Certifiably Accurate Private Data Release with Generative Adversarial Networks

Michael Fine

A thesis submitted for the degree of
Bachelors of Arts

April 2020

Contents

Chapter 1

Introduction

Society is caught in a vise. The exponential growth in the power and ubiquity of computing devices has enabled the collection and analysis of data at an unprecedented scale. This Cambrian explosion in data collection promises enormous benefits across commercial, scientific, and policy fields. Unfortunately, this collection and analysis of increasingly-personal data has also proved to be a grave threat to individual's privacy.

Consider the infamous Netflix Prize. In 2006, Netflix announced a $1 million dollar prize to any researcher that could improve its movie recommendation algorithm [Loh10]. To aid research, they released a dataset of its users movie ratings. Sensitive to the privacy implications of revealing a user's private viewing habits, Netflix attempted to ensure privacy by removing all identifying information from the data release. However, Narayanan and Shmatikov [NS07] demonstrated how this anonymization could easily be broken by matching records with publicly available movie reviews on the IMDB website. This led to widespread public outrage as well as a class-action lawsuit and investigation by the FTC, forcing Netflix to cancel the challenge [Loh10].

This is only one in a growing collection of privacy failures stemming from well-meaning attempts to learn from personal data. Data releases from public health records [Swe02] to personal censuse responses [Abo18]. These examples illustrate the difficulty of balancing the gains from mining of personal data with the privacy risks such releases can present.

Differential privacy presents one way out of this morass. Introduced by Cynthia Dwork, differential privacy provides a rigorous definition of privacy, and shows how one can still learn useful information about a population under this constraint [DR13]. Intuitively, differential privacy guarantees that *the inclusion of an individual's information in a data release will not significantly affect any analysis*

performed on that dataset.

Due to its robust guarantees and practical usability, differential privacy has seen wide adoption in the literature, and is increasingly being deployed to protect privacy in real world settings. The most common, and easiest setting is the *interactive model*. In this setting, analysts can repeatedly ask the curator of a sensitive dataset *queries* about the dataset, such as the average value of a particular column. The curator replies to each query with an approximately correct answer, to ensure privacy will still allowing the analysts to learn useful information.

While simple and practical, in many ways the interactive model is a poor fit for the settings where we would like to ensure privacy. Each query in the interactive model reveals a quantum of information about a user, which limits the number of queries that can be answered while maintaining a fixed privacy budget. Further, the interactive model only allows analysts to interact with the dataset through the bottleneck of asking a query to a remote curator and waiting for a response. This breaks decades of tooling and habits built around being able to iteratively explore the entire dataset. Finally, as noted by Hsu, Roth, and Ullman [HRU13], the queries asked by an analyst may be sensitive, and analysts may not want to reveal their thinking to a third-party curator.

Instead, we may desire a *non-interactive* data release, where the curator generates in a differentially private manner a synthetic database with the property that the result of any query over the synthetic database is close to the result of that query over the true dataset. Unfortunately, a fundamental law of differential privacy proves that it is impossible to release a dataset that accurately answers all possible queries while providing *any* level of privacy [DR13]. However, Blum, Ligett, and Roth [BLR11] showed that for any dataset, there exists a differentially private version of it that can accurately answer *exponentially many* queries. This problem is known as the *query release problem*, or often simply synthetic data generation.

Even if there always exists such a database, a string of computational hardness results show that finding such a dataset subject to differential privacy can be prohibitively difficult [UV11]. Ullman [Ull12] showed how privately releasing data that is accurate for polynomially many queries drawn from a very simple family of queries, two-term conjunctions, can take time linear in the size of the data universe, and therefore exponential in the dimension. In virtue of this, one thought has been to apply the remarkable practical success of deep learning in efficiently solving problems that are also hard in the worst case. If Generative Adversarial Networks (GANs) can learn to generate realistic faces, perhaps they can also generate tabular data under the constraint of differential privacy.

While there has been some progress on this front, a major issue remains: deep learning rarely comes with any guarantees of accuracy. This may be fine for most machine learning applications, where typically the cost of a misclassification or a poor sample is low. The primary purpose of synthetic data, however, is to

enable analysts to learn novel and robust information from the data. It's simply unacceptable, for example, to expect social scientists to publish analyses based on fake data that lacks any guarantee of accuracy. To enable broad usage of synthetic data, we need better heuristics to avoid worst-case hardness bounds. But as equally importantly, we need to come up with verifiable heuristics, that we let us be confident in the validity of the resulting data.

This, then is the central focus of this thesis:

> Can we use GAN heuristics to *efficiently* generate private synthetic data, while still providing accuracy guarantees if the GAN training succeeds?

We answer this question in the (conditional) affirmative. We introduce a new algorithm, QueryGAN, that can efficiently generate private synthetic data with guaranteed accuracy for a number of useful classes of queries that we were previously unable to do efficiently.

1.1 Structure of the Thesis

This chapter gives an overview and motivation of the problem, and surveys related work.

Chapter Two provides a detailed introduction to many of the concepts used in this thesis, including differential privacy, the query release problem, two-player games, neural networks, GANs, and online learning.

Chapter Three shows how to connect the GAN objective to the query release problem. It then introduces the main contribution of this work, QueryGAN. QueryGAN shows how, given access to 1) a provably optimal online algorithm for training a discriminator, and 2) a heuristic algorithm for training a generator, we can privately generate synthetic data with accuracy guarantees.

Chapter Four presents two such optimal discriminator algorithms -- an efficient one for all queries approximated by a shallow neural network, as well as an inefficient one for any arbritrary class of queries.

In Chapters Five and Six, we turn our attention to the practical considerations. The prior results eliminate many of the theoretical hurdles to provably accurate synthetic data. However, at the end of the day this approach is predicated on the hypothesis that machine learning/GAN methods will in practice be able to solve problems that are hard in the worst case. Towards that end, in Chapter 5 we introduce a generator architecture tailored to the synthetic data problem. Then, in Chapter 6 we empirically evaluate the efficiency and accuracy of our approaches for a number of datasets and query classes, comparing them to other means of private synthetic data generation.

Finally, Chapter Seven concludes by summarizing our results, discussing limitations, and presenting possible directions for future work. In particular, we discuss:

1. A central challenge in generating synthetic data -- the impossibility of (guaranteed) global non-convex optimization -- and whether in this context that barrier can be circumvented in practice.
2. Whether the GAN framework, as opposed to other heuristic or deep learning approaches, is best suited to the private data release problem.

1.2 Related Work

There has been a significant amount of work on the problem of generating private synthetic data that accurately answers statistical queries [BLR11; DRV10; GRU11; UV11]. Several approaches are able to generate private synthetic data that accurately answer exponentially many queries. Unfortunately, these all have runtime at least linear in the size of the data universe, and thus exponential in the dimensionality. Indeed, Ullman and Vadhan [UV11] showed that, subject to standard cryptographic assumptions, this exponential runtime is necessary in the worst case for most natural families of queries.[1]

In light of this worst-case hardness, there has been some work on improving the runtime in practice. Hardt, Ligett, and Mcsherry [HLM12] (following Hsu, Roth, and Ullman [HRU13]) showed how a differentially private form of multiplicative weights could be used, but still required manipulating objects linear in the size of the data universe. Gaboardi et al. [Gab+14] viewed the query release problem as a zero-sum game between a data player that sought to generate realistic looking data, and a query player that attempted to find the worst performing query. Their algorithm, DualQuery, used commercial integer programming heuristic optimizers to efficiently solve this problem, requiring much less space and (in practice) less time than the multiplicative weights method. They showed that, while DualQuery doesn't always successfully find a solution, when it does the outputted dataset closely approximates the true dataset.

Recently, Neel, Roth, and Wu [NRW18] provided a general framework for using heuristic optimizers to solve hard query release problems. Treating the optimizers as black box oracles, they described an oracle efficient algorithm for generating synthetic data accurate for any set of queries that had what they termed a *separator set* . Unfortunately, the probabilistic and black box nature of the oracles required a runtime proportional to $1/\delta$ to ensure differential privacy. This $1/\delta$ factor was necessary to compensate for the fact that heuristic oracles had a small

[1] Note that these hardness results only apply to synthetic databases, where the rows of the synthetic database are of the same type as the rows of the input database. There has been active work in efficiently generating synopses – arbitrary data structures that can accurately compute the answer to exponentially many queries [DR13; DRV10]. Unfortunately, there also exist hardness results for differentially privately generating synposes – see [UV11]

probability of failure. The very fact that the oracle failed leaked information about the private data, requiring many passes over the dataset to reduce this probability to acceptable level δ. When δ is cryptographically small, that could be prohibitive. Regardless, this thesis draws heavily from the framework and theory in [NRW18].

A final approach has been to use deep learning's powerful ability to solve non-convex problems in order to get around the worst-case hardness. In particular, Generative Adversarial Networks (GANs), introduced by Goodfellow et al. [Goo+14] have had remarkable success in generating high-dimensional synthetic data, typically images. Abay et al. [Aba+19] used GANs to generate differentially private synthetic health records, while Xie et al. [Xie+18] expanded on this approach. Finally, Jordon and Yoon [JY19] applied the Private Aggregation of Teacher Ensembles (PATE) framework to tighter bound the sensitivity of the learning process, allowing for significant gains in accuracy.

While often effective in practice, so far no deep learning approaches have been able to provide useful worst-case guarantees on query accuracy. Unlike [Gab+14] or [NRW18], they cannot guarantee that all queries over the synthetic dataset will be within an additive factor of the true dataset.

Towards that end, there has been some progress in provable convergence guarantees for online deep learning. A pair of recent papers have provided an oracle efficient for (non-private) non-convex online learning [AGH19a; SN19]. By training against a noisy sum of historical loss functions to stabilize learning, they were able to prove convergence assuming the oracle succeeds. Less generally, [Grn+17] used a similar historical play approach to solve the online learning problem for GANs. Instead of adding noise to guarantee convergence, the authors showed that when the game is convex with respect to one player (and possibly very non-convex with respect to the second), standard methods from online convex optimization could be used.

1.3 Terms and Notation

Below is a reference of commonly used variables in this thesis.

Symbol	Meaning
ϵ	Privacy Loss
δ	Privacy Probability
$\hat{S}$	Sanitized dataset
S	Sensitive dataset
x	Row of S
$\mathcal{X}$	Data Universe
$\mathcal{Q}$	Query Universe
$\Delta\mathcal{X}$	The set of all probability distributions over $\mathcal{X}$

Symbol	**Meaning**
m	Number of columns of dataset
n	Number of rows in dataset
$V(\cdot, \cdot)$	Value of the game
$R(T)$	Cumulative regret
$W(\cdot, \cdot)$	Wasserstein loss
$\ell(\cdot)$	Loss function
ϕ	Generator parameters
θ	Discriminator parameters
$D_\theta(\cdot)$	Discriminator with parameters θ
$G_\phi(\cdot)$	Generator with parameters ϕ
z	Generator noise sample
σ	Non-linear activation function
η	Learning rate
α	Approximation factor
K	Lipschitz constant
β	Smoothness

Chapter 2

Background

In this chapter we introduce many of the fundamental concepts used throughout the thesis.

