THE SAMPLE COMPLEXITY OF PARITY LEARNING IN THE ROBUST SHUFFLE MODEL

A Thesis
submitted to the Faculty of the
Graduate School of Arts and Sciences
of Georgetown University
in partial fulfillment of the requirements for the
degree of
Master of Science
in Computer Science

By

Chao Yan, B.S.

Washington, DC
April 21, 2021
Abstract

Differential privacy [Dwork, McSherry, Nissim, Smith TCC 2006] is a standard of privacy in data analysis of personal information requiring that the information of any single individual should not influence the analysis’ output distribution significantly. The focus of this thesis is the shuffle model of differential privacy [Bittau, Erlingsson, Maniatis, Mironov, Raghunathan, Lie, Rudominer, Kode, Tinnés, Seefel SOSP 2017], [Cheu, Smith, Ullman, Zeber, Zhilyaev EUROCRYPT 2019]. In the shuffle model, users communicate with an analyzer via a trusted shuffler, which permutes all received messages uniformly at random before submitting them to the analyst (then the analyst outputs an aggregate answer). Another model which we will discuss is the pan-privacy model [Dwork, Naor, Pitassi, Rothblum, Yekhanin ICS 2010], where an online algorithm is required to maintain differential privacy of both its internal state and its output (jointly).

We focus on the task of parity learning in the robust shuffle model and obtain the following contributions:

- We construct a reduction from a pan-private parity learner to the robust shuffle parity learner. Given an $(\epsilon, \delta, 1/3)$-robust shuffle private parity learner, we construct an $(\epsilon, \delta)$-pan-private parity learner. Applying recent pan-privacy lower-bounds [Cheu, Ullman 2021], we obtain a lower bound on the sample complexity
$\Omega(2^{d/2})$ in the pan-private parity learner, which in turn implies the same lower bound in the robust shuffle model.

• We present a simple robust shuffle parity learning algorithm with sample complexity $O(d^{2^{d/2}})$. The algorithm evaluates, with differential privacy, the empirical error of all possible parity functions, and selects the one with minimal error.

**INDEX WORDS:** Differential privacy, private learning, parity learning
This thesis could not be completed without the help of many people. I want to show my appreciation here.

I would like to thank my advisor Kobbi Nissim, who gave me a lot of support from the beginning. He taught me a lot during my master, including the knowledge of differential privacy, method of research and thesis writing. I am grateful for his help, guidance, and patience.

I want to thank faculties who have taught me in Georgetown. They are Sasha Golovnev, Bala Kalyanasundaram, Calvin Newport, Kobbi Nissim, Micah Sherr, Richard Squier, Justin Thaler. Their courses helped me to have a basic background of computer science.

I also want to thank Sasha Golovnev and Jonathan Ullman for willing to be my committee members.

At last, I want to thank my parents. Without their support, I would not have chance to study in Georgetown.
# Table of Contents

## Chapter

1. **Introduction** ........................................... 1  
   1.1 Results of this thesis ................................. 3

2. **Preliminaries** ........................................... 4  
   2.1 Differential privacy .................................. 4  
   2.2 Models of computation for differential privacy ................. 6  
   2.3 Private learning ...................................... 12

3. **Related Work** ........................................... 15  
   3.1 Non-private and private learning ......................... 15  
   3.2 Reduction from pan-private algorithms to shuffle private algorithms 15  
   3.3 Hard tasks for pan-private mechanisms ................... 17  
   3.4 Private summation protocol in shuffle model ................ 19

4. **A Lowerbound on the Sample Complexity of Parity Learning in the Shuffle Model** ........................................... 20  
   4.1 From robust shuffle model parity learner to a pan-private parity learner ........................................... 20  
   4.2 From pan-private parity learner to distinguishing hard distributions 22  
   4.3 Tightness of the lowerbound ............................ 25

5. **Conclusion** ............................................. 29

Appendix: Tail Bounds ....................................... 30

BIBLIOGRAPHY ............................................. 31
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The curator model of differential privacy</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>The local model</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>The shuffle model</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>The pan-privacy model</td>
<td>11</td>
</tr>
</tbody>
</table>
Differential privacy [9] is a mathematical definition of privacy applied to analyses of personal data. It requires that any observer of the outcome of the data analysis would not be able to determine whether specific individual is included in the data set or not.

Differential privacy requires that the output distributions on two neighbouring input data are similar, where neighbouring data are two data that only differ on one element, i.e. for \( x = (x_1, \ldots, x_n) \) and \( x' = (x'_1, \ldots, x'_n) \), there only exists one \( i \in \{1, \ldots, n\} \) such that \( x_i \neq x'_i \). More formally, with parameters \( \varepsilon \) and \( \delta \), an algorithm \( M \) satisfies \((\varepsilon, \delta)\)-differential privacy if for any event \( T \),

\[
\Pr[M(x) \in T] \leq e^\varepsilon \cdot \Pr[M(x') \in T] + \delta.
\]

There are some different settings for differential privacy. The most widely researched setting is the curator model, also known as the central model, where the data is collected by a trusted curator and then the curator outputs an aggregate answer, computed with differential privacy. To provide differential privacy, the answer must be randomized. For example, suppose \( x \in \{0, 1\}^n \) and the goal is to approximate \( f(x) = \sum_{i=1}^{n} x_i \). Dwork et al. [9] presented a mechanism satisfying \((\varepsilon, 0)\)-differential privacy:

\[
M(x) = \sum_{i=1}^{n} x_i + \text{Lap}(1/\varepsilon).
\]
A disadvantage of the curator model is that it relies on a trusted curator. The local model [17] avoids this drawback by eliminating the need for a trusted curator. In the local model, instead of directly outputting raw data, each user randomizes their data by applying a differentially private mechanism, called \textit{local randomizer}, before sharing it with an analyst. The data analyst post-processes the messages it received to publish a differentially private answer. As the adversary can view all users’ outputs, the local model must satisfy for all neighboring \((x_1, \ldots, x_n), (x'_1, \ldots, x'_n)\) and all events \(T\):

\[
\Pr[(R_1(x_1), \ldots, R_n(x_n)) \in T] \leq e^\epsilon \cdot \Pr[(R_1(x'_1), \ldots, R_n(x'_n)) \in T] + \delta,
\]

or, equivalently,

\[
\Pr[R_i(x) \in T] \leq e^\epsilon \cdot \Pr[R_i(x') \in T] + \delta,
\]

for all \(i \in [n], x, x' \) and \(T\).

However, processing personal information in the local model restricts utility (compared with the curator model). For example, any differentially private mechanism that computing binary sum in the local model incurs at least \(\Omega(\sqrt{n})\) noise in [5], while it in the curator model, the mechanism presented above yields \(O(1)\) noise [9].

Considering the advantages and issues of the curator and local model, it is natural to seek alternative models. The shuffle model is one alternative approach, which is composed of an untrusted curator, local randomizers and a trusted shuffler [3, 6]. In the shuffle model, each individual sends (potentially noisy) data to the shuffler, which randomly permutes the messages and then submits them to an analyzer. The random permutation potentially disassociate messages from their senders, and hence potentially amplifies privacy.

The pan-private model [10] is another alternative model, where the curator is trusted, but the adversary can intrude the internal state of the algorithm. A pan-private algorithm receives a stream of raw data and starts with a beginning state.
Each time unit the algorithm processes one element and updates its internal state by combining the element and the current state. At the end, the algorithm process its final state and outputs the aggregate answer. The pan-private algorithm requires that both of internal state and the output are differentially private. Balcer et al. [1] discovered a connection between the pan-private model and the shuffle model. Concretely, if a shuffle private algorithm is robust, which means it can maintain differential privacy provided that a certain fraction of the agents participate, there exists a way to construct a corresponding pan-private algorithm.

