even then it might result in a huge loss in utility; hence, it should not be used as a go-to method for all machine learning tasks.

VI. Real World Deployments

Companies like Google and Apple have been at the forefront of adapting differential privacy in their deployments. Differential privacy was used to identify the movement of individuals in COVID-19. In 2023, Google reported that the language models in all of the deployments of Gboard use differential privacy. Federated learning setups are used to learn out of the

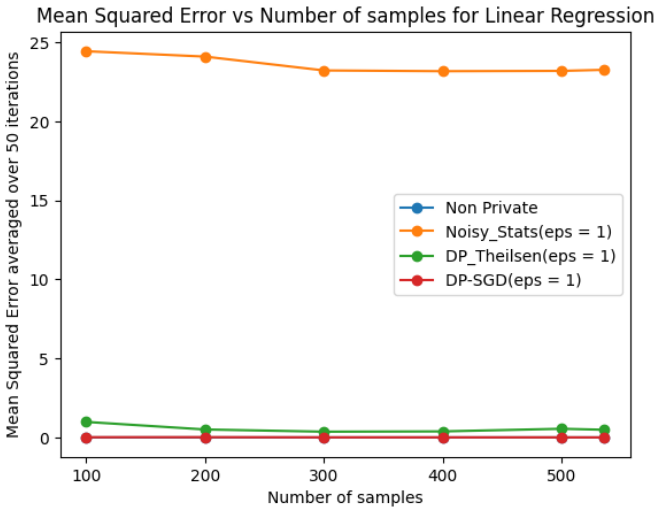

Fig. 7. Linear regression with different methods

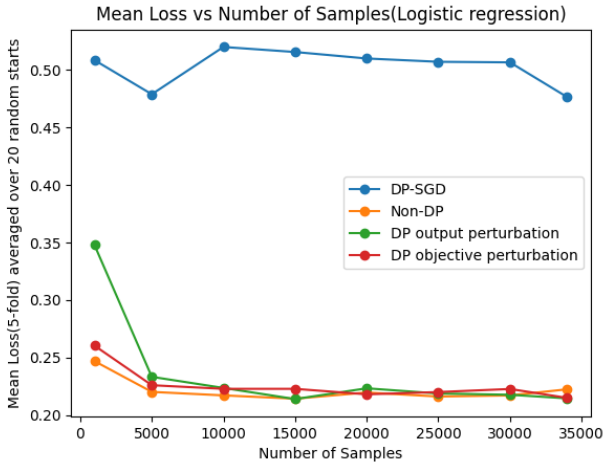

Fig. 8. Logistic regression with different methods

distribution of words. Google Trends uses differential privacy to select which queries to proactively show on the website. There are various deployments of differential privacy(local DP) in Apple that collect data from iOS and macOS and improve their systems, resulting in a better user experience. QuickType suggestions, Safari Energy Draining Domains, and Safari Crashing Domains are a namely a few deployments by Apple. Facebook deploys differential privacy in collecting data when a user shares a URL, which can help Facebook to detect when some unauthorized URLs are being shared and block the attempts.

VII. Future Directions

The next immediate thing that we would like to explore is federated learning. It is a fascinating field of study that is being used extensively in recommendation systems these days. We just scratched the surface in reading about the math behind different composition mechanisms and would want to spend more time reading about PLD accounting, Renyi DP-based accounting, etc. There are certain other notions of Differential Privacy, like Concentrated DP and Renyi DP, which we did not have the time to study in great detail. Of course, we could have studied and implemented more algorithms in the machine learning arena, but our idea was to get a flavor of how DP is used in these algorithms and not cover the DP-fied version of every machine learning algorithm. But more of these algorithms can be found in this survey [21]. There are a lot of questions that we can still ask about how to create attacks that violate data privacy. Creating attacks is hard but there have been a lot of attacks studied in the subfield of differential privacy auditing. We would like to explore that to get an exact idea of what kind of threats we are facing in different models. This can lead to creation of finer mechanisms which potential require less constraints and in-turn give more utility.