2.1 Differential Privacy

Differential privacy has become the de-facto standard to protect the privacy of records in a statistical database. It formalizes the notion of privacy by looking at outcomes over *neighboring databases*. A *database* or *dataset* $S = (x^{(1)}, x^{(2)}, ..., x^{(n)})$ is a multiset of rows x drawn from a universe of all possible rows $\mathcal{X}$. Two databases are said to be neighboring if they differ on a single row.

An *algorithm* or *mechanism* $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ is a randomized function mapping databases S to outputs $\Omega \in Y$.

Definition 2.1.1 (Differential Privacy [DR13])**.** A randomized algorithm $\mathcal{M}$ if for every $\Omega \subseteq \mathcal{Y}$ and for all neighboring databases $S, S' \in \mathcal{X}^n$

$$P[\mathcal{M}(S) \in \Omega] \leqslant e^{\epsilon} \cdot P[\mathcal{M}(S') \in \Omega] + \delta$$

If $\delta = 0$, we say that $\mathcal{M}$ is ϵ-differentially private.

This is the core guarantee of differential privacy, that any computation over your personal data cannot reveal much more information about you than if you hadn't been in the dataset. In particular, the privacy loss parameter ϵ bounds the maximum information that can be leaked.

(ϵ, δ)-differential privacy is weaker than ϵ-differential privacy – it permits the mechanism to violate ϵ-DP with probability δ. Ideally, δ is negligibly small, and certainly $\ll 1/n$ – because a trivial mechanism that chose a row uniformly and

released it in its entirety would satisfy $(\epsilon, 1/n)$-DP. The benefit of this slight relaxation is it allows for a number of useful, efficient mechanisms that do not satisfy ϵ-DP for any ϵ.

A useful property of this definition is that it is immune to post-processing – no computation done without additional private knowledge, over the output of $\mathcal{M}$,can make the output less differentially private.

Theorem 2.1.1 (Post-Processing [DR13])**.** Let $\mathcal{M}$ be a (ϵ, δ)-DP mechanism. Let f be an arbitrary randomized mapping. Then $f \circ \mathcal{M}$ is also (ϵ, δ)-DP.

Moreover, another useful feature is that differentially private mechanisms *compose* well over multiple computations.

Very often, private algorithms often access the same or similar databases over multiple rounds, with the output of each access depending on the prior output. This model is called *k-fold adaptive composition*, and we can more tightly bound the privacy loss of algorithms that access sensitive data in this way.

Theorem 2.1.2 (Advanced Composition [DR13])**.** Let $\mathcal{M}_i$ each be $(\epsilon_i, 0)$-DP for $i \in [k]$ with $\epsilon_i \leqslant \epsilon'$ bounded. Then the composition $(\mathcal{M}_1(S), ..., \mathcal{M}_k(S))$ is $(\sum_{i=1}^{k} \epsilon_i, 0)$-DP, and (ϵ, δ) private for

$$\epsilon = \sqrt{2\log(1/\delta)k}\epsilon' + k\epsilon'(\exp(\epsilon') - 1)$$

for any $\delta \in (0, 1)$.

2.1.1 Mechanisms

Definition 2.1.2 (Global Sensitivity)**.** The global sensitivity of a function $f : \mathcal{X}^n \to \mathbb{R}^k$, is its maximum difference on neighboring databases S, S'

$$\Delta q := \max_{S,S'} ||f(S) - f(S')||$$

We make use of two fundamental mechanisms over real numbers: the Laplace and Gaussian.

Definition 2.1.3 (Laplace Distribution)**.** The *Laplace Distribution* with mean 0 and scale b, denoted $Lap(b)$, has probability density function

$$f(x) := \frac{1}{2b} \exp\left(-|x|/b\right)$$

Theorem 2.1.3 (Laplace Mechanism [DR13])**.** The Vector Laplace Mechanism $\mathcal{M}_L$, which takes a function $f : \mathcal{X}^n \to \mathbb{R}^k$, is defined as

$$\mathcal{M}_L(S, f, \epsilon) := f(S) + (Y_1...Y_k)$$

where Y_i are drawn i.i.d from $Lap(\Delta f/\epsilon)$

The Laplace mechanism is ϵ-DP

The Gaussian mechanism can only guarantee approximate differential privacy, in exchange for better utility.

Theorem 2.1.4 (Gaussian Mechanism [DR13])**.** The Gaussian Mechanism $\mathcal{M}_{\mathcal{N}}$ with parameter σ, unlike the Laplace mechanism, adds noise scaled to $\mathcal{N}(0, \sigma^2)$ to each component of the output.

For $c^2 > 2\log(1.25/\delta)$, the Gaussian Mechanism with parameter $\sigma \geqslant c\Delta f/\epsilon$ is (ϵ, δ)-DP.

2.2 The Query Release Problem

We study the problem of privately generating synthetic data to answer linear queries over a data universe $\mathcal{X}^n$. Formally, a linear query over $\mathcal{X}$ is a function $q : \mathcal{X} \to [0, 1]$. Given a a dataset $S \in \mathcal{X}^n$, we define $q(S)$, the average value of a query over the rows of S

$$q(S) := \frac{1}{n}\sum_{x \in S} q(x) = \mathbb{E}_{x \in S} q(x)$$

Our goal is to produce a synthetic dataset that, for every query in some family of queries, takes approximately the same value as the true dataset.

Definition 2.2.1 ($\alpha-$approximate)**.** We say a synthetic dataset $\hat{S}$ *α-approximates* a true dataset S w.r.t a family of queries $\mathcal{Q}$ if

$$\forall q \in \mathcal{Q} : \quad |q(S) - q(\hat{S})| \leqslant \alpha$$

2.3 Game Theoretic Formulation

Hsu, Roth, and Ullman [HRU13] demonstrate how to formulate the query release problem as a two-player, zero sum game between a discriminator D and a generator G. The generator has an action set $\mathcal{X}$, while the discriminator has an action set $\mathcal{Q}$. The generator aims to output a dataset $\hat{S} \in \mathcal{X}$ that maximally agrees with S, while the discriminator aims to find queries $q \in \mathcal{Q}$ that distinguish $\hat{S}$ and S.

Formally, given a play $x \in \mathcal{X}$ and $q \in Q$, the discriminator gets payoff $V(x, q)$ and the generator gets payoff $-V(x, q)$, where $V(x, q)$ denotes:

Definition 2.3.1 (Payoff)**.**

$$V(x, q) := |q(x) - q(\hat{x})|$$

The goal of both G and D is to maximize their worst case payoffs, thus

$$\max_{q \in \mathcal{Q}} \min_{x \in X} V(x, q) \quad \text{(Goal of } D\text{)} \qquad \textit{and} \qquad \min_{x \in X} \max_{q \in \mathcal{Q}} -V(x, q) \quad \text{(Goal of } G\text{)}$$

If there exists a point (x^*, q^*) such that neither G nor D can improve their payoffs by playing a different move, we call that a *Pure Nash Equilibrium*. Unfortunately, such a pure equilibrium is not always guaranteed to exist (and likely does not in the case of synthetic data generation).

However, the seminal work of [Nas50] showed that there always exists a *Mixed Nash Equilibrium (MNE)*, where the players play *probability distributions* over their action sets, instead of fixed actions.

Let $\Delta(\mathcal{X})$ and $\Delta(\mathcal{Q})$ denote the set of probability distribution over $\mathcal{X}$ and $\mathcal{Q}$. Formally, if G plays a strategy $g \in \Delta(\mathcal{X})$ and D plays $d \in \Delta(\mathcal{Q})$, we define the payoff to be the expected value of a single draw:

$$V(g, d) := \mathbb{E}_{x \sim g, q \sim d} V(x, q)$$

Thus, a pair of strategies $g \in \Delta(\mathcal{X})$ and $d \in \Delta(\mathcal{Q})$ forms an $\alpha-$approximate Mixed Nash Equilibrium if for all strategies $u \in \Delta(\mathcal{X})$ and $w \in \Delta(\mathcal{Q})$

$$V(g, w) \leqslant V(u, w) + \alpha \qquad \textit{and} \qquad V(u, d) \leqslant V(u, w) - \alpha$$

Moreover, [Gab+14] showed how to reduce the problem of finding an $\alpha-$approximate dataset to the problem of finding an an $\alpha-$equilibrium in the query release game:

Theorem 2.3.1. Let (u, w) be the $\alpha-$approximate MNE in a query release game for a dataset $S \in \mathcal{X}$ and a query universe $\mathcal{Q}$. If $\mathcal{Q}$ is closed under negation, then the dataset $\hat{S}$ sampled from u α-approximates S over Q. [Gab+14]

Hence, our task is to provide an algorithm to privately reach an $\alpha-$MNE in the query release game. In the following section, we will provide the background for how this can be done with GANs.

Chapter 3

Machine Learning Background

Before we show how to solve the query release game with GANs, we first provide a background of necessary concepts from machine learning.

3.1 Convexity Primer

The results of this thesis can be understood as reducing the query release problem to a simple function optimization problem. A crucial property in function optimization is convexity:

Definition 3.1.1 (Convexity)**.** A function $f : \mathcal{X} \to \mathbb{R}$ is *convex* iff for all $x_1, x_2 \in \mathcal{X}$ and $t \in [0, 1]$, the following property holds:

$$(tx_1 + (1-t)x_2) \leqslant tf(x_1) + (1-t)f(x_2)$$

Intuitively, this entails that the line segment connecting any two points on the graph of a function lies above the two points. If f is differentiable, we can equivalently express this in terms of its derivative:

$$f(x_1) \geqslant f(x_2) + f'(x_2)(x_1 - x_2)$$

We make use of a few properties of convex functions:

1. If $f(\cdot)$ and $g(\cdot)$ are both convex functions, and g is non-decreasing, then $g \circ f$ is also convex

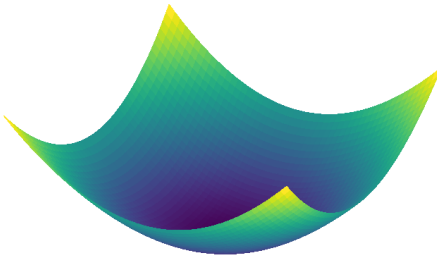

Figure 3.1: A convex function

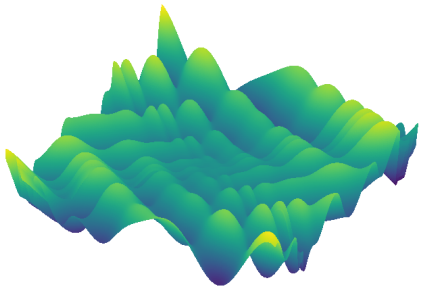

Figure 3.2: A highly non-convex function

2. If f is doubly differentiable, then it is convex iff its second derivative is non-negative over the entire domain.
3. If $f_1, ..., f_n$ are convex functions and $w_1, ..., w_n$ are non-negative real numbers, then the weighted sum $g(x) = w_1 \cdot f_1(x) + ... + w_n \cdot f_n(x)$ is also a convex function.
4. If f is a convex function, then any local minimum is also a global minimum

The final property is crucial. This means (subject to a few conditions) we are able to provably minimize a convex function simply by performing a search for a local minimum – a problem that can be computed easily. It is trivial to verify that a point is a local minimum, and thus for convex we can easily verify that we've reached a global minimum.