This thesis proves a lower bound on the sample complexity of parity learning algorithm in the robust shuffle model. The thesis also provides a robust shuffled parity learning algorithm, which shows that the bound is almost tight.

1.1 Results of this thesis

We first construct a reduction from the pan-private parity learner to the robust shuffle parity learner. This is based on the works of Balcer et al. [1] and Cheu and Ullman [7]. We then prove the lower bound on the sample complexity of parity learning algorithm in robust shuffle model. This applies a result of Cheu and Ullman [7]. Given a family of distributions \( \{p_v\}_{v \in V} \), if a pan-private algorithm can distinguish two mixture distributions \( P^n_v \) and \( U^n \) (see details in Section 3.3), then this algorithm has a sample complexity \( n = \Omega \left( \frac{1}{\varepsilon \| \{p_v\} \|_{\infty \rightarrow 2}} \right) \). They also gave a family of hard distributions \( P_{d,k,\alpha} \) and compute \( \| P_{d,k,\alpha} \|_{\infty \rightarrow 2} \). We show how to use a pan-private learner to distinguish \( P^n_v \) and \( U^n \) when the family of distribution is \( P_{d,k,1/2} \). Then we apply the lower bound of Cheu and Ullman. We prove this lower bound is almost tight by constructing a concrete algorithm to learn the parity function. This uses the algorithm in the work of Balle et al. [2].
2.1 Differential privacy

Differential privacy is a standard that the output is insensitive to any single element of the input data. By introducing randomness, the behaviours of the differentially private analysis are similar on neighbouring data that differ on the entry of any single individual. In mathematical language, \( x, x' \in \mathcal{X} \) is a pair of neighbouring data if \( |\{i : x_i \neq x'_i\}| = 1 \).

Definition 1 (Differential Privacy [9]) Let \( \mathcal{X} \) be a data universe and let \( \varepsilon, \delta \geq 0 \) \( A \) (randomized) mechanism \( M : \mathcal{X}^n \rightarrow \mathcal{Y} \) is \((\varepsilon, \delta)\)-differentially private if for every event \( T \subseteq \mathcal{Y} \) and for all neighboring datasets \( x, x' \in \mathcal{X}^n \),

\[
\Pr[M(x) \in T] \leq e^{\varepsilon} \Pr[M(x') \in T] + \delta,
\]

where the probability is over the randomness of \( M \).

The randomization in differential privacy is necessary, because if an algorithm \( M \) is deterministic and \( M(x) = y_1, M(x') = y_2 \), where \( y_1 \neq y_2 \). When the output is \( y_1 \), we know the input can’t be \( x' \).

One example of a differentially private mechanism is randomized response introduced by Warner [21]. The randomized function \( R(x) \) with input \( x \in \{0, 1\} \) is set as:

\[
R(x) = \begin{cases} 
  x & \text{with probability } \frac{e^\varepsilon}{(e^\varepsilon + 1)} \\
  1 - x & \text{with probability } \frac{1}{(e^\varepsilon + 1)}
\end{cases}
\]
It can be verified that

\[
\Pr[R(1) = 1] \leq e^\varepsilon \cdot \Pr[R(0) = 1], \quad \Pr[R(0) = 0] \leq e^\varepsilon \cdot \Pr[R(1) = 0],
\]

\[
\Pr[R(0) = 1] \leq e^\varepsilon \cdot \Pr[R(1) = 1], \quad \Pr[R(1) = 0] \leq e^\varepsilon \cdot \Pr[R(0) = 0].
\]

Thus \(R(x)\) is \((\varepsilon, 0)\)-differentially private.

Applying any algorithm \(f\) to an \((\varepsilon, \delta)\)-differentially private algorithm \(M\) also satisfies \((\varepsilon, \delta)\)-differential privacy.

**Proposition 1 (Post-Processing)** Let \(M : X^n \rightarrow Y\) be an \((\varepsilon, \delta)\)-differentially private algorithm and \(f : Y \rightarrow Y'\) be any algorithm. Then \(f \circ M : X^n \rightarrow Y'\) is \((\varepsilon, \delta)\)-differentially private.

We can combine two differentially private mechanisms to create a new differentially private mechanism. As each of the two analyses can contribute to the leakage of information, the parameters of the combined analysis cannot be smaller than that of the individual analyses. A collection of composition theorems control how the parameters \(\varepsilon, \delta\) grow. E.g., basic composition, as below, tells that the growth is at most linear.

**Theorem 1 (Basic Composition [8, 11])** Let mechanism \(M_1\) be \((\varepsilon_1, \delta_1)\)-differentially private and let mechanism \(M_2\) be \((\varepsilon_2, \delta_2)\)-differentially private. Define \(M_{1,2}(x) = (M_1(x), M_2(x))\) as the combination of \(M_1\) and \(M_2\), then \(M_{1,2}(x)\) satisfies \((\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)\)-differential privacy.

Better composition theorems exist, where the growth of the privacy parameters is sub-linear. One such theorem is provided by Dwork, Rothblum, and Vadhan:
Theorem 2 (Advanced Composition [12]) Given $M_1, M_2, \ldots, M_k$, which are $(\varepsilon, \delta)$-differentially private, then $M = (M_1, M_2, \ldots, M_k)$ is $(\varepsilon', k\delta + \delta')$-differentially private, where

$$
\varepsilon' = \sqrt{2k \ln 1/\delta'} \cdot \varepsilon + k \cdot \varepsilon (e^\varepsilon - 1).
$$

Specifically, when $\varepsilon' < 1$, to ensure that the composed mechanism is $(\varepsilon', k\delta + \delta')$-differentially private, it suffices that each mechanism is $(\varepsilon, \delta)$-differentially private, where

$$
\varepsilon = \frac{\varepsilon'}{2\sqrt{2k \ln 1/\delta'}}.
$$

2.2 Models of computation for differential privacy

2.2.1 The curator model

In the curator model of differential privacy, it is assumed that the data analyst is trustworthy. The curator collects individual information in the clear, and produces the outcome of a differentially private computation. To make the computation private, the curator introduces random noise in its computation. One widely used method is the Laplace mechanism [9], which promises differential privacy by adding noise sampled from the Laplace distribution.

![Figure 2.1: The curator model of differential privacy](image)

Figure 2.1: The curator model of differential privacy
Definition 2 (The Laplace Distribution) The probability density function of zero-centered Laplace distribution $\text{Lap}(b)$ is
\[
h(x) = \frac{1}{2b} \exp \left( -\frac{|x|}{b} \right).
\]
The variance of $\text{Lap}(b)$ is $2b^2$.

Definition 3 (Global $\ell_1$ sensitivity) Given a real function $f : \mathcal{X}^n \to \mathbb{R}^k$, the global sensitivity of $f$ is
\[
\Delta f = \max \{ f(x) - f(x') \},
\]
where the maximum is taken over neighboring $x, x' \in \mathcal{X}^n$.

Theorem 3 (The Laplace Mechanism [9]) Given a real function $f : \mathcal{X}^n \to \mathbb{R}^k$, the mechanism
\[
M_L(x) = f(x) + (\text{Lap}_1(\Delta f/\varepsilon), \ldots, \text{Lap}_k(\Delta f/\varepsilon))
\]
is $\varepsilon$-differentially private.