VIII. Learning Outcomes

1) I studied an entirely new subject from scratch using research papers and textbooks, which gave me a good research experience.
2) I implemented various papers and carefully tried to reconstruct their results on real-life datasets.
3) I used previous knowledge of the subjects I studied, like approximation algorithms, machine learning, and used several of their concepts to bridge my knowledge to this new field.
4) I used good software programming skills to implement the results from different papers.
5) I wrote this document such that it can appeal to everyone and bridge my knowledge of various papers that I read during my study.

IX. Contributions

1) Although differential privacy is largely studied and researched in the academic community, there is a strong gap when it comes to its applications in the industry.
2) We aim to provide a survey of research on various trends and techniques in the field and serve this document to bridge the gap, which can result in wider adoption of Differential Privacy.
3) This document also discusses privacy utility trade-offs of various algorithms specific to certain machine learning tasks.
4) We also look at some of the latest techniques coming out in the literature and discuss their drawbacks and use cases. We aim to work extensively on this topic further in developing a more insightful document about the math behind differential privacy.
5) Engineers can use this document as a guide while working with differential privacy, and students can use this document to gain significant knowledge about the

field and be research-ready, and a researcher can use this document to see various celebrated algorithms that are currently being used in differentially private statistics estimation and machine learning.

X. References

References

[1] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography, TCC '06*, pages 265–284. Springer, 2006.

[2] Salil Vadhan. The complexity of differential privacy. In Yehuda Lindell (Ed.), *Tutorials on the Foundations of Cryptography*, chapter 7, pages 347–450. Springer, 2017.

[3] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[4] Ilya Mironov. Rényi differential privacy. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF)*, pages 263–275, 2017.

[5] Cynthia Dwork and Guy N. Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016. https://arxiv.org/abs/1603.01887

[6] Clément Canonne, What is δ, and what δDifference does it make? *DifferentialPrivacy.org*, 2021. https://differentialprivacy.org/flavoursofdelta/

[7] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS '07, pages 94–103, Washington, DC, USA, 2007.

[8] Gautam Kamath, Lecture Notes, Algorithms for Private Data Analysis, Course, Fall 2020. http://www.gautamkamath.com/CS860-fa2020.html

[9] Vishesh Karwa and Salil Vadhan. Finite sample differentially private confidence intervals. In Proceedings of the 9th Conference on Innovations in Theoretical Computer Science, ITCS '18,

[10] Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan Ullman. Coinpress: Practical private mean and covariance estimation. arXiv preprint arXiv:2006.06618, 2020

[11] Balle, B., Barthe, G., and Gaboardi, M. Differentially Private Sampling from Distributions. In Proceedings of the 36th International Conference on Machine Learning (ICML),2019

[12] Kamalika Chaudhuri, Claire Monteleoni, Privacy-preserving logistic regression, Advances in Neural Information Processing Systems 21 (NIPS, 2008).

[13] Daniel Alabi, Audra McMillan, Jayshree Sarathy, Adam Smith, and Salil Vadhan, Differentially private simple linear regression, 2020 https://arxiv.org/abs/2007.05157

[14] Nicholas Carlini, Chang Liu, Ulfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In 28th USENIX Security Symposium, USENIX Security '19, pages 267–284. USENIX Association, 2019.

[15] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM Conference on Computer and Communications Security, CCS '16, pages 308–318, New York, NY, USA, 2016.

[16] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science, FOCS '14, pages 464–473, Washington, DC, USA, 2014. IEEE Computer Society.

[17] Natalia et al. , How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy, arxiv 2023. https://arxiv.org/pdf/2303.00654

[18] Yeh, I. (2009). Default of Credit Card Clients [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C55S3H.

[19] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In Proceedings of the 29th IEEE Symposium on Security and Privacy, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society

[20] John Abowd. Tweetorial: Reconstruction-abetted re-identification attacks and other traditional vulnerabilities, April 2019.

[21] Zhanglong Ji, Zachary C. Lipton, Charles Elkan, Differential Privacy and Machine Learning: a Survey and Review, arxiv, 2014.