In contrast, minimizing arbritrary non-convex functions is NP-Hard in general. Moreover, there's no general efficient algorithm to tell how far a local minimum is from the global minimum of a non-convex function, or even verify if you have reached the global minimum at all. This is especially challenging in this instance, because the building block of GANs, neural networks with more than one layer, are in general non-convex. Yet using neural networks to solve the query release problem via a two player game *requires* finding a global minimizer. We will show in chapter 4 how this can be partially circumvented.

3.1.1 Lipschitz and Smooth functions

Definition 3.1.2 (Lipschitz function)**.** A function f is said to be K-Lipschitz if

$$\|f(x) - f(y)\| \leqslant K\|x - y\|$$

for all x,y in the domain. If there exists some finite K for which f is K-Lipschitz, we say it is *Lipschitz continous*

Intuitively, Lipschitz continuity limits how much a function can change from small changes in its inputs. We make use of the following properties

Lemma 3.1.1 (Lipschitz Properties)**.**

- If f is K_1-Lipschitz and g is K_2-Lipschitz, then $f \circ G$ is $(K_1 \cdot K_2)$-Lipschitz
- For a differentiable, multivariate Lipschitz function f, if $\|\nabla f\|_\infty \leqslant K$ then f is K-Lipschitz with the best Lipschitz constant K.

Definition 3.1.3 (Smooth function)**.** A continuously differentiable function f is β-smooth if the gradient ∇f is β-Lipschitz. That is,

$$\|\nabla f(x) - \nabla f(y)\| \leqslant \beta \|x - y\|$$

3.2 Optimization Oracles for Non-convex Minimization

For problems that are hard in theory but have effective heuristics in practice, we would like to be able to show that if the heuristic succeeds, our algorithm will also succeed. We account for this by defining an optimization *oracle*

Definition 3.2.1 (Offline optimization oracle)**.** An offline optimization oracle $\mathcal{O}$ is a function that takes a loss function f, and returns an x^* in x such that

$$x^* \in \arg\min_{x \in \mathcal{X}} f(x)$$

In practice, most oracles can only find an approximate minimizer, and even then they don't always succeed:

Definition 3.2.2. [Approximate Optimization Oracle] We say that $\mathcal{O}$ is an (α, β)-approximate optimization oracle if with probability $1 - \beta$ over the internal randomness of $\mathcal{O}$, the outputted minimizer x^* is with α of the true minimum:

$$f(x^*) \leqslant \min_{x \in \mathcal{X}} f(x) + \alpha$$

Remark 3.2.1. Note that Definition 3.2.2 does not require that $\mathcal{O}$ only output a point x if it can certify that x is within α of the minimum, and output fail otherwise. Instead, $\mathcal{O}$ is allowed to output a point x without certifying its optimality – it only requires that most of the time, x is in fact approximately optimal. This reflects the reality of many non-convex solvers, that it is impossible to even know if a solution is a global minima, or just a saddle point.

3.3 Online Learning

To efficiently find an equilibrium in the zero-sum GAN game, we draw on results from online learning.

The standard model for machine learning is the *batch*, or *offline* setting. Here, a learner is given a dataset $(x_1, .., x_N)$ sampled iid from the true data distribution, and the objective is to find a function f parametrized by θ that minimizes empirical loss over the samples, for some loss function $\ell_i(\cdot)$:

$$\theta := \arg\min \sum_{i<N} \ell_i(\theta)$$

Contrast this with the *online* setting. In each of T rounds the player must choose an action θ_t, and then a loss function ℓ_t is revealed, after which the player suffers loss $\ell_t(\theta_t)$. The loss functions can be chosen adaptively based on the players prior actions, even adversarially. The player's goal is now to minimize the total *regret*.

Definition 3.3.1 (Regret)**.** The *regret* R(T) measures the cumulative loss of a sequence of decisions θ, compared to the best fixed decision in hindsight.

$$R(T) = \sum_{t=1}^{T} \ell_t(x_t) - \min_{x^* \in \mathcal{X}} \sum_{t=1}^{T} \ell_t(x^*)$$

When a strategy provably leads to regret is sublinear in T, we call that *no-regret*, as average regret $\to 0$ as $T \to \infty$.

As [FS99] first demonstrated, no-regret algorithms are a powerful tool for efficiently solving finding equilibria of zero-sum games:

Lemma 3.3.1. [No-regret [FS99]] Suppose two players G, D are playing a zero-sum game with value V. At each step t G attempts to minimize the online loss function $\ell_t^G(\cdot) = V(\cdot, d_t)$ and D attempts to minimize the opposite $\ell_t^D(\cdot) = -V(g_t, \cdot)$

After T iterations, suppose each player has regret at most α_G, α_D respectively. Let g_t denote $G's$ strategy at time t, and likewise for d_t. Then, the time averaged mixed strategies

$$g^* := \frac{1}{T} \sum_{t=0}^{t} g_t \quad \& \quad d^* := \frac{1}{T} \sum_{t=0}^{t} d_t$$

form an $\frac{(\alpha_g + \alpha_d)}{2}$-approximate mixed Nash Equilibrium.

As a consequence, if G and D both play no-regret strategies, their time averaged mixed strategy will asymptotically converge to an exact MNE.

One approach to regret minimization is at each step to choose the action θ_{t+1} that minimizes the cumulative loss over all past loss functions.

$$\theta_{t+1} = \arg\min_{x \in \mathcal{X}} \sum_{t=1}^{T} \ell_t(x)$$

This approach is known as *Follow-The-Leader*. While natural, this approach is easily exploitable by an adversary, and is not no-regret in general. At a high level, this is because it *overfits* to past outcomes, allowing it to oscillate between suboptimal strategies. In particular, the best bound we can get on the regret is

Lemma 3.3.2 (FTL Regret [Vie+])**.** For any sequence of loss functions $\ell_1 ... \ell_T$, FTL outputs a series of predictors θ_T with regret

$$R(T) \leqslant \sum_{t<T} \ell_t(\theta_t) - \ell_t(\theta_{t+1})$$

Instead, to ensure no-regret, we introduce a regularization term that ensures that $\ell_t(\theta_t) - \ell_t(\theta_{t+1})$ is small.

Definition 3.3.2 (Follow-The-Regularized-Leader (FTRL))**.** Given a regularization function $\| \cdot \|$ and a regularization weight η_T, at each step choose $\theta_{t+1} \in \mathcal{X}$ to minimize the regularized cumulative loss:

$$\theta_{t+1} = \arg\min_{x \in \mathcal{X}} \sum_{t=1}^{T} \ell_t(x) + \eta ||x||$$

Lemma 3.3.3. [**Haz19**] Assume that the losses are convex and K-Lipschitz, and let D be the diameter of the action set $\mathcal{X}$. There is a choice of η such that

$$R(T) \leqslant 2KD\sqrt{2T}$$

Therefore, for Lipschitz and convex losses, FTRL is no-regret.

Note that solving the game using $FTRL$ requires efficiently computing the minimizer of a sum of loss functions. In other words, FTRL reduces the online learning problem to the offline, batch problem. When the losses are convex, this is feasible, as there are efficient algorithms for offline batch minimization. However, in later, non-convex formulations, this becomes trickier.

Thus, by 3.3.1 and 3.3.3, we can find the approximate MNE of a game $V(g, d)$ if both G and D play according to FTRL, so long as V is concave with respect to g and convex with respect to d.

We term this property of V convex-concave:

Definition 3.3.3 (Convex-Concave)**.** A two-player, zero sum game $V(\cdot, \cdot)$ is said to be *convex-concave* if for any fixed g_0, $V(g_0, d)$ is convex in d, and likewise for any fixed d_0, $V(g, d_0)$ is concave in g.

Unfortunately, the query release game V is clearly not convex-concave for arbitrary query classes $\mathcal{Q}$. Recall that the pure strategy form of V is defined as

$$V(g, q) := q(g) - q(\hat{g})$$

Clearly, if $q(\cdot)$ is not a convex function, then V is not convex in g. Likewise, if g's action space $\mathcal{X}$ is a non-convex set (or in later formulations, a set containing non-convex functions), V is not concave in q. As expectations of non-convex functions are also non-convex, this extends to the mixed definition of V as well.

Therefore, FTRL is not a no-regret algorithm for the query release game for arbritrary $\mathcal{Q}, G$. The following sections describe two approaches to enable no-regret solutions to the game: 1) Restricting $\mathcal{Q}$ (but not necessarily G) to convex functions to make the game tractable, and 2) using Follow-The-Perturbed-Leader, a randomized variant of FTRL which was recently shown to be no-regret for non-convex losses, at the cost of looser bounds on regret.

3.4 Achieving Equilbrium via Semi-convex games

Here, we describe semi-convexity, a relaxation of the convexity requirement for two player games. Introduced in [Grn+17], it only requires that the game is convex with respect to one of the players. This makes the no-regret optimization problem tractable, while still permitting a large family of useful games.

Definition 3.4.1 (Semi-convex [Grn+17])**.** A game is semi-convex if for any fixed g_0, $V(g_0, d)$ is convex in d, though not necessarily concave in g.

[Grn+17] shows how this if V is semi-convex, if the convex player plays $FTRL$, this enables the non-convex player to play (non-regularized) FTL and still achieve no regret. Of course, the non-convex player still must solve the underlying offline optimization of FTL. We assume the non-convex player has access to an optimization oracle. This result manages to reduce the problem of online optimization to the offline setting. Though FTL is not no-regret for arbritrary convex loss functions, [Grn+17] take advantage of the *predictability* of the actions of the FTRL

player to show how FTL is no-regret for this specific setup.