For example, suppose $x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$ and $f(x) = \sum_{i=1}^n x_i$. We have that $\Delta f = 1$ and hence $M_L = \sum_{i=1}^n x_i + \text{Lap}(1/\varepsilon)$ is $\varepsilon$-differentially private.

The Laplace mechanism applies to computations that map to real numbers. The discrete version of it is the discrete Laplace mechanism, which maps to the integers.

Definition 4 (Discrete Laplace Distribution [14]) For any integer $k$, the probability mass function of discrete Laplace distribution $\text{DLap}(\alpha)$ is
\[
p(k) = \frac{\alpha - 1}{\alpha + 1} \cdot \alpha^{-|k|}.
\]
2.2.2 The local model

In the local model, each user outputs noisy data through a local randomizer, then an analyzer post-processes the received noisy data and publishes the output. The local model requires that each local randomizer is differentially private. Due to the post-processing property of differential privacy, the adversary’s view is differentially private.

Definition 5 (Local Model, Non-Interactive [13, 18, 21]) A local model protocol \( M = ((R_1, R_2, \ldots, R_n), A) \) is \((\varepsilon, \delta)\)-differentially private if \((R_1(x_1), \ldots, R_n(x_n))\) is \((\varepsilon, \delta)\)-differentially private, or, equivalently, \(R_i(x_i)\) is \((\varepsilon, \delta)\)-differentially private for all \(i \in [n]\).

![Figure 2.2: The local model](image)

This definition can be generalized to the interactive local model. In the context of private learning (discussed below) Kasiviswanathan et al. [18] proved an equivalence between the local model and statistical query (SQ) learning.

As an example for a computation in the local model of differential privacy, to compute bit addition, the local randomizer can be set to the randomized response.
functionality [21]:

\[
R(x) = \begin{cases} 
  x & \text{with probability } \frac{e^\varepsilon}{(e^\varepsilon + 1)} \\
  1-x & \text{with probability } \frac{1}{(e^\varepsilon + 1)} 
\end{cases}
\]

The aggregator can be set as:

\[
A(r_1, \ldots, r_n) = \frac{e^\varepsilon + 1}{e^\varepsilon - 1} \left( \sum_{i=1}^{n} r_i - \frac{n}{e^\varepsilon + 1} \right).
\]

It is easy to verify that \( E(A(R(x_1), \ldots, R(x_n))) = E(\sum_{i=1}^{n} x_i) \). By the Chernoff bound, the error is less than \( O(\sqrt{n}/\varepsilon) \) with constant probability.

### 2.2.3 The shuffle model

In the shuffle model, there exists a trusted shuffler that applies a uniformly random permutation to the input messages. In this thesis, we focus on the variant presented in [3, 6] – the Encode, Shuffle, Analyze (ESA) architecture. In the ESA framework, users Encode their data in one or more messages. A shuffler collects and randomly permutes these messages. Finally, an analyst processes the permuted messages to generate an output. A similar model was researched by Ishai el al. [16] in the context of secure multiparty computation.

**Definition 6 (Shuffle Model [3, 6])** A shuffle model protocol \( M = (R, S, A) \) consists of three kinds of randomized algorithms:

1. A set of randomizers: \( R = (R_1, \ldots, R_n) \), where for each \( i \in [n] \), \( R_i : X \to Y^* \) maps the input to a number of massages.

2. A shuffler \( S : Y^* \to Y^* \) permutes the input messages uniformly at random.

3. An analyst \( A : Y^* \to Z \) computes the result of the algorithm.
**Definition 7** A shuffle model protocol \( M = (R, S, A) \) is \((\varepsilon, \delta)\)-differentially private if for every set \( T \) and neighbouring data \( x = (x_1, \ldots, x_n) \) and \( x' = (x'_1, \ldots, x'_n) \),

\[
\Pr[S(R_1(x_1), \ldots, R_n(x_n)) \in T] \leq e^\varepsilon \Pr[S(R_1(x'_1), \ldots, R_n(x'_n)) \in T] + \delta.
\]

![Figure 2.3: The shuffle model](image)

A shuffle model protocol that preserves differential privacy with only a fraction of honest users is called robust shuffle model. Formally, the robust shuffle model is defined as:

**Definition 8 (Robust Shuffle Model [1])** For \( \gamma \in (0, 1] \), a shuffle private algorithm \( M = (R, S, A) \) is \((\varepsilon, \delta, \gamma)\)-robust shuffle private if for any \( \gamma' \geq \gamma \), any random permutation function \( \pi : [n] \to [n] \) and \( x = (x_1, \ldots, x_{\gamma'n}) \),

\[
S(R_{\pi(1)}(x_1), \ldots, R_{\pi(\gamma'n)}(x_{\gamma'n}))
\]

is \((\varepsilon, \delta)\)-differentially private.

### 2.2.4 Pan-Privacy

Pan-privacy is a requirement on online algorithm[2]. The Pan-private model, introduced by Dwork et al. [10], requires that differential privacy would be preserved in face of an intruder that can observe the outcome of the online algorithm as well as examine its internal state.
**Definition 9 (Online Algorithm)** An online algorithm $M = (R, A) : X^n \rightarrow Y$ consists of internal algorithm $R : I \rightarrow I$ (where $I$ is the algorithm’s internal state space) and an analysis algorithm $A : I \rightarrow Y$. The execution of $M$ on input $x \in X^n$ begins with the initial state $S_0$. Then, the internal algorithm $R$ maps the current state $S_{i-1} \in I$ and the input data point $x_i$ to the new state $S_i = R(S_{i-1}, x_i)$. At the end of the data stream, the analysis algorithm maps the final state to the output of the algorithm, $A(S_n)$.

**Definition 10 (Pan-privacy [10])** For an online algorithm $M = (R, A) : X^n \rightarrow Y$, let $S_{\leq t}(x) = (S_0, S_1, \ldots, S_t) \in I^t$ represent the first $t$ internal states when $M$ is executed on input $x$. $M$ is $(\varepsilon, \delta)$-pan-private if for every event $T \in I \times Y$,

$$
\Pr[(S_{\leq t}(x), M(x)) \in T] \leq e^\varepsilon \Pr[(S_{\leq t}(x), M(x')) \in T] + \delta.
$$

![Figure 2.4: The pan-privacy model](image-url)
2.3 Private learning

A concept is a Boolean mapping \( c : \mathcal{X} \rightarrow \{0, 1\} \). A concept class \( C = \{c\} \) is a collection of concepts. Let \( P \) be a distribution over \( \mathcal{X} \), and \( h : \mathcal{X} \rightarrow \{0, 1\} \) be a hypothesis. The generalization error between hypothesis \( h \) and concept \( c \) is defined as:

\[
err_P(c, h) = \Pr_{x \sim P}[h(x) \neq c(x)].
\]

**Definition 11 (PAC Learning [20])** A concept class \( C \) is \((\alpha, \beta, m)\)-PAC learnable if there exist an algorithm \( L \), such that for any distribution \( P \) over \( \mathcal{X} \) and all concepts \( c \in C \),

\[
\Pr \left[ \{x_i\}_{i=1}^m \sim P; h \leftarrow L \left( \{(x_i, c(x_i))\}_{i=1}^m \right); error_P(c, h) \leq \alpha \right] \geq 1 - \beta,
\]

where the probability is over the choice of \( x_1, \ldots, x_m \) i.i.d. from \( P \) and the randomness of \( L \).