Algorithm 1: Non-Private Equilibrium Finding [Grn+17]

Input: game objective V, Steps T, Regularization weight η
Result: Dataset $x \in \mathcal{X}^d$, Accuracy α

for $t \in 1...T$ **do**
  Update D and G according to FT(R)L:
  $\theta_{t+1} \leftarrow \arg_\theta \min \sum_{i<t} \ell_i^D(\theta)$ and $\phi_{t+1} \leftarrow \arg_\phi \max \sum_{i<t} \nabla \ell_i^G(\phi_i)\phi - \frac{\sqrt{T}}{2\eta}\|\phi\|$

  Update losses:
  $\ell_{t+1}^D(\cdot) = V(\cdot, \phi_{t+1})$ *and* $\ell_{t+1}^G(\cdot) = V(\theta_{t+1}, \cdot)$

Output Mixed Strategies $S_1 \sim Unif\{\theta_1...\theta_T\}$ *and* $S_2 \sim Unif\{\phi_1...\phi_T\}$

Lemma 3.4.1 ([Grn+17])**.** Suppose G and D play according to FTRL and FTL respectively in a semi-convex zero sum game that is convex in D. Then, the time averaged mixed strategies $\left(\frac{1}{T}\right)$ form a α-MNE, where $\alpha = O(T^{-1/2})$.

Algorithm 1 provides an online way to achieve equilibrium by solving a batch minimization problem at each step, *provided the game is semi-convex*. While a number of query release problems can be structured as a semi-convex game, the vast majority cannot. In the following section we introduce FTPL, an oracle-efficient no-regret algorithm for non-convex games.

3.5 Achieving Equilibrium in Non-Convex Games

When the game value $V(\cdot, \cdot)$ is non-convex for both players, FTRL fails to induce the *stability* required to ensure online no-regret. Recently, Agarwal, Gonen, and Hazan [AGH19a]showed how a variant of FTRL is no-regret even for non-convex losses (the bounds were later tightened by [SN19]). Instead, of using a deterministic regularization function $R(x)$ to induce stability, FTPL introduces a random

perturbation γ to the loss at each step.

Algorithm 2: Follow the Perturbed Leader (FTPL) for non-convex losses [AGH19b]

Input: Perturbation Magnitude $\eta \geqslant 0$, Step T, Online series of losses $f_1(\cdot), ..., f_T(\cdot) : \mathbb{R}^d \to \mathbb{R}$)

Result: Online prediction $\theta_1, .., \theta_T$

for $t \in 1...T$ **do**

  Draw random vector $\gamma \sim Exp(\eta)^d$

  Output prediction at time t

$$\theta_t \in \arg\min_x \sum_{i<t} f_i(x) - \gamma^T x$$

Because $FTPL$ is a randomized algorithm, we can only bound the *expected regret*, where the expectation is over the random draw of the algorithm. This can easily be converted into a fixed bound that holds with high probability through the use of the Chernoff bound.

Theorem 3.5.1 (FTPL is no-regret[SN19])**.** Let D be the l_∞ diameter of the action set $\mathcal{X} \in \mathbb{R}^d$. Suppose the losses $f_1, ..., f_T$ are L-Lipschitz. Then the predictions made by FTPL have expected regret

$$\mathbb{E}\left[R(T)\right] = O(\eta d^2 D L^2 T)$$

Thus, for appropriate choice of η, FTPL achieves optimal expected regret $O(\sqrt{T})$.

This theorem, combined with 3.3.1, shows that approximate equiblirium is achieved if both players play FTPL in a non-convex game. Like in Algorithm 1, we assume each player has access to an optimization oracle that can in practice solve the underyling offline minimzation problem.

In the next section we will discuss how if G and D are parametrized by neural networks, and $\mathcal{O}$ is a form of SGD, we can view the query release problem as an instance of training a Generative Adversarial Network.

3.6 Generative Adversarial Networks

Generative Adversarial Networks (GANs), introduced by [Goo+14], are an approach to generative deep learning that has shown remarkable promise in generating realistic samples from complex distributions. In the GAN setup, a generator neural network $G_\phi(\cdot)$ is paired with a discriminator neural network $D_\theta(\cdot)$, with parameters ϕ and θ respectively. The discriminator attempts to distinguish real

Figure 3.3: Photo Realistic GAN samples [BDS19]

samples and generated samples, while the generator aims to generate realistic samples that fool the generator.

Let p_{data} denote the probability distribution from which the finite batch of real data $S := (x^(1), ..., x^(n)) \in \mathcal{X}^n$ was drawn. For instance, in a dataset of images of human faces, p_{data} assigns a high probability to a headshot of a person, while assigning low (or zero) probability to an image of a cat. We also define a random noise distribution p_z, typically a standard gaussian.

The goal is to train a generator network $G_\phi(\cdot)$ whose input is a vector drawn from p_z, and whose output is a sample from a distribution p_g over $\mathcal{X}$, where p_g is the generators' attempt to approximate the true distribution p_{data}. The discriminator $D(\cdot;\theta) : \mathcal{X} \to [0, 1]$ is a function trained to estimate the probability that a sample $x \in \mathcal{X}$ was drawn from the true data distribution rather than p_g.

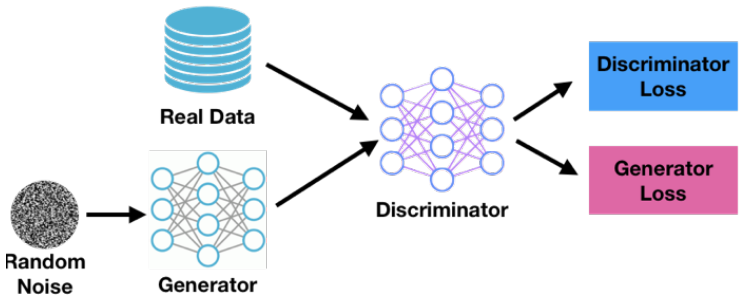

Figure 3.4: The GAN training game

The discriminator and generator are trained simultaneously and adversarially. yielding a two-player, zero-sum game with minimax objective

$$\min_G \max_D V(\theta, \phi) := \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}} \log D_\theta(\mathbf{x}) + \frac{1}{2}\mathbb{E}_{z\sim p_z} \log(1 - D_\theta(G_\phi(z)))$$

However, Arjovsky, Chintala, and Bottou [ACB17] argue that this cost function V_{gan} is not sensible cost function in practice, when the distributions are supported by low-dimensional manifolds. This results in training difficult and convergence failures. Instead, they proposed to use the Earth-Mover, or Wasserstein-1 objective.

Definition 3.6.1 (Earth Mover Distance)**.** The *EM distance* between two distribution p_{data} and p_g is

$$W(p_{data}, p_g) := \inf_{\gamma\in\Pi(p_{data},p_g)} \mathbb{E}_{(x,y)\sim\gamma}|x - y|$$

where $\Pi(p_{data}, p_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively p_{data} and p_g.

While this infinimum is highly intractable to compute, [ACB17] used the Kantorovich-Rubinstein duality [**Vil08**] to show that

$$W(p_{data}, p_g) = \sup_{\|f\|_L\leqslant K} \mathbb{E}_{x\sim p_{data}}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)]$$

where the supremum is over all K-Lipschitz functions. Note that the choice of K is somewhat arbritrary so long as it is finite, for any K-Lipschitz function can be converted into a 1-Lipschitz function by scaling the ouput by $1/K$.

Thus, if our discriminator D is an element of a set of K-Lipschitz functions $\mathcal{D}$, minimizing the following Wasserstein GAN objective also minimizes $W(p_{data}, p_g)$:

$$V(D, G) := \mathbb{E}_{\mathbf{x}\sim p_{data}}[D(\mathbf{x})] - \mathbb{E}_{z\sim p_z}[D(G(z))]$$

Remark 3.6.1. In the Wasserstein GAN, the discriminator is no longer guaranteed to output values in $[0, 1]$, and therefore cannot be interpreted as a probability. For this reason, the Wasserstein discriminator is typically called a *critic* [ACB17].

3.6.1 Enforcing Lipschitz Constraints

Of course, in practice D is not actually optimized over all K-Lipschitz functions. Instead, D is a finite neural network that we train via gradient descent. [ACB17] suggest enforcing the Lipschitz constraint by clipping the weight of D's parameters θ to a fixed interval $[-w, w]$. This ensures that all functions D_θ are K-Lipschitz for a K that depends only on w, and not the particular weights.

Lemma 3.6.1 (Weight clipping implies Lipschitz continuity)**.** Let $D_\theta(\cdot)$ be a l-layer fully connected neural network with input dimension m, and wlog K_σ-Lipschitz activation function ϕ. Suppose the ℓ_∞ norm of the weights $\|\theta\_\infty \leqslant w$ for some fixed w. Then D is K_D-Lipschitz for some finite constant K_D.

Proof. A l-layer neural network D is a composition of functions

$$D = f_l \circ \sigma \circ f_{l-1} \circ \dots \sigma \circ \circ f_1$$

where each $f_i(X) = \theta^T X$. We use the fact that any function with bounded gradient $\|\nabla_X f\|_\infty \leqslant K$ is K-Lipschitz. As $\|\nabla_X(\theta^T X)\|_\infty \leqslant \|\theta\|_\infty = w$, each f_i is therefore w-Lipschitz.

Finally, we use the fact that the composition of two functions that are respectively K_1, K_2-Lipschitz is $(K_1 \cdot K_2)$-Lipschitz. Thus, each layer $\sigma \circ f_i$ is $(K_\sigma \cdot w)$-Lipschitz, and the composition of l layers is $\prod_{i<l}(K_\sigma \cdot w)$-Lipschitz. □

As [ACB17] noted, weight clipping is a blunt means of enforcing a Lipschitz constraint. We can show that for any Lipschitz bound K enforced by a clipping threshold w, there exist parameters θ such that D_θ is K-Lipschitz, but $\|\theta\|_\infty > w$. In fact, Petzka, Fischer, and Lukovnicov [PFL18] proved that in almost all cases, the optimal K-Lipschitz D that minimizes $W(G, \cdot)$ is not in the class of functions that can be generated under the weight clipping constraint.

However, this concern turns out to be less relevant for the problem explored in this thesis. First, our primary result relies on a one-layer discriminator. It is easy to verify that for a single-layer network $\sigma(\theta^T x)$, the weight clipping bound is tight. Moreover, even in the case of deep discriminators there is an independent reason for enforcing weight clipping. We will see in the following sections that $\|\theta\|$ must be bounded both to ensure convergence of the online algorithm, as well as to ensure differential privacy.

Remark 3.6.2. One might worry that the argument for the Wasserstein loss requires a number of unrealistic assumptions that do not actually hold. In particular, the argument assumes that:

1. The class of finite, K-Lipschitz neural networks is sufficient to approximate *all* K-Lipschitz functions
2. Enforcing Lipschitz constraints via weight clipping does not exclude too many valid discriminators
3. The (usually non-convex) discriminator is trained to optimality

However, the contingency of these assumptions is not fatal to our argument. The above arguments aim to suggest that the Wasserstein loss is a practical and efficient objective function for GAN training. Neither the accuracy nor privacy

guarantees rely on any specific properties of the Wasserstein loss besides its structural resemblance to the query release objective. The arguments above simply aim to provide some theoretical understanding of the improved performance of the Wasserstein loss, and how it can be applied here.

3.6.2 Wasserstein GAN Equivalence

In this section, we prove that the Wasserstein GAN objective is equivalent to the query release objective for specific neural networks.

Lemma 3.6.2. [Wasserstein GAN equivalence] Let G and D be neural networks, where D is K-Lipschitz. Define Suppose G and D are trained to an $\alpha-$ mixed equilibrium of the Wasserstein objective with some training dataset S. Then a dataset $\hat{S}$ sampled from G $\alpha-$approximates S for all queries $\mathcal{Q}$ representable by D.

Proof. This proof follows from a simple reduction to Theorem 2.3.1, drawn from [Gab+14]. Theorem 2.3.1 states that

> Let (u, w) be the $\alpha-$approximate MNE in a query release game for a dataset $S \in \mathcal{X}$ and a query universe $\mathcal{Q}$. If $\mathcal{Q}$ is closed under negation, then the dataset $\hat{S}$ sampled from u α-approximates S over $\mathcal{Q}$. [Gab+14]

To apply this theorem to the Wasserstein GAN, we must show that the Wasserstein GAN objective is equivalent to the query release objective, and in addition that $\mathcal{D}$ is closed under negation.

If the activation function ϕ is *odd*, then the fact that the set of weight parameters Θ is closed under negation ensures that $\mathcal{Q}$ is. For non-odd functions, we can simply let $\mathcal{Q}' = \mathcal{Q} \cup \overline{\mathcal{Q}}$, where $\mathcal{Q}$ is the class of negated queries.

We can now show equivalence of the two objectives.

Recall that in the query release game the payoff of V over mixed strategies is defined as

$$\min_{g \in \Delta\mathcal{X}} \max_{q \in \Delta\mathcal{Q}} V(g, d) = |\mathbb{E}_{x \sim g}[q(x)] - \mathbb{E}_{x \sim \mathbb{P}_{data}}[q(x)]|$$

Because Q is closed under negation, we can drop the absolute value around V. Limiting $\mathcal{Q}$ to a family of K-Lipschitz functions, we recover the exact Wasserstein formulation.

□

Importantly, for the Wasserstein GAN this limits us to all queries that are K-Lipschitz over each row. [2]. In this context, that may prove to be infeasibly limiting. Even a simple linear threshold query over a continous domain, q(x) = $\mathbb{1}\{x > 1/2\}$ is not Lipschitz for any finite K. This has not proven to be too onerous of a restriction in prior GAN literature because for visual data, changing 1 pixel intuitively shouldn't change the realism of an image by much, and thus the discriminator has no need to represent queries particular sensitive to individual pixels.

In general, our goal is to find a GAN objective that induces a discriminator function class that contains many queries of interest, while still making training feasible. In later sections we will explore other objectives.

[2] To be clear, this is not the standard Lipschitz condition in differential privacy, where some function $f : \mathcal{X}^{d \times n} \to \mathbb{R}$ is defined over a multi-set of records. Instead, the discriminator (and thus queries) takes a single row $f : \mathcal{X}^d \to \mathbb{R}$. Thus, for f to be 1-Lipschitz, flipping a single *column* of a row $x \in \{0,1\}^d$ can't change $f(x)$ by more than 1.

Chapter 4

QueryGAN

We now have almost all the necessary components for our first result. 3.6.2 reduces the query release problem to the problem of training GANs. 3.4.1 and Theorem 3.5.1 shows how, we can use an optimization oracle as a heuristic to efficiently find the minimum of a semi-convex game and non-convex game respectively. Using stochastic gradient descent over a neural network as our oracle, we can then define an algorithm that is guaranteed to produce an α-accurate synthetic database, conditioned on the oracle succeeding in each round.

However, as noted above, we are unable to certify whether the oracle has successfully found an approximate minimum. Recall that a solution x^* is $\alpha-$approximate if

$$\alpha \leqslant f(x^*) - \min_{x \in \mathcal{X}} f(x)$$

In the case of non-convex gradient descent, without other assumptions it is impossible to in general determine α, because the global minimum $\min f(x)$ could be arbritrarily far from the stationary point x^*. Without this feature, we cannot provide any guarantees regarding the worst case accuracy of a query on the synthetic dataset.

However, while we cannot guarantee (or even certify) convergence to an approximate global minimum in general, we can take advantage of the specific structure of the query release problem. In particular, if D is optimal [3], the global minimum

[3]Note that unlike in many classic GAN proofs [Goo+14], this is not the unrealistic optimality where D has infinite capacity and is optimized in the space of probability density functions. Rather, we merely mean optimal in the space of functions $\mathcal{D}$ that can actually be computed. Because D is designed such that $\mathcal{D}$ contains all the queries of interest, considering this restrictive function class is sufficient for our purposes. By comparison, in the standard GAN setting, where the goal is not just to match a few specific queries, but rather to exactly match the distributions.

of the game value V is known: 0

Lemma 4.0.1. Let d^* be the optimal discriminator. Then, the cumulative regret of the generator's loss $\ell_t = V(d^*, \cdot)$ can be bounded by the empirical sum of the losses:

$$R(T) \leqslant \sum_{t=1}^{T} \ell_t(g_t)$$

Proof. Recall that regret is defined as

$$R(T) = \sum_{t=1}^{T} \ell_t(x_t) - \min_{x^* \in \mathcal{X}} \sum_{t=1}^{T} \ell_t(x^*)$$

To prove the theorem, we show that the term $m := \min_{x^* \in X} \sum f_t(x^*) = 0$ by bounding each term in the sum from above and below.

In the Wasserstein formulation, each $\ell_t(p_g) = W(p_{data}, p_g)$ is a proper distance between probability distributions. By definition, $f_t \geqslant 0$. Moreover, the action where g samples uniformly from the true distribution p_{data} clearly has value at most 0. □

However, given that d^* must be learned in a differentially private manner, we can only guarantee that d^* is α-approximate for some fixed α. In this context, we can bound the regret as:

Lemma 4.0.2. Let D α-approximate the optimal discriminator d^*. Then the cumulative regret of the generators loss $\ell_t^\alpha = V(d_\alpha^*, \cdot)$ can be bounded by

$$R(T) \leqslant (T-1)\alpha + \sum_{t=1}^{T} \ell_t^\alpha(g_t)$$

4.0.1 gives us a framework for obtaining certifiable error guarantees even when using non-certifiable heuristics to train the generator. So long as we can guarantee the discriminator is approximately globally optimal, we can track the total regret by simply summing the generators historical losses.

We formalize this discriminator requirement as:

Definition 4.0.1. We say A is a $(\mathcal{Q}, R(t), \rho)$-online discriminator oracle (ODO) if for any sequence of loss functions fed losses $\ell_1, .., \ell_T$ in an online fashion, after each sample it outputs a function D_t such that with probability rho

$$\sum_{t<T} \ell_t(D_t) - \sup_{q \in Q} \sum_{t<T} \ell_t(q) \leqslant R(T)$$

If A is also $(\epsilon(T), \delta)$-differentially private for the entire sequence of losses $\ell_1 ... \ell_T$, we call it an $(\mathcal{Q}, R(t), \epsilon(T), \delta)$-ODO

Remark 4.0.1. Note that unlike the oracle definitions in section 3.2, the discriminator oracle is an online, rather than batch learner. As shown in section 3.3, we can always convert a batch learner to an online learner using FTRL/FTPL. However, this is an extremely general reduction, and by allowing the discriminator learner to optimize specifically for the online case, we can often obtain better regret bounds.

With this, we have the foundation for our GAN based query release algorithm.

Algorithm 3: QueryGAN

Input: sensitive dataset $S \in \mathcal{X}^n$, $(\mathcal{Q}, R_d(t), \epsilon(T), \delta)$-Online Discriminator Oracle $\mathcal{O}_D$, Offline Generator Oracle $\mathcal{O}_G$, Rounds T, game objective V, output rows N, Privacy Budget $(\epsilon = \epsilon_1 + \epsilon_2, \delta)$, Reporting confidence bound $(1-u) \in (1/2, 1)$, GAN objective $V(\theta, \phi_t)$

Result: Dataset $S \in \mathcal{X}^d$, Accuracy α

Randomly initialize Discriminator θ_0 and Generator parameters ϕ_0
for $t \in 1...T$ **do**
    Obtain discriminator θ_t by passing loss function $\ell_t^d(\cdot) = V(\cdot, \phi_{t-1})$ to $\mathcal{O}_D$

    Let $\ell_t^g(\cdot) = V(\theta_t, \cdot)$
    if *V is convex in θ* **then**
        Let $J_t(\cdot) := \sum_{\tau<t} \ell_\tau^d(\cdot)$ be the FTL objective **??**
    else
        Draw noise $\gamma \sim Exp(\eta_G)$
        Let $J_t(\theta) := \sum_{\tau<t} \ell_\tau^d(\theta) + \gamma^\intercal \theta$ be the FTPL objective 2
    Compute ϕ_t by passing objective J to $\mathcal{O}_G$

Let $k = -\frac{n}{\epsilon_2} \log(1-2u)$
Privately calculate cumulative regret:
$\hat{R} = \sum_{t<T} \ell_t^g(G_t) + Lap(m/\epsilon_2) + k$

for $i \in 1...N$ **do**
    Draw $t \sim Unif([T])$ and $z \sim \mathcal{N}(0, 1)$
    Set $x_i = G_t(z)$

return *Dataset* $\{x_1, ..., x_N\}$, *Approximation factor* $\alpha = \frac{\hat{R} + R_d(T)}{T}$

Remark 4.0.2. As we will prove below, Algorithm 3 shows how we can efficiently generate synthetic data with accuracy guarantees in the GAN framework, provided we have both:

1. A private, online discriminator oracle that can provably train a discriminator to near optimality at each step
2. A heuristic generator oracle that in general can train a generator to produce good samples

The vast literature and experimental evidence on the surprising effectiveness of deep networks suggests that the second problem is feasible with standard tools for training GAN generators. The challenge in later sections is to provide private ODO's for broad classes of useful queries that can be trained in reasonable (typically sub-exponential) time.