Let \( d \) represent the size of elements in \( \mathcal{X} \), we say that the concept class \( C \) is efficiently PAC learnable if it is \((\alpha, \beta, m)\)-PAC learnable with sample complexity \( m = \text{poly}(d, 1/\alpha, \log(1/\beta)) \).

Agnostic PAC learning [15] extends the definition of generalization error in PAC learning. Let \( P \) be a distribution over \( \mathcal{X} \times \{0, 1\} \). Define

\[
err_P(h) = \Pr_{(x,y) \sim P}[h(x) \neq y],
\]

Let \( err_{OPT} \) denote the minimized possible error of the hypothesis taken from the hypothesis class \( \mathcal{H} \), i.e.

\[
err_{OPT} = \min_{h \in \mathcal{H}} err_P(h).
\]

**Definition 12 (Agnostic PAC Learning with hypothesis class \( \mathcal{H} \) [15])** A hypothesis class \( \mathcal{H} \) is \((\alpha, \beta, n)\)-agnostic PAC learnable if there exist an algorithm \( L \), such
that for any distribution $P$ over $(\mathcal{X} \times \{0, 1\})$, 

$$\Pr \left[ \{(x_i, y_i)\}_{i=1}^m \sim P; h \leftarrow L\left(\{(x_i, y_i)\}_{i=1}^m\right); \text{err}_P(h) \leq \text{err}_{OPT} + \alpha \right] \geq 1 - \beta,$$

where the probability is over the choice of $(x_1, y_1), \ldots, (x_m, y_m)$ i.i.d. from $P$ and the randomness of $L$.

Kasiviswanathan et al. [18] defined Private PAC learning algorithms to be PAC learning algorithms which satisfy differential privacy.

**Definition 13 (Privately PAC Learning [18])** A concept class $\mathcal{C}$ is private PAC learnable by algorithm $L$ with parameters $\alpha, \beta, m, \varepsilon, \delta$, if $L$ is $(\varepsilon, \delta)$-differential private and $L$ is $(\alpha, \beta, m)$-PAC learns concept class $\mathcal{C}$.

**Definition 14 (Privately Agnostic PAC Learning [18])** A hypothesis class $\mathcal{H}$ is private PAC learnable by algorithm $L$ with parameters $\alpha, \beta, m, \varepsilon, \delta$, if $L$ is $(\varepsilon, \delta)$-differential private and $L$ is $(\alpha, \beta, m)$-agnostic PAC learns concept class $\mathcal{C}$.

The statistical query (SQ) model [19] restricts the access of the learning algorithm to only observe statistical properties of the distribution. In SQ learning, the learner accesses its data through an SQ oracle, which answers statistical queries with bounded error.

**Definition 15 (SQ Oracle [19])** Let $P$ be a distribution over the domain $\mathcal{X}$. A statistical query consists of a function $f : \mathcal{D} \rightarrow [0, 1]$ and a tolerance parameter $\tau = 1/\text{poly}(d)$. An SQ oracle $O_{P}^{\text{SQ}}$ responds to this statistical query by outputting a value $v \in [0, 1]$ such that:

$$|v - \mathbb{E}_{x \sim P}[f(x)]| \leq \tau.$$
**Definition 16 (SQ Learning [19])** An SQ learning algorithm with parameters \( \alpha, \beta, \tau \) is a PAC learning algorithm which accesses its data through \( \mathcal{O}_P^{SQ} \) and outputs a hypothesis \( h \), such that:

\[
\Pr_{x \sim \mathcal{D}}[\text{err}_P(h) \geq \alpha] \leq \beta.
\]

In this thesis we focus on learning parity function.

**Definition 17 (weight \( k \) parity)** Let \( \text{PARITY}_{d,k} = \{c_{\ell,b}\}_{\ell \subseteq [d], |\ell| \leq k, b \in \{\pm 1\}} \) where \( c_{\ell,b} : \{\pm 1\}^d \to \{\pm 1\} \) is defined as \( c_{\ell,b}(x) = b \cdot \prod_{i \in \ell} x_i \). Where \( k = d \) we omit \( k \) and write \( \text{PARITY}_d \).

**Definition 18** A distribution-free parity learner is a PAC learning algorithm for \( \text{PARITY}_{d,k} \). A uniform distribution parity learner is a PAC learning algorithm for \( \text{PARITY}_{d,k} \) where the underlying distribution \( P \) is known to be uniform over \( \mathcal{X} = \{\pm 1\}^d \).
Chapter 3

Related Work

3.1 Non-private and private learning

Kasiviswanathan et al. [18] proved an equivalence between the statistical queries model and the local model, i.e. an algorithm in one model can be simulated by an algorithm in the another model. For the parity learning problem, Kasiviswanathan et al. [18] presented an efficient private PAC learning algorithm, but Blum et al. [4] proved that parity cannot be efficiently learned in the SQ model. That shows a separation between the local model and private PAC learning model. In this thesis, we prove that learning parity functions in the robust shuffle model requires $\Omega\left(\frac{2^{d/2}}{\varepsilon}\right)$ samples.

3.2 Reduction from pan-private algorithms to shuffle private algorithms

Balcer et al. [1] established a relationship between the robust shuffle model and the pan-private model. Then they used lowerbounds on the pan-privacy settings to establish lowerbounds for the robust shuffle model. As one example from their work, given an $(\varepsilon, \delta, 1/3)$-robust shuffle uniformity testing algorithm (informally, a distinguisher whether samples are from the uniform distribution or from a distribution which is “far” from being uniform), they constructed a pan-private algorithm that for the uniformity testing problem. To keep pan-privacy, the initial state is generated by applying
the 1/3-robust algorithm to $N/3$ data points sampled from the uniform distribution. When a data point arrives, the internal state is updated as follows: (i) The new data point is mapped into several massages, which are concatenated with the current state by the shuffler. (ii) Then, the shuffler randomly permutes all massages and sets them as the new state. Before submitting data to the analyst, the final state adds extra $N/3$ data points sampled from uniform distribution. The analyst then computes and publishes the final output of the algorithm. This idea was generalized by Cheu and Ullman [7], as presented in Algorithm 1.

**Algorithm 1:** $M^\Pi$: a pan-private algorithm constructed from the robust shuffle private algorithm $\Pi$ [7]

Let $\Pi = (R, S, A)$ be a 1/3-robust differentially private shuffle model protocol.

**Input:** Data stream $x \in \{X\}^{n/3}$.

1. Create initial state $s_0 \leftarrow S(R^{n/3}(U^{n/3}))$.
2. Sample $N' \sim \text{Bin}(n, 2/9)$.
3. Set $N' \leftarrow \min(N', n/3)$.
4. for $i \in [n/3]$ do
   5. if $i \in [N']$ then
      6. $w_i \leftarrow x_i$
   7. else
      8. $w_i \sim U$
   9. end
10. $s_i \leftarrow S(s_{i-1}, R(w_i))$
end
12. $s_{\text{final}} \leftarrow S(s_{n/3}, R^{n/3}(U^{n/3}))$
13. return $A(s_{\text{final}})$

**Theorem 4 ( [1, 7])** Given an $(\varepsilon, \delta, 1/3)$-robust shuffle private algorithm $\Pi$, Algorithm $M^\Pi$ is $(\varepsilon, \delta)$-pan-private.