Theorem 4.0.1. [QueryGAN Utility] Suppose QueryGAN is instantiated with a $(\mathcal{Q}, R(t))$-ODO, a reporting failure probability $u \in (0, 1/2)$, and an approximate optimization oracle $\mathcal{O}_G$. Let the game objective $V(D, G)$ be the Wasserstein GAN objective w.r.t to the sensitive data $\hat{x}$

Let x, α be the results of running T iterations of *QueryGAN* with the above parameters.

Then with probability u, x $\alpha-$approximates $\hat{x}$ with respect to $\mathcal{Q}$

Proof. This follows directly from 3.4.1 and Lemma Theorem 2.3.1

Because the regret bound is reported privately, we can only upper bound the private regret with high probability.

Lemma 4.0.3. Let $\hat{R}$ be as defined on line 16 of Algorithm 3. Let $R := \sum_{t<T} \ell_t^g(G_t)$ be the unperturbed regret. Then $P(R \geqslant \hat{R}) \leqslant u$

Proof. This follows by basic probability. Let $Y \sim Lap(n/\epsilon_2)$, $Z \sim Exp(\epsilon_2/n)$, $k = \frac{n}{\epsilon_2}\log(1-2u)$ as in Algorithm 3. Then, We can write $\hat{R}$ as $\hat{R} = R + Z + k$.

$$\begin{aligned}
P(R \geqslant \hat{R}) &= P(R \geqslant R + Z + k) \\
&= 1 - \frac{1}{2}P(|Y| \geqslant k) \text{By symmetry of Laplace} \\
&= 1 - \frac{1}{2}P(Z \geqslant k) \text{Exponential is one-sided laplace} \\
&= 1 - \frac{1}{2}(1 - \exp(k \cdot \epsilon_2/n)) \\
&\leqslant u
\end{aligned}$$

□

Applying the above confidence bound to *Algorithm* 3 proves the claim. □

We now will show that given access to a approximate optimization oracle, QueryGAN converges at rate $T^{-1/2}$

Theorem 4.0.2 (Conditional convergence)**.** Suppose $\mathcal{O}_G$ is an (α)-approximate optimization oracle. Then w the average regret $\frac{1}{T}\mathbb{E}\left[R_G(T)\right] \leqslant O(T^{-1/2} + \alpha/T)$.

Proof. There are two cases:

- If $\ell^d(\cdot)$ is non-convex then $\mathcal{O}_G$ plays FTPL. Then the proof follows directly from the bounds given in Theorem 3.5.1.
- If $\ell^d(\cdot)$ is convex, then it can forego any perturbation by playing just Follow the Leader. The *stability* of the actions θ_t of the discriminator ensures the loss functions ℓ_t^g are also stable, obviating the need for explicit regularization. This proof is similar to [Grn+17], but for general predictions as well as those of FTPL, rather than merely FTRL.

Theorem 4.0.3. Assume $\|V\|_\infty \leqslant C$. Let V be K_D lipchitz with respect to the discriminator parameters . Then, if G plays FTL,

$$\mathbb{E}\left[R_G(T)\right] \leqslant \sum_{t<T} K_D \mathbb{E}\left[\|\theta_{t+1} - \theta_t\|\right] + 2C \tag{4.1}$$

$$\leqslant K_D T \mathbb{E}\left[\|\theta_{t+1} - \theta_t\|\right] + 2C \tag{4.2}$$

Proof. The proof of the bound on generator regret is adopted from [Grn+17], modified here to allow the discriminator to play FTPL rather than FTRL. Recall that, per 3.3.2, the regret of follow-the-leader is bounded by

$$R(T) \leqslant \sum_{t<T} \ell_t(\phi_t) - \ell_t(\phi_{t+1})$$

Using the fact that $\ell_t(\phi) = V(\phi, \theta_t)$ is K_D-Lipschitz with respect to θ as well as that the losses are bounded by C, the lemma follows:

$$
\begin{aligned}
R(T) &\leqslant \sum_{t<T} \ell_t(\phi_t) - \ell_t(\phi_{t+1}) \\
&= \left(\sum_{t=1}^{T-1} \ell_t(\phi_t) - \ell_t(\phi_{t+1}) + \ell_{t+1}(\phi_{t+1}) - \ell_{t+1}(\phi_{t+1}) + (\ell_T(\phi_T) - \ell_T(\phi_{T+1})) \right) \\
&= \left(\sum_{t=1}^{T-1} \ell_{t+1}(\phi_{t+1}) - \ell_t(\phi_{t+1}) \right) + \sum_{t=1}^{T-1} (\ell_t(\phi_t) - \ell_{t+1}(\phi_{t+1})) + (\ell_T(\phi_T) - \ell_T(\phi_{T+1})) \\
&= \left(\sum_{t=1}^{T-1} V(\phi_{t+1}, \theta_{t+1}) - V(\phi_{t+1}, \theta_t) \right) + (\ell_1(\phi_1) - \ell_T(\phi_{T+1})) \\
\mathbb{E}\left[R(T)\right] &\leqslant \left(\sum_{t=1}^{T-1} K_D \cdot \mathbb{E}\left[\|\theta_{t+1} - \theta_t\|\right] \right) + (\ell_1(\phi_1) - \ell_T(\phi_{T+1})) \\
&\leqslant K_D \cdot T \cdot \mathbb{E}\left[\|\theta_{t+1} - \theta_t\|\right] + 2C
\end{aligned}
$$

Theorem 4.0.4. Let $\mathcal{M}(\ell_1, ..., \ell_t)$ be an ϵ-differentially private mechanism with neighbour relation $(\ell_{1:t}||\ell_{1:t+1})$. Let $\theta_t = \mathcal{M}(\ell_1, ..., \ell_t)$ be the output of an ϵ-differentially private mechanism, with respect to the sequence of loss functions.

$$\mathbb{E}\left[\|\theta_{t+1} - \theta_t\|_1\right] \leqslant 2\epsilon \cdot \|\theta\|_\infty p$$

Proof. Dwork and Roth [DR13, Definition 10.1] shows that if θ_t and θ_{t+1} are the output of an ϵ differentially private mechanism $\mathcal{M}$ on neighbouring inputs $\ell_{1:t}, \ell_{1:t+1}$, then

$$|\mathbb{E}_{x \sim \mathcal{M}(\ell_{1:t})}\left[x\right] - \mathbb{E}_{x' \sim \mathcal{M}(\ell_{1:t+1})}\left[x'\right]| \leqslant 2\epsilon \mathbb{E}_{x \sim \mathcal{M}(\ell_{1:t})}\left[x\right]$$

By the reverse triangle inequality, linearity of expectation, and the independence of the mechanism's perturbations:

$$\mathbb{E}\left[||x| - |x'||\right] \leqslant 2\epsilon \mathbb{E}_{\theta \sim \mathcal{M}(\ell_{1:t})}\left[x\right]$$

We treat each coordinate θ_t^i as its i.i.d own random variable x in the above equation. Thus,

$$\mathbb{E}\left[\|\theta_t - \theta_t\|_1\right] \leqslant p \cdot \mathbb{E}\left[|\theta_t^i - \theta_t^i|\right] \leqslant 2\epsilon \mathbb{E}_{\theta \sim \mathcal{M}(\ell_{1:t})}\left[\theta\right]$$

□

This proves the first part of the theorem. For FTPL in particular, note that Each step of FTPL with Laplace noise λ is differentially private for $\epsilon := \frac{\|\theta\|_\infty}{\lambda}$. Plugging in ϵ into the above bounds obtains the desired result.

Using this, if we let $\epsilon \leqslant O\left(\frac{K_D\|\theta\|_\infty}{\sqrt{T}}\right) \ll 1$, then G has expected regret $R_G(T) \leqslant O(\sqrt{T} + 2C)$

□

□

Theorem 4.0.5 (QueryGAN Privacy)**.** Suppose QueryGAN is run for T rounds, over a dataset $\hat{x} \in \mathcal{X}^{m\times n}$, with privacy parameters $\epsilon = \epsilon_1 + \epsilon_2$, and a differentially private discriminator oracle $\mathcal{O}_{\lceil}$. Then QueryGAN satisfies ϵ-differential privacy.

Proof. In QueryGAN, the only interaction with sensitive data comes through the discriminator loss function $f(\cdot) = V(G, \cdot)$. Thus, by ensuring the discriminator is differentially private, the generator privacy (and thus the privacy of the ultimate data release) comes via post-processing. However, the regret bounds also depend directly on the sensitive loss function f, and must be independently perturbed to ensure privacy. The total privacy loss of both computations is bounded by the sum of the privacy loss of calculating the regret bound and training the discriminator.

Lemma 4.0.4. The mechanism to report the regret bound is ϵ_2-DP.

$$R \leftarrow \frac{1}{T}\sum_{t\in T} f_t(G_t) + Lap(m/\epsilon_2) + k$$

Proof. The privacy of reporting the regret bound comes from the addition of Laplace noise. Each objective f_i is a statistical query over the same dataset $\hat{x}$. As f is ensured to be $1-$Lipschitz over a single row, then each individual query f_t has sensitivity m. As they are all computed over the same records, the total sensitivity of the average $\frac{1}{T}\sum_i f_i$ is also m. Thus, by standard Laplace mechanism, R is ϵ_2-DP. □

Because $\mathcal{O}_D$ is defined to be a $(\_, \_, \epsilon_1)$-ODO in Algorithm 3, by the definition of an ODO it is ϵ_1 differentially private over T rounds. Thus, via basic composition, we arrive at a total privacy loss of $\epsilon_1 + \epsilon_2 = \epsilon$ as desired. □

Chapter 5

One Layer Discriminator Oracles

In this section we describe a ODO for the class of (K-Lipschitz) one layer neural networks $\mathcal{F}_1$

$$\mathcal{F}_1 := \left\{ f(x) = \sigma(\theta^T x + b) \;\middle|\; \|\theta\|_\infty \leqslant 1 \right\}$$

.

where σ satisfies the following conditions:

Assumption 5.0.1 (Activation conditions)**.** We assume the activation function σ in $\mathcal{F}_1$ is any β smooth, convex, 1-Lipschitz, monotonic activation function with continous second derivative.

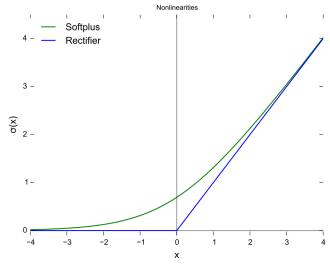

Figure 5.1: The softplus and ReLU activation functions

One such common activation function is *softplus*, a smooth approximator to the ReLU function $\sigma_{ReLU}(x) = \max(0, x)$. The softplus is defined as [Zhe+15]

$$\sigma(x) := \log(1 + e^x) \tag{5.1}$$

Lemma 5.0.1. The softplus function satisfies Assumption 5.0.1.