*Proof. (sketch, following [1, 7])* Let $x$ and $x'$ be two neighboring data sets and let $j$ be the index where $x$ and $x'$ differ. Let $1 \leq t \leq n/3$ be the time an adversary probes into the algorithm's memory.
If \( t \geq j \) then \( S_{\leq t} = (S \circ (R_1, \ldots, R_{n/3+t}))(U^{n/3}, w_1, \ldots, w_t) \) and, as \( M \) is a robust differentially private mechanism \( S_{\leq t} \) preserves \((\varepsilon, \delta)\)-differential privacy. Because \( A(s_{final}) \) is post-processing of \( S_{\leq t} \) the outcome of \( M^\Pi \) is \((\varepsilon, \delta)\)-pan private.

If \( t < j \) then \( S_{\leq t}(x) \) is identically distributed to \( S_{\leq t}(x') \). Note that as \( M \) is a robust differentially private mechanism we get that

\[
\sigma = (S \circ (R_t, \ldots, R_{2n/3-t}))(S_{\leq t}, w_{t+1}, \ldots, w_{N'}, U^{n/3})
\]

preserves \((\varepsilon, \delta)\)-differential privacy. To conclude the proof, note that \((S_{\leq t}(x), A(s_{final}))\) is the result of post-processing \( \sigma \).

### 3.3 Hard Tasks for Pan-private Mechanisms

Cheu and Ullman [7] proved a lower bound on the sample complexity of pan-private algorithms that can distinguish between a family of distributions and their uniform mixture. Concretely, let \( \{P_v\}_{v \in V} \) be a family of distributions and define \( U = \mathbb{E}_{v \in V}(P_v) \). Let \( U^n \) the Cartesian product of \( n \) copies of \( U \) and \( P^n_v = \mathbb{E}_{v \in V}(P^n_v) \). The \((\infty \to 2)\)-norm of \( \{P_v\}_{v \in V} \) is defined as

\[
\|\{P_v\}\|_{\infty \to 2}^2 = \sup_{f: \mathbb{X} \to \{\pm 1\}} \mathbb{E}_{v \sim V} [(\mathbb{E}_{x \sim P_v}(f(x)) - \mathbb{E}_{x \sim U}(f(x)))^2].
\]

**Theorem 5 ([7])** For a family of distributions \( \{P_v\}_{v \in V} \), and an \((\varepsilon, \delta)\)-pan-private algorithm \( M \), such that \( \delta \log (|V|/\delta) = o(\varepsilon^2\|\{P_v\}\|_{\infty \to 2}^2) \), and \( d_{TV}(M(P^n_v), M(U^n)) = \Omega(1) \), then

\[
n = \Omega\left(\frac{1}{\varepsilon\|\{P_v\}\|_{\infty \to 2}^2}\right).
\]

They provided a family of hard distributions. Let \( x \in \{\pm 1\}^d \), a parameter \( \alpha \in (0, 1/2] \) (Cheu and Ullman restricted \( \alpha \) to the open interval \((0, 1/2)\) but their proof also holds for \( \alpha = 1/2 \)), a non-empty set \( \ell \subseteq [d] \) and a bit \( b \in \{\pm 1\} \). The distribution
$P_{d,\ell,b,\alpha}$ is defined as:

$$P_{d,\ell,b,\alpha}(x) = \begin{cases} (1 + 2\alpha)2^{-d} & \text{if } \prod_{i \in \ell} x_i = b \\ (1 - 2\alpha)2^{-d} & \text{if } \prod_{i \in \ell} x_i = -b \end{cases}$$

Define the family of distributions

$$\mathcal{P}_{d,k,\alpha} = \{P_{d,\ell,b,\alpha}(x) : \ell \subseteq [d], |\ell| \leq k, b \in \{\pm 1\}\}.$$ 

In particular, for $\alpha = 1/2$ we get

$$P_{d,\ell,b,1/2}(x) = \begin{cases} 2^{-d+1} & \text{if } \prod_{i \in \ell} x_i = b \\ 0 & \text{if } \prod_{i \in \ell} x_i = -b \end{cases}$$

$$\mathcal{P}_{d,k,1/2} = \{P_{d,\ell,b,1/2}(x) : \ell \subseteq [d], |\ell| \leq k, b \in \{\pm 1\}\}.$$ 

Cheu and Ullman proved a lower bound on the $\infty \rightarrow 2$ norm of $\mathcal{P}_{d,k,\alpha}$:

**Lemma 1 ([7])** For $d \in \mathbb{N}$, $\alpha \in (0, 1/2]$,

$$\|\mathcal{P}_{d,k,\alpha}\|_{\infty \rightarrow 2}^2 \leq \frac{4\alpha^2}{d_{\leq k}}.$$ 

Substituting this bound in theorem 5 results in the following theorem:

**Theorem 6 ([7], rephrased)** Let $P_{d,L,B,\alpha}$ be a uniformly random distribution chosen from $\mathcal{P}_{d,k,\alpha}$, where $L$ is a random set such that $L \subseteq [d]$ and $|L| \leq k$ and $B$ is uniformly chosen from $\{\pm 1\}$. For an $(\varepsilon, \delta)$-pan-private algorithm $M$, such that

$$\delta \log \left( \frac{d}{\delta} \right) = o \left( \varepsilon^2 \alpha^2 / \left( \frac{d}{\delta} \right) \right),$$

and $d_{TV}(M(P^n_{d,L,B,\alpha}), M(U^n)) = \Omega(1)$, then

$$n = \Omega \left( \frac{\sqrt{d_{\leq k}}}{\varepsilon \alpha} \right).$$

We apply this lower bound to prove a lower bound on the sample complexity of parity learning algorithms. Then this lower bound induces the lower bound for robust shuffle parity learners.
3.4 Private summation protocol in shuffle model

Balle et al. [2] constructed a real summation protocol in the shuffle model. In their protocol, for each user’s input $x$, the local randomizer computes noisy number $y = x + \text{polya}(1/n, \alpha) - \text{polya}(1/n, \alpha)$ and encodes $y$ as $k$ uniformly random numbers $(y_1, \ldots, y_k)$ satisfying $\sum_{i=1}^{k} y_i = y$. The curator computes the sum of all received messages and the total additive noise $\sum_{i=1}^{n} (\text{polya}(1/n, \alpha) - \text{polya}(1/n, \alpha))$ follows the discrete Laplace distribution, the protocol is hence differentially private. In our robust shuffle parity learner, we use the summation protocol by Balle et al. to estimate and compare the correctness of all possible parity functions for given samples.
A Lowerbound on the Sample Complexity of Parity Learning in the Shuffle Model

4.1 From robust shuffle model parity learner to a pan-private parity learner

We show how to construct, given a robust shuffle model distribution-free parity learner, a uniform distribution pan-private parity learner. Our reduction–Algorithm LearnParUnif–is described in Algorithm 2. We use a similar technique to the padding presented in [1, 7], with small modifications. To allow the shuffle model protocol use different randomizers $R_1, \ldots, R_n$, the pan-private learner applies these randomizers in a random order (the random permutation $\pi$). The padding is done with samples of the form $(1^d, \hat{b})$ where $\hat{b}$ is a uniformly random selected bit. Finally, as in [7] the number of labeled samples which the pan-private algorithm considers from its input is binomially distributed, so that if $(x_i, y_i)$ are such that $x_i$ is uniform in $X$ and $y_i = c_{r,b}(x_i) = b \cdot \prod_{i \in r} x_i$ then (after a random shuffle) the input distribution presented to the shuffle model protocol is statistically close to a mixture of the two following distributions: (i) a distribution where $\Pr[(x_i, y_i) = (1^d, \hat{b})] = 1$ and (ii) a distribution where $x_i$ is uniformly selected in $\{\pm 1\}^d$ and $y_i = c_{r,b}(x_i)$.

**Proposition 2** Algorithm LearnParUnif is $(\varepsilon, \delta)$-pan private.

*Proof.* The proof is identical to the proof of theorem 4 [1, 7].
Algorithm 2: LearnParUnif, a uniform distribution pan private parity learner

Let $M = (\langle R_1, \ldots, R_n \rangle, S, A)$ be a $1/3$-robust differentially private distribution parity learner.

Input: $n/3$ labeled examples $(x_i, y_i)$ where $x_i \in X$ and $y_i \in \{\pm 1\}$.

1. Randomly choose a permutation $\pi : [n] \rightarrow [n]$.
2. Randomly choose $\hat{b} \in \mathbb{R} \{\pm 1\}$.
3. Create initial state $s_0 \leftarrow S(\mathbb{R}^{\pi(1)}(1^d, \hat{b}), \ldots, R_{\pi(n/3)}(1^d, \hat{b}))$.
4. Sample $N' \sim \text{Bin}(n, 2/9)$.
5. Set $N' \leftarrow \min(N', n/3)$.
6. for $i \in [n/3]$ do
   7. if $i \in [N']$ then
   8. $w_i \leftarrow (x_i, y_i)$
   9. else
   10. $w_i \leftarrow (1^d, \hat{b})$
   11. end
   12. $s_i \leftarrow S(s_{i-1}, R_{\pi(n/3+i)}(w_i))$
7. end
14. $s_{\text{final}} \leftarrow S(s_{n/3}, R_{\pi(2n/3+1)}(1^d, \hat{b}), \ldots, R_{\pi(n)}(1^d, \hat{b}))$
15. return $A(s_{\text{final}})$

Proposition 3 (learning) Let $M$ be a $(\alpha, \beta, m)$ distribution free parity learner, where $\alpha, \beta < 1/4$ and $m = n/9$. Algorithm LearnParUnif is a uniform distribution parity learner that with probability at least $1/4$ correctly identifies the concept $c_{r,b}$.

Proof. (sketch) Algorithm LearnParUnif correctly guesses the label $b$ for $1^d$ with probability $1/2$. Assuming $\hat{b} = b$ the application of $M$ uniquely identifies $r, b$ with probability at least $1/2$. Thus, LearnParUnif recovers $c_{r,b}$ with probability at least $1/4$. 

21
4.2 From pan-private parity learner to distinguishing hard distributions

In this section, we use Theorem 6 to obtain a lower bound on the sample complexity of parity learning in the shuffle model. In Algorithm 3, we provide a reduction from identifies the hard distribution $P_{d,\ell,b,1/2}$ presented in section 3.3 to pan-private parity learning.

**Algorithm 3:** IdentifyHard, a pan-private for identifying the distribution $P_{d,\ell,b,1/2}$

- Let $\Pi$ be a pan-private uniform distribution parity learner.
- **Input:** A sample of $n$ examples $z = (z_1, z_2, \ldots, z_n)$, where each example is of the form $z_j = (z_j[1], z_j[2], \ldots, z_j[d]) \in \{\pm 1\}^d$
- Randomly choose $i^* \in_R [d]$.
- /* Apply the uniform distribution parity learner $\Pi$: */
- for $j \in [n]$ do
  - $y_j \leftarrow z_j[i^*]$
  - $x_j = z_j$
  - $x_j[i^*] = \bot$ /* i.e., $x_j$ equals $z_j$ with entry $i^*$ erased */
  - Provide $(x_j, y_j)$ to $\Pi$.
- end
- $(r, b) \leftarrow \Pi((x_1, y_1), \ldots, (x_n, y_n))$
- $\ell \leftarrow r \cup \{i^*\}$
- return $(\ell, b)$

**Observation 1** The pan-privacy of Algorithm 3 follows from the pan-privacy of algorithm $\Pi$.

**Proposition 4** Given a uniform distribution parity learner that with probability at least $1/4$ correctly identifies the concept $c_{r,b}$, algorithm 3 can correctly identify the distribution $P_{d,\ell,b,1/2}$ with probability at least $\frac{|\ell|}{4d}$.
Proof. Note that with probability $|\ell|/d$ we get that $i^* \in \ell$, in which case the inputs $x_1, \ldots, x_n$ provided to the learner $\Pi$ in Step 7 are uniformly distributed in $\{\pm 1\}^{d-1}$ and $y_j = b \cdot \prod_{i \in \ell \setminus \{i^*\}} x_j[i]$, i.e., the inputs to $\Pi$ are consistent with the concept $c_{\ell \setminus \{i^*\}, b}$.

On the uniform distribution, the generalization error of any parity function is $1/2$.

On $P_{d, \ell, b, 1/2}$ Algorithm 3 succeeds with probability $|\ell|/4d$ to identify $\ell, b$. Algorithm 4 evaluates the generalization error of the concept learned in algorithm 3 towards exhibiting a large total variance distance on $P_{d, L, B, 1/2}$ and $U^n$.

Algorithm 4: DistPU: Distinguisher for $P_{d, L, B, 1/2}^{n+m}$ and $U^{n+m}$

Let $M = ((R_1, \ldots, R_n), S, A)$ be the pan private algorithm described in Algorithm 3.

Input: A sample of $m + n$ examples $z = (z_1, z_2, \ldots, z_{n+m})$, where $m = \max\{512d/k, 64\sqrt{2d/k}/\varepsilon\}$ and each example is of the form $z_j = (z_j[1], z_j[2], \ldots, z_j[d]) \in \{\pm 1\}^d$.

1. Let $(\ell, b)$ be the outcome of executing $M$ on the first $n$ examples $z_1, \ldots, z_n$.
2. $c \leftarrow \text{Lap}(1/\varepsilon)$
3. for $i \in [m]$ do
   4. if $\prod_{j \in \ell} z_{i+n}[j] = b$ then $c \leftarrow c + 1$
5. end
6. $c^* \leftarrow c + \text{Lap}(1/\varepsilon)$
7. if $c^* \geq 3m/4$ then return 1 else return 0

Observe that if $z \sim P_{d, L, B, 1/2}^{n+m}$ then in every execution of Algorithm 4 there exists $\ell \subset [d]$ of cardinality at most $k$ and $b \in \{\pm 1\}$ such that $z \sim P_{d, \ell, b, 1/2}^{n+m}$.

Proposition 5 \Pr_{z \sim P_{d, L, B, 1/2}^{n+m}}[\text{DistPU}(z) = 1] \geq \frac{|\ell|}{8d}$.

Proof. For any $z \sim P_{d, L, B, 1/2}^{n+m}$, we always have $\prod_{i \in \ell} z_i = b$, so

$$\Pr[\text{DistPU}(z) = 1] \geq \Pr[\text{DistPU correctly identifies } (\ell, b)] \cdot \Pr[c^* \geq 3m/4]$$

$$\geq \frac{|\ell|}{4d} \cdot \Pr[\text{Lap}(1/\varepsilon) + \text{Lap}(1/\varepsilon) \geq -m/4]$$

$$\geq \frac{|\ell|}{4d} \cdot \frac{1}{2} \quad \text{(By symmetry of \text{Lap} around 0)}$$

$$= \frac{|\ell|}{8d}.$$
Proposition 6 $\Pr_{z \sim U^{n+m}}[\text{DistPU}(z) = 1] \leq \frac{k}{64d}$.