Proof. The first and second derivatives of softplus are $\frac{e^x}{1+e^x}$ and $\frac{e^x}{(1+e^x)^2}$ respectively.

Note that the first derivative of softplus is the sigmoid function, and thus the first derivative has bound 1. By 3.1.1, that proves *1-Lipschitzness*. Similarly, by the definition of smoothness 3.1.3, function with β-Lipschitz gradients is β-smooth. As sigmoid is 1-Lipschitz, we have that softplus is *β-smooth*.

Softplus is a composition of monotonic functions, and therefore it too is *monotonic*. Finally, as the second derivative is all points differentiable, softplus therefore has a *continous second derivative*. □

Each $f(x) \in \mathcal{F}_1$ is clearly a convex function of x, as both the weighted sum $\theta^T x + b$ and $\sigma(x)$ are convex in x. Because composition of two convex functions is also convex, the overall function is as well.

Algorithm 4 is a simple instantiation of FTPL. At each step t, the algorithm uses any off-the-shelf, non-private convex optimizer, such as gradient descent, to minimize the FTPL objective. Our novel contribution is in showing that the single laplace perturbation ensures *both* no-regret and differential privacy.

Algorithm 4: ODO-1[AGH19b]

Input: Privacy parameter (ϵ, δ), Steps T, Online series of ERM losses $\ell_t(\theta) := \mathcal{W}(D_\theta, G_t)$ for $t \in 1...T$

Result: Online prediction $\theta_1, .., \theta_T$

for $t \in 1...T$ **do**

$$\epsilon_0 := \frac{\epsilon}{\sqrt{2T\log(1/\delta)}} \qquad \lambda := \frac{2\|X\|_\infty}{\epsilon_0}$$

Draw random vector $\gamma \sim Lap(\lambda)^p$

Output prediction at time t

$$\theta_t \in \arg\min_x \sum_{i<t} \ell_i(\theta) - \gamma^T x$$

We will prove the parameters ODO-1 are differentially private. Recall that by post-processing, any dataset generated from those parameters (and no other private data) is also differentially private.

Lemma 5.0.2 (Privacy)**.** The sequence $(\theta_1, ..., \theta_T)$ outputted by Algorithm 4 is (ϵ, δ)-DP

Proof. The difficulty in proving privacy is that unlike in the standard setup of differentially private online learning, where each loss function ℓ_t is assumed to be a single sensitive datapoint, here each loss function depends on the entire dataset. Because of this, we cannot use the standard analyses from private online learning.

Instead, we show that the FTPL objective is equivalent to the batch convex optimization objective $\sum_{i<n} \tilde{\ell}_i(D)$, where each $\tilde{\ell}_i(D)$ depends only on a single record x_i. This lets us analyze the Follow the Leader perturbations in the objective perturbation framework of [CMS].

Let $\ell_{FTPL}(D)$ be the FTPL objective, where γ is a random noise vector.

$$\ell_{FTPL}(D) := \sum_{t<T} \mathcal{L}_t(D) + \gamma^T D = \sum_{t<T} \left(\frac{1}{n} \sum_{i<n} D(x_i) - \frac{1}{m} \sum_{i<m} D(G_t(z_i)) \right)$$

Because the first term $\sum_{i<n} D(x_i)$ does not depend on t, we can swap the order of the summation

$$\ell_{FTPL}(D) = \sum_{i<n} \tilde{\ell}_i(D) + \gamma^T D$$

where

$$\tilde{\ell}_i(D) := D(x_i) - \frac{1}{m} \sum_{j<m} \sum_{t<T} D(G_t(z_j))$$

Note that each $\tilde{\ell}_i$ depends only on the single row x_i, and theore is a convex ERM problem. Thus, the output of one iteration of FTPL can instead be viewed as the output from a differentially private, objective perturbation mechanism. This lets us apply the following lemma from [KST].

Lemma 5.0.3 (Objective Perturbation Privacy [KST])**.** Let $\mathcal{L}(\theta) := \frac{1}{n} \sum_{i<n} \tilde{\ell}_i(\theta)$ be a convex loss function with a continous hessian, let ζ be the upper bound on $\|\nabla \tilde{\ell}_i(\theta)\|_2$, and let and assume the rank of $\nabla^2 \tilde{\ell}(\theta)$ is at most one.

Then the mechanism that outputs the minima of the perturbed objective

$$\min_\theta \frac{1}{n} \sum_{i<n} \tilde{\ell}_i(\theta) + \gamma^T \theta; \quad \gamma \sim p_\gamma(b; \lambda)$$

is ϵ-differentially private when γ is drawn from the distribution with density $p_\gamma(b;\lambda) := \exp\left(\frac{\|b\|_2}{\lambda}\right)$ where $\lambda := \frac{2\zeta}{\epsilon}$

Because D_θ is a convex function of θ, ℓ is also convex. The continuity of the hessian follows from the assumption that the activation function σ has continous second derivatives. We can bound $\|\nabla\tilde{\ell}_i(\theta)\| \leqslant \|X\| = \zeta$. Finally, note that as the above lemma only applies to a single iteration of FTPL, the ϵ is not the total ϵ for T rounds of QueryGAN, but instead ϵ_0, the privacy loss for a single iteration.

Plugging the above values, as well as the constants ϵ_0 and λ from Algorithm 4 into 5.0.3, proves that each iteration is ϵ_0-DP.

Finally, we bound the privacy loss of T iterations of Algorithm 4. Recall that by **??**, running T compositions of an ϵ_0-DP mechanism is (ϵ, δ)-DP for

$$\epsilon = \sqrt{2\log(1/\delta)T}\epsilon_0 + T\epsilon_0(\exp(\epsilon_0) - 1)$$

Assuming that $\epsilon_0 < 1$, we can use that inequality $\exp(\epsilon_0) \leqslant 2\epsilon_0 + 1$ to obtain $\epsilon = \sqrt{2\log(1/\delta)T}\epsilon_0 + 2T\epsilon_0^2$. Plugging our choice of $\epsilon_0 := \epsilon/(\sqrt{2T\log(1/\delta)})$ suffices to prove the result. □

Theorem 5.0.1 (Utility)**.** Algorithm 4 is a $(\mathcal{F}_1, R(T))$-ODO, for

$$R(T) \leqslant O\left(\epsilon\sqrt{T}\|\theta\|_\infty \log^{-\frac{1}{2}}(\frac{1}{\delta}) + \|\theta\|_\infty\|X\|_\infty \log(p) log^{\frac{1}{2}}(\frac{1}{\delta})\right)$$

Proof. We make use of the following theorem

Lemma 5.0.4 ([KV05])**.** For $K - Lipchitz$, convex losses, FTPL instantiated with Laplacian noise with parameter $\frac{1}{\lambda}$ has regret

$$E[R(T)] \leqslant \frac{(1+2K)L_T^*}{\lambda} + 2\|\theta\|_\infty(1 + \log(p)\lambda)$$

where $L_T^* := \min_\theta \sum_{t<T} \ell_t(\theta)$

Plugging in λ lets us obtain the desired bound. □

We have just shown that Algorithm 4 is an a $(\mathcal{F}_1, R(T), \epsilon(T), \delta)$-ODO.

The above lemma, combined with Theorem 4.0.1 shows that QueryGAN instantiated with and α_g-generator oracle, with high probability produces synthetic data that α-approximates $\hat{x}$ for all $q \in \mathcal{F}_1$, where

$$\alpha = O\left(\frac{\sqrt{T} + \alpha + C + \epsilon\sqrt{T/\log(\frac{1}{\delta})} + \|X\|_\infty \log(p) log^{\frac{1}{2}}(\frac{1}{\delta})}{T}\right)$$

While the class of one layer networks $\mathcal{F}_1$ is very limited, it does contain a number of useful query classes. $\mathcal{F}_1$, the set of thresholded linear functions, which can represent, for example, the class of discrete marginals.

5.1 Case study: Marginals

Consider a database that contains records with boolean features such as whether a person is male or female, or whether they have a certain disease. We may be interested in studying some subset of those features. A natural question to ask is how many records in the database have those attributes (eg. "how many men in the database have a heart condition"). This can be captured by a marginal query:

Definition 5.1.1 (Marginal)**.** A marginal $m : \mathcal{X} \to \{0, 1\}$ over a row $x \in \{0, 1\}^p$ is a monotone conjunction, parametrized by some subset S of the input features.

$$m_S(x) = \prod_{i \in S} x_i$$

We extend this to a dataset S of n rows by defining $m(X) = \sum_{x \in X} m(x)$. A k-way marginal is a marginal restricted to k features.

[DR13]

Marginals are a useful way of providing a synopsis of a dataset that still captures complex relationships between features. Producing a differentially private synthetic dataset that agrees with all $k-$way marginals of the true dataset is an extremely well-studied problem in the field, which can be shown to be hard in the worst case [Dwo+09].

However, we can show that if *QueryGAN* succeeds, it is able to match all $k-way$ marginals. This follows from the fact that a linear discriminator can contain all marginals.

This theorem relies on the use of a Rectified Linear Unit activation function

Definition 5.1.2 (ReLU)**.** The ReLU activation function $R(x) : \mathbb{R} \to \mathbb{R}^+ := max(0, x)$

This non-linearity allows us to approximate the nonlinear marginal query with a linear neural network:

Theorem 5.1.1. Let D be single layer discriminator parametrized by θ with a ReLU activation function s.t. $D_\theta(x) = \sigma(\theta^T x + b)$. For any single-row marginal m, there exists θ, b s.t. $D_\theta(x) = m(x)$ for all x.

Proof. This follows from the definition of a marginal. Let m_S be the marginal over the features S. Let $\theta_i = \mathbb{1}_{i \in S}$. Setting $b = 1 - |S|$, it's clear that

$$D_\theta(x) = \begin{Bmatrix} 1, & \prod_{i \in S} x_i = 1 \\ 0, & \text{otherwise} \end{Bmatrix} = m_S(x)$$

□

Theorem 5.1.2. Let x, α be the output of running *QueryGAN* with for T steps. Then x is a private dataset with all marginal counts accurate to within α.

The proof of this theorem follows directly from Theorem 4.0.1 and Theorem 5.1.1.

Chapter 6

Conclusion

In this thesis, we've shown how to apply Generative Adversarial Networks to the query release problem with certifiable accuracy bounds, despite the fundamentally non-convex nature of the problem. This is an important step towards enabling differential privacy to be deployed in more challenging settings, with stronger accuracy requirements and larger datasets. The accuracy guarantees (so far) only hold for one-layer discriminators, but the privacy guarantees hold regardless. This setup alone is powerful enough to allow for certifiable generation of synthetic data that is accurate for all k-way marginals, a strong, if conditional result.