**Proof.** For all $(\ell, b)$, we have that $\Pr_{z \sim U}[\prod_{j \in \ell} z[j] = b] = 1/2$, so we have

$$\Pr_{z \sim U^{n+m}}[\text{DistPU}(z) = 1] = \Pr[\text{Bin}(m, 1/2) + \text{Lap}(1/\varepsilon) + \text{Lap}(1/\varepsilon) \geq m/4]$$

$$\leq \Pr[|\text{Bin}(m, 1/2) + \text{Lap}(1/\varepsilon) + \text{Lap}(1/\varepsilon) - m/2| \geq m/4]$$

$$\leq \frac{m/4 + 2/\varepsilon^2 + 2/\varepsilon^2}{m^2/16} \quad \text{(Chebyshev's inequality)}$$

$$= \frac{4/m + 64/\varepsilon^2 m^2}{m/128d + k/128d} = \frac{k}{64d}.$$ 

Proposition 7 $d_{TV}(\text{DistPU}(U^{n+m}), \text{DistPU}(P_{d,L,B,1/2}^{n+m})) \geq \frac{k}{64d}$.

**Proof.**

$$d_{TV}(\text{DistPU}(U^{n+m}), \text{DistPU}(P_{d,L,B,1/2}^{n+m}))$$

$$\geq \Pr_{z \sim P_{d,L,B,1/2}^{n+m}}[\text{DistPU}(z) = 1] - \Pr_{z \sim U^{n+m}}[\text{DistPU}(z) = 1]$$

$$= \sum_{\ell \in [d], |\ell| \leq k, b \in \{\pm 1\}} \Pr_{z \sim P_{d,L,B,1/2}^{n+m}}[\text{DistPU}(z) = 1] \cdot \Pr([L, B] = (\ell, b)) - \Pr_{z \sim U^{n+m}}[\text{DistPU}(z) = 1]$$

$$\geq \sum_{\ell \in [d], k/2 \leq |\ell| \leq k, b \in \{\pm 1\}} \Pr_{z \sim P_{d,L,B,1/2}^{n+m}}[\text{DistPU}(z) = 1] \cdot \Pr([L, B] = (\ell, b)) - \Pr_{z \sim U^{n+m}}[\text{DistPU}(z) = 1]$$

$$\geq \frac{k}{16d} \cdot \Pr_{z \sim U^{n+m}}[|L| \geq k/2] - \Pr_{z \sim U^{n+m}}[\text{DistPU}(z) = 1]$$

$$= \frac{k}{16d} \cdot \left(\frac{d}{k}\right) - \frac{d}{k} - \Pr_{z \sim U^{n+m}}[\text{DistPU}(z) = 1] \geq \frac{k}{32d} - \Pr_{z \sim U^{n+m}}[\text{DistPU}(z) = 1] \geq \frac{k}{64d}.$$ 

The last inequality follows from $\frac{d}{k} - \frac{d}{k} \geq 1/2$. 

1If $k = d$ then $\frac{d}{k} \geq 2\left(\frac{d}{k}\right)$. Otherwise $k < d$ we get for $0 \leq i \leq \lfloor k/2 \rfloor$ that the difference between $\lfloor k/2 \rfloor + 1 + i$ and $d/2$ is smaller than the difference between $\lfloor k/2 \rfloor - i$ and $d/2$ hence $\sum_{i=1}^{d} d_i < \lfloor k/2 \rfloor$, thus $\left(\frac{d}{k}\right) = \sum_{0 \leq i \leq \lfloor k/2 \rfloor} \left(\begin{array}{c}i \vspace{1em} \sum_{\lfloor k/2 \rfloor + 1 \leq i \leq k} \left(\begin{array}{c}i \vspace{1em} > 2\sum_{0 \leq i \leq \lfloor k/2 \rfloor} \left(\begin{array}{c}i \vspace{1em} = 2\left(\begin{array}{c}k/2 \rfloor\right)$. 

24
In particular, for all $k$ we get that $d_{TV}(\text{DistPU}(U^{n+m}), \text{DistPU}(P_{d,L,B,1/2}^{n+m})) \geq k/64d$ and for $k = d$ we get $d_{TV}(\text{DistPU}(U^{n+m}), \text{DistPU}(P_{d,L,B,1/2}^{n+m})) \geq 1/64$.

**Theorem 7** For any $(\varepsilon, \delta, 1/3)$-robust private distribution-free parity learning algorithm in the shuffle model, where $\varepsilon = O(1)$, the sample complexity is

$$n = \Omega \left( \frac{2^{d/2}}{\varepsilon} \right).$$

**Proof.** Let $k = d$, applying Theorem 6, $\text{DistPU}$ has sample complexity

$$n + m = \Omega \left( \frac{2^{d/2}}{\varepsilon} \right).$$

Since $k \geq 1$, $\varepsilon = O(1)$, $m = O(d/\varepsilon)$. By the of Algorithm $\text{DistPU}$ from a $(\varepsilon, \delta, 1/3)$-robust private parity learning algorithm, any $(\varepsilon, \delta, 1/3)$-robust private parity learning algorithm has sample complexity

$$n = \Omega \left( \frac{2^{d/2}}{\varepsilon} \right).$$

That completes our proof.

### 4.3 Tightness of the lowerbound

We now observe that Theorem 7 is tight as there exists a $1/3$-robust agnostic parity learner in the shuffle model with an almost matching sample complexity. For every possible hypothesis $(\ell, b)$ (there are $2^{d+1}$ hypotheses) the learner estimates the number of samples which are consistent with the hypothesis,

$$c_{\ell,b} = |\{i : b \cdot \prod_{j \in \ell} x_i[j] = y_i\}|.$$

One possibility for counting the number of consistent samples is to use the protocol by Balle et al. [2] which is an $(\varepsilon, \delta)$-differentially private one-round shuffle model.
protocol for estimating $\sum a_i$ where $a_i \in [0, 1]$. The outcome of this protocol is statistically close to $\sum a_i + DLap(e^\epsilon)$ and the statistical distance $\delta$ can be made arbitrarily small by increasing the number of messages sent by each agent. (In the discrete Laplace distribution $DLap(e^\epsilon)$, introduced in Definition 4, the probability of selecting $i \in \mathbb{Z}$ is proportional to $e^{-\epsilon|i|}$). The protocol uses the divisibility of Discrete Laplace random variables, generating Discrete Laplace noise $\nu$ as the sum of differences of Polya random variables:

$$\nu = \sum_{i=1}^{n} Polya(1/n, e^{-\epsilon}) - Polya(1/n, e^{-\epsilon}).$$

To make the protocol $\gamma$-robust, we slightly change the noise generation to guarantee $(\epsilon, \delta)$ differential privacy even in the case where only $n/3$ parties participate in the protocol. This can be done by changing the first parameter of the Polya random variables to $3/n$ resulting in $\nu = \sum_{i=1}^{n} Polya(3/n, e^{-\epsilon}) - Polya(3/n, e^{-\epsilon})$. Observe that $\nu$ is distributed as the sum of three independent $DLap(e^\epsilon)$ random variables. Using this protocol, it is possible for the analyzer to compute a noisy estimate of the number of samples consistent with each hypothesis,

$$\tilde{c}_{\ell,b} = c_{\ell,b} + \nu,$$

and then output

$$(\ell, b) = \arg\max_{\ell, b} (\tilde{c}_{\ell,b}).$$

The sample complexity of this learner is $O_{\alpha, \beta, \epsilon, \delta}(d^{2d/2})$. The resulting protocol is presented in Algorithm 5. We denote by $k$ the size of the concept class, i.e., $k = 2^{d+1}$, and $\text{Par}_{\ell,b}$ denotes the parity function with parameters $\ell$ and $b$.