In addition, to the best of our knowledge we are the first to use the recent result per Agarwal, Gonen, and Hazan [AGH19a] that $FTPL$ is no-regret in order to provide an oracle efficient synthetic private data generation framework. Further, some of our proof techniques to show utility for online learning over batch losses in Chapter 4 may be of independent interest.

The largest open question that remains is whether the one-layer discriminator limitations could be relaxed. The introduction of FTPL removes the need to induce a semi-convex game, as in [Grn+17]. The only barrier to more powerful discriminators that better resemble real world GANs is the need for training algorithms that provably converge, in order to allow us to track the generator regret. Unfortunately, training deep neural is necessarily hard in general, but there is very active research into obtaining provable convergence for deep neural net that circumvent those hardness requirements [Li+14; MS18; AXK17]. In particular, training neural networks via greedy boosting, rather than backpropagation is theoretically promising, if not quite practical yet [Bey+15; Bac; Mar19]. This is doubly promising given the effectiveness of boosted decision trees at classifying discrete and tabular data.

In addition, perhaps better guarantees could be obtained with a discriminator

other than a neural network. While it is hypothesized that some of the effectiveness of GANs comes from the symmetry of the discriminator and generator architecture [Sal+16], expanding our function class would allow us to represent many more queries before running into an NP-Hard optimization problem. If we abandon our desideratum that the discriminator be trained in time sub-linear in the size of the query universe, there are a number of existing differentially private algorithms in the literature that could easily be plugged into this framework, such as differentially private multiplicative weights [HR10].

Finally, this framework is no stronger than the ability of the generator to learn distributions. There is much more room for empirical testing, and for tailoring the generator and training procedure for this particular setup, rather than the more typical visual learning problems.

Bibliography

[Aba+19] Nazmiye Ceren Abay et al. "Privacy Preserving Synthetic Data Release Using Deep Learning". en. In: *Energy Transfer Processes in Polynuclear Lanthanide Complexes*. Singapore: Springer Singapore, 2019, pp. 510–526. ISBN: 9789811360480 9789811360497. DOI: 10.1007/978-3-030-10925-7_31.

[Abo18] John M. Abowd. "The U.S. Census Bureau Adopts Differential Privacy". en. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*. London, United Kingdom: ACM Press, 2018, pp. 2867–2867. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3226070.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". en. In: *arXiv:1701.07875 [cs, stat]* (Dec. 2017). arXiv: 1701.07875 [cs, stat].

[AGH19a] Naman Agarwal, Alon Gonen, and Elad Hazan. "Learning in Non-Convex Games with an Optimization Oracle". en. In: *arXiv:1810.07362 [cs, stat]* (May 2019). arXiv: 1810.07362 [cs, stat].

[AGH19b] Naman Agarwal, Alon Gonen, and Elad Hazan. "Learning in Non-Convex Games with an Optimization Oracle". en. In: *arXiv:1810.07362 [cs, stat]* (May 2019). arXiv: 1810.07362 [cs, stat].

[AXK17] Brandon Amos, Lei Xu, and J. Zico Kolter. "Input Convex Neural Networks". In: *arXiv:1609.07152 [cs, math]* (June 2017). arXiv: 1609.07152 [cs, math].

[Bac] Francis Bach. "Breaking the Curse of Dimensionality with Convex Neural Networks". en. In: (), p. 53. arXiv: 1412.8690.

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". en. In: *arXiv:1809.11096 [cs, stat]* (Feb. 2019). arXiv: 1809.11096 [cs, stat].

[Bey+15] Alina Beygelzimer et al. "Online Gradient Boosting". en. In: *arXiv:1506.04820 [cs]* (Oct. 2015). arXiv: 1506.04820 [cs].

[BLR11] Avrim Blum, Katrina Ligett, and Aaron Roth. "A Learning Theory Approach to Non-Interactive Database Privacy". In: *arXiv:1109.2229 [cs]* (Sept. 2011). arXiv: 1109.2229 [cs].

[CMS] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. "Differentially Private Empirical Risk Minimization". en. In: (), p. 41.

[DR13] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". en. In: *Foundations and Trends® in Theoretical Computer Science* 9.3-4 (2013), pp. 211–407. ISSN: 1551-305X, 1551-3068. DOI: 10.1561/0400000042.

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. "Boosting and Differential Privacy". en. In: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE, Oct. 2010, pp. 51–60. ISBN: 978-1-4244-8525-3. DOI: 10.1109/FOCS.2010.12.

[Dwo+09] Cynthia Dwork et al. "On the Complexity of Differentially Private Data Release: Efficient Algorithms and Hardness Results". en. In: *Proceedings of the 41st Annual ACM Symposium on Symposium on Theory of Computing - STOC '09*. Bethesda, MD, USA: ACM Press, 2009, p. 381. ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536467.

[FS99] Yoav Freund and Robert E. Schapire. "Adaptive Game Playing Using Multiplicative Weights". In: *Games and Economic Behavior* 29.1-2 (1999), pp. 79–103.

[Gab+14] Marco Gaboardi et al. "Dual Query: Practical Private Query Release for High Dimensional Data". In: *arXiv:1402.1526 [cs]* (Feb. 2014). arXiv: 1402.1526 [cs].

[Goo+14] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: *arXiv:1406.2661 [cs, stat]* (June 2014). arXiv: 1406.2661 [cs, stat].

[Grn+17] Paulina Grnarova et al. "An Online Learning Approach to Generative Adversarial Networks". en. In: *arXiv:1706.03269 [cs, stat]* (June 2017). arXiv: 1706.03269 [cs, stat].

[GRU11] Anupam Gupta, Aaron Roth, and Jonathan Ullman. "Iterative Constructions and Private Data Release". In: *arXiv:1107.3731 [cs]* (July 2011). arXiv: 1107.3731 [cs].

[HLM12] Moritz Hardt, Katrina Ligett, and Frank Mcsherry. "A Simple and Practical Algorithm for Differentially Private Data Release". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 2339–2347.

[HR10] M. Hardt and G. N. Rothblum. "A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis". In: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. Oct. 2010, pp. 61–70. DOI: 10.1109/FOCS.2010.85.

[HRU13] Justin Hsu, Aaron Roth, and Jonathan Ullman. "Differential Privacy for the Analyst via Private Equilibrium Computation". In: *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13* (2013), p. 341. DOI: 10.1145/2488608.2488651. arXiv: 1211.0877.

[JY19] James Jordon and Jinsung Yoon. "PATE-GAN: GENERATING SYNTHETIC DATA WITH DIFFERENTIAL PRIVACY GUARANTEES". en. In: (2019), p. 21.

[KST] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. "Private Convex Empirical Risk Minimization and High-Dimensional Regression". en. In: (), p. 40.

[KV05] Adam Kalai and Santosh Vempala. "Efficient Algorithms for Online Decision Problems". en. In: *Journal of Computer and System Sciences* 71.3 (Oct. 2005), pp. 291–307. ISSN: 00220000. DOI: 10.1016/j.jcss.2004.10.016.

[Li+14] Chao Li et al. "A Data- and Workload-Aware Algorithm for Range Queries under Differential Privacy". en. In: *Proceedings of the VLDB Endowment* 7.5 (Jan. 2014), pp. 341–352. ISSN: 21508097. DOI: 10.14778/2732269.2732271.

[Loh10] Steve Lohr. "Netflix Cancels Contest After Concerns Are Raised About Privacy". en-US. In: *The New York Times* (Mar. 2010). ISSN: 0362-4331.

[Mar19] Gonzalo Martínez-Muñoz. "Sequential Training of Neural Networks with Gradient Boosting". en. In: *arXiv:1909.12098 [cs, stat]* (Sept. 2019). arXiv: 1909.12098 [cs, stat].

[MS18] Eran Malach and Shai Shalev-Shwartz. "A Provably Correct Algorithm for Deep Learning That Actually Works". en. In: *arXiv:1803.09522 [cs, stat]* (June 2018). arXiv: 1803.09522 [cs, stat].

[Nas50] John F. Nash. "Equilibrium Points in N-Person Games". en. In: *Proceedings of the National Academy of Sciences of the United States of America* 36.1 (1950), pp. 48–49.

[NRW18] Seth Neel, Aaron Roth, and Zhiwei Steven Wu. "How to Use Heuristics for Differential Privacy". en. In: *arXiv:1811.07765 [cs, stat]* (Nov. 2018). arXiv: 1811.07765 [cs, stat].

[NS07] Arvind Narayanan and Vitaly Shmatikov. "How To Break Anonymity of the Netflix Prize Dataset". en. In: *arXiv:cs/0610105* (Nov. 2007). arXiv: cs/0610105.

[PFL18] Henning Petzka, Asja Fischer, and Denis Lukovnicov. "On the Regularization of Wasserstein GANs". en. In: *arXiv:1709.08894 [cs, stat]* (Mar. 2018). arXiv: 1709.08894 [cs, stat].

[Sal+16] Tim Salimans et al. "Improved Techniques for Training GANs". en. In: *arXiv:1606.03498 [cs]* (June 2016). arXiv: 1606.03498 [cs].

[SN19] Arun Sai Suggala and Praneeth Netrapalli. "Online Non-Convex Learning: Following the Perturbed Leader Is Optimal". en. In: *arXiv:1903.08110 [cs, math, stat]* (Sept. 2019). arXiv: 1903.08110 [cs, math, stat].

[Swe02] Latanya Sweeney. "K-ANONYMITY: A MODEL FOR PROTECTING PRIVACY". en. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (Oct. 2002), pp. 557–570. ISSN: 0218-4885, 1793-6411. DOI: 10.1142/S0218488502001648.

[Ull12] Jonathan Ullman. "Answering $N\hat{\{}2+o(1)\}$ Counting Queries with Differential Privacy Is Hard". In: *arXiv:1207.6945 [cs]* (July 2012). arXiv: 1207.6945 [cs].

[UV11] Jonathan Ullman and Salil Vadhan. "PCPs and the Hardness of Generating Private Synthetic Data". en. In: *Theory of Cryptography*. Ed. by David Hutchison et al. Vol. 6597. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 400–416. ISBN: 978-3-642-19570-9 978-3-642-19571-6. DOI: 10.1007/978-3-642-19571-6_24.

[Vie+] Giuseppe Vietri et al. "New Oracle-Efficient Algorithms for Private Synthetic Data Release". en. In: (), p. 15.

[Xie+18] Liyang Xie et al. "Differentially Private Generative Adversarial Network". en. In: *arXiv:1802.06739 [cs, stat]* (Feb. 2018). arXiv: 1802.06739 [cs, stat].

[Zhe+15] Hao Zheng et al. "Improving Deep Neural Networks Using Softplus Units". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–4. ISBN: 1-4799-1960-8.