**Proposition 8 (privacy)** For $\epsilon < 1$, LearnParity is $(\epsilon, \delta, \gamma)$-robust private, where $\delta = k \cdot \delta' + \delta^*$. 

26
Algorithm 5: LearnParity: an agnostic parity learning algorithm

Let $\varepsilon' = \varepsilon \frac{2}{\sqrt{2d \ln(1/\delta')}}$. Let ShuffleCount be an $(\varepsilon', \delta', \gamma)$-robust shuffle protocol that compute the sum of $\{0, 1\}$ bits.

Input: $n \geq \max \left\{ \frac{36((d+2)\ln 2-\ln \beta)}{\alpha^2}, \frac{48(\ln 3+(d+2)\ln 2)}{\alpha \varepsilon} \sqrt{2d \ln 1/\delta^*} \right\}$ labeled examples $(x_i, y_i)$ where $x_i \in \{-1, 1\}^d$ and $y_i \in \{-1, 1\}$.

1. For $\ell \subseteq [d], b \in \{\pm 1\}$ do
   2. Apply ShuffleCount to obtain a noisy count $\tilde{c}_{\ell, b}$ of samples for which $b \cdot \prod_{j \in \ell} x_i[j] = y_i$.
3. $(\hat{\ell}, \hat{b}) \leftarrow \arg\max_{\ell, b} \{\tilde{c}_{\ell, b} \mid \ell \subseteq [d], b \in \{\pm 1\}\}$
4. Return $(\hat{\ell}, \hat{b})$

Proof. LearnParity performes $k$ counting computations applying ShuffleCount and then selects the largest one. By the corollary of advanced composition, setting $\varepsilon' = \varepsilon \frac{2}{\sqrt{2d \ln 1/\delta^*}}$ can make LearnParity $(\varepsilon, \delta)$-differentially private. Since ShuffleCount is $\gamma$-robust, LearnParity is $\gamma$-robust.

To prove that LearnParity is an $(\alpha, \beta)$-agnostic parity learner, we show that (i) the true number of samples that agree with the parity function is close to the expected number of samples that agree with the parity function (Proposition 9); (ii) the noisy estimate produced by ShuffleCount is close to the true number of samples that agree with the parity function (Proposition 10).

Let $p_{\ell, b}$ represent the probability that one example agrees with the parity function $Par_{\ell, b}$.

Proposition 9

$$\Pr \left[ |p_{\ell, b} \cdot N - c_{\ell, b}| \leq \frac{\alpha N}{4} \right] \geq 1 - e^{-\frac{\alpha^2 N}{m}}$$
Proof. $c_{\ell,b}$ agrees with the distribution $\text{Bin}(N, p_{\ell,b})$, by chernoff bound,
\[
\Pr[c_{\ell,b} > (p_{\ell,b} + \alpha/4) \cdot N] = \Pr[c_{\ell,b} > (1 + \alpha/4p_{\ell,b}) \cdot p_{\ell,b}N] \leq e^{-\frac{\alpha^2 N}{24p_{\ell,b} + 4\alpha}} \leq e^{-\frac{\alpha^2 N}{36}}
\]
\[
\Pr[c_{\ell,b} < (p_{\ell,b} - \alpha/4) \cdot N] = \Pr[c_{\ell,b} < (1 - \alpha/4p_{\ell,b}) \cdot p_{\ell,b}N] \leq e^{-\frac{\alpha^2 N}{32p_{\ell,b}}} \leq e^{-\frac{\alpha^2 N}{36}}
\]

Proposition 10

\[
\Pr \left[ |\tilde{c}_{\ell,b} - c_{\ell,b}| \leq \frac{\alpha N}{4} \right] \geq 1 - 3 \cdot e^{-\frac{\alpha N \epsilon'}{12}}.
\]

Proof. The noise added in $\text{ShuffleCount}$ amounts to the sum of three $\text{DLap}(\epsilon')$ variables. The probability that a $\text{DLap}(\epsilon')$ variable exceeds $\alpha N/12$ is

\[
\Pr[|\text{DLap}(\epsilon')| > \alpha N/12] = 2 \cdot \frac{e^{\epsilon'} - 1}{e^{\epsilon'} + 1} \cdot ((e^{\epsilon'})^{-\frac{\alpha N \epsilon'}{12}} - 1 + (e^{\epsilon'})^{-\frac{\alpha N \epsilon'}{12}} - 2 + \ldots)
\]
\[
= 2 \cdot \frac{e^{\epsilon'} - 1}{e^{\epsilon'} + 1} \cdot \frac{e^{-\epsilon' (\frac{\alpha N}{12} + 1)} - 1}{1 - e^{-\epsilon'}}
\]
\[
= 2 \cdot \frac{e^{-\frac{\alpha N \epsilon'}{12}}}{e^{\epsilon'} + 1} \cdot \frac{1}{1 - e^{-\epsilon'}}
\]
\[
< e^{-\frac{\alpha N \epsilon'}{12}}.
\]

Hence, by union bound, the probability the sum of three $\text{DLap}(\epsilon')$ variables exceeds $\alpha N/4$ is at most $3 \cdot e^{-\frac{\alpha N \epsilon'}{12}}$.

Let $OPT$ be the lowest possible error of the hypothesis taken from all parity functions. If $\tilde{c}_{\ell,b} - Np_{\ell,b} < \frac{\alpha N}{2}$ for all $(\ell, b)$, the error of hypothesis outputted by the algorithm is less than $OPT + \alpha$.

Proposition 11 $\text{LearnParity}$ is $(\alpha, \beta)$-agnostic learning.

Proof. By union bound,
\[
\beta \leq k \cdot e^{-\frac{\alpha^2 n}{36}} + k \cdot 3 \cdot e^{-\frac{\alpha n \epsilon'}{12}} \leq \beta/2 + \beta/2 = \beta.
\]
The shuffle model of differential privacy has been suggested as an appealing alternative to the local model of differential privacy. Indeed, for tasks such as addition and histogram computing, the shuffle model allows for a lower level of noise than that required in the local model. The task of interest for this thesis is parity learning, which was shown to require an exponential sample complexity in the local model (at least when the round complexity of a protocol is limited to be polynomial). An interesting question was whether it is possible in the shuffle model to perform parity learning efficiently. The recent work of Cheu and Ullman suggested that this may not be the case – they gave an exponential lowerbound on the sample complexity of learning parity agnostically in the robust shuffle model. We extend this lowerbound to apply also to distribution free parity learning. We also present a concrete parity learner, whose sample complexity has a extra factor of $\text{poly}(d)$ than our result.

We leave the open question of the sample complexity of parity learning under the uniform distribution.
Appendix

Tail Bounds

Theorem 8 (Chebyshev’s inequality) For a random variable $X$ with expected value $\mathbb{E}(X) = \mu$ and variance $\text{Var}(X) = \delta^2$, and any $a > 0$,

$$\Pr[|X - \mu| \geq a] \leq \frac{\delta^2}{a^2}.$$ 

Theorem 9 (Chernoff bound) For a random variable $X$ with expected value $\mathbb{E}(X) = \mu$, then

1. $\Pr[X \geq (1 + \delta)\mu] \leq e^{-\frac{\delta^2 \mu}{2}}$ for all $\delta > 0$;
2. $\Pr[X \leq (1 - \delta)\mu] \leq e^{-\frac{\delta^2 \mu}{2}}$ for all $0 < \delta < 1$. 

Bibliography


